Foundations of Data Science[1]

Avrim Blum

John Hopcroft

Ravindran Kannan

Version May 14, 2015

These notes are a first draft of a book being written by Blum, Hopcroft and Kannan and in many places are incomplete. However, the notes are in good enough shape to prepare lectures for a modern theoretical course in computer science. Please do not put solutions to exercises online as it is important for students to work out solutions for themselves rather than copy them from the internet.

Thanks
JEH

Contents

Foundations of Data Science[†]

Avrim Blum, John Hopcroft and Ravindran Kannan

May 14, 2015

1 Introduction

Computer science as an academic discipline began in the 60's. Emphasis was on programming languages, compilers, operating systems, and the mathematical theory that supported these areas. Courses in theoretical computer science covered finite automata, regular expressions, context free languages, and computability. In the 70's, algorithms was added as an important component of theory. The emphasis was on making computers useful. Today, a fundamental change is taking place and the focus is more on applications. There are many reasons for this change. The merging of computing and communications has played an important role. The enhanced ability to observe, collect and store data in the natural sciences, in commerce, and in other fields calls for a change in our understanding of data and how to handle it in the modern setting. The emergence of the web and social networks, which are by far the largest such structures, presents both opportunities and challenges for theory.

While traditional areas of computer science are still important and highly skilled individuals are needed in these areas, the majority of researchers will be involved with using computers to understand and make usable massive data arising in applications, not just how to make computers useful on specific well-defined problems. With this in mind we have written this book to cover the theory likely to be useful in the next 40 years, just as automata theory, algorithms and related topics gave students an advantage in the last 40 years. One of the major changes is the switch from discrete mathematics to more of an emphasis on probability, statistics, and numerical methods.

Early drafts of the book have been used for both undergraduate and graduate courses. Background material needed for an undergraduate course has been put in the appendix. For this reason, the appendix has homework problems.

This book starts with the treatment of high dimensional geometry. Modern data in diverse fields such as Information Processing, Search, Machine Learning, etc., is often represented advantageously as vectors with a large number of components. This is so even in cases when the vector representation is not the natural first choice. Our intuition from two or three dimensional space can be surprisingly off the mark when it comes to high dimensional space. Chapter 2 works out the fundamentals needed to understand the differences. The emphasis of the chapter, as well as the book in general, is to get across the mathematical foundations rather than dwell on particular applications that are only briefly described.

The mathematical areas most relevant to dealing with high-dimensional data are matrix algebra and algorithms. We focus on singular value decomposition, a central tool in this area. Chapter 4 gives a from-first-principles description of this. Applications of singular value decomposition include principal component analysis, a widely used technique which we touch upon, as well as modern applications to statistical mixtures of probability densities, discrete optimization, etc., which are described in more detail.

Central to our understanding of large structures, like the web and social networks, is building models to capture essential properties of these structures. The simplest model is that of a random graph formulated by Erdös and Renyi, which we study in detail proving that certain global phenomena, like a giant connected component, arise in such structures with only local choices. We also describe other models of random graphs.

One of the surprises of computer science over the last two decades is that some domain-independent methods have been immensely successful in tackling problems from diverse areas. Machine learning is a striking example. We describe the foundations of machine learning, both learning from given training examples, as well as the theory of Vapnik-Chervonenkis dimension, which tells us how many training examples suffice for learning. Another important domain-independent technique is based on Markov chains. The underlying mathematical theory, as well as the connections to electrical networks, forms the core of our chapter on Markov chains.

The field of algorithms has traditionally assumed that the input data to a problem is presented in random access memory, which the algorithm can repeatedly access. This is not feasible for modern problems. The streaming model and other models have been formulated to better reflect this. In this setting, sampling plays a crucial role and, indeed, we have to sample on the fly. in Chapter 7 we study how to draw good samples efficiently and how to estimate statistical, as well as linear algebra quantities, with such samples.

One of the most important tools in the modern toolkit is clustering, dividing data into groups of similar objects. After describing some of the basic methods for clustering, such as the k-means algorithm, we focus on modern developments in understanding these, as well as newer algorithms. The chapter ends with a study of clustering criteria.

This book also covers graphical models and belief propagation, ranking and voting, sparse vectors, and compressed sensing. The appendix includes a wealth of background material.

A word about notation in the book. To help the student, we have adopted certain notations, and with a few exceptions, adhered to them. We use lower case letters for scaler variables and functions, bold face lower case for vectors, and upper case letters for matrices. Lower case near the beginning of the alphabet tend to be constants, in the middle of the alphabet, such as i, j, and k, are indices in summations, n and m for integer sizes, and x, y and z for variables. Where the literature traditionally uses a symbol for a quantity, we also used that symbol, even if it meant abandoning our convention. If we have a set of points in some vector space, and work with a subspace, we use n for the number of points, d for the dimension of the space, and k for the dimension of the subspace.

The term "almost surely" means with probability one. We use $\ln n$ for the natural logarithm and $\log n$ for the base two logarithm. If we want base ten, we will use $\log_{10}$. To simplify notation and to make it easier to read we use $E^2(1-x)$ for $\big(E(1-x)\big)^2$ and $E(1-x)^2$ for $E\left((1-x)^2\right)$.

2 High-Dimensional Space

2.1 Introduction

High dimensional data has become very important. However, high dimensional space is very different from the two and three dimensional spaces we are familiar with. For example, consider the unit ball: the set of all points x such that $|\mathbf{x}| \leq 1$. If one generates n points at random in the unit d-dimensional ball, for sufficiently large d, with high probability the distances between all pairs of points will be essentially the same. Also the volume of the unit ball in d-dimensions goes to zero as the dimension goes to infinity. In addition, the volume of the unit ball is concentrated near its surface and is also concentrated at its equator. These properties have important consequences which we will consider.

2.2 The Law of Large Numbers

If one generates random points in d-dimensional space using a Gaussian to generate coordinates, the distance between all pairs of points will be essentially the same. The reason is that the square of the distance between two points $\mathbf{x}$ and $\mathbf{y}$,

$$|\mathbf{x}-\mathbf{y}|^2 = \sum_{i=1}^{d}(x_i - y_i)^2,$$

is the sum of d independent random variables. If one averages n independent samples of a random variable x of bounded variance, the result will be close to the expected value of x. In our case, the samples are the squared distances between the two points in each coordinate i, and n equals d, so that the sum is the overall squared distance between the two points. Theorem 2.9 will give a tight concentration bound of this form. For now, we give a less tight but more general bound called the Law of Large Numbers. Specifically, the Law of Large Numbers states that

$$\text{Prob}\left(\left|\frac{x_1 + x_2 + \cdots + x_n}{n} - E(x)\right| > \epsilon\right) \leq \frac{\text{Var}(x)}{n\epsilon^2}. \tag{2.1}$$

The larger the variance of the random variable, the greater the probability that the error will exceed ϵ. The number of points n is in the denominator since the more values that are averaged, the smaller the probability that the difference will exceed ϵ. Similarly the larger ϵ is, the smaller the probability that the difference will exceed ϵ and hence ϵ is in the denominator. Notice that squaring ϵ makes the fraction a dimensionalless quantity.

We use two inequalities to prove the Law of Large Numbers. The first is Markov's inequality which bounds the probability that a nonnegative random variable exceeds a by the expected value of the variable divided by a.

Theorem 2.1 (Markov's inequality) *Let x be a nonnegative random variable. Then for $a > 0$,*

$$Prob(x \geq a) \leq \frac{E(x)}{a}.$$

Proof: For a continuous random variable x with probability density p,

$$E\left(x\right) = \int\limits_0^\infty xp(x)dx \geq \int\limits_a^\infty xp(x)dx \geq a\int\limits_a^\infty p(x)dx = a\text{Prob}(x \geq a).$$

Thus $\text{Prob}(x \geq a) \leq \frac{E(x)}{a}$.

The same proof works for discrete random variables with sums instead of integrals.

■

Corollary 2.2 $Prob\left(x \geq cE(x)\right) \leq \frac{1}{c}$

Markov's inequality bounds the tail of a distribution using only information about the mean. A tighter bound can be obtained by also using the variance of the random variable.

Theorem 2.3 (Chebyshev's inequality) *Let x be a random variable. Then for $b > 0$,*

$$Prob\Big(\,|x - E(x)| \geq b\Big) \leq \frac{Var(x)}{b^2}.$$

Proof: $\text{Prob}\big(|x - E(x)| \geq b\big) = \text{Prob}\big((x - E(x))^2 \geq b^2\big)$. Let $y = (x - E(x))^2$. Note that y is a nonnegative random variable, and $E(y) = \text{Var}(x)$, so Markov's inequality can be applied giving:

$$\text{Prob}(|x - E(x)| \geq b) = \text{Prob}\left(y^2 \geq b^2\right) \leq \frac{E(y)}{b^2} = \frac{\text{Var}(x)}{b^2}.$$

■

The Law of Large Numbers follows from Chebyshev's inequality together with facts about independent random variables. Recall that:

$$\begin{aligned}&E(x + y) = E(x) + E(y),\\&\text{Var}(cx) = c^2\text{Var}(x),\\&\text{Var}(x - c) = \text{Var}(x).\end{aligned}$$

and if x and y are independent, then $E(xy) = E(x)E(y)$. These facts imply that if x and y are independent then $\text{Var}(x + y) = \text{Var}(x) + \text{Var}(y)$, which we can see as follows:

$$\begin{aligned}\text{Var}(x + y) &= E\left((x + y)^2\right) - \big(E(x + y)\big)^2\\&= E(x^2 + 2xy + y^2) - \big(E(x)^2 + 2E(x)E(y) + E(y)^2\big)\\&= E(x^2) - E(x)^2 + E(y^2) - E(y)^2 = \text{Var}(x) + \text{Var}(y),\end{aligned}$$

where we used independence to replace $E(2xy)$ with $2E(x)E(y)$.

Theorem 2.4 (Law of large numbers) *Let $x_1, x_2, \ldots, x_n$ be n samples of a random variable x. Then*

$$Prob\left(\left|\frac{x_1 + x_2 + \cdots + x_n}{n} - E(x)\right| > \epsilon\right) \leq \frac{Var(x)}{n\epsilon^2}$$

Proof: By Chebychev's inequality

$$\begin{aligned}
\text{Prob}\left(\left|\frac{x_1 + x_2 + \cdots + x_n}{n} - E(x)\right| > \epsilon\right) &\leq \frac{\text{Var}\left(\frac{x_1+x_2+\cdots+x_n}{n}\right)}{\epsilon^2} \\
&\leq \frac{1}{n^2\epsilon^2}\text{Var}(x_1 + x_2 + \cdots + x_n) \\
&\leq \frac{1}{n^2\epsilon^2}\left(\text{Var}(x_1) + \text{Var}(x_2) + \cdots + \text{Var}(x_n)\right) \\
&\leq \frac{\text{Var}(x)}{n\epsilon^2}.
\end{aligned}$$

■

The Law of Large Numbers is quite general. Later we will look at tighter concentration bounds for spherical Gaussians and sums of 0-1 valued random variables.

As an application of the Law of Large Numbers, let **x** and **y** be d-dimensional random points whose coordinates are unit variance Gaussians. The square of the distance of **x** from the origin is approximately d. Since the square of the distance of random points are concentrated at distance d from the origin, there is no probability mass close to the origin even though the probability density has its maximum at the origin. This implies that a unit radius ball has zero volume, which explains why the integral of the probability density over the unit radius ball is zero. Similarly $|x - y|^2 \approx 2d$ and thus by Pythagoras's theorem, random d-dimensional **x** and **y** must be approximately orthogonal. This implies that if we scale these random points to be unit length and call **x** the North Pole, much of the surface area of the unit ball must lie near the equator. We will formalize these and related arguments shortly.

2.3 The Geometry of High Dimensions

An important property of high-dimensional objects is that most of their volume is near the surface. Consider any object A in R^d. Shrink each dimension by a factor γ to produce a new object γA (formally, $\gamma A = \{\gamma x : x \in A\}$), then Volume$(\gamma A) = \gamma^d$Volume$(A)$. To see that this is true, partition A into infinitesimal cubes of side-length dx, and notice that this fact holds true both for a cube and for a union of disjoint cubes. Set $\gamma = 1 - \epsilon$ for some small value ϵ. Using the fact that $1 - x \leq e^{-x}$, for any object A in R^d,

$$\frac{\text{Volume}((1-\epsilon)A)}{\text{Volume}(A)} \;=\; (1-\epsilon)^d \;\leq\; e^{-\epsilon d}.$$

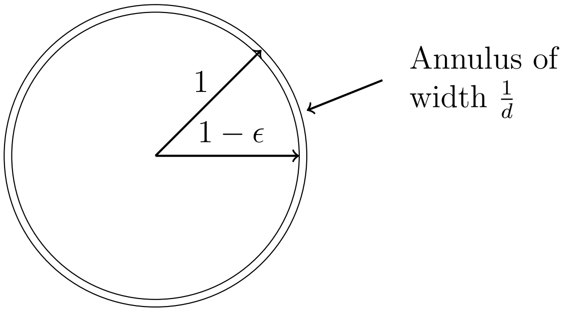


Figure 2.1: Most of the volume of the d-dimensional ball of radius r is contained in an annulus of width $O(r/d)$ near the boundary.

Fixing ϵ and letting $d \to \infty$, the above quantity rapidly approaches zero. This means that nearly all of the volume of A must be in the portion of A that does not belong to the inner region $(1-\epsilon)A$.

Let S denote the unit ball in d dimensions, that is, the set of points within distance one of the origin. An immediate implication of the above is that at least a $1-e^{-\epsilon d}$ fraction of the volume of the unit ball is concentrated in $S \setminus (1-\epsilon)S$, namely in a small annulus of width ϵ at the boundary. In particular, most of the volume of the d-dimensional unit ball is contained in an annulus of width $O(1/d)$ near the boundary. If the ball is of radius r, then the annulus width is $O\left(\frac{r}{d}\right)$.

2.4 Properties of the Unit Ball

We now focus more specifically on properties of the unit ball in d-dimensional space. We just saw that most of its volume is concentrated in a small annulus of width $O(1/d)$ near the boundary. Next we will show that in the limit as d goes to infinity, the volume of the ball goes to zero. The Law of Large Numbers suggested this and the result can be proved in several ways. Here we use integration.

2.4.1 Volume of the Unit Ball

To calculate the volume $V(d)$ of the unit ball in R^d, one can integrate in either Cartesian or polar coordinates. In Cartesian coordinates the volume is given by

$$V(d) = \int\limits_{x_1=-1}^{x_1=1} \int\limits_{x_2=-\sqrt{1-x_1^2}}^{x_2=\sqrt{1-x_1^2}} \cdots \int\limits_{x_d=-\sqrt{1-x_1^2-\cdots-x_{d-1}^2}}^{x_d=\sqrt{1-x_1^2-\cdots-x_{d-1}^2}} dx_d \cdots dx_2 dx_1.$$

Since the limits of the integrals are complicated, it is easier to integrate using polar coordinates. In polar coordinates, $V(d)$ is given by

$$V\left(d\right)=\int\limits_{S^d}\int\limits_{r=0}^{1}r^{d-1}drd\Omega.$$

Since the variables Ω and r do not interact,

$$V\left(d\right)=\int\limits_{S^d}d\Omega\int\limits_{r=0}^{1}r^{d-1}dr=\frac{1}{d}\int\limits_{S^d}d\Omega=\frac{A(d)}{d}$$

where $A(d)$ is the surface area of the d-dimensional unit ball. For instance, for $d=3$ the surface area is 4π and the volume is $\frac{4}{3}\pi$. The question remains, how to determine the surface area $A\left(d\right)=\int\limits_{S^d}d\Omega$ for general d.

Consider a different integral

$$I\left(d\right)=\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty}\cdots\int\limits_{-\infty}^{\infty}e^{-\left(x_1^2+x_2^2+\cdots x_d^2\right)}dx_d\cdots dx_2dx_1.$$

Including the exponential allows integration to infinity rather than stopping at the surface of the sphere. Thus, $I(d)$ can be computed by integrating in both Cartesian and polar coordinates. Integrating in polar coordinates will relate $I(d)$ to the surface area $A(d)$. Equating the two results for $I(d)$ allows one to solve for $A(d)$.

First, calculate $I(d)$ by integration in Cartesian coordinates.

$$I\left(d\right)=\left[\int\limits_{-\infty}^{\infty}e^{-x^2}dx\right]^d=\left(\sqrt{\pi}\right)^d=\pi^{\frac{d}{2}}.$$

Here, we have used the fact that $\int_{-\infty}^{\infty}e^{-x^2}\ dx=\sqrt{\pi}$. For a proof of this, see Section **??** of the appendix. Next, calculate $I(d)$ by integrating in polar coordinates. The volume of the differential element is $r^{d-1}d\Omega dr$. Thus,

$$I\left(d\right)=\int\limits_{S^d}d\Omega\int\limits_{0}^{\infty}e^{-r^2}r^{d-1}dr.$$

The integral $\int\limits_{S^d}d\Omega$ is the integral over the entire solid angle and gives the surface area, $A(d)$, of a unit sphere. Thus, $I\left(d\right)=A\left(d\right)\int\limits_{0}^{\infty}e^{-r^2}r^{d-1}dr$. Evaluating the remaining integral gives

$$\int\limits_0^\infty e^{-r^2}r^{d-1}dr = \frac{1}{2}\int\limits_0^\infty e^{-t}t^{\frac{d}{2}}{}^{-\,1}dt = \frac{1}{2}\Gamma\left(\frac{d}{2}\right)$$

and hence, $I(d) = A(d)\frac{1}{2}\Gamma\left(\frac{d}{2}\right)$ where the gamma function $\Gamma\left(x\right)$ is a generalization of the factorial function for noninteger values of x. $\Gamma\left(x\right) = \left(x-1\right)\Gamma\left(x-1\right)$, $\Gamma\left(1\right) = \Gamma\left(2\right) = 1$, and $\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$. For integer x, $\Gamma\left(x\right) = (x-1)!$.

Combining $I\left(d\right) = \pi^{\frac{d}{2}}$ with $I\left(d\right) = A\left(d\right)\frac{1}{2}\Gamma\left(\frac{d}{2}\right)$ yields

$$A\left(d\right) = \frac{\pi^{\frac{d}{2}}}{\frac{1}{2}\Gamma\left(\frac{d}{2}\right)}$$

establishing the following lemma.

Lemma 2.5 *The surface area $A(d)$ and the volume $V(d)$ of a unit-radius sphere in d dimensions are given by*

$$A\left(d\right) = \frac{2\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \qquad \textit{and} \qquad V\left(d\right) = \frac{2}{d}\frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)}.$$

To check the formula for the volume of a unit sphere, note that $V\left(2\right) = \pi$ and $V\left(3\right) = \frac{2}{3}\frac{\pi^{\frac{3}{2}}}{\Gamma\left(\frac{3}{2}\right)} = \frac{4}{3}\pi$, which are the correct volumes for the unit spheres in two and three dimensions. To check the formula for the surface area of a unit sphere, note that $A(2) = 2\pi$ and $A(3) = \frac{2\pi^{\frac{3}{2}}}{\frac{1}{2}\sqrt{\pi}} = 4\pi$, which are the correct surface areas for the unit sphere in two and three dimensions. Note that $\pi^{\frac{d}{2}}$ is an exponential in $\frac{d}{2}$ and $\Gamma\left(\frac{d}{2}\right)$ grows as the factorial of $\frac{d}{2}$. This implies that $\lim\limits_{d\to\infty} V(d) = 0$, as claimed.

2.4.2 Most of the volume is near the equator

An interesting fact about the unit ball in high dimensions is that most of its volume is concentrated near its equator (no matter what direction one uses to define the "equator"). Arbitrarily letting x_1 denote "north", most of the volume of the unit ball has $|x_1| = O(1/\sqrt{d})$. Using this fact, we will show that two random points in the unit ball are with high probability nearly orthogonal, and also give an alternative proof from the one in Section 2.4.1 that the volume of the unit ball goes to zero as $d \to \infty$.

Theorem 2.6 *For $c \geq 1$ and $d \geq 3$, at least a $1 - \frac{2}{c}e^{-c^2/2}$ fraction of the volume of the d-dimensional unit ball has $|x_1| \leq \frac{c}{\sqrt{d-1}}$.*

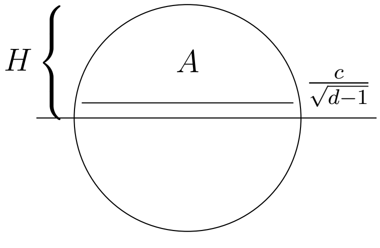


Figure 2.2: Most of the volume of the upper hemisphere of the d-dimensional ball is below the plane $x_1 = \frac{c}{\sqrt{d-1}}$

Proof: By symmetry we just need to prove that at most an $\frac{e}{c}e^{-c^2/2}$ fraction of the half of the ball with $x_1 \geq 0$ has $x_1 \geq \frac{c}{\sqrt{d-1}}$. Let A denote the portion of the ball with $x_1 \geq \frac{c}{\sqrt{d-1}}$.

To calculate the volume of A, integrate an incremental volume that is a disk of width dx_1 and whose face is a sphere of dimension $d-1$ and radius $\sqrt{1-x_1^2}$. The surface area of the disk is $(1-x_1^2)^{\frac{d-1}{2}}V(d-1)$ and the volume above the slice is

$$\text{volume}(A) = \int_{\frac{c}{\sqrt{d-1}}}^{1} (1-x_1^2)^{\frac{d-1}{2}}V(d-1)dx_1$$

To get an upper bound on the above integral, use $1-x \leq e^{-x}$ and integrate to infinity. To integrate, insert $\frac{x_1\sqrt{d-1}}{c}$, which is greater than one in the range of integration, into the integral. Then

$$\text{volume}(A) \leq \int_{\frac{c}{\sqrt{d-1}}}^{\infty} \frac{x_1\sqrt{d-1}}{c} e^{-\frac{d-1}{2}x_1^2}V(d-1)dx_1 = V(d-1)\frac{\sqrt{d-1}}{c}\int_{\frac{c}{\sqrt{d-1}}}^{\infty} x_1 e^{-\frac{d-1}{2}x_1^2}dx_1$$

Now

$$\int_{\frac{c}{\sqrt{d-1}}}^{\infty} x_1 e^{-\frac{d-1}{2}x_1^2}dx_1 = -\frac{1}{d-1}e^{-\frac{d-1}{2}x_1^2}\Big|_{\frac{c}{\sqrt{(d-1)}}}^{\infty} = \frac{1}{d-1}e^{-\frac{c^2}{2}}$$

Thus, an upper bound on volume(A) is $\frac{V(d-1)}{c\sqrt{d-1}}e^{-\frac{c^2}{2}}$.

The volume of the hemisphere below the plane $x_1 = \frac{1}{\sqrt{d-1}}$ is a lower bound on the entire volume of the upper hemisphere and this volume is at least that of a cylinder of height $\frac{1}{\sqrt{d-1}}$ and radius $\sqrt{1-\frac{1}{d-1}}$. The volume of the cylinder is $V(d-1)(1-\frac{1}{d-1})^{\frac{d-1}{2}}\frac{1}{\sqrt{d-1}}$. Using the fact that $(1-x)^a \geq 1-ax$ for $a \geq 1$, the volume of the cylinder is at least $\frac{V(d-1)}{2\sqrt{d-1}}$ for $d \geq 3$.

Thus,

$$\text{ratio} \leq \frac{\text{upper bound above plane}}{\text{lower bound total hemisphere}} = \frac{\frac{V(d-1)}{c\sqrt{d-1}}e^{-\frac{c^2}{2}}}{\frac{V(d-1)}{2\sqrt{d-1}}} = \frac{2}{c}e^{-\frac{c^2}{2}}$$

■

Near orthogonality. One immediate implication of the above analysis is that if we draw two points at random from the unit ball, with high probability their vectors will be nearly orthogonal to each other. Specifically, from our previous analysis in Section 2.3, with high probability both will have length $1 - O(1/d)$. From our analysis above, if we define the vector in the direction of the first point as "north", with high probability the second will have a projection of only $\pm O(1/\sqrt{d})$ in this direction. This implies that with high probability, the angle between the two vectors will be $\pi/2 \pm O(1/\sqrt{d})$. In particular, we have the theorem:

Theorem 2.7 *Consider drawing n points $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ at random from the unit ball. With probability $1 - O(1/n)$*

1. $|\mathbf{x_i}| \geq 1 - \frac{2\ln n}{d}$ *for all i, and*
2. $|\mathbf{x_i} \cdot \mathbf{x_j}| \leq \frac{\sqrt{6\ln n}}{\sqrt{d-1}}$ *for all $i \neq j$.*

Proof: For the first part, for any fixed i, by the analysis of Section 2.3 Prob$(|\mathbf{x_i}| < 1 - \frac{2\ln n}{d}) \leq e^{-(\frac{2\ln n}{d})d} = 1/n^2$. So, by the union bound, the probability there exists i such that $|\mathbf{x_i}| < 1 - \frac{2\ln n}{d}$ is at most $1/n$. For the second part, $\binom{n}{2}$ pairs i and j, and for each such pair, if we define $\mathbf{x_i}$ as "north", the probability that the projection of $\mathbf{x_j}$ onto that direction is more than $\frac{\sqrt{6\ln n}}{\sqrt{d-1}}$ (a necessary condition for the dot-product to be large) is at most $O(e^{-\frac{6\ln n}{2}}) = O(n^{-3})$ by Theorem 2.6. Thus, this condition is violated with probability at most $O\left(\binom{n}{2}n^{-3}\right) = O(1/n)$ as well. ■

Alternative proof that volume goes to zero. Another immediate implication of Theorem 2.6 is that as $d \to \infty$, the volume of the ball approaches zero. Specifically, setting $c = 2\sqrt{\ln d}$ in Theorem 2.6 at most an $O(1/d^2)$ fraction of the volume of the ball has $|x_1| \geq \frac{c}{\sqrt{d-1}}$. Since this is true for each of the d dimensions, at least a $1 - O(\frac{1}{d}) \geq \frac{1}{2}$ fraction of the volume of the ball lies in a cube of side-length $2\frac{c}{\sqrt{d-1}}$. This cube has volume $(\frac{16\ln d}{d-1})^{d/2}$, and this quantity goes to zero as $d \to \infty$. Since the ball has volume at most twice that of this cube, its volume goes to zero as well.

Discussion. One might wonder how can it be that nearly all the points in the unit ball are very close to the surface and yet at the same time nearly all points are in a box of side-length $O\left(\frac{\ln d}{d-1}\right)$? The answer is to remember that for points on the surface of the ball, $x_1^2 + x_2^2 + \ldots + x_d^2 = 1$, so for each coordinate i, a typical value will be $\pm O\left(\frac{1}{\sqrt{d}}\right)$. In fact, it is often helpful to think of picking a random point on the sphere as very similar to picking a random point of the form $\left(\pm\frac{1}{\sqrt{d}}, \pm\frac{1}{\sqrt{d}}, \pm\frac{1}{\sqrt{d}}, \ldots \pm\frac{1}{\sqrt{d}}\right)$.

2.5 Generating Points Uniformly at Random from a Ball

How can one select points uniformly at random from the unit ball? First, let's consider generating points uniformly at random on the *surface* of the unit ball. For the

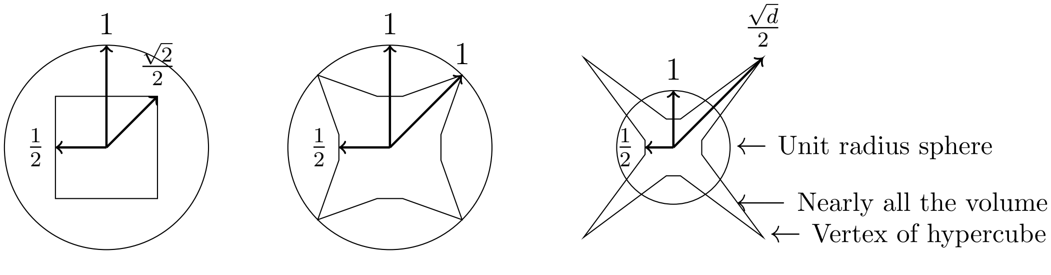


Figure 2.3: Illustration of the relationship between the sphere and the cube in 2, 4, and d-dimensions.

2-dimensional version of generating points on the circumference of a unit-radius circle, independently generate each coordinate uniformly at random from the interval $[-1, 1]$. This produces points distributed over a square that is large enough to completely contain the unit circle. Project each point onto the unit circle. The distribution is not uniform since more points fall on a line from the origin to a vertex of the square than fall on a line from the origin to the midpoint of an edge of the square due to the difference in length. To solve this problem, discard all points outside the unit circle and project the remaining points onto the circle.

In higher dimensions, this method does not work since the fraction of points that fall inside the ball drops to zero and all of the points would be thrown away. The solution is to generate a point each of whose coordinates is an independent Gaussian variable. Generate $x_1, x_2, \ldots, x_d$, using a zero mean, unit variance Gaussian, namely, $\frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$ on the real line.[1] Thus the probability density of $\mathbf{x}$ is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{x_1^2 + x_2^2 + \cdots + x_d^2}{2}}$$

and is spherically symmetric. Normalizing the vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ to a unit vector, namely $\frac{\mathbf{x}}{|\mathbf{x}|}$, gives a distribution that is uniform over the surface of the sphere. Note that once the vector is normalized, its coordinates are no longer statistically independent.

To generate a point $\mathbf{y}$ uniformly over the ball (surface and interior), scale the point $\frac{\mathbf{x}}{|\mathbf{x}|}$ generated on the surface by a scalar $\rho \in [0, 1]$. What is the distribution of ρ as a function of r? It is certainly not uniform, even in 2 dimensions. Indeed, the density of ρ at r is proportional to r for $d = 2$. For $d = 3$, it is proportional to r^2. By similar

[1] One might naturally ask: "how do you generate a random number from a 1-dimensional Gaussian?" To generate a number from any distribution given its cumulative distribution function P, first select a uniform random number $u \in [0, 1]$ and then choose $x = P^{-1}(u)$. This generates a number between x and $x + \delta$ with probability $P(x + \delta) - P(x) = \int_x^{x+\delta} p(z)dz$ as desired. For the 2-dimensional Gaussian, one can generate a point in polar coordinates by choosing angle θ uniform in $[0, 2\pi]$ and radius $r = \sqrt{-2\ln(u)}$ where u is uniform random in $[0, 1]$. This is called the Box-Muller transform.

reasoning, the density of ρ at distance r is proportional to r^{d-1} in d dimensions. Solving $\int_{r=0}^{r=1} cr^{d-1}dr = 1$ (the integral of density must equal 1) we should set $c = d$. Another way to see this formally is that the volume of the radius r ball in d dimensions is r^dV_d, where V_d is the volume of the unit ball. The density at radius r is exactly $\frac{d}{dr}(r^dV_d) = dr^{d-1}V_d$. So, pick $\rho(r)$ with density equal to dr^{d-1} for r over $[0, 1]$.

We have succeeded in generating a point

$$\mathbf{y} = \rho\frac{\mathbf{x}}{|\mathbf{x}|}$$

uniformly at random from the unit ball S by using the convenient spherical Gaussian distribution. In the next sections, we will analyze the spherical Gaussian in more detail.

2.6 Gaussians in High Dimension

A 1-dimensional Gaussian has its mass close to the origin. However, as the dimension is increased something different happens. The d-dimensional spherical Gaussian with zero mean and variance σ^2 in each coordinate has density function

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\,\sigma^d}\exp\left(-\tfrac{|\mathbf{x}|^2}{2\sigma^2}\right).$$

The value of the density is maximum at the origin, but there is very little volume there. When $\sigma = 1$, integrating the probability density over a unit ball centered at the origin yields nearly zero mass since the volume of such a ball is negligible. In fact, one needs to increase the radius of the ball to nearly $\sqrt{d}$ before there is a significant nonzero volume and hence significant probability mass. If one increases the radius much beyond $\sqrt{d}$, the integral barely increases even though the volume increases since the probability density is dropping off at a much higher rate. The following theorem states this formally that nearly all the probability is concentrated in a thin annulus of width $O(1)$ at radius $\sqrt{d}$.

Theorem 2.8 (Gaussian Annulus Theorem) *For a d-dimensional spherical Gaussian with unit variance in each direction, for any $\beta \leq \sqrt{d}$, all but at most $3e^{-c\beta^2}$ of the probability mass lies within the annulus $\sqrt{d} - \beta \leq |\mathbf{x}| \leq \sqrt{d} + \beta$, where c is a fixed positive constant.*

For a high-level intuition, note that $E(|\mathbf{x}|^2) = \sum_{i=1}^{d} E(x_i^2) = dE(x_1^2) = d$, so the mean squared distance of a point from the center is d. The Gaussian Annulus Theorem says that the points are tightly concentrated. We call the square root of the mean squared distance, namely $\sqrt{d}$, the radius of the Gaussian.

To prove the Gaussian Annulus Theorem we make use of a tail inequality for sums of independent random variables of bounded moments.

Theorem 2.9 *Let $x = x_1 + x_2 + \cdots + x_n$, where $x_1, x_2, \ldots, x_n$ are mutually independent random variables with zero mean and variance at most σ^2. Assume that $|E(x_i^s)| \leq \sigma^2 s!$ for $s = 3, 4, \ldots, \lfloor(a^2/4n\sigma^2)\rfloor$. Then for $0 \leq a \leq \sqrt{2}n\sigma^2$ or $\lfloor(a^2/4n\sigma^2)\rfloor \leq n\sigma^2$,*

$$Prob\,(|x| \geq a) \leq 3e^{-a^2/(12n\sigma^2)}.$$

Theorem 2.9 is proved in the appendix. For a brief intuition, consider applying Markov's inequality to the random variable x^r where r is a large even number. Since r is even, x^r is non-negative, and thus $\text{Prob}(|x| \geq a) = \text{Prob}(x^r \geq a^r) \leq E(x^r)/a^r$. So, if $E(x^r)$ is not too large, we will get a good bound. To compute $E(x^r)$, write $E(x)$ as $E((x_1 + \ldots + x_n)^r)$ and distribute the polynomial into its terms. Use the fact that by independence $E(x_i^{r_i} x_j^{r_j}) = E(x_i^{r_i})E(x_j^{r_j})$ to get a collection of simpler expectations that can be bounded using our assumption that $|E(x_i^s)| \leq \sigma^2 s!$. For the full proof, see the appendix. We now prove the Gaussian Annulus Theorem using Theorem 2.9.

Proof (Gaussian Annulus Theorem): Let $\mathbf{y} = (y_1, y_2, \ldots, y_d)$ be a point selected from a unit variance Gaussian centered at the origin, and let $r = |\mathbf{y}|$. If $|r - \sqrt{d}| \geq \beta$ then multiplying both sides by $r + \sqrt{d}$ gives $|r^2 - d| \geq \beta(r + \sqrt{d}) \geq \beta\sqrt{d}$. So, it suffices to bound the probability that $|r^2 - d| \geq \beta\sqrt{d}$.

Rewrite $r^2 - d = (y_1^2 + \ldots + y_d^2) - d = (y_1^2 - 1) + \ldots + (y_d^2 - 1)$ and perform a change of variables: $x_i = y_i^2 - 1$. We want to bound the probability that $|x_1 + \ldots + x_d| \geq \beta\sqrt{d}$, and notice that $E(x_i) = E(y_i^2) - 1 = 0$. To apply Theorem 2.9, we need then to bound the s^{th} moments of x_i.

For $|y_i| \leq 1$, $|x_i|^s \leq 1$ and for $|y_i| \geq 1$, $|x_i|^s \leq |y_i|^{2s}$. Thus

$$\begin{aligned} |E(x_i^s)| &= E(|x_i|^s) \leq E(1 + y_i^{2s}) = 1 + E(y_i^{2s}) \\ &= 1 + \sqrt{\frac{2}{\pi}} \int_0^\infty y^{2s} e^{-y^2/2} dy \end{aligned}$$

Using the substitution $2z = y^2$,

$$\begin{aligned} |E(x_i^s)| &= 1 + \frac{1}{\sqrt{\pi}} \int_0^\infty 2^s z^{s-(1/2)} e^{-z} dz \\ &\leq 2^s s!. \end{aligned}$$

The last inequality is from the Gamma integral.

Since $E(x_i) = 0$, $\text{Var}(x_i) = E(x_i^2) \leq 2^2 2 = 8$. Unfortunately, we do not have $|E(x_i^s)| \leq 8s!$ as required in Theorem 2.9. To fix this problem, perform one more change of variables, using $w_i = x_i/2$. Then, $\text{Var}(w_i) \leq 2$ and $|E(w_i^s)| \leq 2s!$, and our goal is now to bound the probability that $|w_1 + \ldots + w_d| \geq \frac{\beta\sqrt{d}}{2}$. Applying Theorem 2.9 where $\sigma^2 = 2$ and $n = d$, this occurs with probability less than or equal to $3e^{-\frac{\beta^2}{96}}$. ■

In the next sections we will see several uses of the Gaussian Annulus Theorem.

2.7 Random Projection and Johnson-Lindenstrauss Lemma

One of the most frequently used subroutines for high dimensional data is the Nearest Neighbor Search (NNS) problem. The nearest neighbor search has a database of n points in $\mathbf{R}^d$ where n and d are usually large. The database can be preprocessed and stored in an efficient data structure. Thereafter, we are presented "query" points in $\mathbf{R}^d$ and are to find the nearest or approximately nearest database point to the query point. Since the number of queries is often large, query time (time to answer a single query) should be very small (ideally a small function of $\log n$ and $\log d$), whereas preprocessing time could be larger (a polynomial function of n and d). For this and other problems, **dimension reduction**, where one projects the database points to a k dimensional space with $k \ll d$ (usually dependent on $\log d$) can be very useful so long as the relative distances between points are approximately preserved. We will see using the Gaussian Annulus theorem that such a projection indeed exists and is simple.

The projection $f : \mathbf{R}^d \to \mathbf{R}^k$ that we will examine (in fact, many related projections are known to work as well) is the following. Pick k vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, independently from the Gaussian distribution $\frac{1}{(2\pi)^{d/2}} \exp(-|\mathbf{x}|^2/2)$. For any vector $\mathbf{v}$, define the projection $f(\mathbf{v})$ by:

$$f(\mathbf{v}) = (\mathbf{u_1} \cdot \mathbf{v}, \mathbf{u_2} \cdot \mathbf{v}, \ldots, \mathbf{u_k} \cdot \mathbf{v}).$$

The projection $f(\mathbf{v})$ is the vector of dot products of $\mathbf{v}$ with the $\mathbf{u_i}$. We will show that with high probability, $|f(\mathbf{v})| \approx \sqrt{k}|\mathbf{v}|$. For any two vectors $\mathbf{v_1}$ and $\mathbf{v_2}$, $f(\mathbf{v_1} - \mathbf{v_2}) = f(\mathbf{v_1}) - f(\mathbf{v_2})$. Thus, to find the distance $|\mathbf{v_1} - \mathbf{v_2}|$ between two vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ in $\mathbf{R}^d$, it suffices to compute $|f(\mathbf{v_1}) - f(\mathbf{v_2})| = |f(\mathbf{v_1} - \mathbf{v_2})|$ in the k dimensional space since the factor of $\sqrt{k}$ is known and one can divide by it. The reason distances increase when we project to a lower dimensional space is that the basis vectors $\mathbf{u_i}$ are not unit length. Also notice that the basis vectors are not orthogonal. If we had an orthogonal basis, we would have lost statistical independence.

Theorem 2.10 (The Random Projection Theorem) *Let* $\mathbf{v}$ *be a fixed vector in* $\mathbf{R}^d$ *and let* f *be defined as above. Then, for* $\varepsilon \in (0,1)$,

$$\mathit{Prob}\left(\Big| |f(\mathbf{v})| \; - \; \sqrt{k}|\mathbf{v}| \Big| \geq \varepsilon\sqrt{k}|\mathbf{v}| \right) \leq 3e^{-ck\varepsilon^2},$$

where the probability is taken over the random draws of vectors $\mathbf{u_i}$ *used to construct* f.

Proof: By scaling both sides by $|\mathbf{v}|$, we may assume that $|\mathbf{v}| = 1$. The sum of independent normally distributed real variables is also normally distributed where the mean and variance are the sums of the individual means and variances. Since $\mathbf{u_i} \cdot \mathbf{v} = \sum_{j=1}^d u_{ij} v_j$, the random variable $\mathbf{u_i} \cdot \mathbf{v}$ has Gaussian density with zero mean and variance equal to $\sum_{j=1}^d v_j^2 = |\mathbf{v}|^2 = 1$. (Since u_{ij} has variance one and the v_j is a constant, the variance of $u_{ij} v_j$ is v_j^2.) Since $\mathbf{u_1} \cdot \mathbf{v}, \mathbf{u_2} \cdot \mathbf{v}, \ldots, \mathbf{u_k} \cdot \mathbf{v}$ are independent, the theorem follows from the Gaussian Annulus Theorem (Theorem 2.8) with $k = d$. ■

The random projection theorem establishes that the probability of the length of the projection of a single vector differing significantly from its expected value is exponentially small in k, the dimension of the target subspace. By a union bound, the probability that any of $O(n^2)$ pairwise differences $|\mathbf{v_i} - \mathbf{v_j}|$ among n vectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$ differs significantly from their expected values is small, provided $k \geq \frac{3}{c\varepsilon^2} \ln n$. Thus, the projection to a random subspace preserves all relative pairwise distances between points in a set of n points with high probability. This is the content of the Johnson-Lindenstrauss Lemma.

Theorem 2.11 (Johnson-Lindenstrauss Lemma) *For any $0 < \varepsilon < 1$ and any integer n, let $k \geq \frac{3}{c\varepsilon^2} \ln n$ for c as in Theorem 2.8. For any set P of n points in R^d, the random projection $f : R^d \to R^k$ defined above has the property that for all $\mathbf{v_i}$ and $\mathbf{v_j}$ in P, with probability at least $1 - 1.5/n$,*

$$(1-\varepsilon)\sqrt{k}\,|\mathbf{v_i} - \mathbf{v_j}| \leq |f(\mathbf{v_i}) - f(\mathbf{v_j})| \leq (1+\varepsilon)\sqrt{k}\,|\mathbf{v_i} - \mathbf{v_j}|\,.$$

Proof: Applying the random projection theorem (Theorem 2.10), for any fixed $\mathbf{v_i}$ and $\mathbf{v_j}$, the probability that $|f(\mathbf{v_i} - \mathbf{v_j})|$ is outside the range

$$\Big[(1-\varepsilon)\sqrt{k}|\mathbf{v_i} - \mathbf{v_j}|, (1+\varepsilon)\sqrt{k}|\mathbf{v_i} - \mathbf{v_j}|\Big]$$

is at most $3e^{-ck\varepsilon^2} \leq 3/n^3$ for $k \geq \frac{3 \ln n}{c\varepsilon^2}$. Since there are $\binom{n}{2} < n^2/2$ pairs, by the union bound, the probability that any pair has a large distortion is less than $\frac{3}{2n}$. ■

Remark: It is important to note that the conclusion of Theorem 2.11 asserts for all $\mathbf{v_i}$ and $\mathbf{v_j}$ in P, not just for most of them. The weaker assertion for most $\mathbf{v_i}$ and $\mathbf{v_j}$ is typically less useful, since our algorithm for a problem such as nearest-neighbor search might return one of the bad points. A remarkable aspect of the theorem is that the number of dimensions in the projection is only dependent logarithmically on n. Since k is often much less than d, this is called a dimension reduction technique.

For the nearest neighbor problem, if the database has n_1 points and n_2 queries are expected during the lifetime, take $n = n_1 + n_2$ and project the database to a random k-dimensional space, for k as in Theorem 2.11. On receiving a query, project the query to the same subspace and compute nearby database points. The Johnson Lindenstrauss theorem says that with high probability this will yield the right answer whatever the query. Note that the exponentially small in k probability was useful here in making k only dependent on $\ln n$, rather than n.

2.8 Separating Gaussians

Mixtures of Gaussians are often used to model heterogeneous data coming from multiple sources. For example, suppose we are recording the heights of individuals age 20-30 in a city. We know that on average, men tend to be taller than women, so a natural model would be a Gaussian mixture model $p(\mathbf{x}) = w_1 p_1(\mathbf{x}) + w_2 p_2(\mathbf{x})$, where $p_1(\mathbf{x})$ is a Gaussian density representing the typical heights of women, $p_2(\mathbf{x})$ is a Gaussian density representing the typical heights of men, and w_1 and w_2 are the *mixture weights* representing the proportion of women and men in the city. The *parameter estimation problem* for a mixture model is the problem: given access to samples from the overall density p (e.g., heights of people in the city, but without being told whether the person with that height is male or female), reconstruct the parameters for the distribution (e.g., good approximations to the means and variances of p_1 and p_2, as well as the mixture weights).

Now of course there are taller women and shorter men, so even if one solved the parameter estimation problem for heights perfectly, given a data point (a height) one couldn't necessarily tell which population it came from (male or female). In this section, we will look at a problem that is in some ways easier and some ways harder than this problem of heights. It will be harder in that we will be interested in a mixture of two Gaussians in high-dimensions (as opposed to the $d = 1$ case of heights). But it will be easier in that we will assume the means are quite well-separated compared to the variances. Specifically, our focus will be on a mixture of two spherical unit-variance Gaussians whose means are separated by a distance $\Omega(d^{1/4})$. We will show that at this level of separation, we can with high probability in fact uniquely determine which Gaussian each data point came from. The algorithm to do so will actually be quite simple. Calculate the distance between all pairs of points. Points whose distance apart is smaller are from the same Gaussian, whereas points whose distance is larger are from different Gaussians. Later, we will see that with more sophisticated algorithms, even a separation of $\Omega(1)$ suffices.

Consider two spherical unit-variance Gaussians. From Theorem 2.8, most of the probability mass of each Gaussian lies on an annulus of width $O(1)$ at radius $\sqrt{d}$. Also $e^{-|\mathbf{x}|^2/2} = \prod_i e^{-x_i^2/2}$ and almost all of the mass is within the slab $\{ \mathbf{x} \mid -c \leq x_1 \leq c \}$, for $c \in O(1)$. Pick a point $\mathbf{x}$ from the first Gaussian. After picking $\mathbf{x}$, rotate the coordinate system to make the first axis point towards $\mathbf{x}$. Independently pick a second point $\mathbf{y}$ also from the first Gaussian. The fact that almost all of the mass of the Gaussian is within the slab $\{\mathbf{x} \mid -c \leq x_1 \leq c,\ c \in O(1)\}$ at the equator implies that $\mathbf{y}$'s component along $\mathbf{x}$'s direction is $O(1)$ with high probability. Thus, $\mathbf{y}$ is nearly perpendicular to $\mathbf{x}$. So, $|\mathbf{x} - \mathbf{y}| \approx \sqrt{|\mathbf{x}|^2 + |\mathbf{y}|^2}$. See Figure 2.4 (a). More precisely, since the coordinate system has been rotated so that $\mathbf{x}$ is at the North Pole, $\mathbf{x} = (\sqrt{d} \pm O(1), 0, \ldots, 0)$. Since $\mathbf{y}$ is almost on the equator, further rotate the coordinate system so that the component of $\mathbf{y}$ that is perpendicular to the axis of the North Pole is in the second coordinate. Then $\mathbf{y} = (O(1), \sqrt{d} \pm O(1), 0, \ldots, 0)$. Thus,

$$(\mathbf{x} - \mathbf{y})^2 = d \pm O(\sqrt{d}) + d \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$$

and $|\mathbf{x} - \mathbf{y}| = \sqrt{2d} \pm O(1)$ with high probability.

Given two spherical unit variance Gaussians with centers $\mathbf{p}$ and $\mathbf{q}$ separated by a distance Δ, the distance between a randomly chosen point $\mathbf{x}$ from the first Gaussian and a randomly chosen point $\mathbf{y}$ from the second is close to $\sqrt{\Delta^2 + 2d}$, since $\mathbf{x} - \mathbf{p}, \mathbf{p} - \mathbf{q}$,

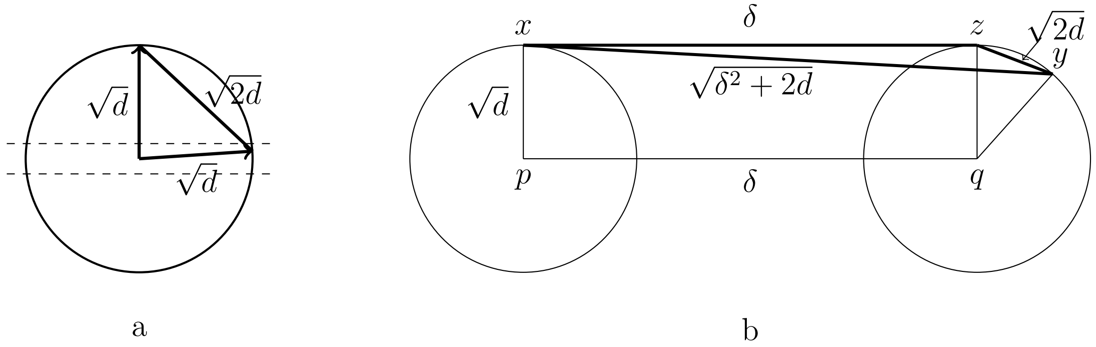


Figure 2.4: (a) indicates that two randomly chosen points in high dimension are almost surely nearly orthogonal. (b) indicates that the distance between a pair of random points from two different unit balls approximating the annuli of two Gaussians.

and $\mathbf{q} - \mathbf{y}$ are nearly mutually perpendicular. Pick $\mathbf{x}$ and rotate the coordinate system so that $\mathbf{x}$ is at the North Pole. Let $\mathbf{z}$ be the North Pole of the ball approximating the second Gaussian. Now pick $\mathbf{y}$. Most of the mass of the second Gaussian is within $O(1)$ of the equator perpendicular to $\mathbf{q} - \mathbf{z}$. Also, most of the mass of each Gaussian is within distance $O(1)$ of the respective equators perpendicular to the line $\mathbf{q} - \mathbf{p}$. See Figure 2.4 (b). Thus,

$$\begin{aligned} |\mathbf{x} - \mathbf{y}|^2 &\approx \Delta^2 + |\mathbf{z} - \mathbf{q}|^2 + |\mathbf{q} - \mathbf{y}|^2 \\ &= \Delta^2 + 2d \pm O(\sqrt{d})). \end{aligned}$$

To ensure that the distance between two points picked from the same Gaussian are closer to each other than two points picked from different Gaussians requires that the upper limit of the distance between a pair of points from the same Gaussian is at most the lower limit of distance between points from different Gaussians. This requires that $\sqrt{2d} + O(1) \leq \sqrt{2d + \Delta^2} - O(1)$ or $2d + O(\sqrt{d}) \leq 2d + \Delta^2$, which holds when $\Delta \in \omega(d^{1/4})$. Thus, mixtures of spherical Gaussians can be separated, provided their centers are separated by $\omega(d^{1/4})$. If we have n points and want to correctly separate all of them with high probability, we need our individual high-probability statements to hold with probability $1 - 1/poly(n)$, which means our $O(1)$ terms from Theorem 2.8 become $O(\sqrt{\log n})$. So we need to include an extra $O(\sqrt{\log n})$ term in the separation distance.

> **Algorithm for separating points from two Gaussians:** Calculate all pairwise distances between points. The cluster of smallest pairwise distances must come from a single Gaussian. Remove these points. The remaining points come from the second Gaussian.

One can actually separate Gaussians where the centers are much closer. In the next chapter we will use Singular Value Decomposition to separate a mixture of two Gaussians when their centers are separated by just a distance $O(1)$.

2.9 Fitting a single spherical Gaussian to data

Given a set of sample points, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$, in a d-dimensional space, we wish to find the spherical Gaussian that best fits the points. Let F be the unknown Gaussian with mean $\boldsymbol{\mu}$ and variance σ^2 in each direction. The probability density for picking these points when sampling according to F is given by

$$c \exp\left(-\ \frac{(\mathbf{x_1}-\boldsymbol{\mu})^2+(\mathbf{x_2}-\boldsymbol{\mu})^2+\cdots+(\mathbf{x_n}-\boldsymbol{\mu})^2}{2\sigma^2}\right)$$

where the normalizing constant c is the reciprocal of $\left[\int e^{-\frac{|\mathbf{x}-\boldsymbol{\mu}|^2}{2\sigma^2}}\,dx\right]^n$. In integrating from $-\infty$ to ∞, one could shift the origin to $\boldsymbol{\mu}$ and thus c is $\left[\int e^{-\frac{|\mathbf{x}|^2}{2\sigma^2}}dx\right]^{-n} = \frac{1}{(2\pi)^{\frac{n}{2}}}$ and is independent of $\boldsymbol{\mu}$.

The *Maximum Likelihood Estimator* (MLE) of F, given the samples $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$, is the F that maximizes the above probability density.

Lemma 2.12 *Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of n points in d-space. Then $(\mathbf{x}_1-\boldsymbol{\mu})^2+(\mathbf{x}_2-\boldsymbol{\mu})^2+\cdots+(\mathbf{x}_n-\boldsymbol{\mu})^2$ is minimized when $\boldsymbol{\mu}$ is the centroid of the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, namely $\boldsymbol{\mu} = \frac{1}{n}(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_n)$.*

Proof: Setting the gradient of $(\mathbf{x}_1-\boldsymbol{\mu})^2+(\mathbf{x}_2-\boldsymbol{\mu})^2+\cdots+(\mathbf{x}_n-\boldsymbol{\mu})^2$ with respect $\boldsymbol{\mu}$ to zero yields

$$-2\left(\mathbf{x}_1-\boldsymbol{\mu}\right)-2\left(\mathbf{x}_2-\boldsymbol{\mu}\right)-\cdots-2\left(\mathbf{x}_n-\boldsymbol{\mu}\right)=0.$$

Solving for $\boldsymbol{\mu}$ gives $\boldsymbol{\mu} = \frac{1}{n}(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_n)$. ■

To determine the maximum likelihood estimate of σ^2 for F, set $\boldsymbol{\mu}$ to the true centroid. Next, we show that σ is set to the standard deviation of the sample. Substitute $\nu = \frac{1}{2\sigma^2}$ and $a = (\mathbf{x}_1-\boldsymbol{\mu})^2+(\mathbf{x}_2-\boldsymbol{\mu})^2+\cdots+(\mathbf{x}_n-\boldsymbol{\mu})^2$ into the formula for the probability of picking the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. This gives

$$\frac{e^{-a\nu}}{\left[\int\limits_x e^{-x^2\nu}dx\right]^n}\ .$$

Now, a is fixed and ν is to be determined. Taking logs, the expression to maximize is

$$-a\nu - n\ln\left[\int\limits_x e^{-\nu x^2}dx\right].$$

To find the maximum, differentiate with respect to ν, set the derivative to zero, and solve for σ. The derivative is

$$-a + n\frac{\int\limits_{x} |x|^2 e^{-\nu x^2} dx}{\int\limits_{x} e^{-\nu x^2} dx}.$$

Setting $y = |\sqrt{\nu}\mathbf{x}|$ in the derivative, yields

$$-a + \frac{n}{\nu}\,\frac{\int\limits_{y} y^2 e^{-y^2} dy}{\int\limits_{y} e^{-y^2} dy}.$$

Since the ratio of the two integrals is the expected distance squared of a d-dimensional spherical Gaussian of standard deviation $\frac{1}{\sqrt{2}}$ to its center, and this is known to be $\frac{d}{2}$, we get $-a + \frac{nd}{2\nu}$. Substituting σ^2 for $\frac{1}{2\nu}$ gives $-a + nd\sigma^2$. Setting $-a + nd\sigma^2 = 0$ shows that the maximum occurs when $\sigma = \frac{\sqrt{a}}{\sqrt{nd}}$. Note that this quantity is the square root of the average coordinate distance squared of the samples to their mean, which is the standard deviation of the sample. Thus, we get the following lemma.

Lemma 2.13 *The maximum likelihood spherical Gaussian for a set of samples is the one with center equal to the sample mean and standard deviation equal to the standard deviation of the sample from the true mean.*

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be a sample of points generated by a Gaussian probability distribution. $\boldsymbol{\mu} = \frac{1}{n}(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_n)$ is an unbiased estimator of the expected value of the distribution. However, if in estimating the variance from the sample set, we use the estimate of the expected value rather than the true expected value, we will not get an unbiased estimate of the variance, since the sample mean is not independent of the sample set. One should use $\boldsymbol{\mu} = \frac{1}{n-1}(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_n)$ when estimating the variance. See Section **??** of the appendix.

2.10 Bibliographic Notes

The word vector model was introduced by Salton [SWY75]. Taylor series remainder material can be found in Whittaker and Watson 1990, pp. 95-96 and Section 12.8.4 of the appendix. There is vast literature on the Gaussian distribution, its properties, drawing samples according to it, etc. The reader can choose the level and depth according to his/her background. For Chernoff bounds and their applications, see [MU05] or [MR95b]. The proof here and the application to heavy-tailed distributions is simplified from [Kan09]. The original proof of the random projection theorem by Johnson and Lindenstrauss was complicated. Several authors used Gaussians to simplify the proof. See [Vem04] for details and applications of the theorem. The proof here is due to Dasgupta and Gupta [DG99].

2.11 Exercises

Exercise 2.1

1. *Let x and y be independent random variables with uniform distribution in $[0,1]$. What is the expected value $E(x)$, $E(x^2)$, $E(x-y)$, $E(xy)$, and $E((x-y)^2)$?*

2. *Let x and y be independent random variables with uniform distribution in $[-\frac{1}{2},\frac{1}{2}]$. What is the expected value $E(x)$, $E(x^2)$, $E(x-y)$, $E(xy)$, and $E((x-y)^2)$?*

3. *What is the expected squared distance between two points generated at random inside a unit d-dimensional cube centered at the origin?*

4. *Randomly generate a number of points inside a d-dimensional unit cube centered at the origin and plot distance between and the angle between the vectors from the origin to the points for all pairs of points.*

Exercise 2.2 *Show that Markov's inequality is tight by showing the following:*

1. *For each of $a = 2, 3$, and 4 give a probability distribution $p(x)$ for a nonnegative random variable x where Prob $\big(x \geq aE(x)\big) = \frac{1}{a}$.*

2. *For arbitrary $a \geq 1$ give a probability distribution for a nonnegative random variable x where Prob $\big(x \geq aE(x)\big) = \frac{1}{a}$.*

Exercise 2.3 *Give a probability distribution $p(x)$ and a value b for which Chebyshev's inequality is tight and a probability distribution and value of b for which it is not tight?*

Exercise 2.4 *Consider the probability function $p(x) = c\frac{1}{x^4}, \quad x \geq 1$, and generate 100 random samples. How close is the average of the samples to the expected value of x?*

Exercise 2.5 *Let G be a d-dimensional spherical Gaussian with variance $\frac{1}{2}$ centered at the origin. Derive the expected squared distance to the origin.*

Exercise 2.6 *How large must ε be for 99% of the volume of a d-dimensional unit-radius ball to lie in the shell of ε-thickness at the surface of the ball?*

Exercise 2.7 *A 3-dimensional cube has vertices, edges, and faces. In a d-dimensional cube, these components are called faces. A vertex is a 0-dimensional face, an edge a 1-dimensional face, etc.*

1. *For $0 \leq i \leq d$, how many i-dimensional faces does a d-dimensional cube have?*

2. *What is the total number of faces of all dimensions? The d-dimensional face is the cube itself which you can include in your count.*

3. *What is the surface area of a unit cube in d-dimensions?*

4. *What is the surface area of the cube if the length of each side was 2?*

5. *Prove that the volume of a unit cube is close to its surface.*

Exercise 2.8 *Repeat Exercise 2.7 for a d-dimensional tetrahedron.*

Exercise 2.9 *Consider the portion of the surface area of a unit radius, 3-dimensional ball with center at the origin that lies within a circular cone whose vertex is at the origin. What is the formula for the incremental unit of area when using polar coordinates to integrate the portion of the surface area of the ball that is lying inside the circular cone? What is the formula for the integral? What is the value of the integral if the angle of the cone is* 36°*? The angle of the cone is measured from the axis of the cone to a ray on the surface of the cone.*

Exercise 2.10 *For what value of d does the volume, $V(d)$, of a d-dimensional unit ball take on its maximum?*
Hint: Consider the ratio $\frac{V(d)}{V(d-1)}$.

Exercise 2.11 *How does the volume of a ball of radius two behave as the dimension of the space increases? What if the radius was larger than two but a constant independent of d? What function of d would the radius need to be for a ball of radius r to have approximately constant volume as the dimension increases?*

Exercise 2.12 *If* $\lim\limits_{d\to\infty} V(d) = 0$*, the volume of a d-dimensional sphere for sufficiently large d must be less than $V(3)$. How can this be if the d-dimensional sphere contains the three dimensional sphere?*

Exercise 2.13 *Consider a unit radius, circular cylinder in 3-dimensions of height one. The top of the cylinder could be an horizontal plane or half of a circular ball. Consider these two possibilities for a unit radius, circular cylinder in 4-dimensions. In 4-dimensions the horizontal plane is 3-dimensional and the half circular ball is 4-dimensional. In each of the two cases, what is the surface area of the top face of the cylinder? You can use $V(d)$ for the volume of a unit radius, d-dimension ball and $A(d)$ for the surface area of a unit radius, d-dimensional ball. An infinite length, unit radius, circular cylinder in 4-dimensions would be the set* $\{(x_1, x_2, x_3, x_4)|x_2^2 + x_3^2 + x_4^2 \leq 1\}$ *where the coordinate x_1 is the axis.*

Exercise 2.14 *Given a d-dimensional circular cylinder of radius r and height h*

1. *What is the surface area in terms of $V(d)$ and $A(d)$?*

2. *What is the volume?*

Exercise 2.15 *Write a recurrence relation for $V(d)$ in terms of $V(d-1)$ by integrating using an incremental unit that is a disk of thickness dr.*

Exercise 2.16 *Verify the formula* $V(d) = 2\int_0^1 V(d-1)(1-x_1^2)^{\frac{d-1}{2}}dx_1$ *for* $d = 1$ *and* $d = 2$ *by integrating and comparing with* $V(2) = \pi$ *and* $V(3) = \frac{4}{3}\pi$

Exercise 2.17 *Consider a unit ball* A *centered at the origin and a unit ball* B *whose center is at distance* s *from the origin. Suppose that a random point* x *is drawn from the mixture distribution: "with probability* $1/2$*, draw at random from* A*; with probability* $1/2$*, draw at random from* B*". Show that a separation* $s = \omega(1/\sqrt{d-1})$ *is sufficient so that* $Prob(x \in A \cap B) = o(1)$ *(i.e., for any* $\epsilon > 0$ *there exists* c *such that if* $s \geq c/\sqrt{d-1}$ *then* $Prob(x \in A \cap B) < \epsilon$*). In other words, this separation means that nearly all of the mixture distribution is identifiable.*

Exercise 2.18 *Prove that* $1 + x \leq e^x$ *for all real* x*. For what values of* x *is the approximation* $1 + x \approx e^x$ *within 0.01?*

Exercise 2.19 *Consider the upper hemisphere of a unit-radius ball in d-dimensions. What is the height of the maximum volume cylinder that can be placed entirely inside the hemisphere? As you increase the height of the cylinder, you need to reduce the cylinder's radius so that it will lie entirely within the hemisphere.*

Exercise 2.20 *What is the volume of the maximum size d-dimensional hypercube that can be placed entirely inside a unit radius d-dimensional ball?*

Exercise 2.21 *For a 1,000-dimensional unit-radius ball centered at the origin, what fraction of the volume of the upper hemisphere is above the plane* $x_1 = 0.1$*? Above the plane* $x_1 = 0.01$*?*

Exercise 2.22 *Calculate the ratio of area above the plane* $x_1 = \epsilon$ *of a unit radius ball in d-dimensions for* $\epsilon = 0.01, 0.02, 0.03, 0.04, 0.05$ *and for* $d = 100$ *and* $d = 1,000$*. Also calculate the ratio for* $\epsilon = 0.001$ *and* $d = 1,000$*.*

Exercise 2.23 *Let* $\left\{\mathbf{x}\,\middle|\,|\mathbf{x}| \leq 1\right\}$ *be a d-dimensional, unit radius ball centered at the origin. What fraction of the volume is the set* $\{(x_1, x_2, \ldots, x_d)\big|\ |x_i| \leq \frac{1}{\sqrt{d}}\}$*?*

Exercise 2.24 *Almost all of the volume of a ball in high dimensions lies in a narrow slice of the ball at the equator. However, the narrow slice is determined by the point on the surface of the ball that is designated the North Pole. Explain how this can be true if several different locations are selected for the location of the North Pole giving rise to different equators.*

Exercise 2.25 *Explain how the volume of a ball in high dimensions can simultaneously be in a narrow slice at the equator and also be concentrated in a narrow annulus at the surface of the ball.*

Exercise 2.26 *Generate 500 points uniformly at random on the surface of a unit-radius ball in 50 dimensions. Then randomly generate five additional points. For each of the five new points, calculate a narrow band at the equator, assuming the point was the North Pole. How many of the 500 points are in each band corresponding to one of the five equators? How many of the points are in all five bands? How wide do the bands need to be for all points to be in all five bands?*

Exercise 2.27 *Consider a slice of a 100-dimensional ball that lies between two parallel planes, each equidistant from the equator and perpendicular to the line from the North Pole to the South Pole. What percentage of the distance from the center of the ball to the poles must the planes be to contain 95% of the surface area?*

Exercise 2.28 *Place n points at random on a d-dimensional unit-radius ball. Assume d is large. Pick a random vector and let it define two parallel hyperplanes on opposite sides of the origin that are equal distance from the origin. How far apart can the hyperplanes be moved and still have the probability that none of the n points lands between them be at least .99?*

Exercise 2.29 *Consider two random vectors in a high-dimensional space. Assume the vectors have been normalized so that their lengths are one and thus the points lie on a unit ball. Assume one of the vectors is the North pole. Prove that the ratio of the area of a cone, with axis at the North Pole of fixed angle say* 45° *to the area of a hemisphere, goes to zero as the dimension increases. Thus, the probability that the angle between two random vectors is at most* 45° *goes to zero. How does this relate to the result that most of the volume is near the equator?*

Exercise 2.30 *Project the volume of a d-dimensions ball of radius $\sqrt{d}$ onto a line through the center. For large d, give an intuitive argument that the projected volume should behave like a Gaussian.*

Exercise 2.31

1. *Write a computer program that generates n points uniformly distributed over the surface of a unit-radius d-dimensional ball.*

2. *Generate 200 points on the surface of a sphere in 50 dimensions.*

3. *Create several random lines through the origin and project the points onto each line. Plot the distribution of points on each line.*

4. *What does your result from (3) say about the surface area of the sphere in relation to the lines, i.e., where is the surface area concentrated relative to each line?*

Exercise 2.32 *The volume of a radius r sphere in d-dimensions increases as r^d. A Gaussian probability distribution decreases as $e^{-\frac{r^2}{2}}$. Use calculus to determine the radius r at which the maximum probability mass of the Gaussian occurs.*

Exercise 2.33 *If one generates points in d-dimensions with each coordinate a unit variance Gaussian, the points will approximately lie on the surface of a sphere of radius $\sqrt{d}$.*

1. *What is the distribution when the points are projected onto a random line through the origin?*
2. *If one uses a Gaussian with variance four, where in d-space will the points lie?*

Exercise 2.34 *Randomly generate a 100 points on the surface of a sphere in 3-dimensions and in 100-dimensions. Create a histogram of all distances between the pairs of points in both cases.*

Exercise 2.35 *Consider drawing a random point $\mathbf{x}$ on the surface of the unit sphere in R^d. What is the variance of x_1 (the first coordinate of $\mathbf{x}$)? See if you can give an argument without doing any integrals.*

Exercise 2.36 *We have claimed that a randomly generated point on a ball lies near the equator of the ball, wherever we place the North Pole. Is the same claim true for a randomly generated point on a cube? To test this claim, randomly generate ten ± 1 valued vectors in 128 dimensions. Think of these ten vectors as ten choices for the North Pole. Then generate some additional ± 1 valued vectors. To how many of the original vectors is each of the new vectors close to being perpendicular; that is, how many of the equators is each new vector close to?*

Exercise 2.37 *Project the vertices of a high-dimensional cube onto a line from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$. Argue that the "density" of the number of projected points (per unit distance) varies roughly as a Gaussian with variance $O(1)$ with the mid-point of the line as center.*

Exercise 2.38 *Define the equator of a d-dimensional unit cube to be the hyperplane* $\left\{ \mathbf{x} \middle| \sum_{i=1}^{d} x_i = \frac{d}{2} \right\}$.

1. *Are the vertices of a unit cube concentrated close to the equator?*
2. *Is the volume of a unit cube concentrated close to the equator?*
3. *Is the surface area of a unit cube concentrated close to the equator?*

Exercise 2.39 *Let x be a random variable with probability density $\frac{1}{4}$ for $0 \le x \le 4$ and zero elsewhere.*

1. *Use Markov's inequality to bound the probability that $x > 3$.*
2. *Make use of $Prob(|x| > a) = Prob(x^2 > a^2)$ to get a tighter bound.*
3. *What is the bound using $Prob(|x| > a) = Prob(x^r > a^r)$?*

Exercise 2.40 *Consider the probability distribution $p(x = 0) = 1 - \frac{1}{a}$ and $p(x = a) = \frac{1}{a}$. Plot the probability that x is greater than or equal to b as a function of b for the bound given by Markov's inequality and by Markov's inequality applied to x^2 and x^4.*

Exercise 2.41 *Generate 20 points uniformly at random on a 1,000-dimensional sphere of radius 100. Calculate the distance between each pair of points. Then, project the data onto subspaces of dimension k=100, 50, 10, 5, 4, 3, 2, 1 and calculate the difference between $\sqrt{k}$ times the original distances and the new pair-wise distances. For each value of k what is the maximum difference as a percent of $\sqrt{k}$.*

Exercise 2.42 *In d-dimensions there are exactly d-unit vectors that are pairwise orthogonal. However, if you wanted a set of vectors that were almost orthogonal you might squeeze in a few more. For example, in 2-dimensions if almost orthogonal meant at least 45 degrees apart you could fit in three almost orthogonal vectors. Suppose you wanted to find 900 almost orthogonal vectors in 100 dimensions where almost orthogonal meant an angle of between 85 and 95 degrees. How would you generate such a set?*
Hint: Consider projecting a 1,000 orthonormal 1,000-dimensional vectors to a random 100-dimensional space.

Exercise 2.43 *Exercise 2.42 finds almost orthogonal vectors using the Johnson Lindenstrauss theorem. One could also create almost orthogonal vectors by generating random Gaussian vectors. Compare the two results to see which does a better job.*

Exercise 2.44 *To preserve pairwise distances between n data points in d space, we projected to a random $O(\ln n/\varepsilon^2)$ dimensional space. To save time in carrying out the projection, we may try to project to a space spanned by sparse vectors, vectors with only a few nonzero entries. That is, choose say $O(\ln n/\varepsilon^2)$ vectors at random, each with 100 nonzero components and project to the space spanned by them. Will this work to preserve approximately all pairwise distances? Why?*

Exercise 2.45 *Suppose there is an object moving at constant velocity along a straight line. You receive the gps coordinates corrupted by Gaussian noise every minute. How do you estimate the current position?*

Exercise 2.46

1. *What is the maximum size rectangle that can be fitted in a unit variance Gaussian?*
2. *What rectangle best approximates a unit variance Gaussian if one measure goodness of fit by how small the symmetric difference of the Gaussian and rectangle is.*

Exercise 2.47 *Let $x_1, x_2, \ldots, x_n$ be independent samples of a random variable $\mathbf{x}$ with mean m and variance σ^2. Let $m_s = \frac{1}{n}\sum_{i=1}^{n} x_i$ be the sample mean. Suppose one estimates the variance using the sample mean rather than the true mean, that is,*

$$\sigma_s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - m_s)^2$$

Prove that $E(\sigma_s^2) = \frac{n-1}{n}\sigma^2$ and thus one should have divided by $n-1$ rather than n.

Hint: First calculate the variance of the sample mean and show that $var(m_s) = \frac{1}{n} var(\mathbf{x})$. Then calculate $E(\sigma_s^2) = E[\frac{1}{n}\sum_{i=1}^{n}(x_i - m_s)^2]$ by replacing $x_i - m_s$ with $(x_i - m) - (m_s - m)$.

Exercise 2.48 *Generate ten values by a Gaussian probability distribution with zero mean and variance one. What is the center determined by averaging the points? What is the variance? In estimating the variance, use both the real center and the estimated center. When using the estimated center to estimate the variance, use both $n = 10$ and $n = 9$. How do the three estimates compare?*

Exercise 2.49 *Suppose you want to estimate the unknown center of a Gaussian in d-space which has variance one in each direction. Show that $O(\log d/\varepsilon^2)$ random samples from the Gaussian are sufficient to get an estimate $\tilde{\mu}$ of the true center μ, so that with probability at least 99/100,*

$$|\mu - \tilde{\mu}|_\infty \leq \varepsilon.$$

How many samples are sufficient to ensure that

$$|\mu - \tilde{\mu}| \leq \varepsilon?$$

Exercise 2.50 *Use the probability distribution $\frac{1}{3\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-5)^2}{9}}$ to generate ten points.*

(a) *From the ten points estimate μ. How close is the estimate of μ to the true mean of 5?*

(b) *Using the true mean of 5, estimate σ^2 by the fomula $\sigma^2 = \frac{1}{10}\sum_{i=1}^{10}(x_i - 5)^2$. How close is the estimate of σ^2 to the true variance of 9?*

(c) *Using your estimate of the mean, estimate σ^2 by the fomula $\sigma^2 = \frac{1}{10}\sum_{i=1}^{10}(x_i - 5)^2$. How close is the estimate of σ^2 to the true variance of 9?*

(d) *Using your estimate of the mean, estimate σ^2 by the fomula $\sigma^2 = \frac{1}{9}\sum_{i=1}^{10}(x_i - 5)^2$. How close is the estimate of σ^2 to the true variance of 9?*

Exercise 2.51 *Create a list of the five most important things that you learned about high dimensions.*

Exercise 2.52 *Write a short essay whose purpose is to excite a college freshman to learn about high dimensions.*

3 Best-Fit Subspaces and Singular Value Decomposition (SVD)

3.1 Introduction and Overview

In this chapter, we will examine the *Singular Value Decomposition* (SVD) of a matrix. Consider each row of an $n \times d$ matrix A as a point in d-dimensional space. The singular value decomposition will find the best-fitting k-dimensional subspace for $k = 1, 2, 3, \ldots,$ for the associated set of n data points. Here, "best" means minimizing the sum of the squares of the perpendicular distances of the points to the subspace, or equivalently, maximizing the sum of squares of the lengths of the projections of the points onto this subspace.[2] We begin with a special case where the subspace is 1-dimensional, namely a line through the origin. We will then show that the best-fitting k-dimensional subspace can be found by k applications of the best fitting line algorithm, where on each iteration i we find the line of best fit subject to being perpendicular to the previous $i-1$ lines. When k reaches the rank of the matrix, from these operations we can get an exact decomposition of the matrix called the Singular Value Decomposition.

In matrix notation, the singular value decomposition of a matrix A with real entries (we assume all our matrices have real entries) is the factorization of A into the product of three matrices, $A = UDV^T$, where the columns of U and V are orthonormal[3] and the matrix D is diagonal with positive real entries. The columns of V are the unit length vectors defining the best fitting lines described above (the i^{th} column being the unit-length vector in the direction of the i^{th} line). The coordinates of a row of U will be the fractions of the corresponding row of A along the direction of each of the lines.

The SVD is useful in many tasks. In many applications, a data matrix A is close to a low rank matrix and it is useful to find a good low rank approximation to A. The singular value decomposition of A gives the best rank k approximations to A, for any k .

If $\mathbf{u^{(i)}}$ and $\mathbf{v^{(i)}}$ are columns of U and V respectively, then the matrix equation $A = UDV^T$ can be rewritten as

$$A = \sum_i D_{ii} \mathbf{u^{(i)}} \mathbf{v^{(i)}}^T,$$

where, note that since $\mathbf{u^{(i)}}$ is a $n \times 1$ matrix and $\mathbf{v^{(i)}}$ is a $d \times 1$ matrix, $\mathbf{u^{(i)}} \mathbf{v^{(i)}}^T$ is an $n \times d$ matrix with the same dimensions as A. The i^{th} term in the sum can be viewed as giving the components of the rows of A along direction $\mathbf{v^{(i)}}$. When the terms are summed, they

[2] This equivalence is due to the Pythagorean Theorem. For each point, its squared length (its distance to the origin squared) is exactly equal to the squared length of its projection onto the subspace plus the squared distance of the point to its projection; therefore, maximizing the sum of the former is equivalent to minimizing the sum of the latter. For further discussion see Section 3.2.

[3] A set of vectors is orthonormal if each is of length one and they are pairwise orthogonal.

reconstruct A.

In addition to the singular value decomposition, there is an eigenvalue decomposition. Let A be a square matrix. A vector $\mathbf{v}$ such that $A\mathbf{v} = \lambda\mathbf{v}$ is called an eigenvector and λ the eigenvalue. When A satisfies a few additional conditions besides being square, the eigenvectors are orthogonal and A can be expressed as $A = VDV^T$ where the eigenvectors are the columns of V and D is a diagonal matrix with the corresponding eigenvalues on its diagonal. If A is symmetric, then the singular values of A are its eigenvalues and the singular vectors of A are the eigenvectors. If a singular value has multiplicity d greater than one, the corresponding singular vectors span a subspace of dimensions d and any orthogonal basis of the subspace can be used as the eigenvectors or singular vectors.

The singular value decomposition is defined for all matrices, whereas the more familiar eigenvector decomposition requires that the matrix A be square and certain other conditions on the matrix to ensure orthogonality of the eigenvectors. In contrast, the columns of V in the singular value decomposition, called the *right-singular vectors* of A, always form an orthogonal set with no assumptions on A. The columns of U are called the *left-singular vectors* and they also form an orthogonal set (this will be proved in Section 3.7). A simple consequence of the orthonormality is that for a square and invertible matrix A, the inverse of A is $VD^{-1}U^T$.

Eigenvalues and eignevectors satisfy $A\mathbf{v} = \lambda\mathbf{v}$. We will show that singular values and vectors satisfy a somewhat analogous relationship. Since $A\mathbf{v^{(i)}}$ is a $n \times 1$ matrix (vector), the matrix A cannot act on it from the left. But A^T which is $d \times n$ can act on this vector. Indeed, we will show that

$$A\mathbf{v^{(i)}} = D_{ii}\mathbf{u^{(i)}} \quad \text{and} \quad A^T\mathbf{u^{(i)}} = D_{ii}\mathbf{v^{(i)}}.$$

In words, A acting on $\mathbf{v^{(i)}}$ produces a scalar multiple of $\mathbf{u^{(i)}}$ and A^T acting on $\mathbf{u^{(i)}}$ produces the same scalar multiple of $\mathbf{v^{(i)}}$.

3.2 Preliminaries

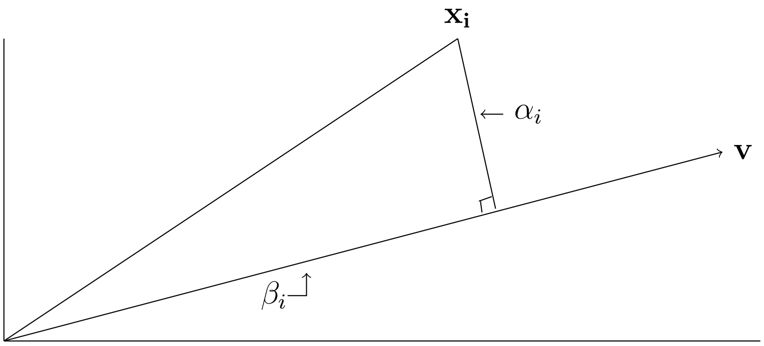
Consider projecting a point $\mathbf{a_i} = (a_{i1}, a_{i2}, \ldots, a_{id})$ onto a line through the origin. Then

$$a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2 = (\text{length of projection})^2 + (\text{distance of point to line})^2 .$$

This holds by the Pythagorean Theorem; see Figure 3.1. Thus

$$(\text{distance of point to line})^2 = a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2 - (\text{length of projection})^2 .$$

Since $\sum\limits_{i=1}^{n} (a_{i1}^2 + a_{i2}^2 + \cdots + a_{id}^2)$ is a constant independent of the line, minimizing the sum of the squares of the distances to the line is equivalent to maximizing the sum of the squares of the lengths of the projections onto the line. Similarly for best-fit subspaces,



Minimizing $\sum\limits_i \alpha_i^2$ is equivalent to maximizing $\sum\limits_i \beta_i^2$

Figure 3.1: The projection of the point $\mathbf{x_i}$ onto the line through the origin in the direction of $\mathbf{v}$.

maximizing the sum of the squared lengths of the projections onto the subspace minimizes the sum of squared distances to the subspace.

Thus we have two interpretations of the best-fit subspace. The first is that it minimizes the sum of squared distances of the data points to it. This interpretation and its use are akin to the notion of least-squares fit from Calculus.[4] The second interpretation of best-fit-subspace is that it maximizes the sum of projections squared of the data points on it. This says that the subspace contains the maximum content of data among all subspaces of the same dimension.

The reader may wonder why we minimize the sum of squared perpendicular distances to the line rather than, say, the sum of distances (not squared). There are examples where the latter definition gives a different answer than the line minimizing the sum of squared perpendicular distances. The choice of the objective function as the sum of squared distances seems a bit arbitrary and in a way it is. But the square has many nice mathematical properties. The first of these, as we have just seen, is that minimizing the sum of squared distances is equivalent to maximizing the sum of squared projections.

3.3 Centering Data

The centroid of a set of points is the coordinate-wise average of the points.

$$\text{Centroid } (S) = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \mathbf{x}.$$

We first show that the line minimizing the sum of squared distances to S, if not restricted to go through the origin, must pass through the centroid of S. This implies that if the

[4] But there is a difference: here we take the perpendicular distance to the line or subspace, whereas, in the calculus notion, given n pairs, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we find a line $l = \{(x, y) : y = mx + b\}$ minimizing the *vertical* distance of the points to it, namely, $\sum_{i=1}^{n}(y_i - mx_i - b)^2$.

centroid is subtracted from each data point, such a line will pass through the origin. The best fit line can be generalized to k dimensional "planes". The operation of subtracting the centroid from all data points is useful in other contexts as well. So we give it the name "centering data".

Lemma 3.1 *The best-fit line (minimizing the sum of perpendicular distances squared) of a set of data points must pass through the centroid of the points.*

Proof: Subtract the centroid from each data point so that the centroid is $\mathbf{0}$. Let ℓ be the best-fit line and assume for contradiction that ℓ does not pass through the origin. The line ℓ can be written as $\{\mathbf{a}+\lambda\mathbf{v}|\lambda \in \mathbf{R}\}$, where $\mathbf{a}$ is the closest point to $\mathbf{0}$ on ℓ and $\mathbf{v}$ is a unit length vector in the direction of ℓ, which is perpendicular to $\mathbf{a}$. For a data point $\mathbf{a_i}$, let $dist(\mathbf{a_i}, \ell)$ denote its perpendicular distance to ℓ. By the Pythagorean Theorem, we have $|\mathbf{a_i}-\mathbf{a}|^2 = dist(\mathbf{a_i},\ell)^2 + (\mathbf{v}\cdot\mathbf{a_i})^2$, or equivalently, $dist(\mathbf{a_i},\ell)^2 = |\mathbf{a_i}-\mathbf{a}|^2 - (\mathbf{v}\cdot\mathbf{a_i})^2$. Summing over all data points:

$$\sum_{i=1}^{n} dist(\mathbf{a_i},\ell)^2 = \sum_{i=1}^{n}\left(|\mathbf{a_i}-\mathbf{a}|^2 - (\mathbf{v}\cdot\mathbf{a_i})^2\right) = \sum_{i=1}^{n}\left(|\mathbf{a_i}|^2 + |\mathbf{a}|^2 - 2\mathbf{a_i}\cdot\mathbf{a} - (\mathbf{v}\cdot\mathbf{a_i})^2\right)$$
$$= \sum_{i=1}^{n}|\mathbf{a_i}|^2 + n|\mathbf{a}|^2 - 2\mathbf{a}\cdot\left(\sum_i \mathbf{a_i}\right) - \sum_{i=1}^{n}(\mathbf{v}\cdot\mathbf{a_i})^2 = \sum_i |\mathbf{a_i}|^2 + n|\mathbf{a}|^2 - \sum_i(\mathbf{v}\cdot\mathbf{a_i})^2,$$

where, we have used the fact that since the centroid is $\mathbf{0}$, $\sum_i \mathbf{a_i} = 0$. The above expression is minimized when $\mathbf{a}=\mathbf{0}$, so the line $\ell' = \{\lambda\mathbf{v} : \lambda \in \mathbf{R}\}$ through the origin is a better fit than ℓ, contradicting ℓ being the best-fit line. ■

Note that this proof, as well as others, would not work for sum of distances or sum of the cubes of distances instead of sum of squares, since the Pythagorean Theorem only applies to squared distances.

A statement analogous to Lemma 3.1 holds for higher dimensional objects. Define an *affine space* as a subspace translated by a vector. So an affine space is a set of the form

$$\{\mathbf{v_0} + \sum_{i=1}^{k}\lambda_i\mathbf{v_i}|\lambda_1,\lambda_2,\ldots,\lambda_k \in \mathbf{R}\}.$$

Here, $\mathbf{v_0}$ is the translation and $\mathbf{v_1},\mathbf{v_2},\ldots,\mathbf{v_k}$ form an orthonormal basis for the subspace.

Lemma 3.2 *The k dimensional affine space which minimizes the sum of squared perpendicular distances to the data points must pass through the centroid of the points.*

Proof: We only give a brief idea of the proof, which is similar to the previous lemma. Instead of $(\mathbf{v}\cdot\mathbf{a_i})^2$, we will now have $\sum_{t=1}^{k}(\mathbf{v_t}\cdot\mathbf{a_i})^2$, where, $\mathbf{v_t}, t = 1,2,\ldots,k$ are an orthonormal basis of the subspace through the origin parallel to the affine space. ■

3.4 Singular Vectors

We now define the *singular vectors* of an $n \times d$ matrix A. Consider the rows of A as n points in a d-dimensional space. Consider the best fit line through the origin. Let $\mathbf{v}$ be a unit vector along this line. The length of the projection of $\mathbf{a_i}$, the i^{th} row of A, onto $\mathbf{v}$ is $|\mathbf{a_i} \cdot \mathbf{v}|$. From this we see that the sum of the squared lengths of the projections is $|A\mathbf{v}|^2$. The best fit line is the one maximizing $|A\mathbf{v}|^2$ and hence minimizing the sum of the squared distances of the points to the line.

With this in mind, define the *first singular vector* $\mathbf{v_1}$ of A, a column vector, as

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} |A\mathbf{v}|.$$

Technically, there may be a tie for the vector attaining the maximum and so we should not use the article "the". If there is a tie, we arbitrarily pick one of the vectors and refer to it as "the first singular vector" avoiding the more cumbersome "one of the the vectors achieving the maximum". We adopt this terminology for all uses of $\arg\max$.

The value $\sigma_1(A) = |A\mathbf{v_1}|$ is called the *first singular value* of A. Note that $\sigma_1^2 = \sum\limits_{i=1}^{n}(\mathbf{a_i} \cdot \mathbf{v_1})^2$ is the sum of the squared lengths of the projections of the points onto the line determined by $\mathbf{v_1}$.

So if the data points were all either on a line or close to a line, intuitively, $\mathbf{v_1}$ should give us the direction of that line. It is possible that data points are not close to one line, but lie close to a 2-dimensional plane or more generally a low dimensional space. A widely applied technique called Principal Component Analysis (PCA) deals with such situations using singular vectors. Suppose we have an algorithm for finding $\mathbf{v_1}$ (we will describe one such algorithm later). How do we use this to find the best-fit 2-dimensional plane or more generally k-dimensional space? As above, we may assume after centering data that the space passes through the origin.

The greedy approach begins by finding $\mathbf{v_1}$ and then finds the best 2-dimensional subspace containing $\mathbf{v_1}$. The fact that we are using the sum of squared distances helps. For every 2-dimensional subspace containing $\mathbf{v_1}$, the sum of squared lengths of the projections onto the subspace equals the sum of squared projections onto $\mathbf{v_1}$ plus the sum of squared projections along a vector perpendicular to $\mathbf{v_1}$ in the subspace. Thus, instead of looking for the best 2-dimensional subspace containing $\mathbf{v_1}$, look for a unit vector $\mathbf{v_2}$ perpendicular to $\mathbf{v_1}$ that maximizes $|A\mathbf{v}|^2$ among all such unit vectors. Using the same greedy strategy to find the best three and higher dimensional subspaces, defines $\mathbf{v_3}, \mathbf{v_4}, \ldots$ in a similar manner. This is captured in the following definitions. There is no apriori guarantee that the greedy algorithm gives the best fit. But, in fact, the greedy algorithm does work and yields the best-fit subspaces of every dimension as we will show.

The *second singular vector*, $\mathbf{v_2}$, is defined by the best fit line perpendicular to $\mathbf{v_1}$.

$$\mathbf{v_2} = \arg\max_{\substack{\mathbf{v}\perp\mathbf{v_1}\\|\mathbf{v}|=1}} |A\mathbf{v}|$$

The value $\sigma_2(A) = |A\mathbf{v_2}|$ is called the *second singular value* of A**.** The *third singular vector* $\mathbf{v}_3$ and *third singular value* are defined similarly by

$$\mathbf{v_3} = \arg\max_{\substack{\mathbf{v}\perp\mathbf{v_1},\mathbf{v_2}\\|\mathbf{v}|=1}} |A\mathbf{v}|$$

and

$$\sigma_3(A) = |A\mathbf{v_3}|,$$

and so on. The process stops when we have found singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$, singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$, and

$$\max_{\substack{\mathbf{v}\perp\mathbf{v_1},\mathbf{v_2},\ldots,\mathbf{v_r}\\|\mathbf{v}|=1}} |A\mathbf{v}| = 0.$$

The greedy algorithm found the $\mathbf{v_1}$ that maximized $|A\mathbf{v}|$ and then the best fit 2-dimensional subspace containing $\mathbf{v_1}$. Is this necessarily the best-fit 2-dimensional subspace overall? Yes the following theorem establishes that the greedy algorithm finds the best subspaces of every dimension.

Theorem 3.3 (The Greedy Algorithm Works) *Let A be an $n \times d$ matrix with singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. For $1 \leq k \leq r$, let V_k be the subspace spanned by $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$. For each k, V_k is the best-fit k-dimensional subspace for A.*

Proof: The statement is obviously true for $k = 1$. For $k = 2$, let W be a best-fit 2-dimensional subspace for A. For any orthonormal basis $(\mathbf{w_1}, \mathbf{w_2})$ of W, $|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2$ is the sum of squared lengths of the projections of the rows of A onto W. Choose an orthonormal basis $(\mathbf{w}_1, \mathbf{w_2})$ of W so that $\mathbf{w_2}$ is perpendicular to $\mathbf{v_1}$. If $\mathbf{v_1}$ is perpendicular to W, any unit vector in W will do as $\mathbf{w_2}$. If not, choose $\mathbf{w_2}$ to be the unit vector in W perpendicular to the projection of $\mathbf{v_1}$ onto W. This makes $\mathbf{w_2}$ perpendicular to $\mathbf{v_1}$. Since $\mathbf{v_1}$ maximizes $|A\mathbf{v}|^2$, it follows that $|A\mathbf{w_1}|^2 \leq |A\mathbf{v_1}|^2$. Since $\mathbf{v_2}$ maximizes $|A\mathbf{v}|^2$ over all $\mathbf{v}$ perpendicular to $\mathbf{v_1}$, $|A\mathbf{w_2}|^2 \leq |A\mathbf{v_2}|^2$. Thus

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2.$$

Hence, V_2 is at least as good as W and so is a best-fit 2-dimensional subspace.

For general k, proceed by induction. By the induction hypothesis, V_{k-1} is a best-fit k-1 dimensional subspace. Suppose W is a best-fit k-dimensional subspace. Choose an orthonormal basis $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_k}$ of W so that $\mathbf{w_k}$ is perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$. Then

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 + \cdots + |A\mathbf{w_{k-1}}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2 + \cdots + |A\mathbf{v_{k-1}}|^2$$

since V_{k-1} is an optimal $k-1$ dimensional subspace. Since $\mathbf{w_k}$ is perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k-1}}$, by the definition of $\mathbf{v_k}$, $|A\mathbf{w_k}|^2 \leq |A\mathbf{v_k}|^2$. Thus

$$|A\mathbf{w_1}|^2 + |A\mathbf{w_2}|^2 + \cdots + |A\mathbf{w_{k-1}}|^2 + |A\mathbf{w_k}|^2 \leq |A\mathbf{v_1}|^2 + |A\mathbf{v_2}|^2 + \cdots + |A\mathbf{v_{k-1}}|^2 + |A\mathbf{v_k}|^2,$$

proving that V_k is at least as good as W and hence is optimal. ■

Note that the n-vector $A\mathbf{v_i}$ is a list of lengths (with signs) of the projections of the rows of A onto $\mathbf{v_i}$. Think of $|A\mathbf{v_i}| = \sigma_i(A)$ as the "component" of the matrix A along $\mathbf{v_i}$. For this interpretation to make sense, it should be true that adding up the squares of the components of A along each of the $\mathbf{v_i}$ gives the square of the "whole content of the matrix A". This is indeed the case and is the matrix analogy of decomposing a vector into its components along orthogonal directions.

Consider one row, say $\mathbf{a_j}$, of A. Since $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ span the space of all rows of A, $\mathbf{a_j} \cdot \mathbf{v} = 0$ for all $\mathbf{v}$ perpendicular to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. Thus, for each row $\mathbf{a_j}$, $\sum\limits_{i=1}^{r} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = |\mathbf{a_j}|^2$. Summing over all rows j,

$$\sum_{j=1}^{n} |\mathbf{a_j}|^2 = \sum_{j=1}^{n} \sum_{i=1}^{r} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = \sum_{i=1}^{r} \sum_{j=1}^{n} (\mathbf{a_j} \cdot \mathbf{v_i})^2 = \sum_{i=1}^{r} |A\mathbf{v_i}|^2 = \sum_{i=1}^{r} \sigma_i^2(A).$$

But $\sum\limits_{j=1}^{n} |\mathbf{a_j}|^2 = \sum\limits_{j=1}^{n} \sum\limits_{k=1}^{d} a_{jk}^2$, the sum of squares of all the entries of A. Thus, the sum of squares of the singular values of A is indeed the square of the "whole content of A", i.e., the sum of squares of all the entries. There is an important norm associated with this quantity, the Frobenius norm of A, denoted $||A||_F$ defined as

$$||A||_F = \sqrt{\sum_{j,k} a_{jk}^2}.$$

Lemma 3.4 *For any matrix A, the sum of squares of the singular values equals the square of the Frobenius norm. That is, $\sum \sigma_i^2(A) = ||A||_F^2$.*

Proof: By the preceding discussion. ■

The vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ are called the *right-singular vectors*. The vectors $A\mathbf{v_i}$ form a fundamental set of vectors and we normalize them to length one by

$$\mathbf{u_i} = \frac{1}{\sigma_i(A)} A\mathbf{v_i}.$$

Later we will show that $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$ similarly maximize $|\mathbf{u}^T A|$ when multiplied on the left and are called the *left-singular vectors*. Clearly, the right-singular vectors are orthogonal by definition. We will show later that the left-singular vectors are also orthogonal.

3.5 Singular Value Decomposition (SVD)

Let A be an $n \times d$ matrix with singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ and corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$. The left-singular vectors of A are $\mathbf{u_i} = \frac{1}{\sigma_i} A\mathbf{v_i}$ where $\sigma_i \mathbf{u_i}$ is a vector whose coordinates correspond to the projections of the rows of A onto $\mathbf{v_i}$. Each $\sigma_i \mathbf{u_i} \mathbf{v_i^T}$ is a rank one matrix whose rows are the "$\mathbf{v_i}$ components" of the rows of A, i.e., the projections of the rows of A in the $\mathbf{v_i}$ direction. We will prove that A can be decomposed into a sum of rank one matrices as

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T.$$

Geometrically, this is just decomposing each point in A into its components along each of the r orthogonal directions given by the $\mathbf{v_i}$. We will also prove this algebraically. We begin with a simple lemma that two matrices A and B are identical if $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$.

Lemma 3.5 *Matrices A and B are identical if and only if for all vectors $\mathbf{v}$, $A\mathbf{v} = B\mathbf{v}$.*

Proof: Clearly, if $A = B$ then $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$. For the converse, suppose that $A\mathbf{v} = B\mathbf{v}$ for all $\mathbf{v}$. Let $\mathbf{e_i}$ be the vector that is all zeros except for the i^{th} component which has value one. Now $A\mathbf{e_i}$ is the i^{th} column of A and thus $A = B$ if for each i, $A\mathbf{e_i} = B\mathbf{e_i}$. ∎

Theorem 3.6 *Let A be an $n \times d$ matrix with right-singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$, left-singular vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_r}$, and corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$. Then*

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T.$$

Proof: We first show that multiplying A and $\sum\limits_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ by $\mathbf{v_j}$ results in equality.

$$\sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{v_j} = \sigma_j \mathbf{u_j} = A\mathbf{v_j}$$

Since any vector $\mathbf{v}$ can be expressed as a linear combination of the singular vectors plus a vector perpendicular to the $\mathbf{v_i}$, $A\mathbf{v} = \sum\limits_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{v}$ for all $\mathbf{v}$ and by Lemma 3.5, $A = \sum\limits_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$. ∎

The decomposition $A = \sum_i \sigma_i \mathbf{u_i} \mathbf{v_i^T}$ is called the *singular value decomposition*, *SVD*, of A. In matrix notation $A = UDV^T$ where the columns of U and V consist of the left and right-singular vectors, respectively, and D is a diagonal matrix whose diagonal entries are

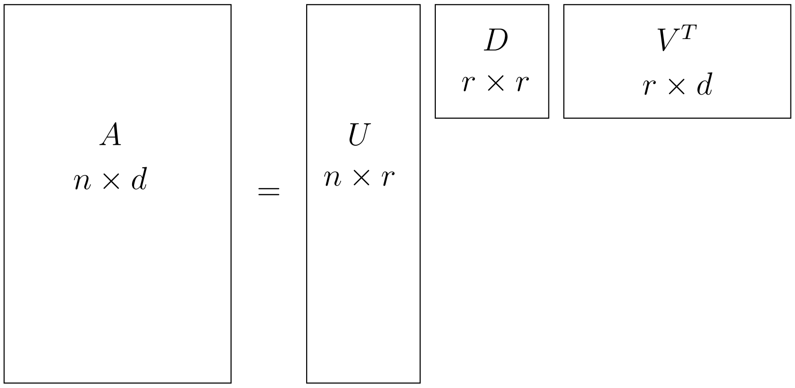


Figure 3.2: The SVD decomposition of an $n \times d$ matrix.

the singular values of A. For any matrix A, the sequence of singular values is unique and if the singular values are all distinct, then the sequence of singular vectors is unique up to signs. However, when some set of singular values are equal, the corresponding singular vectors span some subspace. Any set of orthonormal vectors spanning this subspace can be used as the singular vectors.

3.6 Best Rank-k Approximations

Let A be an $n \times d$ matrix and think of the rows of A as n points in d-dimensional space. Let

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the SVD of A. For $k \in \{1, 2, \ldots, r\}$, let

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the sum truncated after k terms. It is clear that A_k has rank k. We show that A_k is the best rank k approximation to A, where, error is measured in Frobenius norm. Geometrically, this is just saying that $\mathbf{v_1}, \ldots, \mathbf{v_k}$ indeed defines the k-dimensional space minimizing the sum of squared distances of the points to the space. To see why, we need the following lemma.

Lemma 3.7 *The rows of A_k are the projections of the rows of A onto the subspace V_k spanned by the first k singular vectors of A.*

Proof: Let $\mathbf{a}$ be an arbitrary row vector. Since the $\mathbf{v_i}$ are orthonormal, the projection of the vector $\mathbf{a}$ onto V_k is given by $\sum_{i=1}^{k} (\mathbf{a} \cdot \mathbf{v_i}) \mathbf{v_i}^T$. Thus, the matrix whose rows are the projections of the rows of A onto V_k is given by $\sum_{i=1}^{k} A\mathbf{v_i}\mathbf{v_i}^T$. This last expression simplifies to $\sum_{i=1}^{k} A\mathbf{v_i}\mathbf{v_i}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^T = A_k$. ■

Theorem 3.8 *For any matrix B of rank at most k*

$$\|A - A_k\|_F \leq \|A - B\|_F$$

Proof: Let B minimize $\|A - B\|_F^2$ among all rank k or less matrices. Let V be the space spanned by the rows of B. The dimension of V is at most k. Since B minimizes $\|A - B\|_F^2$, it must be that each row of B is the projection of the corresponding row of A onto V: Otherwise replace the row of B with the projection of the corresponding row of A onto V. This still keeps the row space of B contained in V and hence the rank of B is still at most k. But it reduces $\|A - B\|_F^2$, contradicting the minimality of $||A - B||_F$.

Since each row of B is the projection of the corresponding row of A, it follows that $\|A - B\|_F^2$ is the sum of squared distances of rows of A to V. Since A_k minimizes the sum of squared distance of rows of A to any k-dimensional subspace, from Theorem 3.3, it follows that $\|A - A_k\|_F \leq \|A - B\|_F$. ■

There is another matrix norm, which is of interest. To motivate, consider the example of a term-document matrix A. Suppose we have a large database of documents that form rows of an $n \times d$ matrix A. There are d terms and each document is a d-vector with one component per term, which is the number of occurrences of the term in the document. We are allowed to "preprocess" A. After the preprocessing, we receive queries. Each query $\mathbf{x}$ is an d-vector which specifies how important each term is to the querier. The desired answer is a n-vector which gives the similarity (dot product) of the query to each document in the database, namely $A\mathbf{x}$, the "matrix-vector" product. Query time is to be much less than preprocessing time, since the idea is that we need to answer many queries for the same database. Besides this, there are many situations where one performs many matrix vector products with the same matrix. This applicable to these situations as well. Näively, it would take $O(nd)$ time to do the product $A\mathbf{x}$. Suppose we did SVD and took $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^T$ as our approximation to A. Then, we could return $A_k\mathbf{x} = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}(\mathbf{v_i} \cdot \mathbf{x})$ as the approximation to $A\mathbf{x}$. This only takes k dot products of d-vectors, followed by a sum of k n-vectors, and so takes time $O(kd + kn)$, which is a win provided $k \ll \min(d, n)$. How is the error measured? Since $\mathbf{x}$ is unknown, the approximation needs to be good for every $\mathbf{x}$. So we should take the maximum over all $\mathbf{x}$ of $|(A_k - A)\mathbf{x}|$. But unfortunately, this is infinite since $|\mathbf{x}|$ can grow without bound. So we restrict to $|\mathbf{x}| \leq 1$. Formally, we define a new norm of a matrix A by

$$||A||_2 = \max_{\mathbf{x}:|\mathbf{x}|\leq 1} |A\mathbf{x}|.$$

This is called the 2-norm or the spectral norm. Note that it is indeed just equal to $\sigma_1(A)$.

We will prove in Section 3.7 the following theorem stateing that A_k is the best rank k approximation to A, when, error is measured by the spectral norm.

Theorem 3.9 *Let A be an $n \times d$ matrix. For any matrix B of rank at most k*

$$\|A - A_k\|_2 \leq \|A - B\|_2 .$$

3.7 Left Singular Vectors

Theorem 3.10 *The left singular vectors are pairwise orthogonal.*

Proof: First we show that each $\mathbf{u_i}, i \geq 2$ is orthogonal to $\mathbf{u_1}$. Suppose not, and for some $i \geq 2$, $\mathbf{u_1^T u_i} \neq 0$. Without loss of generality assume that $\mathbf{u_1^T u_i} = \delta > 0$. (If $\mathbf{u_1}^T\mathbf{u_i} < 0$ then just replace $\mathbf{u_i}$ with $-\mathbf{u_i}$.) For $\varepsilon > 0$, let

$$\mathbf{v_1'} = \frac{\mathbf{v_1} + \varepsilon \mathbf{v_i}}{|\mathbf{v_1} + \varepsilon \mathbf{v_i}|}.$$

Notice that $\mathbf{v_1'}$ is a unit-length vector.

$$A\mathbf{v_1'} = \frac{\sigma_1 \mathbf{u_1} + \varepsilon \sigma_i \mathbf{u_i}}{\sqrt{1+\varepsilon^2}}$$

has length at least as large as its component along $\mathbf{u_1}$ which is

$$\mathbf{u_1^T}(\frac{\sigma_1 \mathbf{u_1} + \varepsilon \sigma_i \mathbf{u_i}}{\sqrt{1+\varepsilon^2}}) > (\sigma_1 + \varepsilon \sigma_i \delta)\left(1 - \tfrac{\varepsilon^2}{2}\right) > \sigma_1 - \tfrac{\varepsilon^2}{2}\sigma_1 + \tfrac{\varepsilon}{2}\sigma_i\delta > \sigma_1,$$

for $0 < \varepsilon < \frac{\sigma_i \delta}{\sigma_1}$, a contradiction. Thus $\mathbf{u_1} \cdot \mathbf{u_i} = 0$ for $i \geq 2$.

The proof for other $\mathbf{u_i}, \mathbf{u_j}, j > i > 1$ is similar. Suppose without loss of generality that $\mathbf{u_i}^T\mathbf{u_j} > 0$.

$$A\left(\frac{\mathbf{v}_i + \varepsilon \mathbf{v_j}}{|\mathbf{v_i} + \varepsilon \mathbf{v_j}|}\right) = \frac{\sigma_i \mathbf{u_i} + \varepsilon \sigma_j \mathbf{u_j}}{\sqrt{1+\varepsilon^2}}$$

has length at least as large as its component along $\mathbf{u_i}$ which is

$$\mathbf{u_i^T}(\frac{\sigma_1 \mathbf{u_i} + \varepsilon \sigma_j \mathbf{u_j}}{\sqrt{1+\varepsilon^2}}) > \left(\sigma_i + \varepsilon \sigma_j \mathbf{u_i^T u_j}\right)\left(1 - \tfrac{\varepsilon^2}{2}\right) > \sigma_i - \tfrac{\varepsilon^2}{2}\sigma_i + \tfrac{\varepsilon}{2}\sigma_j \mathbf{u_i^T u_j} > \sigma_i,$$

$\varepsilon < \frac{\sigma_j \mathbf{u_i^T u_j}}{\sigma_i}$, a contradiction since $\mathbf{v_i} + \varepsilon \mathbf{v_j}$ is orthogonal to $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{i-1}}$ and σ_i is the maximum of $|A\mathbf{v}|$ over such vectors. ■

Next we prove Theorem (3.9). We first show that the square of the 2-norm of $A - A_k$ is the square of the $(k+1)^{st}$ singular value of A. This is essentially by definition of A_k; that is, A_k represents the projections of the points in A onto the space spanned by the top k singular vectors, and so $A - A_k$ is the remaining portion of those points, whose top singular value will be σ_{k+1}. Formally, we have

Lemma 3.11 $\|A - A_k\|_2^2 = \sigma_{k+1}^2$.

Proof: Let $A = \sum\limits_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ be the singular value decomposition of A. Then $A_k = \sum\limits_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ and $A - A_k = \sum\limits_{i=k+1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$. Let $\mathbf{v}$ be the top singular vector of $A - A_k$. Express $\mathbf{v}$ as a linear combination of $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$. That is, write $\mathbf{v} = \sum\limits_{i=1}^{r} \alpha_i \mathbf{v_i}$. Then

$$
\begin{aligned}
|(A - A_k)\mathbf{v}| &= \left| \sum_{i=k+1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \sum_{j=1}^{r} \alpha_j \mathbf{v_j} \right| = \left| \sum_{i=k+1}^{r} \alpha_i \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{v_i} \right| \\
&= \left| \sum_{i=k+1}^{r} \alpha_i \sigma_i \mathbf{u_i} \right| = \sqrt{\sum_{i=k+1}^{r} \alpha_i^2 \sigma_i^2},
\end{aligned}
$$

since the $\mathbf{u_i}$ are orthonormal. The $\mathbf{v}$ maximizing this last quantity, subject to the constraint that $|\mathbf{v}|^2 = \sum\limits_{i=1}^{r} \alpha_i^2 = 1$, occurs when $\alpha_{k+1} = 1$ and the rest of the α_i are 0. Thus, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$ proving the lemma. ∎

Finally, we prove Theorem 3.9 which states that A_k is the best rank k, 2-norm approximation to A:

Proof: If A is of rank k or less, the theorem is obviously true since $\|A - A_k\|_2 = 0$. Assume that A is of rank greater than k. By Lemma 3.11, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$. The null space of B, the set of vectors $\mathbf{v}$ such that $B\mathbf{v} = 0$, has dimension at least $d - k$. Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}$ be the first $k + 1$ singular vectors of A. By a dimension argument, it follows that there exists a $\mathbf{z} \neq 0$ in

$$\text{Null}\,(B) \cap \text{Span}\,\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\}.$$

Scale $\mathbf{z}$ to be of length one.

$$\|A - B\|_2^2 \geq |(A - B)\,\mathbf{z}|^2.$$

Since $B\mathbf{z} = 0$,

$$\|A - B\|_2^2 \geq |A\mathbf{z}|^2.$$

Since $\mathbf{z}$ is in the Span $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{k+1}}\}$

$$|A\mathbf{z}|^2 = \left| \sum_{i=1}^{n} \sigma_i \mathbf{u_i} \mathbf{v_i}^T \mathbf{z} \right|^2 = \sum_{i=1}^{n} \sigma_i^2 \left(\mathbf{v_i}^T \mathbf{z}\right)^2 = \sum_{i=1}^{k+1} \sigma_i^2 \left(\mathbf{v_i}^T \mathbf{z}\right)^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} \left(\mathbf{v_i}^T \mathbf{z}\right)^2 = \sigma_{k+1}^2.$$

It follows that $\|A - B\|_2^2 \geq \sigma_{k+1}^2$ and the theorem is proved. ∎

We now prove the analog of eigenvectors and eigenvalues for singular values and vectors we discussed in the introduction.

Lemma 3.12 (Analog of eigenvalues and eigenvectors)

$$A\mathbf{v_i} = \sigma_i(A)\mathbf{u_i} \text{ and } A^T\mathbf{u_i} = \sigma_i(A)\mathbf{v_i}.$$

Proof: The first equation is already known. For the second, note that from SVD, we get $A^T\mathbf{u_i} = \sum_j \sigma_j(A)\mathbf{v_j}\mathbf{u_j}^T\mathbf{u_i}$, where, all terms except when $i = j$ are zero since the $\mathbf{u^{(j)}}$ are orthonormal.

3.8 Power Method for Computing the Singular Value Decomposition

Computing the singular value decomposition is an important branch of numerical analysis in which there have been many sophisticated developments over a long period of time. Here we present an "in-principle" method to establish that the approximate SVD of a matrix A can be computed in polynomial time. The reader is referred to numerical analysis texts for more details. The method we present, called the *power method*, is simple and is in fact the conceptual starting point for many algorithms. Let A be a matrix whose SVD is $\sum_i \sigma_i \mathbf{u_i}\mathbf{v_i}^T$. We wish to work with a matrix that is square and symmetric. Let $B = A^TA$. By direct multiplication, using the orthogonality of the $\mathbf{u_i}$'s that was proved in Theorem 3.10,

$$\begin{aligned} B = A^TA &= \left(\sum_i \sigma_i \mathbf{v_i}\mathbf{u_i}^T\right)\left(\sum_j \sigma_j \mathbf{u_j}\mathbf{v_j}^T\right) \\ &= \sum_{i,j} \sigma_i\sigma_j \mathbf{v_i}(\mathbf{u}_i^T \cdot \mathbf{u_j})\mathbf{v_j}^T = \sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T, \end{aligned}$$

The matrix B is square and symmetric, and has the same left and right-singular vectors. In particular, $B\mathbf{v_j} = (\sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T)\mathbf{v_j} = \sigma_j^2\mathbf{v_j}$, so $\mathbf{v_j}$ is an eigenvector of B with eigenvalue of σ_j^2. If A is itself square and symmetric, it will have the same right and left-singular vectors, namely $A = \sum\limits_i \sigma_i \mathbf{v_i}\mathbf{v_i}^T$ and computing B is unnecessary.

Now consider computing B^2.

$$B^2 = \left(\sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T\right)\left(\sum_j \sigma_j^2 \mathbf{v_j}\mathbf{v_j}^T\right) = \sum_{ij} \sigma_i^2\sigma_j^2 \mathbf{v_i}(\mathbf{v_i}^T\mathbf{v_j})\mathbf{v_j}^T$$

When $i \neq j$, the dot product $\mathbf{v_i}^T\mathbf{v_j}$ is zero by orthogonality.[5] Thus, $B^2 = \sum\limits_{i=1}^{r} \sigma_i^4 \mathbf{v_i}\mathbf{v_i}^T$. In computing the k^{th} power of B, all the cross product terms are zero and

$$B^k = \sum_{i=1}^{r} \sigma_i^{2k} \mathbf{v_i}\mathbf{v_i}^T.$$

[5] The "outer product" $\mathbf{v_i}\mathbf{v_j}^T$ is a matrix and is not zero even for $i \neq j$.

If $\sigma_1 > \sigma_2$, then the first term in the summation dominates, so $B^k \to \sigma_1^{2k}\mathbf{v_1}\mathbf{v_1}^T$. This means a close estimate to $\mathbf{v_1}$ can be computes by simply taking the first column of B^k and normalizing it to a unit vector.

3.8.1 A faster method

A problem with the above method is that A may be a very large, sparse matrix, say a $10^8 \times 10^8$ matrix with 10^9 nonzero entries. Sparse matrices are often represented by just a list of non-zero entries, say, a list of triples of the form (i, j, a_{ij}). Though A is sparse, B need not be and in the worse case may have all 10^{16} entries non-zero,[6] and it is then impossible to even write down B, let alone compute the product B^2. Even if A is moderate in size, computing matrix products is costly in time. Thus, a more efficient method is needed.

Instead of computing B^k, select a random vector $\mathbf{x}$ and compute the product $B^k\mathbf{x}$. The vector $\mathbf{x}$ can be expressed in terms of the singular vectors of B augmented to a full orthonormal basis as $\mathbf{x} = \sum c_i\mathbf{v_i}$. Then

$$B^k\mathbf{x} \approx (\sigma_1^{2k}\mathbf{v_1}\mathbf{v_1}^T)\Big(\sum_{i=1}^{d} c_i\mathbf{v_i}\Big) = \sigma_1^{2k}c_1\mathbf{v_1}$$

Normalizing the resulting vector yields $\mathbf{v_1}$, the first singular vector of A. The way $B^k\mathbf{x}$ is computed is by a series of matrix vector products, instead of matrix products. $B^k\mathbf{x} = A^TA\ldots A^TA\mathbf{x}$, which can be computed right-to-left. This consists of $2k$ vector times sparse matrix multiplications.

An issue occurs if there is no significant gap between the first and second singular values of a matrix. Take for example the case when there is a tie for the first singular vector and $\sigma_1 = \sigma_2$. Then, the argument above fails. We will overcome this hurdle. Theorem 3.13 below states that even with ties, the power method converges to some vector in the span of those singular vectors corresponding to the "nearly highest" singular values. The theorem needs a vector $\mathbf{x}$ which has a component of at least δ along the first right singular vector $\mathbf{v_1}$ of A. We will see in Lemma 3.14 that a random vector satisfies this condition.

Theorem 3.13 *Let A be an $n{\times}d$ matrix and $\mathbf{x}$ a unit length vector in $\mathbf{R}^d$ with $|\mathbf{x}^T\mathbf{v_1}| \geq \delta$, where, $\delta > 0$. Let V be the space spanned by the right singular vectors of A corresponding to singular values greater than $(1-\varepsilon)\,\sigma_1$. Let $\mathbf{w}$ be unit vector after $k = \frac{\ln(1/\varepsilon\delta)}{2\varepsilon}$ iterations of the power method, namely,*

$$\mathbf{w} = \frac{\left(A^TA\right)^k\mathbf{x}}{\left|\left(A^TA\right)^k\mathbf{x}\right|}.$$

Then $\mathbf{w}$ has a component of at most ε perpendicular to V.

[6] E.g., suppose each entry in the first row of A is non-zero and the rest of A is zero.

Proof: Let

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

be the SVD of A. If the rank of A is less than d, then for convenience complete $\{\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v}_r\}$ into an orthonormal basis $\{\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v}_d\}$ of d-space. Write $\mathbf{x}$ in the basis of the $\mathbf{v_i}$'s as

$$\mathbf{x} = \sum_{i=1}^{d} c_i \mathbf{v_i}.$$

Since $(A^TA)^k = \sum_{i=1}^{d} \sigma_i^{2k} \mathbf{v_i} \mathbf{v_i}^T$, it follows that $(A^TA)^k \mathbf{x} = \sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{v_i}$. By hypothesis, $|c_1| \geq \delta$.

Suppose that $\sigma_1, \sigma_2, \ldots, \sigma_m$ are the singular values of A that are greater than or equal to $(1-\varepsilon)\,\sigma_1$ and that $\sigma_{m+1}, \ldots, \sigma_d$ are the singular values that are less than $(1-\varepsilon)\,\sigma_1$. Now

$$|(A^TA)^k \mathbf{x}|^2 = \left| \sum_{i=1}^{d} \sigma_i^{2k} c_i \mathbf{v_i} \right|^2 = \sum_{i=1}^{d} \sigma_i^{4k} c_i^2 \geq \sigma_1^{4k} c_1^2 \geq \sigma_1^{4k} \delta^2.$$

The component of $|(A^TA)^k \mathbf{x}|^2$ perpendicular to the space V is

$$\sum_{i=m+1}^{d} \sigma_i^{4k} c_i^2 \leq (1-\varepsilon)^{4k} \sigma_1^{4k} \sum_{i=m+1}^{d} c_i^2 \leq (1-\varepsilon)^{4k} \sigma_1^{4k}$$

since $\sum_{i=1}^{d} c_i^2 = |\mathbf{x}| = 1$. Thus, the component of $\mathbf{w}$ perpendicular to V has squared length at most $\frac{(1-\varepsilon)^{4k} \sigma_1^{4k}}{\sigma_1^{4k} \delta^2}$ and so its length is at most

$$\frac{(1-\varepsilon)^{2k} \sigma_1^{2k}}{\delta \sigma_1^{2k}} = \frac{(1-\varepsilon)^{2k}}{\delta} \leq \frac{e^{-2k\varepsilon}}{\delta} = \varepsilon.$$

■

Lemma 3.14 *Let $\mathbf{y} \in \mathbf{R}^n$ be a random vector with the unit variance spherical Gaussian as its probability density. Let $\mathbf{x} = \mathbf{y}/|\mathbf{y}|$. Let $\mathbf{v}$ be any fixed (not random) unit length vector. Then*

$$Prob(|\mathbf{x}^T \mathbf{v}| \leq \frac{1}{20\sqrt{d}}) \leq \frac{1}{10} + 3e^{-d/64}.$$

Proof: By Theorem (**??**) of Chapter 2 with $c = \sqrt{d}$ substituted in that theorem, the probability that $|\mathbf{y}| \geq 2\sqrt{d}$ is at most $3e^{-d/64}$. Further, $\mathbf{y}^T\mathbf{v}$ is a random variable with the distribution of a unit variance Gaussian with zero mean and so the probability that $|\mathbf{y}^T\mathbf{v}| \leq \frac{1}{10}$ is at most $1/10$. Combining these two facts and using the union bound, establishes the lemma. ■

3.9 Singular Vectors and Eigenvectors

Recall that for a square matrix B, if the vector $\mathbf{x}$ and scalar λ are such that $B\mathbf{x} = \lambda\mathbf{x}$, then $\mathbf{x}$ is an *eigenvector* of B and λ is the corresponding *eigenvalue*. As we saw in Section 3.8, if $B = A^TA$, then the right singular vectors $\mathbf{v_j}$ of A are eigenvectors of B with eigenvalues σ_j^2. The same argument shows that the left singular vectors $\mathbf{u_j}$ of A are eigenvectors of AA^T with eigenvalues σ_j^2.

Notice that $B = A^TA$ has the property that for any vector $\mathbf{x}$ we have $\mathbf{x}^TB\mathbf{x} \geq 0$. This is because $B = \sum_i \sigma_i^2 \mathbf{v_i}\mathbf{v_i}^T$ and for any $\mathbf{x}$ we have $\mathbf{x}^T\mathbf{v_i}\mathbf{v_i}^T\mathbf{x} = (\mathbf{x}^T\mathbf{v_i})^2 \geq 0$. A matrix B with the property that $\mathbf{x}^TB\mathbf{x} \geq 0$ for all $\mathbf{x}$ is called *positive semi-definite*. So, any matrix A^TA is positive semi-definite. In the other direction, any positive semi-definite matrix B can be decomposed into a product A^TA, and so its eigenvalue decomposition can be obtained from the singular value decomposition of A. The interested reader should consult a linear algebra book.

3.10 Applications of Singular Value Decomposition

3.10.1 Principal Component Analysis

The traditional use of SVD is in Principal Component Analysis (PCA). PCA is illustrated by a movie recommendation setting where there are n customers and d movies. Let matrix A with elements a_{ij} represent the amount that customer i likes movie j. One hypothesizes that there are only k underlying basic factors that determine how much a given customer will like a given movie, where k is much smaller than n or d. For example, these could be the amount of comedy, drama, and action, the novelty of the story, etc. Each movie can be described as a k-dimensional vector indicating how much of these basic factors the movie has, and each customer can be described as a k-dimensional vector indicating how important each of these basic factors is to that customer; the dot-product of these two vectors is hypothesized to determine how much that customer will like that movie. In particular, this means that the $n \times d$ matrix A can be expressed as the product of an $n \times k$ matrix U describing the customers and a $k \times d$ matrix V describing the movies. Finding the best rank k approximation A_k by SVD gives such a U and V. One twist is that A may not be exactly equal to UV, in which case $A - UV$ is treated as noise.

In the above setting, A was available fully and we wished to find U and V to identify the basic factors. However, in a case such as movie recommendations, each customer may have seen only a small fraction of the movies, so it may be more natural to assume that we are given just a few elements of A and wish to estimate A. If A was an arbitrary matrix of size $n \times d$, this would require $\Omega(nd)$ pieces of information and cannot be done with a few entries. But again hypothesize that A was a small rank matrix with added noise. If now we also assume that the given entries are randomly drawn according to some known distribution, then there is a possibility that SVD can be used to estimate the whole of A. This area is called collaborative filtering and one of its uses is to recommend movies or to

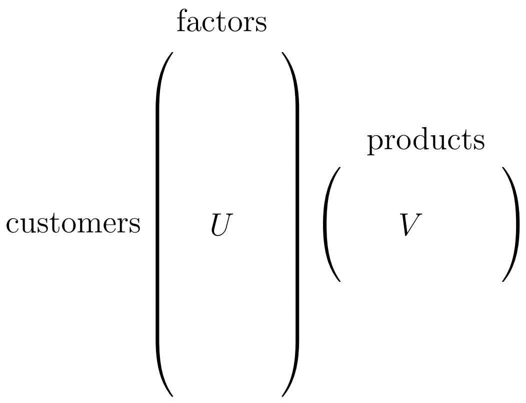


Figure 3.3: Customer-product data

target an ad to a customer based on one or two purchases. We do not describe it here.

3.10.2 Clustering a Mixture of Spherical Gaussians

Clustering is the task of partitioning a set of points in d-space into k subsets or clusters where each cluster consists of "nearby" points. Different definitions of the quality of a clustering lead to different solutions. Clustering is an important area which we will study in detail in Chapter **??**. Here we will see how to solve a particular clustering problem using singular value decomposition.

Mathematical formulations of clustering tend to have the property that finding the highest quality solution to a given set of data is NP-hard. One way around this is to assume stochastic models of input data and devise algorithms to cluster data generated by such models. Mixture models are a very important class of stochastic models. A mixture is a probability density or distribution that is the weighted sum of simple component probability densities. It is of the form

$$F = w_1 p_1 + w_2 p_2 + \cdots + w_k p_k,$$

where $p_1, p_2, \ldots, p_k$ are the basic probability densities and $w_1, w_2, \ldots, w_k$ are positive real numbers called weights that add up to 1. Clearly, F is a probability density and integrates to 1.

The *model fitting problem* is to fit a mixture of k basic densities to n independent, identically distributed samples, each sample drawn according to the same mixture distribution F. The class of basic densities is known, but various parameters such as their means and the component weights of the mixture are not. Here, we deal with the case where the basic densities are all spherical Gaussians. There are two equivalent ways of thinking of the sample generation process (which is hidden; only the samples are given):

1. Pick each sample according to the density F on $\mathbf{R}^d$.
2. Pick a random i from $\{1, 2, \ldots, k\}$ where probability of picking i is w_i. Then, pick a sample according to the density p_i.

One approach to the model-fitting problem is to break it into two subproblems:

1. First, cluster the set of samples into k clusters $C_1, C_2, \ldots, C_k$, where, C_i is the set of samples generated according to p_i (see (2) above) by the hidden generation process.
2. Then, fit a (single) Gaussian distribution to each cluster of sample points.

The second problem is relatively easier and indeed we saw the solution in Chapter (2), where we showed that taking the empirical mean (the mean of the sample) and the empirical standard deviation gives us the best-fit gaussian. The first problem is harder and this is what we discuss here.

If the component Gaussians in the mixture have their centers very close together, then the clustering problem is unresolvable. In the limiting case where a pair of component densities are the same, there is no way to distinguish between them. What condition on the inter-center separation will guarantee unambiguous clustering? First, by looking at 1-dimensional examples, it is clear that this separation should be measured in units of the standard deviation, since the density is a function of the number of standard deviation from the mean. In one dimension, if two Gaussians have inter-center separation at least six times the maximum of their standard deviations, then they hardly overlap. This is summarized in the question: How many standard deviations apart are the means? In one dimension, if the answer is at least six (or a large constant), we can easily tell the Gaussians apart. What is the analog of this in higher dimensions?

We discussed in Chapter (2) distances between two sample points from the same Gaussian as well the distance between two sample points from two different Gaussians. Recall from that discussion that if

- If $\mathbf{x}, \mathbf{y}$ are two independent samples from the same spherical Gaussian with standard deviation[7] σ then
$$|\mathbf{x} - \mathbf{y}|^2 \approx 2(\sqrt{d} \pm O(1))^2\sigma^2.$$
- If $\mathbf{x}, \mathbf{y}$ are samples from different spherical Gaussians each of standard deviation σ and means separated by distance Δ, then
$$|\mathbf{x} - \mathbf{y}|^2 \approx 2(\sqrt{d} \pm O(1))^2\sigma^2 + \Delta^2.$$

Now we would like to assert that points from the same Gaussian are closer to each other than points from different Gaussians. To ensure this, we would need

$$2(\sqrt{d} - O(1))^2\sigma^2 + \Delta^2 > 2(\sqrt{d} + O(1))^2\sigma^2.$$

[7] Since a spherical Gaussian has the same standard deviation in every direction, we call it the standard deviation of the Gaussian.

Expanding the squares, the high order term $2d$ cancels and we need that

$$\Delta > c' d^{1/4},$$

for some constant c'. While this was not a completely rigorous argument, it can be used to show that a distance based clustering approach (see Chapter (2) for an example) requires an inter-mean separation of at least $c' d^{1/4}$ standard deviations to succeed, thus unfortunately not keeping with mnemonic of a constant number of standard deviations separation of the means. Here, indeed, we will show that $\Omega(1)$ standard deviations suffice provided $k \in O(1)$.

The central idea is the following. Suppose we can find the subspace spanned by the k centers and project the sample points to this subspace. The projection of a spherical Gaussian with standard deviation σ remains a spherical Gaussian with standard deviation σ (Lemma 3.15). In the projection, the inter-center separation remains the same. So in the projection, the Gaussians are distinct provided the inter-center separation in the whole space is at least $c' k^{1/4} \sigma$ which is a lot less than $c' d^{1/4} \sigma$ for $k \ll d$. Interestingly, we will see that the subspace spanned by the k-centers is essentially the best-fit k-dimensional subspace that can be found by singular value decomposition.

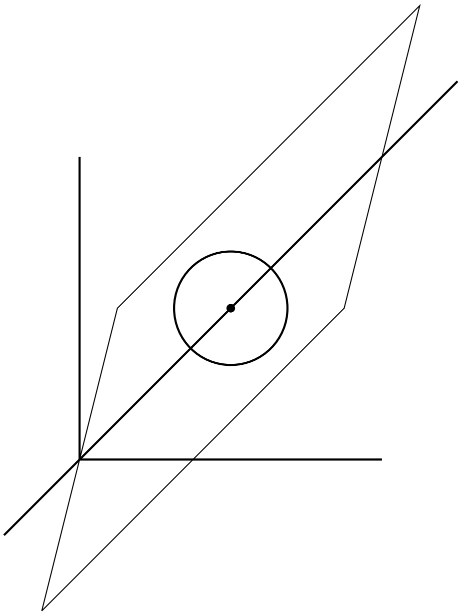
Lemma 3.15 *Suppose p is a d-dimensional spherical Gaussian with center μ and standard deviation σ. The density of p projected onto a k-dimensional subspace V is a spherical Gaussian with the same standard deviation.*

Proof: Rotate the coordinate system so V is spanned by the first k coordinate vectors. The Gaussian remains spherical with standard deviation σ although the coordinates of its center have changed. For a point $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, we will use the notation $\mathbf{x}' = (x_1, x_2, \ldots x_k)$ and $\mathbf{x}'' = (x_{k+1}, x_{k+2}, \ldots, x_n)$. The density of the projected Gaussian at the point $(x_1, x_2, \ldots, x_k)$ is

$$ce^{-\frac{|\mathbf{x}'-\boldsymbol{\mu}'|^2}{2\sigma^2}} \int\limits_{\mathbf{x}''} e^{-\frac{|\mathbf{x}''-\boldsymbol{\mu}''|^2}{2\sigma^2}} d\mathbf{x}'' = c' e^{-\frac{|\mathbf{x}'-\boldsymbol{\mu}'|^2}{2\sigma^2}}.$$

This clearly implies the lemma. ■

We now show that the top k singular vectors produced by the SVD span the space of the k centers. First, we extend the notion of best fit to probability distributions. Then we show that for a single spherical Gaussian whose center is not the origin, the best fit 1-dimensional subspace is the line though the center of the Gaussian and the origin. Next, we show that the best fit k-dimensional subspace for a single Gaussian whose center is not the origin is any k-dimensional subspace containing the line through the Gaussian's center and the origin. Finally, for k spherical Gaussians, the best fit k-dimensional subspace is the subspace containing their centers. Thus, the SVD finds the subspace that contains the centers.



1. The best fit 1-dimension subspace to a spherical Gaussian is the line through its center and the origin.
2. Any k-dimensional subspace containing the line is a best fit k-dimensional subspace for the Gaussian.
3. The best fit k-dimensional subspace for k spherical Gaussians is the subspace containing their centers.

Figure 3.4: Best fit subspace to a spherical Gaussian.

Recall that for a set of points, the best-fit line is the line passing through the origin that maximizes the sum of squared lengths of the projections of the points onto the line. We extend this definition to probability densities instead of a set of points.

Definition 3.1 *If p is a probability density in d space, the best fit line for p is the line $l = \{\lambda \mathbf{v_1} : \lambda \in R\}$ where*

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} \mathop{E}_{\mathbf{x}\sim p}\left[(\mathbf{v}^T\mathbf{x})^2\right].$$

For a spherical Gaussian centered at the origin, it is easy to see that any line passing through the origin is a best fit line. Our next lemma shows that the best fit line for a spherical Gaussian centered at $\boldsymbol{\mu} \neq 0$ is the line passing through $\boldsymbol{\mu}$ and the origin.

Lemma 3.16 *Let the probability density p be a spherical Gaussian with center $\boldsymbol{\mu} \neq 0$. The unique best fit 1-dimensional subspace is the line passing through $\boldsymbol{\mu}$ and the origin. If $\boldsymbol{\mu} = 0$, then any line through the origin is a best-fit line.*

Proof: For a randomly chosen $\boldsymbol{x}$ (according to p) and a fixed unit length vector $\mathbf{v}$,

$$\begin{aligned}
\underset{\mathbf{x}\sim p}{E}\left[(\mathbf{v}^T\mathbf{x})^2\right] &= \underset{\mathbf{x}\sim p}{E}\left[\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)+\mathbf{v}^T\boldsymbol{\mu}\right)^2\right] \\
&= \underset{\mathbf{x}\sim p}{E}\left[\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)^2+2\left(\mathbf{v}^T\boldsymbol{\mu}\right)\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)+\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2\right] \\
&= \underset{\mathbf{x}\sim p}{E}\left[\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)^2\right]+2\left(\mathbf{v}^T\boldsymbol{\mu}\right)E\left[\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right]+\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2 \\
&= \underset{\mathbf{x}\sim p}{E}\left[\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)^2\right]+\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2 \\
&= \sigma^2+\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2
\end{aligned}$$

since $E\left[\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)^2\right]$ is the variance in the direction $\mathbf{v}$ and $E\left(\mathbf{v}^T\left(\mathbf{x}-\boldsymbol{\mu}\right)\right)=0$. The lemma follows from the fact that the best fit line $\mathbf{v}$ is the one that maximizes $\left(\mathbf{v}^T\boldsymbol{\mu}\right)^2$ which is maximized when $\mathbf{v}$ is aligned with the center $\boldsymbol{\mu}$. To see the uniqueness, just note that if $\boldsymbol{\mu}\neq 0$, then $\mathbf{v}^T\boldsymbol{\mu}$ is strictly less when $\mathbf{v}$ is not aligned with the center. ■

We now extend Definition 3.1 to k-dimensional subspaces.

Definition 3.2 *If p is a probability density in d-space then the best-fit k-dimensional subspace V_k is*

$$V_k = \underset{V:dim(V)=k}{\operatorname{argmax}}\ \underset{\mathbf{x}\sim p}{E}\left[|\mathrm{proj}(\mathbf{x},V)|^2\right],$$

where proj$(\mathbf{x},V)$ is the orthogonal projection of $\mathbf{x}$ onto V.

Lemma 3.17 *For a spherical Gaussian with center $\boldsymbol{\mu}$, a k-dimensional subspace is a best fit subspace if and only if it contains $\boldsymbol{\mu}$.*

Proof: If $\boldsymbol{\mu}=\mathbf{0}$, then by symmetry any $k-$dimensional subspace is a best-fit subspace. If $\boldsymbol{\mu}\neq\mathbf{0}$, then, the best-fit line must pass through $\boldsymbol{\mu}$ by Lemma (3.16). Now, as in the greedy algorithm for finding subsequent singular vectors, we would project perpendicular to the first singular vector. But after the projection, the mean of the Gaussian becomes $\mathbf{0}$ and then any vectors will do as subsequent best-fit directions. ■

This leads to the following theorem.

Theorem 3.18 *If p is a mixture of k spherical Gaussians, then the best fit k-dimensional subspace contains the centers. In particular, if the means of the Gaussians are linearly independent, the space spanned by them is the unique best-fit k dimensional subspace.*

Proof: Let p be the mixture $w_1p_1+w_2p_2+\cdots+w_kp_k$. Let V be any subspace of dimension k or less. Then,

$$\underset{\mathbf{x}\sim p}{E}\left[|\mathrm{proj}(\mathbf{x},V)|^2\right]=\sum_{i=1}^{k}w_i\underset{\mathbf{x}\sim p_i}{E}\left[|\mathrm{proj}(\mathbf{x},V)|^2\right]$$

If V contains the centers of the densities p_i, by Lemma 3.17, each term in the summation is individually maximized, which implies the entire summation is maximized, proving the theorem. ■

For an infinite set of points drawn according to the mixture, the k-dimensional SVD subspace gives exactly the space of the centers. In reality, we have only a large number of samples drawn according to the mixture. However, it is intuitively clear that as the number of samples increases, the set of sample points approximates the probability density and so the SVD subspace of the sample is close to the space spanned by the centers. The details of how close it gets as a function of the number of samples are technical and we do not carry this out here.

3.10.3 Singular Vectors and Ranking Documents

An important task for a document collection is to rank the documents according to their intrinsic relevance to the collection. A good candidate is a document's projection onto the best-fit direction for the collection of term-document vectors, namely the top left-singular vector of the term-document matrix. An intuitive reason for this is that this direction has the maximum sum of squared projections of the collection and so can be thought of as a synthetic term-document vector best representing the document collection.

Ranking in order of the projection of each document's term vector along the best fit direction has a nice interpretation in terms of the power method. For this, we consider a different example, that of the web with hypertext links. The World Wide Web can be represented by a directed graph whose nodes correspond to web pages and directed edges to hypertext links between pages. Some web pages, called *authorities*, are the most prominent sources for information on a given topic. Other pages called *hubs*, are ones that identify the authorities on a topic. Authority pages are pointed to by many hub pages and hub pages point to many authorities. One is led to what seems like a circular definition: a hub is a page that points to many authorities and an authority is a page that is pointed to by many hubs.

One would like to assign hub weights and authority weights to each node of the web. If there are n nodes, the hub weights form a n-dimensional vector $\mathbf{u}$ and the authority weights form a n-dimensional vector $\mathbf{v}$. Suppose A is the adjacency matrix representing the directed graph. Here a_{ij} is 1 if there is a hypertext link from page i to page j and 0 otherwise. Given hub vector $\mathbf{u}$, the authority vector $\mathbf{v}$ could be computed by the formula

$$v_j = \sum_{i=1}^{d} u_i a_{ij}$$

since the right hand side is the sum of the hub weights of all the nodes that point to node j. In matrix terms,

$$\mathbf{v} = A^T\mathbf{u}.$$

[DO WE WANT TO SCALE A SO THAT LARGEST SINGULAR VALUE IS 1? OR RENORMALIZE AFTER COMPUTING THE SUM? OTHERWISE THIS PROCESS WILL PRODUCE LARGER AND LARGER VECTORS] Similarly, given an authority vector $\mathbf{v}$, the hub vector $\mathbf{u}$ could be computed by $\mathbf{u} = A\mathbf{v}$. Of course, at the start, we have neither vector. But the above discussion suggests a power iteration. Start with any $\mathbf{v}$. Set $\mathbf{u} = A\mathbf{v}$; then set $\mathbf{v} = A^T\mathbf{u}$ and repeat the process. We know from the power method that this converges to the left and right-singular vectors. So after sufficiently many iterations, we may use the left vector $\mathbf{u}$ as hub weights vector and project each column of A onto this direction and rank columns (authorities) in order of this projection. But the projections just form the vector $A^T\mathbf{u}$ which equals $\mathbf{v}$. So we can just rank by order of the v_j. This is the basis of an algorithm called the HITS algorithm, which was one of the early proposals for ranking web pages.

A different ranking called *pagerank* is widely used. It is based on a random walk on the graph described above. We will study random walks in detail in Chapter 5.

3.10.4 An Application of SVD to a Discrete Optimization Problem

In clustering a mixture of Gaussians, SVD was used as a dimension reduction technique. It found a k-dimensional subspace (the space of centers) of a d-dimensional space and made the Gaussian clustering problem easier by projecting the data to the subspace. Here, instead of fitting a model to data, we have an optimization problem. Again applying dimension reduction to the data makes the problem easier. The use of SVD to solve discrete optimization problems is a relatively new subject with many applications. We start with an important NP-hard problem, the maximum cut problem for a directed graph $G(V, E)$.

The maximum cut problem is to partition the nodes of an n-node directed graph into two subsets S and $\bar{S}$ so that the number of edges from S to $\bar{S}$ is maximized. Let A be the adjacency matrix of the graph. With each vertex i, associate an indicator variable x_i. The variable x_i will be set to 1 for $i \in S$ and 0 for $i \in \bar{S}$. The vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is unknown and we are trying to find it or equivalently the cut, so as to maximize the number of edges across the cut. The number of edges across the cut is precisely

$$\sum_{i,j} x_i(1 - x_j)a_{ij}.$$

Thus, the maximum cut problem can be posed as the optimization problem

$$\text{Maximize} \sum_{i,j} x_i(1 - x_j)a_{ij} \quad \text{subject to } x_i \in \{0, 1\}.$$

In matrix notation,

$$\sum_{i,j} x_i(1 - x_j)a_{ij} = \mathbf{x}^T A(\mathbf{1} - \mathbf{x}),$$

where **1** denotes the vector of all 1's . So, the problem can be restated as

$$\text{Maximize } \mathbf{x}^T A(\mathbf{1}-\mathbf{x}) \quad \text{subject to } x_i \in \{0,1\}. \tag{3.1}$$

The SVD is used to solve this problem approximately by computing the SVD of A and replacing A by $A_k = \sum\limits_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^T$ in (3.1) to get

$$\text{Maximize } \mathbf{x}^T A_k(\mathbf{1}-\mathbf{x}) \quad \text{subject to } x_i \in \{0,1\}. \tag{3.2}$$

Note that the matrix A_k is no longer a 0-1 adjacency matrix.

We will show that:

1. For each 0-1 vector $\mathbf{x}$, $\mathbf{x}^T A_k(\mathbf{1}-\mathbf{x})$ and $\mathbf{x}^T A(\mathbf{1}-\mathbf{x})$ differ by at most $\frac{n^2}{\sqrt{k+1}}$. Thus, the maxima in (3.1) and (3.2) differ by at most this amount.
2. A near optimal $\mathbf{x}$ for (3.2) can be found by exploiting the low rank of A_k, which by Item 1 is near optimal for (3.1) where near optimal means with additive error of at most $\frac{n^2}{\sqrt{k+1}}$.

First, we prove Item 1. Since $\mathbf{x}$ and $\mathbf{1}-\mathbf{x}$ are 0-1 n-vectors, each has length at most $\sqrt{n}$. By the definition of the 2-norm, $|(A-A_k)(\mathbf{1}-\mathbf{x})| \le \sqrt{n}||A-A_k||_2$. Now since $\mathbf{x}^T(A-A_k)(\mathbf{1}-\mathbf{x})$ is the dot product of the vector $\mathbf{x}$ with the vector $(A-A_k)(\mathbf{1}-\mathbf{x})$,

$$|\mathbf{x}^T(A-A_k)(\mathbf{1}-\mathbf{x})| \le n||A-A_k||_2.$$

By Lemma 3.11, $||A-A_k||_2 = \sigma_{k+1}(A)$. The inequalities,

$$(k+1)\sigma_{k+1}^2 \le \sigma_1^2 + \sigma_2^2 + \cdots \sigma_{k+1}^2 \le ||A||_F^2 = \sum_{i,j} a_{ij}^2 \le n^2$$

imply that $\sigma_{k+1}^2 \le \frac{n^2}{k+1}$ and hence $||A-A_k||_2 \le \frac{n}{\sqrt{k+1}}$ proving Item 1.

Next we focus on Item 2. It is instructive to look at the special case when k=1 and A is approximated by the rank one matrix A_1. An even more special case when the left and right-singular vectors $\mathbf{u}$ and $\mathbf{v}$ are required to be identical is already NP-hard to solve exactly because it subsumes the problem of whether for a set of n integers, $\{a_1, a_2, \ldots, a_n\}$, there is a partition into two subsets whose sums are equal. So, we look for algorithms that solve the maximum cut problem approximately.

For Item 2, we want to maximize $\sum_{i=1}^k \sigma_i(\mathbf{x}^T\mathbf{u_i})(\mathbf{v_i}^T(\mathbf{1}-\mathbf{x}))$ over 0-1 vectors $\mathbf{x}$. A piece of notation will be useful. For any $S \subseteq \{1, 2, \ldots n\}$, write $\mathbf{u_i}(S)$ for the sum of coordinates of the vector $\mathbf{u_i}$ corresponding to elements in the set S and also for $\mathbf{v_i}$. That is, $\mathbf{u}_i(S) = \sum\limits_{j \in S} u_{ij}$. We will maximize $\sum_{i=1}^k \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\bar{S})$ using dynamic programming.

For a subset S of $\{1, 2, \ldots, n\}$, define the $2k$-dimensional vector

$$\mathbf{w}(S) = (\mathbf{u_1}(S), \mathbf{v_1}(\bar{S}), \mathbf{u_2}(S), \mathbf{v_2}(\bar{S}), \ldots, \mathbf{u_k}(S), \mathbf{v_k}(\bar{S})).$$

If we had the list of all such vectors, we could find $\sum_{i=1}^{k} \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\bar{S})$ for each of them and take the maximum. There are 2^n subsets S, but several S could have the same $\mathbf{w}(S)$ and in that case it suffices to list just one of them. Round each coordinate of each $\mathbf{u_i}$ to the nearest integer multiple of $\frac{1}{nk^2}$. Call the rounded vector $\tilde{\mathbf{u}}_\mathbf{i}$. Similarly obtain $\tilde{\mathbf{v}}_\mathbf{i}$. Let $\tilde{\mathbf{w}}(S)$ denote the vector $(\tilde{\mathbf{u}}_\mathbf{1}(S), \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}), \tilde{\mathbf{u}}_\mathbf{2}(S), \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}), \ldots, \tilde{\mathbf{u}}_\mathbf{k}(S), \tilde{\mathbf{v}}_\mathbf{k}(\bar{S}))$. We will construct a list of all possible values of the vector $\tilde{\mathbf{w}}(S)$. Again, if several different S's lead to the same vector $\tilde{\mathbf{w}}(S)$, we will keep only one copy on the list. The list will be constructed by dynamic programming. For the recursive step of dynamic programming, assume we already have a list of all such vectors for $S \subseteq \{1, 2, \ldots, i\}$ and wish to construct the list for $S \subseteq \{1, 2, \ldots, i+1\}$. Each $S \subseteq \{1, 2, \ldots, i\}$ leads to two possible $S' \subseteq \{1, 2, \ldots, i+1\}$, namely, S and $S \cup \{i+1\}$. In the first case, the vector $\tilde{\mathbf{w}}(S') = (\tilde{\mathbf{u}}_\mathbf{1}(S), \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}) + \tilde{v}_{1,i+1}, \tilde{\mathbf{u}}_\mathbf{2}(S), \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}) + \tilde{v}_{2,i+1}, \ldots, ...)$. In the second case, it is $\tilde{\mathbf{w}}(S') = (\tilde{\mathbf{u}}_\mathbf{1}(S) + \tilde{u}_{1,i+1}, \tilde{\mathbf{v}}_\mathbf{1}(\bar{S}), \tilde{\mathbf{u}}_\mathbf{2}(S) + \tilde{u}_{2,i+1}, \tilde{\mathbf{v}}_\mathbf{2}(\bar{S}), \ldots, ...)$ We put in these two vectors for each vector in the previous list. Then, crucially, we prune - i.e., eliminate duplicates.

Assume that k is constant. Now, we show that the error is at most $\frac{n^2}{\sqrt{k+1}}$ as claimed. Since $\mathbf{u_i}, \mathbf{v_i}$ are unit length vectors, $|\mathbf{u_i}(S)|, |\mathbf{v_i}(\bar{S})| \leq \sqrt{n}$. Also $|\tilde{\mathbf{u}}_\mathbf{i}(S) - \mathbf{u_i}(S)| \leq \frac{n}{nk^2} = \frac{1}{k^2}$ and similarly for $\mathbf{v_i}$. To bound the error, we use an elementary fact: if a, b are reals with $|a|, |b| \leq M$ and we estimate a by a' and b by b' so that $|a - a'|, |b - b'| \leq \delta \leq M$, then $a'b'$ is an estimate of ab in the sense

$$|ab - a'b'| = |a(b - b') + b'(a - a')| \leq |a||b - b'| + (|b| + |b - b'|)|a - a'| \leq 3M\delta.$$

Using this, we get that

$$\left| \sum_{i=1}^{k} \sigma_i \tilde{\mathbf{u}}_\mathbf{i}(S)\tilde{\mathbf{v}}_\mathbf{i}(\bar{S}) \quad - \quad \sum_{i=1}^{k} \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(S) \right| \leq 3k\sigma_1\sqrt{n}/k^2 \leq 3n^{3/2}/k \leq n^2/k,$$

and this meets the claimed error bound.

Next, we show that the running time is polynomially bounded. $|\tilde{\mathbf{u}}_\mathbf{i}(S)|, |\tilde{\mathbf{v}}_\mathbf{i}(S)| \leq 2\sqrt{n}$. Since $\tilde{\mathbf{u}}_\mathbf{i}(S), \tilde{\mathbf{v}}_\mathbf{i}(S)$ are all integer multiples of $1/(nk^2)$, there are at most $2/\sqrt{n}k^2$ possible values of $\tilde{\mathbf{u}}_\mathbf{i}(S), \tilde{\mathbf{v}}_\mathbf{i}(S)$ from which it follows that the list of $\tilde{\mathbf{w}}(S)$ never gets larger than $(1/\sqrt{n}k^2)^{2k}$ which for fixed k is polynomially bounded.

We summarize what we have accomplished.

Theorem 3.19 *Given a directed graph $G(V, E)$, a cut of size at least the maximum cut minus $O\left(\frac{n^2}{\sqrt{k}}\right)$ can be computed in polynomial time n for any fixed k.*

It would be quite a surprise to have an algorithm that actually achieves the same accuracy in time polynomial in n and k because this would give an exact max cut in polynomial time.

3.11 Bibliographic Notes

Singular value decomposition is fundamental to numerical analysis and linear algebra. There are many texts on these subjects and the interested reader may want to study these. A good reference is [GvL96]. The material on clustering a mixture of Gaussians in Section 3.10.2 is from [VW02]. Modeling data with a mixture of Gaussians is a standard tool in statistics. Several well-known heuristics like the expectation-minimization algorithm are used to learn (fit) the mixture model to data. Recently, in theoretical computer science, there has been modest progress on provable polynomial-time algorithms for learning mixtures. Some references are [DS07], [AK], [AM05], [MV10]. The application to the discrete optimization problem is from [FK99]. The section on ranking documents/webpages is from two influential papers, one on hubs and authorities by Jon Kleinberg [Kle99] and the other on pagerank by Page, Brin, Motwani and Winograd [BMPW98].

3.12 Exercises

Exercise 3.1 (Best fit functions versus best least squares fit) *In many experiments one collects the value of a parameter at various instances of time. Let y_i be the value of the parameter y at time x_i. Suppose we wish to construct the best linear approximation to the data in the sense that we wish to minimize the mean square error. Here error is measured vertically rather than perpendicular to the line. Develop formulas for m and b to minimize the mean square error of the points $\{(x_i, y_i)\,|1 \le i \le n\}$ to the line $y = mx + b$.*

Exercise 3.2 *Given five observed parameters, height, weight, age, income, and blood pressure of n people, how would one find the best least squares fit subspace of the form*

$$a_1\,(\textit{height}) + a_2\,(\textit{weight}) + a_3\,(\textit{age}) + a_4\,(\textit{income}) + a_5\,(\textit{blood pressure}) = 0$$

Here $a_1, a_2, \ldots, a_5$ are the unknown parameters. If there is a good best fit 4-dimensional subspace, then one can think of the points as lying close to a 4-dimensional sheet rather than points lying in 5-dimensions. Why is it better to use the perpendicular distance to the subspace rather than vertical distance where vertical distance to the subspace is measured along the coordinate axis corresponding to one of the unknowns?

Exercise 3.3 *Find the best fit lines through the points in the sets below. Subtract the center of gravity of the points in the set from each of the points in the set and find the best fit line for the resulting points. Does the best fit line go through the origin?*

1. *(4,4) (6,2)*
2. *(4,2) (4,4) (6,2) (6,4)*
3. *(3,2) (3,5) (5,1) (5,4)*

Exercise 3.4 *What is the best fit line through the origin for each of the following set of points? Is the best fit line unique? Justify your answer for each of the subproblems.*

1. $\{(0,1)\,, (1,0)\}$
2. $\{(0,1)\,, (2,0)\}$
3. *The rows of the matrix*

$$\begin{pmatrix} 17 & 4 \\ -2 & 26 \\ 11 & 7 \end{pmatrix}$$

Exercise 3.5 *Find the left and right-singular vectors, the singular values, and the SVD decomposition of the matrices in Figure 3.5.*

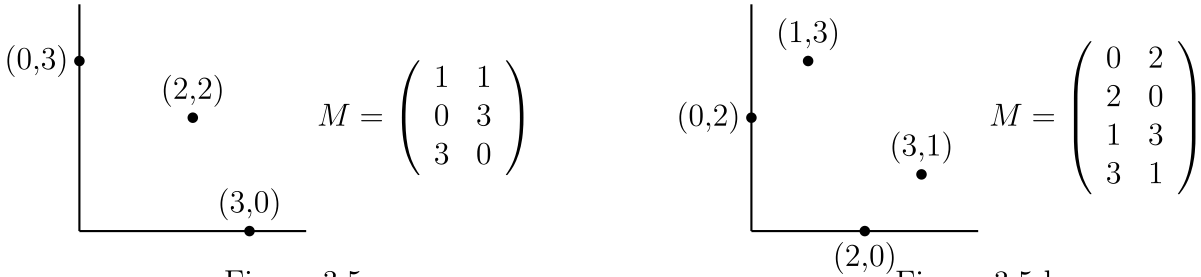

(0,3)
(2,2)
(3,0)
$M = \begin{pmatrix} 1 & 1 \\ 0 & 3 \\ 3 & 0 \end{pmatrix}$
(1,3)
(0,2)
(3,1)
(2,0)
$M = \begin{pmatrix} 0 & 2 \\ 2 & 0 \\ 1 & 3 \\ 3 & 1 \end{pmatrix}$

Figure 3.5 a

Figure 3.5 b

Figure 3.5: SVD problem

Exercise 3.6 *Consider the matrix*

$$A = \begin{bmatrix} 1 & 2 \\ -1 & 2 \\ 1 & -2 \\ -1 & -2 \end{bmatrix}$$

1. *Run the power method starting from $x = \binom{1}{1}$ for $k = 3$ steps. What does this give as an estimate of v_1?*
2. *What actually are the v_i's, σ_i's, and u_i's? It may be easiest to do this by computing the eigenvectors of $B = A^T A$.*
3. *Suppose matrix A is a database of restaurant ratings: each row is a person, each column is a restaurant, and a_{ij} represents how much person i likes restaurant j. What might v_1 represent? What about u_1? How about the gap $\sigma_1 - \sigma_2$?*

Exercise 3.7 *Let A be a square $n \times n$ matrix whose rows are orthonormal. Prove that the columns of A are orthonormal.*

Exercise 3.8 *Suppose A is a $n \times n$ matrix with block diagonal structure with k equal size blocks where all entries of the i^{th} block are a_i with $a_1 > a_2 > \cdots > a_k > 0$. Show that A has exactly k nonzero singular vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ where $\mathbf{v_i}$ has the value $(\frac{k}{n})^{1/2}$ in the coordinates corresponding to the i^{th} block and 0 elsewhere. In other words, the singular vectors exactly identify the blocks of the diagonal. What happens if $a_1 = a_2 = \cdots = a_k$? In the case where the a_i are equal, what is the structure of the set of all possible singular vectors?*
Hint: *By symmetry, the top singular vector's components must be constant in each block.*

Exercise 3.9 *Interpret the first right and left-singular vectors for the document term matrix.*

Exercise 3.10 *Verify that the sum of r-rank one matrices* $\sum_{i=1}^{r} c_i \mathbf{x_i} \mathbf{y_i}^T$ *can be written as* XCY^T*, where the* $\mathbf{x_i}$ *are the columns of* X*, the* $\mathbf{y_i}$ *are the columns of* Y*, and* C *is a diagonal matrix with the constants* c_i *on the diagonal.*

Exercise 3.11 *Let* $\sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$ *be the SVD of* A*. Show that* $\left|\mathbf{u_1^T} A\right| = \max_{|\mathbf{u}|=1} \left|\mathbf{u^T} A\right|$ *equals* σ_1*.*

Exercise 3.12 *If* $\sigma_1, \sigma_2, \ldots, \sigma_r$ *are the singular values of* A *and* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ *are the corresponding right-singular vectors, show that*

1. $A^T A = \sum_{i=1}^{r} \sigma_i^2 \mathbf{v_i} \mathbf{v_i}^T$
2. $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_r$ *are eigenvectors of* $A^T A$*.*
3. *Assuming that the set of eigenvectors of a matrix is unique, conclude that the set of singular values of the matrix is unique.*

See the appendix for the definition of eigenvectors.

Exercise 3.13 *Let* $\sum_i \sigma_i u_i v_i^T$ *be the singular value decomposition of a rank* r *matrix* A*. Let* $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ *be a rank* k *approximation to* A*. Express the following quantities in terms of the singular values* $\{\sigma_i, 1 \le i \le r\}$*.*

1. $||A_k||_F^2$
2. $||A_k||_2^2$
3. $||A - A_k||_F^2$
4. $||A - A_k||_2^2$

Exercise 3.14 *If* A *is a symmetric matrix with distinct singular values, show that the left and right singular vectors are the same and that* $A = VDV^T$*.*

Exercise 3.15 *Let* A *be a matrix. How would you compute*

$$\mathbf{v_1} = \arg\max_{|\mathbf{v}|=1} |A\mathbf{v}|?$$

How would you use or modify your algorithm for finding $\mathbf{v_1}$ *to compute the first few singular vectors of* A*.*

Exercise 3.16 *Compute the singular valued decomposition of the matrix*

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Exercise 3.17 *Write a program to implement the power method for computing the first singular vector of a matrix. Apply your program to the matrix*

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & 9 & 10 \\ 2 & 3 & 4 & \cdots & 10 & 0 \\ \vdots & \vdots & \vdots & & & \vdots \\ 9 & 10 & 0 & \cdots & 0 & 0 \\ 10 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Exercise 3.18 *Modify the power method to find the first four singular vectors of a matrix A as follows. Randomly select four vectors and find an orthonormal basis for the space spanned by the four vectors. Then multiple each of the basis vectors times A and find a new orthonormal basis for the space spanned by the resulting four vectors. Apply your method to find the first four singular vectors of matrix A of Exercise 3.17*

Exercise 3.19 *Let A be a real valued matrix. Prove that $B = AA^T$ is positive semi definite. A matrix B is positive semi definite if for all $\mathbf{x}$, $\mathbf{x}^T B\mathbf{x} \geq 0$.*

Exercise 3.20 *Let A be the adjacency matrix of a graph. The Laplacian of A is $L = D - A$ where D is a diagonal matrix whose diagonal entries are the row sums of A. Prove that L is positive semi definite. A matrix L is positive semi definite if for all $\mathbf{x}$, $\mathbf{x}^T L\mathbf{x} \geq 0$.*

Exercise 3.21 *Prove that the eigenvalues of a symmetric real valued matrix are real.*

Exercise 3.22 *Suppose A is a square invertible matrix and the SVD of A is $A = \sum\limits_i \sigma_i u_i v_i^T$. Prove that the inverse of A is $\sum\limits_i \frac{1}{\sigma_i} v_i u_i^T$.*

Exercise 3.23 *Suppose A is square, but not necessarily invertible and has SVD $A = \sum\limits_{i=1}^{r} \sigma_i u_i v_i^T$. Let $B = \sum\limits_{i=1}^{r} \frac{1}{\sigma_i} v_i u_i^T$. Show that $BA\mathbf{x} = \mathbf{x}$ for all $\mathbf{x}$ in the span of the right-singular vectors of A. For this reason B is sometimes called the pseudo inverse of A and can play the role of A^{-1} in many applications.*

Exercise 3.24

1. *For any matrix A, show that $\sigma_k \leq \frac{||A||_F}{\sqrt{k}}$.*

2. *Prove that there exists a matrix B of rank at most k such that $||A - B||_2 \leq \frac{||A||_F}{\sqrt{k}}$.*

3. *Can the 2-norm on the left hand side in (b) be replaced by Frobenius norm?*

Exercise 3.25 *Suppose an $n \times d$ matrix A is given and you are allowed to preprocess A. Then you are given a number of d-dimensional vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ and for each of these vectors you must find the vector $A\mathbf{x_i}$ approximately, in the sense that you must find a vector $\mathbf{u_i}$ satisfying $|\mathbf{u_i} - A\mathbf{x_i}| \leq \varepsilon ||A||_F|\mathbf{x_i}|$. Here $\varepsilon >0$ is a given error bound. Describe an algorithm that accomplishes this in time $O\left(\frac{d+n}{\varepsilon^2}\right)$ per $\mathbf{x_i}$ not counting the preprocessing time.*

Exercise 3.26 ***(Constrained Least Squares Problem using SVD)*** *Given A, $\mathbf{b}$, and m, use the SVD algorithm to find a vector $\mathbf{x}$ with $|\mathbf{x}| < m$ minimizing $|A\mathbf{x} - \mathbf{b}|$. This problem is a learning exercise for the advanced student. For hints/solution consult Golub and van Loan, Chapter 12.*

Exercise 3.27 (Document-Term Matrices): *Suppose we have a $m \times n$ document-term matrix where each row corresponds to a document where the rows have been normalized to length one. Define the "similarity" between two such documents by their dot product.*

1. *Consider a "synthetic" document whose sum of squared similarities with all documents in the matrix is as high as possible. What is this synthetic document and how would you find it?*

2. *How does the synthetic document in (1) differ from the center of gravity?*

3. *Building on (1), given a positive integer k, find a set of k synthetic documents such that the sum of squares of the mk similarities between each document in the matrix and each synthetic document is maximized. To avoid the trivial solution of selecting k copies of the document in (1), require the k synthetic documents to be orthogonal to each other. Relate these synthetic documents to singular vectors.*

4. *Suppose that the documents can be partitioned into k subsets (often called clusters), where documents in the same cluster are similar and documents in different clusters are not very similar. Consider the computational problem of isolating the clusters. This is a hard problem in general. But assume that the terms can also be partitioned into k clusters so that for $i \neq j$, no term in the i^{th} cluster occurs in a document in the j^{th} cluster. If we knew the clusters and arranged the rows and columns in them to be contiguous, then the matrix would be a block-diagonal matrix. Of course the clusters are not known. By a "block" of the document-term matrix, we mean a submatrix with rows corresponding to the i^{th}cluster of documents and columns corresponding to the i^{th}cluster of terms . We can also partition any n vector into blocks. Show that any right-singular vector of the matrix must have the property that each of its blocks is a right-singular vector of the corresponding block of the document-term matrix.*

5. *Suppose now that the singular values of all the blocks are distinct (also across blocks). Show how to solve the clustering problem.*

Hint: *(4) Use the fact that the right-singular vectors must be eigenvectors of A^TA. Show that A^TA is also block-diagonal and use properties of eigenvectors.*

Exercise 3.28 *(Newcomb/Binford) The frequency distribution of first digits in many data sets is not uniform. One might expect the distribution to be scale free in that changing the units of measure should not change the distribution. Determine a distribution where multiplying each number by two does not change the distribution. Hint: Construct a graph with nine vertices where each vertex corresponds to one of the nine first digits. The edge from vertex i to vertex j is labeled with the probability that multiplying a number whose first digit is i by 2 results in a number whose first digit is j. What is the stationary probability of a random walk on this graph?*

Exercise 3.29 *Show that maximizing $\mathbf{x}^T\mathbf{u}\mathbf{u}^T(\mathbf{1}-\mathbf{x})$ subject to $x_i \in \{0,1\}$ is equivalent to partitioning the coordinates of $\mathbf{u}$ into two subsets where the sum of the elements in both subsets are equal.*

Exercise 3.30 *Read in a photo and convert to a matrix. Perform a singular value decomposition of the matrix. Reconstruct the photo using only 10%, 25%, 50% of the singular values.*

1. *Print the reconstructed photo. How good is the quality of the reconstructed photo?*

2. *What percent of the Forbenius norm is captured in each case?*

Hint: *If you use Matlab, the command to read a photo is imread. The types of files that can be read are given by imformats. To print the file use imwrite. Print using jpeg format. To access the file afterwards you may need to add the file extension .jpg. The command imread will read the file in uint8 and you will need to convert to double for the SVD code. Afterwards you will need to convert back to uint8 to write the file. If the photo is a color photo you will get three matrices for the three colors used.*

Exercise 3.31 *Create a set of 100, 100×100 matrices of random numbers between 0 and 1 such that each entry is highly correlated with the adjacency entries. Find the SVD of A. What fraction of the Frobenius norm of A is captured by the top 10 singular vectors? How many singular vectors are required to capture 95% of the Frobenius norm?*

Exercise 3.32 *Create a 100×100 matrix A of random numbers between 0 and 1 such that each entry is highly correlated with the adjacency entries and find the first 10 vectors for a single basis that is reasonably good for all 100 matrices. How does one do this? What fraction of the Frobenius norm of a new matrix is captured by the basis?*

Exercise 3.33 *Show that the running time for the maximum cut algorithm in Section* **??** *can be carried out in time $O(n^3 + poly(n)k^k)$, where poly is some polynomial.*

Exercise 3.34 *Let $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ be n points in d-dimensional space and let X be the $n \times d$ matrix whose rows are the n points. Suppose we know only the matrix D of pairwise distances between points and not the coordinates of the points themselves. The set of points $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ giving rise to the distance matrix D is not unique since any translation, rotation, or reflection of the coordinate system leaves the distances invariant. Fix the origin of the coordinate system so that the centroid of the set of points is at the origin. That is, $\sum_{i=1}^{n} \mathbf{x_i} = 0$.*

1. *Show that the elements of XX^T are given by*

$$\mathbf{x_i}\mathbf{x_j^T} = -\frac{1}{2}\left[d_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 \right].$$

2. *Describe an algorithm for determining the matrix X whose rows are the $\mathbf{x_i}$.*

Exercise 3.35

1. *Consider the pairwise distance matrix for twenty US cities given below. Use the algorithm of Exercise 3.34 to place the cities on a map of the US. The algorithm is called classical multidimensional scaling, cmdscale, in Matlab. Alternatively use the pairwise distance matrix to place the cities on a map of China.*

   Note: Any rotation or a mirror image of the map will have the same pairwise distances.

2. *Suppose you had airline distances for 50 cities around the world. Could you use these distances to construct a world map?*

■

	B	B	C	D	D	H	L	M	M	M
	O	U	H	A	E	O	A	E	I	I
	S	F	I	L	N	U		M	A	M
Boston	-	400	851	1551	1769	1605	2596	1137	1255	1123
Buffalo	400	-	454	1198	1370	1286	2198	803	1181	731
Chicago	851	454	-	803	920	940	1745	482	1188	355
Dallas	1551	1198	803	-	663	225	1240	420	1111	862
Denver	1769	1370	920	663	-	879	831	879	1726	700
Houston	1605	1286	940	225	879	-	1374	484	968	1056
Los Angeles	2596	2198	1745	1240	831	1374	-	1603	2339	1524
Memphis	1137	803	482	420	879	484	1603	-	872	699
Miami	1255	1181	1188	1111	1726	968	2339	872	-	1511
Minneapolis	1123	731	355	862	700	1056	1524	699	1511	-
New York	188	292	713	1374	1631	1420	2451	957	1092	1018
Omaha	1282	883	432	586	488	794	1315	529	1397	290
Philadelphia	271	279	666	1299	1579	1341	2394	881	1019	985
Phoenix	2300	1906	1453	887	586	1017	357	1263	1982	1280
Pittsburgh	483	178	410	1070	1320	1137	2136	660	1010	743
Saint Louis	1038	662	262	547	796	679	1589	240	1061	466
Salt Lake City	2099	1699	1260	999	371	1200	579	1250	2089	987
San Francisco	2699	2300	1858	1483	949	1645	347	1802	2594	1584
Seattle	2493	2117	1737	1681	1021	1891	959	1867	2734	1395
Washington D.C.	393	292	597	1185	1494	1220	2300	765	923	934

	N	O	P	P	P	S	S	S	S	D
	Y	M	H	H	I	t	L	F	E	C
		A	I	O	T	L	C		A	
Boston	188	1282	271	2300	483	1038	2099	2699	2493	393
Buffalo	292	883	279	1906	178	662	1699	2300	2117	292
Chicago	713	432	666	1453	410	262	1260	1858	1737	597
Dallas	1374	586	1299	887	1070	547	999	1483	1681	1185
Denver	1631	488	1579	586	1320	796	371	949	1021	1494
Houston	1420	794	1341	1017	1137	679	1200	1645	1891	1220
Los Angeles	2451	1315	2394	357	2136	1589	579	347	959	2300
Memphis	957	529	881	1263	660	240	1250	1802	1867	765
Miami	1092	1397	1019	1982	1010	1061	2089	2594	2734	923
Minneapolis	1018	290	985	1280	743	466	987	1584	1395	934
New York	-	1144	83	2145	317	875	1972	2571	2408	230
Omaha	1144	-	1094	1036	836	354	833	1429	1369	1014
Philadelphia	83	1094	-	2083	259	811	1925	2523	2380	123
Phoenix	2145	1036	2083	-	1828	1272	504	653	1114	1973
Pittsburgh	317	836	259	1828	-	559	1668	2264	2138	192
Saint Louis	875	354	811	1272	559	-	1162	1744	1724	712
Salt Lake City	1972	833	1925	504	1668	1162	-	600	701	1848
San Francisco	2571	1429	2523	653	2264	1744	600	-	678	2442
Seattle	2408	1369	2380	1114	2138	1724	701	678	-	2329
Washington D.C.	230	1014	123	1973	192	712	1848	2442	2329	-

City	Beijing	Tianjin	Shanghai	Chongqing	Hohhot	Urumqi	Lhasa	Yinchuan	Nanning	Harbin	Changchun	Shenyang
Beijing	0	125	1239	3026	480	3300	3736	1192	2373	1230	979	684
Tianjin	125	0	1150	1954	604	3330	3740	1316	2389	1207	955	661
Shanghai	1239	1150	0	1945	1717	3929	4157	2092	1892	2342	2090	1796
Chongqing	3026	1954	1945	0	1847	3202	2457	1570	993	3156	2905	2610
Hohhot	480	604	1717	1847	0	2825	3260	716	2657	1710	1458	1164
Urumqi	3300	3330	3929	3202	2825	0	2668	2111	4279	4531	4279	3985
Lhasa	3736	3740	4157	2457	3260	2668	0	2547	3431	4967	4715	4421
Yinchuan	1192	1316	2092	1570	716	2111	2547	0	2673	2422	2170	1876
Nanning	2373	2389	1892	993	2657	4279	3431	2673	0	3592	3340	3046
Harbin	1230	1207	2342	3156	1710	4531	4967	2422	3592	0	256	546
Changchun	979	955	2090	2905	1458	4279	4715	2170	3340	256	0	294
Shenyang	684	661	1796	2610	1164	3985	4421	1876	3046	546	294	0

4 Random Graphs

Large graphs appear in many contexts such as the World Wide Web, the internet, social networks, journal citations, and other places. What is different about the modern study of large graphs from traditional graph theory and graph algorithms is that here one seeks statistical properties of these very large graphs rather than an exact answer to questions. This is akin to the switch physics made in the late 19^{th} century in going from mechanics to statistical mechanics. Just as the physicists did, one formulates abstract models of graphs that are not completely realistic in every situation, but admit a nice mathematical development that can guide what happens in practical situations. Perhaps the most basic such model is the $G\,(n,p)$ model of a random graph. In this chapter, we study properties of the $G(n,p)$ model as well as other models.

4.1 The $G(n,p)$ Model

The $G\,(n,p)$ model, due to Erdös and Rényi, has two parameters, n and p. Here n is the number of vertices of the graph and p is the edge probability. For each pair of distinct vertices, v and w, p is the probability that the edge (v,w) is present. The presence of each edge is statistically independent of all other edges. The graph-valued random variable with these parameters is denoted by $G\,(n,p)$. When we refer to "the graph $G\,(n,p)$", we mean one realization of the random variable. In many cases, p will be a function of n such as $p = d/n$ for some constant d. In this case, the expected degree of a vertex of the graph is $\frac{d}{n}(n-1) \approx d$. The interesting thing about the $G(n,p)$ model is that even though edges are chosen independently with no "collusion", certain global properties of the graph emerge from the independent choices. For small p, with $p = d/n$, $d < 1$, each connected component in the graph is small. For $d > 1$, there is a giant component consisting of a constant fraction of the vertices. In addition, there is a rapid transition at the threshold $d = 1$. Below the threshold, the probability of a giant component is very small, and above the threshold, the probability is almost one.

The phase transition at the threshold $d = 1$ from very small $o(n)$ size components to a giant $\Omega(n)$ sized component is illustrated by the following example. Suppose the vertices represent people and an edge means the two people it connects know each other. Given a chain of connections, such as A knows B, B knows C, C knows D, ..., and Y knows Z, we say that A indirectly knows Z. Thus, all people belonging to a connected component of the graph indirectly know each other. Suppose each pair of people, independent of other pairs, tosses a coin that comes up heads with probability $p = d/n$. If it is heads, they know each other; if it comes up tails, they don't. The value of d can be interpreted as the expected number of people a single person directly knows. The question arises as to how large are sets of people who indirectly know each other ?

If the expected number of people each person knows is more than one, then a giant component of people, all of whom indirectly know each other, will be present consisting

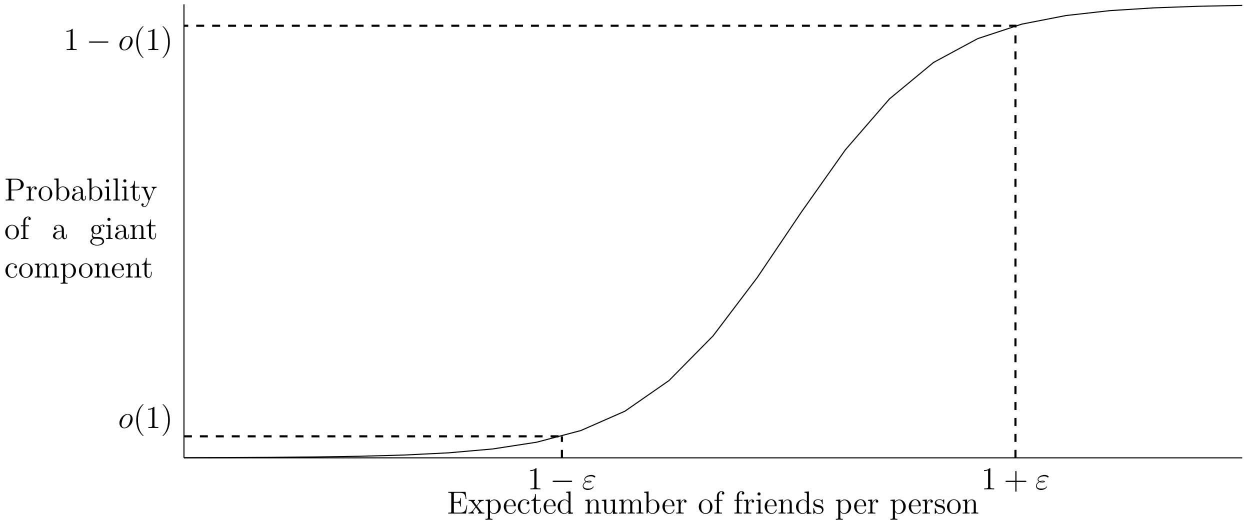


Figure 4.1: Probability of a giant component as a function of the expected number of people each person knows directly.

of a constant fraction of all the people. On the other hand, if in expectation, each person knows less than one person, the largest set of people who know each other indirectly is a vanishingly small fraction of the whole. Furthermore, the transition from the vanishing fraction to a constant fraction of the whole, happens abruptly between d slightly less than one to d slightly more than one. See Figure 4.1. Note that there is no global coordination of who knows whom. Each pair of individuals decides independently. Indeed, many large real-world graphs, with constant average degree, have a giant component. This is perhaps the most important global property of the $G(n, p)$ model.

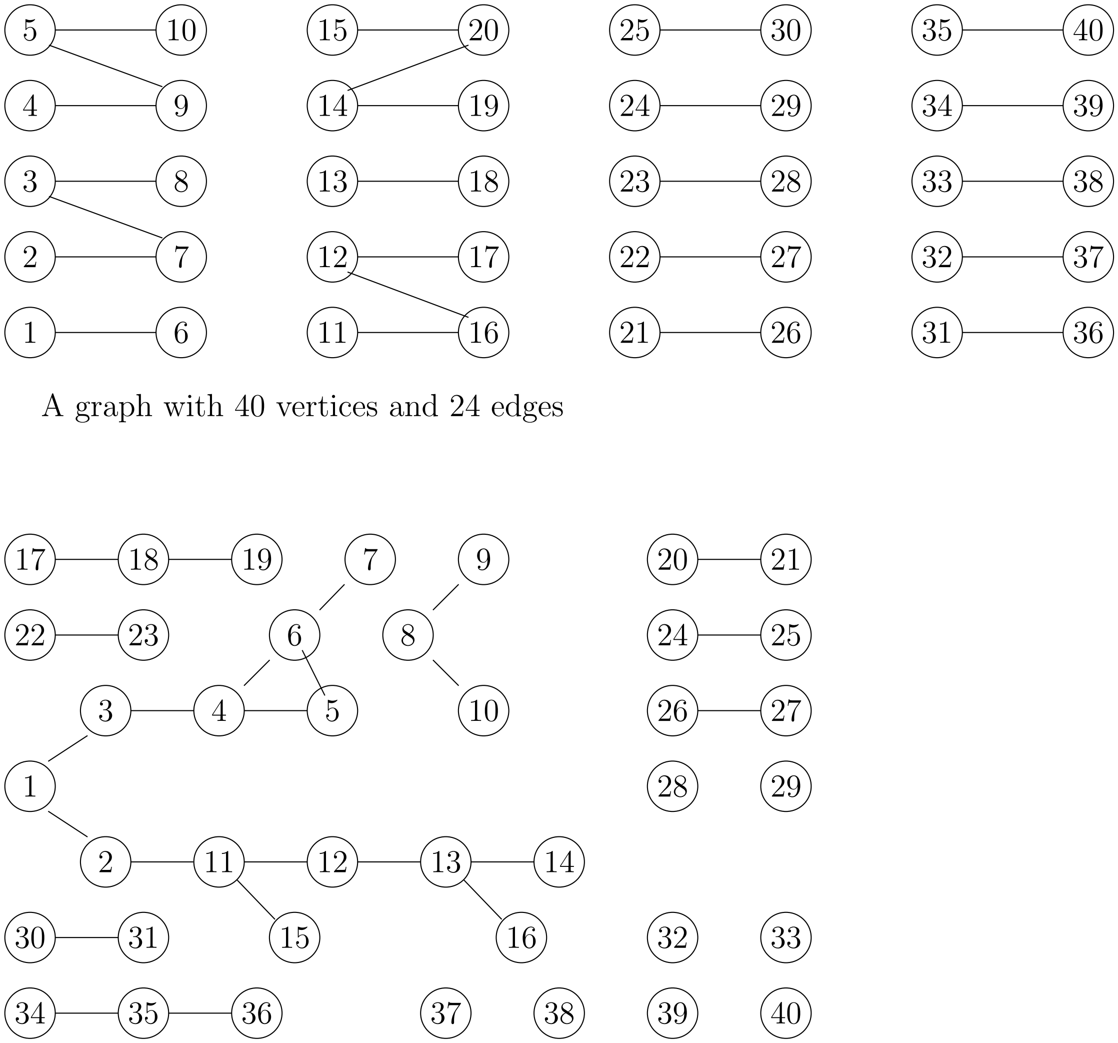
4.1.1 Degree Distribution

One of the simplest quantities to observe in a real graph is the number of vertices of given degree, called the vertex degree distribution. It is also very simple to study these distributions in $G\left(n, p\right)$ since the degree of each vertex is the sum of $n - 1$ independent random variables, which results in a binomial distribution. Since we will be dealing with graphs where the number of vertices n, is large, from here on we often replace $n - 1$ by n to simplify formulas.

Example: In $G(n, \frac{1}{2})$, each vertex is of degree close to $n/2$. In fact, for any $\varepsilon > 0$, the degree of each vertex almost surely is within $1 \pm \varepsilon$ times $n/2$. To see this, note that the probability that a vertex is of degree k is

$$\text{Prob}\left(k\right) = \binom{n-1}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} \approx \binom{n}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} = \frac{1}{2^n} \binom{n}{k}.$$

This probability distribution has a mean $m = n/2$ and variance $\sigma^2 = n/4$. To see this, observe that the degree k is the sum of n indicator variables that take on value zero or



A graph with 40 vertices and 24 edges

A randomly generated $G(n, p)$ graph with 40 vertices and 24 edges

Figure 4.2: Two graphs, each with 40 vertices and 24 edges. The second graph was randomly generated using the $G(n, p)$ model with $p = 1.2/n$. A graph similar to the top graph is almost surely not going to be randomly generated in the $G(n, p)$ model, whereas a graph similar to the lower graph will almost surely occur. Note that the lower graph consists of a giant component along with a number of small components that are trees.

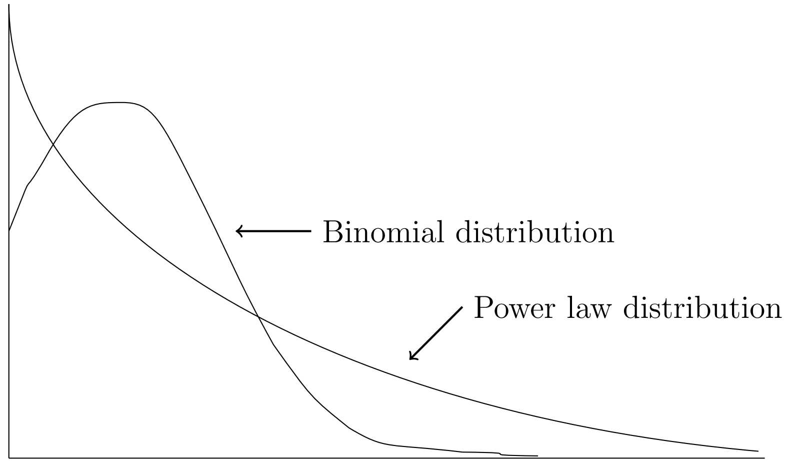


Figure 4.3: Illustration of the binomial and the power law distributions.

one depending whether an edge is present or not. The expected value of the sum is the sum of the expected values and the variance of the sum is the sum of the variances.

Near the mean, the binomial distribution is well approximated by the normal distribution. See Section 12.4.9 in the appendix.

$$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}\frac{(k-m)^2}{\sigma^2}} = \frac{1}{\sqrt{\pi n/2}}e^{-\frac{(k-n/2)^2}{n/2}}$$

The standard deviation of the normal distribution is $\frac{\sqrt{n}}{2}$ and essentially all of the probability mass is within an additive term $\pm c\sqrt{n}$ of the mean $n/2$ for some constant c and thus is certainly within a multiplicative factor of $1 \pm \varepsilon$ of $n/2$ for sufficiently large n. ∎

The degree distribution of $G(n,p)$ for general p is also binomial. Since p is the probability of an edge being present, the expected degree of a vertex is $d \approx pn$. The actual degree distribution is given by

$$\text{Prob(vertex has degree } k) = \binom{n-1}{k}p^k(1-p)^{n-k-1} \approx \binom{n}{k}p^k(1-p)^{n-k}.$$

The quantity $\binom{n-1}{k}$ is the number of ways of choosing k edges, out of the possible $n-1$ edges, and $p^k(1-p)^{n-k-1}$ is the probability that the k selected edges are present and the remaining $n-k-1$ are not. Since n is large, replacing $n-1$ by n does not cause much error.

The binomial distribution falls off exponentially fast as one moves away from the mean. However, the degree distributions of graphs that appear in many applications do not exhibit such sharp drops. Rather, the degree distributions are much broader. This is often referred to as having a "heavy tail". The term tail refers to values of a random variable far away from its mean, usually measured in number of standard deviations. Thus, although the $G(n,p)$ model is important mathematically, more complex models are needed to represent real world graphs.

Consider an airline route graph. The graph has a wide range of degrees, from degree one or two for a small city, to degree 100, or more, for a major hub. The degree distribution is not binomial. Many large graphs that arise in various applications appear to have power law degree distributions. A power law degree distribution is one in which the number of vertices having a given degree decreases as a power of the degree, as in

$$\text{Number(degree } k \text{ vertices)} = c\tfrac{n}{k^r},$$

for some small positive real r, often just slightly less than three. Later, we will consider a random graph model giving rise to such degree distributions.

The following theorem claims that the degree distribution of the random graph $G\left(n,p\right)$ is tightly concentrated about its expected value. That is, the probability that the degree of a vertex differs from its expected degree, np, by more than $\lambda\sqrt{np}$, drops off exponentially fast with λ.

Theorem 4.1 *Let v be a vertex of the random graph $G(n,p)$. Let α be a real number in $(0,\sqrt{np})$.*

$$Prob(|np - deg(v)| \geq \alpha\sqrt{np}) \leq 3e^{-\alpha^2/8}.$$

Proof: The degree deg(v) is the sum of $n-1$ independent Bernoulli random variables, $y_1, y_2, \ldots, y_{n-1}$, where, y_i is the indicator variable that the i^{th} edge from v is present. So the theorem follows from Theorem **??**. ■

Theorem 4.1.1 was for one vertex. The corollary below deals with all vertices.

Corollary 4.2 *Suppose ε is a positive constant. If p is $\Omega(\ln n/n\varepsilon^2)$, then, almost surely, every vertex has degree in the range $(1-\varepsilon)np$ to $(1+\varepsilon)np$.*

Proof: Apply Theorem with $\alpha = \varepsilon\sqrt{np}$ to get that the probability that an individual vertex has degree outside the range $[(1-\varepsilon)np, (1+\varepsilon)np]$ is at most $3e^{-\varepsilon^2 np/8}$. By the union bound, the probability that some vertex has degree outside this range is at most $3ne^{-\varepsilon^2 np/8}$. For this to be $o(1)$, it suffices for p to be $\Omega(\ln n/n\varepsilon^2)$. Hence the Corollary. ■

Note that the assumption p is $\Omega(\ln n/n\varepsilon^2)$ is necessary. If $p = d/n$ for d a constant, then, indeed, some vertices may have degrees outside the range. Without the $\Omega(\ln n/n\varepsilon^2)$ assumption, for $p = \frac{1}{n}$, Corollary 4.1.1 would claim almost surely no vertex had a degree that was greater than a constant independent of n. But shortly we will see that it is highly likely that for $p = \frac{1}{n}$ there is a vertex of degree $\Omega(\log n/\log\log n)$.

When p is a constant, the expected degree of vertices in $G\left(n,p\right)$ increases with n. For example, in $G\left(n,\frac{1}{2}\right)$, the expected degree of a vertex is $n/2$. In many real applications, we will be concerned with $G(n,p)$ where $p = d/n$, for d a constant, i.e., graphs whose expected degree is a constant d independent of n. Holding $d = np$ constant as n goes to infinity, the binomial distribution

$$\text{Prob}\,(k) = \binom{n}{k} p^k \,(1-p)^{n-k}$$

approaches the Poisson distribution

$$\text{Prob}(k) = \frac{(np)^k}{k!} e^{-np} = \frac{d^k}{k!} e^{-d}.$$

To see this, assume $k = o(n)$ and use the approximations $n - k \cong n$, $\binom{n}{k} \cong \frac{n^k}{k!}$, and $\left(1 - \frac{1}{n}\right)^{n-k} \cong e^{-1}$ to approximate the binomial distribution by

$$\lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} = \frac{n^k}{k!} \left(\frac{d}{n}\right)^k (1 - \frac{d}{n})^n = \frac{d^k}{k!} e^{-d}.$$

Note that for $p = \frac{d}{n}$, where d is a constant independent of n, the probability of the binomial distribution falls off rapidly for $k > d$, and is essentially zero for all but some finite number of values of k. This justifies the $k = o(n)$ assumption. Thus, the Poisson distribution is a good approximation.

Example: In $G(n, \frac{1}{n})$ many vertices are of degree one, but not all. Some are of degree zero and some are of degree greater than one. In fact, it is highly likely that there is a vertex of degree $\Omega(\log n/\log\log n)$. The probability that a given vertex is of degree k is

$$\text{Prob}\,(k) = \binom{n}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \approx \frac{e^{-1}}{k!}.$$

If $k = \log n/\log\log n$,

$$\log k^k = k \log k \cong \frac{\log n}{\log\log n} \left(\log\log n - \log\log\log n\right) \cong \log n$$

and thus $k^k \cong n$. Since $k! \leq k^k \cong n$, the probability that a vertex has degree $k = \log n/\log\log n$ is at least $\frac{1}{k!} e^{-1} \geq \frac{1}{en}$. If the degrees of vertices were independent random variables, then this would be enough to argue that there would be a vertex of degree $\log n/\log\log n$ with probability at least $1 - \left(1 - \frac{1}{en}\right)^n = 1 - e^{-\frac{1}{e}} \cong 0.31$. But the degrees are not quite independent since when an edge is added to the graph it affects the degree of two vertices. This is a minor technical point, which one can get around. ∎

4.1.2 Existence of Triangles in $G(n, d/n)$

What is the expected number of triangles in $G\left(n, \frac{d}{n}\right)$, when d is a constant? As the number of vertices increases one might expect the number of triangles to increase, but this is not the case. Although the number of triples of vertices grows as n^3, the probability of an edge between two specific vertices decreases linearly with n. Thus, the probability of all three edges between the pairs of vertices in a triple of vertices being present goes down as n^{-3}, exactly canceling the rate of growth of triples.

A random graph with n vertices and edge probability d/n, has an expected number of triangles that is independent of n, namely $d^3/6$. There are $\binom{n}{3}$ triples of vertices. Each triple has probability $\left(\frac{d}{n}\right)^3$ of being a triangle. Let Δ_{ijk} be the indicator variable for the triangle with vertices i, j, and k being present. That is, all three edges (i, j), (j, k), and (i, k) being present. Then the number of triangles is $x = \sum_{ijk} \Delta_{ijk}$. Even though the existence of the triangles are not statistically independent events, by linearity of expectation, which does not assume independence of the variables, the expected value of a sum of random variables is the sum of the expected values. Thus, the expected number of triangles is

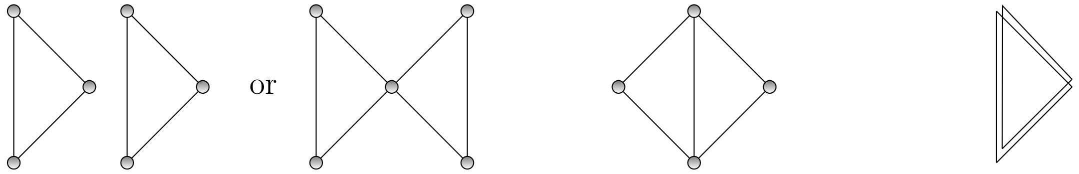
$$E(x) = E\Big(\sum_{ijk} \Delta_{ijk}\Big) = \sum_{ijk} E(\Delta_{ijk}) = \binom{n}{3}\left(\frac{d}{n}\right)^3 \approx \frac{d^3}{6}.$$

Even though on average there are $\frac{d^3}{6}$ triangles per graph, this does not mean that with high probability a graph has a triangle. Maybe half of the graphs have $\frac{d^3}{3}$ triangles and the other half have none for an average of $\frac{d^3}{6}$ triangles. Then, with probability 1/2, a graph selected at random would have no triangle. If $1/n$ of the graphs had $\frac{d^3}{6}n$ triangles and the remaining graphs had no triangles, then as n goes to infinity, the probability that a graph selected at random would have a triangle would go to zero.

We wish to assert that with some nonzero probability there is at least one triangle in $G(n, p)$ when $p = \frac{d}{n}$. If all the triangles were on a small number of graphs, then the number of triangles in those graphs would far exceed the expected value and hence the variance would be high. A second moment argument rules out this scenario where a small fraction of graphs have a large number of triangles and the remaining graphs have none.

Calculate $E(x^2)$ where x is the number of triangles. Write x as $x = \sum_{ijk} \Delta_{ijk}$, where Δ_{ijk} is the indicator variable of the triangle with vertices i, j, and k being present. Expanding the squared term

$$E(x^2) = E\Big(\sum_{i,j,k} \Delta_{ijk}\Big)^2 = E\Big(\sum_{\substack{i,\,j,\,k\\ i',j',k'}} \Delta_{ijk}\Delta_{i'j'k'}\Big).$$



The two triangles of Part 1 are either disjoint or share at most one vertex

The two triangles of Part 2 share an edge

The two triangles in Part 3 are the same triangle

Figure 4.4: The triangles in Part 1, Part 2, and Part 3 of the second moment argument for the existence of triangles in $G(n, \frac{d}{n})$.

Split the above sum into three parts. Split the above sum into three parts. In Part 1, let S_1 be the set of i, j, k and i', j', k' which share at most one vertex and hence the two triangles share no edge. In this case, Δ_{ijk} and $\Delta_{i'j'k'}$ are independent and

$$E\Big(\sum_{S_1}\Delta_{ijk}\Delta_{i'j'k'}\Big) = \sum_{S_1}E(\Delta_{ijk})E(\Delta_{i'j'k'}) \le \Big(\sum_{\substack{\text{all}\\ ijk}}E(\Delta_{ijk})\Big)\Big(\sum_{\substack{\text{all}\\ i'j'k'}}E(\Delta_{i'j'k'})\Big) = E^2(x).$$

In Part 2, i, j, k and i', j', k' share two vertices and hence one edge. See Figure 4.4. Four vertices and five edges are involved overall. There are at most $\binom{n}{4} \in O(n^4)$, 4-vertex subsets and $\binom{4}{2}$ ways to partition the four vertices into two triangles with a common edge. The probability of all five edges in the two triangles being present is p^5, so this part sums to $O(n^4p^5) = O(d^5/n)$ and is $o(1)$. There are so few triangles in the graph, the probability of two triangles sharing an edge is extremely unlikely.

In Part 3, i, j, k and i', j', k' are the same sets. The contribution of this part of the summation to $E(x^2)$ is $\binom{n}{3}p^3 = \frac{d^3}{6}$. Thus,

$$E(x^2) \le E^2(x) + \frac{d^3}{6} + o(1),$$

which implies

$$\text{Var}(x) = E(x^2) - E^2(x) \le \frac{d^3}{6} + o(1).$$

For x to be less than or equal to zero, it must differ from its expected value by at least its expected value. Thus,

$$\text{Prob}(x = 0) \le \text{Prob}\left(|x - E(x)| \ge E(x)\right).$$

By Chebychev inequality,

$$\text{Prob}(x = 0) \le \frac{\text{Var}(x)}{E^2(x)} \le \frac{d^3/6 + o(1)}{d^6/36} \le \frac{6}{d^3} + o(1). \tag{4.1}$$

Thus, for $d > \sqrt[3]{6} \cong 1.8$, Prob$(x = 0) < 1$ and $G(n,p)$ has a triangle with nonzero probability. For $d < \sqrt[3]{6}$ and very close to zero, there simply are not enough edges in the graph for there to be a triangle.

4.2 Phase Transitions

Many properties of random graphs undergo structural changes as the edge probability passes some threshold value. This phenomenon is similar to the abrupt phase transitions in physics, as the temperature or pressure increases. Some examples of this are the abrupt appearance of cycles in $G(n,p)$ when p reaches $1/n$ and the disappearance of isolated vertices when p reaches $\frac{\log n}{n}$. The most important of these transitions is the emergence of a giant component, a connected component of size $\Theta(n)$, which happens at $d = 1$. Recall Figure 4.1.

For these and many other properties of random graphs, a threshold exists where an abrupt transition from not having the property to having the property occurs. If there exists a function $p\left(n\right)$ such that when $\lim\limits_{n\to\infty} \frac{p_1(n)}{p(n)} = 0$, $G\left(n, p_1\left(n\right)\right)$ almost surely does not have the property, and when $\lim\limits_{n\to\infty} \frac{p_2(n)}{p(n)} = \infty$, $G\left(n, p_2\left(n\right)\right)$ almost surely has the property, then we say that a *phase transition* occurs, and $p\left(n\right)$ is the *threshold*. Recall that $G(n,p)$ "almost surely does not have the property" means that the probability that it has the property goes to zero in the limit, as n goes to infinity. We shall soon see that every increasing property has a threshold. This is true not only for increasing properties of $G\left(n,p\right)$, but for increasing properties of any combinatorial structure. If for $cp\left(n\right)$, $c < 1$, the graph almost surely does not have the property and for $cp\left(n\right)$, $c > 1$, the graph almost surely has the property, then $p\left(n\right)$ is a *sharp threshold.* The existence of a giant component has a sharp threshold at $1/n$. We will prove this later.

In establishing phase transitions, we often use a variable $x(n)$ to denote the number of occurrences of an item in a random graph. If the expected value of $x(n)$ goes to zero as n goes to infinity, then a graph picked at random almost surely has no occurrence of the item. This follows from Markov's inequality. Since x is a nonnegative random variable Prob$(x \geq a) \leq \frac{1}{a}E(x)$, which implies that the probability of $x(n) \geq 1$ is at most $E(x(n))$. That is, if the expected number of occurrences of an item in a graph goes to zero, the probability that there are one or more occurrences of the item in a randomly selected graph goes to zero. This is called the *first moment method.*

The previous section showed that the property of having a triangle has a threshold at $p(n) = 1/n$. If the edge probability $p_1(n)$ is $o(1/n)$, then the expected number of triangles goes to zero and by the first moment method, the graph almost surely has no triangle. However, if the edge probability $p_2(n)$ satisfies $np_2(n) \to \infty$, then from (4.1), the probability of having no triangle is at most $6/d^3 + o(1) = 6/(np_2(n))^3 + o(1)$, which goes to zero. This latter case uses what we call the second moment method. The first

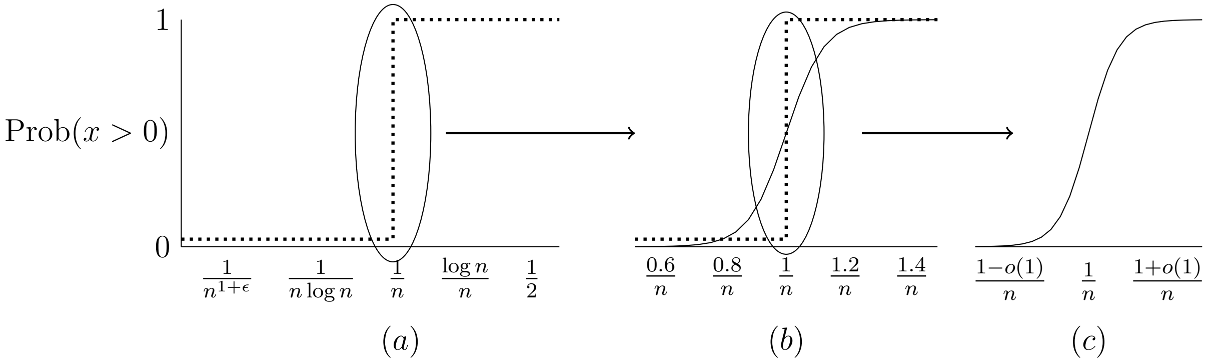


Figure 4.5: Figure 4.5(a) shows a phase transition at $p = \frac{1}{n}$. The dotted line shows an abrupt transition in Prob(x) from 0 to 1. For any function asymptotically less than $\frac{1}{n}$, Prob(x)>0 is zero and for any function asymptotically greater than $\frac{1}{n}$, Prob(x)>0 is one. Figure 4.5(b) expands the scale and shows a less abrupt change in probability unless the phase transition is sharp as illustrated by the dotted line. Figure 4.5(c) is a further expansion and the sharp transition is now more smooth.

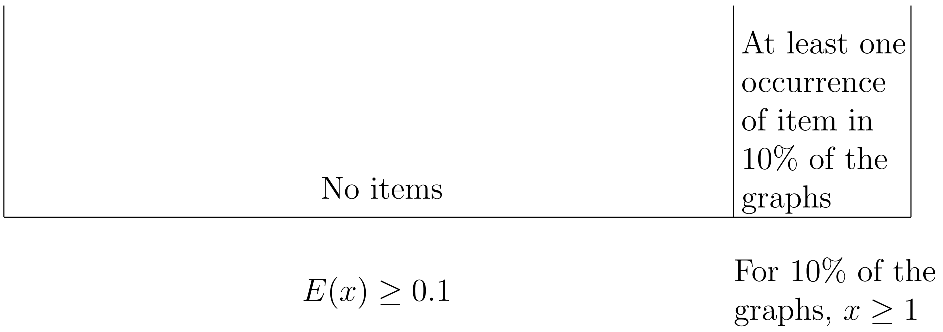

No items
At least one occurrence of item in 10% of the graphs
$E(x) \geq 0.1$
For 10% of the graphs, $x \geq 1$

Figure 4.6: If the expected fraction of the number of graphs in which an item occurs did not go to zero, then $E(x)$, the expected number of items per graph, could not be zero. Suppose 10% of the graphs had at least one occurrence of the item. Then the expected number of occurrences per graph must be at least 0.1. Thus, $E(x) = 0$ implies the probability that a graph has an occurrence of the item goes to zero. However, the other direction needs more work. If $E(x)$ were not zero, a second moment argument is needed to conclude that the probability that a graph picked at random had an occurrence of the item was nonzero since there could be a large number of occurrences concentrated on a vanishingly small fraction of all graphs. The second moment argument claims that for a nonnegative random variable x with $E(x) > 0$, if Var(x) is $o(E^2(x))$ or alternatively if $E(x^2) \leq E^2(x)(1 + o(1))$, then almost surely $x > 0$.

and second moment methods are broadly used. We describe the second moment method in some generality now.

When the expected value of $x(n)$, the number of occurrences of an item, goes to infinity, we cannot conclude that a graph picked at random will likely have a copy since the items may all appear on a small fraction of the graphs. We resort to a technique called the *second moment method.* It is a simple idea based on Chebyshev's inequality.

Theorem 4.3 (Second Moment method) *Let $x(n)$ be a random variable with $E(x) > 0$. If*

$$Var(x) = o\left(E^2(x)\right),$$

then x is almost surely greater than zero.

Proof: If $E(x) > 0$, then for x to be less than or equal to zero, it must differ from its expected value by at least its expected value. Thus,

$$\text{Prob}(x \leq 0) \leq \text{Prob}\left(|x - E(x)| \geq E(x)\right).$$

By Chebyshev inequality

$$\text{Prob}\left(|x - E(x)| \geq E(x)\right) \leq \frac{\text{Var}(x)}{E^2(x)} \to 0.$$

Thus, $\text{Prob}(x \leq 0)$ goes to zero if $\text{Var}(x)$ is $o\left(E^2(x)\right)$. ∎

Corollary 4.4 *Let x be a random variable with $E(x) > 0$. If*

$$E(x^2) \leq E^2(x)(1 + o(1)),$$

then x is almost surely greater than zero.

Proof: If $E(x^2) \leq E^2(x)(1 + o(1))$, then

$$Var(x) = E(x^2) - E^2(x) \leq E^2(x)o(1) = o(E^2(x)).$$

∎

Threshold for graph diameter two

We now present the first example of a sharp phase transition for a property. This means that slightly increasing the edge probability p near the threshold takes us from almost surely not having the property to almost surely having it. The property is that of a random graph having diameter less than or equal to two. The diameter of a graph is the maximum length of the shortest path between a pair of nodes.

The following technique for deriving the threshold for a graph having diameter two is a standard method often used to determine the threshold for many other objects. Let x be a random variable for the number of objects such as triangles, isolated vertices, or Hamilton circuits, for which we wish to determine a threshold. Then we determine the value of p, say p_0, where the expected value of x goes from zero to infinity. For $p < p_0$ almost surely a graph selected at random will not have a copy of x. For $p > p_0$, a second moment argument is needed to establish that the items are not concentrated on a vanishingly small fraction of the graphs and that a graph picked at random will almost surely have a copy.

Our first task is to figure out what to count to determine the threshold for a graph having diameter two. A graph has diameter two if and only if for each pair of vertices i and j, either there is an edge between them or there is another vertex k to which both i and j have an edge. The set of neighbors of i and the set of neighbors of j are random subsets of expected cardinality np. For these two sets to intersect requires $np \approx \sqrt{n}$ or $p \approx \frac{1}{\sqrt{n}}$. Such statements often go under the general name of "birthday paradox" though it is not a paradox. In what follows, we will prove a threshold of $O(\sqrt{\ln n}/\sqrt{n})$ for a graph to have diameter two. The extra factor of $\sqrt{\ln n}$ ensures that every one of the $\binom{n}{2}$ pairs of i and j has a common neighbor. When $p = c\sqrt{\frac{\ln n}{n}}$, for $c < \sqrt{2}$, the graph almost surely has diameter greater than two and for $c > \sqrt{2}$, the graph almost surely has diameter less than or equal to two.

Theorem 4.5 *The property that $G(n,p)$ has diameter two has a sharp threshold at $p = \sqrt{2}\sqrt{\frac{\ln n}{n}}$.*

Proof: If G has diameter greater than two, then there exists a pair of nonadjacent vertices i and j such that no other vertex of G is adjacent to both i and j. This motivates calling such a pair *bad*.

Introduce a set of indicator random variables I_{ij}, one for each pair of vertices (i, j) with $i < j$, where I_{ij} is 1 if and only if the pair (i, j) is bad. Let

$$x = \sum_{i<j} I_{ij}$$

be the number of bad pairs of vertices. Putting $i < j$ in the sum ensures each pair (i, j) is counted only once. A graph has diameter at most two if and only if it has no bad pair, i.e., $x = 0$. Thus, if $\lim_{n\to\infty} E(x) = 0$, then for large n, almost surely, a graph has no bad pair and hence has diameter at most two.

The probability that a given vertex is adjacent to both vertices in a pair of vertices (i,j) is p^2. Hence, the probability that the vertex is not adjacent to both vertices is $1-p^2$. The probability that no vertex is adjacent to the pair (i,j) is $\left(1-p^2\right)^{n-2}$ and the probability that i and j are not adjacent is $1-p$. Since there are $\binom{n}{2}$ pairs of vertices, the expected number of bad pairs is

$$E\left(x\right)=\binom{n}{2}\left(1-p\right)\left(1-p^2\right)^{n-2}.$$

Setting $p=c\sqrt{\frac{\ln n}{n}}$,

$$\begin{aligned}E\left(x\right)&\cong \tfrac{n^2}{2}\left(1-c\sqrt{\tfrac{\ln n}{n}}\right)\left(1-c^2\tfrac{\ln n}{n}\right)^n\\&\cong \tfrac{n^2}{2}e^{-c^2\ln n}\\&\cong \tfrac{1}{2}n^{2-c^2}.\end{aligned}$$

For $c>\sqrt{2}$, $\lim\limits_{n\to\infty}E\left(x\right)\to 0$. Thus, by the first moment method, for $p=c\sqrt{\frac{\ln n}{n}}$ with $c>\sqrt{2}$, $G\left(n,p\right)$ almost surely has no bad pair and hence has diameter at most two.

Next, consider the case $c<\sqrt{2}$ where $\lim\limits_{n\to\infty}E\left(x\right)\to\infty$. We appeal to a second moment argument to claim that almost surely a graph has a bad pair and thus has diameter greater than two.

$$E(x^2)=E\left(\sum_{i<j}I_{ij}\right)^2=E\left(\sum_{i<j}I_{ij}\sum_{k<l}I_{kl}\right)=E\left(\sum_{\substack{i<j\\k<l}}I_{ij}I_{kl}\right)=\sum_{\substack{i<j\\k<l}}E\left(I_{ij}I_{kl}\right).$$

The summation can be partitioned into three summations depending on the number of distinct indices among i,j,k, and l. Call this number a.

$$E\left(x^2\right)=\underset{a=4}{\sum_{\substack{i<j\\k<l}}}E\left(I_{ij}I_{kl}\right)+\underset{a=3}{\sum_{\substack{i<j\\i<k}}}E\left(I_{ij}I_{ik}\right)+\underset{a=2}{\sum_{i<j}}E\left(I_{ij}^2\right). \tag{4.2}$$

Consider the case $a=4$ where i,j,k, and l are all distinct. If $I_{ij}I_{kl}=1$, then both pairs (i,j) and (k,l) are bad and so for each $u\notin\{i,j,k,l\}$, one of the edges (i,u) or (j,u) is absent and, in addition, one of the edges (k,u) or (l,u) is absent. The probability of this for one u not in $\{i,j,k,l\}$ is $(1-p^2)^2$. As u ranges over all the $n-4$ vertices not in $\{i,j,k,l\}$, these events are all independent. Thus,

$$E(I_{ij}I_{kl})\le(1-p^2)^{2(n-4)}\le(1-c^2\frac{\ln n}{n})^{2n}(1+o(1))\le n^{-2c^2}(1+o(1))$$

and the first sum is

$$\sum_{\substack{i<j\\k<l}} E(I_{ij}I_{kl}) \le n^{4-2c^2}(1+o(1)).$$

For the second summation, observe that if $I_{ij}I_{ik}=1$, then for every vertex u not equal to i, j, or k, either there is no edge between i and u or there is an edge (i,u) and both edges (j,u) and (k,u) are absent. The probability of this event for one u is

$$1-p+p(1-p)^2 = 1-2p^2+p^3 \approx 1-2p^2.$$

Thus, the probability for all such u is $(1-2p^2)^{n-3}$. Substituting $c\sqrt{\frac{\ln n}{n}}$ for p yields

$$\left(1-\tfrac{2c^2\ln n}{n}\right)^{n-3} \cong e^{-2c^2\ln n} = n^{-2c^2},$$

which is an upper bound on $E(I_{ij}I_{kl})$ for one i,j,k, and l with $a=3$. Summing over all distinct triples yields n^{3-2c^2} for the second summation in (4.2).

For the third summation, since the value of I_{ij} is zero or one, $E\left(I_{ij}^2\right)=E\left(I_{ij}\right)$. Thus,

$$\sum_{ij} E\left(I_{ij}^2\right) = E\left(x\right).$$

Hence, $E\left(x^2\right) \le n^{4-2c^2}+n^{3-2c^2}+n^{2-c^2}$ and $E\left(x\right) \cong n^{2-c^2}$, from which it follows that for $c<\sqrt{2}$, $E\left(x^2\right) \le E^2\left(x\right)\left(1+o(1)\right)$. By a second moment argument, Corollary 4.4, a graph almost surely has at least one bad pair of vertices and thus has diameter greater than two. Therefore, the property that the diameter of $G(n,p)$ is less than or equal to two has a sharp threshold at $p=\sqrt{2}\sqrt{\frac{\ln n}{n}}$ ■

Disappearance of Isolated Vertices

The disappearance of isolated vertices in $G\left(n,p\right)$ has a sharp threshold at $\frac{\ln n}{n}$. At this point the giant component has absorbed all the small components and with the disappearance of isolated vertices, the graph becomes connected.

Theorem 4.6 *The disappearance of isolated vertices in $G\left(n,p\right)$ has a sharp threshold of $\frac{\ln n}{n}$.*

Proof: Let x be the number of isolated vertices in $G\left(n,p\right)$. Then,

$$E\left(x\right) = n\left(1-p\right)^{n-1}.$$

Since we believe the threshold to be $\frac{\ln n}{n}$, consider $p=c\frac{\ln n}{n}$. Then,

$$\lim_{n\to\infty} E\left(x\right) = \lim_{n\to\infty} n\left(1-\tfrac{c\ln n}{n}\right)^n = \lim_{n\to\infty} ne^{-c\ln n} = \lim_{n\to\infty} n^{1-c}.$$

If c >1, the expected number of isolated vertices, goes to zero. If $c < 1$, the expected number of isolated vertices goes to infinity. If the expected number of isolated vertices goes to zero, it follows that almost all graphs have no isolated vertices. On the other hand, if the expected number of isolated vertices goes to infinity, a second moment argument is needed to show that almost all graphs have an isolated vertex and that the isolated vertices are not concentrated on some vanishingly small set of graphs with almost all graphs not having isolated vertices.

Assume $c < 1$. Write $x = I_1 + I_2 + \cdots + I_n$ where I_i is the indicator variable indicating whether vertex i is an isolated vertex. Then $E\left(x^2\right) = \sum\limits_{i=1}^{n} E\left(I_i^2\right) + 2\sum\limits_{i<j} E\left(I_i I_j\right)$. Since I_i equals 0 or 1, $I_i^2 = I_i$ and the first sum has value $E\left(x\right)$. Since all elements in the second sum are equal

$$\begin{aligned} E\left(x^2\right) &= E\left(x\right) + n\left(n-1\right)E\left(I_1 I_2\right) \\ &= E\left(x\right) + n\left(n-1\right)\left(1-p\right)^{2(n-1)-1}. \end{aligned}$$

The minus one in the exponent $2(n-1)-1$ avoids counting the edge from vertex 1 to vertex 2 twice. Now,

$$\begin{aligned} \frac{E\left(x^2\right)}{E^2\left(x\right)} &= \frac{n\left(1-p\right)^{n-1} + n\left(n-1\right)\left(1-p\right)^{2(n-1)-1}}{n^2\left(1-p\right)^{2(n-1)}} \\ &= \frac{1}{n\left(1-p\right)^{n-1}} + \left(1-\frac{1}{n}\right)\frac{1}{1-p}. \end{aligned}$$

For $p = c\frac{\ln n}{n}$ with $c < 1$, $\lim\limits_{n\to\infty} E\left(x\right) = \infty$ and

$$\lim_{n\to\infty} \frac{E\left(x^2\right)}{E^2\left(x\right)} = \lim_{n\to\infty}\left[\frac{1}{n^{1-c}} + \left(1-\frac{1}{n}\right)\frac{1}{1-c\frac{\ln n}{n}}\right] = 1 + o(1).$$

By the second moment argument, Corollary 4.4, the probability that $x = 0$ goes to zero implying that almost all graphs have an isolated vertex. Thus, $\frac{\ln n}{n}$ is a sharp threshold for the disappearance of isolated vertices. For $p = c\frac{\ln n}{n}$, when $c > 1$ there almost surely are no isolated vertices, and when $c < 1$ there almost surely are isolated vertices. ■

Hamilton circuits

So far in establishing phase transitions in the $G(n, p)$ model for an item such as the disappearance of isolated vertices, we introduced a random variable x that was the number of occurrences of the item. We then determined the probability p for which the expected value of x went from zero to infinity. For values of p for which $E(x) = 0$, we argued that with probability one, a graph generated at random had no occurrences of x. For values of x for which $E(x) \to \infty$, we used the second moment argument to conclude

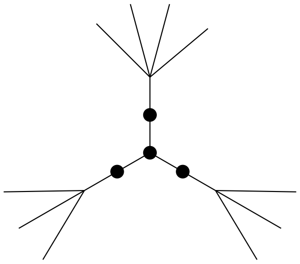


Figure 4.7: A degree three vertex with three adjacent degree two vertices. Graph cannot have a Hamilton circuit.

that with probability one a graph generated at random had occurrences of x. That is, the occurrences that forced $E(x)$ to infinity were not all concentrated on a vanishingly small fraction of the graphs. One might raise the question for the $G(n,p)$ graph model, do there exist items that are so concentrated on a small fraction of the graphs that the value of p where $E(x)$ goes from zero to infinity is not the threshold? An example where this happens is Hamilton circuits.

Let x be the number of Hamilton circuits in $G(n,p)$ and let $p = \frac{d}{n}$ for some constant d. There are $\frac{1}{2}(n-1)!$ potential Hamilton circuits in a graph and each has probability $(\frac{d}{n})^n$ of actually being a Hamilton circuit. Thus,

$$\begin{aligned} E(x) &= \frac{1}{2}(n-1)!\left(\frac{d}{n}\right)^n \\ &\simeq \left(\frac{n}{e}\right)^n\left(\frac{d}{n}\right)^n \\ &= \begin{cases} 0 & d < e \\ \infty & d > e \end{cases}. \end{aligned}$$

This suggests that the threshold for Hamilton circuits occurs when d equals Euler's constant e. This is not possible since the graph still has isolated vertices and is not even connected for $p = \frac{e}{n}$. Thus, the second moment argument is indeed necessary.

The actual threshold for Hamilton circuits is $d = \omega(\log n + \log\log n)$. For any $p(n)$ asymptotically greater than $\frac{1}{n}(\log n + \log\log n)$, $G(n,p)$ will have a Hamilton circuit with probability one. This is the same threshold as for the disappearance of degree one vertices. Clearly a graph with a degree one vertex cannot have a Hamilton circuit. But it may seem surprising that Hamilton circuits appear as soon as degree one vertices disappear. You may ask why at the moment degree one vertices disappear there cannot be a subgraph consisting of a degree three vertex adjacent to three degree two vertices as shown in Figure 4.7. The reason is that the frequency of degree two and three vertices in the graph is very small and the probability that four such vertices would occur together in such a subgraph is too small for it to happen.

4.3 The Giant Component

Consider $G(n, p)$ as p grows. Starting with $p = 0$, the graph has n vertices and no edges. As p increases and edges are added, a forest of trees emerges. When p is $o(1/n)$ the graph is almost surely a forest of trees, i.e., there are no cycles. When p is d/n, d a constant, cycles appear. For $d < 1$, no connected component has asymptotically more than $\log n$ vertices. The number of components containing a single cycle is a constant independent of n. Thus, the graph consists of a forest of trees plus a few components that have a single cycle with no $\Omega(\log n)$ size components.

At p equal $1/n$, a phase transition occurs in which a giant component emerges. The transition consists of a double jump. At $p = 1/n$, components of $n^{2/3}$ vertices emerge, which are almost surely trees. Then at $p = d/n$, $d > 1$, a true giant component emerges that has a number of vertices proportional to n. This is a seminal result in random graph theory and the main subject of this section. Giant components also arise in many real world graphs; the reader may want to look at large real-world graphs, like portions of the web and find the size of the largest connected component.

When one looks at the connected components of large graphs that appear in various contexts, one observes that often there is one very large component. One example is a graph formed from a data base of protean interactions[8] where vertices correspond to proteins and edges correspond to pairs of proteins that interact. By an interaction, one means two amino acid chains that bind to each other for a function. The graph has 2735 vertices and 3602 edges. At the time we looked at the data base, the associated graph had the number of components of various sizes shown in Table 3.1. There are a number of small components, but only one component of size greater than 16, and that is a giant component of size 1851. As more proteins are added to the data base the giant component will grow even larger and eventually swallow up all the smaller components.

Size of component	1	2	3	4	5	6	7	8	9	10	11	12	$\cdots$	15	16	$\cdots$	1851
Number of components	48	179	50	25	14	6	4	6	1	1	1	0	0	0	1	0	1

Table 1: Table 3.1 Size of components in the graph implicit in the database of interacting proteins.

The existence of a giant component is not unique to the graph produced from the protein data set. Take any data set that one can convert to a graph and it is likely that the graph will have a giant component, provided that the ratio of edges to vertices is a small number greater than one half. Table 3.2 gives two other examples. This phenomenon, of the existence of a giant component in many real world graphs deserves study.

[8] Science 1999 July 30 Vol. 285 No. 5428 pp751-753.

ftp://ftp.cs.rochester.edu/pub/u/joel/papers.lst
Vertices are papers and edges mean that two papers shared an author.

1	2	3	4	5	6	7	8	14	27488
2712	549	129	51	16	12	8	3	1	1

http://www.gutenberg.org/etext/3202
Vertices represent words and edges connect words that are synonyms of one another.

1	2	3	4	5	14	16	18	48	117	125	128	30242
7	1	1	1	0	1	1	1	1	1	1	1	1

Table 2: Table 3.2 Size of components in two graphs constructed from data sets.

Returning to $G(n,p)$, as p increases beyond d/n, all nonisolated vertices are absorbed into the giant component, and at $p = \frac{1}{2}\frac{\ln n}{n}$, the graph consists only of isolated vertices plus a giant component. At $p = \frac{\ln n}{n}$, the graph becomes completely connected. By $p = 1/2$, the graph is not only connected, but is sufficiently dense that it has a clique of size $(2-\varepsilon)\log n$ for any $\varepsilon > 0$. We prove many of these facts in this chapter.

To compute the size of a connected component of $G(n,p)$, do a breadth first search of a component starting from an arbitrary vertex and generate an edge only when the search process needs to know if the edge exists. Start at an arbitrary vertex and mark it discovered and unexplored. At a general step, select a discovered, but unexplored vertex v, and explore it as follows. For each undiscovered vertex u, independently decide with probability $p = d/n$ whether the edge (v,u) is in and if it is, mark u discovered and unexplored. After this, mark v explored. Discovered but unexplored vertices are called the frontier. The algorithm has found the entire connected component when the frontier becomes empty.

For each vertex u, other than the start vertex, the probability that u is undiscovered after the first i steps is precisely $(1-\frac{d}{n})^i$. A step is the full exploration of one vertex. Let z_i be the number of vertices discovered in the first i steps of the search. The distribution of z_i is Binomial$\Big(n-1, 1-\big(1-\frac{d}{n}\big)^i\Big)$.

Consider the case $d > 1$. For small values of i, the probability that a vertex is undiscovered after i steps is

$$\left(1-\frac{d}{n}\right)^i \approx 1-\frac{id}{n}.$$

The probability that a vertex is discovered after i steps is $\frac{id}{n}$. The expected number of

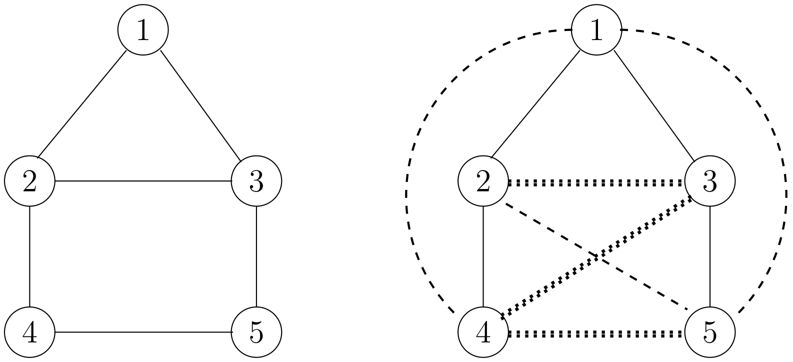


Figure 4.8: A graph (left) and the breadth first search of the graph (right). At vertex 1 the algorithm queried all edges. The solid edges are real edges, the dashed edges are edges that were queried but do not exist. At vertex 2 the algorithm queried all possible edges to vertices not yet discovered. The algorithm does not query whether the edge (2,3) exists since vertex 3 has already been discovered when the algorithm is at vertex 2. Potential edges not queried are illustrated with dotted edges.

discovered vertices grows as id and the expected size of the frontier grows as $(d-1)\,i$. As the fraction of discovered vertices increases, the expected rate of growth of newly discovered vertices decreases since many of the vertices adjacent to the vertex currently being searched have already been discovered. Once $\frac{d-1}{d}n$ vertices have been discovered, the growth of newly discovered vertices slows to one at each step. Eventually for d >1, the growth of discovering new vertices drops below one per step and the frontier starts to shrink. For d <1, $(d-1)\,i$, the expected size of the frontier is negative. The expected rate of growth is less than one, even at the start.

It is easy to make this argument rigorous to prove that for the $d < 1$ case, almost surely, there is no connected component of size $\Omega(\ln n)$. We do this before tackling the more difficult $d > 1$ case.

Theorem 4.7 *Let p=d/n with $d < 1$. The probability that $G(n, p)$ has a component of size more than $c\frac{\ln n}{(1-d)^2}$ is at most $1/n$ for a suitable constant c depending on d but not on n.*

Proof: There is a connected component of size at least k containing a particular vertex v only if the breadth first search started at v has a nonempty frontier at all times up to k. Let z_k be the number of discovered vertices after k steps. The probability that v is in a connected component of size greater than or equal to k is less than or equal to $\text{Prob}(z_k > k)$. Now the distribution of z_k is $\text{Binomial}\big(n-1, 1-(1-d/n)^k\big)$. Since $(1-d/n)^k \geq 1-dk/n$, the mean of $\text{Binomial}\big(n-1, 1-(1-d/n)^k\big)$ is less than the mean of $\text{Binomial}(n, \frac{dk}{n})$. Since $\text{Binomial}(n, \frac{dk}{n})$ has mean dk, the mean of z_k is at most dk where $d < 1$. By a Chernoff bound, the probability that z_k is greater than k is at most $e^{-c_0 k}$ for

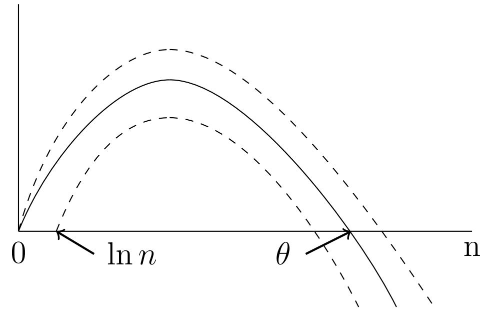


Figure 4.9: The solid curve is the expected size of the frontier. The two dashed curves indicate the range of possible values for the actual size of the frontier.

some constant $c_0 > 0$. If $k \geq c \ln n$ for a suitably large c, then this probability is at most $1/n^2$. This bound is for a single vertex v. Multiplying by n for the union bound completes the proof. ∎

Now assume $d > 1$. As we saw, the expected size of the frontier grows as $(d-1)i$ for small i. The actual size of the frontier is a random variable. What is the probability that the actual size of the frontier will differ from the expected size of the frontier by a sufficient amount so that the actual size of the frontier is zero? To answer this, we need to understand the distribution of the number of discovered vertices after i steps. For small i, the probability that a vertex has been discovered is $1-(1-d/n)^i \approx id/n$ and the binomial distribution for the number of discovered vertices, binomial$(n, \frac{id}{n})$, is well approximated by the Poisson distribution with the same mean id. The probability that a total of k vertices have been discovered in i steps is approximately $e^{-di}\frac{(di)^k}{k!}$. For a connected component to have exactly i vertices, the frontier must drop to zero for the first time at step i. A necessary condition is that exactly i vertices must have been discovered in the first i steps. The probability of this approximately equals

$$e^{-di}\frac{(di)^i}{i!} = e^{-di}\frac{d^i i^i}{i^i}e^i = e^{-(d-1)i}d^i = e^{-(d-1-\ln d)i}.$$

For $d > 1$, $\ln d \leq d-1$ and hence $d-1-\ln d > 0$. This probability drops off exponentially with i. For $i > c \ln n$ and sufficiently large c, the probability that the breadth first search starting from a particular vertex terminates with a component of size i is $o(1/n)$ as long as the Poisson approximation is valid. In the range of this approximation, the probability that a breadth first search started from any vertex terminates with $i > c \ln n$ vertices is $o(1)$. Intuitively, if the component has not stopped growing within $\Omega(\ln n)$ steps, it is likely to continue to grow until it becomes much larger and the expected value of the size of the frontier again becomes small. While the expected value of the frontier is large, the probability that the actual size will differ from the expected size sufficiently for the actual size of the frontier to be zero is vanishingly small.

In Theorem 4.9, we prove that there is one giant component of size $\Omega(n)$ along with a number of components of size $O(\ln n)$. We first prove a technical lemma stating that the probability of a vertex being in a small component is strictly less than one and hence there is a giant component. We refer to a connected component of size $O(\log n)$ as a small component.

Lemma 4.8 *Assume $d > 1$. The probability that $cc(v)$, the connected component containing vertex v, is small (i.e., of size $O(\log n)$) is a constant strictly less than 1.*

Proof: Let p be the probability that $cc(v)$ is small, i.e., the probability that the breadth first search started at v terminates before $c_1 \log n$ vertices are discovered. Slightly modify the breadth first search as follows: If in exploring a vertex u at some point, there are m undiscovered vertices, choose the number k of vertices which will be adjacent to u from Binomial$(m, \frac{d}{n})$ distribution. Having picked k, pick one of the $\binom{m}{k}$ subsets of m undiscovered vertices to be the set of vertices adjacent to u, and make the other $m - k$ vertices not adjacent to u. This process has the same distribution as picking each edge from u independently at random to be present with probability d/n. As the search proceeds, m decreases. If cc(v) is small, m is always greater than $s = n - c_1 \log n$. Modify the process once more picking k from Binomial$(s, \frac{d}{n})$ instead of from Binomial$(m, \frac{d}{n})$. Let p' be the probability that cc(v) is small for the modified process. Clearly, $p' \geq p$, so it suffices to prove that p' is a constant strictly less than one. The mean of the binomial now is $d_1 = sd/n$ which is strictly greater than one. It is clear that the probability that the modified process ends before $c_1 \log n$ vertices are discovered is at least the probability for the original process, since picking from $n - c_1 \log n$ vertices has decreased the number of newly discovered vertices each time. Modifying the process so that the newly discovered vertices are picked from a fixed size set, converts the problem to what is called a branching process..

A branching process is a method for creating a possibly infinite random tree. There is a nonnegative integer-valued random variable y that is the number of children of the node being explored. First, the root v of the tree chooses a value of y according to the distribution of y and spawns that number of children. Each of the children independently chooses a value according to the same distribution of y and spawns that many children. The process terminates when all of the vertices have spawned children. The process may go on forever. If it does terminate with a finite tree, we say that the process has become "extinct". Let Binomial$(s, \frac{d}{n})$ be the distribution of y. Let q be the probability of extinction. Then, $q \geq p'$, since, the breadth first search terminating with at most $c_1 \log n$ vertices is one way of becoming extinct. Let $p_i = \binom{s}{i}(d/n)^i(1 - (d/n))^{s-i}$ be the probability that y spawns i children. We have $\sum_{i=0}^{s} p_i = 1$ and $\sum_{i=1}^{s} ip_i = E(y) = ds/n > 1$.

The depth of a tree is at most the number of nodes in the tree. Let a_t be the probability that the branching process terminates at depth at most t. If the root v has no children, then the process terminates with depth one where the root is counted as a depth one node which is at most t. If v has i children, the process from v terminates at depth at most t if

For a small number i of steps, the probability distribution of the size of the set of discovered vertices at time i is $p(k) = e^{-di}\frac{(di)^k}{k!}$ and has expected value di. Thus, the expected size of the frontier is $(d-1)i$. For the frontier to be empty would require that the size of the set of discovered vertices be smaller than its expected value by $(d-1)i$. That is, the size of the set of discovered vertices would need to be $di - (d-1)i = i$. The probability of this is

$$e^{-di}\frac{(di)^i}{i!} = e^{-di}\frac{d^i i^i}{i^i}e^i = e^{-(d-1)i}d^i = e^{-(d-1-\ln d)i}$$

which drops off exponentially fast with i provided $d > 1$. Since $d - 1 - \ln d$ is some constant $c > 0$, the probability is e^{-ci} which for $i = \ln n$ is $e^{-c\ln n} = \frac{1}{n^c}$. Thus, with high probability, the largest small component in the graph is of size at most $\ln n$.

Illustration 4.1

and only if the i sub processes, one rooted at each child of v terminate at depth $t-1$ or less. The i processes are independent, so the probability that they all terminate at depth at most $t-1$ is exactly a_{t-1}^i. With this we get:

$$a_t = p_0 + \sum_{i=1}^{s} p_i a_{t-1}^i = \sum_{i=0}^{s} p_i a_{t-1}^i.$$

We have $a_1 = p_0 < 1$. There is a constant $\alpha \in [p_0, 1)$ such that whenever $a_{t-1} \leq \alpha$, the above recursion implies that $a_t \leq \alpha$. This would finish the proof since then $a_1 \leq \alpha$ implies $a_2 \leq \alpha$ which implies $a_3 \leq \alpha$ etc. and so $q = \lim_{t\to\infty} a_t \leq \alpha$.

To prove the claim, consider the polynomial

$$h(x) = x - \sum_{i=0}^{s} p_i x^i.$$

We see that $h(1) = 0$ and $h'(1) = 1 - \sum_{i=1}^{s} ip_i \approx 1 - \frac{sd}{n}$, which is at most a strictly negative constant. By continuity of $h(\cdot)$, there is exists some $x_0 < 1$ such that $h(x) \geq 0$ for $x \in [x_0, 1]$. Take $\alpha = \text{Max}(x_0, p_0)$. Now since $\sum_{i=0}^{s} p_i x^i$ has all nonnegative coefficients, it is an increasing function of x and so if a_{t-1} is at least α, then, $\sum_{i=0}^{s} p_i a_{t-1}^i$ is at least $\sum_{i=0}^{s} p_i \alpha^i \geq \alpha$. Now, if $a_{t-1} \leq \alpha$,

$$a_t = \sum_{i=0}^{s} p_i a_{t-1}^i \geq \sum_{i=1}^{s} p_i \alpha^i = \alpha - h(\alpha) \leq \alpha,$$

proving the claim. ■

We now prove in Theorem 4.9 that in $G(n, \frac{d}{n})$, $d > 1$ there is one giant component containing a fraction of the n vertices and that the remaining vertices are in components of size less than some constant c_1 times $\log n$. There are no components greater than $c_1 \log n$ other than the giant component.

Theorem 4.9 *Let $p{=}d/n$ with $d > 1$.*

1. *There are constants c_1 and c_2 such that the probability that there is a connected component of size between $c_1 \log n$ and $c_2 n$ is at most $1/n$.*
2. *The number of vertices in components of size $O(\ln n)$ is almost surely at most cn for some $c < 1$. Thus, with probability $1 - o(1)$, there is a connected component of size $\Omega(n)$.*
3. *The probability that there are two or more connected components, each of size more than $n^{2/3}$, is at most $1/n$.*

Proof: In the breadth first search of a component, the probability that a vertex has not been discovered in i steps is $\left(1 - \frac{d}{n}\right)^i$. It is easy to see that the approximation $(1 - d/n)^i \approx 1 - id/n$ is valid as long as $i \le c_2 n$ for a suitable constant c_2 since the error term in the approximation is $O(i^2 d^2/n^2)$, which for $i \le c_2 n$ is at most a small constant times id/n. This establishes (1).

Next consider (2). For a vertex v, let $\mathrm{cc}(v)$ denote the set of vertices in the connected component containing v. By (1), almost surely, $\mathrm{cc}(v)$ is a small set of size at most $c_1 \log n$ or a large set of size at least $c_2 n$ for every vertex v. The central part of the proof of (2) that the probability of a vertex being in a small component is strictly less than one was established in Lemma 4.8. Let x be the number of vertices in a small connected component. Lemma 4.8 implies that the expectation of the random variable x equals the number of vertices in small connected components is at most some $c_3 n$, for a constant c_3 strictly less than one. But we need to show that for any graph almost surely the actual number x of such vertices is at most some constant strictly less than one times n. For this, we use the second moment method. In this case, the proof that the variance of x is $o(E^2(x))$ is easy. Let x_i be the indicator random variable of the event that $\mathrm{cc}(i)$ is small. Let S and T run over all small sets. Noting that for $i \ne j$, $\mathrm{cc}(i)$ and $\mathrm{cc}(j)$ either are the same or are disjoint,

$$
\begin{aligned}
E(x^2) &= E\left(\left(\sum_{i=1}^{n} x_i\right)^2\right) = \sum_{i,j} E(x_i x_j) = \sum_i E(x_i^2) + \sum_{i\neq j} E(x_i x_j) \\
&= E(x) + \sum_{i\neq j}\sum_{S} \text{Prob}\big(\text{cc}(i) = \text{cc}(j) = S\big) + \sum_{i\neq j}\sum_{\substack{S,T \\ \text{disjoint}}} \text{Prob}\big(\text{cc}(i) = S;\ \text{cc}(j) = T\big) \\
&= E(x) + \sum_{i\neq j}\sum_{S} \text{Prob}\big(\text{cc}(i) = \text{cc}(j) = S\big) \\
&\qquad\qquad + \sum_{i\neq j}\ \sum_{\substack{S,T \\ \text{disjoint}}} \text{Prob}\big(\text{cc}(i) = S\big)\, \text{Prob}\big(\text{cc}(j) = T\big)\,(1-p)^{-|S||T|} \\
&\leq O(n) + (1-p)^{-|S||T|}\left(\sum_{S} \text{Prob}\big(\text{cc}(i) = S\big)\right)\left(\sum_{T} \text{Prob}\big(\text{cc}(j) = T\big)\right) \\
&\leq O(n) + \big(1 + o(1)\big)\, E(x)E(x).
\end{aligned}
$$

In the next to last line, if S containing i and T containing j are disjoint sets, then the two events, S is a connected component and T is a connected component, depend on disjoint sets of edges except for the $|S||T|$ edges between S vertices and T vertices. Let c_4 be a constant in the interval $(c_3, 1)$. Then, by Chebyshev inequality,

$$
\text{Prob}(x > c_4 n) \leq \frac{\text{Var}(x)}{(c_4 - c_3)^2 n^2} \leq \frac{O(n) + o(1)c_3^2 n^2}{(c_4 - c_3)^2 n^2} = o(1).
$$

For the proof of (3) suppose a pair of vertices u and v belong to two different connected components, each of size at least $n^{2/3}$. With high probability, they should have merged into one component producing a contradiction. First, run the breadth first search process starting at v for $\frac{1}{2}n^{2/3}$ steps. Since v is in a connected component of size $n^{2/3}$, there are $\Omega(n^{2/3})$ frontier vertices. The expected size of the frontier continues to grow until some constant times n and the actual size of the frontier does not differ significantly from the expected size. The size of the component also grows linearly with n. Thus, the frontier is of size $n^{\frac{2}{3}}$. See Exercise 4.23. By the assumption, u does not belong to this connected component. Now, temporarily stop the breadth first search tree of v and begin a breadth first search tree starting at u, again for $\frac{1}{2}n^{2/3}$ steps. It is important to understand that this change of order of building $G(n,p)$ does not change the resulting graph. We can choose edges in any order since the order does not affect independence or conditioning. The breadth first search tree from u also will have $\Omega(n^{2/3})$ frontier vertices with high probability . Now grow the u tree further. The probability that none of the edges between the two frontier sets is encountered is $(1-p)^{\Omega(n^{4/3})} \leq e^{-\Omega(dn^{1/3})}$, which converges to zero. So almost surely, one of the edges is encountered and u and v end up in the same connected component. This argument shows for a particular pair of vertices

u and v, the probability that they belong to different large connected components is very small. Now use the union bound to conclude that this does not happen for any of the $\binom{n}{2}$ pairs of vertices. The details are left to the reader. ∎

4.4 Branching Processes

A *branching process* is a method for creating a random tree. Starting with the root node, each node has a probability distribution for the number of its children. The root of the tree denotes a parent and its descendants are the children with their descendants being the grandchildren. The children of the root are the first generation, their children the second generation, and so on. Branching processes have obvious applications in population studies, but also in exploring a connected component in a random graph.

We analyze a simple case of a branching process where the distribution of the number of children at each node in the tree is the same. The basic question asked is what is the probability that the tree is finite, i.e., the probability that the branching process dies out? This is called the *extinction probability.*

Our analysis of the branching process will give the probability of extinction, as well as the expected size of the components conditioned on extinction. Not surprisingly, the expected size of components conditioned on extinction is $O(1)$. This says that in $G(n, \frac{d}{n})$, with $d > 1$, there is one giant component of size $\Omega(n)$, the rest of the components are $O(\ln n)$ in size and the expected size of the small components is $O(1)$.

An important tool in our analysis of branching processes is the generating function. The generating function for a nonnegative integer valued random variable y is $f(x) = \sum\limits_{i=0}^{\infty} p_i x^i$ where p_i is the probability that y equals i. The reader not familiar with generating functions should consult Section 12.7 of the appendix.

Let the random variable z_j be the number of children in the j^{th} generation and let $f_j(x)$ be the generating function for z_j. Then $f_1(x) = f(x)$ is the generating function for the first generation where $f(x)$ is the generating function for the number of children at a node in the tree. The generating function for the 2^{nd} generation is $f_2(x) = f(f(x))$. In general, the generating function for the $j+1^{st}$ generation is given by $f_{j+1}(x) = f_j(f(x))$. To see this, observe two things.

First, the generating function for the sum of two identically distributed integer valued random variables x_1 and x_2 is the square of their generating function

$$f^2(x) = p_0^2 + (p_0p_1 + p_1p_0)\,x + (p_0p_2 + p_1p_1 + p_2p_0)\,x^2 + \cdots .$$

For $x_1 + x_2$ to have value zero, both x_1 and x_2 must have value zero, for $x_1 + x_2$ to have value one, exactly one of x_1 or x_2 must have value zero and the other have value one, and

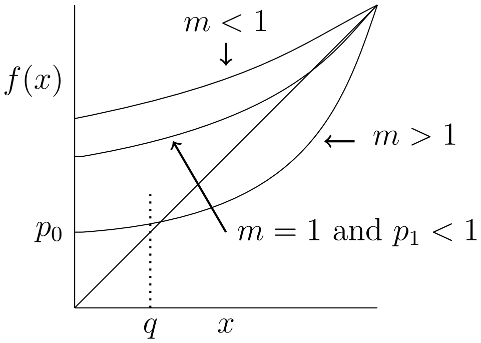


Figure 4.10: Illustration of the root of equation $f(x) = x$ in the interval [0,1).

so on. In general, the generating function for the sum of i independent random variables, each with generating function $f(x)$, is $f^i(x)$.

The second observation is that the coefficient of x^i in $f_j(x)$ is the probability of there being i children in the j^{th} generation. If there are i children in the j^{th} generation, the number of children in the $j + 1^{st}$ generation is the sum of i independent random variables each with generating function $f(x)$. Thus, the generating function for the $j+1^{st}$ generation, given i children in the j^{th} generation, is $f^i(x)$. The generating function for the $j + 1^{st}$ generation is given by

$$f_{j+1}(x) = \sum_{i=0}^{\infty} \text{Prob}(z_j = i) f^i(x).$$

If $f_j(x) = \sum\limits_{i=0}^{\infty} a_i x^i$, then f_{j+1} is obtained by substituting $f(x)$ for x in $f_j(x)$.

Since $f(x)$ and its iterates, $f_2, f_3, \ldots$, are all polynomials in x with nonnegative coefficients, $f(x)$ and its iterates are all monotonically increasing and convex on the unit interval. Since the probabilities of the number of children of a node sum to one, if $p_0 < 1$, some coefficient of x to a power other than zero in $f(x)$ is nonzero and $f(x)$ is strictly increasing.

Let q be the probability that the branching process dies out. If there are i children in the first generation, then each of the i subtrees must die out and this occurs with probability q^i. Thus, q equals the summation over all values of i of the product of the probability of i children times the probability that i subtrees will die out. This gives $q = \sum_{i=0}^{\infty} p_i q^i$. Thus, q is the root of $x = \sum_{i=0}^{\infty} p_i x^i$, that is $x = f(x)$.

This suggests focusing on roots of the equation $f(x) = x$ in the interval [0,1]. The value $x = 1$ is always a root of the equation $f(x) = x$ since $f(1) = \sum\limits_{i=0}^{\infty} p_i = 1$. When is there a smaller nonnegative root? The derivative of $f(x)$ at $x = 1$ is $f'(1) = p_1 + 2p_2 + 3p_3 + \cdots$. Let $m = f'(1)$. Thus, m is the expected number of children of a node. If $m > 1$, one might expect the tree to grow forever, since each node at time j is expected to have more than one child. But this does not imply that the probability of extinction is zero. In fact, if $p_0 > 0$, then with positive probability, the root will have no children and the process will become extinct right away. Recall that for $G(n, \frac{d}{n})$, the expected number of children is d, so the parameter m plays the role of d.

If $m < 1$, then the slope of $f(x)$ at $x = 1$ is less than one. This fact along with convexity of $f(x)$ implies that $f(x) > x$ for x in $[0, 1)$ and there is no root of $f(x) = x$ in the interval $[0, 1)$.

If $m = 1$ and $p_1 < 1$, then once again convexity implies that $f(x) > x$ for $x \in [0, 1)$ and there is no root of $f(x) = x$ in the interval $[0, 1)$. If $m = 1$ and $p_1 = 1$, then $f(x)$ is the straight line $f(x) = x$.

If m >1, then the slope of $f(x)$ is greater than the slope of x at $x = 1$. This fact, along with convexity of $f(x)$, implies $f(x) = x$ has a unique root in [0,1). When $p_0 = 0$, the root is at $x = 0$.

Let q be the smallest nonnegative root of the equation $f(x) = x$. For $m < 1$ and for m=1 and $p_0 < 1$, q equals one and for m >1, q is strictly less than one. We shall see that the value of q is the *extinction probability* of the branching process and that $1 - q$ is the *immortality probability*. That is, q is the probability that for some j, the number of children in the j^{th} generation is zero. To see this, note that for $m > 1$, $\lim_{j\to\infty} f_j(x) = q$ for $0 \le x < 1$. Figure 4.11 illustrates the proof which is given in Lemma 4.10. Similarly note that when $m < 1$ or $m = 1$ with $p_0 < 1$, $f_j(x)$ approaches one as j approaches infinity.

Lemma 4.10 *Assume $m > 1$. Let q be the unique root of f(x)=x in [0,1). In the limit as j goes to infinity, $f_j(x) = q$ for x in $[0, 1)$.*

Proof: If $0 \le x \le q$, then $x < f(x) \le f(q)$ and iterating this inequality

$$x < f_1(x) < f_2(x) < \cdots < f_j(x) < f(q) = q.$$

Clearly, the sequence converges and it must converge to a fixed point where $f(x) = x$. Similarly, if $q \le x < 1$, then $f(q) \le f(x) < x$ and iterating this inequality

$$x > f_1(x) > f_2(x) > \cdots > f_j(x) > f(q) = q.$$

In the limit as j goes to infinity $f_j(x) = q$ for all x, $0 \le x < 1$. ■

Recall that $f_j(x)$ is the generating function $\sum_{i=0}^{\infty} \text{Prob}(z_j = i)\, x^i$. The fact that in the limit the generating function equals the constant q, and is not a function of x, says

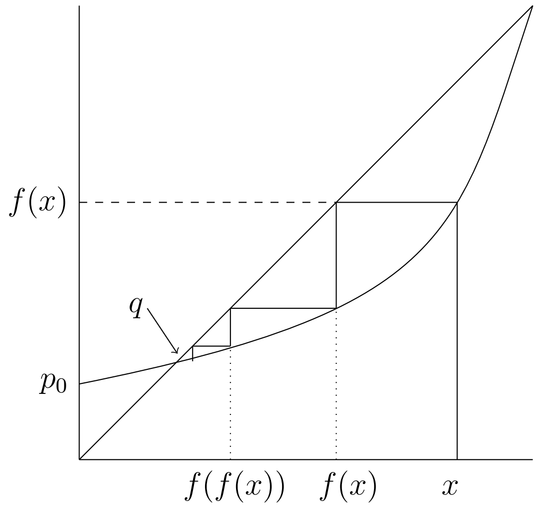


Figure 4.11: Illustration of convergence of the sequence of iterations $f_1(x), f_2(x), \ldots$ to q.

that $\text{Prob}\,(z_j = 0) = q$ and $\text{Prob}\,(z_j = i) = 0$ for all finite nonzero values of i. The remaining probability is the probability of a nonfinite component. Thus, when m >1, q is the extinction probability and 1-q is the probability that z_j grows without bound, i.e., immortality.

Theorem 4.11 *Consider a tree generated by a branching process. Let $f(x)$ be the generating function for the number of children at each node.*

1. *If the expected number of children at each node is less than or equal to one, then the probability of extinction is one unless the probability of exactly one child is one.*

2. *If the expected number of children of each node is greater than one, then the probability of extinction is the unique solution to $f(x) = x$ in $[0, 1)$.*

Proof: Let p_i be the probability of i children at each node. Then $f(x) = p_0 + p_1 x + p_2 x^2 + \cdots$ is the generating function for the number of children at each node and $f'(1) = p_1 + 2p_2 + 3p_3 + \cdots$ is the slope of $f(x)$ at $x = 1$. Observe that $f'(1)$ is the expected number of children at each node.

Since the expected number of children at each node is the slope of $f(x)$ at $x = 1$, if the expected number of children is less than or equal to one, the slope of $f(x)$ at $x = 1$ is less than or equal to one and the unique root of $f(x) = x$ in $(0, 1]$ is at $x = 1$ and the probability of extinction is one unless $f'(1) = 1$ and $p_1 = 1$. If $f'(1) = 1$ and $p_1 = 1$, $f(x) = x$ and the tree is an infinite degree one chain. If the slope of $f(x)$ at $x = 1$ is greater than one, then the probability of extinction is the unique solution to $f(x) = x$ in $[0, 1)$. ■

A branching process with m <1 or m=1 and $p_1 < 1$ dies out with probability one. If m=1 and $p_1 = 1$, then the branching process consists of an infinite chain with no fan out. If m >1, then the branching process will die out with some probability less than one unless $p_0 = 0$ in which case it cannot die out, since a node always has at least one descendent.

Note that the branching process corresponds to finding the size of a component in an infinite graph. In a finite graph, the probability distribution of descendants is not a constant as more and more vertices of the graph get discovered.

The simple branching process defined here either dies out or goes to infinity. In biological systems there are other factors, since processes often go to stable populations. One possibility is that the probability distribution for the number of descendants of a child depends on the total population of the current generation.

Expected size of extinct families

We now show that the expected size of an extinct family is finite, provided that $m \neq 1$. Note that at extinction, the size must be finite. However, the expected size at extinction could conceivably be infinite, if the probability of dying out did not decay fast enough. To see how the expected value of a random variable that is always finite could be infinite, let x be an integer valued random variable. Let p_i be the probability that $x = i$. If $\sum\limits_{i=1}^{\infty} p_i = 1$, then with probability one, x will be finite. However, the expected value of x may be infinite. That is, $\sum\limits_{i=0}^{\infty} ip_i = \infty$. For example, if for i >0, $p_i = \frac{6}{\pi}\frac{1}{i^2}$, then $\sum\limits_{i=1}^{\infty} p_i = 1$, but $\sum\limits_{i=1}^{\infty} ip_i = \infty$. The value of the random variable x is always finite, but its expected value is infinite. This does not happen in a branching process, except in the special case where the slope $m = f'(1)$ equals one and $p_1 \neq 1$

Lemma 4.12 *If the slope $m = f'(1)$ does not equal one, then the expected size of an extinct family is finite. If the slope m equals one and $p_1 = 1$, then the tree is an infinite degree one chain and there are no extinct families. If m=1 and $p_1 < 1$, then the expected size of the extinct family is infinite.*

Proof: Let z_i be the random variable denoting the size of the i^{th} generation and let q be the probability of extinction. The probability of extinction for a tree with k children in the first generation is q^k since each of the k children has an extinction probability of q. Note that the expected size of z_1, the first generation, over extinct trees will be smaller than the expected size of z_1 over all trees since when the root node has a larger number of children than average, the tree is more likely to be infinite.

By Bayes rule

$$\text{Prob}\left(z_1 = k|\text{extinction}\right) = \text{Prob}\left(z_1 = k\right) \frac{\text{Prob}\left(\text{extinction}|z_1 = k\right)}{\text{Prob}\left(\text{extinction}\right)} = p_k \frac{q^k}{q} = p_k q^{k-1}.$$

Knowing the probability distribution of z_1 given extinction, allows us to calculate the expected size of z_1 given extinction.

$$E\left(z_1|\text{extinction}\right) = \sum_{k=0}^{\infty} k p_k q^{k-1} = f'\left(q\right).$$

We now prove, using independence, that the expected size of the i^{th} generation given extinction is

$$E\left(z_i|\text{extinction}\right) = \left(f'\left(q\right)\right)^i.$$

For $i = 2$, z_2 is the sum of z_1 independent random variables, each independent of the random variable z_1. So, $E(z_2|z_1 = j \text{ and extinction}) = E(\text{ sum of } j \text{ copies of } z_1|\text{extinction}) = jE(z_1|\text{extinction})$. Summing over all values of j

$$\begin{aligned} E(z_2|\text{extinction}) &= \sum_{j=1}^{\infty} E(z_2|z_1 = j \text{ and extinction})\text{Prob}(z_1 = j|\text{extinction}) \\ &= \sum_{j=1}^{\infty} jE(z_1|\text{extinction})\text{Prob}(z_1 = j|\text{extinction}) \\ &= E(z_1|\text{extinction}) \sum_{j=1}^{\infty} j\text{Prob}(z_1 = j|\text{extinction}) = E^2(z_1|\text{extinction}). \end{aligned}$$

Since $E(z_1|\text{extinction}) = f'(q)$, $E\left(z_2|\text{extinction}\right) = \left(f'\left(q\right)\right)^2$. Similarly, $E\left(z_i|\text{extinction}\right) = \left(f'\left(q\right)\right)^i$. The expected size of the tree is the sum of the expected sizes of each generation. That is,

$$\begin{array}{l}\text{Expected size of} \\ \text{tree given extinction}\end{array} = \sum_{i=0}^{\infty} E\left(z_i|\text{extinction}\right) = \sum_{i=0}^{\infty} \left(f'\left(q\right)\right)^i = \frac{1}{1 - f'\left(q\right)}.$$

Thus, the expected size of an extinct family is finite since $f'\left(q\right) < 1$ provided $m \neq 1$.

The fact that $f'(q) < 1$ is illustrated in Figure 4.10. If m <1, then q=1 and $f'(q) = m$ is less than one. If m >1, then $q \in [0, 1)$ and again $f'(q)$ <1 since q is the solution to $f(x) = x$ and $f'(q)$ must be less than one for the curve $f(x)$ to cross the line x. Thus, for m <1 or m >1, $f'(q)$ <1 and the expected tree size of $\frac{1}{1-f'(q)}$ is finite. For m=1 and $p_1 < 1$, one has q=1 and thus $f'(q) = 1$ and the formula for the expected size of the tree diverges. ■

4.5 Cycles and Full Connectivity

This section considers when cycles form and when the graph becomes fully connected. For both of these problems, we look at each subset of k vertices and see when they form either a cycle or a connected component.

4.5.1 Emergence of Cycles

The emergence of cycles in $G\,(n,p)$ has a threshold when p equals to $1/n$.

Theorem 4.13 *The threshold for the existence of cycles in $G\,(n,p)$ is $p = 1/n$.*

Proof: Let x be the number of cycles in $G\,(n,p)$. To form a cycle of length k, the vertices can be selected in $\binom{n}{k}$ ways. Given the k vertices of the cycle, they can be ordered by arbitrarily selecting a first vertex, then a second vertex in one of k-1 ways, a third in one of $k-2$ ways, etc. Since a cycle and its reversal are the same cycle, divide by 2. Thus, there are $\binom{n}{k}\frac{(k-1)!}{2}$ cycles of length k and

$$E\,(x) = \sum_{k=3}^{n} \binom{n}{k} \tfrac{(k-1)!}{2} p^k \leq \sum_{k=3}^{n} \tfrac{n^k}{2k} p^k \leq \sum_{k=3}^{n} (np)^k = (np)^3\ \tfrac{1-(np)^{n-2}}{1-np} \leq 2(np)^3,$$

provided that $np < 1/2$. When p is asymptotically less than $1/n$, then $\lim\limits_{n\to\infty} np = 0$ and $\lim\limits_{n\to\infty} \sum\limits_{k=3}^{n} (np)^k = 0$. So, as n goes to infinity, $E(x)$ goes to zero. Thus, the graph almost surely has no cycles by the first moment method. A second moment argument can be used to show that for $p = d/n$, $d > 1$, a graph will have a cycle with probability tending to one. ■

The argument above does not yield a sharp threshold since we argued that $E(x) \to 0$ only under the assumption that p is asymptotically less that $\frac{1}{n}$.. A sharp threshold requires $E(x) \to 0$ for $p = d/n$, $d < 1$.

Consider what happens in more detail when $p = d/n$, d a constant.

$$\begin{aligned} E\,(x) &= \sum_{k=3}^{n} \binom{n}{k} \frac{(k-1)!}{2} p^k \\ &= \frac{1}{2} \sum_{k=3}^{n} \frac{n(n-1)\cdots(n-k+1)}{k!} (k-1)!\; p^k \\ &= \frac{1}{2} \sum_{k=3}^{n} \frac{n(n-1)\cdots(n-k+1)}{n^k} \frac{d^k}{k}. \end{aligned}$$

$E\,(x)$ converges if $d < 1$, and diverges if $d \geq 1$. If $d < 1$, $E\,(x) \leq \frac{1}{2} \sum\limits_{k=3}^{n} \frac{d^k}{k}$ and $\lim\limits_{n\to\infty} E\,(x)$ equals a constant greater than zero. If $d = 1$, $E\,(x) = \frac{1}{2} \sum\limits_{k=3}^{n} \frac{n(n-1)\cdots(n-k+1)}{n^k} \frac{1}{k}$. Consider

Property	Threshold
cycles	$1/n$
giant component	$1/n$
giant component + isolated vertices	$\frac{1}{2}\frac{\ln n}{n}$
connectivity, disappearance of isolated vertices	$\frac{\ln n}{n}$
diameter two	$\sqrt{\frac{2\ln n}{n}}$

only the first $\log n$ terms of the sum. Since $\frac{n}{n-i} = 1 + \frac{i}{n-i} \leq e^{i/n-i}$, it follows that $\frac{n(n-1)\cdots(n-k+1)}{n^k} \geq 1/2$. Thus,

$$E(x) \geq \frac{1}{2}\sum_{k=3}^{\log n} \frac{n(n-1)\cdots(n-k+1)}{n^k}\frac{1}{k} \geq \frac{1}{4}\sum_{k=3}^{\log n}\frac{1}{k}.$$

Then, in the limit as n goes to infinity

$$\lim_{n\to\infty} E(x) \geq \lim_{n\to\infty}\frac{1}{4}\sum_{k=3}^{\log n}\frac{1}{k} \geq \lim_{n\to\infty}(\log\log n) = \infty.$$

For $p = d/n$, $d < 1$, $E(x)$ converges to a nonzero constant and with some nonzero probability, graphs will have a constant number of cycles independent of the size of the graph. For $d > 1$, $E(x)$ converges to infinity and a second moment argument shows that graphs will have an unbounded number of cycles increasing with n.

4.5.2 Full Connectivity

As p increases from $p = 0$, small components form. At $p = 1/n$ a giant component emerges and swallows up smaller components, starting with the larger components and ending up swallowing isolated vertices forming a single connected component at $p = \frac{\ln n}{n}$, at which point the graph becomes connected. We begin our development with a technical lemma.

Lemma 4.14 *The expected number of connected components of size k in $G(n,p)$ is at most*

$$\binom{n}{k} k^{k-2}p^{k-1}(1-p)^{kn-k^2}.$$

Proof: The probability that k vertices form a connected component consists of the product of two probabilities. The first is the probability that the k vertices are connected, and the second is the probability that there are no edges out of the component to the remainder of the graph. The first probability is at most the sum over all spanning trees of the k vertices, that the edges of the spanning tree are present. The "at most" in the lemma statement is because $G(n,p)$ may contain more than one spanning tree on these nodes and, in this case, the union bound is higher than the actual probability. There are k^{k-2} spanning trees on k nodes. See Section 12.8.6 in the appendix. The probability of all the $k-1$ edges of one spanning tree being present is p^{k-1} and the probability that there are no edges connecting the k vertices to the remainder of the graph is $(1-p)^{k(n-k)}$. Thus, the probability of one particular set of k vertices forming a connected component is at most $k^{k-2}p^{k-1}(1-p)^{kn-k^2}$. Thus, the expected number of connected components of size k is $\binom{n}{k}k^{k-2}p^{k-1}(1-p)^{kn-k^2}$. ■

We now prove that for $p = \frac{1}{2}\frac{\ln n}{n}$, the giant component has absorbed all small components except for isolated vertices.

Theorem 4.15 *Let $p = c\frac{\ln n}{n}$. For $c > 1/2$, almost surely there are only isolated vertices and a giant component. For $c > 1$, almost surely the graph is connected.*

Proof: We prove that almost surely for $c > 1/2$, there is no connected component with k vertices for any k, $2 \le k \le n/2$. This proves the first statement of the theorem since, if there were two or more components that are not isolated vertices, both of them could not be of size greater than $n/2$. The second statement that for $c > 1$ the graph is connected then follows from Theorem 4.6 which states that isolated vertices disappear at $c = 1$.

We now show that for $p = c\frac{\ln n}{n}$, the expected number of components of size k, $2 \le k \le n/2$, is less than n^{1-2c} and thus for $c > 1/2$ there are no components, except for isolated vertices and the giant component. Let x_k be the number of connected components of size k. Substitute $p = c\frac{\ln n}{n}$ into $\binom{n}{k}k^{k-2}p^{k-1}(1-p)^{kn-k^2}$ and simplify using $\binom{n}{k} \le (en/k)^k$, $1-p \le e^{-p}$, $k-1 < k$, and $x = e^{\ln x}$ to get

$$E(x_k) \le \exp\left(\ln n + k + k\ln\ln n - 2\ln k + k\ln c - ck\ln n + ck^2\frac{\ln n}{n}\right).$$

Keep in mind that the leading terms here for large k are the last two and, in fact, at $k = n$, they cancel each other so that our argument does not prove the fallacious statement for $c \ge 1$ that there is no connected component of size n, since there is. Let

$$f(k) = \ln n + k + k\ln\ln n - 2\ln k + k\ln c - ck\ln n + ck^2\frac{\ln n}{n}.$$

Differentiating with respect to k,

$$f'(k) = 1 + \ln\ln n - \frac{2}{k} + \ln c - c\ln n + \frac{2ck\ln n}{n}$$

and

$$f''(k) = \frac{2}{k^2} + \frac{2c\ln n}{n} > 0.$$

Thus, the function $f(k)$ attains its maximum over the range $[2, n/2]$ at one of the extreme points 2 or $n/2$. At $k = 2$, $f(2) \approx (1-2c)\ln n$ and at $k = n/2$, $f(n/2) \approx -c\frac{n}{4}\ln n$. So

$f(k)$ is maximum at $k = 2$. For $k = 2$, $E(x)_k = e^{f(k)}$ is approximately $e^{(1-2c)\ln n} = n^{1-2c}$ and is geometrically falling as k increases from 2. At some point $E(x_k)$ starts to increase but never gets above $n^{-\frac{c}{4}n}$. Thus, the expected sum of the number of components of size k, for $2 \le k \le n/2$ is

$$E\left(\sum_{k=2}^{n/2} x_k\right) = O(n^{1-2c}).$$

This expected number goes to zero for $c > 1/2$ and the first-moment method implies that, almost surely, there are no components of size between 2 and $n/2$. This completes the proof of Theorem 4.15. ■

4.5.3 Threshold for $\mathbf{O}(\ln \mathbf{n})$ Diameter

We now show that within a constant factor of the threshold for graph connectivity, not only is the graph connected, but its diameter is $O(\ln n)$. That is, if p is $\Omega(\ln n/n)$, the diameter of $G(n, p)$ is $O(\ln n)$.

Consider a particular vertex v. Let S_i be the set of vertices at distance i from v. We argue that as i grows, $|S_1| + |S_2| + \cdots + |S_i|$ grows by a constant factor up to a size of $n/1000$. This implies that in $O(\ln n)$ steps, at least $n/1000$ vertices are connected to v. Then, there is a simple argument at the end of the proof of Theorem 4.17 that a pair of $n/1000$ sized subsets, connected to two different vertices v and w, have an edge between them.

Lemma 4.16 *Consider $G(n, p)$ for sufficiently large n with $p = c \ln n/n$ for any $c > 0$. Let S_i be the set of vertices at distance i from some fixed vertex v. If $|S_1|+|S_2|+\cdots+|S_i| \le n/1000$, then*

$$\textit{Prob}\left(|S_{i+1}| < 2(|S_1| + |S_2| + \cdots + |S_i|)\right) \le e^{-10|S_i|}.$$

Proof: Let $|S_i| = k$. For each vertex u not in $S_1 \cup S_2 \cup \ldots \cup S_i$, the probability that u is not in S_{i+1} is $(1-p)^k$ and these events are independent. So, $|S_{i+1}|$ is the sum of $n - (|S_1| + |S_2| + \cdots + |S_i|)$ independent Bernoulli random variables, each with probability of

$$1 - (1-p)^k \ge 1 - e^{-ck \ln n/n}$$

of being one. Note that $n - (|S_1| + |S_2| + \cdots + |S_i|) \ge 999n/1000$. So,

$$E(|S_{i+1}|) \ge \frac{999n}{1000}(1 - e^{-ck\frac{\ln n}{n}}).$$

Subtracting $200k$ from each side

$$E(|S_{i+1}|) - 200k \ge \frac{n}{2}\left(1 - e^{-ck\frac{\ln n}{n}} - 400\frac{k}{n}\right).$$

Let $\alpha = \frac{k}{n}$ and $f(\alpha) = 1 - e^{-c\alpha \ln n} - 400\alpha$. By differentiation $f''(\alpha) \le 0$, so f is concave and the minimum value of f over the interval $[0, 1/1000]$ is attained at one of the end points. It is easy to check that both $f(0)$ and $f(1/1000)$ are greater than or equal to zero for sufficiently large n. Thus, f is nonnegative throughout the interval proving that $E(|S_{i+1}|) \geq 200|S_i|$. The lemma follows from Chernoff bounds. ∎

Theorem 4.17 *For $p \geq c \ln n/n$, where c is a sufficiently large constant, almost surely, $G(n,p)$ has diameter $O(\ln n)$.*

Proof: By Corollary 4.2, almost surely, the degree of every vertex is $\Omega(np) = \Omega(\ln n)$, which is at least $20 \ln n$ for c sufficiently large. Assume this holds. So, for a fixed vertex v, S_1 as defined in Lemma 4.16 satisfies $|S_1| \geq 20 \ln n$.

Let i_0 be the least i such that $|S_1|+|S_2|+\cdots+|S_i| > n/1000$. From Lemma 4.16 and the union bound, the probability that for some $i, 1 \leq i \leq i_0-1$, $|S_{i+1}| < 2(|S_1|+|S_2|+\cdots+|S_i|)$ is at most $\sum_{k=20 \ln n}^{n/1000} e^{-10k} \leq 1/n^4$. So, with probability at least $1-(1/n^4)$, each S_{i+1} is at least double the sum of the previous S_j 's, which implies that in $O(\ln n)$ steps, i_0+1 is reached.

Consider any other vertex w. We wish to find a short $O(\ln n)$ length path between v and w. By the same argument as above, the number of vertices at distance $O(\ln n)$ from w is at least $n/1000$. To complete the argument, either these two sets intersect in which case we have found a path from v to w of length $O(\ln n)$ or they do not intersect. In the latter case, with high probability there is some edge between them. For a pair of disjoint sets of size at least $n/1000$, the probability that none of the possible $n^2/10^6$ or more edges between them is present is at most $(1-p)^{n^2/10^6} = e^{-\Omega(n \ln n)}$. There are at most 2^{2n} pairs of such sets and so the probability that there is some such pair with no edges is $e^{-\Omega(n \ln n)+O(n)} \to 0$. Note that there is no conditioning problem since we are arguing this for every pair of such sets. Think of whether such an argument made for just the n subsets of vertices, which are vertices at distance at most $O(\ln n)$ from a specific vertex, would work. ∎

4.6 Phase Transitions for Increasing Properties

For many graph properties such as connectivity, having no isolated vertices, having a cycle, etc., the probability of a graph having the property increases as edges are added to the graph. Such a property is called an increasing property. Q is an *increasing property* of graphs if when a graph G has the property, any graph obtained by adding edges to G must also have the property. In this section we show that any increasing property, in fact, has a threshold, although not necessarily a sharp one.

The notion of increasing property is defined in terms of adding edges. The following lemma proves that if Q is an increasing property, then increasing p in $G(n,p)$ increases the probability of the property Q.

Lemma 4.18 *If Q is an increasing property of graphs and $0 \leq p \leq q \leq 1$, then the probability that $G\left(n, q\right)$ has property Q is greater than or equal to the probability that $G\left(n, p\right)$ has property Q.*

Proof: This proof uses an interesting relationship between $G\left(n, p\right)$ and $G\left(n, q\right)$. Generate $G\left(n, q\right)$ as follows. First generate $G\left(n, p\right)$. This means generating a graph on n vertices with edge probabilities p. Then, independently generate another graph $G\left(n, \frac{q-p}{1-p}\right)$ and take the union by putting in an edge if either of the two graphs has the edge. Call the resulting graph H. The graph H has the same distribution as $G\left(n, q\right)$. This follows since the probability that an edge is in H is $p + (1-p)\frac{q-p}{1-p} = q$, and, clearly, the edges of H are independent. The lemma follows since whenever $G\left(n, p\right)$ has the property Q, H also has the property Q. ■

We now introduce a notion called *replication*. An m-fold replication of $G(n,p)$ is a random graph obtained as follows. Generate m independent copies of $G(n,p)$. Include an edge in the m-fold replication if the edge is in any one of the m copies of $G(n,p)$. The resulting random graph has the same distribution as $G(n,q)$ where $q = 1-(1-p)^m$ since the probability that a particular edge is not in the m-fold replication is the product of probabilities that it is not in any of the m copies of $G(n,p)$. If the m-fold replication of $G(n,p)$ does not have an increasing property Q, then none of the m copies of $G(n,p)$ has the property. The converse is not true. If no copy has the property, their union may have it. Since Q is an increasing property and $q = 1-(1-p)^m \leq 1-(1-mp) = mp$

$$\text{Prob}\left(G(n,mp) \text{ has } Q\right) \geq \text{Prob}\left(G(n,q) \text{ has } Q\right) \tag{4.3}$$

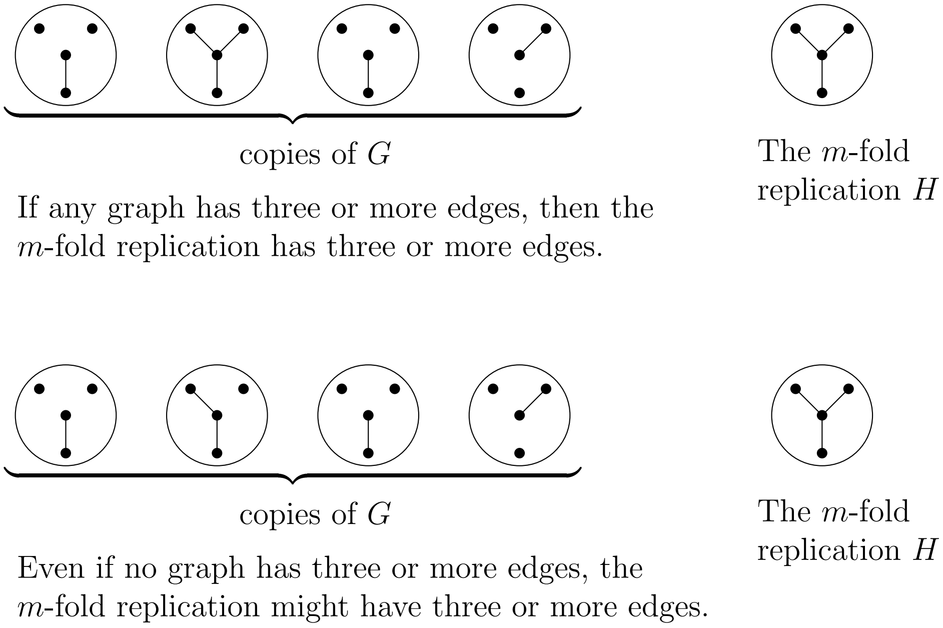
We now show that any increasing property Q has a phase transition. The transition occurs at the point at which the probability that $G(n,p)$ has property Q is $\frac{1}{2}$. We will prove that for any function asymptotically less then $p(n)$ that the probability of having property Q goes to zero as n goes to infinity.

Theorem 4.19 *Every increasing property Q of $G(n,p)$ has a phase transition at $p(n)$, where for each n, $p(n)$ is the minimum real number a_n for which the probability that $G(n,a_n)$ has property Q is 1/2.*

Proof: Let $p_0(n)$ be any function such that

$$\lim_{n\to\infty} \frac{p_0(n)}{p(n)} = 0.$$

We assert that almost surely $G(n,p_0)$ does not have the property Q. Suppose for contradiction, that this is not true. That is, the probability that $G(n,p_0)$ has the property Q does not converge to zero. By the definition of a limit, there exists $\varepsilon > 0$ for which the probability that $G(n,p_0)$ has property Q is at least ε on an infinite set I of n. Let $m = \lceil (1/\varepsilon) \rceil$. Let $G(n,q)$ be the m-fold replication of $G(n,p_0)$. The probability that



If any graph has three or more edges, then the m-fold replication has three or more edges.

Even if no graph has three or more edges, the m-fold replication might have three or more edges.

Figure 4.12: The property that G has three or more edges is an increasing property. Let H be the m-fold replication of G. If any copy of G has three or more edges, H has three or more edges. However, H can have three or more edges even if no copy of G has three or more edges.

$G(n,q)$ does not have Q is at most $(1-\varepsilon)^m \le e^{-1} \le 1/2$ for all $n \in I$. For these n, by (11.4)

$$\text{Prob}(G(n, mp_0) \text{ has } Q) \ge \text{Prob}(G(n,q) \text{ has } Q) \ge 1/2.$$

Since $p(n)$ is the minimum real number a_n for which the probability that $G(n, a_n)$ has property Q is $1/2$, it must be that $mp_0(n) \ge p(n)$. This implies that $\frac{p_0(n)}{p(n)}$ is at least $1/m$ infinitely often, contradicting the hypothesis that $\lim\limits_{n\to\infty} \frac{p_0(n)}{p(n)} = 0$.

A symmetric argument shows that for any $p_1(n)$ such that $\lim\limits_{n\to\infty} \frac{p(n)}{p_1(n)} = 0$, $G(n, p_1)$ almost surely has property Q. ∎

4.7 Phase Transitions for CNF-sat

Phase transitions occur not only in random graphs, but in other random structures as well. An important example is that of satisfiability for a Boolean formula in conjunctive normal form.

Generate a random CNF formula f with n variables, m clauses, and k literals per clause. Each clause is picked independently with k literals picked uniformly at random from the set of $2n$ possible literals to form the clause. Here, the number of clauses n

is going to infinity, m is a function of n, and k is a fixed constant. A reasonable value to think of for k is $k = 3$. A literal is a variable or its negation. Unsatisfiability is an increasing property since adding more clauses preserves unsatisfiability. By arguments similar to the last section, there is a phase transition, i.e., a function $m(n)$ such that if $m_1(n)$ is $o(m(n))$, a random formula with $m_1(n)$ clauses is, almost surely, satisfiable and for $m_2(n)$ with $m_2(n)/m(n) \to \infty$, a random formula with $m_2(n)$ clauses is, almost surely, unsatisfiable. It has been conjectured that there is a constant r_k independent of n such that $r_k n$ is a sharp threshold.

Here we derive upper and lower bounds on r_k. It is relatively easy to get an upper bound on r_k. A fixed truth assignment satisfies a random k clause with probability $1 - \frac{1}{2^k}$. Of the 2^k truth assignments to the k variables in the clause, only one fails to satisfy the clause. Thus, with probability $\frac{1}{2^k}$, the clause is not satisfied, and with probability $1 - \frac{1}{2^k}$, the clause is satisfied. Let $m = cn$. Now, cn independent clauses are all satisfied by the fixed assignment with probability $\left(1 - \frac{1}{2^k}\right)^{cn}$. Since there are 2^n truth assignments, the expected number of satisfying assignments for a formula with cn clauses is $2^n \left(1 - \frac{1}{2^k}\right)^{cn}$. If $c = 2^k \ln 2$, the expected number of satisfying assignments is

$$2^n \left(1 - \tfrac{1}{2^k}\right)^{n2^k \ln 2}.$$

$\left(1 - \frac{1}{2^k}\right)^{2^k}$ is at most $1/e$ and approaches $1/e$ in the limit. Thus,

$$2^n \left(1 - \tfrac{1}{2^k}\right)^{n2^k \ln 2} \leq 2^n e^{-n \ln 2} = 2^n 2^{-n} = 1.$$

For $c > 2^k \ln 2$, the expected number of satisfying assignments goes to zero as $n \to \infty$. Here the expectation is over the choice of clauses which is random, not the choice of a truth assignment. From the first moment method, it follows that a random formula with cn clauses is almost surely not satisfiable. Thus, $r_k \leq 2^k \ln 2$.

The other direction, showing a lower bound for r_k, is not that easy. From now on, we focus only on the case $k = 3$. The statements and algorithms given here can be extended to $k \geq 4$, but with different constants. It turns out that the second moment method cannot be directly applied to get a lower bound on r_3 because the variance is too high. A simple algorithm, called the Smallest Clause Heuristic (abbreviated SC), yields a satisfying assignment with probability tending to one if $c < \frac{2}{3}$, proving that $r_3 \geq \frac{2}{3}$. Other more difficult to analyze algorithms, push the lower bound on r_3 higher.

The Smallest Clause Heuristic repeatedly executes the following. Assign true to a random literal in a random smallest length clause and delete the clause since it is now satisfied. Pick at random a 1-literal clause, if one exists, and set that literal to true. If there is no 1-literal clause, pick a 2-literal clause, select one of its two literals and set the literal to true. Otherwise, pick a 3-literal clause and a literal in it and set the literal to true. If we encounter a 0-length clause, then we have failed to find a satisfying assignment;

otherwise, we have found one.

A related heuristic, called the Unit Clause Heuristic, selects a random clause with one literal, if there is one, and sets the literal in it to true. Otherwise, it picks a random as yet unset literal and sets it to true. The "pure literal" heuristic sets a random "pure literal", a literal whose negation does not occur in any clause, to true, if there are any pure literals; otherwise, it sets a random literal to true.

When a literal w is set to true, all clauses containing w are deleted, since they are satisfied, and $\bar{w}$ is deleted from any clause containing $\bar{w}$. If a clause is reduced to length zero (no literals), then the algorithm has failed to find a satisfying assignment to the formula. The formula may, in fact, be satisfiable, but the algorithm has failed.

Example: Consider a 3-CNF formula with n variables and cn clauses. With n variables there are $2n$ literals, since a variable and its complement are distinct literals. The expected number of times a literal occurs is calculated as follows. Each clause has three literals. Thus, each of the $2n$ different literals occurs $\frac{(3cn)}{2n} = \frac{3}{2}c$ times on average. Suppose $c = 5$. Then each literal appears 7.5 times on average. If one sets a literal to true, one would expect to satisfy 7.5 clauses. However, this process is not repeatable since after setting a literal to true there is conditioning so that the formula is no longer random. ■

Theorem 4.20 *If the number of clauses in a random 3-CNF formula grows as cn where c is a constant less than 2/3, then with probability* $1 - o(1)$*, the Shortest Clause Heuristic finds a satisfying assignment.*

The proof of this theorem will take the rest of the section. A general impediment to proving that simple algorithms work for random instances of many problems is conditioning. At the start, the input is random and has properties enjoyed by random instances. But, as the algorithm is executed; the data is no longer random, it is conditioned on the steps of the algorithm so far. In the case of SC and other heuristics for finding a satisfying assignment for a Boolean formula, the argument to deal with conditioning is relatively simple.

We supply some intuition before going to the proof. Imagine maintaining a queue of 1 and 2-clauses. A 3-clause enters the queue when one of its literals is set to false and it becomes a 2-clause. SC always picks a 1 or 2-clause if there is one and sets one of its literals to true. At any step when the total number of 1 and 2-clauses is positive, one of the clauses is removed from the queue. Consider the arrival rate, the expected number of arrivals into the queue. For a particular clause to arrive into the queue at time t to become a 2-clause, it must contain the negation of the literal being set to true at time t. It can contain any two other literals not yet set. The number of such clauses is $\binom{n-t}{2}2^2$. So, the probability that a particular clause arrives in the queue at time t is at most

$$\frac{\binom{n-t}{2}2^2}{\binom{n}{3}2^3} \leq \frac{3}{2(n-2)}.$$

Since there are cn clauses in total, the arrival rate is $\frac{3c}{2}$, which for $c < 2/3$ is a constant strictly less than one. The arrivals into the queue of different clauses occur independently (Lemma 4.21), the queue has arrival rate strictly less than one, and the queue loses one or more clauses whenever it is nonempty. This implies that the queue never has too many clauses in it. A slightly more complicated argument will show that no clause remains as a 1 or 2-clause for $\Omega(\ln n)$ steps (Lemma 4.22). This implies that the probability of two contradictory 1-length clauses, which is a precursor to a 0-length clause, is very small.

Lemma 4.21 *Let T_i be the first time that clause i turns into a 2-clause. T_i is ∞ if clause i gets satisfied before turning into a 2-clause. The T_i are mutually independent and for any t,*

$$Prob(T_i = t) \leq \frac{3}{2(n-2)}.$$

Proof: For the proof, generate the clauses in a different way. The important thing is that the new method of generation, called the method of "deferred decisions", results in the same distribution of input formulae as the original. The method of deferred decisions is tied in with the SC algorithm and works as follows. At any time, the length of each clause (number of literals) is all that we know; we have not yet picked which literals are in each clause. At the start, every clause has length three and SC picks one of the clauses uniformly at random. Now, SC wants to pick one of the three literals in that clause to set to true, but we do not know which literals are in the clause. At this point, we pick uniformly at random one of the $2n$ possible literals. Say for illustration, we picked $\bar{x}_{102}$. The literal $\bar{x}_{102}$ is placed in the clause and set to true. The literal x_{102} is set to false. We must also deal with occurrences of the literal or its negation in all other clauses, but again, we do not know which clauses have such an occurrence. We decide that now. For each clause, independently, with probability $3/n$, include the variable x_{102} or $\bar{x}_{102}$ in the clause and if included, with probability 1/2, include the literal $\bar{x}_{102}$ in the clause and with the other 1/2 probability include its negation, namely, x_{102}. In either case, we decrease the residual length of the clause by one. The algorithm deletes the clause since it is satisfied and we do not care which other literals are in it. If we had included the negation of the literal instead, then we delete just that occurrence, and decrease the length of the clause by one.

At a general stage, suppose the fates of i variables have already been decided and $n-i$ remain. The residual length of each clause is known. Among the clauses that are not yet satisfied, choose a random shortest length clause. Among the $n-i$ variables remaining, pick one uniformly at random, then pick it or its negation as the new literal. Include this literal in the clause thereby satisfying it. Since the clause is satisfied, the algorithm deletes it. For each other clause, do the following. If its residual length is l, decide with probability $l/(n-i)$ to include the new variable in the clause and if so with probability 1/2 each, include it or its negation. If literal v is included in a clause, delete the clause as it is now satisfied. If $\bar{v}$ is included in a clause, then just delete the literal and decrease the residual length of the clause by one.

Why does this yield the same distribution as the original one? First, observe that the order in which the variables are picked by the method of deferred decisions is independent of the clauses; it is just a random permutation of the n variables. Look at any one clause. For a clause, we decide in order whether each variable or its negation is in the clause. So for a particular clause and a particular triple i, j, and k with $i < j < k$, the probability that the clause contains the i^{th}, the j^{th}, and k^{th} literal (or their negations) in the order determined by deferred decisions is:

$$\begin{aligned}
&\left(1-\tfrac{3}{n}\right)\left(1-\tfrac{3}{n-1}\right)\cdots\left(1-\tfrac{3}{n-i+2}\right)\tfrac{3}{n-i+1}\\
&\left(1-\tfrac{2}{n-i}\right)\left(1-\tfrac{2}{n-i-1}\right)\cdots\left(1-\tfrac{2}{n-j+2}\right)\tfrac{2}{n-j+1}\\
&\left(1-\tfrac{1}{n-j}\right)\left(1-\tfrac{1}{n-j-1}\right)\cdots\left(1-\tfrac{1}{n-k+2}\right)\tfrac{1}{n-k+1}=\tfrac{3}{n(n-1)(n-2)},
\end{aligned}$$

where the $(1-\cdots)$ factors are for not picking the current variable or negation to be included and the others are for including the current variable or its negation. Independence among clauses follows from the fact that we have never let the occurrence or nonoccurrence of any variable in any clause influence our decisions on other clauses.

Now, we prove the lemma by appealing to the method of deferred decisions to generate the formula. $T_i = t$ if and only if the method of deferred decisions does not put the current literal at steps $1, 2, \ldots, t-1$ into the i^{th} clause, but puts the negation of the literal at step t into it. Thus, the probability is precisely

$$\tfrac{1}{2}\left(1-\tfrac{3}{n}\right)\left(1-\tfrac{3}{n-1}\right)\cdots\left(1-\tfrac{3}{n-t+2}\right)\tfrac{3}{n-t+1}\leq\tfrac{3}{2(n-2)},$$

as claimed. Clearly the T_i are independent since again deferred decisions deal with different clauses independently. ■

Lemma 4.22 *With probability $1-o(1)$, no clause remains a 2 or 1-clause for $\Omega(\ln n)$ steps. I.e., once a 3-clause becomes a 2-clause, it is either satisfied or reduced to a 0-clause in $O(\ln n)$ steps.*

Proof: Without loss of generality, again focus on the first clause. Suppose it becomes a 2-clause at step s_1 and remains a 2 or 1-clause until step s. Suppose $s - s_1 \geq c_2 \ln n$. Let r be the last time before s when there are no 2 or 1-clauses at all. Since at time 0, there are no 2 or 1-clauses, r is well-defined. We have $s - r \geq c_2 \ln n$. In the interval r to s, at each step, there is at least one 2 or 1-clause. Since SC always decreases the total number of 1 and 2-clauses by one whenever it is positive, we must have generated at least $s - r$ new 2-clauses between r and s. Now, define an indicator random variable for each 3-clause which has value one if the clause turns into a 2-clause between r and s. By Lemma 4.21 these variables are independent and the probability that a particular 3-clause turns into a 2-clause at a time t is at most $3/(2(n-2))$. Summing over t between r and s,

$$\text{Prob}\left(\text{a 3-clause turns into a 2-clause during } [r,s]\right)\leq\frac{3(s-r)}{2(n-2)}.$$

Since there are cn clauses in all, the expected sum of the indicator random variables is $cn\frac{3(s-r)}{2(n-2)} \approx \frac{3c(s-r)}{2}$. Note that $3c/2 < 1$, which implies the arrival rate into the queue of 2 and 1-clauses is a constant strictly less than one. Using Chernoff bounds, the probability that more than $s-r$ clauses turn into 2-clauses between r and s is at most $o(1/n^5)$. This is for one choice of a clause, one choice of s_1 and one choice each of r and s within $O(\ln n)$ of s_1. Applying the union bound over $O(n^3)$ choices of clauses, $O(n)$ choices of s_1 and $O(\ln n)^2$ choices of r and s, we get that the probability that any clause remains a 2 or 1-clause for $\Omega(\ln n)$ steps is $o(1)$. ■

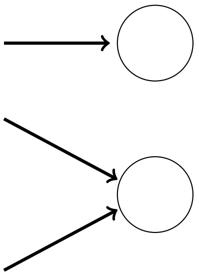
Now, suppose SC terminates in failure. At some time t, the algorithm generates a 0-clause. At time $t-1$, this clause must have been a 1-clause. Suppose the clause consists of the literal w. Since at time $t-1$, there is at least one 1-clause, the shortest clause rule of SC selects a 1-clause and sets the literal in that clause to true. This other clause must have been $\bar{w}$. Let t_1 be the first time either of these two clauses, w or $\bar{w}$, became a 2-clause. We have $t - t_1 \in O(\ln n)$. Clearly, until time t, neither of these two clauses is picked by SC. So, the literals which are set to true during this period are chosen independent of these clauses. Say the two clauses were $w + x + y$ and $\bar{w} + u + v$ at the start. x, y, u, and v must all be negations of literals set to true during steps t_1 to t. So, there are only $O\left((\ln n)^4\right)$ choices for x, y, u, and v. There are $O(n)$ choices of w, $O(n^2)$ choices of which two clauses of the input become these w and $\bar{w}$, and n choices for t_1. Thus, there are $O\left(n^4(\ln n)^4\right)$ choices for these clauses. The probability of these choices is therefore $O\left(n^4(\ln n)^4/n^6\right) = o(1)$, as required.

4.8 Nonuniform and Growth Models of Random Graphs

4.8.1 Nonuniform Models

So far we have considered the random graph $G(n,p)$ in which all vertices have the same expected degree and showed that the degree is concentrated close to its expectation. However, large graphs occurring in the real world tend to have power law degree distributions. For a power law degree distribution, the number $f(d)$ of vertices of degree d plotted as a function of d satisfies $f(d) \leq c/d^{\alpha}$, where α and c are constants.

To generate such graphs, we stipulate that there are $f(d)$ vertices of degree d and choose uniformly at random from the set of graphs with this degree distribution. Clearly, in this model the graph edges are not independent and this makes these random graphs harder to analyze. But the question of when phase transitions occur in random graphs with arbitrary degree distributions is still of interest. In this section, we consider when a random graph with a nonuniform degree distribution has a giant component. Our treatment in this section, and subsequent ones, will be more intuitive without providing rigorous proofs.



Consider a graph in which half of the vertices are degree one and half are degree two. If a vertex is selected at random, it is equally likely to be degree one or degree two. However, if we select an edge at random and walk to its endpoint, the vertex is twice as likely to be degree two as degree one. In many graph algorithms, a vertex is reached by randomly selecting an edge and traversing the edge to reach an endpoint. In this case, the probability of reaching a degree i vertex is proportional to $i\lambda_i$ where λ_i is the fraction of vertices that are degree i.

Figure 4.13: Probability of encountering a degree d vertex when following a path in a graph.

4.8.2 Giant Component in Random Graphs with Given Degree Distribution

Molloy and Reed address the issue of when a random graph with a nonuniform degree distribution has a giant component. Let λ_i be the fraction of vertices of degree i. There will be a giant component if and only if $\sum\limits_{i=0}^{\infty} i(i-2)\lambda_i > 0$.

To see intuitively that this is the correct formula, consider exploring a component of a graph starting from a given seed vertex. Degree zero vertices do not occur except in the case where the vertex is the seed. If a degree one vertex is encountered, then that terminates the expansion along the edge into the vertex. Thus, we do not want to encounter too many degree one vertices. A degree two vertex is neutral in that the vertex is entered by one edge and left by the other. There is no net increase in the size of the frontier. Vertices of degree i greater than two increase the frontier by $i-2$ vertices. The vertex is entered by one of its edges and thus there are $i-1$ edges to new vertices in the frontier for a net gain of $i-2$. The $i\lambda_i$ in $i\,(i-2)\,\lambda_i$ is proportional to the probability of reaching a degree i vertex and the $i-2$ accounts for the increase or decrease in size of the frontier when a degree i vertex is reached.

Example: Consider applying the Molloy Reed conditions to the $G(n,p)$ model. The summation $\sum_{i=0}^{n} i(i-2)p_i$ gives value zero precisely when $p = 1/n$, the point at which the phase transition occurs. At $p = 1/n$, the average degree of each vertex is one and there are $n/2$ edges. However, the actual degree distribution of the vertices is binomial, where the probability that a vertex is of degree i is given by $p_i = \binom{n}{i}p^i(1-p)^{n-i}$. We now show that $\lim\limits_{n\to\infty}\sum\limits_{i=0}^{n} i(i-2)p_i = 0$ for $p_i = \binom{n}{i}p^i(1-p)^{n-i}$ when $p = 1/n$.

$$\begin{aligned}
&\lim_{n\to\infty}\sum_{i=0}^{n} i(i-2)\binom{n}{i}\left(\frac{1}{n}\right)^{i}\left(1-\frac{1}{n}\right)^{n-i}\\
&\qquad=\lim_{n\to\infty}\sum_{i=0}^{n} i(i-2)\frac{n(n-1)\cdots(n-i+1)}{i!\;n^{i}}\left(1-\frac{1}{n}\right)^{n}\left(1-\frac{1}{n}\right)^{-i}\\
&\qquad=\frac{1}{e}\lim_{n\to\infty}\sum_{i=0}^{n} i(i-2)\frac{n(n-1)\cdots(n-i+1)}{i!\;n^{i}}\left(\frac{n}{n-1}\right)^{i}\\
&\qquad\le\sum_{i=0}^{\infty}\frac{i(i-2)}{i!}.
\end{aligned}$$

To see that $\sum\limits_{i=0}^{\infty}\frac{i(i-2)}{i!}=0$, note that

$$\sum_{i=0}^{\infty}\frac{i}{i!}=\sum_{i=1}^{\infty}\frac{i}{i!}=\sum_{i=1}^{\infty}\frac{1}{(i-1)!}=\sum_{i=0}^{\infty}\frac{1}{i!}$$

and

$$\sum_{i=0}^{\infty}\frac{i^2}{i!}=\sum_{i=1}^{\infty}\frac{i}{(i-1)!}=\sum_{i=0}^{\infty}\frac{i+1}{i!}=\sum_{i=0}^{\infty}\frac{i}{i!}+\sum_{i=0}^{\infty}\frac{1}{i!}=2\sum_{i=0}^{\infty}\frac{1}{i!}.$$

Thus,

$$\sum_{i=0}^{\infty}\tfrac{i(i-2)}{i!}=\sum_{i=0}^{\infty}\tfrac{i^2}{i!}-2\sum_{i=0}^{\infty}\tfrac{i}{i!}=0.$$

■

4.9 Growth Models

4.9.1 Growth Model Without Preferential Attachment

Many graphs that arise in the outside world started as small graphs that grew over time. In a model for such graphs, vertices and edges are added to the graph over time. In such a model there are many ways in which to select the vertices for attaching a new edge. One is to select two vertices uniformly at random from the set of existing vertices. Another is to select two vertices with probability proportional to their degree. This latter method is referred to as preferential attachment. A variant of this method would be to add a new vertex at each unit of time and with probability δ add an edge where one end of the edge is the new vertex and the other end is a vertex selected with probability proportional to its degree. The graph generated by this latter method is a tree.

Consider a growth model for a random graph without preferential attachment. Start with zero vertices at time zero. At each unit of time a new vertex is created and with probability δ, two vertices chosen at random are joined by an edge. The two vertices may already have an edge between them. In this case, we will add another edge. So, the resulting structure is a multi-graph, rather then a graph. But since at time t, there are t vertices and in expectation only $O(\delta t)$ edges where there are t^2 pairs of vertices, it is very unlikely that there will be multiple edges.

The degree distribution for this growth model is calculated as follows. The number of vertices of degree k at time t is a random variable. Let $d_k(t)$ be the expectation of the number of vertices of degree k at time t. The number of isolated vertices increases by one at each unit of time and decreases by the number of isolated vertices, $b(t)$, that are picked to be end points of the new edge. $b(t)$ can take on values 0,1, or 2. Taking expectations,

$$d_0(t+1) = d_0(t) + 1 - E(b(t)).$$

Now $b(t)$ is the sum of two 0-1 valued random variables whose values are the number of degree zero vertices picked for each end point of the new edge. Even though the two random variables are not independent, the expectation of $b(t)$ is the sum of the expectations of the two variables and is $2\delta\frac{d_0(t)}{t}$. Thus,

$$d_0(t+1) = d_0(t) + 1 - 2\delta\frac{d_0(t)}{t}.$$

The number of degree k vertices increases whenever a new edge is added to a degree $k-1$ vertex and decreases when a new edge is added to a degree k vertex. Reasoning as above,

$$d_k\left(t+1\right) = d_k(t) + 2\delta\frac{d_{k-1}(t)}{t} - 2\delta\frac{d_k(t)}{t}. \tag{4.4}$$

Note that this formula, as others in this section, is not quite precise. For example, the same vertex may be picked twice, so that the new edge is a self-loop. For $k << t$, this problem contributes a minuscule error. Restricting k to be a fixed constant and letting $t \to \infty$ in this section avoids these problems.

Assume that the above equations are exactly valid. Clearly, $d_0(1) = 1$ and $d_1(1) = d_2(1) = \cdots = 0$. By induction on t, there is a unique solution to (4.4), since given $d_k(t)$ for all k, the equation determines $d_k(t+1)$ for all k. There is a solution of the form $d_k(t) = p_k t$, where p_k depends only on k and not on t, provided k is fixed and $t \to \infty$. Again, this is not precisely true, $d_1(1) = 0$ and $d_1(2) > 0$ clearly contradict the existence of a solution of the form $d_1(t) = p_1 t$.

Set $d_k(t) = p_k t$. Then,

$$\begin{aligned}(t+1)\,p_0 &= p_0 t + 1 - 2\delta\frac{p_0 t}{t}\\ p_0 &= 1 - 2\delta p_0\end{aligned}$$

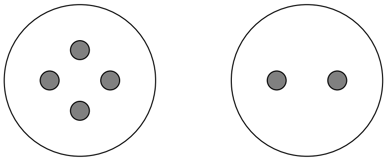


Figure 4.14: In selecting a component at random, each of the two components is equally likely to be selected. In selecting the component containing a random vertex, the larger component is twice as likely to be selected.

$$p_0 = \frac{1}{1+2\delta}$$

and

$$\begin{aligned}(t+1)\,p_k &= p_k t + 2\delta\frac{p_{k-1}t}{t} - 2\delta\frac{p_k t}{t}\\ p_k &= 2\delta p_{k-1} - 2\delta p_k\end{aligned}$$

$$\begin{aligned}p_k &= \frac{2\delta}{1+2\delta}p_{k-1}\\ &= \left(\frac{2\delta}{1+2\delta}\right)^k p_0\\ &= \frac{1}{1+2\delta}\left(\frac{2\delta}{1+2\delta}\right)^k. \qquad (4.5)\end{aligned}$$

Thus, the model gives rise to a graph with a degree distribution that falls off exponentially fast with degree.

The generating function for component size

Let $n_k(t)$ be the expected number of components of size k at time t. Then $n_k(t)$ is proportional to the probability that a randomly picked component is of size k. This is not the same as picking the component containing a randomly selected vertex (see Figure 4.14). Indeed, the probability that the size of the component containing a randomly selected vertex is k is proportional to $kn_k(t)$. We will show that there is a solution for $n_k(t)$ of the form $a_k t$ where a_k is a constant independent of t. After showing this, we focus on the generating function $g(x)$ for the numbers $ka_k(t)$ and use $g(x)$ to find the threshold for giant components.

Consider $n_1(t)$, the expected number of isolated vertices at time t. At each unit of time, an isolated vertex is added to the graph and an expected $\frac{2\delta n_1(t)}{t}$ many isolated vertices are chosen for attachment and thereby leave the set of isolated vertices. Thus,

$$n_1(t+1) = n_1(t) + 1 - 2\delta\frac{n_1(t)}{t}.$$

For k >1, $n_k(t)$ increases when two smaller components whose sizes sum to k are joined by an edge and decreases when a vertex in a component of size k is chosen for attachment. The probability that a vertex selected at random will be in a size k component is $\frac{kn_k(t)}{t}$. Thus,

$$n_k(t+1) = n_k(t) + \delta \sum_{j=1}^{k-1} \frac{jn_j(t)}{t} \frac{(k-j)n_{k-j}(t)}{t} - 2\delta \frac{kn_k(t)}{t}.$$

To be precise, one needs to consider the actual number of components of various sizes, rather than the expected numbers. Also, if both vertices at the end of the edge are in the same k-vertex component, then $n_k(t)$ does not go down as claimed. These small inaccuracies can be ignored.

Consider solutions of the form $n_k(t) = a_k t$. Note that $n_k(t) = a_k t$ implies the number of vertices in a connected component of size k is $ka_k t$. Since the total number of vertices at time t is t, ka_k is the probability that a random vertex is in a connected component of size k. The recurrences here are valid only for k fixed as $t \to \infty$. So $\sum_{k=0}^{\infty} ka_k$ may be less than 1, in which case, there are nonfinite size components whose sizes are growing with t. Solving for a_k yields $a_1 = \frac{1}{1+2\delta}$ and $a_k = \frac{\delta}{1+2k\delta} \sum\limits_{j=1}^{k-1} j(k-j)a_j a_{k-j}$.

Consider the generating function $g(x)$ for the distribution of component sizes where the coefficient of x^k is the probability that a vertex chosen at random is in a component of size k.

$$g(x) = \sum_{k=1}^{\infty} ka_k x^k.$$

Now, $g(1) = \sum_{k=0}^{\infty} ka_k$ is the probability that a randomly chosen vertex is in a finite sized component. For $\delta = 0$, this is clearly one, since all vertices are in components of size one. On the other hand, for $\delta = 1$, the vertex created at time one has expected degree $\log n$, so it is in a nonfinite size component. This implies that for $\delta = 1$, $g(1) < 1$ and there is a nonfinite size component. Assuming continuity, there is a $\delta_{critical}$ above which $g(1) < 1$. From the formula for the $a_i's$, we will derive the differential equation

$$g = -2\delta x g' + 2\delta x g g' + x$$

and then use the equation for g to determine the value of δ at which the phase transition for the appearance of a nonfinite sized component occurs.

Derivation of $g(x)$

From

$$a_1 = \frac{1}{1+2\delta}$$

and

$$a_k = \frac{\delta}{1+2k\delta} \sum_{j=1}^{k-1} j(k-j) a_j a_{k-j}$$

derive the equations

$$a_1\left(1+2\delta\right)-1=0$$

and

$$a_k\left(1+2k\delta\right)=\delta\sum_{j=1}^{k-1}j(k-j)a_ja_{k-j}$$

for $k\geq 2$. The generating function is formed by multiplying the k^{th} equation by kx^k and summing over all k. This gives

$$-x+\sum_{k=1}^{\infty}ka_kx^k+2\delta x\sum_{k=1}^{\infty}a_kk^2x^{k-1}=\delta\sum_{k=1}^{\infty}kx^k\sum_{j=1}^{k-1}j(k-j)a_ja_{k-j}.$$

Note that

$$g(x)=\sum_{k=1}^{\infty}ka_kx^k \text{ and } g'(x)=\sum_{k=1}^{\infty}a_kk^2x^{k-1}.$$

Thus,

$$-x+g(x)+2\delta xg'(x)=\delta\sum_{k=1}^{\infty}kx^k\sum_{j=1}^{k-1}j(k-j)a_ja_{k-j}.$$

Working with the right hand side

$$\delta\sum_{k=1}^{\infty}kx^k\sum_{j=1}^{k-1}j(k-j)a_ja_{k-j}=\delta x\sum_{k=1}^{\infty}\sum_{j=1}^{k-1}j(k-j)(j+k-j)x^{k-1}a_ja_{k-j}.$$

Now breaking the $j+k-j$ into two sums gives

$$\delta x\sum_{k=1}^{\infty}\sum_{j=1}^{k-1}j^2a_jx^{j-1}(k-j)a_{k-j}x^{k-j}+\delta x\sum_{k=1}^{\infty}\sum_{j=1}^{k-1}ja_jx^j(k-j)^2a_{k-j}x^{k-j-1}.$$

Notice that the second sum is obtained from the first by substituting $k-j$ for j and that both terms are $\delta xg'g$. Thus,

$$-x+g(x)+2\delta xg'(x)=2\delta xg'(x)g(x).$$

Hence,

$$g'=\frac{1}{2\delta}\frac{1-\frac{g}{x}}{1-g}.$$

Phase transition for nonfinite components

The generating function $g(x)$ contains information about the finite components of the graph. A finite component is a component of size $1,2,\ldots$ which does not depend on t.

Observe that $g(1) = \sum\limits_{k=0}^{\infty} ka_k$ and hence $g(1)$ is the probability that a randomly chosen vertex will belong to a component of finite size. If $g(1) = 1$ there are no nonfinite components. When $g(1) \neq 1$, then $1 - g(1)$ is the expected fraction of the vertices that are in nonfinite components. Potentially, there could be many such nonfinite components. But an argument similar to Part 3 of Theorem 4.9 concludes that two fairly large components would merge into one. Suppose there are two connected components at time t, each of size at least $t^{4/5}$. Consider the earliest created $\frac{1}{2}t^{4/5}$ vertices in each part. These vertices must have lived for at least $\frac{1}{2}t^{4/5}$ time after creation. At each time, the probability of an edge forming between two such vertices, one in each component, is at least $\delta\Omega(t^{-2/5})$ and so the probability that no such edge formed is at most $\left(1 - \delta t^{-2/5}\right)^{t^{4/5}/2} \leq e^{-\Omega(\delta t^{2/5})} \to 0$. So with high probability, such components would have merged into one. But this still leaves open the possibility of many components of size t^{ε}, $(\ln t)^2$, or some other slowly growing function of t.

We now calculate the value of δ at which the phase transition for a nonfinite component occurs. Recall that the generating function for $g\left(x\right)$ satisfies

$$g'\left(x\right) = \frac{1}{2\delta} \frac{1 - \frac{g(x)}{x}}{1 - g\left(x\right)}.$$

If δ is greater than some $\delta_{critical}$, then $g(1) \neq 1$. In this case the above formula simplifies with $1 - g(1)$ canceling from the numerator and denominator, leaving just $\frac{1}{2\delta}$. Since ka_k is the probability that a randomly chosen vertex is in a component of size k, the average size of the finite components is $g'(1) = \sum\limits_{k=1}^{\infty} k^2 a_k$. Now, $g'(1)$ is given by

$$g'(1) = \frac{1}{2\delta} \tag{4.6}$$

for all δ greater than $\delta_{critical}$. If δ is less than $\delta_{critical}$, then all vertices are in finite components. In this case $g(1) = 1$ and both the numerator and the denominator approach zero. Appling L'Hopital's rule

$$\lim_{x \to 1} g'(x) = \tfrac{1}{2\delta} \left. \frac{\frac{xg'(x) - g(x)}{x^2}}{g'(x)} \right|_{x=1}$$

or

$$(g'(1))^2 = \tfrac{1}{2\delta}\left(g'(1) - g(1)\right).$$

The quadratic $(g'(1))^2 - \frac{1}{2\delta}g'(1) + \frac{1}{2\delta}g(1) = 0$ has solutions

$$g'(1) = \frac{\frac{1}{2\delta} \pm \sqrt{\frac{1}{4\delta^2} - \frac{4}{2\delta}}}{2} = \frac{1 \pm \sqrt{1 - 8\delta}}{4\delta}. \tag{4.7}$$

The two solutions given by (4.7) become complex for $\delta > 1/8$ and thus can be valid only for $0 \le \delta \le 1/8$. For $\delta > 1/8$, the only solution is $g'(1) = \frac{1}{2\delta}$ and a nonfinite component exists. As δ is decreased, at $\delta = 1/8$ there is a singular point where for $\delta < 1/8$ there are three possible solutions, one from (4.6) which implies a giant component and two from (4.7) which imply no giant component. To determine which one of the three solutions is valid, consider the limit as $\delta \to 0$. In the limit all components are of size one since there are no edges. Only (4.7) with the minus sign gives the correct solution

$$g'(1) = \frac{1-\sqrt{1-8\delta}}{4\delta} = \frac{1-\left(1-\frac{1}{2}8\delta-\frac{1}{4}64\delta^2+\cdots\right)}{4\delta} = 1+4\delta+\cdots = 1.$$

In the absence of any nonanalytic behavior in the equation for $g'(x)$ in the region $0 \le \delta < 1/8$, we conclude that (4.7) with the minus sign is the correct solution for $0 \le \delta < 1/8$ and hence the critical value of δ for the phase transition is $1/8$. As we shall see, this is different from the static case.

As the value of δ is increased, the average size of the finite components increase from one to

$$\left.\frac{1-\sqrt{1-8\delta}}{4\delta}\right|_{\delta=1/8} = 2$$

when δ reaches the critical value of $1/8$. At $\delta = 1/8$, the average size of the finite components jumps to $\left.\frac{1}{2\delta}\right|_{\delta=1/8} = 4$ and then decreases as $\frac{1}{2\delta}$ as the giant component swallows up the finite components starting with the larger components.

Comparison to static random graph

Consider a static random graph with the same degree distribution as the graph in the growth model. Again let p_k be the probability of a vertex being of degree k. From (4.5)

$$p_k = \frac{(2\delta)^k}{(1+2\delta)^{k+1}}\,.$$

Recall the Molloy Reed analysis of random graphs with given degree distributions which asserts that there is a phase transition at $\sum\limits_{i=0}^{\infty} i(i-2)p_i = 0$. Using this, it is easy to see that a phase transition occurs for $\delta = 1/4$. For $\delta = 1/4$,

$$p_k = \tfrac{(2\delta)^k}{(1+2\delta)^{k+1}} = \frac{\left(\frac{1}{2}\right)^k}{\left(1+\frac{1}{2}\right)^{k+1}} = \frac{\left(\frac{1}{2}\right)^k}{\frac{3}{2}\left(\frac{3}{2}\right)^k} = \tfrac{2}{3}\left(\tfrac{1}{3}\right)^k$$

and

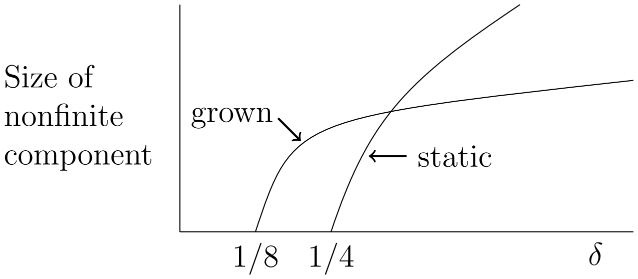


Figure 4.15: Comparison of the static random graph model and the growth model. The curve for the growth model is obtained by integrating g'.

$$\sum_{i=0}^{\infty} i(i-2)\tfrac{2}{3}\left(\tfrac{1}{3}\right)^i = \tfrac{2}{3}\sum_{i=0}^{\infty} i^2\left(\tfrac{1}{3}\right)^i - \tfrac{4}{3}\sum_{i=0}^{\infty} i\left(\tfrac{1}{3}\right)^i = \tfrac{2}{3}\times\tfrac{3}{2} - \tfrac{4}{3}\times\tfrac{3}{4} = 0.$$

Recall that $1+a+a^2+\cdots = \frac{1}{1-a}$, $a+2a^2+3a^3\cdots = \frac{a}{(1-a)^2}$, and $a+4a^2+9a^3\cdots = \frac{a(1+a)}{(1-a)^3}$.

See references at end of the chapter for calculating the size S_{static} of the giant component in the static graph. The result is

$$S_{static} = \begin{cases} 0 & \delta \le \frac{1}{4} \\ 1 - \frac{1}{\delta+\sqrt{\delta^2+2\delta}} & \delta > \frac{1}{4} \end{cases}$$

4.9.2 Growth Model With Preferential Attachment

Consider a growth model with preferential attachment. At each time unit, a vertex is added to the graph. Then with probability δ, an edge is attached to the new vertex and to a vertex selected at random with probability proportional to its degree. This model generates a tree with a power law distribution.

Let $d_i(t)$ be the expected degree of the i^{th} vertex at time t. The sum of the degrees of all vertices at time t is $2\delta t$ and thus the probability that an edge is connected to vertex i at time t is $\frac{d_i(t)}{2\delta t}$. The degree of vertex i is governed by the equation

$$\frac{\partial}{\partial t} d_i(t) = \delta \frac{d_i\left(t\right)}{2\delta t} = \frac{d_i(t)}{2t}$$

where δ is the probability that an edge is added at time t and $\frac{d_i(t)}{2\delta t}$ is the probability that the vertex i is selected for the end point of the edge.

The two in the denominator governs the solution which is of the form $at^{\frac{1}{2}}$. The value of a is determined by the initial condition $d_i\left(t\right) = \delta$ at $t = i$. Thus, $\delta = ai^{\frac{1}{2}}$ or $a = \delta i^{-\frac{1}{2}}$.

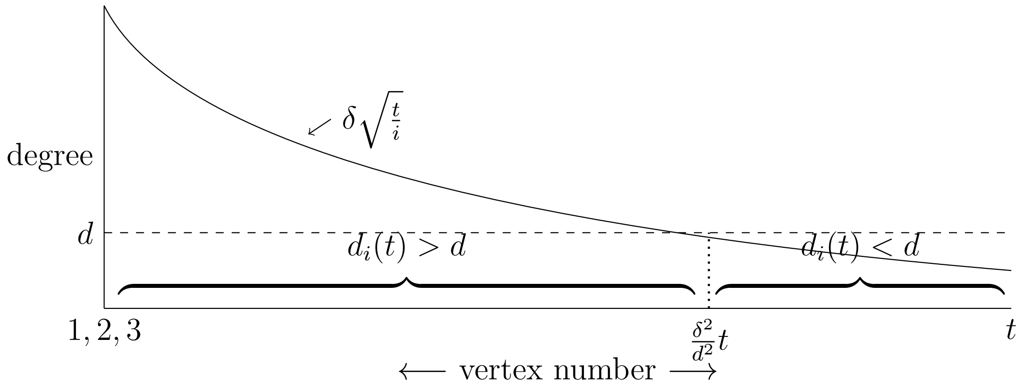


Figure 4.16: Illustration of degree of i^{th} vertex at time t. At time t, vertices numbered 1 to $\frac{\delta^2}{d^2}t$ have degrees greater than d.

Hence, $d_i(t) = \delta\sqrt{\frac{t}{i}}$.

Next, we determine the probability distribution of vertex degrees. Now, $d_i(t)$ is less than d provided $i > \frac{\delta^2}{d^2}t$. The fraction of the t vertices at time t for which $i > \frac{\delta^2}{d^2}t$ and thus that the degree is less than d is $1 - \frac{\delta^2}{d^2}$. Hence, the probability that a vertex has degree less than d is $1 - \frac{\delta^2}{d^2}$. The probability density $P(d)$ satisfies

$$\int_0^d P(d)\partial d = \text{Prob}(\text{degree} < d) = 1 - \frac{\delta^2}{d^2}$$

and can be obtained from the derivative of Prob(degree $< d$).

$$P(d) = \frac{\partial}{\partial d}\left(1 - \frac{\delta^2}{d^2}\right) = 2\frac{\delta^2}{d^3},$$

a power law distribution.

4.10 Small World Graphs

In the 1960's, Stanley Milgram carried out an experiment that indicated that any two individuals in the United States were connected by a short sequence of acquaintances. Milgram would ask a source individual, say in Nebraska, to start a letter on its journey to a target individual in Massachusetts. The Nebraska individual would be given basic information about the target including his address and occupation and asked to send the letter to someone he knew on a first name basis, who was closer to the target individual, in order to transmit the letter to the target in as few steps as possible. Each person receiving the letter would be given the same instructions. In successful experiments, it would take on average five to six steps for a letter to reach its target. This research generated the phrase "six degrees of separation" along with substantial research in social science on the interconnections between people. Surprisingly, there was no work on how to find the short paths using only local information.

In many situations, phenomena are modeled by graphs whose edges can be partitioned into local and long distance. We adopt a simple model of a directed graph due to Kleinberg, having local and long distance edges. Consider a 2-dimensional $n \times n$ grid where each vertex is connected to its four adjacent vertices. In addition to these local edges, there is one long distance edge out of each vertex. The probability that the long distance edge from vertex u terminates at v, $v \neq u$, is a function of the distance $d(u, v)$ from u to v. Here distance is measured by the shortest path consisting only of local grid edges. The probability is proportional to $1/d^r\,(u, v)$ for some constant r. This gives a one parameter family of random graphs. For r equal zero, $1/d^0\,(u, v) = 1$ for all u and v and thus the end of the long distance edge at u is uniformly distributed over all vertices independent of distance. As r increases the expected length of the long distance edge decreases. As r approaches infinity, there are no long distance edges and thus no paths shorter than that of the lattice path. What is interesting is that for r less than two, there are always short paths, but no local algorithm to find them. A local algorithm is an algorithm that is only allowed to remember the source, the destination, and its current location and can query the graph to find the long-distance edge at the current location. Based on this information, it decides the next vertex on the path.

The difficulty is that for $r < 2$, the end points of the long distance edges tend to be uniformly distributed over the vertices of the grid. Although short paths exist, it is unlikely on a short path to encounter a long distance edge whose end point is close to the destination. When r equals two, there are short paths and the simple algorithm that always selects the edge that ends closest to the destination will find a short path. For r greater than two, again there is no local algorithm to find a short path. Indeed, with high probability, there are no short paths at all.

The probability that the long distance edge from u goes to v is proportional to $d^{-r}(u, v)$. Note that the constant of proportionality will vary with the vertex u depending on where u is relative to the border of the $n \times n$ grid. However, the number of vertices at distance exactly k from u is at most $4k$ and for $k \leq n/2$ is at least k. Let $c_r(u) = \sum_v d^{-r}(u, v)$ be the normalizing constant. It is the inverse of the constant of proportionality.

For $r > 2$, $c_r(u)$ is lower bounded by

$$c_r(u) = \sum_v d^{-r}(u, v) \geq \sum_{k=1}^{n/2} (k) k^{-r} = \sum_{k=1}^{n/2} k^{1-r} \geq 1.$$

No matter how large r is the first term of $\sum_{k=1}^{n/2} k^{1-r}$ is at least one.

$r > 2$ The lengths of long distance edges tend to be short so the probability of encountering a sufficiently long, long-distance edge is too low.

$r = 2$ Selecting the edge with end point closest to the destination finds a short path.

$r < 2$ The ends of long distance edges tend to be uniformly distributed. Short paths exist but a polylog length path is unlikely to encounter a long distance edge whose end point is close to the destination.

Figure 4.17: Effects of different values of r on the expected length of long distance edges and the ability to find short paths.

For $r = 2$ the normalizing constant $c_r(u)$ is upper bounded by

$$c_r(u) = \sum_v d^{-r}(u,v) \le \sum_{k=1}^{2n} (4k)k^{-2} \le 4\sum_{k=1}^{2n} \frac{1}{k} = \theta(\ln n).$$

For $r < 2$, the normalizing constant $c_r(u)$ is lower bounded by

$$c_r(u) = \sum_v d^{-r}(u,v) \ge \sum_{k=1}^{n/2} (k)k^{-r} \ge \sum_{k=n/4}^{n/2} k^{1-r}.$$

The summation $\sum\limits_{k=n/4}^{n/2} k^{1-r}$ has $\frac{n}{4}$ terms, the smallest of which is $\left(\frac{n}{4}\right)^{1-r}$ or $\left(\frac{n}{2}\right)^{1-r}$ depending on whether r is greater or less than one. This gives the following lower bound on $c_r(u)$.

$$c_r(u) \ge \frac{n}{4}\omega(n^{1-r}) = \omega(n^{2-r}).$$

No short paths exist for the $r > 2$ case.

For $r > 2$, we first show that for at least one half the pairs of vertices there is no short path between them. We begin by showing that the expected number of edges of length greater than $n^{\frac{r+2}{2r}}$ goes to zero. The probability of an edge from u to v is $d^{-r}(u,v)/c_r(u)$ where $c_r(u)$ is lower bounded by a constant. Thus, the probability that a long edge is of length greater than or equal to $n^{\frac{r+2}{2r}}$ is upper bounded by some constant c times $\left(n^{\frac{r+2}{2r}}\right)^{-r}$ or $cn^{-(\frac{r+2}{2})}$. Since there are n^2 long edges, the expected number of edges of length at least $n^{\frac{r+2}{2r}}$ is at most $cn^2n^{-\frac{(r+2)}{2}}$ or $cn^{\frac{2-r}{2}}$, which for $r > 2$ goes to zero. Thus, by the first moment method, almost surely, there are no such edges.

For at least one half of the pairs of vertices, the grid distance, measured by grid edges between the vertices, is greater than or equal to $n/4$. Any path between them must have at least $\frac{1}{4}n/n^{\frac{r+2}{2r}} = \frac{1}{4}n^{\frac{r-2}{2r}}$ edges since there are no edges longer than $n^{\frac{r+2}{2r}}$ and so there is no polylog length path.

An algorithm for the $r = 2$ case

For $r = 2$, the local algorithm that selects the edge that ends closest to the destination t finds a path of expected length $O(\ln n)^3$. Suppose the algorithm is at a vertex u which is a at distance k from t. Then within an expected $O(\ln n)^2$ steps, the algorithm reaches a point at distance at most $k/2$. The reason is that there are $\Omega(k^2)$ vertices at distance at most $k/2$ from t. Each of these vertices is at distance at most $k + k/2 = O(k)$ from u. See Figure 4.18. Recall that the normalizing constant c_r is upper bounded by $O(\ln n)$, and hence, the constant of proportionality is lower bounded by some constant times $1/\ln n$. Thus, the probability that the long-distance edge from u goes to one of these vertices is at least

$$\Omega(k^2 k^{-r}/\ln n) = \Omega(1/\ln n).$$

Consider $\Omega(\ln n)^2$ steps of the path from u. The long-distance edges from the points visited at these steps are chosen independently and each has probability $\Omega(1/\ln n)$ of reaching within $k/2$ of t. The probability that none of them does is

$$\big(1 - \Omega(1/\ln n)\big)^{c(\ln n)^2} = c_1 e^{-\ln n} = \frac{c_1}{n}$$

for a suitable choice of constants. Thus, the distance to t is halved every $O(\ln n)^2$ steps and the algorithm reaches t in an expected $O(\ln n)^3$ steps.

A local algorithm cannot find short paths for the $r < 2$ case

For $r < 2$ no local polylog time algorithm exists for finding a short path. To illustrate the proof, we first give the proof for the special case $r = 0$, and then give the proof for $r < 2$.

When $r = 0$, all vertices are equally likely to be the end point of a long distance edge. Thus, the probability of a long distance edge hitting one of the n vertices that are within distance $\sqrt{n}$ of the destination is $1/n$. Along a path of length $\sqrt{n}$, the probability that the path does not encounter such an edge is $(1 - 1/n)^{\sqrt{n}}$. Now,

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^{\sqrt{n}} = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^{n\frac{1}{\sqrt{n}}} = \lim_{n\to\infty} e^{-\frac{1}{\sqrt{n}}} = 1.$$

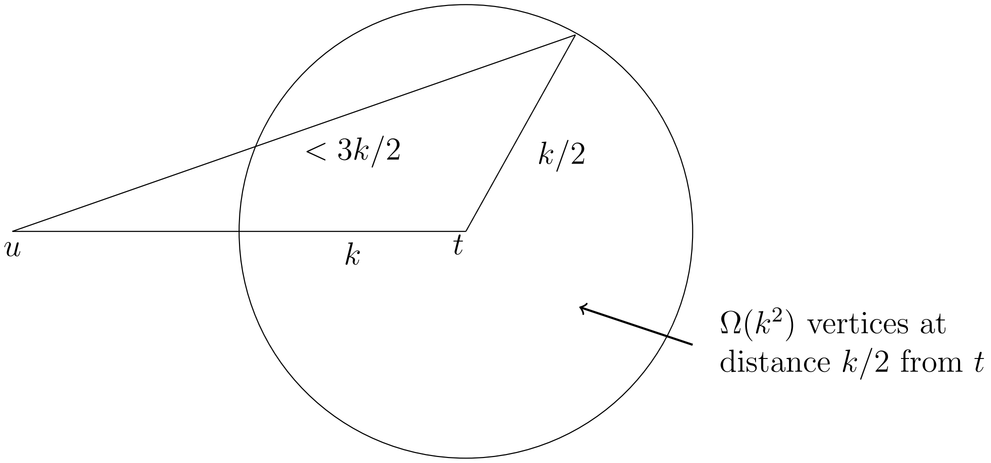


Figure 4.18: Small worlds.

Since with probability 1/2 the starting point is at distance at least $n/4$ from the destination and in $\sqrt{n}$ steps, the path will not encounter a long distance edge ending within distance $\sqrt{n}$ of the destination, for at least half of the starting points the path length will be at least $\sqrt{n}$. Thus, the expected time is at least $\frac{1}{2}\sqrt{n}$ and hence not in polylog time.

For the general $r < 2$ case, we show that a local algorithm cannot find paths of length $O(n^{(2-r)/4})$. Let $\delta = (2-r)/4$ and suppose the algorithm finds a path with at most n^δ edges. There must be a long-distance edge on the path which terminates within distance n^δ of t; otherwise, the path would end in n^δ grid edges and would be too long. There are $O(n^{2\delta})$ vertices within distance n^δ of t and the probability that the long distance edge from one vertex of the path ends at one of these vertices is at most $n^{2\delta}\left(\frac{1}{n^{2-r}}\right) = n^{(r-2)/2}$. To see this, recall that the lower bound on the normalizing constant is $\theta(n^{2-r})$ and hence an upper bound on the probability of a long distance edge hitting v is $\theta\left(\frac{1}{n^{2-r}}\right)$ independent of where v is. Thus, the probability that the long distance edge from one of the n^δ vertices on the path hits any one of the $n^{2\delta}$ vertices within distance n^δ of t is $n^{2\delta}\frac{1}{n^{2-r}} = n^{\frac{r-2}{2}}$. The probability that this happens for any one of the n^δ vertices on the path is at most $n^{\frac{r-2}{2}}n^\delta = n^{\frac{r-2}{2}}n^{\frac{2-r}{4}} = n^{(r-2)/4} = o(1)$ as claimed.

Short paths exist for $r < 2$

Finally we show for $r < 2$ that there are $O(\ln n)$ length paths between s and t. The proof is similar to the proof of Theorem 4.17 showing $O(\ln n)$ diameter for $G(n, p)$ when p is $\Omega(\ln n/n)$, so we do not give all the details here. We give the proof only for the case when $r = 0$.

For a particular vertex v, let S_i denote the set of vertices at distance i from v. Using only local edges, if i is $O(\sqrt{\ln n})$, then $|S_i|$ is $\Omega(\ln n)$. For later i, we argue a constant factor growth in the size of S_i as in Theorem 4.17. As long as $|S_1|+|S_2|+\cdots+|S_i| \leq n^2/2$, for each of the $n^2/2$ or more vertices outside, the probability that the vertex is not in S_{i+1} is $(1-\frac{1}{n^2})^{|S_i|} \leq 1-\frac{|S_i|}{2n^2}$ since the long-distance edge from each vertex of S_i chooses a long-distance neighbor at random. So, the expected size of S_{i+1} is at least $|S_i|/4$ and using Chernoff, we get constant factor growth up to $n^2/2$. Thus, for any two vertices v and w, the number of vertices at distance $O(\ln n)$ from each is at least $n^2/2$. Any two sets of cardinality at least $n^2/2$ must intersect giving us a $O(\ln n)$ length path from v to w.

4.11 Bibliographic Notes

The $G(n,p)$ random graph model is from Erdös Rényi [ER60]. Among the books written on properties of random graphs a reader may wish to consult Palmer [Pal85], Jansen, Luczak and Ruciński [JLR00],or Bollobás [Bol01]. Material on phase transitions can be found in [BT87]. The work on phase transitions for CNF was started by Chao and Franco [CF86]. Further work was done in [FS96], [AP03], [Fri99], and others. The proof here that the SC algorithm produces a solution when the number of clauses is cn for $c < \frac{2}{3}$ is from [Chv92].

For material on the giant component consult [Kar90] or [JKLP93]. Material on branching process can be found in [AN72]. The phase transition for giant components in random graphs with given degree distributions is from Molloy and Reed [MR95a].

There are numerous papers on growth models. The material in this chapter was based primarily on [CHK$^+$] and [BA]. The material on small world is based on Kleinberg, [Kle00] which follows earlier work by Watts and Strogatz [WS98a].

4.12 Exercises

Exercise 4.1 *Search the World Wide Web to find some real world graphs in machine readable form or data bases that could automatically be converted to graphs.*

1. *Plot the degree distribution of each graph.*
2. *Compute the average degree of each graph.*
3. *Count the number of connected components of each size in each graph.*
4. *Describe what you find.*
5. *What is the average vertex degree? If the graph were a $G(n, p)$ graph, what would the value of p be?*
6. *Spot differences between your graph and $G(n, p)$ for p from the last part. [Look at sizes of connected components, cycles.]*

Exercise 4.2 *In $G(n, p)$ the probability of a vertex having degree k is $\binom{n}{k}p^k(1-p)^{n-k}$.*

1. *Show by direct calculation that the expected degree is np.*
2. *Compute directly the variance of the distribution.*
3. *Where is the mode of the binomial distribution for a given value of p? The mode is the point at which the probability is maximum.*

Exercise 4.3

1. *Plot the degree distribution for $G(1000, 0.003)$.*
2. *Plot the degree distribution for $G(1000, 0.030)$.*

Exercise 4.4 *To better understand the binomial distribution plot $\binom{n}{k}p^k(1-p)^{n-k}$ as a function of k for $n = 50$ and $k = 0.05, 0.5, 0.95$. For each value of p check the sum over all k to ensure that the sum is one.*

Exercise 4.5 *In $G\left(n, \frac{1}{n}\right)$, what is the probability that there is a vertex of degree $\log n$? Give an exact formula; also derive simple approximations.*

Exercise 4.6 *The example of Section 4.1.1 showed that if the degrees in $G(n, \frac{1}{n})$ were independent there would almost surely be a vertex of degree $\log n / \log\log n$. However, the degrees are not independent. Show how to overcome this difficulty.*

Exercise 4.7 *Let $f(n)$ be a function that is asymptotically less than n. Some such functions are $1/n$, a constant d, $\log n$ or $n^{\frac{1}{3}}$. Show that*

$$\left(1+\tfrac{f(n)}{n}\right)^n \simeq e^{f(n)}.$$

for large n. That is

$$\lim_{n\to\infty} \frac{\left(1+\frac{f(n)}{n}\right)^n}{e^{f(n)}} = 1.$$

Exercise 4.8

1. *In the limit as n goes to infinity, how does $\left(1-\frac{1}{n}\right)^{n\ln n}$ behave.*
2. *What is $\lim\limits_{n\to\infty} \left(\frac{n+1}{n}\right)^n$?*

Exercise 4.9 *Consider a random permutation of the integers 1 to n. The integer i is said to be a fixed point of the permutation if i is the integer in the i^{th} position of the permutation. Use indicator variables to determine the expected number of fixed points in a random permutation.*

Exercise 4.10 *Generate a graph $G\left(n, \frac{d}{n}\right)$ with $n = 1000$ and d=2, 3, and 6. Count the number of triangles in each graph. Try the experiment with n=100.*

Exercise 4.11 *What is the expected number of squares (4-cycles) in $G\left(n, \frac{d}{n}\right)$? What is the expected number of 4-cliques in $G\left(n, \frac{d}{n}\right)$?*

Exercise 4.12 *Carry out an argument, similar to the one used for triangles, to show that $p = \frac{1}{n^{2/3}}$ is a threshold for the existence of a 4-clique. A 4-clique consists of four vertices with all $\binom{4}{2}$ edges present.*

Exercise 4.13 *What is the expected number of paths of length 3, $\log n$, $\sqrt{n}$, and $n-1$ in $G(n, \frac{d}{n})$? The expected number of paths of a given length being infinite does not imply that a graph selected at random has such a path.*

Exercise 4.14 *Let x be an integer chosen uniformly at random from $\{1, 2, \ldots, n\}$. Count the number of distinct prime factors of n. The exercise is to show that the number of prime factors almost surely is $\Theta(\ln\ln n)$. Let p stand for a prime number between 2 and n.*

1. *For each fixed prime p, let I_p be the indicator function of the event that p divides x. Show that $E(I_p) = \frac{1}{p} + O\left(\frac{1}{n}\right)$. It is known that $\sum\limits_{p\le n} \frac{1}{p} = \ln\ln n$ and you may assume this.*
2. *The random variable of interest, $y = \sum\limits_{p} I_p$, is the number of prime divisors of x picked at random. Show that the variance of y is $O(\ln\ln n)$. For this, assume the known result that the number of primes up to n is $O(n/\ln n)$. To bound the variance of y, think of what $E(I_pI_q)$ is for $p \neq q$, both primes.*

3. *Use (1) and (2) to prove that the number of prime factors is almost surely* $\theta(\ln \ln n)$*.*

Exercise 4.15 *Suppose one hides a clique of size* k *in a random graph* $G\left(n, \frac{1}{2}\right)$*. I.e., in the random graph, choose some subset* S *of* k *vertices and put in the missing edges to make* S *a clique. Presented with the modified graph, find* S*. The larger* S *is, the easier it should be to find. In fact, if* k *is more than* $c\sqrt{n \ln n}$*, then the clique leaves a telltale sign identifying* S *as the* k *vertices of largest degree. Prove this statement by appealing to Theorem 4.1.1. It remains a puzzling open problem to do this when* k *is smaller, say,* $O(n^{1/3})$*.*

Exercise 4.16 *The clique problem in a graph is to find the maximal size clique. This problem is known to be NP-hard and so a polynomial time algorithm is thought unlikely. We can ask the corresponding question about random graphs. For example, in* $G\left(n, \frac{1}{2}\right)$ *there almost surely is a clique of size* $(2-\varepsilon)\log n$ *for any* $\varepsilon > 0$*. But it is not known how to find one in polynomial time.*

1. *Show that in* $G(n, \frac{1}{2})$*, there are, almost surely, no cliques of size* $2\log_2 n$*.*

2. *Use the second moment method to show that in* $G(n, \frac{1}{2})$*, almost surely there are cliques of size* $(2-\varepsilon)\log_2 n$*.*

3. *Show that for any* $\varepsilon > 0$*, a clique of size* $(2-\varepsilon)\log n$ *can be found in* $G\left(n, \frac{1}{2}\right)$ *in time* $n^{O(\ln n)}$*.*

4. *Give an* $O\left(n^2\right)$ *algorithm for finding a clique of size* $\Omega\left(\log n\right)$ *in* $G(n, \frac{1}{2})$*. Hint: use a greedy algorithm. Apply your algorithm to* $G\left(1000, \frac{1}{2}\right)$*. What size clique do you find?*

5. *An independent set of vertices in a graph is a set of vertices, no two of which are connected by an edge. Give a polynomial time algorithm for finding an independent set in* $G\left(n, \frac{1}{2}\right)$ *of size* $\Omega\left(\log n\right)$*.*

Exercise 4.17 *Suppose* H *is a fixed graph on* cn *vertics with* $\frac{1}{4}c^2(\log n)^2$ *edges. Show that if* $c \geq 2$*, whp,* H *does not occur as a subgraph of* $G(n, 1/4)$*.*

Exercise 4.18 *Given two instances,* G_1 *and* G_2 *of* $G(n, \frac{1}{2})$*, what is the largest subgraph common to both* G_1 *and* G_2*?*

Exercise 4.19 *(Birthday problem) What is the number of integers that must be drawn with replacement from a set of* n *integers so that some integer, almost surely, will be selected twice?*

Exercise 4.20 *Suppose the graph of a social network has 20,000 vertices. You have a program that starting from a random seed produces a community. A community is a set of vertices where each vertex in the set has more edges connecting it toother vertices in the set than to vertices outside of the set. In running the algorithm you find thousands of communities and wonder how many communities there are in the graph. Finally, when you find the* $10,000^{th}$ *community, it is a duplicate. It is the same community as one found earlier.*

1. *Use the birthday problem to derive a lower and an upper bound on the number of communities.*

Exercise 4.21 *Let $d > 1$ be a constant and $p_i = 1 - \left(1 - \frac{d}{n}\right)^i$. We saw that if we do breadth first search in $G(n, \frac{d}{n})$ starting at some vertex, z_i, the number of discovered vertices after i steps has the distribution Binomial(n, p_i). We also saw that that if the connected component found has i vertices, then $z_i = i$. Show that as $n \to \infty$ (and d is a fixed constant), Prob$(z_i = i) \in o(1/n)$ unless $i \leq c_1 \ln n$ or $i \geq c_2 n$ for some constants c_1, c_2.*

Exercise 4.22 *Let s be the expected number of vertices discovered as a function of the number of steps t in a breadth first search of $G\left(n, \frac{d}{n}\right)$. Write a differential equation using expected values for the size of s. Show that the normalized size $f = \frac{s-t}{n}$ of the frontier is $f(x) = 1 - e^{-dx} - x$ where $x = \frac{t}{n}$ is the normalized time.*

Exercise 4.23 *For $f(x) = 1 - e^{-dx} - x$, what is the value of $x_{max} = \arg\max f(x)$? What is the value of $f(x_{max})$? Where does the maximum expected value of the frontier of a breadth search in $G(n, \frac{d}{n})$ occur as a function of n?*

Exercise 4.24 *If y and z are independent, nonnegative random variables, then the generating function of the sum $y + z$ is the product of the generating function of y and z. Show that this follows from $E(x^{y+z}) = E(x^y x^z) = E(x^y)E(x^z)$.*

Exercise 4.25 *Let $f_j(x)$ be the j^{th} iterate of the generating function $f(x)$ of a branching process. When $m > 1$, $lim_{j\to\infty} f_j(x) = q$ for $0 < x < 1$. In the limit this implies Prob$(z_j = 0) = q$ and Prob$(z_j = i) = 0$ for all nonzero finite values of i. Shouldn't the probabilities add up to 1? Why is this not a contradiction?*

Exercise 4.26 *Try to create a probability distribution for a branching process which varies with the current population in which future generations neither die out, nor grow to infinity.*

Exercise 4.27 *Let d be a constant strictly greater than 1. Show that for a branching process with number of children distributed as Binomial$(n - c_1 n^{2/3}, \frac{d}{n})$, the root of the $f(x) = 1$ in $(0, 1)$ is at most a constant strictly less than 1.*

Exercise 4.28 *Randomly generate $G(50, p)$ for several values of p. Start with $p = \frac{1}{50}$.*

1. *For what value of p do cycles first appear?*

2. *For what value of p do isolated vertices disappear and the graphs become connected?*

Exercise 4.29 *Consider $G(n, p)$ with $p = \frac{1}{3n}$. Then, we saw that almost surely, there are no cycles of length 10.*

1. *Use the second moment method to show that, almost surely, there is a simple path of length 10.*

2. *Exaplin these two results - that there is a simple path of length 10, but no cycle of length 10.*

Exercise 4.30 *Complete the second moment argument of Theorem 4.13 to show that for* $p = \frac{d}{n}$, $d > 1$, *$G(n, p)$ almost surely has a cycle.*
Hint: If two cycles share one or more edges, then the union of the two cycles is at least one greater than the union of the vertices.

Exercise 4.31 *Draw a tree with 10 vertices and label each vertex with a unique integer from 1 to 10. Construct the Prüfer sequence (Appendix 12.8.6) for the tree. Given the Prüfer sequence, recreate the tree.*

Exercise 4.32 *Construct the tree corresponding to the following Prüfer sequences (Appendix 12.8.6)*

1. *113663 (1,2),(1,3),(1,4),(3,5),(3,6),(6,7), and (6,8)*

2. *552833226.*

Exercise 4.33 *What is the expected number of isolated vertices in $G(n, p)$ for $p = \frac{1}{2}\frac{\ln n}{n}$?*

Exercise 4.34 *Theorem 4.17 shows that for some $c > 0$ and $p = c \ln n/n$, $G(n, p)$ has diameter $O(\ln n)$. Tighten the argument to pin down as low a value as possible for c.*

Exercise 4.35 *What is diameter of G(n,p) for various values of p?*

Exercise 4.36

1. *List five increasing properties of $G(n, p)$.*

2. *List five non increasing properties .*

Exercise 4.37 *Consider generating the edges of a random graph by flipping two coins, one with probability p_1 of heads and the other with probability p_2 of heads. Add the edge to the graph if either coin comes down heads. What is the value of p for the generated $G(n, p)$ graph?*

Exercise 4.38 *In the proof of Theorem 4.19, we proved for $p_0(n)$ such that $\lim\limits_{n\to\infty} \frac{p_0(n)}{p(n)} = 0$ that $G(n, p_0)$ almost surely did not have property Q. Give the symmetric argument that for any $p_1(n)$ such that $\lim\limits_{n\to\infty} \frac{p(n)}{p_1(n)} = 0$, $G(n, p_1)$ almost surely has property Q.*

Exercise 4.39 *Consider a model of a random subset $N(n, p)$ of integers $\{1, 2, \ldots n\}$ where, $N(n, p)$ is the set obtained by independently at random including each of $\{1, 2, \ldots n\}$ into the set with probability p. Define what an "increasing property" of $N(n, p)$ means. Prove that every increasing property of $N(n, p)$ has a threshold.*

Exercise 4.40 *$N(n,p)$ is a model of a random subset of integers $\{1, 2, \ldots n\}$ where, $N(n,p)$ is the set obtained by independently at random including each of $\{1, 2, \ldots n\}$ into the set with probability p. What is the threshold for $N(n,p)$ to contain*

1. *a perfect square,*
2. *a perfect cube,*
3. *an even number,*
4. *three numbers such that $x + y = z$?*

Exercise 4.41 *Explain why the property, that $N(n,p)$ contains the integer 1, has a threshold. What is the threshold?*

Exercise 4.42 *Is there a condition such that any property satisfying the condition has a sharp threshold? For example, is monotonicity such a condition?*

Exercise 4.43 *The Sudoku game consists of a 9×9 array of squares. The array is partitioned into nine 3×3 squares. Each small square should be filled with an integer between 1 and 9 so that each row, each column, and each 3×3 square contains exactly one copy of each integer. Initially the board has some of the small squares filled in in such a way that there is exactly one way to complete the assignments of integers to squares. Some simple rules can be developed to fill in the remaining squares such as if the row and column containing a square already contain a copy of every integer except one, that integer should be placed in the square.*

Start with a 9×9 array of squares with each square containing a number between 1 and 9 such that no row, column, or 3×3 square has two copies of any integer.

1. *How many integers can you randomly erase and there still be only one way to correctly fill in the board?*
2. *Develop a set of simple rules for filling in squares such as if a row does not contain a given integer and if every column except one in which the square in the row is blank contains the integer, then place the integer in the remaining blank entry in the row. How many integers can you randomly erase and your rules will still completely fill in the board?*

Exercise 4.44 *Generalize the Sudoku game for arrays of size $n^2 \times n^2$. Develop a simple set of rules for completing the game. An example of a rule is the following. If the a row does not contain a given integer and if every column except one in which the square in the row is blank contains the integer, then place the integer in the remaining blank entry in the row. Start with a legitimate completed array and erase k entries at random.*

1. *Is there a threshold for the integer k such that if only k entries of the array are erased, your set of rules will find a solution?*

2. *Experimentally determine k for some large value of n.*

Exercise 4.45 *Let* $\{x_i|1 \leq i \leq n\}$*, be a set of indicator variables with identical probability distributions. Let* $x = \sum\limits_{i=1}^{n} x_i$ *and suppose* $E(x) \rightarrow \infty$*. Show that if the* x_i *are statistically independent, then* $Prob\,(x = 0) \rightarrow 0$*.*

Exercise 4.46 *In a square* $n \times n$ *grid, each of the* $O(n^2)$ *edges is randomly chosen to be present with probability p and absent with probability* $1 - p$*. Consider the increasing property that there is a path from the bottom left corner to the top right corner which always goes to the right or up. Show that* $p = 1/2$ *is a threshold for the property. Is it a sharp threshold?*

Exercise 4.47 *The threshold property seems to be related to uniform distributions. What if we considered other distributions? Consider a model where i is selected from the set* $\{1, 2, \ldots, n\}$ *with probability* $\frac{c(n)}{i}$*. Is there a threshold for perfect squares? Is there a threshold for arithmetic progressions?*

Exercise 4.48 *Modify the proof that every increasing property of* $G(n, p)$ *has a threshold to apply to the 3-CNF satisfiability problem.*

Exercise 4.49 *Evaluate* $\left(1 - \frac{1}{2^k}\right)^{2^k}$ *for k=3, 5, and 7. How close is it to 1/e?*

Exercise 4.50 *Randomly generate clauses for a Boolean formula in 3-CNF. Compute the number of solutions and the number of connected components of the solution set as a function of the number of clauses generated. What happens?*

Exercise 4.51 *Consider a random process for generating a Boolean function f in conjunctive normal form where each of c clauses is generated by placing each of n variables in the clause with probability p and complementing the variable with probability* $1/2$*. What is the distribution of clause sizes for various p such as* $p = 3/n$*,* $1/2$*, other values? Experimentally determine the threshold value of p for f to cease to be satisfied.*

Exercise 4.52 *For a random 3-CNF formula with n variables and cn clauses, what is the expected number of satisfying assignments?*

Exercise 4.53 *Which of the following variants of the SC algorithm admit a theorem like Theorem 4.21?*

1. *Among all clauses of least length, pick the first one in the order in which they appear in the formula.*

2. *Set the literal appearing in most clauses independent of length to 1.*

Exercise 4.54 *Suppose we have a queue of jobs serviced by one server. There is a total of n jobs in the system. At time t, each remaining job independently decides to join the queue to be serviced with probability $p = d/n$, where $d < 1$ is a constant. Each job has a processing time of 1 and at each time the server services one job, if the queue is nonempty. Show that with high probability, no job waits more than $\Omega(\ln n)$ time to be serviced once it joins the queue.*

Exercise 4.55 *Consider $G(n, p)$.*

1. *Where is phase transition for 2-colorability? Hint: For $p = d/n$ with $d < 1$, $G(n, p)$ is acyclic, so it is bipartite and hence 2-colorable. When $pn \to \infty$, the expected number of triangles goes to infinity. Show that, almost surely, there is a triangle? What does this do for 2-colorability?*

2. *What about 3-colorability?*

Exercise 4.56 *A vertex cover of size k for a graph is a set of k vertices such that one end of each edge is in the set. Experimentally play with the following problem. For $G(n, \frac{1}{2})$, for what value of k is there a vertex cover of size k?*

Exercise 4.57 *Consider graph 3-colorability. Randomly generate the edges of a graph and compute the number of solutions and the number of connected components of the solution set as a function of the number of edges generated. What happens?*

Exercise 4.58 *In $G(n, p)$, let x_k be the number of connected components of size k. Using x_k, write down the probability that a randomly chosen vertex is in a connected component of size k. Also write down the expected size of the connected component containing a randomly chosen vertex.*

Exercise 4.59 *For p asymptotically greater than $\frac{1}{n}$, show that*
$\sum\limits_{i=0}^{\infty} i(i-2)\lambda_i > 0.$

Exercise 4.60 *Consider generating a random graph adding one edge at a time. Let n(i,t) be the number of components of size i at time t.*

$$\begin{aligned}
&n(1,1) = n \\
&n(1,t) = 0 \quad t > 1 \\
&n(i,t) = n(i,t-1) + \sum \frac{j(i-j)}{n^2} n(j,t-1)\, n(i-j,t-1) - \frac{2i}{n} n(i)
\end{aligned}$$

Compute n(i,t) for a number of values of i and t. What is the behavior? What is the sum of n(i,t) for fixed t and all i? Can you write a generating function for n(i,t)?

Exercise 4.61 *The global clustering coefficient of a graph is defined as follows. Let d_v be the degree of vertex v and let e_v be the number of edges connecting vertices adjacent to vertex v. The global clustering coefficient c is given by*

$$c = \sum_v \tfrac{2e_v}{d_v(d_v-1)}.$$

In a social network, for example, it measures what fraction of pairs of friends of each person are themselves friends. If many are, the clustering coefficient is high. What is c for a random graph with $p = \frac{d}{n}$? For a denser graph? Compare this value to that for some social network.

Exercise 4.62 *Consider a structured graph, such as a grid or cycle, and gradually add edges or reroute edges at random. Let L be the average distance between all pairs of vertices in a graph and let C be the ratio of triangles to connected sets of three vertices. Plot L and C as a function of the randomness introduced.*

Exercise 4.63 *Consider an $n \times n$ grid in the plane.*

1. *Prove that for any vertex u, there are at least k vertices at distance k for $1 \leq k \leq n/2$.*

2. *Prove that for any vertex u, there are at most $4k$ vertices at distance k.*

3. *Prove that for one half of the pairs of points, the distance between them is at least $4/4$.*

Exercise 4.64 *Show that in a small-world graph with $r \leq 2$, that there exist short paths with high probability. The proof for $r = 0$ is in the text.*

Exercise 4.65 *Change the small worlds graph as follows. Start with a $n \times n$ grid where each vertex has one long-distance edge to a vertex chosen uniformly at random. These are exactly like the long-distance edges for $r = 0$. But the grid edges are not present. Instead, we have some other graph with the property that for each vertex, there are $\Theta(t^2)$ vertices at distance t from the vertex for $t \leq n$. Show that, almost surely, the diameter is $O(\ln n)$.*

Exercise 4.66 *Given an n node directed graph with two random out edges from each node. For two vertices s and t chosen at random, prove that there exists a path of length at most $O(\ln n)$ from s to t with high probability.*

Exercise 4.67 *How does the diameter of a graph consisting of a cycle change as one adds a few random long distance edges? This question explores how much randomness is needed to get a small world.*

Exercise 4.68 *Ideas and diseases spread rapidly in small world graphs. What about spread of social contagion? A disease needs only one contact and with some probability transfers. Social contagion needs several contacts. How many vertices must one start with to spread social contagion, if the spread of contagion requires two adjacent vertices?*

Exercise 4.69 *How many edges are needed to disconnect a small world graph? By disconnect we mean at least two pieces each of reasonable size. Is this connected to the emergence of a giant component?*

Exercise 4.70 *In the small world model, would it help if the algorithm could look at edges at any node at a cost of one for each node looked at?*

Exercise 4.71 *Consider the $n \times n$ grid in the section on small world graphs. If the probability of an edge from vertex u to vertex v is proportional to $d^{-r}(u,v)$, show that the constant of proportionality $c_r(u)$ is*

$$\begin{array}{ll} \theta(n^{2-r}) & \textit{for } r > 2 \\ \theta(\ln n) & \textit{for } r = 2 \\ \theta(1) & \textit{for } r < 2 \end{array}$$

Exercise 4.72 *In the $n \times n$ grid prove that for at least half of the pairs of vertices, the distance between the vertices is greater than or equal to $n/4$*

Exercise 4.73 *Show that for $r < 2$ in the small world graph model that short paths exist but a polylog length path is unlikely to encounter a long distance edge whose end point is close to the destination.*

Exercise 4.74 *Make a list of the ten most interesting things you learned about random graphs.*

5 Random Walks and Markov Chains

A random walk on a directed graph consists of a sequence of vertices generated from a start vertex by selecting an edge, traversing the edge to a new vertex, and repeating the process. We will see that if the graph is strongly connected, then the fraction of time the walk spends at the various vertices of the graph converges to a stationary probability distribution.

Since the graph is directed, there might be vertices with no out edges and hence nowhere for the walk to go. Vertices in a strongly connected component with no in edges from the remainder of the graph can never be reached unless the component contains the start vertex. Once a walk leaves a strongly connected component it can never return. Most of our discussion of random walks will involve strongly connected graphs.

Start a random walk at a vertex x_0 and think of the starting probability distribution as putting a mass of one on x_0 and zero on every other vertex. More generally, one could start with any probability distribution $\mathbf{p}$, where $\mathbf{p}$ is a row vector with nonnegative components summing to one, with p_x being the probability of starting at vertex x. The probability of being at vertex x at time $t+1$ is the sum over each adjacent vertex y of being at y at time t and taking the transition from y to x. Let $\mathbf{p^{(t)}}$ be a row vector with a component for each vertex specifying the probability mass of the vertex at time t and let $\mathbf{p^{(t+1)}}$ be the row vector of probabilities at time $t+1$. In matrix notation[9]

$$\mathbf{p^{(t)}}P = \mathbf{p^{(t+1)}}$$

where the ij^{th} entry of the matrix P is the probability of the walk at vertex i selecting the edge to vertex j.

A fundamental property of a random walk is that in the limit, the long-term average probability of being at a particular vertex is independent of the start vertex, or an initial probability distribution over vertices, provided only that the underlying graph is strongly connected. The limiting probabilities are called the *stationary probabilities*. This fundamental theorem is proved in the next section.

A special case of random walks, namely random walks on undirected graphs, has important connections to electrical networks. Here, each edge has a parameter called *conductance*, like electrical conductance. If the walk is at vertex u, it chooses an edge from among all edges incident to u to walk to the next vertex with probability proportional to its conductance. Certain basic quantities associated with random walks are hitting time, the expected time to reach vertex y starting at vertex x, and cover time, the expected time to visit every vertex. Qualitatively, these quantities are all bounded above by polynomials in the number of vertices. The proofs of these facts will rely on the

[9] Probability vectors are represented by row vectors to simplify notation in equations like the one here.

random walk	**Markov chain**
graph	stochastic process
vertex	state
strongly connected	persistent
aperiodic	aperiodic
strongly connected and aperiodic	ergotic
undirected graph	time reversible

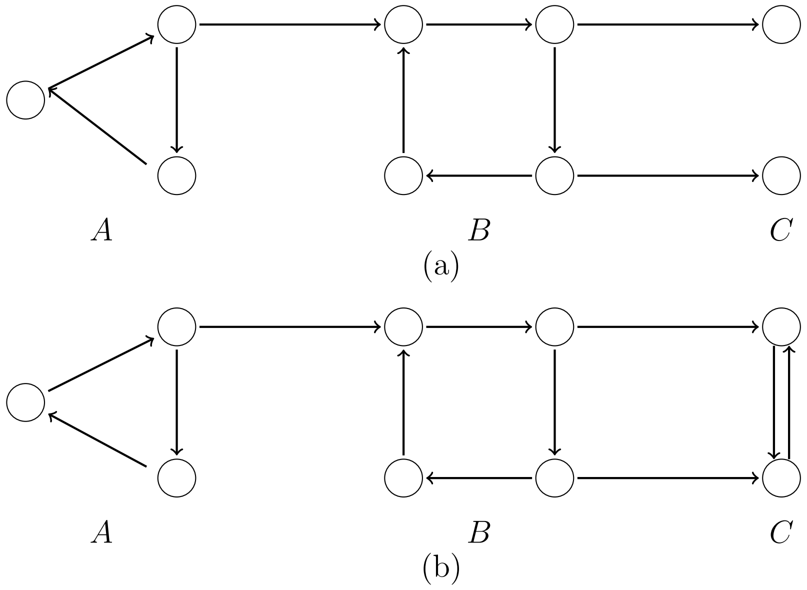
Table 5.1: Correspondence between terminology of random walks and Markov chains

analogy between random walks and electrical networks.

Aspects of the theory of random walks were developed in computer science with an important application in defining the pagerank of pages on the World Wide Web by their stationary probability. An equivalent concept called a *Markov chain* had previously been developed in the statistical literature. A Markov chain has a finite set of *states*. For each pair of states x and y, there is a *transition probability* p_{xy} of going from state x to state y where for each x, $\sum_y p_{xy} = 1$. A random walk in the Markov chain starts at some state. At a given time step, if it is in state x, the next state y is selected randomly with probability p_{xy}. A Markov chain can be represented by a directed graph with a vertex representing each state and an edge with weight p_{xy} from vertex x to vertex y. We say that the Markov chain is *connected* if the underlying directed graph is strongly connected. That is, if there is a directed path from every vertex to every other vertex. The matrix P consisting of the p_{xy} is called the *transition probability matrix* of the chain. The terms "random walk" and "Markov chain" are used interchangeably. The correspondence between the terminologies of random walks and Markov chains is given in Table 5.1.

A state of a Markov chain is *persistent* if it has the property that should the state ever be reached, the random process will return to it with probability one. This is equivalent to the property that the state is in a strongly connected component with no out edges. For most of the chapter, we assume that the underlying directed graph is strongly connected. We discuss here briefly what might happen if we do not have strong connectivity. Consider the directed graph in Figure 5.1b with three strongly connected components, A, B, and C. Starting from any vertex in A, there is a nonzero probability of eventually reaching any vertex in A. However, the probability of returning to a vertex in A is less than one and thus vertices in A, and similarly vertices in B, are not persistent. From any vertex in C, the walk eventually will return with probability one to the vertex, since there is no way of leaving component C. Thus, vertices in C are persistent.

A connected Markov Chain is said to be *aperiodic* if the greatest common divisor of



(a)

(b)

Figure 5.1: (a) A directed graph with vertices having no out out edges and a strongly connected component A with no in edges.
(b) A directed graph with three strongly connected components.

the lengths of directed cycles is 1. It is known (though we do not prove it here) that for connected aperiodic chains, the probability distribution of the random walk converges to a unique stationary distribution.

Markov chains are used to model situations where all the information of the system necessary to predict the future can be encoded in the current state. A typical example is speech, where for a small k the current state encodes the last k syllables uttered by the speaker. Given the current state, there is a certain probability of each syllable being uttered next and these can be used to calculate the transition probabilities. Another example is a gambler's assets, which can be modeled as a Markov chain where the current state is the amount of money the gambler has on hand. The model would only be valid if the gambler's bets depend only on current assets, not the past history.

Later in the chapter, we study the widely used Markov Chain Monte Carlo method (MCMC). Here, the objective is to sample a large space according to some probability distribution p. The number of elements in the space may be very large, say 10^{100}. One designs a Markov chain where states correspond to the elements of the space. The transition probabilities of the chain are designed so that the stationary probability of the chain is the probability distribution p with which we want to sample. One samples by taking a random walk until the probability distribution is close to the stationary distribution of the chain and then selects the point the walk is at. The walk continues a number of steps until the probability distribution is no longer dependent on where the walk was when the first element was selected. A second point is then selected, and so on. Although it is impossible to store the graph in a computer since it has 10^{100} vertices, to do the walk one needs only store the vertex the walk is at and be able to generate the adjacent vertices by some algorithm. What is critical is that the probability of the walk converges to the stationary probability in time logarithmic in the number of states.

We mention two motivating examples. The first is to estimate the probability of a region R in d-space according to a probability density like the Gaussian. Put down a grid and make each grid point that is in R a state of the Markov chain. Given a probability density p, design transition probabilities of a Markov chain so that the stationary distribution is exactly p. In general, the number of states grows exponentially in the dimension d, but the time to converge to the stationary distribution grows polynomially in d.

A second example is from physics. Consider an $n \times n$ grid in the plane with a particle at each grid point. Each particle has a spin of ± 1. There are 2^{n^2} spin configurations. The energy of a configuration is a function of the spins. A central problem in statistical mechanics is to sample a spin configuration according to itsprobability. It is easy to design a Markov chain with one state per spin configuration so that the stationary probability of a state is proportional to the state's energy. If a random walk gets close to the stationary probability in time polynomial to n rather than 2^{n^2}, then one can sample spin configurations according to their probability.

A quantity called the *mixing time*, loosely defined as the time needed to get close to the stationary distribution, is often much smaller than the number of states. In Section 5.4, we relate the mixing time to a combinatorial notion called *normalized conductance* and derive upper bounds on the mixing time in many cases.

5.1 Stationary Distribution

Let $\mathbf{p^{(t)}}$ be the probability distribution after t steps of a random walk. Define the *long-term probability distribution* $\mathbf{a^{(t)}}$ by

$$\mathbf{a^{(t)}} = \frac{1}{t}\left(\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}}\right).$$

The fundamental theorem of Markov chains asserts that the long-term probability distribution of a connected Markov chain converges to a unique limit probability vector, denoted $\boldsymbol{\pi}$. Executing one more step, starting from this limit distribution, we get back the same distribution. In matrix notation, $\boldsymbol{\pi}P = \boldsymbol{\pi}$ where P is the matrix of transition probabilities. There is a unique probability vector (nonnegative components summing to one) satisfying $\boldsymbol{\pi}P = \boldsymbol{\pi}$ and this vector is the limit. Also since one step does not change the distribution, any number of steps does not. For this reason, $\boldsymbol{\pi}$ is called the *stationary distribution.*

Before proving the fundamental theorem of Markov chains, we first prove a technical lemma.

Lemma 5.1 *Let P be the transition probability matrix for a connected Markov chain. The $n \times (n+1)$ matrix $A = [P - I\ ,\ \mathbf{1}]$ obtained by augmenting the matrix $P - I$ with an additional column of ones has rank n.*

Proof: If the rank of $A = [P - I, \mathbf{1}]$ was less than n there would be two linearly independent solutions to $A\mathbf{x} = \mathbf{0}$. Each row in P sums to one so each row in $P - I$ sums to zero. Thus $\mathbf{x} = (\mathbf{1}, 0)$, where all but the last coordinate of $\mathbf{x}$ is 1, is one solution to $A\mathbf{x} = \mathbf{0}$. Assume there was a second solution $(\mathbf{x}, \alpha)$ perpendicular to $(\mathbf{1}, 0)$. Then $(P - I)\mathbf{x} + \alpha\mathbf{1} = \mathbf{0}$. Thus for each i $\sum\limits_j p_{ij}x_j - x_i + \alpha = 0$ or $x_i = \sum_j p_{ij}x_j + \alpha$. Each x_i is a convex combination of some x_j plus α. Let S be the set of i for which x_i attains its maximum value. $\bar{S}$ is not empty since x is perpendicular to $\mathbf{1}$ and hence some x_j must be negative. Connectedness implies that some x_k of maximum value is adjacent to some x_l of lower value. Thus, $x_k > \sum_j p_{kj}x_j$. Therefore α must be greater than 0 in $x_k = \sum_j p_{kj}x_j + \alpha$. A symmetric argument with T the set of i with x_i taking its minimum value implies $\alpha < 0$ producing a contradiction proving the lemma. ∎

Theorem 5.2 (Fundamental Theorem of Markov Chains) *Let P be the transition probability matrix for a connected Markov chain, $\mathbf{p^{(t)}}$ the probability distribution after t steps of a random walk, and*

$$\mathbf{a^{(t)}} = \frac{1}{t}\left(\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}}\right)$$

the long term probability distribution. Then there is a unique probability vector $\boldsymbol{\pi}$ satisfying $\boldsymbol{\pi}P = \boldsymbol{\pi}$. Moreover, for any starting distribution, $\lim\limits_{t\to\infty} \mathbf{a^{(t)}}$ exists and equals $\boldsymbol{\pi}$.

Proof: Note that $\mathbf{a^{(t)}}$ is itself a probability vector, since its components are nonnegative and sum to one. After one step, the distribution of the Markov chain starting from the distribution $\mathbf{a^{(t)}}$ is $\mathbf{a^{(t)}}P$. The change in probabilities due to this step.

$$\begin{aligned}
\mathbf{a^{(t)}}P - \mathbf{a^{(t)}} &= \frac{1}{t}\left[\mathbf{p^{(0)}}P + \mathbf{p^{(1)}}P + \cdots + \mathbf{p^{(t-1)}}P\right] - \frac{1}{t}\left[\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}}\right] \\
&= \frac{1}{t}\left[\mathbf{p^{(1)}} + \mathbf{p^{(2)}} + \cdots + \mathbf{p^{(t)}}\right] - \frac{1}{t}\left[\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}}\right] \\
&= \frac{1}{t}\left(\mathbf{p^{(t)}} - \mathbf{p^{(0)}}\right).
\end{aligned}$$

Thus, $\mathbf{b^{(t)}} = \mathbf{a^{(t)}}P - \mathbf{a^{(t)}}$ satisfies $|\mathbf{b^{(t)}}| \le \frac{2}{t}$ and goes to zero as $t \to \infty$. Thus, $\lim\limits_{t\to\infty} a^t$ exists.

By Lemma 5.1 above, $A = [P - I, \mathbf{1}]$ has rank n. Since the first n columns of A sum to zero, the $n \times n$ submatrix B of A consisting of all its columns except the first is invertible. Let $\mathbf{c^{(t)}}$ be obtained from $\mathbf{b^{(t)}} = \mathbf{a^{(t)}}P - \mathbf{a^{(t)}}$ by removing the first entry so that $\mathbf{a^{(t)}}B = [\mathbf{c^{(t)}}, 1]$. Then $\mathbf{a^{(t)}} = [\mathbf{c^{(t)}}\ ,\ 1]B^{-1} \to [\mathbf{0}\ ,\ 1]B^{-1}$ implying $\boldsymbol{\pi} = [\mathbf{0}\ ,\ 1]B^{-1}$. ∎

Observe that the expected time r_x for a Markov chain starting in state x to return to state x is the reciprocal of the stationary probability of x. That is $r_x = \frac{1}{\pi_x}$. Intuitively this follows by observing that if a long walk always returns to state x in exactly r_x steps, the frequency of being in a state x would be $\frac{1}{r_x}$. A rigorous proof requires the Strong Law of Large Numbers.

We finish this section with the following lemma useful in establishing that a probability distribution is the stationary probability distribution for a random walk on a connected graph with edge probabilities.

Lemma 5.3 *For a random walk on a strongly connected graph with probabilities on the edges, if the vector* $\boldsymbol{\pi}$ *satisfies* $\pi_x p_{xy} = \pi_y p_{yx}$ *for all* x *and* y *and* $\sum_x \pi_x = 1$*, then* $\boldsymbol{\pi}$ *is the stationary distribution of the walk.*

Proof: Since $\boldsymbol{\pi}$ satisfies $\pi_x p_{xy} = \pi_y p_{yx}$, summing both sides, $\pi_x = \sum\limits_y \pi_y p_{yx}$ and hence $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi} = \boldsymbol{\pi} P$. By Theorem 5.2, $\boldsymbol{\pi}$ is the unique stationary probability. ∎

5.2 Markov Chain Monte Carlo

The Markov Chain Monte Carlo (MCMC) method is a technique for sampling a multivariate probability distribution $p(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$. The MCMC method is used to estimate the expected value of a function $f(\mathbf{x})$

$$E(f) = \sum_{\mathbf{x}} f(\mathbf{x}) p(\mathbf{x}).$$

If each x_i can take on two or more values, then there are at least 2^d values for $\mathbf{x}$, so an explicit summation requires exponential time. Instead, one could draw a set of samples, each sample $\mathbf{x}$ with probability $p(\mathbf{x})$. Averaging f over these samples provides an estimate of the sum.

To sample according to $p(\mathbf{x})$, design a Markov Chain whose states correspond to the possible values of $\mathbf{x}$ and whose stationary probability distribution is $p(\mathbf{x})$. There are two general techniques to design such a Markov Chain: the Metropolis-Hastings algorithm and Gibbs sampling. The Fundamental Theorem of Markov Chains, Theorem 5.2, states that the average of f over states seen in a sufficiently long run is a good estimate of $E(f)$. The harder task is to show that the number of steps needed before the long-run average probabilities are close to the stationary distribution grows polynomially in d, though the total number of states may grow exponentially in d. This phenomenon known as *rapid mixing* happens for a number of interesting examples. Section 5.4 presents a crucial tool used to show rapid mixing.

We used $\mathbf{x} \in \mathbf{R}^d$ to emphasize that distributions are multi-variate. From a Markov chain perspective, each value $\mathbf{x}$ can take on is a state, i.e., a vertex of the graph on which the random walk takes place. Henceforth, we will use the subscripts $i, j, k, \ldots$ to denote states and will use p_i instead of $p(x_1, x_2, \ldots, x_d)$ to denote the probability of the state corresponding to a given set of values for the variables. Recall that in the Markov chain terminology, vertices of the graph are called states.

Recall the notation that $\mathbf{p^{(t)}}$ is the row vector of probabilities of the random walk being at each state (vertex of the graph) at time t. So, $\mathbf{p^{(t)}}$ has as many components as there are states and its i^{th} component, $p_i^{(t)}$, is the probability of being in state i at time t. Recall the long-term t-step average is

$$\mathbf{a^{(t)}} = \frac{1}{t}\left[\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}}\right]. \tag{5.1}$$

The expected value of the function f under the probability distribution $\mathbf{p}$ is $E(f) = \sum_i f_i p_i$ where f_i is the value of f at state i. Our estimate of this quantity will be the average value of f at the states seen in a t step walk. Call this estimate a. Clearly, the expected value of a is

$$E(a) = \sum_i f_i \left(\frac{1}{t} \sum_{j=1}^{t} \text{Prob}\left(\text{walk is in state } i \text{ at time } j\right)\right) = \sum_i f_i a_i^{(t)}.$$

The expectation here is with respect to the "coin tosses" of the algorithm, not with respect to the underlying distribution p. Let $f_{\max}$ denote the maximum absolute value of f. It is easy to see that

$$\left| \sum_i f_i p_i - E(a) \right| \leq f_{\max} \sum_i |p_i - a_i^{(t)}| = f_{\max} |\mathbf{p} - \mathbf{a^{(t)}}|_1 \tag{5.2}$$

where the quantity $|\mathbf{p} - \mathbf{a^{(t)}}|_1$ is the l_1 distance between the probability distributions $\mathbf{p}$ and $\mathbf{a^{(t)}}$ and is often called the "total variation distance" between the distributions. We will build tools to upper bound $|\mathbf{p} - \mathbf{a^{(t)}}|_1$. Since $\mathbf{p}$ is the stationary distribution, the t for which $|\mathbf{p} - \mathbf{a^{(t)}}|_1$ becomes small is determined by the rate of convergence of the Markov chain to its steady state.

The following proposition is often useful.

Proposition 5.4 *For two probability distributions* $\mathbf{p}$ *and* $\mathbf{q}$,

$$|\mathbf{p} - \mathbf{q}|_1 = 2\sum_i (p_i - q_i)^+ = 2\sum_i (q_i - p_i)^+$$

where $x^+ = x$ *if* $x \geq 0$ *and* $x^+ = 0$ *if* $x < 0$.

The proof is left as an exercise.

5.2.1 Metropolis-Hasting Algorithm

The Metropolis-Hasting algorithm is a general method to design a Markov chain whose stationary distribution is a given target distribution p. Start with a connected undirected graph G on the set of states. If the states are the lattice points $(x_1, x_2, \ldots, x_d)$ in $\mathbf{R}^d$ with $x_i \in \{0, 1, 2, , \ldots, n\}$, then G is the lattice graph with $2d$ coordinate edges at each interior vertex. In general, let r be the maximum degree of any vertex of G. The transitions of the Markov chain are defined as follows. At state i select neighbor j with probability $\frac{1}{r}$. Since the degree of i may be less than r, with some probability no edge is selected and the walk remains at i. If a neighbor j is selected and $p_j \geq p_i$, go to j. If $p_j < p_i$, go to j with probability p_j/p_i and stay at i with probability $1 - \frac{p_j}{p_i}$. Intuitively, this favors "heavier" states with higher p values. So, for i, adjacent to j in G,

$$p_{ij} = \frac{1}{r} \min\left(1, \frac{p_j}{p_i}\right)$$

and

$$p_{ii} = 1 - \sum_{j \neq i} p_{ij}.$$

Thus,

$$p_i p_{ij} = \frac{p_i}{r} \min\left(1, \frac{p_j}{p_i}\right) = \frac{1}{r} \min(p_i, p_j) = \frac{p_j}{r} \min\left(1, \frac{p_i}{p_j}\right) = p_j p_{ji}.$$

By Lemma 5.3, the stationary probabilities are indeed $p(\mathbf{x})$ as desired.

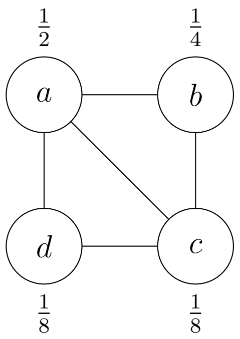
Example: Consider the graph in Figure 5.2. Using the Metropolis-Hasting algorithm, assign transition probabilities so that the stationary probability of a random walk is $p(a) = \frac{1}{2}$, $p(b) = \frac{1}{4}$, $p(c) = \frac{1}{8}$, and $p(d) = \frac{1}{8}$. The maximum degree of any vertex is three, so at a, the probability of taking the edge (a, b) is $\frac{1}{3}\frac{1}{4}\frac{2}{1}$ or $\frac{1}{6}$. The probability of taking the edge (a, c) is $\frac{1}{3}\frac{1}{8}\frac{2}{1}$ or $\frac{1}{12}$ and of taking the edge (a, d) is $\frac{1}{3}\frac{1}{8}\frac{2}{1}$ or $\frac{1}{12}$. Thus, the probability of staying at a is $\frac{2}{3}$. The probability of taking the edge from b to a is $\frac{1}{3}$. The probability of taking the edge from c to a is $\frac{1}{3}$ and the probability of taking the edge from d to a is $\frac{1}{3}$. Thus, the stationary probability of a is $\frac{1}{4}\frac{1}{3} + \frac{1}{8}\frac{1}{3} + \frac{1}{8}\frac{1}{3} + \frac{1}{2}\frac{2}{3} = \frac{1}{2}$, which is the desired probability. ■

5.2.2 Gibbs Sampling

Gibbs sampling is another Markov Chain Monte Carlo method to sample from a multivariate probability distribution. Let $p(\mathbf{x})$ be the target distribution where $\mathbf{x} = (x_1, \ldots, x_d)$. Gibbs sampling consists of a random walk on an undirectd graph whose vertices correspond to the values of $\mathbf{x} = (x_1, \ldots, x_d)$ and in which there is an edge from $\mathbf{x}$ to $\mathbf{y}$ if $\mathbf{x}$ and $\mathbf{y}$ differ in only one coordinate. Thus, the underlying graph is like a d-dimensional lattice except that the vertices in the same coordinate line form a clique.

To generate samples of $\mathbf{x} = (x_1, \ldots, x_d)$ with a target distribution $p(\mathbf{x})$, the Gibbs sampling algorithm repeats the following steps. One of the variables x_i is chosen to be

$p(a) = \frac{1}{2}$
$p(b) = \frac{1}{4}$
$p(c) = \frac{1}{8}$
$p(d) = \frac{1}{8}$



$a \to b$	$\frac{1}{3}\,\frac{1}{4}\,\frac{2}{1} = \frac{1}{6}$	$c \to a$	$\frac{1}{3}$
$a \to c$	$\frac{1}{3}\,\frac{1}{8}\,\frac{2}{1} = \frac{1}{12}$	$c \to b$	$\frac{1}{3}$
$a \to d$	$\frac{1}{3}\,\frac{1}{8}\,\frac{2}{1} = \frac{1}{12}$	$c \to d$	$\frac{1}{3}$
$a \to a$	$1 - \frac{1}{6} - \frac{1}{12} - \frac{1}{12} = \frac{2}{3}$	$c \to c$	$1 - \frac{1}{3} - \frac{1}{3} - \frac{1}{3} = 0$
$b \to a$	$\frac{1}{3}$	$d \to a$	$\frac{1}{3}$
$b \to c$	$\frac{1}{3}\,\frac{1}{8}\,\frac{4}{1} = \frac{1}{6}$	$d \to c$	$\frac{1}{3}$
$b \to b$	$1 - \frac{1}{3} - \frac{1}{6} = \frac{1}{2}$	$d \to d$	$1 - \frac{1}{3} - \frac{1}{3} = \frac{1}{3}$

$$\begin{aligned} p(a) &= p(a)p(a \to a) + p(b)p(b \to a) + p(c)p(c \to a) + p(d)p(d \to a) \\ &= \frac{1}{2}\,\frac{2}{3} + \frac{1}{4}\,\frac{1}{3} + \frac{1}{8}\,\frac{1}{3} + \frac{1}{8}\,\frac{1}{3} = \frac{1}{2} \end{aligned}$$

$$\begin{aligned} p(b) &= p(a)p(a \to b) + p(b)p(b \to b) + p(c)p(c \to b) \\ &= \frac{1}{2}\,\frac{1}{6} + \frac{1}{4}\,\frac{1}{2} + \frac{1}{8}\,\frac{1}{3} = \frac{1}{4} \end{aligned}$$

$$\begin{aligned} p(c) &= p(a)p(a \to c) + p(b)p(b \to c) + p(c)p(c \to c) + p(d)p(d \to c) \\ &= \frac{1}{2}\,\frac{1}{12} + \frac{1}{4}\,\frac{1}{6} + \frac{1}{8}\,0 + \frac{1}{8}\,\frac{1}{3} = \frac{1}{8} \end{aligned}$$

$$\begin{aligned} p(d) &= p(a)p(a \to d) + p(c)p(c \to d) + p(d)p(d \to d) \\ &= \frac{1}{2}\,\frac{1}{12} + \frac{1}{8}\,\frac{1}{3} + \frac{1}{8}\,\frac{1}{3} = \frac{1}{8} \end{aligned}$$

Figure 5.2: Using the Metropolis-Hasting algorithm to set probabilities for a random walk so that the stationary probability will be the desired probability.

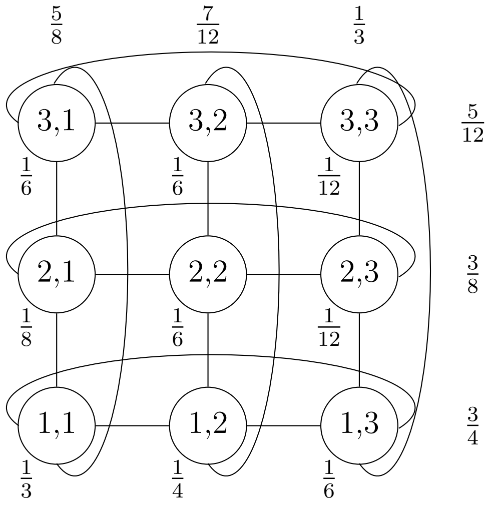
updated. Its new value is chosen based on the marginal probability of x_i with the other variables fixed. There are two commonly used schemes to determine which x_i to update. One scheme is to choose x_i randomly, the other is to choose x_i by sequentially scanning from x_1 to x_d.

Suppose that $\mathbf{x}$ and $\mathbf{y}$ are two states that differ in only one coordinate. Without loss of generality let that coordinate be the first. Then, in the scheme where a coordinate is randomly chosen to modify, the probability $p_{\mathbf{xy}}$ of going from $\mathbf{x}$ to $\mathbf{y}$ is

$$p_{\mathbf{xy}} = \frac{1}{d} p(y_1 | x_2, x_3, \ldots, x_d).$$

The normalizing constant is $1/d$ since $\sum\limits_{y_1} (y_1 | x_2, x_3, \ldots, x_d)$ equals 1 and summing over d coordinates gives a value of d. Similarly,

$$\begin{aligned} p_{\mathbf{yx}} =& \frac{1}{d} p(x_1 | y_2, y_3, \ldots, y_d) \\ =& \frac{1}{d} p(x_1 | x_2, x_3, \ldots, x_d). \end{aligned}$$



$$
\begin{aligned}
p(1,1) &= \tfrac{1}{3}\\
p(1,2) &= \tfrac{1}{4}\\
p(1,3) &= \tfrac{1}{6}\\
p(2,1) &= \tfrac{1}{8}\\
p(2,2) &= \tfrac{1}{6}\\
p(2,3) &= \tfrac{1}{12}\\
p(3,1) &= \tfrac{1}{6}\\
p(3,2) &= \tfrac{1}{6}\\
p(3,3) &= \tfrac{1}{12}
\end{aligned}
$$

$$p_{(11)(12)} = \frac{1}{d}p_{12}/(p_{11}+p_{12}+p_{13} = \frac{1}{2}\frac{1}{4}/(\frac{1}{3}\frac{1}{4}\frac{1}{6} = \frac{1}{2}\frac{1}{4}/\frac{9}{12} = \frac{1}{2}\frac{1}{4}\frac{4}{3} = \frac{1}{6}$$

Calculation of edge probability $p_{(11)(12)}$

$$
\begin{array}{llll}
p_{(11)(12)} = \frac{1}{2}\frac{1}{4}\frac{4}{3} = \frac{1}{6} & p_{(12)(11)} = \frac{1}{2}\frac{1}{3}\frac{4}{3} = \frac{2}{9} & p_{(13)(11)} = \frac{1}{2}\frac{1}{3}\frac{4}{3} = \frac{2}{9} & p_{(21)(22)} = \frac{1}{2}\frac{1}{6}\frac{8}{3} = \frac{2}{9}\\
p_{(11)(13)} = \frac{1}{2}\frac{1}{6}\frac{4}{3} = \frac{1}{9} & p_{(12)(13)} = \frac{1}{2}\frac{1}{6}\frac{4}{3} = \frac{1}{9} & p_{(13)(12)} = \frac{1}{2}\frac{1}{4}\frac{4}{3} = \frac{1}{6} & p_{(21)(23)} = \frac{1}{2}\frac{1}{12}\frac{8}{3} = \frac{1}{9}\\
p_{(11)(21)} = \frac{1}{2}\frac{1}{8}\frac{8}{5} = \frac{1}{10} & p_{(12)(22)} = \frac{1}{2}\frac{1}{6}\frac{12}{7} = \frac{1}{7} & p_{(13)(23)} = \frac{1}{2}\frac{1}{12}\frac{3}{1} = \frac{1}{8} & p_{(21)(11)} = \frac{1}{2}\frac{1}{3}\frac{8}{5} = \frac{4}{15}\\
p_{(11)(31)} = \frac{1}{2}\frac{1}{6}\frac{8}{5} = \frac{2}{15} & p_{(12)(32)} = \frac{1}{2}\frac{1}{6}\frac{12}{7} = \frac{1}{7} & p_{(13)(33)} = \frac{1}{2}\frac{1}{12}\frac{3}{1} = \frac{1}{8} & p_{(21)(31)} = \frac{1}{2}\frac{1}{6}\frac{8}{5} = \frac{2}{15}
\end{array}
$$

Edge probabilities.

$$
\begin{aligned}
&p_{11}p_{(11)(12)} = \frac{1}{3}\frac{1}{6} = \frac{1}{4}\frac{2}{9} = p_{12}p_{(12)(11)}\\
&p_{11}p_{(11)(13)} = \frac{1}{3}\frac{1}{9} = \frac{1}{6}\frac{2}{9} = p_{13}p_{(13)(11)}\\
&p_{11}p_{(11)(21)} = \frac{1}{3}\frac{1}{10} = \frac{1}{8}\frac{4}{15} = p_{21}p_{(21)(11)}
\end{aligned}
$$

Verification of a few edges.

Note that the edge probabilities out of a state such as (1,1) do not add up to one. That is, with some probability the walk stays at the state that it is in. For example, $p_{(11)(11)} = 1 - (p_{(11)(12)} + p_{(11)(13)} + p_{(11)(21)} + p_{(11)(31)}) = 1 - \frac{1}{6} - \frac{1}{24} - \frac{1}{32} - \frac{1}{24} = \frac{9}{32}$.

Figure 5.3: Using the Gibbs algorithm to set probabilities for a random walk so that the stationary probability will be a desired probability.

Here use was made of the fact that for $j \neq 1$, $x_j = y_j$.

It is simple to see that this chain has stationary probability proportional to $p\left(\mathbf{x}\right)$. Rewrite $p_{\mathbf{xy}}$ as

$$\begin{aligned} p_{\mathbf{xy}} &= \frac{1}{d}\frac{p(y_1|x_2,x_3,\dots,x_d)p(x_2,x_3,\dots,x_d)}{p(x_2,x_3,\dots,x_d)} \\ &= \frac{1}{d}\frac{p(y_1,x_2,x_3,\dots,x_d)}{p(x_2,x_3,\dots,x_d)} \\ &= \frac{1}{d}\frac{p(\mathbf{y})}{p(x_2,x_3,\dots,x_d)} \end{aligned}$$

again using $x_j = y_j$ for $j \neq 1$. Similarly write

$$p_{\mathbf{yx}} = \frac{1}{d}\frac{p(\mathbf{x})}{p(x_2,x_3,\dots,x_d)}$$

from which it follows that $p(\mathbf{x})p_{xy} = p(\mathbf{y})p_{yx}$. By Lemma 5.3 the stationary probability of the random walk is $p(\mathbf{x})$.

5.3 Areas and Volumes

Computing areas and volumes is a classical problem. For many regular figures in two and three dimensions there are closed form formulae. In Chapter 2, we saw how to compute volume of a high dimensional sphere by integration. For general convex sets in d-space, there are no closed form formulae. Can we estimate volumes of d-dimensional convex sets in time that grows as a polynomial function of d? The MCMC method answes this question in the affirmative.

One way to estimate the area of the region is to enclose it in a rectangle and estimate the ratio of the area of the region to the area of the rectangle by picking random points in the rectangle and seeing what proportion land in the region. Such methods fail in high dimensions. Even for a sphere in high dimension, a cube enclosing the sphere has exponentially larger area, so exponentially many samples are required to estimate the volume of the sphere.

It turns out that the problem of estimating volumes of sets is reducible to the problem of drawing uniform random samples from sets. Suppose one wants to estimate the volume of a convex set R. Create a concentric series of larger and larger spheres $S_1, S_2, \dots, S_k$ such that S_1 is contained in R and S_k contains R. Then

$$\mathrm{Vol}(R) = \mathrm{Vol}(S_k \cap R) = \frac{\mathrm{Vol}(S_k \cap R)}{\mathrm{Vol}(S_{k-1} \cap R)}\frac{\mathrm{Vol}(S_{k-1} \cap R)}{\mathrm{Vol}(S_{k-2} \cap R)}\cdots\frac{\mathrm{Vol}(S_2 \cap R)}{\mathrm{Vol}(S_1 \cap R)}\mathrm{Vol}(S_1)$$

If the radius of the sphere S_i is $1+\frac{1}{d}$ times the radius of the sphere S_{i-1}, then the value of

$$\frac{\text{Vol}(S_{k-1}\cap R)}{\text{Vol}(S_{k-2}\cap R)}$$

can be estimated by rejection sampling provided one can select points at random from a d-dimensional region. Since the radii of the spheres grows as $1+\frac{1}{d}$, the number of spheres is at most

$$O(\log_{1+(1/d)} r) = O(rd)$$

where r is the ratio of the radius of S_k to the radius of S_1.

It remains to show how to draw a uniform random sample from a d-dimensional set. It is at this point that we require the set to be convex so that the Markov chain technique will converge quickly to its stationary probability. To select a random sample from a d-dimensional convex set impose a grid on the region and do a random walk on the grid points. At each time, pick one of the $2d$ coordinate neighbors of the current grid point, each with probability $1/(2d)$ and go to the neighbor if it is still in the set; otherwise, stay put and repeat. If the grid length in each of the d coordinate directions is at most some a, the total number of grid points in the set is at most a^d. Although this is exponential in d, the Markov chain turns out to be rapidly mixing (the proof is beyond our scope here) and leads to polynomial time bounded algorithm to estimate the volume of any convex set in $\mathbf{R}^d$.

5.4 Convergence of Random Walks on Undirected Graphs

The Metropolis-Hasting algorithm and Gibbs sampling both involve a random walk. Initial states of the walk are highly dependent on the start state of the walk. Both these walks are random walks on edge-weighted undirected graphs. Such Markov chains are derived from electrical networks. Recall the following notation which we will use throughout this section. Given a network of resistors, the conductance of edge (x, y) is denoted c_{xy} and the normalizing constant c_x equals $\sum_y c_{xy}$. The Markov chain has transition probabilities $p_{xy} = c_{xy}/c_x$. We assume the chain is connected. Since

$$c_x p_{xy} = c_c c_{xy}/c_x = c_{xy} = c_{yx} = c_y c_{yx}/c_y = c_y p_{xy}$$

the stationary probabilities are proportional to c_x where the normalization constant is $c_0 = \sum_x c_x$.

An important question is how fast the walk starts to reflect the stationary probability of the Markov process. If the convergence time was proportional to the number of states, the algorithms would not be very useful since the number of states can be exponentially large.

There are clear examples of connected chains that take a long time to converge. A chain with a constriction, see Figure 5.4, takes a long time to converge since the walk is

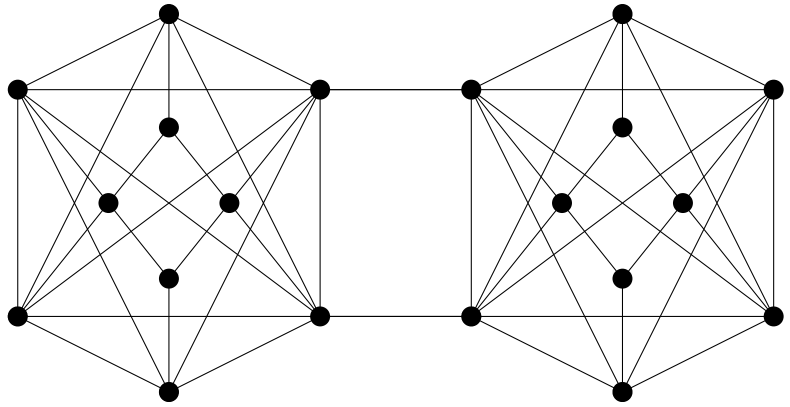


Figure 5.4: A network with a constriction.

unlikely to reach the narrow passage between the two halves, both of which are reasonably big. We will show in Theorem 5.5 that the time to converge is quantitatively related to the tightest constriction.

A function is unimodal if it has a single maximum, i.e., it increases and then decreases. A unimodal function like the normal density has no constriction blocking a random walk from getting out of a large set of states, whereas a bimodal function can have a constriction. Interestingly, many common multivariate distributions as well as univariate probability distributions like the normal and exponential are unimodal and sampling according to these distributions can be done using the methods here.

A natural problem is estimating the probability of a convex region in d-space according to a normal distribution. One technique to do this is rejection sampling. Let R be the region defined by the inequality $x_1 + x_2 + \cdots + x_{d/2} \leq x_{d/2+1} + \cdots + x_d$. Pick a sample according to the normal distribution and accept the sample if it satisfies the inequality. If not, reject the sample and retry until one gets a number of samples satisfying the inequality. The probability of the region is approximated by the fraction of the samples that satisfied the inequality. However, suppose R was the region $x_1 + x_2 + \cdots + x_{d-1} \leq x_d$. The probability of this region is exponentially small in d and so rejection sampling runs into the problem that we need to pick exponentially many samples before we accept even one sample. This second situation is typical. Imagine computing the probability of failure of a system. The object of design is to make the system reliable, so the failure probability is likely to be very low and rejection sampling will take a long time to estimate the failure probability.

In general, there could be constrictions that prevent rapid convergence of a Markov chain to its stationary probability. However, if the set is convex in any number of dimensions, then there are no constrictions and there is rapid convergence although the proof of this is beyond the scope of this book.

We define below a combinatorial measure of constriction for a Markov chain, called the *normalized conductance*, and relate this quantity to the rate at which the chain converges to the stationarity probability.

Definition 5.1 *For a subset S of vertices, the normalized conductance $\Phi(S)$ of S is the probability of taking a step from S to outside S conditioned on starting in S in the stationary probability distribution π.*

$$\Phi(S) = \frac{\sum\limits_{(x,y)} \pi_x p_{xy}}{\sum\limits_{x \in S} \pi_x}$$

■

Definition 5.2 *The* normalized conductance *of the Markov chain, denoted Φ, is defined by*

$$\Phi = \min_{\substack{S \\ \pi(S) \leq 1/2}} \Phi(S).$$

■

The restriction to sets with $\pi \leq 1/2$ in the definition of Φ is natural. The definition of Φ guarantees that if Φ is high, there is high probability of moving from S to $\bar{S}$ so it is unlikely to get stuck in S provided $\pi(S) \leq \frac{1}{2}$. If $\pi(S) > \frac{1}{2}$, say $\pi(S) = \frac{3}{4}$, then since for every edge $\pi_i p_{ij} = \pi_j p_{ji}$

$$\Phi(S) = \frac{\sum_{i \in S} \pi_i p_{ij}}{\sum_{i \in S} \pi_i} = \frac{\sum_{j \in \overline{S}} \pi_j p_{ji}}{3 \sum_{j \in \overline{S}} \pi_k} = \Phi(\bar{S})/3$$

Since $\Phi(\bar{S}) \geq \Phi$, we still have at least $\Phi/3$ probability of moving out of S. The larger $\pi(S)$ is the smaller the probability of moving out, which is as it should be. We cannot move out of the whole set! One does not need to escape from big sets. Note that a constriction would mean a small Φ.

Definition 5.3 *Fix $\varepsilon > 0$. The* ε-mixing time *of a Markov chain is the minimum integer t such that for any starting distribution $\mathbf{p^{(0)}}$, the 1-norm distance between the t-step running average probability distribution*[10] *and the stationary distribution is at most ε.* ■

The theorem below states that if Φ, the normalized conductance of the Markov chain, is large, then there is fast convergence of the running average probability. Intuitively, if

[10] Recall that $\mathbf{a^{(t)}} = \frac{1}{t}(\mathbf{p^{(0)}} + \mathbf{p^{(1)}} + \cdots + \mathbf{p^{(t-1)}})$ is called the running average distribution.

Φ is large, the walk rapidly leaves any subset of states. Later we will see examples where the mixing time is much smaller than the cover time. That is, the number of steps before a random walk reaches a random state independent of its starting state is much smaller than the average number of steps needed to reach every state. In fact for some graphs, called expenders, the mixing time is logarithmic in the number of states.

Theorem 5.5 *The ε-mixing time of a random walk on an undirected graph is*

$$O\left(\frac{\ln(1/\pi_{min})}{\Phi^2\varepsilon^3}\right)$$

where π_{min} is the minimum stationary probability of any state.

Proof: Let

$$t = \frac{c\ln(1/\pi_{\min})}{\Phi^2\varepsilon^2},$$

for a suitable constant c. Let $\mathbf{a} = \mathbf{a^{(t)}}$ be the running average distribution for this value of t. We need to show that $|\mathbf{a} - \boldsymbol{\pi}| \leq \varepsilon$.

Let v_i denote the ratio of the long term average probability for state i at time t divided by the stationary probability for state i. Thus, $v_i = \frac{a_i}{\pi_i}$. Renumber states so that $v_1 \geq v_2 \geq \cdots$. A state i for which $v_i > 1$ has more probability than its stationary probability. Execute one step of the Markov chain starting at probabilities $\mathbf{a}$. The probability vector after that step is $\mathbf{a}P$. Now, $\mathbf{a} - \mathbf{a}P$ is the net loss of probability for each state due to the step. Let k be any integer with $v_k > 1$. Let $A = \{1, 2, \ldots, k\}$. A is a "heavy" set, consisting of states with $a_i \geq \pi_i$. The net loss of probability for each state from the set A in one step is $\sum_{i=1}^{k}(a_i - (\mathbf{a}P)_i) \leq \frac{2}{t}$ as in the proof of Theorem 5.2.

Another way to reckon the net loss of probability from A is to take the difference of the probability flow from A to $\bar{A}$ and the flow from $\bar{A}$ to A. For $i < j$,

$$\text{net-flow}(i,j) = \text{flow}(i,j) - \text{flow}(j,i) = \pi_i p_{ij} v_i - \pi_j p_{ji} v_j = \pi_j p_{ji}(v_i - v_j) \geq 0,$$

Thus, for any $l \geq k$, the flow from A to $\{k+1, k+2, \ldots, l\}$ minus the flow from $\{k+1, k+2, \ldots, l\}$ to A is nonnegative. At each step, heavy sets loose probability. Since for $i \leq k$ and $j > l$, we have $v_i \geq v_k$ and $v_j \leq v_{l+1}$, the net loss from A is at least

$$\sum_{\substack{i\leq k\\ j>l}} \pi_j p_{ji}(v_i - v_j) \geq (v_k - v_{l+1}) \sum_{\substack{i\leq k\\ j>l}} \pi_j p_{ji}.$$

Thus,

$$(v_k - v_{l+1}) \sum_{\substack{i\leq k\\ j>l}} \pi_j p_{ji} \leq \frac{2}{t}.$$

If the total stationary probability $\pi(\{i|v_i \leq 1\})$ of those states where the current probability is less than their stationary probability is less than $\varepsilon/2$, then

$$|\mathbf{a} - \boldsymbol{\pi}|_1 = 2 \sum_{\substack{i \\ v_i \leq 1}} (1 - v_i)\pi_i \leq \varepsilon,$$

so we are done. Assume $\pi(\{i|v_i \leq 1\}) > \varepsilon/2$ so that $\pi(A) \geq \varepsilon \min(\pi(A), \pi(\bar{A}))/2$. Choose l to be the largest integer greater than or equal to k so that

$$\sum_{j=k+1}^{l} \pi_j \leq \varepsilon \Phi \pi(A)/2.$$

Since

$$\sum_{i=1}^{k} \sum_{j=k+1}^{l} \pi_j p_{ji} \leq \sum_{j=k+1}^{l} \pi_j \leq \varepsilon \Phi \pi(A)/2$$

by the definition of Φ,

$$\sum_{i \leq k < j} \pi_j p_{ji} \geq \Phi \min(\pi(A), \pi(\bar{A})) \geq \varepsilon \Phi \pi(A).$$

Thus, $\sum\limits_{\substack{i \leq k \\ j > l}} \pi_j p_{ji} \geq \varepsilon \Phi \pi(A)/2$. Substituting into the inequality 5.4 gives

$$v_k - v_{l+1} \leq \frac{8}{t \varepsilon \Phi \pi(A)}. \tag{5.3}$$

This inequality says that v does not drop too much as we go from k to $l + 1$. On the other hand, the cumulative total of π will have increased, since, $\pi_1 + \pi_2 + \cdots + \pi_{l+1} \geq \rho(\pi_1 + \pi_2 + \cdots + \pi_k)$, where, $\rho = 1 + \frac{\varepsilon\Phi}{2}$. We will be able to use this repeatedly to argue that overall v does not drop too much. If that is the case (in the extreme, for example, if all the v_i are 1 each), then intuitively, $\mathbf{a} \approx \boldsymbol{\pi}$, which is what we are trying to prove. Unfortunately, the technical execution of this argument is a bit messy - we have to divide $\{1, 2, \ldots, n\}$ into groups and consider the drop in v as we move from one group to the next and then add up. We do this now.

Divide $\{1, 2, \ldots\}$ into groups as follows. The first group G_1 is $\{1\}$. In general, if the r^{th} group G_r begins with state k, the next group G_{r+1} begins with state $l + 1$ where l is as defined above. Let i_0 be the largest integer with $v_{i_0} > 1$. Stop with G_m, if G_{m+1} would begin with an $i > i_0$. If group G_r begins in i, define $u_r = v_i$.

$$|\mathbf{a} - \boldsymbol{\pi}|_1 \leq 2 \sum_{i=1}^{i_0} \pi_i (v_i - 1) \leq \sum_{r=1}^{m} \pi(G_r)(u_r - 1) = \sum_{r=1}^{m} \pi(G_1 \cup G_2 \cup \ldots \cup G_r)(u_r - u_{r+1}),$$

where the analog of integration by parts for sums is used in the last step using the convention that $u_{m+1} = 1$. Since $u_r - u_{r+1} \leq 8/\varepsilon\Phi\pi(G_1 \cup \ldots \cup G_r)$, the sum is at most $8m/t\varepsilon\Phi$. Since $\pi_1 + \pi_2 + \cdots + \pi_{l+1} \geq \rho(\pi_1 + \pi_2 + \cdots + \pi_k)$,

$$m \leq \ln_\rho(1/\pi_1) \leq \ln(1/\pi_1)/(\rho - 1).$$

Thus $|\mathbf{a} - \boldsymbol{\pi}|_1 \leq O(\ln(1/\pi_{\min})/t\Phi^2\varepsilon^2) \leq \varepsilon$ for a suitable choice of c and this completes the proof. ■

5.4.1 Using Normalized Conductance to Prove Convergence

We now apply Theorem 5.5 to some examples to illustrate how the normalized conductance bounds the rate of convergence. Our first examples will be simple graphs. The graphs do not have rapid converge, but their simplicity helps illustrate how to bound the normalized conductance and hence the rate of convergence.

A 1-dimensional lattice

Consider a random walk on an undirected graph consisting of an n-vertex path with self-loops at the both ends. With the self loops, the stationary probability is a uniform $\frac{1}{n}$ over all vertices. The set with minimum normalized conductance is the set with probability $\pi <= \frac{1}{2}$ and the maximum number of vertices with the minimum number of edges leaving it. This set consists of the first $n/2$ vertices, for which total conductance of edges from S to $\bar{S}$ is $\pi_{\frac{n}{2}} p_{\frac{n}{2},\frac{n}{2}+1} = \Omega(\frac{1}{n})$ and $\pi(S) = \frac{1}{2}$. ($\pi_{\frac{n}{2}}$ is the stationary probability of vertex numbered $\frac{n}{2}$.) Thus

$$\Phi(S) = 2\pi_{\frac{n}{2}}\, p_{\frac{n}{2},\frac{n}{2}+1} = \Omega(1/n).$$

By Theorem 5.5, for ε a constant such as $1/100$, after $O(n^2 \log n)$ steps, $|\mathbf{a}^{(\mathbf{t})} - \boldsymbol{\pi}|_1 \leq 1/100$. This graph does not have rapid convergence. The hitting time and the cover time are $O(n^2)$. In many interesting cases, the mixing time may be much smaller than the cover time. We will see such an example later.

A 2-dimensional lattice

Consider the $n \times n$ lattice in the plane where from each point there is a transition to each of the coordinate neighbors with probability $1/4$. At the boundary there are self-loops with probability 1-(number of neighbors)/4. It is easy to see that the chain is connected. Since $p_{ij} = p_{ji}$, the function $f_i = 1/n^2$ satisfies $f_i p_{ij} = f_j p_{ji}$ and by Lemma 5.3 is the stationary probability. Consider any subset S consisting of at most half the states. Index states by their x and y coordinates. For at least half the states in S, either row x or column y intersects $\bar{S}$ (Exercise 5.5). So at least $\Omega(|S|/n)$ points in S are adjacent to points in $\bar{S}$. Each such point contributes $\pi_i p_{ij} = \Omega(1/n^2)$ to flow$(S, \bar{S})$. So

$$\sum_{i \in S} \sum_{j \in \bar{S}} \pi_i p_{ij} \geq c|S|/n^3.$$

Thus, $\Phi \geq \Omega(1/n)$. By Theorem 5.5, after $O(n^2 \ln n/\varepsilon^2)$ steps, $|\mathbf{a}^{(\mathbf{t})} - \boldsymbol{\pi}|_1 \leq 1/100$.

A lattice in d-dimensions

Next consider the $n \times n \times \cdots \times n$ lattice in d-dimensions with a self-loop at each boundary point with probability $1 - (\text{number of neighbors})/2d$. The self loops make all π_i equal to n^{-d}. View the lattice as an undirected graph and consider the random walk on this undirected graph. Since there are n^d states, the cover time is at least n^d and thus exponentially dependent on d. It is possible to show (Exercise 5.21) that Φ is $\Omega(1/dn)$. Since all π_i are equal to n^{-d}, the mixing time is $O(d^3 n^2 \ln n/\varepsilon^2)$, which is polynomially bounded in n and d.

The d-dimensional lattice is related to the Metropolis-Hastings algorithm and Gibbs sampling although in those constructions there is a nonuniform probability distribution at the vertices. However, the d-dimension lattice case suggests why the Metropolis-Hastings and Gibbs sampling constructions might converge fast.

A clique

Consider an n vertex clique with a self loop at each vertex. For each edge, $p_{xy} = \frac{1}{n}$ and thus for each vertex, $\pi_x = \frac{1}{n}$. Let S be a subset of the vertices. Then

$$\sum_{x \in S} \pi_x = \frac{|S|}{n}.$$

$$\sum_{(x,y)} \pi_x p_{xy} = \frac{1}{n^2} |S||\overline{S}|$$

and

$$\Phi(S) = \frac{\sum_{(x,y)} \pi_x p_{xy}}{\sum_{x \in S} \pi_x} = \frac{|\overline{S}|}{n}.$$

Now $\Phi = min\ \Phi(S)$ for $|S| \leq \frac{n}{2}$ and hence $|\overline{S}| \geq \frac{n}{2}$. Thus $\Phi = \min\limits_{\substack{S \\ \pi(S) \leq 1/2}} \Phi(S) = \frac{1}{2}$. This gives a mixing time of

$$O\left(\frac{\ln \frac{1}{\pi_{\min}}}{\Phi^2 \epsilon^3}\right) = O\left(\frac{\ln n}{\frac{1}{4}\epsilon^3}\right) = O(\ln n).$$

A connected undirected graph

Next consider a random walk on a connected n vertex undirected graph where at each vertex all edges are equally likely. The stationary probability of a vertex equals the degree of the vertex divided by the sum of degrees which equals twice the number of edges. The sum of the vertex degrees is at most n^2 and thus, the steady state probability of each vertex is at least $\frac{1}{n^2}$. Since the degree of a vertex is at most n, the probability of each edge at a vertex is at least $\frac{1}{n}$. For any S, the total conductance of edges out of S is greater than or equal to

$$\frac{1}{n^2}\frac{1}{n} = \frac{1}{n^3}.$$

Thus, Φ is at least $\frac{1}{n^3}$. Since $\pi_{\min} \geq \frac{1}{n^2}$, $\ln \frac{1}{\pi_{min}} = O(\ln n)$. Thus, the mixing time is $O(n^6 \ln n/\varepsilon^2)$.

The Gaussian distribution on the interval [-1,1]

Consider the interval $[-1, 1]$. Let δ be a "grid size" specified later and let G be the graph consisting of a path on the $\frac{2}{\delta}+1$ vertices $\{-1, -1+\delta, -1+2\delta, \ldots, 1-\delta, 1\}$ having self loops at the two ends. Let $\pi_x = ce^{-\alpha x^2}$ for $x \in \{-1, -1+\delta, -1+2\delta, \ldots, 1-\delta, 1\}$ where $\alpha > 1$ and c has been adjusted so that $\sum_x \pi_x = 1$.

We now describe a simple Markov chain with the π_x as its stationary probability and argue its fast convergence. With the Metropolis-Hastings' construction, the transition probabilities are

$$p_{x,x+\delta} = \frac{1}{2}\min\left(1, \frac{e^{-\alpha(x+\delta)^2}}{e^{-\alpha x^2}}\right) \text{ and } p_{x,x-\delta} = \frac{1}{2}\min\left(1, \frac{e^{-\alpha(x-\delta)^2}}{e^{-\alpha x^2}}\right).$$

Let S be any subset of states with $\boldsymbol{\pi}(S) \leq \frac{1}{2}$. Consider the case when S is an interval $[k\delta, 1]$ for $k \geq 1$. It is easy to see that

$$\begin{aligned}\boldsymbol{\pi}(S) &\leq \int_{x=(k-1)\delta}^{\infty} ce^{-\alpha x^2}\, dx \\ &\leq \int_{(k-1)\delta}^{\infty} \frac{x}{(k-1)\delta} ce^{-\alpha x^2}\, dx \\ &= O\left(\frac{ce^{-\alpha((k-1)\delta)^2}}{\alpha(k-1)\delta}\right).\end{aligned}$$

Now there is only one edge from S to $\bar{S}$ and total conductance of edges out of S is

$$\sum_{i\in S}\sum_{j\notin S} \pi_i p_{ij} = \pi_{k\delta} p_{k\delta,(k-1)\delta} = \min(ce^{-\alpha k^2\delta^2}, ce^{-\alpha(k-1)^2\delta^2}) = ce^{-\alpha k^2\delta^2}.$$

Using $1 \leq k \leq 1/\delta$ and $\alpha \geq 1$, $\Phi(S)$ is

$$\begin{aligned}\Phi(S) = \frac{\text{flow}(S,\bar{S})}{\pi(S)} &\geq ce^{-\alpha k^2\delta^2}\frac{\alpha(k-1)\delta}{ce^{-\alpha((k-1)\delta)^2}} \\ &\geq \Omega(\alpha(k-1)\delta e^{-\alpha\delta^2(2k-1)}) \geq \Omega(\delta e^{-O(\alpha\delta)}).\end{aligned}$$

For $\delta < \frac{1}{\alpha}$, we have $\alpha\delta < 1$, so $e^{-O(\alpha\delta)} = \Omega(1)$, thus, $\Phi(S) \geq \Omega(\delta)$. Now, $\pi_{\min} \geq ce^{-\alpha} \geq e^{-1/\delta}$, so $\ln(1/\pi_{\min}) \leq 1/\delta$.

If S is not an interval of the form $[k, 1]$ or $[-1, k]$, then the situation is only better since there is more than one "boundary" point which contributes to flow$(S, \bar{S})$. We do not present this argument here. By Theorem 5.5 in $\Omega(1/\delta^3\varepsilon^2)$ steps, a walk gets within ε of the steady state distribution.

In these examples, we have chosen simple probability distributions. The methods extend to more complex situations.

5.5 Electrical Networks and Random Walks

In the next few sections, we study the relationship between electrical networks and random walks on undirected graphs. The graphs have nonnegative weights on each edge. A step is executed by picking a random edge from the current vertex with probability proportional to the edge's weight and traversing the edge.

An electrical network is a connected, undirected graph in which each edge (x, y) has a resistance $r_{xy} > 0$. In what follows, it is easier to deal with conductance defined as the reciprocal of resistance, $c_{xy} = \frac{1}{r_{xy}}$, rather than resistance. Associated with an electrical network is a random walk on the underlying graph defined by assigning a probability $p_{xy} = \frac{c_{xy}}{c_x}$ to the edge (x, y) incident to the vertex x, where the normalizing constant c_x equals $\sum\limits_y c_{xy}$. Note that although c_{xy} equals c_{yx}, the probabilities p_{xy} and p_{yx} may not be equal due to the normalization required to make the probabilities at each vertex sum to one. We shall soon see that there is a relationship between current flowing in an electrical network and a random walk on the underlying graph.

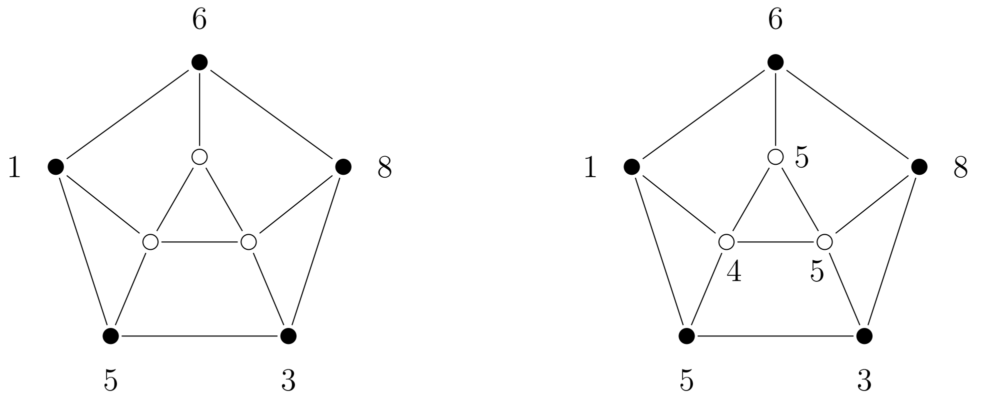
Since we assume that the undirected graph is connected, by Theorem 5.2 there is a unique stationary probability distribution.The stationary probability distribution is $\boldsymbol{\pi}$ where $\pi_x = \frac{c_x}{c_0}$ where $c_0 = \sum\limits_x c_x$. To see this, for all x and y

$$\pi_x p_{xy} = \frac{c_x}{c_0}\frac{c_{xy}}{c_x} = \frac{c_{xy}}{c_0} = \frac{c_y}{c_0}\frac{c_{yx}}{c_y} = \pi_y p_{yx}$$

and hence by Lemma 5.3, $\boldsymbol{\pi}$ is the unique stationary probability.

Harmonic functions

Harmonic functions are useful in developing the relationship between electrical networks and random walks on undirected graphs. Given an undirected graph, designate



Graph with boundary vertices dark and boundary conditions specified.

Values of harmonic function satisfying boundary conditions where the edge weights at each vertex are equal

Figure 5.5: Graph illustrating an harmonic function.

a nonempty set of vertices as boundary vertices and the remaining vertices as interior vertices. A harmonic function g on the vertices is one in which the value of the function at the boundary vertices is fixed to some boundary condition and the value of g at any interior vertex x is a weighted average of the values at all the adjacent vertices y, with weights p_{xy} satisfying $\sum_y p_{xy} = 1$ for each x. Thus, if at every interior vertex x for some set of weights p_{xy} satisfying $\sum_y p_{xy} = 1$, $g_x = \sum\limits_y g_y p_{xy}$, then g is an harmonic function.

Example: Convert an electrical network with conductances c_{xy} to a weighted, undirected graph with probabilities p_{xy}. Let $\mathbf{f}$ be a function satisfying $\mathbf{f}P = \mathbf{f}$ where P is the matrix of probabilities. It follows that the function $g_x = \frac{f_x}{c_x}$ is harmonic.

$$\begin{aligned} g_x &= \tfrac{f_x}{c_x} = \tfrac{1}{c_x} \sum_y f_y p_{yx} = \tfrac{1}{c_x} \sum_y f_y \tfrac{c_{yx}}{c_y} \\ &= \tfrac{1}{c_x} \sum_y f_y \tfrac{c_{xy}}{c_y} = \sum_y \tfrac{f_y}{c_y} \tfrac{c_{xy}}{c_x} = \sum_y g_y p_{xy} \end{aligned}$$

■

A harmonic function on a connected graph takes on its maximum and minimum on the boundary. Suppose the maximum does not occur on the boundary. Let S be the set of interior vertices at which the maximum value is attained. Since S contains no boundary vertices, $\bar{S}$ is nonempty. Connectedness implies that there is at least one edge (x, y) with $x \in S$ and $y \in \bar{S}$. The value of the function at x is the average of the value at its neighbors, all of which are less than or equal to the value at x and the value at y is strictly less, a contradiction. The proof for the minimum value is identical.

There is at most one harmonic function satisfying a given set of equations and boundary conditions. For suppose there were two solutions, $f(x)$ and $g(x)$. The difference of two solutions is itself harmonic. Since $h(x) = f(x) - g(x)$ is harmonic and has value zero on the boundary, by the min and max principles it has value zero everywhere. Thus $f(x) = g(x)$.

The analogy between electrical networks and random walks

There are important connections between electrical networks and random walks on undirected graphs. Choose two vertices a and b. For reference purposes let the voltage v_b equal zero. Attach a current source between a and b so that the voltage v_a equals one. Fixing the voltages at v_a and v_b induces voltages at all other vertices along with a current flow through the edges of the network. The analogy between electrical networks and random walks is the following. Having fixed the voltages at the vertices a and b, the voltage at an arbitrary vertex x equals the probability of a random walk starting at x reaching a before reaching b. If the voltage v_a is adjusted so that the current flowing into vertex a corresponds to one walk, then the current flowing through an edge is the net frequency with which a random walk from a to b traverses the edge.

Probabilistic interpretation of voltages

Before showing that the voltage at an arbitrary vertex x equals the probability of a random walk starting at x reaching a before reaching b, we first show that the voltages form a harmonic function. Let x and y be adjacent vertices and let i_{xy} be the current flowing through the edge from x to y. By Ohm's law,

$$i_{xy} = \frac{v_x - v_y}{r_{xy}} = (v_x - v_y)c_{xy}.$$

By Kirchhoff's law the currents flowing out of each vertex sum to zero.

$$\sum_y i_{xy} = 0$$

Replacing currents in the above sum by the voltage difference times the conductance yields

$$\sum_y (v_x - v_y)c_{xy} = 0$$

or

$$v_x \sum_y c_{xy} = \sum_y v_y c_{xy}.$$

Observing that $\sum\limits_y c_{xy} = c_x$ and that $p_{xy} = \frac{c_{xy}}{c_x}$, yields $v_x c_x = \sum\limits_y v_y p_{xy} c_x$. Hence, $v_x = \sum\limits_y v_y p_{xy}$. Thus, the voltage at each vertex x is a weighted average of the voltages at the adjacent vertices. Hence the voltages form a harmonic function with $\{a, b\}$ as the boundary.

Let p_x be the probability that a random walk starting at vertex x reaches a before b. Clearly $p_a = 1$ and $p_b = 0$. Since $v_a = 1$ and $v_b = 0$, it follows that $p_a = v_a$ and $p_b = v_b$. Furthermore, the probability of the walk reaching a from x before reaching b is the sum over all y adjacent to x of the probability of the walk going from x to y in the first step and then reaching a from y before reaching b. That is

$$p_x = \sum_y p_{xy} p_y.$$

Hence, p_x is the same harmonic function as the voltage function v_x and $\mathbf{v}$ and $\mathbf{p}$ satisfy the same boundary conditions at a and b.. Thus, they are identical functions. The probability of a walk starting at x reaching a before reaching b is the voltage v_x.

Probabilistic interpretation of current

In a moment, we will set the current into the network at a to have a value which we will equate with one random walk. We will then show that the current i_{xy} is the net frequency with which a random walk from a to b goes through the edge xy before reaching b. Let u_x be the expected number of visits to vertex x on a walk from a to b before reaching b. Clearly $u_b = 0$. Every time the walk visits x, x not equal to a, it must come to x from some vertex y. Thus, the number of visits to x before reaching b is the sum over all y of the number of visits u_y to y before reaching b times the probability p_{yx} of going from y to x. For x not equal to b or a

$$u_x = \sum_{y \neq b} u_y p_{yx}.$$

Since $u_b = 0$ and $c_x p_{xy} = c_y p_{yx}$

$$u_x = \sum_{\text{all } y} u_y \frac{c_x p_{xy}}{c_y}$$

and hence $\frac{u_x}{c_x} = \sum\limits_y \frac{u_y}{c_y} p_{xy}$. It follows that $\frac{u_x}{c_x}$ is harmonic with a and b as the boundary where the boundary conditions are $u_b = 0$ and u_a equals some fixed value. Now, $\frac{u_b}{c_b} = 0$. Setting the current into a to one, fixed the value of v_a. Adjust the current into a so that v_a equals $\frac{u_a}{c_a}$. Now $\frac{u_x}{c_x}$ and v_x satisfy the same boundary conditions and thus are the same harmonic function. Let the current into a correspond to one walk. Note that if the walk starts at a and ends at b, the expected value of the difference between the number of times the walk leaves a and enters a must be one. This implies that the amount of current into a corresponds to one walk.

Next we need to show that the current i_{xy} is the net frequency with which a random walk traverses edge xy.

$$i_{xy} = (v_x - v_y)c_{xy} = \left(\frac{u_x}{c_x} - \frac{u_y}{c_y}\right) c_{xy} = u_x \frac{c_{xy}}{c_x} - u_y \frac{c_{xy}}{c_y} = u_x p_{xy} - u_y p_{yx}$$

The quantity $u_x p_{xy}$ is the expected number of times the edge xy is traversed from x to y and the quantity $u_y p_{yx}$ is the expected number of times the edge xy is traversed from y to x. Thus, the current i_{xy} is the expected net number of traversals of the edge xy from x to y.

Effective resistance and escape probability

Set $v_a = 1$ and $v_b = 0$. Let i_a be the current flowing into the network at vertex a and out at vertex b. Define the *effective resistance* r_{eff} between a and b to be $r_{eff} = \frac{v_a}{i_a}$ and the *effective conductance* c_{eff} to be $c_{eff} = \frac{1}{r_{eff}}$. Define the *escape probability*, p_{escape}, to be the probability that a random walk starting at a reaches b before returning to a. We now show that the escape probability is $\frac{c_{eff}}{c_a}$. For convenience, assume that a and b are not adjacent. A slight modification of the argument suffices for the case when a and b are adjacent.

$$i_a = \sum_y (v_a - v_y)c_{ay}$$

Since $v_a = 1$,

$$\begin{aligned} i_a &= \sum_y c_{ay} - c_a \sum_y v_y \frac{c_{ay}}{c_a} \\ &= c_a \left[1 - \sum_y p_{ay} v_y \right]. \end{aligned}$$

For each y adjacent to the vertex a, p_{ay} is the probability of the walk going from vertex a to vertex y. Earlier we showed that v_y is the probability of a walk starting at y going to a before reaching b. Thus, $\sum\limits_y p_{ay} v_y$ is the probability of a walk starting at a returning to a before reaching b and $1 - \sum\limits_y p_{ay} v_y$ is the probability of a walk starting at a reaching b before returning to a. Thus, $i_a = c_a p_{escape}$. Since $v_a = 1$ and $c_{eff} = \frac{i_a}{v_a}$, it follows that $c_{eff} = i_a$. Thus, $c_{eff} = c_a p_{escape}$ and hence $p_{escape} = \frac{c_{eff}}{c_a}$.

For a finite connected graph, the escape probability will always be nonzero. Now consider an infinite graph such as a lattice and a random walk starting at some vertex a. Form a series of finite graphs by merging all vertices at distance d or greater from a into a single vertex b for larger and larger values of d. The limit of p_{escape} as d goes to infinity is the probability that the random walk will never return to a. If $p_{escape} \to 0$, then eventually any random walk will return to a. If $p_{escape} \to q$ where $q > 0$, then a fraction of the walks never return. Thus, the escape probability terminology.

5.6 Random Walks on Undirected Graphs with Unit Edge Weights

We now focus our discussion on random walks on undirected graphs with uniform edge weights. At each vertex, the random walk is equally likely to take any edge. This corresponds to an electrical network in which all edge resistances are one. Assume the graph is connected. We consider questions such as what is the expected time for a random walk starting at a vertex x to reach a target vertex y, what is the expected time until the random walk returns to the vertex it started at, and what is the expected time to reach every vertex?

Hitting time

The *hitting time* h_{xy}, sometimes called *discovery time*, is the expected time of a random walk starting at vertex x to reach vertex y. Sometimes a more general definition is given where the hitting time is the expected time to reach a vertex y from a given starting probability distribution.

One interesting fact is that adding edges to a graph may either increase or decrease h_{xy} depending on the particular situation. Adding an edge can shorten the distance from x to y thereby decreasing h_{xy} or the edge could increase the probability of a random walk going to some far off portion of the graph thereby increasing h_{xy}. Another interesting fact is that hitting time is not symmetric. The expected time to reach a vertex y from a vertex x in an undirected graph may be radically different from the time to reach x from y.

We start with two technical lemmas. The first lemma states that the expected time to traverse a path of n vertices is $\Theta\left(n^2\right)$.

Lemma 5.6 *The expected time for a random walk starting at one end of a path of n vertices to reach the other end is $\Theta\left(n^2\right)$.*

Proof: Consider walking from vertex 1 to vertex n in a graph consisting of a single path of n vertices. Let h_{ij}, $i < j$, be the hitting time of reaching j starting from i. Now $h_{12} = 1$ and

$$h_{i,i+1} = \tfrac{1}{2} + \tfrac{1}{2}(1 + h_{i-1,i+1}) = 1 + \tfrac{1}{2}\left(h_{i-1,i} + h_{i,i+1}\right) \quad 2 \le i \le n-1.$$

Solving for $h_{i,i+1}$ yields the recurrence

$$h_{i,i+1} = 2 + h_{i-1,i}.$$

Solving the recurrence yields

$$h_{i,i+1} = 2i - 1.$$

To get from 1 to n, go from 1 to 2, 2 to 3, etc. Thus

$$\begin{aligned} h_{1,n} &= \sum_{i=1}^{n-1} h_{i,i+1} = \sum_{i=1}^{n-1} (2i-1) \\ &= 2\sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} 1 \\ &= 2\frac{n\,(n-1)}{2} - (n-1) \\ &= (n-1)^2 . \end{aligned}$$

■

The lemma says that in a random walk on a line where we are equally likely to take one step to the right or left each time, the farthest we will go away from the start in n steps is $\Theta(\sqrt{n})$.

The next lemma shows that the expected time spent at vertex i by a random walk from vertex n to vertex n in a chain of n vertices is $2(i-1)$ for $2 \le i \le n-1$.

Lemma 5.7 *Consider a random walk from vertex 1 to vertex n in a chain of n vertices. Let t(i) be the expected time spent at vertex i. Then*

$$t\,(i) = \begin{cases} n-1 & i=1 \\ 2\,(n-i) & 2 \le i \le n-1 \\ 1 & i=n. \end{cases}$$

Proof: Now $t\,(n) = 1$ since the walk stops when it reaches vertex n. Half of the time when the walk is at vertex $n-1$ it goes to vertex n. Thus $t\,(n-1) = 2$. For $3 \le i < n-1$, $t\,(i) = \frac{1}{2}\left[t\,(i-1) + t\,(i+1)\right]$ and $t\,(1)$ and $t\,(2)$ satisfy $t\,(1) = \frac{1}{2}t\,(2) + 1$ and $t\,(2) = t\,(1) + \frac{1}{2}t\,(3)$. Solving for $t(i+1)$ for $3 \le i < n-1$ yields

$$t(i+1) = 2t(i) - t(i-1)$$

which has solution $t(i) = 2(n-i)$ for $3 \le i < n-1$. Then solving for $t(2)$ and $t(1)$ yields $t\,(2) = 2\,(n-2)$ and $t\,(1) = n-1$. Thus, the total time spent at vertices is

$$n-1+2\,(1+2+\cdots+n-2)+1 = (n-1)+2\frac{(n-1)(n-2)}{2}+1 = (n-1)^2+1$$

which is one more than h_{1n} and thus is correct. ■

Adding edges to a graph might either increase or decrease the hitting time h_{xy}. Consider the graph consisting of a single path of n vertices. Add edges to this graph to get the graph in Figure 5.6 consisting of a clique of size $n/2$ connected to a path of $n/2$ vertices.

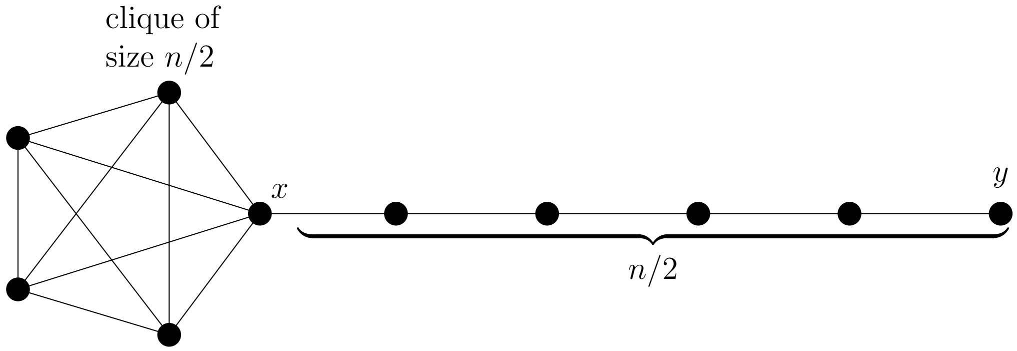


Figure 5.6: Illustration that adding edges to a graph can either increase or decrease hitting time.

Then add still more edges to get a clique of size n. Let x be the vertex at the midpoint of the original path and let y be the other endpoint of the path consisting of $n/2$ vertices as shown in the figure. In the first graph consisting of a single path of length n, $h_{xy} = \Theta\left(n^2\right)$. In the second graph consisting of a clique of size $n/2$ along with a path of length $n/2$, $h_{xy} = \Theta\left(n^3\right)$. To see this latter statement, note that starting at x, the walk will go down the path towards y and return to x $n/2$ times on average before reaching y for the first time. Each time the walk in the path returns to x, with probability $(n/2-1)/(n/2)$ it enters the clique and thus on average enters the clique $\Theta(n)$ times before starting down the path again. Each time it enters the clique, it spends $\Theta(n)$ time in the clique before returning to x. Thus, each time the walk returns to x from the path it spends $\Theta(n^2)$ time in the clique before starting down the path towards y for a total expected time that is $\Theta(n^3)$ before reaching y. In the third graph, which is the clique of size n, $h_{xy} = \Theta\left(n\right)$. Thus, adding edges first increased h_{xy} from n^2 to n^3 and then decreased it to n.

Hitting time is not symmetric even in the case of undirected graphs. In the graph of Figure 5.6, the expected time, h_{xy}, of a random walk from x to y, where x is the vertex of attachment and y is the other end vertex of the chain, is $\Theta(n^3)$. However, h_{yx} is $\Theta(n^2)$.

Commute time

The *commute time*, commute(x, y), is the expected time of a random walk starting at x reaching y and then returning to x. So commute$(x, y) = h_{xy} + h_{yx}$. Think of going from home to office and returning home. We now relate the commute time to an electrical quantity, the effective resistance. The *effective resistance* between two vertices x and y in an electrical network is the voltage difference between x and y when one unit of current is inserted at vertex x and withdrawn from vertex y.

Theorem 5.8 *Given an undirected graph, consider the electrical network where each edge of the graph is replaced by a one ohm resistor. Given vertices x and y, the commute time,* *commute(x, y), equals $2mr_{xy}$ where r_{xy} is the effective resistance from x to y and m is the number of edges in the graph.*

Proof: Insert at each vertex i a current equal to the degree d_i of vertex i. The total current inserted is $2m$ where m is the number of edges. Extract from a specific vertex j all of this $2m$ current. Let v_{ij} be the voltage difference from i to j. The current into i divides into the d_i resistors at vertex i. The current in each resistor is proportional to the voltage across it. Let k be a vertex adjacent to i. Then the current through the resistor between i and k is $v_{ij} - v_{kj}$, the voltage drop across the resister. The sum of the currents out of i through the resisters must equal d_i, the current injected into i.

$$d_i = \sum_{\substack{k \text{ adj} \\ \text{to } i}} (v_{ij} - v_{kj}) = d_i v_{ij} - \sum_{\substack{k \text{ adj} \\ \text{to } i}} v_{kj}.$$

Solving for v_{ij}

$$v_{ij} = 1 + \sum_{\substack{k \text{ adj} \\ \text{to } i}} \tfrac{1}{d_i} v_{kj} = \sum_{\substack{k \text{ adj} \\ \text{to } i}} \tfrac{1}{d_i}(1 + v_{kj}). \tag{5.4}$$

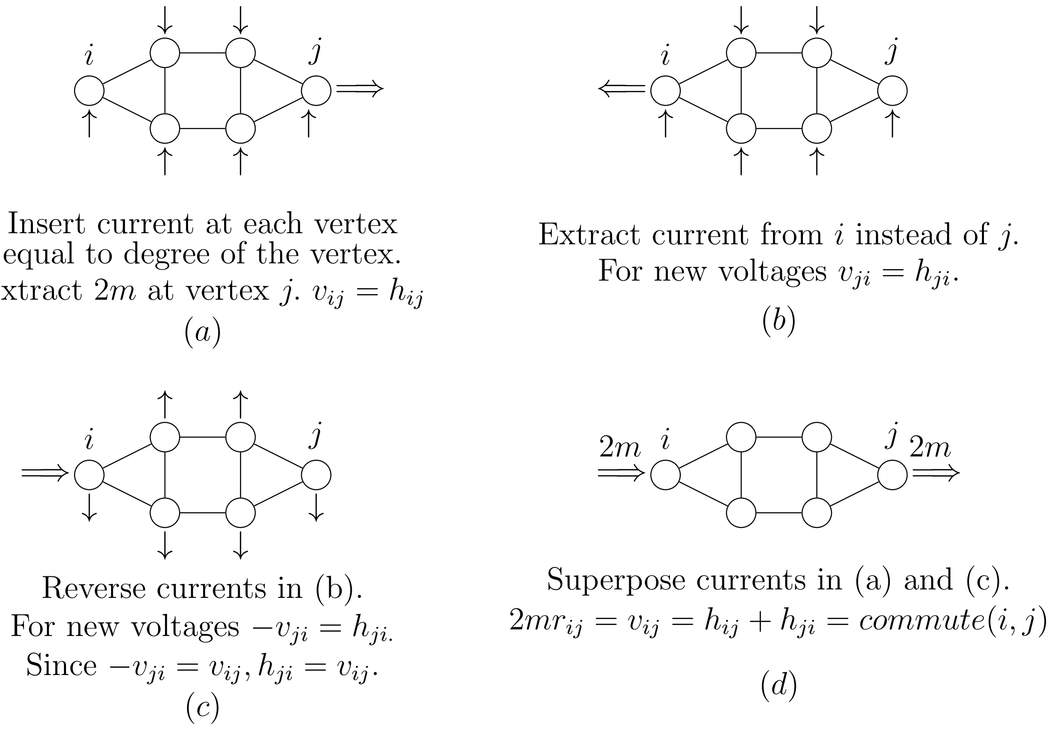
Now the hitting time from i to j is the average time over all paths from i to k adjacent to i and then on from k to j. This is given by

$$h_{ij} = \sum_{\substack{k \text{ adj} \\ \text{to } i}} \tfrac{1}{d_i}(1 + h_{kj}). \tag{5.5}$$

Subtracting (5.5) from (5.4), gives $v_{ij} - h_{ij} = \sum\limits_{\substack{k \text{ adj} \\ \text{to } i}} \frac{1}{d_i}(v_{kj} - h_{kj})$. Thus, the function $v_{ij} - h_{ij}$ is harmonic. Designate vertex j as the only boundary vertex. The value of $v_{ij} - h_{ij}$ at $i = j$, namely $v_{jj} - h_{jj}$, is zero, since both v_{jj} and h_{jj} are zero. So the function $v_{ij} - h_{ij}$ must be zero everywhere. Thus, the voltage v_{ij} equals the expected time h_{ij} from i to j.

To complete the proof, note that $h_{ij} = v_{ij}$ is the voltage from i to j when currents are inserted at all vertices in the graph and extracted at vertex j. If the current is extracted from i instead of j, then the voltages change and $v_{ji} = h_{ji}$ in the new setup. Finally, reverse all currents in this latter step. The voltages change again and for the new voltages $-v_{ji} = h_{ji}$. Since $-v_{ji} = v_{ij}$, we get $h_{ji} = v_{ij}$.

Thus, when a current is inserted at each vertex equal to the degree of the vertex and the current is extracted from j, the voltage v_{ij} in this set up equals h_{ij}. When we extract the current from i instead of j and then reverse all currents, the voltage v_{ij} in this new set up equals h_{ji}. Now, superpose both situations, i.e., add all the currents and



Insert current at each vertex equal to degree of the vertex. Extract $2m$ at vertex j. $v_{ij} = h_{ij}$

(a)

Extract current from i instead of j. For new voltages $v_{ji} = h_{ji}$.

(b)

Reverse currents in (b). For new voltages $-v_{ji} = h_{ji.}$ Since $-v_{ji} = v_{ij}, h_{ji} = v_{ij}$.

(c)

Superpose currents in (a) and (c). $2mr_{ij} = v_{ij} = h_{ij} + h_{ji} = commute(i, j)$

(d)

Figure 5.7: Illustration of proof that commute$(x, y) = 2mr_{xy}$ where m is the number of edges in the undirected graph and r_{xy} is the effective resistance between x and y.

voltages. By linearity, for the resulting v_{ij}, which is the sum of the other two v_{ij}'s, is $v_{ij} = h_{ij} + h_{ji}$. All currents cancel except the $2m$ amps injected at i and withdrawn at j. Thus, $2mr_{ij} = v_{ij} = h_{ij} + h_{ji} =$ commute(i, j) or commute$(i, j) = 2mr_{ij}$ where r_{ij} is the effective resistance from i to j. ■

The following corollary follows from Theorem 5.8 since the effective resistance r_{uv} is less than or equal to one when u and v are connected by an edge.

Corollary 5.9 *If vertices x and y are connected by an edge, then $h_{xy} + h_{yx} \leq 2m$ where m is the number of edges in the graph.*

Proof: If x and y are connected by an edge, then the effective resistance r_{xy} is less than or equal to one. ■

Corollary 5.10 *For vertices x and y in an n vertex graph, the commute time, commute(x, y), is less than or equal to n^3.*

Proof: By Theorem 5.8 the commute time is given by the formula commute$(x, y) = 2mr_{xy}$ where m is the number of edges. In an n vertex graph there exists a path from x

to y of length at most n. Since the resistance can not be greater than that of any path from x to y, $r_{xy} \le n$. Since the number of edges is at most $\binom{n}{2}$

$$\text{commute}(x, y) = 2mr_{xy} \le 2\binom{n}{2}n \cong n^3.$$

■

Again adding edges to a graph may increase or decrease the commute time. To see this consider three graphs: the graph consisting of a chain of n vertices, the graph of Figure 5.6, and the clique on n vertices.

Cover time

The *cover time*, $\text{cover}(x, G)$, is the expected time of a random walk starting at vertex x in the graph G to reach each vertex at least once. We write $\text{cover}(x)$ when G is understood. The cover time of an undirected graph G, denoted $\text{cover}(G)$, is

$$\text{cover}(G) = \max_{x} \text{cover}(x, G).$$

For cover time of an undirected graph, increasing the number of edges in the graph may increase or decrease the cover time depending on the situation. Again consider three graphs, a chain of length n which has cover time $\Theta(n^2)$, the graph in Figure 5.6 which has cover time $\Theta(n^3)$, and the complete graph on n vertices which has cover time $\Theta(n \log n)$. Adding edges to the chain of length n to create the graph in Figure 5.6 increases the cover time from n^2 to n^3 and then adding even more edges to obtain the complete graph reduces the cover time to $n \log n$.

Note: The cover time of a clique is $\theta(n \log n)$ since this is the time to select every integer out of n integers with high probability, drawing integers at random. This is called the *coupon collector problem.* The cover time for a straight line is $\Theta(n^2)$ since it is the same as the hitting time. For the graph in Figure 5.6, the cover time is $\Theta(n^3)$ since one takes the maximum over all start states and $\text{cover}(x, G) = \Theta\left(n^3\right)$ where x is the vertex of attachment.

Theorem 5.11 *Let G be a connected graph with n vertices and m edges. The time for a random walk to cover all vertices of the graph G is bounded above by $4m(n-1)$.*

Proof: Consider a depth first search of the graph G starting from some vertex z and let T be the resulting depth first search spanning tree of G. The depth first search covers every vertex. Consider the expected time to cover every vertex in the order visited by the depth first search. Clearly this bounds the cover time of G starting from vertex z. Note that each edge in T is traversed twice, once in each direction.

$$\text{cover}\left(z, G\right) \le \sum_{\substack{(x,y)\in T \\ (y,x)\in T}} h_{xy}.$$

If (x,y) is an edge in T, then x and y are adjacent and thus Corollary 5.9 implies $h_{xy} \leq 2m$. Since there are $n-1$ edges in the dfs tree and each edge is traversed twice, once in each direction, cover$(z) \leq 4m(n-1)$. This holds for all starting vertices z. Thus, cover$(G) \leq 4m(n-1)$. ∎

The theorem gives the correct answer of n^3 for the $n/2$ clique with the $n/2$ tail. It gives an upper bound of n^3 for the n-clique where the actual cover time is $n \log n$.

Let r_{xy} be the effective resistance from x to y. Define the resistance $r_{eff}(G)$ of a graph G by $r_{eff}(G) = \max\limits_{x,y}(r_{xy})$.

Theorem 5.12 *Let G be an undirected graph with m edges. Then the cover time for G is bounded by the following inequality*

$$mr_{eff}(G) \leq \textit{cover}(G) \leq 2e^3 mr_{eff}(G) \ln n + n$$

where e=2.71 is Euler's constant and $r_{eff}(G)$ is the resistance of G.

Proof: By definition $r_{eff}(G) = \max\limits_{x,y}(r_{xy})$. Let u and v be the vertices of G for which r_{xy} is maximum. Then $r_{eff}(G) = r_{uv}$. By Theorem 5.8, commute$(u,v) = 2mr_{uv}$. Hence $mr_{uv} = \frac{1}{2}$commute(u,v). Clearly the commute time from u to v and back to u is less than twice the $\max(h_{uv}, h_{vu})$. Finally, $\max(h_{uv}, h_{vu})$ is less than max(cover(u,G), cover(v,G)) which is clearly less than the cover time of G. Putting these facts together gives the first inequality in the theorem.

$$mr_{eff}(G) = mr_{uv} = \tfrac{1}{2}\text{commute}(u,v) \leq \max(h_{uv}, h_{vu}) \leq \text{cover}(G)$$

For the second inequality in the theorem, by Theorem 5.8, for any x and y, commute(x,y) equals $2mr_{xy}$ which is less than or equal to $2mr_{eff}(G)$, implying $h_{xy} \leq 2mr_{eff}(G)$. By the Markov inequality, since the expected time to reach y starting at any x is less than $2mr_{eff}(G)$, the probability that y is not reached from x in $2mr_{eff}(G)e^3$ steps is at most $\frac{1}{e^3}$. Thus, the probability that a vertex y has not been reached in $2e^3mr_{eff}(G)\log n$ steps is at most $\frac{1}{e^3}^{\ln n} = \frac{1}{n^3}$ because a random walk of length $2e^3mr(G)\log n$ is a sequence of $\log n$ independent random walks, each of length $2e^3mr(G)r_{eff}(G)$. Suppose after a walk of $2e^3mr_{eff}(G)\log n$ steps, vertices $v_1, v_2, \ldots, v_l$ had not been reached. Walk until v_1 is reached, then v_2, etc. By Corollary 5.10 the expected time for each of these is n^3, but since each happens only with probability $1/n^3$, we effectively take $O(1)$ time per v_i, for a total time at most n. More precisely,

$$\text{cover}(G) \leq 2e^3mr_{eff}(G)\log n + \sum_v \text{Prob}\left(v \text{ was not visited in the first } 2e^3mr_{eff}(G) \text{ steps}\right)n^3$$

$$\leq 2e^3mr_{eff}(G)\log n + \sum_v \frac{1}{n^3}n^3 \leq 2e^3mr_{eff}(G) + n.$$

∎

5.7 Random Walks in Euclidean Space

Many physical processes such as Brownian motion are modeled by random walks. Random walks in Euclidean d-space consisting of fixed length steps parallel to the coordinate axes are really random walks on a d-dimensional lattice and are a special case of random walks on graphs. In a random walk on a graph, at each time unit an edge from the current vertex is selected at random and the walk proceeds to the adjacent vertex. We begin by studying random walks on lattices.

Random walks on lattices

We now apply the analogy between random walks and current to lattices. Consider a random walk on a finite segment $-n, \ldots, -1, 0, 1, 2, \ldots, n$ of a one dimensional lattice starting from the origin. Is the walk certain to return to the origin or is there some probability that it will escape, i.e., reach the boundary before returning? The probability of reaching the boundary before returning to the origin is called the escape probability. We shall be interested in this quantity as n goes to infinity.

Convert the lattice to an electrical network by replacing each edge with a one ohm resister. Then the probability of a walk starting at the origin reaching n or $-n$ before returning to the origin is the escape probability given by

$$p_{escape} = \frac{c_{eff}}{c_a}$$

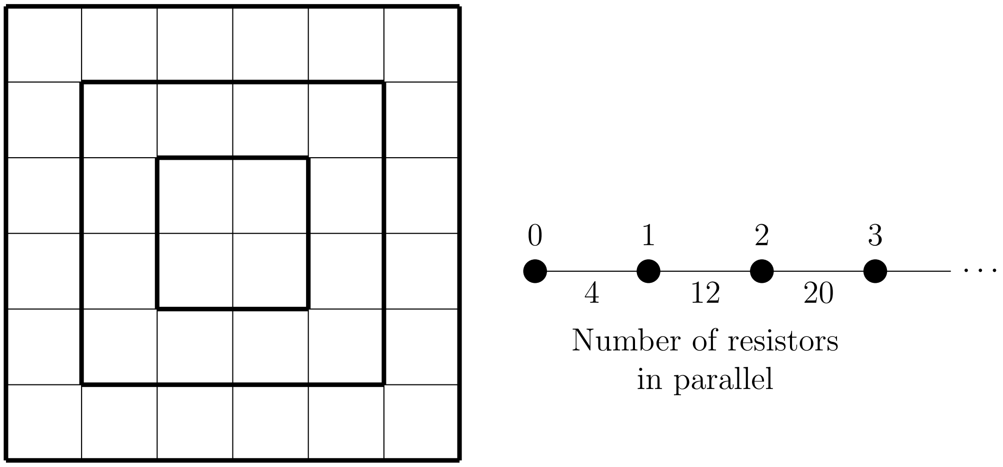
where c_{eff} is the effective conductance between the origin and the boundary points and c_a is the sum of the conductance's at the origin. In a d-dimensional lattice, $c_a = 2d$ assuming that the resistors have value one. For the d-dimensional lattice

$$p_{escape} = \frac{1}{2d\ r_{eff}}$$

In one dimension, the electrical network is just two series connections of n one ohm resistors connected in parallel. So as n goes to infinity, r_{eff} goes to infinity and the escape probability goes to zero as n goes to infinity. Thus, the walk in the unbounded one dimensional lattice will return to the origin with probability one. This is equivalent to flipping a balanced coin and keeping tract of the number of heads minus the number of tails. The count will return to zero infinitely often. By Hoffding-Chernoff inequality, in n steps with high probability, the walk will be within $O(\sqrt{n})$ of the origin. in n steps with high probability the walk will be within $\sqrt{n}$ distance of the origin.

Two dimensions

For the 2-dimensional lattice, consider a larger and larger square about the origin for the boundary as shown in Figure 5.8a and consider the limit of r_{eff} as the squares get larger. Shorting the resistors on each square can only reduce r_{eff}. Shorting the resistors



(a) (b)

Figure 5.8: 2-dimensional lattice along with the linear network resulting from shorting resistors on the concentric squares about the origin.

results in the linear network shown in Figure 5.8b. As the paths get longer, the number of resistors in parallel also increases. The resistor between vertex i and $i + 1$ is really $4(2i + 1)$ unit resistors in parallel. The effective resistance of $4(2i + 1)$ resistors in parallel is $1/4(2i + 1)$. Thus,

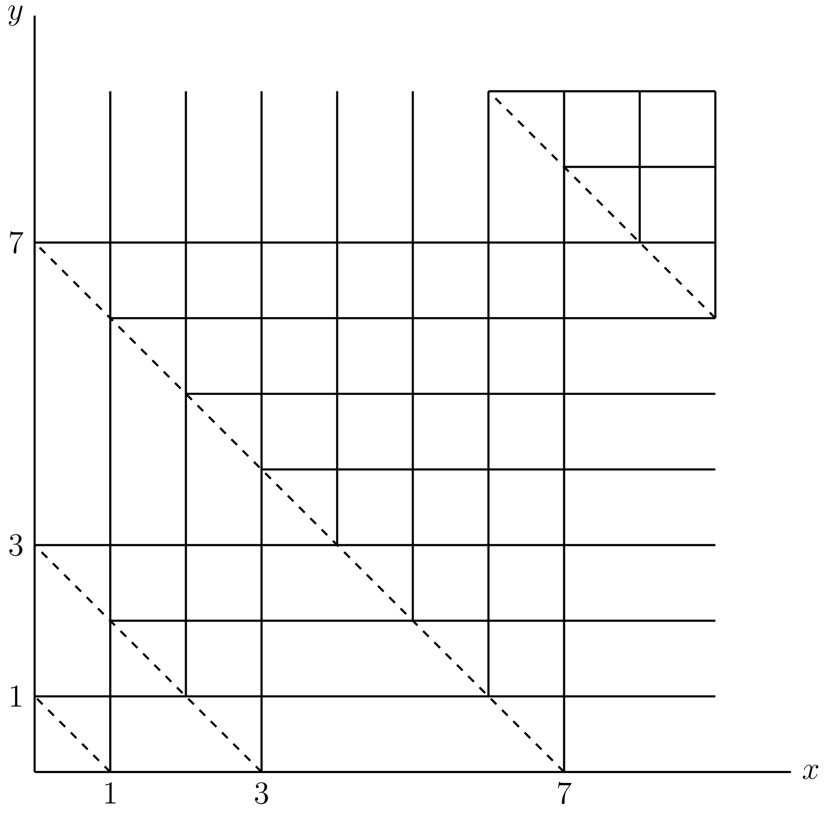
$$r_{eff} \geq \tfrac{1}{4} + \tfrac{1}{12} + \tfrac{1}{20} + \cdots = \tfrac{1}{4}(1 + \tfrac{1}{3} + \tfrac{1}{5} + \cdots) = \Theta(\ln n).$$

Since the lower bound on the effective resistance and hence the effective resistance goes to infinity, the escape probability goes to zero for the 2-dimensional lattice.

Three dimensions

In three dimensions, the resistance along any path to infinity grows to infinity but the number of paths in parallel also grows to infinity. It turns out there are a sufficient number of paths that r_{eff} remains finite and thus there is a nonzero escape probability. We will prove this now. First note that shorting any edge decreases the resistance, so we do not use shorting in this proof, since we seek to prove an upper bound on the resistance. Instead we remove some edges, which increases their resistance to infinity and hence increases the effective resistance, giving an upper bound. To simplify things we consider walks on on quadrant rather than the full grid. The resistance to infinity derived from only the quadrant is an upper bound on the resistance of the full grid.

The construction used in three dimensions is easier to explain first in two dimensions. Draw dotted diagonal lines at $x + y = 2^n - 1$. Consider two paths that start at the origin. One goes up and the other goes to the right. Each time a path encounters a dotted



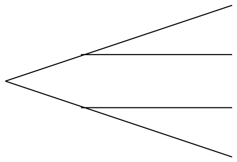


Figure 5.9: Paths in a 2-dimensional lattice obtained from the 3-dimensional construction applied in 2-dimensions.

diagonal line, split the path into two, one which goes right and the other up. Where two paths cross, split the vertex into two, keeping the paths separate. By a symmetry argument, splitting the vertex does not change the resistance of the network. Remove all resistors except those on these paths. The resistance of the original network is less than that of the tree produced by this process since removing a resistor is equivalent to increasing its resistance to infinity.

The distances between splits increase and are 1, 2, 4, etc. At each split the number of paths in parallel doubles. See Figure 5.10. Thus, the resistance to infinity in this two dimensional example is

$$\frac{1}{2} + \frac{1}{4}2 + \frac{1}{8}4 + \cdots = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \cdots = \infty.$$

In the analogous three dimensional construction, paths go up, to the right, and out of the plane of the paper. The paths split three ways at planes given by $x + y + z = 2^n - 1$.

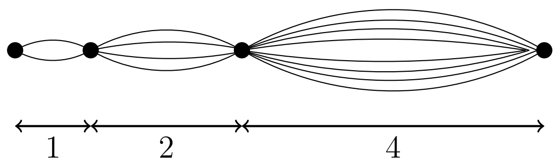


Figure 5.10: Paths obtained from 2-dimensional lattice. Distances between splits double as do the number of parallel paths.

Each time the paths split the number of parallel segments triple. Segments of the paths between splits are of length 1, 2, 4, etc. and the resistance of the segments are equal to the lengths. The resistance out to infinity for the tree is

$$\tfrac{1}{3}+\tfrac{1}{9}2+\tfrac{1}{27}4+\cdots=\tfrac{1}{3}\left(1+\tfrac{2}{3}+\tfrac{4}{9}+\cdots\right)=\tfrac{1}{3}\tfrac{1}{1-\frac{2}{3}}=1$$

The resistance of the three dimensional lattice is less. It is important to check that the paths are edge-disjoint and so the tree is a subgraph of the lattice. Going to a subgraph is equivalent to deleting edges which only increases the resistance. That is why the resistance of the lattice is less than that of the tree. Thus, in three dimensions the escape probability is nonzero. The upper bound on r_{eff} gives the lower bound

$$p_{escape}=\tfrac{1}{2d}\tfrac{1}{r_{eff}}\geq\tfrac{1}{6}.$$

A lower bound on r_{eff} gives an upper bound on p_{escape}. To get the upper bound on p_{escape}, short all resistors on surfaces of boxes at distances $1, 2, 3,$, etc. Then

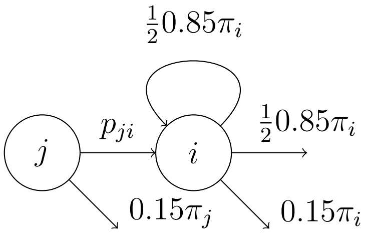
$$r_{eff}\geq\tfrac{1}{6}\left[1+\tfrac{1}{9}+\tfrac{1}{25}+\cdots\right]\geq\tfrac{1.23}{6}\geq 0.2$$

This gives

$$p_{escape}=\tfrac{1}{2d}\tfrac{1}{r_{eff}}\leq\tfrac{5}{6}.$$

5.8 The Web as a Markov Chain

A modern application of random walks on directed graphs comes from trying to establish the importance of pages on the World Wide Web. One way to do this would be to take a random walk on the web viewed as a directed graph with an edge corresponding to each hypertext link and rank pages according to their stationary probability. A connected, undirected graph is strongly connected in that one can get from any vertex to any other vertex and back again. Often the directed case is not strongly connected. One difficulty occurs if there is a vertex with no out edges. When the walk encounters this vertex the walk disappears. Another difficulty is that a vertex or a strongly connected component with no in edges is never reached. One way to resolve these difficulties is to introduce a random restart condition. At each step, with some probability r, jump to a vertex selected uniformly at random and with probability $1-r$ select an edge at random



$$\pi_i = 0.85\pi_j p_{ji} + \tfrac{0.85}{2}\pi_i$$

$$\pi_i = 1.48\pi_j p_{ji}$$

Figure 5.11: Impact on page rank of adding a self loop

and follow it. If a vertex has no out edges, the value of r for that vertex is set to one. This has the effect of converting the graph to a strongly connected graph so that the stationary probabilities exist.

Page rank and hitting time

The page rank of a vertex in a directed graph is the stationary probability of the vertex, where we assume a positive restart probability of say $r = 0.15$. The restart ensures that the graph is strongly connected. The page rank of a page is the fractional frequency with which the page will be visited over a long period of time. If the page rank is p, then the expected time between visits or return time is $1/p$. Notice that one can increase the pagerank of a page by reducing the return time and this can be done by creating short cycles.

Consider a vertex i with a single edge in from vertex j and a single edge out. The stationary probability $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi} P = \boldsymbol{\pi}$, and thus

$$\pi_i = \pi_j p_{ji}.$$

Adding a self-loop at i, results in a new equation

$$\pi_i = \pi_j p_{ji} + \frac{1}{2}\pi_i$$

or

$$\pi_i = 2\ \pi_j p_{ji}.$$

Of course, π_j would have changed too, but ignoring this for now, pagerank is doubled by the addition of a self-loop. Adding k self loops, results in the equation

$$\pi_i = \pi_j p_{ji} + \frac{k}{k+1}\pi_i,$$

and again ignoring the change in π_j, we now have $\pi_i = (k+1)\pi_j p_{ji}$. What prevents one from increasing the page rank of a page arbitrarily? The answer is the restart. We neglected the 0.15 probability that is taken off for the random restart. With the restart taken into account, the equation for π_i when there is no self-loop is

$$\pi_i = 0.85\pi_j p_{ji}$$

whereas, with k self-loops, the equation is

$$\pi_i = 0.85\pi_j p_{ji} + 0.85\frac{k}{k+1}\pi_i.$$

Solving for π_i yields

$$\pi_i = \frac{0.85k + 0.85}{0.15k + 1}\pi_j p_{ji}$$

which for $k = 1$ is $\pi_i = 1.48\pi_j P_{ji}$ and in the limit as $k \to \infty$ is $\pi_i = 5.67\pi_j p_{ji}$. Adding a single loop only increases pagerank by a factor of 1.74 and adding k loops increases it by at most a factor of 6.67 for arbitrarily large k.

Hitting time

Related to page rank is a quantity called hitting time. Hitting time is closely related to return time and thus to the reciprocal of page rank. One way to return to a vertex v is by a path in the graph from v back to v. Another way is to start on a path that encounters a restart, followed by a path from the random restart vertex to v. The time to reach v after a restart is the hitting time. Thus, return time is clearly less than the expected time until a restart plus hitting time. The fastest one could return would be if there were only paths of length two since self loops are ignored in calculating page rank. If r is the restart value, then the loop would be traversed with at most probability $(1-r)^2$. With probability $r + (1-r)\,r = (2-r)\,r$ one restarts and then hits v. Thus, the return time is at least $2\,(1-r)^2 + (2-r)\,r \times$ (hitting time). Combining these two bounds yields

$$2\,(1-r)^2 + (2-r)\,rE\,(\text{hitting time}) \le E\,(\text{return time}) \le E\,(\text{hitting time})\,.$$

The relationship between return time and hitting time can be used to see if a vertex has unusually high probability of short loops. However, there is no efficient way to compute hitting time for all vertices as there is for return time. For a single vertex v, one can compute hitting time by removing the edges out of the vertex v for which one is computing hitting time and then run the page rank algorithm for the new graph. The hitting time for v is the reciprocal of the page rank in the graph with the edges out of v removed. Since computing hitting time for each vertex requires removal of a different set of edges, the algorithm only gives the hitting time for one vertex at a time. Since one is probably only interested in the hitting time of vertices with low hitting time, an alternative would be to use a random walk to estimate the hitting time of low hitting time vertices.

Spam

Suppose one has a web page and would like to increase its page rank by creating some other web pages with pointers to the original page. The abstract problem is the following. We are given a directed graph G and a vertex v whose page rank we want to increase. We may add new vertices to the graph and add edges from v or from the new vertices to any vertices we want. We cannot add edges out of other vertices. We can also delete edges from v.

The page rank of v is the stationary probability for vertex v with random restarts. If we delete all existing edges out of v, create a new vertex u and edges (v, u) and (u, v), then the page rank will be increased since any time the random walk reaches v it will be captured in the loop $v \to u \to v$. A search engine can counter this strategy by more frequent random restarts.

A second method to increase page rank would be to create a star consisting of the vertex v at its center along with a large set of new vertices each with a directed edge to v. These new vertices will sometimes be chosen as the target of the random restart and hence the vertices increase the probability of the random walk reaching v. This second method is countered by reducing the frequency of random restarts.

Notice that the first technique of capturing the random walk increases page rank but does not effect hitting time. One can negate the impact of someone capturing the random walk on page rank by increasing the frequency of random restarts. The second technique of creating a star increases page rank due to random restarts and decreases hitting time. One can check if the page rank is high and hitting time is low in which case the page rank is likely to have been artificially inflated by the page capturing the walk with short cycles.

Personalized page rank

In computing page rank, one uses a restart probability, typically 0.15, in which at each step, instead of taking a step in the graph, the walk goes to a vertex selected uniformly at random. In personalized page rank, instead of selecting a vertex uniformly at random, one selects a vertex according to a personalized probability distribution. Often the distribution has probability one for a single vertex and whenever the walk restarts it restarts at that vertex.

Algorithm for computing personalized page rank

First, consider the normal page rank. Let α be the restart probability with which the random walk jumps to an arbitrary vertex. With probability $1 - \alpha$ the random walk selects a vertex uniformly at random from the set of adjacent vertices. Let $\mathbf{p}$ be a row vector denoting the page rank and let A be the adjacency matrix with rows normalized to sum to one. Then

$$\mathbf{p} = \tfrac{\alpha}{n}\,(1, 1, \ldots, 1) + (1-\alpha)\,\mathbf{p}A$$

$$\mathbf{p}[I-(1-\alpha)A] = \frac{\alpha}{n}(1, 1, \ldots, 1)$$

or

$$\mathbf{p} = \tfrac{\alpha}{n}\,(1, 1, \ldots, 1)\,[I-(1-\alpha)\,A]^{-1}.$$

Thus, in principle, $\mathbf{p}$ can be found by computing the inverse of $[I-(1-\alpha)A]^{-1}$. But this is far from practical since for the whole web one would be dealing with matrices with billions of rows and columns. A more practical procedure is to run the random walk and observe using the basics of the power method in Chapter 3 that the process converges to the solution $\mathbf{p}$.

For the personalized page rank, instead of restarting at an arbitrary vertex, the walk restarts at a designated vertex. More generally, it may restart in some specified neighborhood. Suppose the restart selects a vertex using the probability distribution s. Then, in the above calculation replace the vector $\frac{1}{n}\,(1, 1, \ldots, 1)$ by the vector $\mathbf{s}$. Again, the computation could be done by a random walk. But, we wish to do the random walk calculation for personalized pagerank quickly since it is to be performed repeatedly. With more care this can be done, though we do not describe it here.

5.9 Bibliographic Notes

The material on the analogy between random walks on undirected graphs and electrical networks is from [DS84] as is the material on random walks in Euclidean space. Additional material on Markov chains can be found in [MR95b], [MU05], and [per10]. For material on Markov Chain Monte Carlo methods see [Jer98] and [Liu01].

The use of normalized conductance to prove convergence of Markov Chains is by Sinclair and Jerrum, [SJ] and Alon [Alo86]. A polynomial time bounded Markov chain based method for estimating the volume of convex sets was developed by Dyer, Frieze and Kannan [DFK91].

5.10 Exercises

Exercise 5.1 *The Fundamental Theorem of Markov chains proves that for a connected Markov chain, the long-term average distribution* $\mathbf{a}^{(\mathbf{t})}$ *converges to a stationary distribution. Does the t step distribution* $\mathbf{p}^{(\mathbf{t})}$ *also converge for every connected Markov Chain ? Consider the following examples: (i) A two-state chain with* $p_{12} = p_{21} = 1$*. (ii) A three state chain with* $p_{12} = p_{23} = p_{31} = 1$ *and the other* $p_{ij} = 0$*. Generalize these examples to produce Markov Chains with many states.*

Exercise 5.2

Exercise 5.3 *Let* $p(\mathbf{x})$*, where* $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ $x_i \in \{0, 1\}$*, be a multivariate probability distribution. For* $d = 100$*, how would you estimate the marginal distribution*

$$p(x_1) = \sum_{x_2,\ldots,x_d} p(x_1, x_2, \ldots, x_d)?$$

Exercise 5.4 *Prove Proposition 5.4 that for two probability distributions* $\mathbf{p}, \mathbf{q}$, $|\mathbf{p}-\mathbf{q}|_1 = 2\sum_i (p_i - q_i)^+$.

Exercise 5.5 *Suppose* S *is a subset of at most* $n^2/2$ *points in the* $n \times n$ *lattice. Show that*

$$\left|\{(i,j) \in S \middle| \textit{all elements in row } i \textit{ and all elements in column } j \textit{ are in } S\}\right| \leq |S|/2.$$

Exercise 5.6 *Show that the stationary probabilities of the chain described in the Gibbs sampler is the correct* p*.*

Exercise 5.7 *A Markov chain is said to be symmetric if for all* i *and* j, $p_{ij} = p_{ji}$*. What is the stationary distribution of a connected symmetric chain? Prove your answer.*

Exercise 5.8 *How would you integrate a multivariate polynomial distribution over some region?*

Exercise 5.9 *Given a time-reversible Markov chain, modify the chain as follows. At the current state, stay put (no move) with probability 1/2. With the other probability 1/2, move as in the old chain. Show that the new chain has the same stationary distribution. What happens to the convergence time in this modification?*

Exercise 5.10 *Using the Metropolis-Hasting Algorithm create a Markov chain whose stationary probability is that given in the following table.*

x_1x_2	00	01	02	10	11	12	20	21	22
Prob	1/16	1/8	1/16	1/8	1/4	1/8	1/16	1/8	1/16

Exercise 5.11 *Let* $\mathbf{p}$ *be a probability vector (nonnegative components adding up to 1) on the vertices of a connected graph. Set* p_{ij} *(the transition probability from* i *to* j*) to* p_j *for all* $i \neq j$ *which are adjacent in the graph. Show that the stationary probability vector for the chain is* $\mathbf{p}$*. Is running this chain an efficient way to sample according to a distribution close to* $\mathbf{p}$*? Think, for example, of the graph* G *being the* $n \times n \times n \times \cdots n$ *grid.*

Exercise 5.12 *Construct the edge probability for a three state Markov chain where each pair of states is connected by an edge so that the stationary probability is* $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)$.

Exercise 5.13 *Consider a three state Markov chain with stationary probability* $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)$. *Consider the Metropolis-Hastings algorithm with* G *the complete graph on these three vertices. What is the expected probability that we would actually make a move along a selected edge?*

Exercise 5.14 *Try Gibbs sampling on* $p\left(x\right) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$.

What happens? How does the Metropolis Hasting Algorithm do?

Exercise 5.15 *Consider* $p(\mathbf{x})$*, where,* $\mathbf{x} = (x_1, \ldots, x_{100})$ *and* $p\left(\mathbf{0}\right) = \frac{1}{2}$, $p\left(\mathbf{x}\right) = \frac{1}{(2^{100}-1)} \quad \mathbf{x} \neq \mathbf{0}$. *How does Gibbs sampling behave?*

Exercise 5.16 *Construct, program, and execute an algorithm to compute the volume of a unit radius sphere in 20 dimensions by carrying out a random walk on a* 20 *dimensional grid with 0.1 spacing.*

Exercise 5.17 *Given a graph* G *and an integer* k *how would you generate connected subgraphs of* G *with* k *vertices with probability proportional to the number of edges in the subgraph induced on those vertices? The probabilities need not be exactly proportional to the number of edges and you are not expected to prove your algorithm for this problem.*

Exercise 5.18 *Suppose one wishes to generate uniformly at random regular, degree three undirected, connected multi-graphs each with 1,000 vertices. A multi-graph may have multiple edges between a pair of vertices and self loops. One decides to do this by a Markov Chain Monte Carlo technique. They design a network where each vertex is a regular degree three, 1,000 vertex multi-graph. For edges they say that the vertices corresponding to two graphs are connected by an edge if one graph can be obtained from the other by a flip of a pair of disjoint edges. In a flip, a pair of edges* (a, b) *and* (c, d) *are replaced by* (a, c) *and* (b, d).

1. *Prove that a swap on a connected multi-graph results in a connected multi-graph.*

2. *Prove that the network whose vertices correspond to the desired graphs is connected.*

3. *Prove that the stationary probability of the random walk is uniform.*

4. *Give an upper bound on the diameter of the network.*

In order to use a random walk to generate the graphs uniformly at random, the random walk must rapidly converge to the stationary probability. Proving this is beyond the material in this book.

Exercise 5.19 *What is the mixing time for*

1. *Two cliques connected by a single edge?*
2. *A graph consisting of an n vertex clique plus one additional vertex connected to one vertex in the clique.*

Exercise 5.20 *What is the mixing time for*

1. *$G(n,p)$ with $p = \frac{\log n}{n}$?*
2. *a circle with n vertices where at each vertex an edge has been added to another vertex chosen at random. On average each vertex will have degree four, two circle edges, and edge from that vertex to a vertex chosen at random, and possible some edges that are the ends of the random edges from other vertices.*

Exercise 5.21 *Show that for the $n \times n \times \cdots \times n$ grid in d space, the normalized conductance is $\Omega(1/dn)$.*
Hint: The argument is a generalization of the argument in Exercise 5.5. Argue that for any subset S containing at most $1/2$ the grid points, for at least $1/2$ the grid points in S, among the d coordinate lines through the point, at least one intersects $\bar{S}$.

1. *What is the set of possible harmonic functions on a connected graph if there are only interior vertices and no boundary vertices that supply the boundary condition?*
2. *Let q_x be the stationary probability of vertex x in a random walk on an undirected graph where all edges at a vertex are equally likely and let d_x be the degree of vertex x. Show that $\frac{q_x}{d_x}$ is a harmonic function.*
3. *If there are multiple harmonic functions when there are no boundary conditions, why is the stationary probability of a random walk on an undirected graph unique?*
4. *What is the stationary probability of a random walk on an undirected graph?*

Exercise 5.22 *In Section* **??** *we associate a graph and edge probabilities with an electric network such that voltages and currents in the electrical network corresponded to properties of random walks on the graph. Can we go in the reverse order and construct the equivalent electrical network from a graph with edge probabilities?*

Exercise 5.23 *Given an undirected graph consisting of a single path of five vertices numbered 1 to 5, what is the probability of reaching vertex 1 before vertex 5 when starting at vertex 4.*

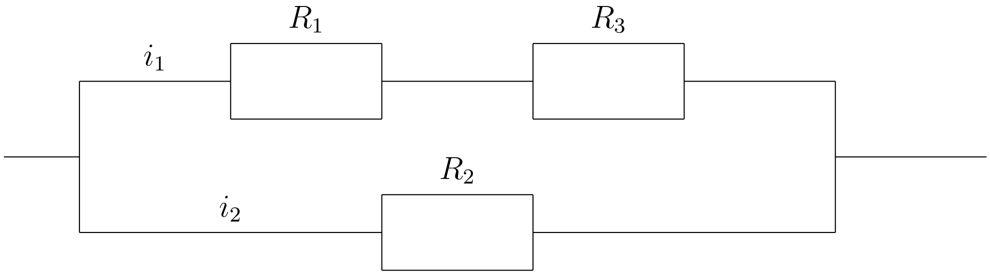


Figure 5.12: An electrical network of resistors.

Exercise 5.24 *Consider the electrical resistive network in Figure 5.12 consisting of vertices connected by resistors. Kirchoff's law states that the currents at each vertex sum to zero. Ohm's law states that the voltage across a resistor equals the product of the resistance times the current through it. Using these laws calculate the effective resistance of the network.*

Exercise 5.25 *Consider the electrical network of Figure 5.13.*

1. *Set the voltage at a to one and at b to zero. What are the voltages at c and d?*
2. *What is the current in the edges a to c, a to d, c to d. c to b and d to b?*
3. *What is the effective resistance between a and b?*
4. *Convert the electrical network to a graph. What are the edge probabilities at each vertex?*
5. *What is the probability of a walk starting at c reaching a before b? a walk starting at d reaching a before b/?*
6. *What is the net frequency that a walk from a to b goes through the edge from c to d?*
7. *What is the probability that a random walk starting at a will return to a before reaching b?*

Exercise 5.26 *Consider a graph corresponding to an electrical network with vertices a and b. Prove directly that $\frac{c_{eff}}{c_a}$ must be less than or equal to one. We know that this is the escape probability and must be at most 1. But, for this exercise, do not use that fact.*

Exercise 5.27 *(Thomson's Principle) The energy dissipated by the resistance of edge xy in an electrical network is given by $i_{xy}^2 r_{xy}$. The total energy dissipation in the network*

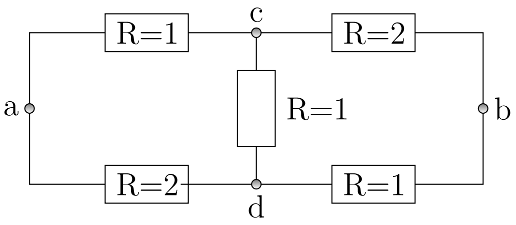


Figure 5.13: An electrical network of resistors.

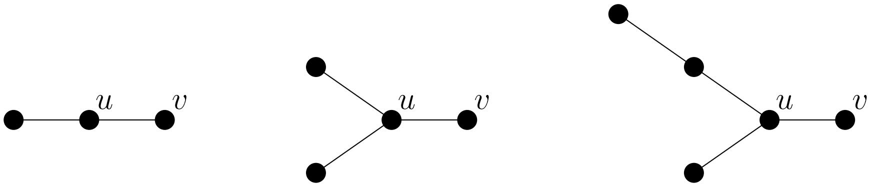


Figure 5.14: Three graphs

is $E = \frac{1}{2} \sum\limits_{x,y} i_{xy}^2 r_{xy}$ *where the* $\frac{1}{2}$ *accounts for the fact that the dissipation in each edge is counted twice in the summation. Show that the actual current distribution is that distribution satisfying Ohm's law that minimizes energy dissipation.*

Exercise 5.28 *(Rayleigh's law) Prove that reducing the value of a resistor in a network cannot increase the effective resistance. Prove that increasing the value of a resistor cannot decrease the effective resistance. You may use Thomson's principle Exercise 5.27.*

Exercise 5.29 *What is the hitting time* h_{uv} *for two adjacent vertices on a cycle of length* n*? What is the hitting time if the edge* (u, v) *is removed?*

Exercise 5.30 *What is the hitting time* h_{uv} *for the three graphs if Figure 5.14.*

Exercise 5.31 *Show that adding an edge can either increase or decrease hitting time by calculating* h_{24} *for the three graphs in Figure 5.15.*

Exercise 5.32 *Consider the* n *vertex connected graph shown in Figure 5.16 consisting of an edge* (u, v) *plus a connected graph on* $n - 1$ *vertices and* m *edges. Prove that* $h_{uv} = 2m + 1$ *where* m *is the number of edges in the* $n - 1$ *vertex subgraph.*

Exercise 5.33 *What is the most general solution to the difference equation* $t(i + 2) - 5t(i + 1) + 6t(i) = 0$*. How many boundary conditions do you need to make the solution unique?*

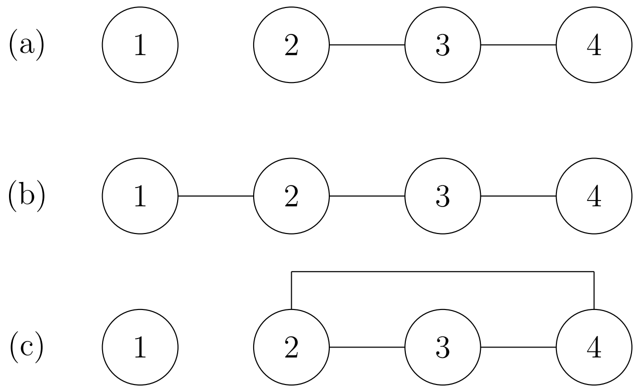


Figure 5.15: Three graph

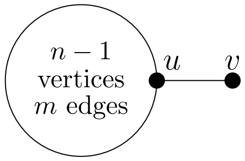


Figure 5.16: A connected graph consisting of $n - 1$ vertices and m edges along with a single edge (u, v).

Exercise 5.34 *Given the difference equation $a_kt(i+k)+a_{k-1}t(i+k-1)+\cdots+a_1t(i+1)+a_0t(i)=0$ the polynomial $a_kt^k+a_{k-i}t^{k-1}+\cdots+a_1t+a_0=0$ is called the characteristic polynomial.*

1. *If the equation has a set of r distinct roots, what is the most general form of the solution?*
2. *If the roots of the characteristic polynomial are not unique what is the most general form of the solution?*
3. *What is the dimension of the solution space?*
4. *If the difference equation is not homogeneous and f(i) is a specific solution to the nonhomogeneous difference equation, what is the full set of solutions to the difference equation?*

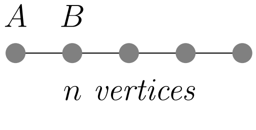
Exercise 5.35 *Given the integers 1 to n, what is the expected number of draws with replacement until the integer 1 is drawn.*

Exercise 5.36 *Consider the set of integers $\{1, 2, \ldots, n\}$. What is the expected number of draws d with replacement so that every integer is drawn?*

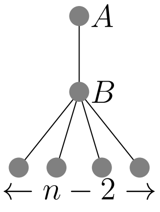
Exercise 5.37 *Consider a random walk on a clique of size n. What is the expected number of steps before a given vertex is reached?*

Exercise 5.38 *Show that adding an edge to a graph can either increase or decrease commute time.*

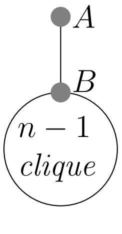
Exercise 5.39 *For each of the three graphs below what is the return time starting at vertex A? Express your answer as a function of the number of vertices, n, and then express it as a function of the number of edges m.*



a



b



c

Exercise 5.40 *Suppose that the clique in Exercise 5.39 was an arbitrary graph with $m-1$ edges. What would be the return time to A in terms of m, the total number of edges.*

Exercise 5.41 *Suppose that the clique in Exercise 5.39 was an arbitrary graph with $m-d$ edges and there were d edges from A to the graph. What would be the expected length of a random path starting at A and ending at A after returning to A exactly d times.*

Exercise 5.42 *Given an undirected graph with a component consisting of a single edge find two eigenvalues of the Laplacian $L = D - A$ where D is a diagonal matrix with vertex degrees on the diagonal and A is the adjacency matrix of the graph.*

Exercise 5.43 *A researcher was interested in determining the importance of various edges in an undirected graph. He computed the stationary probability for a random walk on the graph and let p_i be the probability of being at vertex i. If vertex i was of degree d_i, the frequency that edge (i, j) was traversed from i to j would to $\frac{1}{d_i}p_i$ and the frequency that the edge was traversed in the opposite direction would be $\frac{1}{d_j}p_j$. Thus, he assigned an importance of $\left|\frac{1}{d_i}p_i - \frac{1}{d_j}p_j\right|$ to the edge. What is wrong with his idea?*

Exercise 5.44 *Prove that two independent random walks starting at the origin on a two dimensional lattice will eventually meet with probability one.*

Exercise 5.45 *Suppose two individuals are flipping balanced coins and each is keeping tract of the number of heads minus the number of tails. Will both individual's count return to zero at the same time?*

Exercise 5.46 *Consider the lattice in 2-dimensions. In each square add the two diagonal edges. What is the escape probability for the resulting graph?*

Exercise 5.47 *Determine by simulation the escape probability for the 3-dimensional lattice.*

Exercise 5.48 *What is the escape probability for a random walk starting at the root of an infinite binary tree?*

Exercise 5.49 *Consider a random walk on the positive half line, that is the integers $0, 1, 2, \ldots$. At the origin, always move right one step. At all other integers move right with probability 2/3 and left with probability 1/3. What is the escape probability?*

Exercise 5.50 *Consider the graphs in Figure 5.17. Calculate the stationary distribution for a random walk on each graph and the flow through each edge. What condition holds on the flow through edges in the undirected graph? In the directed graph?*

Exercise 5.51 *Create a random directed graph with 200 vertices and roughly eight edges per vertex. Add k new vertices and calculate the page rank with and without directed edges from the k added vertices to vertex 1. How much does adding the k edges change the page rank of vertices for various values of k and restart frequency? How much does adding a loop at vertex 1 change the page rank? To do the experiment carefully one needs to consider the page rank of a vertex to which the star is attached. If it has low page rank its page rank is likely to increase a lot.*

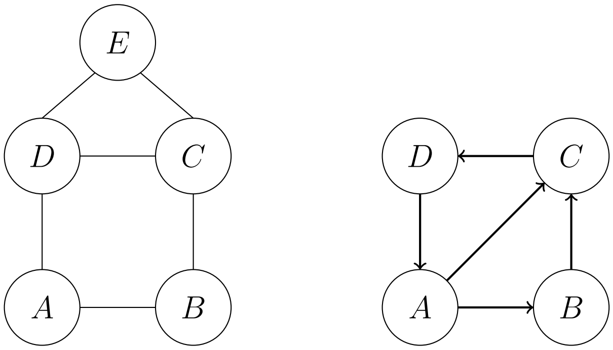


Figure 5.17: An undirected and a directed graph.

Exercise 5.52 *Repeat the experiment in Exercise 5.51 for hitting time.*

Exercise 5.53 *Search engines ignore self loops in calculating page rank. Thus, to increase page rank one needs to resort to loops of length two. By how much can you increase the page rank of a page by adding a number of loops of length two?*

Exercise 5.54 *Number the vertices of a graph* $\{1, 2, \ldots, n\}$. *Define hitting time to be the expected time from vertex 1. In (2) assume that the vertices in the cycle are sequentially numbered.*

1. *What is the hitting time for a vertex in a complete directed graph with self loops?*

2. *What is the hitting time for a vertex in a directed cycle with n vertices?*

Create exercise relating strongly connected and full rank
Full rank implies strongly connected.
Strongly connected does not necessarily imply full rank

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Is graph aperiodic iff $\lambda_1 > \lambda_2$?

Exercise 5.55 *Using a web browser bring up a web page and look at the source html. How would you extract the url's of all hyperlinks on the page if you were doing a crawl of the web? With Internet Explorer click on "source" under "view" to access the html representation of the web page. With Firefox click on "page source" under "view".*

Exercise 5.56 *Sketch an algorithm to crawl the World Wide Web. There is a time delay between the time you seek a page and the time you get it. Thus, you cannot wait until the page arrives before starting another fetch. There are conventions that must be obeyed if one were to actually do a search. Sites specify information as to how long or which files can be searched. Do not attempt an actual search without guidance from a knowledgeable person.*

6 Machine Learning

6.1 Introduction

Machine learning algorithms are general purpose tools that solve problems from many domains without detailed domain-specific knowledge. They have proven to be very effective in a large number of contexts, including computer vision, speech recognition, document classification, spam filtering, and many other areas.

A core problem underlying many machine learning applications that we will focus on in this chapter is the problem of learning a good classifier from labeled data. This problem consists of a domain of interest $\mathcal{X}$, called the *instance space*, such as email messages or patient records, and a classification task, such as classifying email messages into spam versus non-spam or determining which patients will respond well to a given medical treatment. To do this, our algorithm is given labeled *training examples*, items from the domain paired with their correct classification. For example, the training data could be a collection of email messages, each labeled as spam or not spam, or a collection of patients, each labeled by whether or not they responded well to the given medical treatment. We want our algorithm to use the training examples to produce a classification rule that performs well over new data. In this chapter we focus on *binary classification* where items in $\mathcal{X}$ are classified into two categories, as in the medical and spam-detection examples above.

The function that maps each item in $\mathcal{X}$ to the correct category for that item is called the *target function* which we denote by f^*. In other words, there is some function $f^* : \mathcal{X} \to \{-1, +1\}$ and our goal is to approximate f^* from labeled data. For this reason, the problem of learning a good classifier is often called the *function approximation* problem. We assume that our data items are represented using d *features* relevant to the classification we are trying to perform. These could be binary features such as "did this email message come from an email address the user has previously sent email to?" or real-valued features such as "what is the patient's LDL cholesterol level?" From now on we will associate $\mathcal{X}$ with R^d or $\{0, 1\}^d$. Additionally, our learning algorithm will be trying to approximate f^* by a classifier from some class $\mathcal{H}$ called the *hypothesis class*. $\mathcal{H}$ might be the set of all *linear separators*: functions $h_{\mathbf{w}}$ such that $h_{\mathbf{w}}(\mathbf{x}) = 1$ if $\mathbf{w}^T \cdot \mathbf{x} > 0$ and $h_{\mathbf{w}}(\mathbf{x}) = -1$ if $\mathbf{w}^T \cdot \mathbf{x} \leq 0$. [11]

We consider two standard models:

Batch learning: In this model there is a *probability distribution* $\mathcal{D}$ over the instance space $\mathcal{X}$. We are given a training sample S consisting of draws from $\mathcal{X}$ with probability according to the distribution $\mathcal{D}$ and our goal is to perform well on new data

[11] Technically, this is the class of *homogeneous* linear separators because the separating hyperplane passes through the origin. More generally, one could consider a nonzero threshold b. Note that by adding a "dummy feature" x_0 that is equal to 1 in every example, one can achieve the same classification with a homogeneous linear separator by setting $w_0 = -b$.

items also drawn from $\mathcal{X}$. For instance, in the case of determining which patients will respond well to a given medical treatment, $\mathcal{X}$ would correspond to the overall population of people who are candidates for that treatment, and our assumption would be that our training data is a random sample from this population. A nice feature of this model is there is a well-defined notion of the *true error rate* of some proposed classifier h, namely $\text{Prob}_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f^*(\mathbf{x})]$, the chance of making a mistake on a new example from $\mathcal{D}$. We use the notation $\text{Prob}_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f^*(\mathbf{x})]$ for the probability sum that $h(\mathbf{x}) \neq f^*(\mathbf{x})$.

Online learning: In this model, learning proceeds as a sequence of *trials*. In each trial, we are presented with some example $\mathbf{x} \in \mathcal{X}$, and asked to predict its true label $f^*(\mathbf{x})$. We are told if we made a mistake. For example, each morning you might look out the window and decide whether or not to bring an umbrella to school; at the end of the day, you know if you made the right decision. In this model, our goal is to bound the total number of *mistakes* made over time. Note that we might or might not learn anything new from a new example; e.g., if we see the same $\mathbf{x}$ over and over again, we won't learn anything new, but this is fine since we will not be making mistakes on it either. The key challenge is that when we make a mistake, we learn from it. Algorithms in this model need to explicitly be able to learn from their mistakes.

The following sections consider batch learning and then online learning. We will see a formal connection between learning and the notion of Occam's razor, and discuss a range of algorithms including the perceptron algorithm, stochastic gradient descent, kernel methods, and boosting. We will examine notions of regularization and confidence bounds, and see the important notion of VC-dimension for controlling overfitting.

6.2 Batch Learning, Occam's Razor, and Uniform Convergence

We now develop algorithms for the batch learning model mentioned above and discuss a formal connection to the philosophical notion of Occam's razor. In the batch model one is given a sample S of labeled training data; e.g., 100 email messages correctly labeled whether or not they are spam. These data points are assumed to be drawn from some probability distribution $\mathcal{D}$ over the instance space and our goal is to use this data to produce a classifier $h \in \mathcal{H}$ that performs well on new examples from $\mathcal{D}$. [12] In particular, we define the *true error rate* of a classifier h produced by the algorithm as $err_{\mathcal{D}}(h) = \text{Prob}_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f^*(\mathbf{x})]$, which we want to be low. There is also a natural notion of the training error rate of a proposed h, namely the fraction of data points in training data S on which h makes a mistake. We call this $err_S(h)$. A proposed rule h is said to *overfit* the training data if the training error, $err_S(h)$ is substantially less than the true error, $err_{\mathcal{D}}(h)$. As part of our analysis and algorithm development, we need to find ways to ensure that our algorithms are not fooled into producing a classifier

[12] In the case of email messages, this is not a good model because what a "typical" spam or non-spam email looks like may change over time.

with of high true error that only appear to be good because they overfit the training data.

How can we design good learning algorithms and also prevent overfitting? We begin with a simple problem of learning disjuctions.

6.2.1 Learning Disjunctions

Suppose that the instance space $\mathcal{X}$ equals $\{0,1\}^d$ and the target function f^* is a disjunction over features, such as $f^*(\mathbf{x}) = x_1 \vee x_4 \vee x_7 \vee x_8$. For example, in trying to predict whether an email message is spam or not, the features correspond to the presence or absence of different indicators of spam-ness. This corresponds to the belief that there is some subset of these indicators such that every spam email has at least one of them and every non-spam email has none of them. In what follows we (a) design an efficient algorithm that finds a disjunction consistent with our training sample S if one exists, and (b) argue for a large training sample S that it is highly unlikely such a rule overfits too much.

Simple Disjunction Learner: Given sample S, discard all features that occur in any negative example in S. Output the function h that is the disjunction of all remaining features.

Lemma 6.1 *The Simple Disjunction Learner produces a disjunction h that is consistent with the sample S; i.e., with sample error, $err_S(h)$, equal to zero whenever such a disjunction exists.*

Proof: Suppose there exists a disjunction f with $err_S(f) = 0$. Then for any x_i in f, x_i will not occur in any negative example. Therefore, h will contain x_i as well, i.e., $\{x_i | x_i \text{ in } h\} \supseteq \{x_i | x_i \text{ in } f\}$. This means that h will be correct on all positive examples in S since each one must have some $x_i \in f$ set to one. Furthermore, h will be correct on all negative examples in S since by design all features set to one in any negative example were discarded. Therefore, h is correct on all examples in S. ■

We now argue that so long as S is sufficiently large, the hypothesis h produced by the above algorithm is unlikely to have overfit by much. We will show this by arguing that it is unlikely there will be any disjunction of high true error consistent with the training data.

Lemma 6.2 *Let $\epsilon, \delta > 0$. If a training sample S is drawn from $\mathcal{X}$ of size*

$$|S| \geq \frac{1}{\epsilon}[d\ln(2) + \ln(1/\delta)],$$

then with probability greater than or equal to $1-\delta$, every disjunction h with $err_{\mathcal{D}}(h) \geq \epsilon$ has $err_S(h) > 0$. Equivalently, every disjunction h with $err_S(h) = 0$ has $err_{\mathcal{D}}(h) < \epsilon$.

Proof: Let $h_1, h_2, \ldots, h_n$ be the set of all disjunctions with $err_{\mathcal{D}}(h) \geq \epsilon$; i.e., these are the bad disjunctions that we don't want to output. Consider now drawing the sample S and let A_i be the event that h_i is consistent with S. Since every example in S is drawn from the instance space $\mathcal{X}$, for any given i we have:

$$\text{Prob}[A_i] \;\leq\; (1-\epsilon)^{|S|},$$

by the fact that h_i has true error rate at least ϵ. If we fix h_i and then draw our sample S, the chance that h_i makes no mistakes on S is at most the probability that a coin of bias ϵ comes up tails $|S|$ times in a row, which is $(1-\epsilon)^{|S|}$. Therefore, by the union bound and the fact that $n \leq 2^d$ (since there are only 2^d disjunctions in total):

$$\text{Prob}[\cup_i A_i] \;\leq\; 2^d(1-\epsilon)^{|S|}.$$

Finally, using the fact that $(1-\epsilon)^{1/\epsilon} \leq 1/e$, the probability that any disjunction h with $err_{\mathcal{D}}(h) \geq \epsilon$ has $err_S(h) = 0$ is at most $2^d e^{-\epsilon|S|}$. Plugging in the sample size bound from the lemma, this is at most $2^d e^{-d\ln(2)-\ln(1/\delta)} = \delta$ as desired. ∎

Together, Lemma 6.1 and Lemma 6.2 give us the following theorem:

Theorem 6.3 *If the target function f^* is a disjunction, then given a training sample S of size at least $\frac{1}{\epsilon}[d\ln(2) + \ln(1/\delta)]$, with probability greater than or equal to $1-\delta$, the simple disjunction learner returns a classifier h with $err_{\mathcal{D}}(h) < \epsilon$.*

What we have just shown is sometimes called a "PAC-learning guarantee" since we have argued that the rule produced by our algorithm is probably approximately correct.

6.2.2 Occam's razor

Occam's razor is the notion, stated by William of Occam around AD 1320, that in general one should prefer simpler explanations over more complicated ones.[13] Why should one do this, and can we make a formal claim about why this is a good idea? What if each of us disagrees about precisely which explanations are simpler than others? It turns out we will be able to build on the analysis in the proof of Lemma 6.2 to make a general and compelling mathematical statement of Occam's razor that addresses these issues.

First, the reader might notice that in the proof of Lemma 6.2, the only place we used the fact that h was a disjunction was in arguing that $N \leq 2^d$. We could just have easiliy used any hypothesis class $\mathcal{H}$, replacing the term "$d\ln(2)$" with $\ln(|\mathcal{H}|)$. In particular, we have:

Theorem 6.4 *Fix any hypothesis class $\mathcal{H}$ and let $\epsilon, \delta > 0$. If a training sample S is drawn from $\mathcal{D}$ of size*

$$|S| \geq \frac{1}{\epsilon}[\ln(|\mathcal{H}|) + \ln(1/\delta)],$$

[13] The statement more explicitly was that "Entities should not be multiplied unnecessarily."

then with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) \geq \epsilon$ has $err_S(h) > 0$. Equivalently, with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ with $err_S(h) = 0$ has

$$err_{\mathcal{D}}(h) < \frac{\ln(|\mathcal{H}|) + \ln(1/\delta)}{|S|}.$$

Now, what do we mean by a rule being "simple"? Let's assume that each of us has some way of describing rules, using bits (since we are computer scientists). The methods, also called *description languages*, used by each of us may be different, but one fact we can say for certain is that in any given description language, there are at most 2^b rules that can be described using fewer than b bits (because $1 + 2 + 4 + \ldots + 2^{b-1} < 2^b$). Therefore, by setting $\mathcal{H}$ to be the set of all rules that can be described in fewer than b bits and plugging into Theorem 6.4, we have the following:

Theorem 6.5 (Occam's razor) *Fix any description language, and consider a training sample S drawn from distribution $\mathcal{D}$. With probability $\geq 1-\delta$, any rule consistent with S that can be described in this language using fewer than b bits will have $err_X(h) \leq \epsilon$ for*

$$|S| = \frac{1}{\epsilon}[b\ln(2) + \ln(1/\delta)], \quad \textit{or equivalently} \quad err_X(h) \leq \frac{b\ln(2) + \ln(1/\delta)}{|S|}.$$

For example, using the fact that $\ln(2) < 1$ and ignoring the low-order $\ln(1/\delta)$ term, this means that if the number of bits it takes to write down our rule is at most 10% of the number of data points in our sample, then we can be confident it will have error at most 10% with respect to $\mathcal{D}$. What is perhaps surprising about this theorem is that it means that we can each have different ways of describing rules and yet all use Occam's razor. Note that the theorem does *not* say that complicated rules are necessarily bad, only that simple rules are unlikely to fool us since there are just not that many simple rules.

6.2.3 Application: Learning Decision Trees

One popular practical method for machine learning is to learn a *decision tree*; see Figure **??**. While finding the smallest decision tree that fits a given training sample S is NP-hard, there are a number of heuristics that are used in practice. One popular heuristic, called ID3, selects the feature to put inside any given node v by choosing the feature of largest *information gain*, a measure of how much it is directly improving prediction.[14] This then continues until all leaves are pure—they have only positive or only negative examples. Suppose we run ID3 on a training set S and it outputs a tree with n nodes. Such a tree can be described using $O(n \log d)$ bits: $\log_2(d)$ bits to give the index of the feature in the root, $O(1)$ bits to indicate for each child if it is a leaf and if so what label it should

[14] Formally, using S_v to denote the set of examples in S that reach node v, and supposing that feature x_i partitions S_v into S_v^0 and S_v^1 (the examples in S_v with $x_i = 0$ and $x_i = 1$, respectively), the information gain of x_i is defined as: $Ent(S_v) - [\frac{|S_v^0|}{|S_v|}Ent(S_v^0) + \frac{|S_v^1|}{|S_v|}Ent(S_v^1)]$. Here, $Ent(S')$ is the binary entropy of the label proportions in set S'; that is, if a p fraction of the examples in S' are positive, then $Ent(S') = p\log_2(1/p) + (1-p)\log_2(1/(1-p))$, defining $0\log_2(0) = 0$.

have, and then $O(n_L \log d)$ and $O(n_R \log d)$ bits respectively to describe the left and right subtrees, where n_L is the number of nodes in the left subtree and n_R is the number of nodes in the right subtree. So, by Theorem 6.5, we can be confident the true error is low if we can produce a consistent tree with much fewer than $\epsilon|S|/\log(d)$ nodes.

6.2.4 Agnostic Learning and Uniform Convergence

Our analysis so far has addressed the case that we are able to find a hypothesis with zero error on the training set S. But what if the best $h \in \mathcal{H}$ we can find has, say, 5% error on S? Can we still be confident that its true error under $\mathcal{D}$ is low, say at most 10%? For this, we want an analog of Theorem 6.4 that says that with high probability, *every* $h \in \mathcal{H}$ has $err_S(h)$ within $\pm\epsilon$ of $err_X(h)$. Such a statement is called *uniform convergence* because we are asking that empirical errors converge to their true errors uniformly over all functions in $\mathcal{H}$. These are also often called "agnostic learning" bounds because they are most relevant when we do not necessarily believe that f^* lies in $\mathcal{H}$, we simply want to perform as well as we can using the hypothesis class $\mathcal{H}$ at hand.[15]

To prove uniform convergence bounds, we will need to use a tail inequality for sums of independent Bernoulli random variables (i.e., coin tosses). The following is particularly convenient and is a variation on the Chernoff bounds in Section 12.4.11 of the appendix.

Theorem 6.6 (Hoeffding bounds) *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables with $p = \Pr[X_i = 1]$, and let $s = \sum_i X_i$ (equivalently, flip n coins of bias p and let s be the total number of heads). Then for any $0 \le \alpha \le 1$,*

$$\begin{aligned} \Pr[s/n > p + \alpha] &\le e^{-2n\alpha^2} \\ \Pr[s/n < p - \alpha] &\le e^{-2n\alpha^2}. \end{aligned}$$

Using Theorem 6.6 we immediately have the following uniform convergence analog of Theorem 6.4.

Theorem 6.7 (Uniform convergence) *Fix any hypothesis class $\mathcal{H}$ and let $\epsilon, \delta > 0$. With probability $\ge 1 - \delta$, if a training sample S of size*

$$|S| \ge \frac{1}{2\epsilon^2}[\ln(|\mathcal{H}|) + \ln(2/\delta)]$$

is drawn from $\mathcal{D}$, then $|err_S(h) - err_X(h)| \le \epsilon$ for all $h \in \mathcal{H}$.

Proof: Fix some $h \in \mathcal{H}$ and let X_i be the indicator random variable for the event that h makes a mistake on the ith example in S. These are independent $\{0,1\}$ random variables with $\Pr[X_i = 1] = err_X(h)$, and the fraction of the X_i's equal to 1 is exactly the empirical

[15] It could also be that the features we are measuring are simply insufficient to make a perfect prediction. E.g., in general from observable features it may be impossible to perfectly predict whether a patient will respond well to a given medical treatment.

error of h. T herefore, Hoeffding bounds guarantee that $\Pr[|err_S(h) - err_X(h)| \geq \epsilon] \leq 2e^{-2|S|\epsilon^2}$. Applying the union bound to all $h \in \mathcal{H}$ we have

$$\Pr[\exists h \in \mathcal{H} : |err_S(h) - err_X(h)| \leq \epsilon] \quad \leq \quad 2|\mathcal{H}|e^{-2|S|\epsilon^2}$$

Plugging in the value of $|S|$ from the theorem statement we find that the right-hand-side above is at most δ as desired. ■

Theorem 6.7 justifies the approach of optimizing over our training sample S even if we are not able to find a rule of zero empirical error. If our training set S is sufficiently large, we are guaranteed that with high probability, good performance on S will translate to good performance under $\mathcal{D}$.

6.2.5 Regularization

If we take Theorem 6.7 and rewrite the guarantee to solve for ϵ in terms of $|S|$ and $|\mathcal{H}|$, we get that with probability $\geq 1 - \delta$, all $h \in \mathcal{H}$ satisfy

$$err_X(h) \quad \leq \quad err_S(h) + \sqrt{\frac{\ln(|\mathcal{H}|) + \ln(2/\delta)}{2|S|}}.$$

The second term on the right-hand-side above is a bound on the maximum amount of overfitting that occurs over all functions in $\mathcal{H}$, with high probability. Here, the "$\ln(|\mathcal{H}|)$" term can be viewed as a measure of complexity of this class of functions. Now, suppose we want to optimize over functions of multiple levels of complexity. To address this, consider fixing some description language and let $\mathcal{H}_i$ denote those functions that can be described in i bits in this language, so $|\mathcal{H}_i| \leq 2^i$. Let $\delta_i = \delta/2^i$. We know that with probability at least $1 - \delta_i$, all $h \in \mathcal{H}_i$ satisfy $err_X(h) \leq err_S(h) + \sqrt{\frac{\ln(|\mathcal{H}_i|)+\ln(2/\delta_i)}{2|S|}}$. Now, applying the union bound over all i, using the fact that $\delta_1 + \delta_2 + \delta_3 + \ldots = \delta$, and also the fact that $\ln(|\mathcal{H}_i|) + \ln(2/\delta_i) \leq i\ln(4) + \ln(2/\delta)$, we get the following corollary.

Corollary 6.8 *Fix any description language, and consider a training sample S drawn from distribution $\mathcal{D}$. With probability $\geq 1 - \delta$, all hypotheses h satisfy*

$$err_X(h) \quad \leq \quad err_S(h) + \sqrt{\frac{\text{size}(h)\ln(4) + \ln(2/\delta)}{2|S|}}$$

where size(h) *denotes the number of bits needed to describe h in the given language.*

Corollary 6.8 has an interesting implication. It tells us that rather than searching for a rule of low training error, we instead may want to search for a rule with a low right-hand-side in the displayed formula. If we can find one for which this quantity is small, we can be confident true error will be low as well. This is the idea of *regularization*. We add onto the training error a term called a *regularizer* that penalizes complex functions, and then search for a rule with a low sum of training error plus regularizer. Later, we will see in Support Vector Machines the use of regularizers such that if we replace training error with an upper bound called "hinge loss," we can efficiently optimize the sum via convex optimization.

6.3 The Perceptron Algorithm and Stochastic Gradient Descent

We now describe a classic algorithm for learning linear separators called the Perceptron algorithm, and a widely-used generalization of this algorithm known as Stochastic Gradient Descent.

6.3.1 The Perceptron Algorithm

The *Perceptron algorithm* is a classic algorithm for learning a linear separator [**?**, **?**, **?**]. We will describe it here as an algorithm in the batch learning model, though it is also naturally viewed as an online algorithm, and we will examine its properties in the online model in Section 6.4. The Perceptron algorithm learns a homogeneous linear separator, i.e., a linear separator that passes through the origin, and we will assume that all data points have been pre-scaled to lie in the unit ball. Note that scaling the length of the examples does not affect which side of the separator they are on.

The Perceptron Algorithm:

Given: training sample S of points in R^d with labels $f^*(\mathbf{x}) \in \{-1, 1\}$. Assume $|\mathbf{x}| \leq 1$ for all $\mathbf{x} \in S$.

Goal: produce a weight vector $\mathbf{w}$ such that $h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$ is correct for all $\mathbf{x} \in S$.

1. Start with the all-zeroes weight vector $\mathbf{w} = \mathbf{0}$.
2. For each $\mathbf{x} \in S$, if $\text{sgn}(\mathbf{w} \cdot x) \neq f^*(\mathbf{x})$, update: $\mathbf{w} \leftarrow \mathbf{w} + f^*(\mathbf{x})\mathbf{x}$.
3. Repeat Step 2 until $\text{sgn}(\mathbf{w} \cdot x) = f^*(\mathbf{x})$ for all $\mathbf{x} \in S$.

Assume that our data set S is indeed linearly separable: namely, there exists a weight vector $\mathbf{w}^*$ such that $f^*(\mathbf{x}) = \text{sgn}(\mathbf{w}^* \cdot \mathbf{x})$ for all $\mathbf{x} \in S$. Without loss of generality, assume $\mathbf{w}^*$ is a unit-length vector; this does not affect the sign of the dot products. Define the *margin* of separation

$$\gamma = \min_{\mathbf{x} \in S} |\mathbf{w}^* \cdot \mathbf{x}|.$$

This is the minimum distance between any data point and the hyperplane $\mathbf{w}^* \cdot \mathbf{x} = 0$ (see Figure 6.1). We will show that the Perceptron algorithm finds a consistent weight vector $\mathbf{w}$ after at most $1/\gamma^2$ updates. In particular, this will be true when $\mathbf{w}^*$ is defined to be the maximum margin separator: the separator of maximum γ.

Theorem 6.9 *The Perceptron algorithm finds a consistent weight vector after at most $1/\gamma^2$ updates.*

Proof: We will examine two quantities: $\mathbf{w} \cdot \mathbf{w}^*$ and $|\mathbf{w}|^2$.

The first claim is that after each update, $\mathbf{w} \cdot \mathbf{w}^*$ increases by at least γ. That is because, by definition of γ, if we update by adding a positive example $\mathbf{x}$, then we have

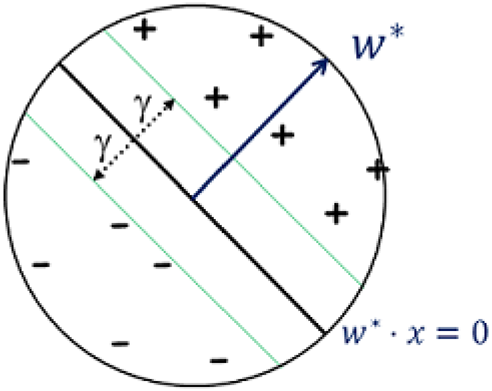


Figure 6.1: Positive and negative examples separated by margin γ. The vector w^* is orthogonal to the hyperplane $w^* \cdot x = 0$.

$(\mathbf{w} + \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w} \cdot \mathbf{w}^* + \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w} \cdot \mathbf{w}^* + \gamma$. Similarly, if we update by subtracting a negative example, then we have $(\mathbf{w} - \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w} \cdot \mathbf{w}^* - \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w} \cdot \mathbf{w}^* + \gamma$.

The second claim is that after each update, $|\mathbf{w}|^2$ increases by at most 1. This uses the fact that we only update on examples for which we made a mistake. Specifically, if we update by adding a positive example $\mathbf{x}$, then we have

$$(\mathbf{w} + \mathbf{x}) \cdot (\mathbf{w} + \mathbf{x}) = |\mathbf{w}|^2 + 2\mathbf{w} \cdot \mathbf{x} + |\mathbf{x}|^2 \leq |\mathbf{w}_t|^2 + 1.$$

The last inequality above uses the fact that (a) $\mathbf{w} \cdot \mathbf{x} \leq 0$ since we only update when $\text{sgn}(\mathbf{w} \cdot \mathbf{x}) \neq f^*(\mathbf{x})$, and (b) $|\mathbf{x}| \leq 1$. The exact same thing holds (flipping signs) when we update on a negative $\mathbf{x}$, namely $(\mathbf{w} - \mathbf{x}) \cdot (\mathbf{w} - \mathbf{x}) = |\mathbf{w}|^2 - 2\mathbf{w} \cdot \mathbf{x} + |\mathbf{x}|^2 \leq |\mathbf{w}_t|^2 + 1$.

Putting the above two claims together, after T updates we have $\mathbf{w} \cdot \mathbf{w}^* \geq \gamma T$ and $|\mathbf{w}|^2 \leq T$. Using the fact that $|\mathbf{w}| \geq \mathbf{w} \cdot \mathbf{w}^*$ (since $\mathbf{w}^*$ is a unit-length vector), we have:

$$\sqrt{T} \geq |\mathbf{w}| \geq \mathbf{w} \cdot \mathbf{w}^* \geq \gamma T.$$

This implies that $T \leq 1/\gamma^2$ as desired. ∎

What can we say about performance on new data? Using Theorem 6.5, one statement we can make is that if we run the Perceptron algorithm on a machine that stores each weight as a 64-bit floating-point number, then if S has size at least $\frac{1}{\epsilon}[64d\ln(2)+\ln(1/\delta)]$, we can be confident that any rule $h_\mathbf{w}$ produced with $err_S(h_\mathbf{w}) = 0$ will have $err_\mathcal{D}(h_\mathbf{w}) \leq \epsilon$. In Section 6.4.4 (Online to Batch Conversion) we will see a different generalization guarantee we can give in terms of γ rather than d, and in Section 6.9 (VC-dimension) we will see a bound in terms of d that does not depend on the bit-precision of our machine.

6.3.2 Stochastic Gradient Descent

We now describe a very practical and widely-used algorithm in machine learning, called *stochastic gradient descent* (SGD). The Perceptron algorithm we examined in Section 6.3.1 can be viewed as a special case of this algorithm, as can methods for deep learning.

For stochastic gradient descent, we assume our hypotheses can be described using a vector of weights. That is, $\mathcal{H} = \{h_{\mathbf{w}}\}$, where $h_{\mathbf{w}}$ is parametrized by a weight vector $\mathbf{w} \in R^m$, and typically $m \geq d$. The function $h_{\mathbf{w}}$ is of the form $h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(f_{\mathbf{w}}(\mathbf{x}))$, where $f_{\mathbf{w}} : R^d \to R$ is a real-valued function in an auxiliary class $\mathcal{F}$. So, $f_{\mathbf{w}}$ is actually doing all the work, and $h_{\mathbf{w}}$ is just converting its real-valued prediction to "positive" or "negative". To apply stochastic gradient descent, we also need a *loss function* $L(f_{\mathbf{w}}(\mathbf{x}), f^*(\mathbf{x}))$ that describes the real-valued penalty we will associate with function $f_{\mathbf{w}}$ for its prediction on an example $\mathbf{x}$ whose true label is $f^*(\mathbf{x})$. The algorithm is then the following:

Stochastic Gradient Descent:

Given: starting point $\mathbf{w} = \mathbf{w}_{init}$ and learning rates $\lambda_1, \lambda_2, \lambda_3, \ldots$
(e.g., $\mathbf{w}_{init} = \mathbf{0}$ and $\lambda_t = 1$ for all t, or $\lambda_t = 1/\sqrt{t}$).

Consider a sequence of random examples[16] $(\mathbf{x}_1, f^*(\mathbf{x}_1)), (\mathbf{x}_2, f^*(\mathbf{x}_2)), \ldots$.

1. Given example $(\mathbf{x}_t, f^*(\mathbf{x}_t))$, compute the gradient $\nabla L(f_{\mathbf{w}}(\mathbf{x}_t), f^*(\mathbf{x}_t))$ of the loss of $f_{\mathbf{w}}(\mathbf{x}_t)$ with respect to the weights $\mathbf{w}$. This is a vector in R^m whose ith component is $\frac{\partial L(f_{\mathbf{w}}(\mathbf{x}_t), f^*(\mathbf{x}_t))}{\partial w_i}$.
2. Update: $\mathbf{w} \leftarrow \mathbf{w} - \lambda_t \nabla L(f_{\mathbf{w}}(\mathbf{x}_t), f^*(\mathbf{x}_t))$.

Let's now try to understand the algorithm better by seeing a few examples of instantiating the class of functions $\mathcal{F}$ and loss function L.

First, consider $m = d$ and $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, so $\mathcal{F}$ is the class of linear predictors. Let's also assume we have pre-scaled each example to lie inside the unit ball. Consider the loss function $L(f_{\mathbf{w}}(\mathbf{x}), f^*(\mathbf{x})) = \max(0, -f^*(\mathbf{x}) f_{\mathbf{w}}(\mathbf{x}))$, and recall that $f^*(\mathbf{x}) \in \{-1, 1\}$. In other words, if $f_{\mathbf{w}}(\mathbf{x})$ has the correct sign, then we have a loss of 0, otherwise we have a loss equal to the magnitude of $f_{\mathbf{w}}(\mathbf{x})$. In this case, if $f_{\mathbf{w}}(\mathbf{x})$ has the correct sign and is non-zero, then the gradient will be zero since an infinitesimal change in any of the weights will not change the sign. So, when $h_{\mathbf{w}}(\mathbf{x})$ is correct, the algorithm will leave $\mathbf{w}$ alone. On the other hand, if $f_{\mathbf{w}}(\mathbf{x})$ has the wrong sign, then $\frac{\partial L}{\partial w_i} = -f^*(\mathbf{x}) \frac{\partial \mathbf{w} \cdot \mathbf{x}}{\partial w_i} = -f^*(\mathbf{x}) x_i$. So, using $\lambda_t = 1$, the algorithm will update $w \leftarrow w + f^*(\mathbf{x})\mathbf{x}$. Note that this is exactly the Perceptron algorithm. (Technically we must address the case that $f_{\mathbf{w}}(\mathbf{x}) = 0$; in this case, we should view $f_{\mathbf{w}}$ as having the wrong sign just barely.)

[16] In practice we will be cycling through the training set S. Bottou [**?**] recommends randomly shuffling S at the start just in case (in practice) it was not actually an i.i.d random sample in the first place, along with other useful practical recommendations.

As a small modification to the above example, consider the same class of linear predictors $\mathcal{F}$ but now slightly modify the loss function to $L(f_{\mathbf{w}}(\mathbf{x}), f^*(\mathbf{x})) = \max(0, 1 - f^*(\mathbf{x})f_{\mathbf{w}}(\mathbf{x}))$. This loss function, called *hinge loss*, now requires $f_{\mathbf{w}}(\mathbf{x})$ to have the correct sign *and* magnitude at least 1 in order to be zero. Hinge loss has the useful property that it is an upper bound on error rate: for any sample S, $err_S(h_{\mathbf{w}}) \leq \sum_{\mathbf{x} \in S} L(f_{\mathbf{w}}(\mathbf{x}), f^*(\mathbf{x}))$. If we instantiate SGD with this $(\mathcal{F}, L)$ we get a method called the *margin perceptron* algorithm.

More generally, we could have a much more complex class $\mathcal{F}$. For example, consider a layered circuit of "soft threshold" gates: each node in the circuit computes a linear function of its inputs and then passes this value through an "activation function" such as $a(z) = \tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$. This circuit could have multiple layers with the output of layer i being used as the input to layer $i + 1$. The vector $\mathbf{w}$ would be the concatenation of all the weight vectors in the network, so we might have $m \gg d$. This is the idea of *deep neural networks* discussed further in Section 6.10.4.

While it is difficult to give general guarantees on when SGD will succeed in finding a hypothesis of low error on its training set S, we can use Theorems 6.5 and 6.7 to argue that if it does so and if S is sufficiently large, we can be confident that its true error will be low as well. Specifically, suppose that SGD is run on a machine where each weight is stored as a 64-bit floating point number. This means that its hypothesis can each be described using $64m$ bits. So, applying Theorem 6.5, if S has size at least $\frac{1}{\epsilon}[64m\ln(2) + \ln(1/\delta)]$, it is unlikely any hypothesis in $\mathcal{H}$ of true error greater than ϵ will be consistent with the sample, and so if it finds a hypothesis consistent with S, we can be confident its true error is at most ϵ. Or, using Theorem 6.7, if $|S| \geq \frac{1}{2\epsilon^2}[64m\ln(2) + \ln(2/\delta)]$ then we can be confident that the final hypothesis h produced by SGD satisfies $err_{\mathcal{D}}(h) \leq err_S(h) + \epsilon$.

6.4 Online learning

We now switch to the online learning model. Recall that in this model, learning proceeds in a sequence of trials: in each trial we are given an example x, asked to predict its label $f^*(x)$, and then are told whether we were correct or not. Rather than assuming that data necessarily arrives from some probability distribution and aiming to find a function of low "true error", we will simply aim to minimize the number of mistakes made over time. In particular, we will aim for theoretical guarantees that apply to any sequence of examples. We begin by revisiting the disjunction-learning problem discussed in Section 6.2.1 but now in the online model, examine the Perceptron algorithm discussed in Section 6.3.1 from this perspective, and then give a general online-to-batch conversion procedure that converts any algorithm with a good mistake bound in the online model to one with strong guarantees in the batch (distributional) model.

6.4.1 Online learning of disjunctions

Recall the setting of Section 6.2.1: our instance space $\mathcal{X} = \{0,1\}^d$ and we are told the target function f^* can be represented as an OR-function over features. For this problem, we have the following simple algorithm that guarantees never to make more than d mistakes.

Simple Online Disjunction Learner:

Initialize $h = x_1 \vee x_2 \vee \ldots \vee x_d$.

1. Given example x, predict $h(x)$.
2. If $h(x)$ predicted positive but $f^*(x)$ is negative, remove from h any variable set to 1 in x.

Theorem 6.10 *The Simple Online Disjunction Learner makes at most d mistakes whenever the target function f^* is a disjunction.*

Proof: We first claim that the algorithm maintains the invariant that $\{x_i \in h\} \supseteq \{x_i \in f^*\}$. The reason is that (a) this holds at the start by the initialization step, and (b) variables are only removed from h when they are set to 1 in negative examples, which means they cannot be in f^*. Next, observe that the invariant guarantees that the algorithm will never make a mistake on a positive example, since if x satisfies f^*, it must satisfy h as well. Finally, each mistake on a negative example removes at least one variable from h. Since we begin with d variables in h and the total number of variable can never be negative, this means the total number of mistakes is at most d. ∎

We complement Theorem 6.10 with a matching lower bound. For this lower bound, assume that a disjunction of size 0 is allowed and corresponds to the "all false" function.

Theorem 6.11 *For any deterministic algorithm A there exists a sequence of examples σ and disjunction f^* such that A makes d mistakes on sequence σ labeled by f^*.*

Proof: Let σ be the sequence $e_1, e_2, \ldots, e_d$ where e_j is the example that is zero everywhere except for a 1 in the jth position. Imagine running A on sequence σ and telling A it made a mistake on every example; that is, if A predicts positive on e_j we set $f^*(e_j) = -1$ and if A predicts negative on e_j we set $f^*(e_j) = +1$. This target corresponds to the disjunction of all x_j such that A predicted negative on e_j, so it is a legal disjunction. Since A is deterministic, the fact that we constructed f^* by running A is not a problem: it would make the same mistakes if re-run from scratch on the same sequence and same target. Therefore, A makes d mistakes on this σ and f^*. ∎

6.4.2 Halving and Occam's razor

If we are not concerned with running time, a simple algorithm that guarantees to make at most $\log_2(|\mathcal{H}|)$ mistakes for a target belonging to any given class $\mathcal{H}$ is called the *halving algorithm.* This algorithm simply maintains the *version space* $\mathcal{H}' \subseteq \mathcal{H}$ consisting of all $h \in \mathcal{H}$ consistent with the labels on every example seen so far, and predicts based on majority vote. Each mistake is guaranteed to reduce the size of $\mathcal{H}'$ by at least half (hence the name).

Suppose that we are interested in a class such as decision trees where some hypotheses are more complicated than others. Specifically, assume we have some description language that allows us to describe each hypothesis using bits, and let us assume this description language is *prefix free*, meaning that no hypothesis has a description that is a prefix of the description of any other hypothesis. E.g., if we think of binary a tree where a 0 means "branch left" and a 1 means "branch right" then each hypothesis can be placed as a leaf of this tree, at depth equal to the number of bits in its description. Now, suppose we give each hypothesis h a weight $w(h) = 1/2^{\mathrm{size}(h)}$, where $\mathrm{size}(h)$ is the number of bits in its description. Then we have $\sum_h w(h) \leq 1$.[17] This means that if we modify the halving algorithm so that it predicts based on a *weighted* majority vote over its version space, then since the total sum of weights is initially at most 1 and can never be less than $w(f^*)$, the total number of mistakes is at most $\mathrm{size}(f^*)$. This is called the *generalized halving algorithm.* Thus we have the following analog to Theorem 6.5.

Theorem 6.12 *Given any prefix-free description language, generalized halving makes at most* $\mathrm{size}(f^*)$ *mistakes on any sequence of examples consistent with* f^*.

6.4.3 The Perceptron Algorithm

We can restate the Perceptron algorithm discussed earlier as a learning algorithm in the online model. Assume that all examples have been pre-scaled to lie inside the unit ball.

The Perceptron Algorithm:

1. Start with the all-zeroes weight vector $\mathbf{w} = \mathbf{0}$.
2. Given example $\mathbf{x}$, predict $\mathrm{sgn}(\mathbf{w} \cdot x)$.
3. If the prediction on $\mathbf{x}$ was incorrect, update: $\mathbf{w} \leftarrow \mathbf{w} + f^*(\mathbf{x})\mathbf{x}$.

We can restate Theorem 6.9 in the online model as follows. Say that a sequence of examples is consistent with a separator of margin γ if there exists a unit-length vector $\mathbf{w}^*$ such that $f^*(\mathbf{x})(\mathbf{w}^* \cdot \mathbf{x}) \geq \gamma$ for all examples $\mathbf{x}$ in the sequence.

[17] This can be seen by imagining placing 1 ounce of gold dust at the root of the tree, then splitting this dust equally between its children, and continuing the process down to the leaves; a leaf at depth d will receive exactly $1/2^d$ ounces of the gold dust.

Theorem 6.13 *The Perceptron algorithm makes at most $1/\gamma^2$ updates on any sequence of examples consistent with a separator of margin γ.*

6.4.4 Online to Batch Conversion

Suppose we have an online algorithm with a good mistake bound. Can we use it to get a guarantee in the distributional (batch) learning setting? Intuitively, the answer should be yes since the online setting is only harder. Indeed, this intuition is correct. We present here two natural approaches for such online to batch conversion.

Conversion procedure 1: Random Stopping. Suppose we have an online algorithm $\mathcal{A}$ with mistake-bound M. Say we run the algorithm in a single pass on a sample S of size M/ϵ. Let X_i be the indicator random variable for the event that $\mathcal{A}$ makes a mistake on the ith example. Since $\sum_{i=1}^{|S|} X_i \leq M$ for *any* set S, we certainly have that $\mathbf{E}[\sum_{i=1}^{|S|} X_i] \leq M$ where the expectation is taken over the random draw of S from $\mathcal{D}^{|S|}$. By linearity of expectation, and dividing both sides by $|S| = M/\epsilon$ we therefore have:

$$\frac{1}{|S|}\sum_{i=1}^{|S|}\mathbf{E}[X_i] \;\; \leq \;\; \epsilon. \tag{6.1}$$

Let h_i denote the hypothesis used by algorithm $\mathcal{A}$ to predict on the ith example. Since the ith example was randomly drawn from $\mathcal{D}$, we have $\mathbf{E}[err_{\mathcal{D}}(h_i)] = \mathbf{E}[X_i]$. This means that if we choose i at random from 1 to $|S|$ (i.e., stop the algorithm at a random time), the expected error of the resulting prediction rule, taken over the randomness in the draw of S and the choice of i, is at most ϵ as given by equation (6.1). Thus we have:

Theorem 6.14 (Online to Batch via Random Stopping) *If an online algorithm $\mathcal{A}$ with mistake-bound M is run on a sample S of size M/ϵ and stopped at a random time between* 1 *and $|S|$, the expected error of the hypothesis h produced satisfies* $\mathbf{E}[err_{\mathcal{D}}(h)] \leq \epsilon$.

Conversion procedure 2: Large Gap. A second natural approach to using an online learning algorithm $\mathcal{A}$ in the distributional setting is to just run it on random examples until a sufficiently large gap between consecutive mistakes is observed. For simplicity, assume for now that algorithm $\mathcal{A}$ is "conservative," meaning that it does not change its hypothesis when it predicts correctly (e.g., the Perceptron and disjunction algorithms are all conservative). Consider running $\mathcal{A}$ on a series of random examples from $\mathcal{D}$, and let e_i denote the error rate of the hypothesis used between the i^{th} and $(i+1)^{st}$ mistake. If $e_i > \epsilon$ then the chance that this hypothesis predicts correctly for $\frac{1}{\epsilon}\log(\frac{1}{\delta_i})$ examples in a row is at most

$$(1-\epsilon)^{\frac{1}{\epsilon}\log(\frac{1}{\delta_i})} \leq \delta_i.$$

Let $\delta_i = \delta/(i+2)^2$ so we have $\sum_{i=0}^{\infty}\delta_i = (\frac{\pi^2}{6} - 1)\delta \leq \delta$. Applying the union bound over all i, we can halt with confidence whenever $\mathcal{A}$ predicts correctly for so many examples in a row:

Theorem 6.15 (Online to Batch via Large Gap) *Let $\mathcal{A}$ be a conservative online learning algorithm, and suppose we run $\mathcal{A}$ until it has predicted correctly for $\frac{1}{\epsilon}\log(\frac{1}{\delta_i})$ examples in a row after its ith mistake. Then with probability at least $1-\delta$, this procedure will either run forever or halt with a hypothesis of error at most ϵ. Moreover, if $\mathcal{A}$ has a mistake bound of M, then this procedure will halt after $O(\frac{M}{\epsilon}\log(\frac{M}{\delta}))$ examples.*

Proof: The first claim follows from the discussion above. The second claim follows from the fact that $\sum_{i=0}^{M}\frac{1}{\epsilon}\log(1/\delta_i) \leq \sum_{i=0}^{M}\frac{1}{\epsilon}\log((M+2)^2/\delta) = O((\frac{M}{\epsilon}\log(\frac{M}{\delta}))$. ■

We can remove the assumption that $\mathcal{A}$ is conservative by referring not to the error of the final hypothesis but instead to the average error of hypotheses used since its last mistake, and using the fact that $(1-\epsilon_1)(1-\epsilon_2) \leq (1-\frac{\epsilon_1+\epsilon_2}{2})^2$. In other words, algorithm $\mathcal{A}$ cannot increase its chance of satisfying our stopping criteria with average error greater than ϵ by varying the error rates of the hypotheses it is using. This has a nice implication for Stochastic Gradient Descent, which in general is not conservative. If we run SGD until the gap between consecutive mistakes is sufficiently large, we can be confident that the average error of hypotheses produced in that gap is low.

6.5 Margins, Hinge-loss, Support-Vector Machines, and Perceptron revisited

So far we have considered the problem of learning a linear separator when there is a perfect separator to be found. Often, however, data will only be "mostly" linearly separable. Unfortunately, finding the separator that makes the fewest mistakes on a given dataset S is NP-complete. However, one task we *can* solve efficiently is to find a separator of minimum *hinge loss*. Specifically, for each example $\mathbf{x}_i$ we define a slack variable $\xi_i \geq 0$ and then solve the linear program:

$$\begin{array}{rl} \text{minimize} & \displaystyle\sum_i \xi_i \\ \text{subject to} & \mathbf{w}\cdot\mathbf{x}_i \geq 1-\xi_i \text{ for all positive examples } \mathbf{x}_i \\ & \mathbf{w}\cdot\mathbf{x}_i \leq -1+\xi_i \text{ for all negative examples } \mathbf{x}_i \\ & \xi_i \geq 0 \text{ for all } i \end{array}$$

The sum of slack variables is called the total hinge loss (see also Section 6.3.2). In practice, one problem with this linear program is that given a choice of a perfect separator with a very tiny margin, or a separator with a small amount of hinge loss but a larger margin, one may prefer the latter. The reason is that separators with large margin turn out to enjoy stronger overfitting guarantees. In particular, we may wish to use $1/\gamma^2$ as a regularization term where γ is the margin of the separator. Recall that when we defined margin, we normalized by the length of the prediction vector; if we do that in the linear program above, we see that the margin is $1/|\mathbf{w}|$, so $1/\gamma^2 = |\mathbf{w}|^2$. The method known as Support Vector Machines (SVMs) does exactly this regularization, optimizing for a combination of hinge loss and margin. Specifically, SVMs solve the convex optimization problem (here, C is a constant that is determined empirically):

$$\begin{array}{rl} \text{minimize} & |\mathbf{w}|^2 + C\sum_i \xi_i \\ \text{subject to} & \mathbf{w}\cdot\mathbf{x}_i \geq 1-\xi_i \text{ for all positive examples } \mathbf{x}_i \\ & \mathbf{w}\cdot\mathbf{x}_i \leq -1+\xi_i \text{ for all negative examples } \mathbf{x}_i \\ & \xi_i \geq 0 \text{ for all } i. \end{array}$$

Alternatively, rather than perform this convex optimization, it turns out that the simple Perceptron algorithm actually enjoys strong guarantees in terms of hinge loss and margin, *automatically* doing nearly as well as the optimal tradeoff between the two quantities:

Theorem 6.16 *On any sequence of examples in the unit ball, the number of mistakes of the Perceptron algorithm satisfies:*

$$\#\ \textit{mistakes} \leq \min_{\mathbf{w}^*}\left[|\mathbf{w}^*|^2 + 2(\textit{total hinge loss of}\ \mathbf{w}^*)\right].$$

Proof: Fix some $\mathbf{w}^*$ and consider the Perceptron algorithm run on a sequence of examples $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$. Define ξ_i to be the minimum non-negative value such that $f^*(\mathbf{x}_i)(\mathbf{w}^*\cdot\mathbf{x}_i) \geq 1-\xi_i$, i.e., the hinge loss of $\mathbf{w}^*$ on $\mathbf{x}_i$. Now, if the Perceptron algorithm makes a mistake on some $\mathbf{x}_i$, we add $f^*(\mathbf{x}_i)\mathbf{x}_i$ to $\mathbf{w}$. This increases $\mathbf{w}^*\cdot\mathbf{w}$ by at least $1-\xi_i$. So, if the Perceptron algorithm makes M mistakes we have $\mathbf{w}\cdot\mathbf{w}^* \geq M-L$ where L is the total hinge loss of $\mathbf{w}^*$. On the other hand, each mistake increases $|\mathbf{w}|^2$ by at most 1 since if we make a mistake on a example $\mathbf{x}_i$ we have

$$|\mathbf{w}+f^*(\mathbf{x}_i)\mathbf{x}_i|^2 = |\mathbf{w}|^2 + 2(\mathbf{w}\cdot f^*(\mathbf{x}_i)\mathbf{x}_i) + |\mathbf{x}_i|^2 \leq |\mathbf{w}^2| + 1$$

where the last inequality comes from the fact that $\mathbf{w}$ made a mistake so $\mathbf{w}\cdot f^*(\mathbf{x}_i)\mathbf{x}_i \leq 0$, and $\mathbf{x}_i$ lies in the unit ball so $|\mathbf{x}_i|^2 \leq 1$. Now, using the fact that $\mathbf{w}^*\cdot\mathbf{w} \leq |\mathbf{w}^*||\mathbf{w}|$ we get

$$\begin{array}{rcll} M-L & \leq & |\mathbf{w}^*|\sqrt{M} & \\ M^2-2ML+L^2 & \leq & |\mathbf{w}^*|^2 M & \text{(square both sides)} \\ M-2L+L^2/M & \leq & |\mathbf{w}^*|^2 & \text{(divide by } M) \\ M & \leq & |\mathbf{w}^*|^2+2L. & \text{(ignore negative term } -L^2/M) \end{array}$$

■

6.6 Nonlinear Separators and Kernel Functions

What if our data doesn't even have a "pretty good" linear separator? For example, perhaps the positive examples lie inside the unit ball and the negative examples lie outside the unit ball. Or perhaps there is some other smooth but nonlinear surface that separates the positive and negative examples. An approach to addressing problems of this nature is to use a tool called Kernel functions, also called the *kernel trick*.

One thing we might like to do is map our data to a higher dimensional space, e.g., look at all products of pairs or triples of features, in the hope that data will be linearly separable in this $O(d^2)$ or $O(d^3)$-dimensional space. If we are lucky, data will be separable by a large margin in this new space so we don't have to pay a lot in terms of mistakes if we run, say, the Perceptron algorithm. But this is going to be a huge amount of work computationally if we have to explicitly compute all these products to map our data into this higher-dimensional space. However, it turns out that many learning algorithms only access data through performing dot-products—we will see how to interpret the Perceptron algorithm in this way shortly. So, perhaps we can perform our mapping *implicitly* by just providing a simple way to compute the associated dot-product. This is the idea behind kernel functions.

Definition 6.1 *A* kernel function *is a function $K(\mathbf{x}, \mathbf{x}')$ such that for some $\phi : \mathcal{X} \to R^m$ (m could even be infinite) we have*

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}').$$

Some examples of kernel functions are:

- $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^k$ for integer $k \geq 1$. This is called the polynomial kernel.
- $K(\mathbf{x}, \mathbf{x}') = (1 + x_1 x_1')(1 + x_2 x_2')...(1 + x_d x_d')$. This is called the all-products kernel.
- String kernels, which count how many different substrings of some given length p two strings $\mathbf{x}$ and $\mathbf{x}'$ have in common.

Let's see why the above functions indeed satisfy the kernel definition. The easiest is the string kernel. This corresponds to a ϕ that maps examples into an implicit feature space with one coordinate for each possible string of length p: $\phi(x)$ has a 1 in coordinate j if x has string j as a substring, and a 0 if not.

For the all-products kernel, let's first look at the case $d = 2$. Here we get $K(\mathbf{x}, \mathbf{x}') = 1 + x_1 x_1' + x_2 x_2' + x_1 x_2 x_1' x_2'$. Thus, this corresponds to a mapping $\phi(\mathbf{x}) = (1, x_1, x_2, x_1 x_2)$. For $d = 3$ we get the previous kernel times $(1 + x_3 x_3')$ which corresponds to the mapping $\phi(\mathbf{x}) = (1, x_1, x_2, x_1 x_2, x_3, x_1 x_3, x_2 x_3, x_1 x_2 x_3)$. More generally, by induction, we can see this kernel corresponds to a dot-product in a space with one coordinate for every subset $A \subseteq \{1, \ldots, d\}$ where coordinate A of $\phi(\mathbf{x})$ equals $\prod_{i \in A} x_i$.

For the polynomial kernel, it is helpful to establish some composition properties of kernel functions. The fact that it is a legal kernel follows immediately from the following theorem (plus the fact that the constant function $K(\mathbf{x}, \mathbf{x}') = 1$ is a legal kernel).

Theorem 6.17 *Suppose K and K' are kernel functions. Then*

1. For any constant $c \geq 0$, cK is a legal kernel.

2. The sum $K + K'$, is a legal kernel.

3. The product, KK', is a legal kernel.

Proof: Let ϕ, ϕ' denote mappings such that $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$ and $K'(\mathbf{x}, \mathbf{x}') = \phi'(\mathbf{x}) \cdot \phi'(\mathbf{x}')$. We can now define ϕ'' for the new kernel in each case as follows.

1. Use $\phi''(\mathbf{x}) = \sqrt{c}\phi(\mathbf{x})$. So $\phi''(\mathbf{x}) \cdot \phi''(\mathbf{x}') = \sqrt{c}\phi(\mathbf{x}) \cdot \sqrt{c}\phi(\mathbf{x}') = cK(\mathbf{x}, \mathbf{x}')$.

2. Use $\phi''(\mathbf{x}) = \phi(\mathbf{x}) \circ \phi'(\mathbf{x})$ where "$\circ$" is concatenation. Then $\phi''(\mathbf{x}) \cdot \phi''(\mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}') + \phi'(\mathbf{x}) \cdot \phi'(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + K'(\mathbf{x}, \mathbf{x}')$.

3. Use $\phi''(\mathbf{x})$ as the outer-product of $\phi(\mathbf{x})$ and $\phi'(\mathbf{x})$. In particular, $\phi''(\mathbf{x})_{ij} = \phi(\mathbf{x})_i \phi'(\mathbf{x})_j$. Now we get:

$$\begin{aligned} \phi''(\mathbf{x}) \cdot \phi''(\mathbf{x}') &= \sum_{ij} \phi(\mathbf{x})_i \phi'(\mathbf{x})_j \phi(\mathbf{x}')_i \phi'(\mathbf{x}')_j \\ &= (\sum_i \phi(\mathbf{x})_i \phi(\mathbf{x}')_i)(\sum_j \phi'(x)_j \phi'(\mathbf{x}')_j) \\ &= K(\mathbf{x}, \mathbf{x}')K'(\mathbf{x}, \mathbf{x}'). \end{aligned}$$

■

What makes kernel functions important for learning is that many of the algorithms for learning linear separators only access their data via taking dot-products. So, if we use the kernel function any time that a dot product is requested, the algorithm will act *as if* we had explicitly mapped all examples $\mathbf{x}$ using $\phi(\mathbf{x})$. Let's consider the Perceptron algorithm. Let $\mathcal{M}$ denote the set of examples on which a mistake was made by the algorithm so far. Then, the current weight vector $\mathbf{w}$ is exactly:

$$\mathbf{w} = \sum_{\mathbf{x}_i \in \mathcal{M}} f^*(\mathbf{x}_i)\mathbf{x}_i.$$

The algorithm makes predictions by computing $\mathbf{w} \cdot \mathbf{x}$. So, to run it in the ϕ-space (i.e., on the examples $\phi(\mathbf{x})$) we just plug in the above formula for $\mathbf{w}$ and replace each dot-product with a kernel function computation. That is, we predict on example $\mathbf{x}$ using:

$$\text{sgn}\left(\sum_{\mathbf{x}_i \in \mathcal{M}} f^*(\mathbf{x}_i)K(\mathbf{x}_i, \mathbf{x})\right). \qquad (6.2)$$

Since our algorithm is making predictions exactly as the Perceptron algorithm would on the points $\phi(\mathbf{x})$, this means that if there exists a linear separator with low hinge-loss at a large margin in the ϕ-space, then Theorem 6.16 applies and we will make few mistakes. Specifically, we have the following.

Corollary 6.18 *Let K be a kernel function corresponding to mapping ϕ, such that for all $\mathbf{x}$ in the unit ball in R^d, $\phi(\mathbf{x})$ is in the unit ball in R^N, where N is the dimension of the ϕ-space. Then for any sequence of examples in the unit ball, the number of mistakes of the kernelized Perceptron algorithm satisfies:*

$$\#\ \textit{mistakes} \leq \min_{\mathbf{w}^* \in R^N} \left[|\mathbf{w}^*|^2 + 2(\textit{total hinge loss of } \mathbf{w}^* \textit{ over the examples } \phi(\mathbf{x})) \right].$$

Note that for a kernel to be helpful, it is crucial not only that data be (nearly) linearly separable in the ϕ-space but also that this separator have a large margin. For example, consider the "trivial kernel" $K(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x} = \mathbf{x}'$ else $K(\mathbf{x}, \mathbf{x}') = 0$. This corresponds to a mapping ϕ in which each example $\mathbf{x}$ gets its own coordinate. Any set of labeled examples is linearly separable, and yet an algorithm such as Perceptron will merely be memorizing its previously-seen data and will predict a default value of sgn(0) on any new example not yet seen.

Support Vector Machines can similarly be "kernelized", that is, run using a kernel function instead of dot-product, by writing them in their dual form.

Thinking about and choosing kernel functions: One way to think about what the kernelized Perceptron algorithm is doing in formula (6.2) is as follows. Given a new example $\mathbf{x}$, the algorithm looks at all previous examples $\mathbf{x}_i$ on which it made a mistake and has each one "vote" on the label of $\mathbf{x}$. Example $\mathbf{x}_i$ votes for its own label $f^*(\mathbf{x}_i)$ and the weight of its vote is $K(\mathbf{x}_i, \mathbf{x})$. Thus, from this perspective, a rule of thumb for choosing a kernel is to choose a function that seems like a reasonable measure of similarity for the type of data one has. This means that more similar examples will vote with higher weight. For example, with text data, a reasonable measure of similarity between two pieces of text might be how many substrings they share in common, perhaps weighted by the length of the substrings; so, this would be a natural kernel function. For data where distance in the original feature space is a reasonable measure of similarity, a natural kernel is the Gaussian "radial basis function" (RBF) kernel $K_\sigma(\mathbf{x}, \mathbf{x}') = exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2\sigma^2})$. With this kernel, the weight of the vote of $\mathbf{x}_i$ on example $\mathbf{x}$ is a decreasing function of the distance between $\mathbf{x}_i$ and $\mathbf{x}$. Thus, with this kernel, the algorithm is acting as a form of weighted nearest-neighbor algorithm. There has also been theoretical work showing that intuition of this form can be made precise more generally, and giving guarantees on kernel functions that talk in terms of their properties as similarity functions rather than in terms of implicit ϕ-spaces.

6.7 Strong and Weak Learning - Boosting

We now describe an approach called *boosting*, which is important both as a theoretical result and as a practical and easy-to-use learning method.

A strong learner for a class $\mathcal{H}$ is an algorithm that if $f^* \in \mathcal{H}$ is able with high probability to achieve any desired error rate ϵ, using a number of samples that may depend polynomially on $1/\epsilon$. For example, we presented a strong learner for the class of disjunctions. A weak learner for a class $\mathcal{H}$ is an algorithm that just needs to do a little bit better than random guessing. It is only required to with high probability get error rate $\leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. We show here that if we have a weak-learner for a class $\mathcal{H}$, and this algorithm achieves the weak-learning guarantee for any distribution of data $\mathcal{D}$, we can "boost" it to a strong learner, using the technique of Boosting.

For simplicity, lets assume our given weak learning algorithm A outputs hypotheses that can be described using at most b bits. Our boosted algorithm will produce hypotheses that will be majority votes over T hypotheses from A, for T defined below. This means the hypotheses of our boosted algorithm can be described using $O(bT)$ bits. So, by Theorem 6.5, it will suffice to draw a sample S of n labeled examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ at the beginning of the process for $n \gg \frac{1}{\epsilon}(bT + \log(1/\delta))$ and show that the rule produced by our procedure is correct on the sample.

Our assumption is that A is a weak-learner over any distribution on data. This in particular includes distributions that correspond to different ways of weighting the points in the training sample S. This is in fact all we will need. Specifically, we can define our assumption on A as follows.

Definition 6.2 (γ-Weak learner on sample) *A weak learner over a training sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is an algorithm that given the examples, their labels, and a nonnegative real weight w_i on each example $\mathbf{x}_i$ as input, produces a classifier that correctly labels a subset of examples with total weight at least $(\frac{1}{2} + \gamma) \sum_{i=1}^{n} w_i$.*

Theorem 6.19 *Suppose A satisfies Definition 6.2 for sample S. Then the boosting procedure below will produce a classifier h that is a majority vote of $T = O(\frac{1}{\gamma^2} \log n)$ classifiers produced by running algorithm A on different weightings of S, such that $err_S(h) = 0$.*

Proof: At the high level, Boosting makes use of the intuitive notion that if an example was misclassified, one needs to pay more attention to it. The procedure we will examine is as follows.

Boosting algorithm

> Make the first call to the weak learner with all w_i set equal to 1.
>
> At time $t+1$ multiply the weight of each example that was misclassified the previous time by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave the other weights as they are. Make a call to the weak learner.
>
> After T steps, stop and output the following classifier:
>
> > Label each of the examples $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ by the label given to it by a majority of calls to the weak learner. Assume T is odd, so there is no tie for the majority.

Suppose m is the number of examples the final classifier gets wrong. Each of these m examples was misclassified at least $T/2$ times so each has weight at least $\alpha^{T/2}$. This says the total weight is at least $m\alpha^{T/2}$. On the other hand, at time $t+1$, only the weights of examples misclassified at time t were increased. By the property of weak learning, the total weight of misclassified examples is at most $(\frac{1}{2}-\gamma)$ of the total weight at time t. Let weight(t) be the total weight at time t. Then

$$\begin{aligned}\text{weight}(t+1) &\leq \left(\alpha\left(\frac{1}{2}-\gamma\right)+\left(\frac{1}{2}+\gamma\right)\right)\times \text{weight}(t)\\ &= (1+2\gamma)\times\text{weight}(t).\end{aligned}$$

Thus, since weight$(0) = n$,

$$m\alpha^{T/2} \leq \text{Total weight at end} \leq n(1+2\gamma)^T.$$

Plugging in $\alpha = \frac{1/2+\gamma}{1/2-\gamma} = \frac{1+2\gamma}{1-2\gamma}$ and rearranging terms we get:

$$m \;\leq\; n[(1-2\gamma)(1+2\gamma)]^{T/2} \;=\; n[1-4\gamma^2]^{T/2}.$$

Finally, using the fact that $1-x \leq e^{-x}$ we have:

$$m \;\leq\; ne^{-2T\gamma^2}.$$

For $T > \frac{\ln n}{2\gamma^2}$ we have $m < 1$, so the number of misclassified items m must be 0. ■

Now that we have completed the proof of the boosting result, here are two interesting observations:

Connection to Hoeffding bounds: The boosting result applies even if our weak learning algorithm is "adversarial", giving us the least helpful classifier possible subject to satisfying Definition 6.2. (For instance, this is why we don't want to set α to be *too* large in the Boosting algorithm: because then the weak learner could just give us the negation of the classifier it gave us last time.) Suppose, though, that the weak learning algorithm was "nice" and each time just gave a classifier that for each example $\mathbf{x}_i$, flipped a coin and produced the correct answer with probability $\frac{1}{2}+\gamma$ and the wrong answer with probability $\frac{1}{2}-\gamma$ (so it is a γ-weak learner in expectation). In that case, if we repeatedly called it T times, for any fixed $\mathbf{x}_i$, Hoeffding bounds imply the chance the majority vote of those classifiers is incorrect on $\mathbf{x}_i$ is at most $e^{-2T\gamma^2}$. So, the expected total number of mistakes m is at most $ne^{-2T\gamma^2}$. What is interesting is that this is the exact bound we get from boosting (without the expectation) for an adversarial weak-learner!

A minimax view: Consider a 2-player zero-sum game with one row for each example $\mathbf{x}_i$ and one column for each hypothesis h_j that the weak-learning algorithm might possibly output. Say that if the row player chooses row i and the column player chooses column j, then the column player gets a payoff of 1 if $h_j(\mathbf{x}_i)$ is correct, else it gets a payoff of 0 if $h_j(\mathbf{x}_i)$ is incorrect. The γ-weak learning assumption implies that for any randomized strategy for the row player (any "mixed strategy" in the language of game theory), there exists a response h_j that gives the column player an expected payoff at least $\frac{1}{2}+\gamma$. The von Neumann minimax theorem states that this implies that there exists a probability distribution on the columns (a mixed strategy for the column player) such that for any $\mathbf{x}_i$, at least a $\frac{1}{2}+\gamma$ probability mass of the columns under this distribution is correct on $\mathbf{x}_i$. We can think of boosting as a fast way of finding a very simple probability distribution on the columns (just an average over $O(\log n)$ columns, possibly with repetitions) that is nearly as good (for any $\mathbf{x}_i$, more than half are correct) that moreover works even if our only access to the columns is by running the weak learner and observing what it outputs.

One last interesting point about boosting. We argued above that $T = O(\frac{1}{\gamma^2}\log n)$ rounds of boosting are sufficient to produce a majority-vote rule h that will classify all of S correctly. Using our Occam bounds and a bit of algebra, this implies that if each weak hypothesis can be described in b bits, then a sample size $n = O(\frac{1}{\epsilon}(\frac{b}{\gamma^2}\log(\frac{b}{\epsilon\gamma\delta}))$ is sufficient to conclude that with probability $1-\delta$ we have $err_{\mathcal{D}}(h) \leq \epsilon$. In fact, it turns out that running the boosting procedure for larger values of T (i.e., continuing past the point where S is classified correctly by the final majority vote), does not actually lead to greater overfitting. The reason is that using the same type of analysis we used to prove Theorem 6.19, one can show that as T increases, then not only will the majority vote be correct on each $\mathbf{x} \in S$, but in fact each example will be correctly classified by a $\frac{1}{2}+\gamma'$ fraction of the classifiers, where $\gamma' \to \gamma$ as $T \to \infty$. (I.e., the vote is approaching the minimax optimal strategy for the column player in the minimax view given above.) This in turn implies that h can be well-approximated over S by a vote of a random sample of $O(1/\gamma^2)$ of its component weak hypotheses h_j. Since these small random samples are not overfitting by much (by our Occam's razor theorem), one can show that this implies that h cannot be overfitting by much either.

6.8 Combining (Sleeping) Expert Advice

Imagine you have access to a large collection of rules-of-thumb that specify what to predict in different situations. For example, in classifying news articles, you might have one that says "if the article has the word 'football' then classify it as sports" and another that says "if the article contains a dollar figure, then classify it as business". These rules might at times contradict each other (e.g., a news article that has both the word 'football' and a dollar figure) and it may be that none is perfectly accurate and indeed some are much better than others. We present here an algorithm for combining a large number of such rules with the guarantee that if any of them indeed are good, the algorithm will perform nearly as well as each good rule on the examples on which that rule applies.

Formally, assume we have n rules $h_1, \ldots, h_n$, and let S_i denote the subset of examples in which rule h_i fires, i.e., the examples satisfying its if-condition. We consider the online learning model, and let $mistakes(A, S)$ denote the number of mistakes of some algorithm A on a sequence of examples S. Then the guarantee of our algorithm A is that:

$$\text{For all } i, E[mistakes(A, S_i)] \leq (1+\epsilon) \cdot mistakes(h_i, S_i) + O\left(\frac{\log n}{\epsilon}\right)$$

where ϵ is a parameter of algorithm A and the expectation is over internal randomness in the randomized algorithm A.

As a special case, if $h_1, \ldots, h_n$ always fire, and indeed are the members of a hypothesis class $\mathcal{H}$, then A performs nearly as well as the best function in $\mathcal{H}$. This can be viewed as a noise-tolerant version of the Halving Algorithm of Section 6.4.2 for the case that no function in $\mathcal{H}$ is perfect. The case of rules that all fire on every example is often called the problem of *combining expert advice*, and the more general case of rules that sometimes fire is called the *sleeping experts* problem (viewing a rule not firing as it being asleep).

Combining Sleeping Experts Algorithm:

Initialize each expert h_i with a weight $w_i = 1$. Let $\epsilon \in (0, 1)$. For each example x seen, do the following:

1. [Make prediction] let H_x denote the experts h_i that make a prediction on x, and let $W_x = \sum_{h_j \in H_x} w_j$. Choose $h_i \in H_x$ with probability $p_{ix} = w_i / W_x$ and predict $h_i(x)$.

2. [Receive feedback] Given the correct label, for each $h_i \in H_x$ let $m_{ix} = 1$ if $h_i(x)$ was incorrect, else let $m_{ix} = 0$.

3. [Update weights] For each $h_i \in H_x$, update its weight as follows:

   - Let $r_{ix} = \left(\sum_{h_j \in H_x} p_{jx} m_{jx}\right) / (1+\epsilon) - m_{ix}$.
   - Update $w_i \leftarrow w_i (1+\epsilon)^{r_{ix}}$.

Note that $\sum_{h_j \in H_x} p_{jx} m_{jx}$ represents the algorithm's probability of making a mistake on example x. So, h_i is rewarded for predicting correctly ($m_{ix} = 0$) especially when the algorithm had a high probability of making a mistake, and h_i is penalized for predicting incorrectly ($m_{ix} = 1$) especially when the algorithm had a low probability of making a mistake.

For each $h_i \notin H_x$, leave w_i alone.

Theorem 6.20 *For any set of n if-then rules (sleeping experts) $h_1, \ldots, h_n$, and for any sequence of examples S, the Combining Sleeping Experts Algorithm A satisfies:*

$$\textit{For all } i, E[\textit{mistakes}(A, S_i)] \le (1+\epsilon) \cdot \textit{mistakes}(h_i, S_i) + O\left(\frac{\log n}{\epsilon}\right)$$

where $S_i = \{x \in S : h_i \in H_x\}$.

Proof: Consider some sleeping expert h_i. The weight of h_i after the sequence of examples S is exactly:

$$\begin{aligned} w_i &= (1+\epsilon)^{\sum_{x \in S_i}\left[\left(\sum_{h_j \in H_x} p_{jx} m_{jx}\right)/(1+\epsilon) - m_{ix}\right]} \\ &= (1+\epsilon)^{E[\textit{mistakes}(A,S_i)]/(1+\epsilon) - \textit{mistakes}(h_i,S_i)}. \end{aligned}$$

Let $W = \sum_j w_j$. Clearly $w_i \le W$. Therefore, taking logs, we have:

$$E[\textit{mistakes}(A, S_i)]/(1+\epsilon) - \textit{mistakes}(h_i, S_i) \le \log_{1+\epsilon} W.$$

So, using the fact that $\log_{1+\epsilon} W = O(\frac{\log W}{\epsilon})$,

$$E[\textit{mistakes}(A, S_i)] \le (1+\epsilon) \cdot \textit{mistakes}(h_i, S_i) + O\left(\frac{\log W}{\epsilon}\right).$$

Initially, $W = n$. To prove the theorem, it thus is enough to just prove that W never increases. To do so, we need to show that for each x, $\sum_{h_i \in H_x} w_i (1+\epsilon)^{r_{ix}} \le \sum_{h_i \in H_x} w_i$, or equivalently (dividing both sides by $\sum_{h_j \in H_x} w_j$) that $\sum_i p_{ix}(1+\epsilon)^{r_{ix}} \le 1$, where for convenience we define $p_{ix} = 0$ for $h_i \notin H_x$.

For this we will use the inequalities that for $\beta, z \in [0,1]$ we have $\beta^z \le 1 - (1-\beta)z$ and $\beta^{-z} \le 1 + (1-\beta)z/\beta$. Specifically, we will use $\beta = (1+\epsilon)^{-1}$. We now have:

$$\begin{aligned} \sum_i p_{ix}(1+\epsilon)^{r_{ix}} &= \sum_i p_{ix} \beta^{m_{ix} - (\sum_j p_{jx} m_{jx})\beta} \\ &\le \sum_i p_{ix} \Big(1 - (1-\beta) m_{ix}\Big) \left(1 + (1-\beta)\left(\sum_j p_{jx} m_{jx}\right)\right) \\ &\le \left(\sum_i p_{ix}\right) - (1-\beta)\sum_i p_{ix} m_{ix} + (1-\beta)\sum_i p_{ix} \sum_j p_{jx} m_{jx} \\ &= 1 - (1-\beta)\sum_i p_{ix} m_{ix} + (1-\beta)\sum_j p_{jx} m_{jx} \\ &= 1, \end{aligned}$$

where the second-to-last line follows from using $\sum_i p_{ix} = 1$ in two places. So W never increases and the bound follows as desired. ∎

6.9 VC-Dimension

In Section 6.2 we presented several theorems showing that so long as the training set S is sufficiently large compared to $\log(|\mathcal{H}|)$, we can be confident that functions $h \in \mathcal{H}$ that perform well on S will also perform well on $\mathcal{D}$. In essence, these results used $\log(|\mathcal{H}|)$ as a measure of complexity of class $\mathcal{H}$. VC-dimension is a different, tighter measure of complexity for a class of functions and, as we will see, also is sufficient to yield confidence bounds. For any class $\mathcal{H}$, $\text{VCdim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$ and often it is quite a bit smaller. For example, there are infintely many linear separators in R^2 and yet their VC-dimension is only 3. We begin with an auxiliary definition:

Definition 6.3 *Given a set S of examples and class of functions $\mathcal{H}$, let $\mathcal{H}[S]$ denote the set of all labelings of the points in S that are consistent with some function in $\mathcal{H}$. We say that S is* **shattered** *if $|\mathcal{H}[S]| = 2^{|S|}$.*

For example, any set S of three non-collinear points in the plane is shattered by the set of linear separators, since any of the 8 possible ways of labeling them with labels in $\{-1, 1\}$ can be realized by a function in this class.

Definition 6.4 *For integer m and class $\mathcal{H}$, let $\mathcal{H}[m] = \max_{|S|=m} |\mathcal{H}[S]|$; this is called the* **growth function** *of $\mathcal{H}$. The* **VC-dimension** *of $\mathcal{H}$ is the largest m such that $\mathcal{H}[m] = 2^m$. That is, VC-dimension is the size of the largest shattered set.*

For example, consider the class of linear separators in the plane. As noted above, any set of three non-collinear points is shattered by the class of linear separators, and yet it is not hard to see that no set of 4 points can be shattered. In particular, for any set of 4 points in the plane, out of all the 16 possible binary labelings, at most 14 will be realizable using linear separators. So, the VC-dimension of linear separators in the plane is 3. The growth function of this class satisfies $\mathcal{H}[m] = O(m^2)$, since for any labeling consistent with a linear separator, you can describe a consistent separator using two boundary data points along with $O(1)$ additional bits giving the labels of those two points and stating which side is which. More generally, the VC-dimension of linear separators in R^d is $d+1$ (see Section 6.9.4). As another example, the class of intervals $[a, b]$ on the real line has VC-dimension 2 since any set of two points is shattered, and yet for three points $x_1 < x_2 < x_3$ it is not possible to label x_1 and x_3 positive but x_2 negative. The growth function of this class also satisfies $\mathcal{H}[m] = O(m^2)$. Notice that $\text{VCdim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$ since $\mathcal{H}$ must have at least 2^d functions in order to shatter d points.

We now will prove three important theorems relating VC-dimension to learnability. The first two can be viewed as analogs of Theorem 6.4 and Theorem 6.7 respectively, replacing the size of the class $\mathcal{H}$ with its growth function. The third relates the growth function of a class to its VC-dimension.

Theorem 6.21 (VC Occam bound) *For any class $\mathcal{H}$ and distribution $\mathcal{D}$, if a training sample S is drawn from $\mathcal{D}$ of size*

$$m \;\geq\; \frac{2}{\epsilon}[\log_2(2\mathcal{H}[2m]) + \log_2(1/\delta)]$$

then with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) \geq \epsilon$ has $err_S(h) > 0$ (equivalently, every $h \in \mathcal{H}$ with $err_S(h) = 0$ has $err_{\mathcal{D}}(h) < \epsilon$).

Theorem 6.22 (VC uniform convergence) *For any class $\mathcal{H}$ and distribution $\mathcal{D}$, if a training sample S is drawn from $\mathcal{D}$ of size*

$$m \;\geq\; \frac{8}{\epsilon^2}[\ln(2\mathcal{H}[2m]) + \ln(1/\delta)]$$

then with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ will have $|err_S(h) - err_{\mathcal{D}}(h)| \leq \epsilon$.

Theorem 6.23 (Sauer's lemma) $\mathcal{H}[m] \leq \sum_{i=0}^{\mathrm{VCdim}(\mathcal{H})} \binom{m}{i} \leq m^{\mathrm{VCdim}(\mathcal{H})}$.

Notice that Sauer's lemma is not necessarily tight: e.g., in the case of linear separators in the plane we have $\mathcal{H}[m] = O(m^2)$ and yet $\mathrm{VCdim}(\mathcal{H}) = 3$. Putting Theorems 6.21 and 6.23 together, with a little algebra we get the following corollary (a similar corollary results by combining Theorems 6.22 and 6.23):

Corollary 6.24 *For any class $\mathcal{H}$ and distribution $\mathcal{D}$, a training sample S of size*

$$O\left(\frac{1}{\epsilon}[\mathrm{VCdim}(\mathcal{H})\log(1/\epsilon) + \log(1/\delta)]\right)$$

is sufficient to ensure that with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) \geq \epsilon$ has $err_S(h) > 0$ (equivalently, every $h \in \mathcal{H}$ with $err_S(h) = 0$ has $err_{\mathcal{D}}(h) < \epsilon$).

Before proving the above theorems, let's first use Sauer's lemma to show a useful fact about VC-dimension. Given a class $\mathcal{H}$ and integer k, define $\mathrm{MAJ}_k(\mathcal{H})$ to be the set of functions achievable by taking a majority vote over k functions in $\mathcal{H}$. Then we have:

Corollary 6.25 *If $\mathcal{H}$ has VC-dimension d then $MAJ_k(\mathcal{H})$ has VC-dimension $O(kd\log(kd))$.*

Proof: Let m be the VC-dimension of $\mathrm{MAJ}_k(\mathcal{H})$, so by definition, there must exist a set S of m points shattered by $\mathrm{MAJ}_k(\mathcal{H})$. We know by Sauer's lemma that there are at most m^d ways of partitioning the points in S using functions in $\mathcal{H}$. Since each function in $\mathrm{MAJ}_k(\mathcal{H})$ is determined by k functions in $\mathcal{H}$, this means there are at most $(m^d)^k = m^{kd}$ ways of partitioning the points using functions in $\mathrm{MAJ}_k(\mathcal{H})$. Since S is shattered, we therefore must have $2^m \leq m^{kd}$, or equivalently $m \leq kd\log_2(m)$. We can solve this as follows. First, assuming $m \geq 16$ we have $\log_2(m) \leq \sqrt{m}$ so $kd\log_2(m) \leq kd\sqrt{m}$ which implies that $m \leq (kd)^2$. To get the better bound, we can then just plug back in to our original inequality: since $m \leq (kd)^2$, it must be that $\log_2(m) \leq 2\log_2(kd)$, so our original inequality implies $m \leq 2kd\log_2(kd)$. ∎

Corollary 6.25 implies that for our boosting algorithm of Section 6.7, so long as the weak learning algorithm A selects rules from a class of bounded VC-dimension d, then the class of rules that can be produced by the booster running for $T = \tilde{O}(1/\gamma^2)$ rounds (using the $\tilde{O}$ notation to ignore logarithmic factors) has VC-dimension $\tilde{O}(d/\gamma^2)$. This in turn gives a bound on the number of samples needed, via Corollary 6.24.

6.9.1 Proof of Theorem 6.21 (VC Occam bound)

Consider drawing a set S of m examples from $\mathcal{D}$ and let A denote the event that there exists $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) > \epsilon$ but $err_S(h) = 0$. Our goal is to prove that $\Pr[A] \leq \delta$.

Now, consider drawing *two* sets S, S' of m examples each from $\mathcal{D}$. Define A as above and let B denote the event that there exists $h \in \mathcal{H}$ with $err_S(h) = 0$ but $err_{S'}(h) \geq \epsilon/2$.

Lemma 6.26 $\Pr[B] \geq \Pr[A]/2$.

Proof: Clearly, $\Pr[B] \geq \Pr[A \cap B] = \Pr[A]\Pr[B|A]$. Consider drawing set S and suppose event A occurs. Let h be some hypothesis in $\mathcal{H}$ with $err_{\mathcal{D}}(h) > \epsilon$ but $err_S(h) = 0$. Now, draw set S'. $\mathbf{E}[err_{S'}(h)] = err_{\mathcal{D}}(h) > \epsilon$. So, by Chernoff bounds, since $m > 8/\epsilon$, $\Pr[err_{S'}(h) \geq \epsilon/2] \geq 1/2$. So, we have $\Pr[B|A] \geq 1/2$ and thus $\Pr[B] \geq \Pr[A]/2$ as desired. ■

So, it suffices to prove that $\Pr[B] \leq \delta/2$. Now, let us consider a third experiment. Suppose we randomly draw a set S'' of $2m$ points from $\mathcal{D}$ and then randomly partition S'' into two sets S and S' of m points each. Let B^* denote the event that there exists $h \in \mathcal{H}$ with $err_S(h) = 0$ but $err_{S'}(h) \geq \epsilon/2$ under this experiment. $\Pr[B^*] = \Pr[B]$ since drawing $2m$ points i.i.d. from $\mathcal{D}$ and randomly partitioning them into two sets of size m produces the same distribution on (S, S') as does drawing S and S' directly. The advantage of this new experiment, however, is that we can now argue that $\Pr[B^*]$ is low by attempting to argue that for *any* set S'' of size $2m$, $\Pr[B^*|S'']$ is low, with probability now taken over just the random partition of S'' into S and S'. The key point now is that since S'' is fixed, there are only $|\mathcal{H}[S'']|$ events to worry about. Specifically, it suffices to prove that for any fixed $h \in \mathcal{H}[S'']$, the probability over the partition of S'' that h makes 0 mistakes on S but more than $\epsilon m/2$ mistakes on S' is at most $\delta/(2\mathcal{H}[2m])$. We can then just apply the union bound over all $h \in \mathcal{H}[S'']$.

To make the calculations easier, let's consider the following specific method for partitioning S'' into S and S'. First, randomly put the points in S'' into pairs: (a_1, b_1), (a_2, b_2), $\ldots$, (a_m, b_m). Then, for each index i, flip a fair coin: if heads put a_i into S and b_i into S', else if tails put a_i into S' and b_i into S. Now, fix some $h \in \mathcal{H}[S'']$ and let us consider the probability over these m fair coin flips that h makes 0 mistakes on S but more than $\epsilon m/2$ mistakes on S'. First of all, if for any index i, h makes a mistake on both a_i and b_i then the probability is zero (because it cannot possibly make 0 mistakes on S). Second, if there are fewer than $\epsilon m/2$ indices i such that h makes a mistake on either a_i or b_i then again the probability is zero (because it cannot possibly make more than $\epsilon m/2$ mistakes on S'). So, we may assume there are $r \geq \epsilon m/2$ indices i such that h makes a mistake on exactly one of a_i or b_i. In this case, the chance that *all* of those mistakes land in S' is exactly $1/2^r$. This quantity is at most $1/2^{\epsilon m/2} \leq \delta/(2\mathcal{H}[2m])$ as desired for m as given in the theorem statement. ∎

6.9.2 Proof of Theorem 6.22 (VC uniform convergence)

This proof is identical to the proof of Theorem 6.21 except B^* is now the event that there exists $h \in \mathcal{H}[S'']$ such that $|err_S(h) - err_{S'}(h)| \geq \epsilon/2$. We again consider the experiment where we randomly put the points in S'' into pairs (a_i, b_i) and then flip a fair coin for each index i, if heads placing a_i into S and b_i into S', else placing a_i into S' and b_i into S. Let us consider the difference between the number of mistakes h makes on S and the number of mistakes h makes on S' and observe how this difference changes as we flip coins for $i = 1, 2, \ldots, m$. Initially, the difference is 0. If h makes a mistake on both or neither of (a_i, b_i) then the difference does not change. Else, if h makes a mistake on exactly one of a_i or b_i, then with probability 1/2 the difference increases by 1 and with probability 1/2 the difference decreases by 1. Say there are $r \leq m$ such pairs. Then we are asking the question: if we take a random walk of $r \leq m$ steps, what is the probability that we end up more than $\epsilon m/2$ steps away from the origin? This is equivalent to asking: if we flip $r \leq m$ fair coins, what is the probability the number of heads differs from its expectation by more than $\epsilon m/4$. By Hoeffding bounds, this is at most $2e^{-\epsilon^2 m/8}$. This quantity is at most $\delta/(2\mathcal{H}[2m])$ as desired for m as given in the theorem statement. ∎

6.9.3 Proof of Theorem 6.23 (Sauer's lemma)

Let $d = \text{VCdim}(\mathcal{H})$. Our goal is to prove for any set S of m points that $|\mathcal{H}[S]| \leq \binom{m}{\leq d}$, where we are defining $\binom{m}{\leq d} = \sum_{i=0}^{d} \binom{m}{i}$; this is the number of distinct ways of choosing d or fewer elements out of m. We will do so by induction on m. As a base case, our theorem is trivially true if $m \leq d$.

As a first step in the proof, notice that:

$$\binom{m}{\leq d} = \binom{m-1}{\leq d} + \binom{m-1}{\leq d-1} \tag{6.3}$$

because we can partition the ways of choosing d or fewer items into those that do not include the first item (leaving $\leq d$ to be chosen from the remainder) and those that do include the first item (leaving $\leq d-1$ to be chosen from the remainder).

Now, consider any set S of m points and pick some arbitrary point $x \in S$. By induction, we may assume that $|\mathcal{H}[S \setminus \{x\}]| \leq \binom{m-1}{\leq d}$. So, by equation (6.3) all we need to show is that $|\mathcal{H}[S]| - |\mathcal{H}[S \setminus \{x\}]| \leq \binom{m-1}{\leq d-1}$. Thus, our problem has reduced to analyzing how many *more* labelings there are of S than there are of $S \setminus \{x\}$ using functions in $\mathcal{H}$.

If $\mathcal{H}[S]$ is larger than $\mathcal{H}[S \setminus \{x\}]$, it is because of pairs of labelings in $\mathcal{H}[S]$ that differ only on point x and therefore collapse to the same labeling when x is removed. For labeling $h \in \mathcal{H}[S]$ (this is a Boolean function defined only on S), define $\mathsf{twin}(h)$ to be the same as h except giving the opposite value to x; this may or may not belong to $\mathcal{H}[S]$. Let $\mathcal{T} = \{h \in \mathcal{H}[S] : h(x) = 1 \text{ and } \mathsf{twin}(h) \in \mathcal{H}[S]\}$. Notice $|\mathcal{H}[S]| - |\mathcal{H}[S \setminus \{x\}]| = |\mathcal{T}|$.

Now, what is the VC-dimension of $\mathcal{T}$? If $d' = \text{VCdim}(\mathcal{T})$, this means there is some set R of d' points in $S \setminus \{x\}$ that are shattered by $\mathcal{T}$. By definition of $\mathcal{T}$, all $2^{d'}$ labelings of R can be extended both ways to x using labelings in $\mathcal{H}[S]$; i.e., $R \cup \{x\}$ is shattered by $\mathcal{H}$. This means, $d' + 1 \leq d$. Since $\text{VCdim}(\mathcal{T}) \leq d - 1$, by induction we have $|\mathcal{T}| \leq \binom{m-1}{\leq d-1}$ as desired.

6.9.4 The VC-dimension of linear separators

We saw earlier that the class of linear separators in R^2 has a VC-dimension of 3. Here, we prove that the class of linear separators in R^d has a VC-dimension of $d + 1$.

The easy direction is that there exists a set of size $d + 1$ that can be shattered. Select the d unit-coordinate vectors plus the origin to be the $d + 1$ points. Suppose A is any subset of these $d + 1$ points not including the origin. Take a 0-1 vector $\mathbf{w}$ which has 1's precisely in the coordinates corresponding to vectors in A. Then the linear separator $\mathbf{w} \cdot \mathbf{x} \geq 1/2$ labels A as positive and its complement as negative, and similarly $\mathbf{w} \cdot \mathbf{x} \leq 1/2$ labels A negative and its complement as positive.

We now show that no set of $d + 2$ points in d-dimensions can be shattered by this class. We will do this by proving that any set of $d + 2$ points can be partitioned into two disjoint subsets A and B of points whose convex hulls intersect. This establishes the claim since any linear separator with A one one side must have its entire convex hull on that side,[18] so it is not possible to have a linear separator with A on one side and B on the other.

Let *convex*(S) denote the convex hull of point set S.

Theorem 6.27 (**Radon**)*: Any set $S \subseteq R^d$ with $|S| \geq d + 2$, can be partitioned into two disjoint subsets S_1 and S_2 such that convex$(S_1) \cap$ convex$(S_2) \neq \phi$.*

Proof: Without loss of generality, assume $|S| = d + 2$. Form a $d \times (d + 2)$ matrix with one column for each point of S. Call the matrix A. Add an extra row of all 1's to construct a $(d + 1) \times (d + 2)$ matrix B. Clearly, since the rank of this matrix is at most $d + 1$, the columns are linearly dependent. Say $\mathbf{x} = (x_1, x_2, \ldots, x_{d+2})$ is a nonzero vector with $B\mathbf{x} = 0$. Reorder the columns so that $x_1, x_2, \ldots, x_s \geq 0$ and $x_{s+1}, x_{s+2}, \ldots, x_{d+2} < 0$. Normalize $\mathbf{x}$ so $\sum\limits_{i=1}^{s} |x_i| = 1$. Let $\mathbf{b_i}$ (respectively $\mathbf{a_i}$) be the i^{th}

[18] If any two points $\mathbf{x}_1$, $\mathbf{x}_2$ lie on the same side of a separator, so must any convex combination: if $\mathbf{w} \cdot \mathbf{x}_1 \geq b$ and $\mathbf{w} \cdot \mathbf{x}_2 \geq b$ then $\mathbf{w} \cdot (a\mathbf{x}_1 + (1 - a)\mathbf{x}_2) \geq b$.

column of B (respectively A). Then, $\sum\limits_{i=1}^{s}|x_i|\mathbf{b_i} = \sum\limits_{i=s+1}^{d+2}|x_i|\mathbf{b_i}$ from which it follows that $\sum\limits_{i=1}^{s}|x_i|\mathbf{a_i} = \sum\limits_{i=s+1}^{d+2}|x_i|\mathbf{a_i}$ and $\sum\limits_{i=1}^{s}|x_i| = \sum\limits_{i=s+1}^{d+2}|x_i|$. Since $\sum\limits_{i=1}^{s}|x_i| = 1$ and $\sum\limits_{i=s+1}^{d+2}|x_i| = 1$ each side of $\sum\limits_{i=1}^{s}|x_i|\mathbf{a_i} = \sum\limits_{i=s+1}^{d+2}|x_i|\mathbf{a_i}$ is a convex combination of columns of A which proves the theorem. Thus, S can be partitioned into two sets, the first consisting of the first s points after the rearrangement and the second consisting of points $s+1$ through $d+2$. Their convex hulls intersect as required. ■

As noted above, Radon's theorem immediately implies that linear separators in R^d cannot shatter any set of $d+2$ points.

6.9.5 Other measures of complexity

VC-dimension and number of bits needed to describe a rule are not the only measures of complexity one can use to derive generalization guarantees. In fact, there has been significant work on a variety of measures. For example, Rademacher complexity measures the extent to which the given class of functions $\mathcal{H}$ can fit random noise. Given a set of m examples $S = \{x_1, \ldots, x_m\}$, the *empirical Rademacher complexity* of $\mathcal{H}$ is defined as $R_S(\mathcal{H}) = \mathbf{E}_{\sigma_1,\ldots,\sigma_m} \max_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)$, where $\sigma_i \in \{-1,1\}$ are independent random labels with $\Pr[\sigma_i = 1] = 0.5$. E.g., if you assign random ±1 labels to the points in S and the best classifier in $\mathcal{H}$ on average gets error 0.45 then $R_S(\mathcal{H}) = 0.55 - 0.45 = 0.1$. One can then prove that with probability $\geq 1-\delta$, every $h \in \mathcal{H}$ satisfies $err_{\mathcal{D}}(h) \leq err_S(h) + R_S(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}$. For more on results such as this, see, e.g., [**?**].

6.10 Current directions

We now briefly discuss a few current directions in machine learning, focusing on *semi-supervised* learning, *active* learning, *multi-task* learning, and *deep* learning.

6.10.1 Semi-supervised learning

Semi-supervised learning refers to the idea of trying to use a large unlabeled data set U to augment a given labeled data set L in order to produce more accurate rules than would have been achieved using just L alone. The motivation is that in many settings (e.g., document classification, image classification, speech recognition), unlabeled data is much more plentiful than labeled data, so one would like to make use of it if possible. Of course, unlabeled data is missing the labels! Nonetheless it often contains information that an algorithm can take advantage of.

As an example, suppose one believes the target function is a linear separator that separates most of the data by a large margin. By observing enough unlabeled data to estimate the probability mass near to any given linear separator, one could in principle then discard separators in advance that slice through dense regions and instead focus attention on just those that indeed separate most of the distribution by a large margin. This is the high level idea behind a technique known as Semi-Supervised SVMs. Alternatively, suppose data objects can be described by two different "kinds" of features (e.g., a webpage could be described using words on the page itself or using words on links pointing *to* the page), and one believes that each kind should be sufficient to produce an accurate classifier. Then one might want to train a *pair* of classifiers (one on each type of feature) and use unlabeled data for which one is confident but the other is not to bootstrap, labeling such examples with the confident classifier and then feeding them as training data to the less-confident one. This is the high-level idea behind a technique known as Co-Training. Or, if one believes "similar examples should generally have the same label", one might construct a graph with an edge between examples that are sufficiently similar, and aim for a classifier that is correct on the labeled data and has a small cut value on the unlabeled data; this is the high-level idea behind graph-based methods.

A formal model: The batch learning model introduced in Sections 6.1 and 6.2 in essence assumes that one's prior beliefs about the target function be described in terms of a class of functions $\mathcal{H}$. In order to capture the reasoning used in semi-supervised learning, we need to also describe beliefs about the *relation* between the target function and the data distribution. A clean way to do this is via a *notion of compatibility* χ between a hypothesis h and a distribution $\mathcal{D}$. Formally, χ maps pairs $(h, \mathcal{D})$ to $[0, 1]$ with $\chi(h, \mathcal{D}) = 1$ meaning that h is highly compatible with $\mathcal{D}$ and $\chi(h, \mathcal{D}) = 0$ meaning that h is very *in*compatible with $\mathcal{D}$. The quantity $1 - \chi(h, \mathcal{D})$ is called the *unlabeled error rate* of h, and denoted $err_{unl}(h)$. Note that for χ to be useful, it must be estimatable from a finite sample; to this end, let us further require that χ is an expectation over individual examples. That is, overloading notation for convenience, we require $\chi(h, \mathcal{D}) = \mathbf{E}_{x \sim \mathcal{D}}[\chi(h, x)]$, where $\chi : \mathcal{H} \times \mathcal{X} \to [0, 1]$.

For instance, suppose we believe the target should separate most data by margin γ. We can represent this belief by defining $\chi(h, x) = 0$ if x is within distance γ of the decision boundary of h, and $\chi(h, x) = 1$ otherwise. In this case, $err_{unl}(h)$ will denote the probability mass of $\mathcal{D}$ within distance γ of h's decision boundary. As a different example, in co-training, we assume each example can be described using two "views" that each are sufficient for classification; that is, there exist f_1^*, f_2^* such that for each example $x = \langle x_1, x_2 \rangle$ we have $f_1^*(x_1) = f_2^*(x_2)$. We can represent this belief by defining a hypothesis $h = \langle h_1, h_2 \rangle$ to be compatible with an example $\langle x_1, x_2 \rangle$ if $h_1(x_1) = h_2(x_2)$ and incompatible otherwise; $err_{unl}(h)$ is then the probability mass of examples on which h_1 and h_2 disagree.

As with the class $\mathcal{H}$, one can either assume that the target is fully compatible (i.e., $err_{unl}(f^*) = 0$) or instead aim to do well as a function of how compatible the target is. The case that we assume $f^* \in \mathcal{H}$ and $err_{unl}(f^*) = 0$ is termed the "doubly realizable case". The concept class $\mathcal{H}$ and compatibility notion χ are both viewed as *known*.

Intuition: In this framework, the way that unlabeled data helps in learning can be intuitively described as follows. Suppose one is given a concept class $\mathcal{H}$ (such as linear separators) and a compatibility notion χ (such as penalizing h for points within distance γ of the decision boundary). Suppose also that one believes $f^* \in \mathcal{H}$ (or at least is close) and that $err_{unl}(f^*) = 0$ (or at least is small). Then, unlabeled data can help by allowing one to estimate the *unlabeled error rate* of all $h \in \mathcal{H}$, thereby in principle reducing the search space from $\mathcal{H}$ (all linear separators) down to just the subset of $\mathcal{H}$ that is highly compatible with $\mathcal{D}$. The key challenge is how this can be done efficiently (in theory, in practice, or both) for natural notions of compatibility, as well as identifying types of compatibility that data in important problems can be expected to satisfy.

A theorem: The following is a semi-supervised analog of our basic Occam's razor Theorem 6.4. First, fix some set of functions $\mathcal{H}$ and compatibility notion χ. Given a labeled sample L, define $\widehat{err}(h)$ to be the fraction of mistakes of h on L. Given an unlabeled sample U, define $\chi(h, U) = \mathbf{E}_{x\sim U}[\chi(h, x)]$ and define $\widehat{err}_{unl}(h) = 1 - \chi(h, U)$. That is, $\widehat{err}(h)$ and $\widehat{err}_{unl}(h)$ are the empirical error rate and unlabeled error rate of h, respectively. Finally, given $\alpha > 0$, define $\mathcal{H}_{\mathcal{D},\chi}(\alpha)$ to be the set of functions $f \in \mathcal{H}$ such that $err_{unl}(f) \leq \alpha$.

Theorem 6.28 *If $f^* \in \mathcal{H}$ then with probability at least $1 - \delta$, for labeled set L and unlabeled set U drawn from $\mathcal{D}$, the $h \in \mathcal{H}$ that optimizes $\widehat{err}_{unl}(h)$ subject to $\widehat{err}(h) = 0$ will have $err_{\mathcal{D}}(h) \leq \epsilon$ for $|U| \geq \frac{2}{\epsilon^2}\left[\ln|\mathcal{H}| + \ln\frac{4}{\delta}\right]$, and $|L| \geq \frac{1}{\epsilon}\left[\ln|\mathcal{H}_{\mathcal{D},\chi}(err_{unl}(f^*) + 2\epsilon)| + \ln\frac{2}{\delta}\right]$. Equivalently, for $|U|$ satisfying this bound, for any $|L|$, whp the $h \in \mathcal{H}$ that minimizes $\widehat{err}_{unl}(h)$ subject to $\widehat{err}(h) = 0$ has $err_{\mathcal{D}}(h) \leq \frac{1}{|L|}\left[\ln|\mathcal{H}_{\mathcal{D},\chi}(err_{unl}(f^*) + 2\epsilon)| + \ln\frac{2}{\delta}\right]$.*

Proof: By Hoeffding bounds, $|U|$ is sufficiently large so that with probability at least $1 - \delta/2$, all $h \in \mathcal{H}$ have $|\widehat{err}_{unl}(h) - err_{unl}(h)| \leq \epsilon$. Thus we have:

$$\{f \in \mathcal{H} : \widehat{err}_{unl}(f) \leq err_{unl}(f^*) + \epsilon\} \subseteq \mathcal{H}_{\mathcal{D},\chi}(err_{unl}(f^*) + 2\epsilon).$$

The given bound on $|L|$ is sufficient so that with probability at least $1 - \delta$, all $h \in \mathcal{H}$ with $\widehat{err}(h) = 0$ and $\widehat{err}_{unl}(h) \leq err_{unl}(f^*) + \epsilon$ have $err_{\mathcal{D}}(h) \leq \epsilon$; furthermore, $\widehat{err}_{unl}(f^*) \leq err_{unl}(f^*) + \epsilon$, so such a function h exists. Therefore, with probability at least $1 - \delta$, the $h \in \mathcal{H}$ that optimizes $\widehat{err}_{unl}(h)$ subject to $\widehat{err}(h) = 0$ has $err_{\mathcal{D}}(h) \leq \epsilon$, as desired. ■

One can view Theorem 6.28 as bounding the number of labeled examples needed to learn well as a function of the "helpfulness" of the distribution $\mathcal{D}$ with respect to χ. Namely, a helpful distribution is one in which $\mathcal{H}_{\mathcal{D},\chi}(\alpha)$ is small for α slightly larger than the compatibility of the true target function, so we do not need much labeled data to identify a good function among those in $\mathcal{H}_{\mathcal{D},\chi}(\alpha)$. For more information on semi-supervised learning, see [**?**, **?**, **?**, **?**, **?**].

6.10.2 Active learning

Active learning refers to algorithms that take an active role in the selection of which examples are labeled. The algorithm is given an initial unlabeled set U of data points drawn from distribution $\mathcal{D}$ and then interactively requests for the labels of a small number of these examples. The aim is to reach a desired error rate ϵ using much fewer labels than would be needed by just labeling random examples (i.e., passive learning).

As a simple example, suppose that data consists of points on the real line and $\mathcal{H} = \{f_a : f_a(x) = 1 \text{ iff } x \geq a\}$ for $a \in R$. That is, $\mathcal{H}$ is the set of all threshold functions on the line. It is not hard to show (see Exercise 6.1) that a random labeled sample of size $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ is sufficient to ensure that with probability $\geq 1-\delta$, any consistent threshold a' has error at most ϵ. Moreover, it is not hard to show that $\Omega(\frac{1}{\epsilon})$ random examples are necessary for passive learning. However, with active learning we can achieve error ϵ using only $O(\log(\frac{1}{\epsilon}) + \log\log(\frac{1}{\delta}))$ labels. Specifically, first draw an unlabeled sample U of size $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$. Then query the leftmost and rightmost points: if these are both negative then output $a' = \infty$, and if these are both positive then output $a' = -\infty$. Otherwise (the leftmost is negative and the rightmost is positive), perform binary search to find two adjacent examples x, x' such that x is negative and x' is positive, and output $a' = (x+x')/2$. This threshold a' is consistent with the labels on the entire set U, and so by the above argument, has error $\leq \epsilon$ with probability $\geq 1-\delta$.

The agnostic case, where the target need not belong in the given class $\mathcal{H}$ is quite a bit more subtle, and addressed in a quite general way in the "A^2" Agnostic Active learning algorithm [**?**]. For more information on active learning, see [**?**, **?**].

6.10.3 Multi-task learning

In this chapter we have focused on scenarios where our goal is to learn a single target function f^*. However, there are also scenarios where one would like to learn *multiple* target functions $f_1^*, f_2^*, \ldots, f_n^*$. If these functions are related in some way, then one could hope to do so with less data per function than one would need to learn each function separately. This is the idea of *multi-task learning.*

One natural example is object recognition. Given an image $\mathbf{x}$, $f_1^*(\mathbf{x})$ might be 1 if $\mathbf{x}$ is a coffee cup and 0 otherwise; $f_2^*(\mathbf{x})$ might be 1 if $\mathbf{x}$ is a pencil and 0 otherwise; $f_3^*(\mathbf{x})$ might be 1 if $\mathbf{x}$ is a laptop and 0 otherwise. These recognition tasks are related in that image features that are good for one task are likely to be helpful for the others as well. Thus, one approach to multi-task learning is to try to learn a common representation under which each of the target functions can be described as a simple function. Another natural example is personalization. Consider a speech recognition system with n different users. In this case there are n target tasks (recognizing the speech of each user) that are clearly related to each other. Some good references for multi-task learning are [**?**, **?**].

6.10.4 Deep learning

Deep learning, or *deep neural networks*, refers to training many-layered networks of nonlinear computational units. The input to the network is an example $\mathbf{x} \in R^d$. The first layer of the network transforms the example into a new vector $f_1(\mathbf{x})$. Then the second layer transforms $f_1(\mathbf{x})$ into a new vector $f_2(f_1(\mathbf{x}))$, and so on. Finally, the last kth layer outputs the final (typically scalar) prediction $f_k(f_{k-1}(\ldots(f_1(\mathbf{x}))))$. The motivation for deep learning is that often we are interested in data, such as images, that are given to us in terms of very low-level features, such as pixel intensity values. Yet our goal is to achieve some higher-level understanding of the example, such as what objects are in the image and what are they doing. To do so, it is natural that we should first convert the given low-level representation into one of higher-level features. That is what the layers of the network aim to do. Deep learning is also motivated by multi-task learning, with the idea that a good higher-level representation of data should be useful for a wide range of tasks. Indeed, a common use of deep learning for multi-task learning is to share all but the final level of the network across tasks.

To be concrete, a typical architecture of a deep neural network, as discussed in Section 6.3.2, is for each node in each layer of the network to compute a weighted soft-threshold function of the outputs of the previous layer. That is, node j in layer i would contain a weight vector $\mathbf{w}_{ij}$, and then compute the value

$$f_{ij} = \tanh(\mathbf{w}_{ij} \cdot \mathbf{z}), \quad \text{where } \mathbf{z} = f_{i-1}(\ldots(f_1(\mathbf{x}))).$$

The vector of all these outputs over the nodes j in layer i then gives the output of function f_i. Since these functions are differentiable, these weights can then be trained using stochastic gradient descent, as described in Section 6.3.2.

One difficulty in using stochastic gradient descent to train a deep neural network from scratch is that the optimization can get stuck in one of many suboptimal local minima. This is a problem both theoretically (the optimization problem is NP-complete) and can be a problem in practice as well. To address this issue, a popular approach is to perform *pre-training* of the lower-level layers on unlabeled data using the idea of *auto-encoding*.

An auto-encoder is a pair of mappings (f_E, f_D), where f_E is the encoder and f_D is the decoder, such that the combined function $f_D(f_E(\mathbf{x})) \approx \mathbf{x}$ for most $\mathbf{x}$ in the training data. Of course it is trivial to produce an auto-encoder by having both f_E and f_D be the identity function, but the goal is to do so using a function f_E that produces a "simpler" representation of $\mathbf{x}$ than the original input representation. For example, if data lies on, or close to, a k-dimensional subspace of the input space R^d for $k < d$, then SVD would provide a representation of each example in terms of its coordinates along the axes of the singular vectors. This would be simpler than the original representation in that $f_E(\mathbf{x})$ is only a k-dimensional vector. The function f_D would then add together the given multiples of the singular vectors to reconstruct $\tilde{\mathbf{x}}$, the projection of $\mathbf{x}$ into the span of the singular vectors.

In practice, a popular approach is to use a sparse overcomplete representation, where $f_E(\mathbf{x})$ has dimension $k \geq d$, but $\mathbf{x}$ is mapped to a sparse vector in this representation. As an extreme case, one can consider an algorithm that identifies k cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$ for the data and then maps each $\mathbf{x}$ to its nearest cluster center. This would correspond to a function f_E that maps each $\mathbf{x}$ to a unit coordinate vector $\mathbf{e}_i$ (where i is the index of the closest cluster center to $\mathbf{x}$) and then a decoder f_D that just outputs the associated cluster center. Sparsity-based auto-encoders can be viewed as an extension of this approach.

Sparse overcomplete autoencoders have been shown in practice to provide a significant benefit for deep learning. The idea is that one performs several levels of autoencoding (just using the encoding portion f_E), and then either just performs supervised learning at the top levels, or else uses these as a starting point for stochastic gradient descent.

For more information about deep learning, see [**?**].[19]

6.11 Bibliographic Notes

[TO BE FILLED IN]

[19] See also the tutorials: `http://deeplearning.net/tutorial/deeplearning.pdf` and `http://deeplearning.stanford.edu/tutorial/`.

6.12 Exercises

Exercise 6.1 (Section 6.2; easy) *Consider the instance space $\mathcal{X} = R$, and the class of functions $\mathcal{H} = \{f_a : f_a(x) = 1$ iff $x \geq a\}$ for $a \in R$. That is, $\mathcal{H}$ is the set of all threshold functions on the line. Prove that for any distribution $\mathcal{D}$, a sample S of size $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ is sufficient to ensure that with probability $\geq 1 - \delta$, any $f_{a'}$ such that $err_S(f_{a'}) = 0$ has $err_{\mathcal{D}}(f_{a'}) \leq \epsilon$.*

Exercise 6.2 (Perceptron; Section 6.3.1, 6.4.3; easy) *Consider running the Perceptron algorithm in the online model on some sequence of examples S. Let S' be the same set of examples as S but presented in a different order. Does the Perceptron algorithm necessarily make the same number of mistakes on S as it does on S'? If so, why? If not, show such an S and S' (consisting of the same set of examples in a different order) where the Perceptron algorithm makes a different number of mistakes on S' than it does on S.*

Exercise 6.3 (representation and linear separators; easy) *Show that any disjunction (see Section 6.2.1) over $\{0,1\}^d$ can be represented as a linear separator. Show that moreover the margin of separation is $\Omega(1/\sqrt{d})$.*

Exercise 6.4 (Linear separators; easy) *Show that the parity function on $d \geq 2$ Boolean variables cannot be represented by a linear threshold function. The parity function is 1 if and only if an odd number of inputs is 1.*

Exercise 6.5 (Perceptron; Section 6.3, 6.4.3) *We know the Perceptron algorithm makes at most $1/\gamma^2$ mistakes on any sequence of examples that is separable by margin γ (we assume all examples are normalized to have length 1). However, it need not find a separator of large margin. If we also want to find a separator of large margin, a natural alternative is to update on any example $\mathbf{x}$ such that $f^*(\mathbf{x})(\mathbf{w}\cdot\mathbf{x}) < 1$; this is called the* margin perceptron *algorithm.*

1. *Argue why margin perceptron is equivalent to running stochastic gradient descent on the class of linear predictors ($f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}$) using hinge loss as the loss function and using $\lambda_t = 1$.*

2. *Prove that on any sequence of examples that are separable by margin γ, this algorithm will make at most $3/\gamma^2$ updates.*

3. *In part 2 you probably proved that each update increases $|\mathbf{w}|^2$ by at most 3. Use this (and your result from part 2) to conclude that if you have a dataset S that is separable by margin γ, and cycle through the data until the margin perceptron algorithm makes no more updates, that it will find a separator of margin at least $\gamma/3$.*

Exercise 6.6 (Decision trees, regularization; Section 6.2) *Pruning a decision tree: Let S be a labeled sample drawn iid from some distribution $\mathcal{D}$ over $\{0,1\}^n$, and suppose we have used S to create some decision tree T. However, the tree T is large, and we are* *concerned we might be overfitting. Give a polynomial-time algorithm for* pruning *T that finds the pruning h of T that optimizes the right-hand-side of Corollary 6.8, i.e., that for a given $\delta > 0$ minimizes:*

$$err_S(h) + \sqrt{\frac{\text{size}(h)\ln(4) + \ln(2/\delta)}{2|S|}}.$$

To discuss this, we need to define what we mean by a "pruning" of T and what we mean by the "size" of h. A pruning h of T is a tree in which some internal nodes of T have been turned into leaves, labeled "+" or "−" depending on whether the majority of examples in S that reach that node are positive or negative. Let size$(h) = L(h)\log(n)$ where $L(h)$ is the number of leaves in h.

Hint #1: it is sufficient, for each integer $L = 1, 2, \ldots, L(T)$, to find the pruning of T with L leaves of lowest empirical error on S, that is, $h_L = \text{argmin}_{h:L(h)=L} err_S(h)$. Then you can just plug them all into the displayed formula above and pick the best one.

Hint #2: use dynamic programming.

Exercise 6.7 (Decision trees, sleeping experts; Sections 6.2, 6.8) *"Pruning" a Decision Tree Online via Sleeping Experts: Suppose that, as in the above problem, we are given a decision tree T, but now we are faced with a sequence of examples that arrive online. One interesting way we can make predictions is as follows. For each node v of T (internal node or leaf) create two sleeping experts: one that predicts positive on any example that reaches v and one that predicts negative on any example that reaches v. So, the total number of sleeping experts is $O(L(T))$.*

1. *Say why any pruning h of T, and any assignment of $\{+, -\}$ labels to the leaves of h, corresponds to a subset of sleeping experts with the property that exactly one sleeping expert in the subset makes a prediction on any given example.*

2. *Prove that for any sequence S of examples, and any given number of leaves L, if we run the sleeping-experts algorithm using $\epsilon = \sqrt{\frac{L\log(L(T))}{|S|}}$, then the expected error rate of the algorithm on S (the total number of mistakes of the algorithm divided by $|S|$) will be at most $err_S(h_L) + O(\sqrt{\frac{L\log(L(T))}{|S|}})$, where $h_L = \text{argmin}_{h:L(h)=L} err_S(h)$ is the pruning of T with L leaves of lowest error on S.*

3. *In the above question, we assumed L was given. Explain how we can remove this assumption and achieve a bound of $\min_L \left[err_S(h_L) + O(\sqrt{\frac{L\log(L(T))}{|S|}})\right]$ by instantiating $L(T)$ copies of the above algorithm (one for each value of L) and then combining* these *algorithms using the experts algorithm (in this case, none of them will be sleeping).*

Exercise 6.8 (VC-dimension; Section 6.9) *What is the VC-dimension V of the class $\mathcal{H}$ of axis-parallel boxes in R^d? That is, $\mathcal{H} = \{h_{\mathbf{a},\mathbf{b}} : \mathbf{a}, \mathbf{b} \in R^d\}$ where $h_{\mathbf{a},\mathbf{b}}(\mathbf{x}) = 1$ if $a_i \leq x_i \leq b_i$ for all $i = 1, \ldots, d$ and $h_{\mathbf{a},\mathbf{b}}(\mathbf{x}) = -1$ otherwise.*

1. *Prove that the VC-dimension is at least your chosen V by giving a set of V points that is shattered by the class (and explaining why it is shattered).*

2. *Prove that the VC-dimension is at most your chosen V by proving that no set of $V + 1$ points can be shattered.*

Exercise 6.9 (VC-dimension, Perceptron, and Margins; Sections 6.4.3, 6.9) *Say that a set of points S is* shattered by linear separators of margin γ *if every labeling of the points in S is achievable by a linear separator of margin at least γ. Prove that no set of $1/\gamma^2 + 1$ points in the unit ball is shattered by linear separators of margin γ.*

Hint: think about the Perceptron algorithm and try a proof by contradiction.

Exercise 6.10 ***(Linear separators)*** *Suppose the instance space $\mathcal{X}$ is $\{0, 1\}^d$ and consider the target function f^* that labels an example $\mathbf{x}$ as positive if the least index i for which $x_i = 1$ is odd, else labels $\mathbf{x}$ as negative. In other words, $f^*(\mathbf{x})$ = "if $x_1 = 1$ then positive else if $x_2 = 1$ then negative else if $x_3 = 1$ then positive else ... else negative". Show that the rule can be represented by a linear threshold function.*

Exercise 6.11 ***(Linear separators; harder)*** *Prove that for the problem of Exercise 6.10, we cannot have a linear separator with margin at least $1/f(d)$ where $f(d)$ is bounded above by a polynomial function of d.*

Exercise 6.12 ***VC-dimension*** *Prove that the VC-dimension of circles in the plane is three.*

Exercise 6.13 ***VC-dimension*** *Show that the VC-dimension of arbitrary right triangles in the plane is seven.*

Exercise 6.14 ***VC-dimension*** *Prove that the VC-dimension of triangles in the plane is seven.*

Exercise 6.15 ***VC-dimension*** *Prove that the VC dimension of convex polygons in the plane is infinite.*

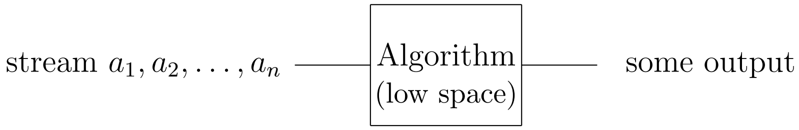


Figure 7.1: High-level representation of the streaming model

7 Algorithms for Massive Data Problems: Streaming, Sketching, and Sampling

7.1 Introduction

This chapter deals with massive data problems where the input data is too large to be stored in random access memory. One model for such problems is the streaming model, where n data items $a_1, a_2, \ldots, a_n$ arrive one at a time. For example, the a_i might be IP addresses being observed by a router on the Internet. The goal is for our algorithm to compute some statistics, property, or summary of these data items without using too much memory—much less than n. More specifically, we assume each a_i itself is a b-bit quantity where b is not too large; for example, each a_i might be an integer in $\{1, \ldots, m\}$ where $m = 2^b$. Our goal will be to produce some desired output using space polynomial in b and $\log n$; see Figure 7.1.

For example, a very easy problem to solve in the streaming model is to compute the sum of all the a_i. If each a_i is an integer between 1 and $m = 2^b$, then the sum of all the a_i is an integer between 1 and mn and so the number of bits of memory needed to maintain the sum is $O(b + \log n)$. A harder problem, which we will discuss shortly, is computing the number of *distinct* numbers in the input sequence.

One natural approach for tackling a number of problems in the streaming model is to perform random sampling of the input "on the fly". To introduce the basic flavor of sampling on the fly, consider the following problem. Suppose that each a_i is a pair (w_i, ℓ_i), and our goal is to select one ℓ_i with probability proportional to w_i. For example, in the (sleeping) experts problem considered in Chapter 6, w_i could be the weight of expert i and ℓ_i could be its prediction. We can solve this problem as follows. We begin with $W = w_1$ and $\ell = \ell_1$; in general, W will be the sum of all weights w_j seen so far and ℓ will be (by induction) correctly distributed among the previous ℓ_j. Now, given a new pair (w_i, ℓ_i) we set $\ell = \ell_i$ with probability $\frac{w_i}{W+w_i}$. It is clear that ℓ_i is chosen with the correct probability by definition. In addition, since each previous ℓ_j was chosen with probability $\frac{w_j}{W}$, the new probability that $\ell = \ell_j$ is $\frac{w_j}{W}(1 - \frac{w_i}{W+w_i}) = \frac{w_j}{W+w_i}$ as desired. Finally, we update the sum: $W \leftarrow W + w_i$.

7.2 Frequency Moments of Data Streams

An important class of problems concerns the frequency moments of data streams. As mentioned above, a data stream $a_1, a_2, \ldots, a_n$ of length n consists of symbols a_i from an alphabet of m possible symbols which for convenience we denote as $\{1, 2, \ldots, m\}$. Throughout this section, n, m, and a_i will have these meanings and s (for symbol) will denote a generic element of $\{1, 2, \ldots, m\}$. The frequency f_s of the symbol s is the number of occurrences of s in the stream. For a nonnegative integer p, the p^{th} frequency moment of the stream is

$$\sum_{s=1}^{m} (f_s)^p.$$

Note that the $p = 0$ frequency moment corresponds to the number of distinct symbols occurring in the stream. The first frequency moment is just n, the length of the string. The second frequency moment, $\sum_s f_s^2$, is useful in computing the variance of the stream, i.e., the average squared difference from the average frequency:

$$\frac{1}{m}\sum_{s=1}^{m}\left(f_s - \frac{n}{m}\right)^2 = \frac{1}{m}\sum_{s=1}^{m}\left(f_s^2 - 2\frac{n}{m}f_s + \left(\frac{n}{m}\right)^2\right) = \left(\frac{1}{m}\sum_{s=1}^{m} f_s^2\right) - \frac{n^2}{m^2}$$

In the limit as p becomes large, $\left(\sum\limits_{s=1}^{m} f_s^p\right)^{1/p}$ is the frequency of the most frequent element(s).

We will describe sampling based algorithms to compute these quantities for streaming data shortly. But first a note on the motivation for these various problems. The identity and frequency of the the most frequent item or more generally, items whose frequency exceeds a fraction of n, is clearly important in many applications. If the items are packets on a network with source and destination addresses, the high frequency items identify the heavy bandwidth users. If the data is purchase records in a supermarket, the high frequency items are the best-selling items. Determining the number of distinct symbols is the abstract version of determining such things as the number of accounts, web users, or credit card holders. The second moment and variance are useful in networking as well as in database and other applications. Large amounts of network log data are generated by routers that can record the source address, destination address, and the number of packets for all the messages passing through them. This massive data cannot be easily sorted or aggregated into totals for each source/destination. But it is important to know if some popular source-destination pairs have a lot of traffic for which the variance is one natural measure.

7.2.1 Number of Distinct Elements in a Data Stream

Consider a sequence $a_1, a_2, \ldots, a_n$ of n elements, each a_i an integer in the range 1 to m where n and m are very large. Suppose we wish to determine the number of distinct a_i in the sequence. Each a_i might represent a credit card number extracted from a sequence of credit card transactions and we wish to determine how many distinct credit card accounts there are. Note that this is easy to do in $O(m)$ space by just storing a bit-vector that records which symbols have been seen so far and which have not. It is also easy to do in $O(n \log m)$ space by storing a list of all distinct symbols that have been seen. However, our goal is to use space logarithmic in m and n. We first show that this is impossible using exact deterministic algorithms: we will show that any deterministic algorithm that determines the number of distinct elements exactly must use at least m bits of memory on some input sequence of length $O(m)$. We then will show how we can get around this problem using randomization and approximation.

Lower bound on memory for exact deterministic algorithm

We show that any exact deterministic algorithm must use at least m bits of memory on some sequence of length $m+1$. Suppose we have seen the first m symbols, and suppose for sake of contradiction that our algorithm uses less than m bits of memory on all such sequences. There are $2^m - 1$ possible subsets of $\{1, 2, \ldots, m\}$ that the sequence could contain and yet only 2^{m-1} possible states of our algorithm's memory. Therefore there must be two different subsets S_1, S_2 that lead to the same memory state. If S_1 and S_2 are of different sizes, then clearly this implies an error for one of the input sequences. On the other hand, if they are the same size, then if the next symbol is in $S_1 \setminus S_2$, the algorithm will give the same answer in both cases and therefore must give an incorrect answer on at least one of them.

Algorithm for the Number of distinct elements

Intuition: To beat the above lower bound, we will look at *approximating* the number of distinct elements—our algorithm will produce a number that is within a constant factor of the correct answer—and we will use *randomization*, allowing a small probability of failure. First, the idea: suppose the set S of distinct elements was itself chosen uniformly at random from $\{1, \ldots, m\}$. Let *min* denote the minimum element in S. What is the expected value of *min*? If there was 1 distinct element, then its expected value would be roughly $\frac{m}{2}$. If there were 2 distinct elements, the expected value of the minimum would be roughly $\frac{m}{3}$. More generally, for a random set S, the expected value of the minimum is approximately $\frac{m}{|S|+1}$. See Figure 7.2. Solving $min = \frac{m}{|S|+1}$ yields $|S| = \frac{m}{min} - 1$. This suggests keeping track of the minimum element, which can be done in $O(\log m)$ space, and then using this equation to give us an estimate of $|S|$.

Converting the intuition into an algorithm via hashing: Of course, in general the set S might not have been chosen uniformly at random. For instance, if the elements of S were obtained by selecting the $|S|$ smallest elements of $\{1, 2, \ldots, m\}$, the above technique would give a very bad answer. However, we can convert our intuition into an algorithm that works well with high probability on *every* sequence via hashing. Specifically, we will

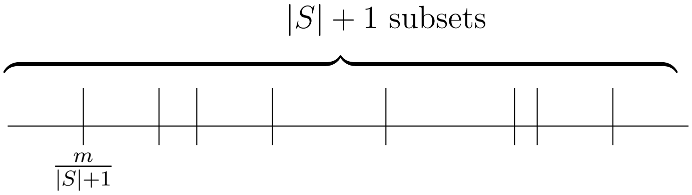


Figure 7.2: Estimating the size of S from the minimum element in S which has value approximately $\frac{m}{|S|+1}$. The elements of S partition the set $\{1, 2, \ldots, m\}$ into $|S|+1$ subsets each of size approximately $\frac{m}{|S|+1}$.

use a hash function h where

$$h : \{1, 2, \ldots, m\} \to \{0, 1, 2, \ldots, M-1\},$$

and then instead of keeping track of the minimum *element* $a_i \in S$, we will keep track of the minimum *hash value*. The question now is: what properties of a hash function do we need? Since we need to store h, we cannot use a totally random mapping since that would take too many bits. Luckily, we can do well with just a pairwise independent hash function which can be stored much more compactly.

We recall the formal definition of 2-way independence below. But first recall that a hash function is always chosen at random from a family of hash functions and phrases like "probability of collision" refer to the probability in the choice of hash function.

2-Universal (aka Pairwise Independent) Hash Functions

A set of hash functions

$$H = \{h \mid h : \{1, 2, \ldots, m\} \to \{0, 1, 2, \ldots, M-1\}\}$$

is *2-universal* or *pairwise independent* if for all x and y in $\{1, 2, \ldots, m\}$, $x \neq y$, and for all z and w in $\{0, 1, 2, \ldots, M-1\}$

$$\text{Prob}_{h\sim H}\big(h(x) = z \text{ and } h(y) = w\big) = \tfrac{1}{M^2}$$

for a randomly chosen h. The concept of a 2-universal family of hash functions is that given x, $h(x)$ is equally likely to be any element of $\{0, 1, 2, \ldots, M-1\}$ (which can be seen by summing the above probability over all M values of w) and for $x \neq y$, $h(x)$ and $h(y)$ are independent.

We now give an example of a 2-universal family of hash functions. Let $M > m$ be a prime. For each pair of integers a and b in the range $[0, M-1]$, define a hash function

$$h_{ab}(x) = ax + b \pmod{M}$$

To store the hash function h_{ab}, store the two integers a and b. This requires only $O(\log M)$ space. To see that the family is 2-universal note that $h(x) = z$ and $h(y) = w$ if and only if

$$\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} z \\ w \end{pmatrix} \pmod{M}$$

If $x \neq y$, the matrix $A = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}$ is invertible modulo M: here, we are using the primality of M to ensure that inverses of elements exist in Z_M^* and we are using the fact that $M > m$ to ensure that if $x \neq y$ then $x \neq y \pmod{M}$. This means that $\binom{a}{b} = A^{-1}\binom{z}{w} \pmod{M}$. Since each $\binom{a}{b}$ yields some $\binom{z}{w}$ and each $\binom{z}{w}$ solves to a unique $\binom{a}{b}$, we have a 1-1 correspondence. Thus, for a and b chosen uniformly at random, the probability of the equation holding is exactly $\frac{1}{M^2}$.

Analysis of distinct element counting algorithm

Let $b_1, b_2, \ldots, b_d$ be the distinct values that appear in the input. Then the set $S = \{h(b_1), h(b_2), \ldots, h(b_d)\}$ is a set of d random and 2-way independent values from the set $\{0, 1, 2, \ldots, M-1\}$. We now show that $\frac{M}{\min}$ is a good estimate for d, the number of distinct elements in the input, where min=min(S).

Lemma 7.1 *With probability at least* $\frac{2}{3} - \frac{d}{M}$*, we have* $\frac{d}{6} \leq \frac{M}{\min} \leq 6d$*, where min is the smallest element of S.*

Proof: First, we show that $\text{Prob}\left(\frac{M}{\min} > 6d\right) < \frac{1}{6} + \frac{d}{M}$. This part does not require pairwise independence.

$$\begin{aligned}\text{Prob}\left(\frac{M}{\min} > 6d\right) &= \text{Prob}\left(\min < \frac{M}{6d}\right) = \text{Prob}\left(\exists k,\ h(b_k) < \frac{M}{6d}\right) \\ &\leq \sum_{i=1}^{d} \text{Prob}\left(h(b_i) < \frac{M}{6d}\right) \leq d\left(\frac{\left\lceil \frac{M}{6d} \right\rceil}{M}\right) \leq d\left(\frac{1}{6d} + \frac{1}{M}\right) \leq \frac{1}{6} + \frac{d}{M}.\end{aligned}$$

Next, we show that $\text{Prob}\left(\frac{M}{\min} < \frac{d}{6}\right) < \frac{1}{6}$. This part will use pairwise independence. First, we can write $\text{Prob}\left(\frac{M}{\min} < \frac{d}{6}\right) = \text{Prob}\left[\min > \frac{6M}{d}\right] = \text{Prob}\left(\forall k,\ h(b_k) > \frac{6M}{d}\right)$. For $i = 1, 2, \ldots, d$ define the indicator variable $y_i = \begin{cases} 0 & \text{if } h(b_i) > \frac{6M}{d} \\ 1 & \text{otherwise} \end{cases}$ and let $y = \sum_{i=1}^{d} y_i$. We want to show that with good probability, we *do* see some hash value in $[0, \frac{6M}{d}]$, i.e., that $\text{Prob}(y = 0)$ is small. Now $\text{Prob}(y_i = 1) \geq \frac{6}{d}$, $E(y_i) \geq \frac{6}{d}$, and $E(y) \geq 6$. For 2-way independent random variables, the variance of their sum is the sum of their variances. So $\text{Var}(y) = d\text{Var}(y_1)$. Further, it is easy to see since y_1 is 0 or 1 that $\text{Var}(y_1) = E\left[(y_1 - E(y_1))^2\right] = E(y_1^2) - E^2(y_1) = E(y_1) - E^2(y_1) \leq E(y_1)$. Thus $\text{Var}(y) \leq E(y)$.

Now by the Chebyshev inequality,

$$\begin{aligned}\text{Prob}\left(\frac{M}{\min} < \frac{d}{6}\right) &= \text{Prob}\left(\min > \tfrac{6M}{d}\right) = \text{Prob}\left(\forall k\ h\left(b_i\right) > \frac{6M}{d}\right)\\ &= \text{Prob}\left(y = 0\right) \leq \text{Prob}\left(\left|y - E\left(y\right)\right| \geq E\left(y\right)\right)\\ &\leq \frac{\text{Var}(y)}{E^2\left(y\right)} \leq \frac{1}{E\left(y\right)} \leq \frac{1}{6}\end{aligned}$$

Since $\frac{M}{\min} > 6d$ with probability at most $\frac{1}{6} + \frac{d}{M}$ and $\frac{M}{\min} < \frac{d}{6}$ with probability at most $\frac{1}{6}$, $\frac{d}{6} \leq \frac{M}{\min} \leq 6d$ with probability at least $\frac{2}{3} - \frac{d}{M}$. ■

7.2.2 Counting the Number of Occurrences of a Given Element.

To count the number of occurrences of a given element in a stream requires at most $\log n$ space where n is the length of the stream. Clearly, for any length stream that occurs in practice, we can afford $\log n$ space. For this reason, the following material may never be used in practice, but the technique is interesting and may give insight into how to solve some other problem.

Consider a string of 0's and 1's of length n in which we wish to count the number of occurrences of 1's. Clearly if we had $\log n$ bits of memory we could keep track of the exact number of 1's. However, we can approximate the number with only $\log \log n$ bits.

Let m be the number of 1's that occur in the sequence. Keep a value k such that 2^k is approximately the number of occurrences m. Storing k requires only $\log \log n$ bits of memory. The algorithm works as follows. Start with k=0. For each occurrence of a 1, add one to k with probability $1/2^k$. At the end of the string, the quantity $2^k - 1$ is the estimate of m. To obtain a coin that comes down heads with probability $1/2^k$, flip a fair coin, one that comes down heads with probability ½, k times and report heads if the fair coin comes down heads in all k flips.

Given k, on average it will take 2^k ones before k is incremented. Thus, the expected number of 1's to produce the current value of k is $1 + 2 + 4 + \cdots + 2^{k-1} = 2^k - 1$.

7.2.3 Counting Frequent Elements

The Majority and Frequent Algorithms

First consider the very simple problem of n people voting. There are m candidates, $\{1, 2, \ldots, m\}$. We want to determine if one candidate gets a majority vote and if so who. Formally, we are given a stream of integers $a_1, a_2, \ldots, a_n$, each a_i belonging to $\{1, 2, \ldots, m\}$, and want to determine whether there is some $s \in \{1, 2, \ldots, m\}$ which occurs more than $n/2$ times and if so which s. It is easy to see that to solve the problem exactly on read-once streaming data with a deterministic algorithm, requires $\Omega(\min(n, m))$

space. Suppose n is even and the last $n/2$ items are identical. Suppose also that after reading the first $n/2$ items, there are two different sets of elements that result in the same contents of our memory. In that case, a mistake would occur if the second half of the stream consists solely of an element that is in one set, but not in the other. If $n/2 \geq m$ then there are $2^m - 1$ possible subsets that the first $n/2$ elements could comprise; else there are $\sum_{i=1}^{n/2} \binom{m}{i}$ subsets. By the above argument, the number of bits of memory must be at least the base-2 logarithm of the number of subsets, which is $\Omega(\min(m, n))$.

Surprisingly, we can bypass the above lower bound by just slightly weakening our goal. Let us again require that if some element appears more than $n/2$ times, then we must output it. But now, let us say that if no element appears more than $n/2$ times, then our algorithm may output whatever it wants, rather than requiring that it output “no”. That is, there may be “false positives”, but no “false negatives”.

Majority Algorithm

> Store a_1 and initialize a counter to one. For each subsequent a_i, if a_i is the same as the currently stored item, increment the counter by one. If it differs, decrement the counter by one provided the counter is nonzero. If the counter is zero, then store a_i and set the counter to one.

To analyze the algorithm, it is convenient to view the decrement counter step as “eliminating” two items, the new one and the one that caused the last increment in the counter. It is easy to see that if there is a majority element s, it must be stored at the end. If not, each occurrence of s was eliminated; but each such elimination also causes another item to be eliminated and so for a majority item not to be stored at the end, we must have eliminated more than n items, a contradiction.

Next we modify the above algorithm so that not just the majority, but also items with frequency above some threshold are detected. More specifically, the algorithm below will find the frequency (number of occurrences) of each element of $\{1, 2, \ldots, m\}$ to within an additive term of $\frac{n}{k+1}$. That is, for each symbol s, the algorithm will produce a value $\tilde{f}_s \in [f_s - \frac{n}{k+1}, f_s]$, where f_s is the true number of occurrences of symbol s in the sequence. It will do so using $O(k \log n + k \log m)$ space by keeping k counters instead of just one counter.

Algorithm Frequent

> Maintain a list of items being counted. Initially the list is empty. For each item, if it is the same as some item on the list, increment its counter by one. If it differs from all the items on the list, then if there are less than k items on the list, add the item to the list with its counter set to one. If there are already k items on the list decrement each of the current counters by one. Delete an element from the list if its count becomes zero.

Theorem 7.2 *At the end of Algorithm Frequent, for each $s \in \{1, 2, \ldots, m\}$, its counter on the list $\tilde{f}_s$ satisfies $\tilde{f}_s \in [f_s - \frac{n}{k+1}, f_s]$. In particular, if some s does not occur on the list, its counter is zero and the theorem asserts that $f_s \leq \frac{n}{k+1}$.*

Proof: The fact that $\tilde{f}_s \leq f_s$ is immediate. To show $\tilde{f}_s \geq f_s - \frac{n}{k+1}$, view each decrement counter step as eliminating some items. An item is eliminated if it is the current a_i being read and there are already k symbols different from it on the list in which case it and k other items are simultaneously eliminated. Thus, the elimination of each occurrence of an $s \in \{1, 2, \ldots, m\}$ is really the elimination of $k+1$ items. Thus, no more than $n/(k+1)$ occurrences of any symbol can be eliminated. Now, it is clear that if an item is not eliminated, then it must still be on the list at the end. This proves the theorem. ■

Theorem 7.2 implies that we can compute the true relative frequency, the number of occurrences divided by n, of every $s \in \{1, 2, \ldots, m\}$ to within an additive term of $\frac{n}{k+1}$.

7.2.4 The Second Moment

This section focuses on computing the second moment of a stream with symbols from $\{1, 2, \ldots, m\}$. Again, let f_s denote the number of occurrences of symbol s in the stream, and recall that the second moment of the stream is given by $\sum_{s=1}^{m} f_s^2$. To calculate the second moment, for each symbol s, $1 \leq s \leq m$, independently set a random variable x_s to ± 1 with probability 1/2. In particular, think of x_s as the output of a random hash function $h(s)$ whose range is just the two buckets $\{-1, 1\}$, where let us for now think of h as a fully independent hash function. Maintain a sum by adding x_s to the sum each time the symbol s occurs in the stream. At the end of the stream, the sum will equal $\sum_{s=1}^{m} x_s f_s$. The expected value of the sum will be zero where the expectation is over the choice of the ± 1 value for the x_s.

$$E\left(\sum_{s=1}^{m} x_s f_s\right) = 0.$$

Although the expected value of the sum is zero, its actual value is a random variable and the expected value of the square of the sum is given by

$$E\left(\sum_{s=1}^{m} x_s f_s\right)^2 = E\left(\sum_{s=1}^{m} x_s^2 f_s^2\right) + 2E\left(\sum_{s \neq t} x_s x_t f_s f_t\right) = \sum_{s=1}^{m} f_s^2,$$

The last equality follows since $E\left(x_s x_t\right) = E(x_s)E(x_t) = 0$ for $s \neq t$, where here we are using pairwise independence of the random variables. Thus

$$a = \left(\sum_{s=1}^{m} x_s f_s\right)^2$$

is an estimator of $\sum_{s=1}^{m} f_s^2$. Note that at this point we could use Markov's inequality to state, for instance, that Prob$(a \geq 3\sum_{s=1}^{m} f_s^2) \leq 1/3$, but we want to get a tighter guarantee. To do so, let us look at the second moment of a:

$$E[a^2] = E\left[\sum_{s=1}^{m} x_s f_s\right]^4 = E\left[\sum_{1\leq s,t,u,v\leq m} x_s x_t x_u x_v f_s f_t f_u f_v\right].$$

The last equality is by expansion. Let us now assume that the random variables x_s are 4-wise independent, or equivalently that we are producing them using a 4-wise independent hash function. Then, since the x_s are independent in the last sum, if any one of s, u, t, or v is distinct from the others, then the expectation of the whole term is zero. Thus, we need to deal only with terms of the form $x_s^2 x_t^2$ for $t \neq s$ and terms of the form x_s^4.

Each term in the above sum has four indices, s, t, u, v, and there are $\binom{4}{2}$ ways of choosing two indices that have the same x value. Thus,

$$\begin{aligned} E[a^2] &\leq \binom{4}{2} E\left(\sum_{s=1}^{m}\sum_{t=s+1}^{m} x_s^2 x_t^2 f_s^2 f_t^2\right) + E\left(\sum_{s=1}^{m} x_s^4 f_s^4\right) \\ &= 6\sum_{s=1}^{m}\sum_{t=s+1}^{m} f_s^2 f_t^2 + \sum_{s=1}^{m} f_s^4 \\ &\leq 3\left(\sum_{s=1}^{m} f_s^2\right)^2 \; = \; 3E[a]^2. \end{aligned}$$

Therefore, $Var(a) = E[a^2] - E[a]^2 \leq 2E[a]^2$.

Since the variance is comparable to the square of the expectation, this implies that if we repeat the process several times and take the average, we will get high accuracy with high probability. Specifically,

Theorem 7.3 *If we use $r = \frac{2}{\varepsilon^2\delta}$ independently chosen 4-way independent sets of random variables, and let X be the average of the estimates $a_1, \ldots, a_r$ produced, then*

$$Prob\left(|X - E[X]| > \varepsilon E[X]\right) < \frac{Var(X)}{\epsilon^2 E[X]} \leq \delta.$$

Proof: The proof follows from the fact that taking the average of r independent repetitions reduces variance by a factor of r, so that $Var(X) \leq \delta\varepsilon^2 E[X]$, and then applying Chebyshev's inequality. ■

What remains now is to show that we can implement the desired 4-way independent random variables using $O(\log m)$ space. We earlier gave a construction for a pairwise-independent set of hash functions; now, we need 4-wise independence, though only into a range of $\{-1, 1\}$. Below we present one such construction.

Error-Correcting codes, polynomial interpolation and limited-way independence

Consider the problem of generating a random m-vector $\mathbf{x}$ of ± 1's so that any subset of four coordinates is mutually independent. We will show that such an m-dimensional vector may be generated from a truly random "seed" of only $O(\log m)$ mutually independent bits. Thus, we need only store the $O(\log m)$ bits and can generate any of the m coordinates when needed. The first fact needed for this is that for any k, there is a finite field F with exactly 2^k elements, each of which can be represented with k bits and arithmetic operations in the field can be carried out in $O(k^2)$ time. Here, k will be the ceiling of $\log_2 m$. We also assume another basic fact about polynomial interpolation; a polynomial of degree at most three is uniquely determined by its value over any field F at four points. More precisely, for any four distinct points $a_1, a_2, a_3, a_4 \in F$ and any four possibly not distinct values $b_1, b_2, b_3, b_4 \in F$, there is a unique polynomial $f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3$ of degree at most three, so that with computations done over F, $f(a_1) = b_1, f(a_2) = b_2, f(a_3) = b_3$, and $f(a_4) = b_4$.

The definition of the pseudo-random ± 1 vector $\mathbf{x}$ with 4-way independence is simple. Choose four elements f_0, f_1, f_2, f_3 at random from F and form the polynomial $f(s) = f_0 + f_1 s + f_2 s^2 + f_3 s^3$. This polynomial represents $\mathbf{x}$ as follows. For $s = 1, 2, \ldots, m$, x_s is the leading bit of the k-bit representation of $f(s)$. Thus, the m-dimensional vector $\mathbf{x}$ requires only $O(k)$ bits where $k = \lceil \log m \rceil$.

Lemma 7.4 *The* $\mathbf{x}$ *defined above has 4-way independence.*

Proof: Assume that the elements of F are represented in binary using ± 1 instead of the traditional 0 and 1. Let s, t, u, and v be any four coordinates of $\mathbf{x}$ and let $\alpha, \beta, \gamma, \delta \in \{-1, 1\}$. There are exactly 2^{k-1} elements of F whose leading bit is α and similarly for β, γ, and δ. So, there are exactly $2^{4(k-1)}$ 4-tuples of elements b_1, b_2, b_3, $b_4 \in F$ so that the leading bit of b_1 is α, the leading bit of b_2 is β, the leading bit of b_3 is γ, and the leading bit of b_4 is δ. For each such b_1, b_2, b_3, and b_4, there is precisely one polynomial f so that $f(s) = b_1$, $f(t) = b_2$, $f(u) = b_3$, and $f(v) = b_4$. The probability that $x_s = \alpha$, $x_t = \beta$, $x_u = \gamma$, and $x_v = \delta$ is precisely

$$\frac{2^{4(k-1)}}{\text{total number of } f} = \frac{2^{4(k-1)}}{2^{4k}} = \frac{1}{16}$$

as asserted. ■

Lemma 7.4 describes how to get one vector $\mathbf{x}$ with 4-way independence. However, we need $r = O(1/\varepsilon^2)$ vectors. Also the vectors must be mutually independent. But this is easy, just choose r independent polynomials at the outset.

To implement the algorithm with low space, store only the polynomials in memory. This requires $4k = O(\log m)$ bits per polynomial for a total of $O(\frac{\log m}{\varepsilon^2})$ bits. When a symbol s in the stream is read, compute each polynomial at s to obtain the value for the corresponding value of the x_s and update the running sums. x_s is just the leading bit of the polynomial evaluated at s; this calculation is in $O(\log m)$ time. Thus, we repeatedly compute the x_s from the "seeds", namely the coefficients of the polynomials.

This idea of polynomial interpolation is also used in other contexts. Error-correcting codes is an important example. Say we wish to transmit n bits over a channel which may introduce noise. One can introduce redundancy into the transmission so that some channel errors can be corrected. A simple way to do this is to view the n bits to be transmitted as coefficients of a polynomial $f(x)$ of degree $n-1$. Now transmit f evaluated at points $1, 2, 3, \ldots, n+m$. At the receiving end, any n correct values will suffice to reconstruct the polynomial and the true message. So up to m errors can be tolerated. But even if the number of errors is at most m, it is not a simple matter to know which values are corrupted. We do not elaborate on this here.

7.3 Matrix Algorithms using sampling

We now move from the streaming model to a model where the input is stored in memory, but because the input is so large, one would like to produce a much smaller approximation to it, or perform an approximate computation on it in low space. For instance, the input might be stored in a large slow memory and we would like a small "sketch" that can be stored in smaller fast memory and yet retains the important properties of the original input. In fact, one can view a number of results from the chapter on machine learning in this way: we have a large population, and we want to take a small sample, perform some optimization on the sample, and then argue that the optimum solution on the sample will be approximately optimal over the whole population. In the chapter on machine learning, our sample consisted of independent random draws from the overall population or data distribution. Here, we will be looking at matrix algorithms, and to achieve our desired guarantees—in particular, to achieve good relative-error bounds rather than additive-error bounds—we will want to perform *non-uniform* sampling.

In fact, it turns out that sampling the rows/columns of a matrix with probabilities proportional to the length squared of the row/column is a good idea in many contexts. We present two examples here; matrix multiplication and the sketch of a matrix. In the discussion below, think of the matrices we are given as having low rank or at least having a good low-rank approximation. In particular, think of them as matrices on which SVD would provide a good approximation using just a small number of top singular vectors. Our goal will be to do nearly as well as SVD, but via a much faster procedure that uses actual rows and columns of the input matrices rather than linear combinations of them.

7.3.1 Matrix Multiplication Using Sampling

Suppose A is an $m \times n$ matrix and B is an $n \times p$ matrix and the product AB is desired. We show how to use sampling to get an approximate product faster than the traditional multiplication. Let $A\left(:,k\right)$ denote the k^{th} column of A. $A\left(:,k\right)$ is a $m \times 1$ matrix. Let $B\left(k,:\right)$ be the k^{th} row of B. $B\left(k,:\right)$ is a $1 \times n$ matrix. It is easy to see that

$$AB = \sum_{k=1}^{n} A\left(:,k\right) B\left(k,:\right).$$

Note that for each value of k, $A(:,k)B(k,:)$ is an $m \times p$ matrix each element of which is a single product of elements of A and B. An obvious use of sampling suggests itself. Sample some values for k and compute $A\left(:,k\right)B\left(k,:\right)$ for the sampled k's and use their suitably scaled sum as the estimate of AB. It turns out that nonuniform sampling probabilities are useful. Define a random variable z that takes on values in $\{1, 2, \ldots, n\}$. Let p_k denote the probability that z assumes the value k. We will solve for a good choice of probabilities later, but for now just consider the p_k as nonnegative numbers that sum to one. Define an associated random matrix variable that has value

$$X = \frac{1}{p_k} A\left(:,k\right) B\left(k,:\right) \tag{7.1}$$

with probability p_k. Let $E\left(X\right)$ denote the entry-wise expectation.

$$E\left(X\right) = \sum_{k=1}^{n} \text{Prob}(z=k) \frac{1}{p_k} A\left(:,k\right) B\left(k,:\right) = \sum_{k=1}^{n} A\left(:,k\right) B\left(k,:\right) = AB.$$

This explains the scaling by $\frac{1}{p_k}$ in X. In particular, X is a matrix-valued random variable each of whose components is correct in expectation. We will be interested in

$$E\left(||AB - X||_F^2\right).$$

This can be viewed as the variance of X, defined as the sum of the variances of all its entries.

$$\text{Var}(X) = \sum_{i=1}^{m}\sum_{j=1}^{p} \text{Var}\left(x_{ij}\right) = \sum_{ij} E\left(x_{ij}^2\right) - E\left(x_{ij}\right)^2 = \left(\sum_{ij}\sum_{k} p_k \frac{1}{p_k^2} a_{ik}^2 b_{kj}^2\right) - ||AB||_F^2.$$

We want to choose p_k to minimize this quantity, and notice that we can ignore the $||AB||_F^2$ term since it doesn't depend on the p_k's at all. We can now simplify by exchanging the order of summations to get

$$\sum_{ij}\sum_{k} p_k \frac{1}{p_k^2} a_{ik}^2 b_{kj}^2 = \sum_{k} \frac{1}{p_k} \left(\sum_{i} a_{ik}^2\right)\left(\sum_{j} b_{kj}^2\right) = \sum_{k} \frac{1}{p_k} |A\left(:,k\right)|^2 |B\left(k,:\right)|^2.$$

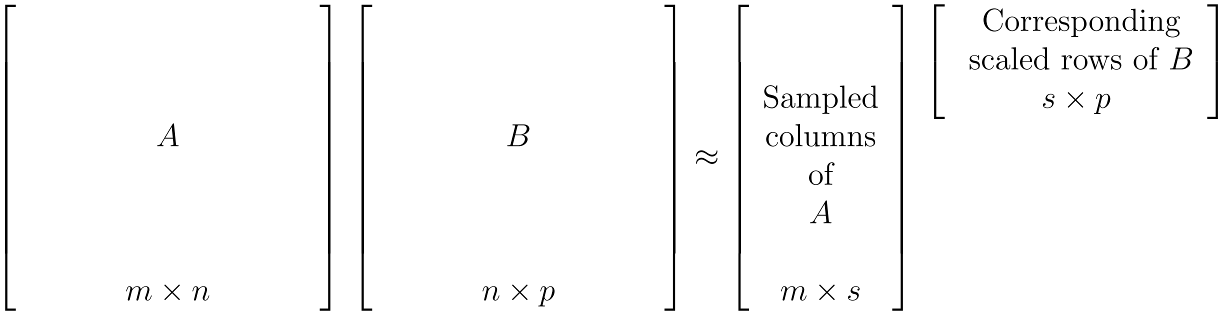


Figure 7.3: Approximate Matrix Multiplication using sampling

What is the best choice of p_k to minimize this sum? It can be seen by calculus[20] that the minimizing p_k are proportional to $|A(:,k)||B(k,:)|$. In the important special case when $B = A^T$, this means to pick columns of A with probabilities proportional to the squared length of the columns. In fact, even in the general case when B is not A^T, doing so simplifies the bounds, so we will use it. This sampling is called "length squared sampling". If p_k is proportional to $|A\,(:,k)\,|^2$, i.e, $p_k = \frac{|A(:,k)|^2}{||A||_F^2}$, then

$$E\left(||AB - X||_F^2\right) = \text{Var}(X) \le ||A||_F^2 \sum_k |B\,(k,:)\,|^2 = ||A||_F^2 ||B||_F^2.$$

To reduce the variance, we can do s independent trials. Each trial i, $i = 1, 2, \ldots, s$ yields a matrix X_i as in (7.1). We take $\frac{1}{s}\sum_{i=1}^s X_i$ as our estimate of AB. Since the variance of a sum of independent random variables is the sum of variances, the variance of $\frac{1}{s}\sum_{i=1}^s X_i$ is $\frac{1}{s}\text{Var}(X)$ and so is at most $\frac{1}{s}||A||_F^2||B||_F^2$. Let $k_1, \ldots, k_s$ be the k's chosen in each trial. Expanding this out, we get:

$$\frac{1}{s}\sum_{i=1}^s X_i = \frac{1}{s}\left(\frac{A\,(:,k_1)\,B\,(k_1,:)}{p_{k_1}} + \frac{A\,(:,k_2)\,B\,(k_2,:)}{p_{k_2}} + \cdots + \frac{A\,(:,k_s)\,B\,(k_s,:)}{p_{k_s}}\right) = C\tilde{B},$$

where, C is the $m \times s$ matrix of the chosen columns of A and $\tilde{B}$ is an $s \times p$ matrix with the corresponding rows of B scaled, namely, $\tilde{B}$ has rows $\frac{B(k_1,:)}{sp_{k_1}}, \frac{B(k_2,:)}{sp_{k_2}}, \ldots \frac{B(k_s,:)}{sp_{k_s}}$. This is represented in Figure 7.3. We summarize our discussion in Theorem 7.5.

Theorem 7.5 *Suppose A is an $m \times n$ matrix and B is an $n \times p$ matrix. The product AB can be estimated by $C\tilde{B}$, where, C is an $m \times s$ matrix consisting of s columns of A picked according to length-squared distribution and $\tilde{B}$ is the $s \times p$ matrix consisting of the corresponding rows of B scaled as above. The error is bounded by:*

$$E\left(||AB - C\tilde{B}||_F^2\right) \le \frac{||A||_F^2\,||B||_F^2}{s}.$$

[20] One can see by taking derivatives that for any set of nonnegative numbers c_k, to minimize $\sum_k \frac{c_k^2}{p_k}$, one should pick p_k proportional to c_k.

When is this a good approximation and when is it not? Let's focus on the case that $B = A^T$ so we have just one matrix to consider. If A is the identity matrix, then the guarantee is *not* very good. In this case, $||AA^T||_F^2 = n$, but the right-hand-side of the inequality is $\frac{n^2}{s}$. So we would need $s > n$ for the bound to be any better than approximating the product with the zero matrix. On the other hand, if all rows of A are identical—say, identical unit-length vectors for concreteness—then $||AA^T||_F^2 = n^2$ and the right-hand-side is still $\frac{n^2}{s}$. So, we would just need $s = \frac{1}{\epsilon}$ for the bound to be quite strong. More generally, if each row of A is a unit-length vector, so the right-hand-side is $\frac{n^2}{s}$, the quality of the bound depends on how large $||AA^T||_F^2$ is, which can be viewed as n^2 times the expected value of $(A_i \cdot A_j)^2$ when rows A_i, A_j are selected randomly from A.

7.3.2 Sketch of a Large Matrix

The main result of this section is that for any matrix, a sample of columns and rows, each picked according to length squared distribution provides a good sketch of the matrix in a formal sense that will be described shortly. Let A be an $m \times n$ matrix. Pick s columns of A according to length squared distribution. Let C be the $m \times s$ matrix containing the picked columns. Similarly, pick r rows of A according to length squared distribution on the rows of A. Let R be the $r \times n$ matrix of the picked rows.[21] From C and R, we can find a matrix U so that $A \approx CUR$. The schematic diagram is given in Figure 7.4.

The proof that this is a good approximation makes crucial use of the fact that the sampling of rows and columns is with probability proportional to the squared length. One may recall that the top k singular vectors of the SVD of A, give a similar picture; but the SVD takes more time to compute, requires all of A to be stored in RAM, and does not have the property that the rows and columns are directly from A. The last property - that the approximation involves actual rows/columns of the matrix rather than linear combinations - is called an *interpolative approximation* and is useful in many contexts. However, the SVD does yield the best 2-norm approximation. Error bounds for the approximation CUR are weaker.

We briefly touch upon two motivations for such a sketch. Suppose A is the document-term matrix of a large collection of documents. We are to "read" the collection at the outset and store a sketch so that later, when a query represented by a vector with one entry per term arrives, we can find its similarity to each document in the collection. Similarity is defined by the dot product. In Figure 7.4 it is clear that the matrix-vector product of a query with the right hand side can be done in time $O(ns + sr + rm)$ which would be linear in n and m if s and r are $O(1)$. To bound errors for this process, we need to show that the difference between A and the sketch of A has small 2-norm. Recall that the 2-norm $||A||_2$ of a matrix A is $\max\limits_{|\mathbf{x}|=1} |A\mathbf{x}|$. The fact that the sketch is an interpolative

[21] We will scale each row i of R by $\frac{1}{\sqrt{rp_i}}$ where p_i is the probability with which we picked that row, so that $R^T R$ is an approximation to $A^T A$ using Theorem 7.5. The reasons for this scaling will become clear in the proof.

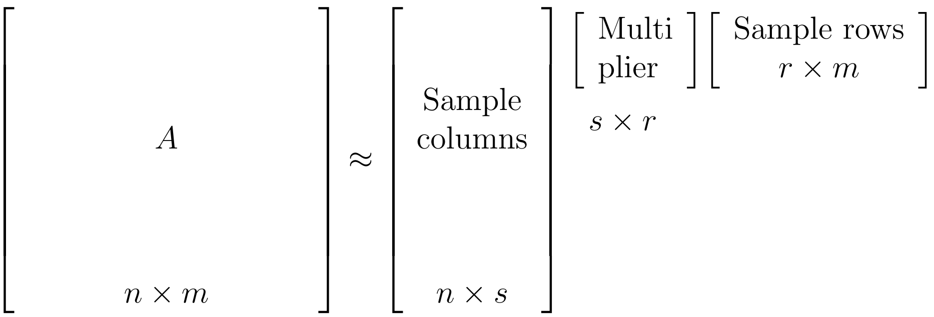


Figure 7.4: Schematic diagram of the approximation of A by a sample of s columns and r rows.

approximation means that our approximation essentially consists a subset of documents and a subset of terms, which may be thought of as a representative set of documents and terms. Additionally, if A is *sparse* in its rows and columns—each document contains only a small fraction of the terms and each term is in only a small fraction of the documents—then this property will be preserved in C and R, unlike with SVD.

A second motivation comes from recommendation systems. Here A would be a customer-product matrix whose $(i, j)^{th}$ entry is the preference of customer i for product j. The objective is to collect a few sample entries of A and based on them, get an approximation to A so that we can make future recommendations. A few sampled rows of A (all preferences of a few customers) and a few sampled columns (all customers' preferences for a few products) give a good approximation to A provided that the samples are drawn according to the length-squared distribution.

It remains now to describe how to find U from C and R. We describe this assuming RR^T is invertible. Through the rest of this section, we make the assumption that RR^T is invertible. This case will convey the essential ideas. Also, note that since r in general will be much smaller than n and m, unless the matrix A is degenerate, it is likely that the r rows in the sample R will be linearly independent giving us invertibility of RR^T.

We begin with some intuition. In particular, we first present a simpler idea that does not work, but that will then motivate the idea that does. Write A as AI, where I is the $n \times n$ identity matrix. Now, let's approximate the product AI using the algorithm of Theorem 7.5 from the last section, i.e., by sampling s columns of A according to length-squared. Then, as in the last section, write $AI \approx CW$, where, W consists of a scaled version of the s rows of I corresponding to the s columns of A that were picked. Theorem 7.5 bounds the error $||A - CW||_F^2$ by $||A||_F^2 ||I||_F^2 / s = ||A||_F^2 \frac{n}{s}$. But we would like the error to be a small fraction of $||A||_F^2$ which would require $s \geq n$, which clearly is of no use since this would pick as many or more columns than the whole of A.

This intuition, however, suggests the following idea. We assumed that RR^T is invertible. Then it is easy to see, Lemma 7.9, that $R^T(RR^T)^{-1}R$ (we denote $R^T(RR^T)^{-1}R$ by P for convenience) acts as the identity matrix on the space V spanned by the rows of R. Let's use this identity-like matrix P instead of I in the above discussion. Using the fact that R is picked according to length squared we will show the following proposition later.

Proposition 7.6 *$A \approx AP$ and the error $E\left(||A - AP||_2^2\right)$ is at most $||A||_F^2/\sqrt{r}$.*

We then use Theorem 7.5 to argue that instead of doing the multiplication AP, we can use the sampled columns of A and the corresponding rows of P. The sampled s columns of A form C. We have to take the corresponding s rows of $P = R^T(RR^T)^{-1}R$, which is the same as taking the corresponding s rows of R^T, and multiplying this by $(RR^T)^{-1}R$. It is easy to check that this leads to an expression of the form CUR. Further, by Theorem 7.5, the error is bounded by

$$E\left(||AP - CUR||_2^2\right) \le E\left(||AP - CUR||_F^2\right) \le \frac{||A||_F^2||P||_F^2}{s} \le \frac{r}{s}||A||_F^2, \tag{7.2}$$

since we will show later that:

Proposition 7.7 *$||P||_F^2 \le r$.*

Putting (7.2) and Proposition 7.6 together, and using the fact that by triangle inequality we have $||A - CUR||_2 \le ||A - AP||_2 + ||AP - CUR||_2$ which in turn implies that $||A - CUR||_2^2 \le 2||A - AP||_2^2 + 2||AP - CUR||_2^2$, we get the main result:

Theorem 7.8 *Suppose A is any $m \times n$ matrix and r and s are positive integers. Suppose C is a $m \times s$ matrix of s columns of A picked according to length squared sampling and similarly R is a matrix of r rows of A picked according to length squared sampling. Then, we can find from C, R an $s \times r$ matrix U so that*

$$E\left(||A - CUR||_2^2\right) \le ||A||_F^2\left(\frac{2}{\sqrt{r}} + \frac{2r}{s}\right).$$

Choosing $s = r/\varepsilon$ and $r = 1/\varepsilon^2$, the bound becomes $O(\varepsilon)||A||_F^2$. When is this bound meaningful? We discuss this further after first proving all the claims used in the discussion above.

Lemma 7.9 *If RR^T is invertible, then $P = R^T(RR^T)^{-1}R$ acts as the identity matrix on the row space of R. I.e., $P\mathbf{x} = \mathbf{x}$ for every vector $\mathbf{x}$ of the form $\mathbf{x} = R^T\mathbf{y}$ (this defines the row space of R). Furthermore, if $\mathbf{x}$ is orthogonal to the row space of R, then $P\mathbf{x} = \mathbf{0}$.*

Proof: For $\mathbf{x} = R^T\mathbf{y}$, $R^T(RR^T)^{-1}R\mathbf{x} = R^T(RR^T)^{-1}RR^T\mathbf{y} = R^T\mathbf{y} = \mathbf{x}$. If $\mathbf{x}$ is orthogonal to every row of R, then $R\mathbf{x} = \mathbf{0}$, so $P\mathbf{x} = \mathbf{0}$. ∎

Now we prove Proposition 7.6. First, recall that

$$||A - AP||_2^2 = \max_{\{\mathbf{x}:|\mathbf{x}|=1\}} |(A - AP)\mathbf{x}|^2.$$

Let's first suppose $\mathbf{x}$ is in the row space V of R. From Lemma 7.9 we have $P\mathbf{x} = \mathbf{x}$, so for $\mathbf{x} \in V$, we have $(A - AP)\mathbf{x} = \mathbf{0}$. Now, since every vector can be written as a sum of a vector in V plus a vector orthogonal to V, this implies that the maximum must therefore occur at some $\mathbf{x} \in V^\perp$. For such $\mathbf{x}$, by Lemma 7.9 we have $(A - AP)\mathbf{x} = A\mathbf{x}$. Thus, the question now becomes: for unit-length $\mathbf{x} \in V^\perp$, how large can $|A\mathbf{x}|^2$ be? To analyze this, we can write:

$$|A\mathbf{x}|^2 = \mathbf{x}^T A^T A\mathbf{x} = \mathbf{x}^T(A^T A - R^T R)\mathbf{x} \leq ||A^T A - R^T R||_2 |\mathbf{x}|^2 \leq ||A^T A - R^T R||_2.$$

This implies that we get $||A - AP||_2^2 \leq ||A^T A - R^T R||_2$. So, it suffices to prove that $||A^T A - R^T R||_2^2 \leq ||A||_F^4/r$ which follows directly from Theorem 7.5, since we can think of $R^T R$ as a way of estimating $A^T A$ by picking (according to length-squared distribution) columns of A^T, i.e., rows of A. This proves Proposition 7.6.

Proposition 7.7 is easy to see: Since by Lemma 7.9, P is the identity on the space V spanned by the rows of R, and $P\mathbf{x} = 0$ for $\mathbf{x}$ perpendicular to the rows of R, we have that $||P||_F^2$ is the sum of its singular values squared which is at most r as claimed.

We now briefly look at the time needed to compute U. The only involved step in computing U is to find $(RR^T)^{-1}$. But note that RR^T is an $s \times s$ matrix and since s is to much smaller than n, m, this is fast.

Understanding the bound in Theorem 7.8: Let us now aim to better understand the bound in Theorem 7.8 by considering when it is meaningful and when it is not. First, let's choose parameters $s = \Theta(1/\varepsilon^3)$ and $r = \Theta(1/\varepsilon^2)$ so that the bound becomes $E(||A - CUR||_2^2) \leq \varepsilon ||A||_F^2$. Now, recall that $||A||_F^2 = \sum_i \sigma_i^2(A)$, i.e., the sum of squares of all the singular values of A. Also, let's for convenience scale A so that $\sigma_1^2(A) = 1$. So, we have:

$$\sigma_1^2(A) = ||A||_2^2 = 1 \quad \text{and} \quad E(||A - CUR||_2^2) \leq \varepsilon \sum_i \sigma_i^2(A).$$

From this, we can get an intuitive sense of when the guarantee is good and when it is not. First, if the top k singular values of A are all $\Omega(1)$ for $k \gg m^{1/3}$, so that $\sum_i \sigma_i^2(A) \gg m^{1/3}$, then the guarantee is only meaningful when $\varepsilon = o(m^{-1/3})$, which is not interesting because it requires $s > m$. On the other hand, if say just the first few singular values of A are large and the rest are quite small—e.g, A represents a collection of points that lie very close to a low-dimensional pancake—and in particular if $\sum_i \sigma_i^2(A)$ is a constant, then to be meaningful the bound just requires ε to be a small constant. In this case, the guarantee is indeed meaningful because it implies that by just selecting a constant number of rows and columns we can provide a good 2-norm approximation to A.

7.4 Sketches of Documents

Suppose one wished to store all the web pages from the WWW. Since there are billions of web pages, one might store just a sketch of each page where a sketch is a few hundred bits that capture sufficient information to do whatever task one had in mind. A web page or a document is a sequence. We begin this section by showing how to sample a set and then how to convert the problem of sampling a sequence into a problem of sampling a set.

Consider subsets of size 1000 of the integers from 1 to 10^6. Suppose one wished to compute the resemblance of two subsets A and B by the formula

$$\text{resemblance}\,(A, B) = \tfrac{|A\cap B|}{|A\cup B|}$$

Suppose that instead of using the sets A and B, one sampled the sets and compared random subsets of size ten. How accurate would the estimate be? One way to sample would be to select ten elements uniformly at random from A and B. However, this method is unlikely to produce overlapping samples. Another way would be to select the ten smallest elements from each of A and B. If the sets A and B overlapped significantly one might expect the sets of ten smallest elements from each of A and B to also overlap. One difficulty that might arise is that the small integers might be used for some special purpose and appear in essentially all sets and thus distort the results. To overcome this potential problem, rename all elements using a random permutation.

Suppose two subsets of size 1000 overlapped by 900 elements. What would the overlap be of the 10 smallest elements from each subset? One would expect the nine smallest elements from the 900 common elements to be in each of the two subsets for an overlap of 90%. The resemblance(A, B) for the size ten sample would be 9/11=0.81.

Another method would be to select the elements equal to zero mod m for some integer m. If one samples mod m the size of the sample becomes a function of n. Sampling mod m allows us to also handle containment.

In another version of the problem one has a sequence rather than a set. Here one converts the sequence into a set by replacing the sequence by the set of all short subsequences of some length k. Corresponding to each sequence is a set of length k subsequences. If k is sufficiently large, then two sequences are highly unlikely to give rise to the same set of subsequences. Thus, we have converted the problem of sampling a sequence to that of sampling a set. Instead of storing all the subsequences, we need only store a small subset of the set of length k subsequences.

Suppose you wish to be able to determine if two web pages are minor modifications of one another or to determine if one is a fragment of the other. Extract the sequence of words occurring on the page. Then define the set of subsequences of k consecutive words from the sequence. Let $S(D)$ be the set of all subsequences of length k occurring in document D. Define resemblance of A and B by

$$\text{resemblance}\,(A, B) = \tfrac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

And define containment as

$$\text{containment}\,(A, B) = \tfrac{|S(A) \cap S(B)|}{|S(A)|}$$

Let W be a set of subsequences. Define $\min\,(W)$ to be the s smallest elements in W and define $\text{mod}\,(W)$ as the set of elements of w that are zero mod m.

Let π be a random permutation of all length k subsequences. Define $F(A)$ to be the s smallest elements of A and $V(A)$ to be the set mod m in the ordering defined by the permutation.

Then

$$\frac{F(A) \cap F(B)}{F(A) \cup F(B)}$$

and

$$\tfrac{|V(A) \cap V(B)|}{|V(A) \cup V(B)|}$$

are unbiased estimates of the resemblance of A and B. The value

$$\tfrac{|V(A) \cap V(B)|}{|V(A)|}$$

is an unbiased estimate of the containment of A in B.

7.5 Bibliography

TO DO

7.6 Exercises

Algorithms for Massive Data Problems

Exercise 7.1 *Given a stream of n positive real numbers $a_1, a_2, \ldots, a_n$, upon seeing $a_1, a_2, \ldots, a_i$ keep track of the sum $a = a_1 + a_2 + \cdots + a_i$ and a sample a_j, $j \leq i$ drawn with probability proportional to its value. On reading a_{i+1}, with probability $\frac{a_{i+1}}{a+a_{i+1}}$ replace the current sample with a_{i+1} and update a. Prove that the algorithm selects an a_i from the stream with the probability of picking a_i being proportional to its value.*

Exercise 7.2 *Given a stream of symbols $a_1, a_2, \ldots, a_n$, give an algorithm that will select one symbol uniformly at random from the stream. How much memory does your algorithm require?*

Exercise 7.3 *Give an algorithm to select an a_i from a stream of symbols $a_1, a_2, \ldots, a_n$ with probability proportional to a_i^2.*

Exercise 7.4 *How would one pick a random word from a very large book where the probability of picking a word is proportional to the number of occurrences of the word in the book?*

Exercise 7.5 *For the streaming model give an algorithm to draw s independent samples each with the probability proportional to its value. Justify that your algorithm works correctly.*

Frequency Moments of Data Streams

Number of Distinct Elements in a Data Stream

Lower bound on memory for exact deterministic algorithm

Algorithm for the Number of distinct elements

Universal Hash Functions

Exercise 7.6 *Show that for a 2-universal hash family $Prob\,(h(x) = z) = \frac{1}{M+1}$ for all $x \in \{1, 2, \ldots, m\}$ and $z \in \{0, 1, 2, \ldots, M\}$.*

Exercise 7.7 *Let p be a prime. A set of hash functions*

$$H = \{h|\,\{0, 1, \ldots, p-1\} \rightarrow \{0, 1, \ldots, p-1\}\}$$

is 3-universal if for all u,v,w,x,y, and z in $\{0, 1, \ldots, p-1\}$

$$Prob\,(h\,(x) = u,\ h\,(y) = v,\ h\,(z) = w) = \frac{1}{p^3}.$$

(a) *Is the set $\{h_{ab}(x) = ax + b \mod p | 0 \leq a, b < p\}$ of hash functions 3-universal?*

(b) *Give a 3-universal set of hash functions.*

Exercise 7.8 *Give an example of a set of hash functions that is not 2-universal.*

Analysis of distinct element counting algorithm

Counting the Number of Occurrences of a Given Element.

Exercise 7.9

(a) *What is the variance of the method in Section 7.2.2 of counting the number of occurrences of a 1 with* $\log\log n$ *memory?*

(b) *Can the algorithm be iterated to use only* $\log\log\log n$ *memory? What happens to the variance?*

Exercise 7.10 *Consider a coin that comes down heads with probability* p*. Prove that the expected number of flips before a head occurs is* $1/p$*.*

Exercise 7.11 *Randomly generate a string* $x_1x_2\cdots x_n$ *of* 10^6 *0's and 1's with probability* $1/2$ *of* x_i *being a 1. Count the number of ones in the string and also estimate the number of ones by the approximate counting algorithm. Repeat the process for p=1/4, 1/8, and 1/16. How close is the approximation?*

Counting Frequent Elements

The Majority and Frequent Algorithms

The Second Moment

Exercise 7.12 *Construct an example in which the majority algorithm gives a false positive, i.e., stores a nonmajority element at the end.*

Exercise 7.13 *Construct examples where the frequent algorithm in fact does as badly as in the theorem, i.e., it "under counts" some item by n/(k+1).*

Exercise 7.14 *Recall basic statistics on how an average of independent trials cuts down variance and complete the argument for relative error* ε *estimate of* $\sum\limits_{s=1}^{m} f_s^2$*.*

Error-Correcting codes, polynomial interpolation and limited-way independence

Exercise 7.15 *Let* F *be a field. Prove that for any four distinct points* $a_1, a_2, a_3,$ *and* a_4 *in* F *and any four (possibly not distinct) values* $b_1, b_2, b_3,$ *and* b_4 *in* F*, there is a unique polynomial* $f(x) = f_0+f_1x+f_2x^2+f_3x^3$ *of degree at most three so that* $f(a_1) = b_1,\ f(a_2) = b_2,\ f(a_3) = b_3\ f(a_4) = b_4$ *with all computations done over* F*.*

Sketch of a Large Matrix

Exercise 7.16 *Suppose we want to pick a row of a matrix at random where the probability of picking row i is proportional to the sum of squares of the entries of that row. How would we do this in the streaming model? Do not assume that the elements of the matrix are given in row order.*

(a) *Do the problem when the matrix is given in column order.*

(b) *Do the problem when the matrix is represented in sparse notation: it is just presented as a list of triples (i, j, a_{ij}), in arbitrary order.*

Matrix Multiplication Using Sampling

Exercise 7.17 *Suppose A and B are two matrices. Show that* $AB = \sum\limits_{k=1}^{n} A\,(:, k)B\,(k, :)$.

Exercise 7.18 *Generate two 100 by 100 matrices A and B with integer values between 1 and 100. Compute the product AB both directly and by sampling. Plot the difference in L_2 norm between the results as a function of the number of samples. In generating the matrices make sure that they are skewed. One method would be the following. First generate two 100 dimensional vectors a and b with integer values between 1 and 100. Next generate the i^{th} row of A with integer values between 1 and a_i and the i^{th} column of B with integer values between 1 and b_i.*

Approximating a Matrix with a Sample of Rows and Columns

Exercise 7.19 *Show that ADD^TB is exactly* ***FIX***

$$\frac{1}{s}\left(\frac{A\,(:, k_1)\,B\,(k_1, :)}{p_{k_1}} + \frac{A\,(:, k_2)\,B\,(k_2, :)}{p_{k_2}} + \cdots + \frac{A\,(:, k_s)\,B\,(k_s, :)}{p_{k_s}}\right)$$

Exercise 7.20 *Suppose $a_1, a_2, \ldots, a_m$ are nonnegative reals. Show that the minimum of $\sum\limits_{k=1}^{m} \frac{a_k}{x_k}$ subject to the constraints $x_k \geq 0$ and $\sum\limits_{k} x_k = 1$ is attained when the x_k are proportional to $\sqrt{a_k}$.*

Sketches of Documents

Exercise 7.21 *Consider random sequences of length n composed of the integers 0 through 9. Represent a sequence by its set of length k-subsequences. What is the resemblance of the sets of length k-subsequences from two random sequences of length n for various values of k as n goes to infinity?*

Exercise 7.22 *What if the sequences in the Exercise 7.21 were not random? Suppose the sequences were strings of letters and that there was some nonzero probability of a given letter of the alphabet following another. Would the result get better or worse?*

Exercise 7.23 *Consider a random sequence of length 10,000 over an alphabet of size 100.*

1.

(a) *For $k = 3$ what is probability that two possible successor subsequences for a given subsequence are in the set of subsequences of the sequence?*

(b) *For $k = 5$ what is the probability?*

Exercise 7.24 *How would you go about detecting plagiarism in term papers?*

Exercise 7.25 *Suppose you had one billion web pages and you wished to remove duplicates. How would you do this?*

Exercise 7.26 *Construct two sequences of 0's and 1's having the same set of subsequences of width w.*

Exercise 7.27 *Consider the following lyrics:*

> *When you walk through the storm hold your head up high and don't be afraid of the dark. At the end of the storm there's a golden sky and the sweet silver song of the lark.*
> *Walk on, through the wind, walk on through the rain though your dreams be tossed and blown. Walk on, walk on, with hope in your heart and you'll never walk alone, you'll never walk alone.*

How large must k be to uniquely recover the lyric from the set of all subsequences of symbols of length k? Treat the blank as a symbol.

Exercise 7.28 ***Blast**: Given a long sequence a, say 10^9 and a shorter sequence b, say 10^5, how do we find a position in a which is the start of a subsequence b' that is close to b? This problem can be solved by dynamic programming but not in reasonable time. Find a time efficient algorithm to solve this problem.*
*Hint: **(Shingling approach)** One possible approach would be to fix a small length, say seven, and consider the shingles of a and b of length seven. If a close approximation to b is a substring of a, then a number of shingles of b must be shingles of a. This should allows us to find the approximate location in a of the approximation of b. Some final algorithm should then be able to find the best match.*

8 Clustering

8.1 Some Clustering Examples

Clustering refers to the process of partitioning a set of objects into subsets consisting of similar objects. Clustering comes up in many contexts. For example, one might want to cluster journal articles into clusters of articles on related topics. In doing this, one first represents a document by a vector. This can be done using the vector space model introduced in Chapter 2. Each document is represented as a vector with one component for each term giving the frequency of the term in the document. Alternatively, a document may be represented by a vector whose components correspond to documents in the collection and the j^{th} component of the i^{th} vector is a 0 or 1 depending on whether the i^{th} document referenced the j^{th} document. Once one has represented the documents as vectors, the problem becomes one of clustering vectors.

Another context where clustering is important is the study of the evolution and growth of communities in social networks. Here one constructs a graph where nodes represent individuals and there is an edge from one node to another if the person corresponding to the first node sent an email or instant message to the person corresponding to the second node. A community is defined as a set of nodes where the frequency of messages within the set is higher than what one would expect if the set of nodes in the community were a random set. Clustering partitions the set of nodes of the graph into sets of nodes where the sets consist of nodes that send more messages to one another than one would expect by chance. Note that clustering generally asks for a strict partition into subsets, although in reality a node may belong to several communities.

In these clustering problems, one defines either a similarity measure between pairs of objects or a distance measure, a notion of dissimilarity. One measure of similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ is the cosine of the angle between them:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^T\mathbf{b}}{|\mathbf{a}|\,|\mathbf{b}|}.$$

To get a distance measure, subtract the cosine similarity from one.

$$\text{dist}(\mathbf{a}, \mathbf{b}) = 1 - \cos(\mathbf{a}, \mathbf{b})$$

Another distance measure is the Euclidean distance. There is an obvious relationship between cosine similarity and Euclidean distance. If $\mathbf{a}$ and $\mathbf{b}$ are unit vectors, then

$$|\mathbf{a}-\mathbf{b}|^2 = (\mathbf{a}-\mathbf{b})^T(\mathbf{a}-\mathbf{b}) = |\mathbf{a}|^2 + |\mathbf{b}|^2 - 2\mathbf{a}^T\mathbf{b} = 2\left(1 - \cos\left(\mathbf{a}, \mathbf{b}\right)\right).$$

In determining the distance function to use, it is useful to know something about the origin of the data. In clustering the nodes of a graph, we may represent each node as a vector, namely, as the row of the adjacency matrix corresponding to the node. One notion of dissimilarity here is the square of the Euclidean distance. For 0-1 vectors, this measure is just the number of "uncommon" 1's, whereas, the dot product is the number of common 1's.

In many situations one has a stochastic model of how the data was generated. An example is customer behavior. Suppose there are d products and n customers. A reasonable assumption is that each customer generates from a probability distribution, the basket of goods he or she buys. A basket specifies the amount of each good bought. One hypothesis is that there are only k types of customers, $k << n$. Each customer type is characterized by a probability density used by all customers of that type to generate their baskets of goods. The densities may all be Gaussians with different centers and covariance matrices. We are not given the probability densities, only the basket bought by each customer, which is observable. Our task is to cluster the customers into the k types. We may identify the customer with his or her basket which is a vector. One way to formulate the problem mathematically is by a clustering criterion that is then optimized. Some potential criteria are to partition the customers into k clusters so as to minimize

1. the sum of distances between all pairs of customers in the same cluster,
2. the sum of distances of all customers to their "cluster center" (any point in space may be designated as the cluster center), or
3. the sum of squared distances to the cluster center.

The last criterion is called the *k-means* criterion and is widely used. The variant (2) above called the *k-median* criterion minimizes the sum of distances (not squared) to the cluster center**.** Another possibility, called the *k-center* criterion, is to minimize the maximum distance of any point to its cluster center.

The chosen criterion can affect the results. To illustrate, suppose that the data was generated according to an equal weight mixture of k spherical Gaussian densities centered at $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \ldots, \boldsymbol{\mu_k}$, each with variance one in every direction. Then the density of the mixture is

$$F(\mathbf{x}) = \text{Prob}(\mathbf{x}) = \frac{1}{k}\frac{1}{(2\pi)^{d/2}}\sum_{i=1}^{k} e^{-|\mathbf{x}-\boldsymbol{\mu_i}|^2}.$$

Denote by $\boldsymbol{\mu}(\mathbf{x})$ the center nearest to $\mathbf{x}$. Since the exponential function falls off fast, we can approximate $\sum_{i=1}^{k} e^{-|\mathbf{x}-\boldsymbol{\mu_i}|^2}$ by $e^{-|\mathbf{x}-\boldsymbol{\mu}(\mathbf{x})|^2}$. Thus

$$F(\mathbf{x}) \approx \frac{1}{k}\frac{1}{(2\pi)^{d/2}} e^{-|\mathbf{x}-\boldsymbol{\mu}(\mathbf{x})|^2}.$$

The likelihood of drawing the sample of points $\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots, \mathbf{x^{(n)}}$ from the mixture, if the centers were $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \ldots, \boldsymbol{\mu_k}$, is approximately

$$\frac{1}{k^n}\frac{1}{(2\pi)^{nd/2}}\prod_{i=1}^{n} e^{-|\mathbf{x^{(i)}}-\boldsymbol{\mu}(\mathbf{x^{(i)}})|^2} = ce^{-\sum_{i=1}^{n}|\mathbf{x^{(i)}}-\boldsymbol{\mu}(\mathbf{x^{(i)}})|^2}.$$

Minimizing the sum of squared distances to cluster centers finds the maximum likelihood $\boldsymbol{\mu_1, \mu_2, \ldots, \mu_k}$. This suggests using the sum of distance squared to the cluster centers.

On the other hand, if the generating process had an exponential probability distribution, with the probability law

$$\text{Prob}\left[(x_1, x_2, \ldots, x_d)\right] = \frac{1}{2^d}\prod_{i=1}^{d} e^{-\,|x_i-\mu_i|} = \frac{1}{2^d}e^{-\sum\limits_{i=1}^{d}|x_i-\mu_i|} = \frac{1}{2^d}e^{-\,|\mathbf{x}-\boldsymbol{\mu}\,|_1},$$

one would use the L_1 norm, not the L_2 or the square of the L_1, since the probability density decreases as the L_1 distance from the center. The intuition here is that the distance used to cluster data should be related to the actual distribution of the data.

The choice of whether to use a distance measure and cluster together points that are close or use a similarity measure and cluster together points with high similarity, and what particular distance or similarity measure to use, can be crucial to the application. However, there is not much theory on these choices; they are determined by empirical domain-specific knowledge. One general observation is worth making. Using distance squared instead of distance, favors outliers since the square function magnifies large values, which means a small number of outliers may make a clustering look bad. On the other hand, distance squared has some mathematical advantages; see for example Corollary 8.2 that asserts that with the distance squared criterion, the centroid is the correct cluster center. The widely used k-means criterion is based on sum of squared distances.

There are in general two variations of the clustering problem for each of the criteria. We could require that each cluster center be a data point or allow a cluster center to be any point in space. If we require each cluster center to be a data point, the optical clustering of n data points into k clusters can be solved in time $\binom{n}{k}$ times a polynomial in the length of the data. First, exhaustively enumerate all sets of k data points as the possible sets of k cluster centers, then associate each point to its nearest center and select the best clustering. No such naive enumeration procedure is available when cluster centers can be any point in space. But, for the k-means problem, Corollary 8.2 shows that once we have identified the data points that belong to a cluster, the best choice of cluster center is the centroid. Note that the centroid might not be a data point.

In the formulations discussed so far, we have one number (e.g. sum of distances squared to the cluster center) as the measure of goodness of a clustering and we try to optimize that number to find the best clustering according to the measure. This approach does not always yield desired results, since it can be hard to find the optimum exactly. Although most clustering problems are NP-hard, often there are polynomial time algorithms to find an approximately optimal solution. But such a solution may be far from the optimal or desired clustering. We will see in Section 8.4 how to formalize some realistic conditions under which an approximate optimal solution gives us a desired clustering as well. But first we see some simple algorithms for getting a good clustering according to some natural measures.

8.2 A k-means Clustering Algorithm

There are many algorithms for clustering high dimensional data. We start with a widely used algorithm that uses the k-means criterion. In the k-means criterion, a set $A = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ of n points in d-dimensions is partitioned into k-clusters, $S_1, S_2, \ldots, S_k$, so as to minimize the sum of squared distances of each point to its cluster center. That is, A is partitioned into clusters, $S_1, S_2, \ldots, S_k$, and a center is assigned to each cluster so as to minimize

$$d\left(S_1, S_2, \ldots, S_k\right) = \sum_{j=1}^{k} \sum_{\mathbf{a_i} \in S_j} \left(\mathbf{c_j} - \mathbf{a_i}\right)^2$$

where $\mathbf{c_j}$ is the center of cluster j.

Suppose we have already determined the clustering or the partitioning into $S_1, S_2, \ldots, S_k$. What are the best centers for the clusters? The following lemma shows that the answer is the centroids, the coordinate means, of the clusters.

Lemma 8.1 *Let $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ be a set of points. The sum of the squared distances of the $\mathbf{a_i}$ to any point $\mathbf{x}$ equals the sum of the squared distances to the centroid plus the number of points times the squared distance from the point $\mathbf{x}$ to the centroid. That is,*

$$\sum_i \left|\mathbf{a_i} - \mathbf{x}\right|^2 = \sum_i \left|\mathbf{a_i} - \mathbf{c}\right|^2 + n\left|\mathbf{c} - \mathbf{x}\right|^2$$

where $\mathbf{c} = \frac{1}{n} \sum\limits_{i=1}^{n} \mathbf{a_i}$ is the centroid of the set of points.

Proof:

$$\begin{aligned} \sum_i \left|\mathbf{a_i} - \mathbf{x}\right|^2 &= \sum_i \left|\mathbf{a_i} - \mathbf{c} + \mathbf{c} - \mathbf{x}\right|^2 \\ &= \sum_i \left|\mathbf{a_i} - \mathbf{c}\right|^2 + 2(\mathbf{c} - \mathbf{x}) \cdot \sum_i \left(\mathbf{a_i} - \mathbf{c}\right) + n\left|\mathbf{c} - \mathbf{x}\right|^2 \end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\sum\limits_i \left(\mathbf{a_i} - \mathbf{c}\right) = 0$. Thus, $\sum\limits_i \left|\mathbf{a_i} - \mathbf{x}\right|^2 = \sum\limits_i \left|\mathbf{a_i} - \mathbf{c}\right|^2 + n\left|\mathbf{c} - \mathbf{x}\right|^2$ ■

A corollary of Lemma 8.1 is that the centroid minimizes the sum of squared distances since the second term, $n\left\|\mathbf{c} - \mathbf{x}\right\|^2$, is always non-negative.

Corollary 8.2 *Let $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ be a set of points. The sum of squared distances of the $\mathbf{a_i}$ to a point $\mathbf{x}$ is minimized when $\mathbf{x}$ is the centroid, namely $\mathbf{x} = \frac{1}{n} \sum\limits_i \mathbf{a_i}$.*

Another expression for the sum of squared distances of a set of n points to their centroid is the sum of all pairwise distances squared divided by n. First, a simple observation. For a set of points $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$, $\sum\limits_{i=1}^{n} \sum\limits_{j=i+1}^{n} |\mathbf{a_i} - \mathbf{a_j}|^2$ counts the quantity $|\mathbf{a_i} - \mathbf{a_j}|^2$ once for each ordered pair (i, j), $j > i$. However, $\sum\limits_{i,j} |\mathbf{a_i} - \mathbf{a_j}|^2$ counts each $|\mathbf{a_i} - \mathbf{a_j}|^2$ twice, so the later sum is twice the first sum.

Lemma 8.3 *Let $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ be a set of points. The sum of the squared distances between all pairs of points equals the number of points times the sum of the squared distances of the points to the centroid of the points. That is, $\sum\limits_{i} \sum\limits_{j>i} |\mathbf{a_i} - \mathbf{a_j}|^2 = n \sum\limits_{i} |\mathbf{a_i} - \mathbf{c}|^2$ where $\mathbf{c}$ is the centroid of the set of points.*

Proof: Lemma 8.1 states that for every $\mathbf{x}$,

$$\sum_i |\mathbf{a_i} - \mathbf{x}|^2 = \sum_i |\mathbf{a_i} - \mathbf{c}|^2 + n\, |\mathbf{c} - \mathbf{x}|^2 .$$

Letting $\mathbf{x}$ range over all $\mathbf{a_j}$ and summing the n equations yields

$$\begin{aligned}\sum_{i,j} |\mathbf{a_i} - \mathbf{a_j}|^2 &= n \sum_i |\mathbf{a_i} - \mathbf{c}|^2 + n \sum_j |\mathbf{c} - \mathbf{a_j}|^2 \\ &= 2n \sum_i |\mathbf{a_i} - \mathbf{c}|^2 .\end{aligned}$$

Observing that

$$\sum_{i,j} |\mathbf{a_i} - \mathbf{a_j}|^2 = 2 \sum_i \sum_{j>i} |\mathbf{a_i} - \mathbf{a_j}|^2$$

yields the result that

$$\sum_i \sum_{j>i} |\mathbf{a_i} - \mathbf{a_j}|^2 = n \sum_i |\mathbf{a_i} - \mathbf{c}|^2 .$$

■

The k-means clustering algorithm

A natural algorithm for k-means clustering is given below. There are three unspecified aspects of the algorithm. One is k, the number of clusters, a second is the actual set of starting centers and the third is the stopping condition.

The k-means algorithm

Start with k centers.

Cluster each point with the center nearest to it.

Find the centroid of each cluster and replace the set of old centers with the centroids.

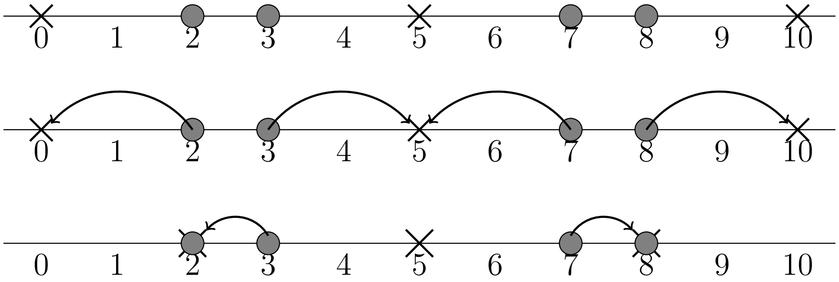
Repeat the above two steps until the centers converge (according to some criterion).

The k-means algorithm always converges but often to a local minimum. To show convergence, we argue that the sum of the squares of the distances of each point to its cluster center, always improves. Each iteration consists of two steps. First, consider the step that finds the centroid of each cluster and replaces the old centers with the new centers. By Corollary 8.2, this step improves the sum of internal cluster distances squared. The second step reclusters by assigning each point to its nearest cluster center, which also improves the internal cluster distances.

One way to determine a good value of k is to run the algorithm for each value of k and plot the sum of squared distances to the cluster centers as a function of k. If the value of the sum drops sharply going from some value of k to $k+1$, then this suggests that $k+1$ corresponds to the number of clusters in a natural partition of the data.

A problem that arises with some implementations of the k-means clustering algorithm is that one or more of the clusters becomes empty and there is no center from which to measure distance. A simple case where this occurs is illustrated in the following example. You might think how you would modify the code to resolve this issue.

Example: Consider running the k-means clustering algorithm to find three clusters on the following 1-dimension data set: 2,3,7,8 starting with center 0,5,10.



The center at 5 ends up with no items and there are only two clusters instead of the desired three. ■

Another issue that arises is whether the clusters have any real significance. The k-means algorithm will find k clusters even in $G(n,p)$. But note that since the graph $G(n,p)$ should look uniform everywhere, there aren't really k meaningful clusters where a clustering is meaningful if any close to optimal clustering is almost identical to it. In fact, there are many ways of clustering the vertices of this graph all of which will be nearly optimal with respect to the k-means or k-median criteria.

8.3 A Greedy Algorithm for k-Center Criterion Clustering

In this section, instead of using the k-means clustering criterion, we use the k-center criterion. The k-center criterion partitions the points into k clusters so as to minimize the maximum distance of any point to its cluster center. Call the maximum distance of any point to its cluster center the radius of the clustering. There is a k-clustering of radius r if and only if there are k spheres, each of radius r, which together cover all the points. Below, we give a simple algorithm to find k spheres covering a set of points. The following lemma shows that this algorithm only needs to use a radius that is "off by a factor of at most two" from the optimal k-center solution.

The Greedy k-clustering Algorithm

> Pick any data point to be the first cluster center. At time t, for $t = 2, 3, \ldots, k$, pick any data point that is not within distance r of an existing cluster center; make it the t^{th} cluster center.

Lemma 8.4 *If there is a k-clustering of radius $\frac{r}{2}$, then the above algorithm finds a k-clustering with radius at most r.*

Proof: Suppose for contradiction that the algorithm using radius r fails to find a k-clustering. This means that after the algorithm chooses k centers, there is still at least one data point that is not in any sphere of radius r around a picked center. This is the only possible mode of failure. But then there are $k + 1$ data points, with each pair more than distance r apart. Clearly, no two such points can belong to the same cluster in any k-clustering of radius $\frac{r}{2}$ contradicting the hypothesis. ∎

8.4 Spectral Clustering

In this section we give two contexts where spectral clustering is used. The first is used for finding communities in graphs and the second for clustering a general set of data points. We begin with a simple explanation as to how spectral clustering works when applied to the adjacency matrix of a graph.

Spectral clustering applied to graphs

In spectral clustering of the vertices of a graph, one first creates a new matrix V whose columns correspond to the first k singular vectors of the adjacency matrix. Each row of V is the projection of a row of the adjacency matrix to the space spanned by the k singular vectors. In the example below, the graph has five vertices divided into two cliques, one consisting of the first three vertices and the other the last two vertices. The top two right singular vectors of the adjacency matrix, not normalized to length one, are $(1, 1, 1, 0, 0)^T$ and $(0, 0, 0, 1, 1)^T$. The five rows of the adjacency matrix projected to these vectors form the 5×2 matrix in Figure 8.1. Here, in fact there are two ideal clusters with all edges inside a cluster being present including all self-loops and all edges between clusters being absent. The five rows project to just two points, depending on which cluster the rows are in. If the clusters were not so ideal and instead of the graph consisting of two disconnected

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad V = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

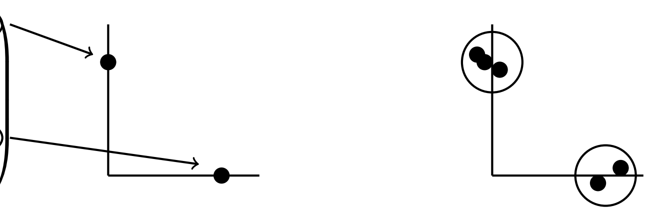


Figure 8.1: Illustration of spectral clustering.

cliques, the graph consisted of two dense subsets of vertices where the two sets were connected by only a few edges, then the singular vectors would not be indicator vectors for the clusters but close to indicator vectors. The rows would be mapped to two clusters of points instead of two points. A k-means clustering algorithm would find the clusters.

If the clusters were overlapping, then instead of two clusters of points, there would be three clusters of points where the third cluster corresponds to the overlapping vertices of the two clusters. Instead of using k-means clustering, we might instead find the minimum 1-norm vector in the space spanned by the two singular vectors. The minimum 1-norm vector will not be an indicator vector, so we would threshold its values to create an indicator vector for a cluster. Instead of finding the minimum 1-norm vector in the space spanned by the singular vectors in V, we might actually look for a small 1-norm vector close to the subspace.

$$\min_{\mathbf{x}}(1 - |\mathbf{x}|_1 + \alpha\cos(\theta))$$

Here θ is the cosine of the angle between $\mathbf{x}$ and the space spanned by the two singular vectors. α is a control parameter that determines how close we want the vector to be to the subspace. When α is large, $\mathbf{x}$ must be close to the subspace. When α is zero, $\mathbf{x}$ can be anywhere.

Finding the minimum 1-norm vector in the space spanned by a set of vectors can be formulated as a linear programming problem. To find the minimum 1-norm vector in V, write $V\mathbf{x} = \mathbf{y}$ where we want to solve for both $\mathbf{x}$ and $\mathbf{y}$. Note that the format is different from the usual format for a set of linear equations $A\mathbf{x} = \mathbf{b}$ where $\mathbf{b}$ is a known vector.

Finding the minimum 1-norm vector looks like a nonlinear problem.

$$\min |\mathbf{y}|_1 \text{ subject to } V\mathbf{x} = \mathbf{y}$$

To remove the absolute value sign, write $\mathbf{y} = \mathbf{y_1} - \mathbf{y_2}$ with $\mathbf{y_1} \geq 0$ and $\mathbf{y_2} \geq 0$. Then solve

$$\min \left(\sum_{i=1}^{n} y_{1i} + \sum_{i=1}^{n} y_{2i} \right) \text{ subject to } V\mathbf{x} = \mathbf{y}, \mathbf{y_1} \geq 0, \text{ and } \mathbf{y}_2 \geq 0.$$

Write $V\mathbf{x} = \mathbf{y_1} - \mathbf{y_2}$ as $V\mathbf{x} - \mathbf{y_1} + \mathbf{y_2} = 0$. then we have the linear equations in a format we are accustomed to.

$$[V, -I, I] \begin{pmatrix} \mathbf{x} \\ \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

This is a linear programming problem. The solution, however, happens to be $\mathbf{x} = 0$, $\mathbf{y_1} = 0$, and $\mathbf{y_2} = 0$. To resolve this, add the equation $y_{1i} = 1$ to get a community containing the vertex i.

Often we are looking for communities of 50 or 100 vertices in graphs with hundreds of million of vertices. We want a method to find such communities in time proportional to the size of the community and not the size of the entire graph. Here spectral clustering can be used but instead of calculating singular vectors of the entire graph, we do something else. Consider a random walk on a graph. If we walk long enough the probability distribution converges to the first eigenvector. However, if we take only a few steps from a start vertex or small group of vertices that we believe define a cluster, the probability will distribute over the cluster with some of the probability leaking out to the remainder of the graph. To get the early convergence of several vectors which would ultimately converge to the first few singular vectors, we take a subspace $[\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, A^3\mathbf{x}]$ and propagate the subspace. At each iteration we find an orthonormal basis and then multiply each basis vector by A. We then take the resulting basis vectors after a few steps, say five, and find a minimum 1-norm vector in the subspace.

Spectral clustering applied to data

Consider n data points arranged as the rows of an $n \times d$ matrix A. These are to be partitioned into k clusters where k is much smaller than n or d. Finding the best k-means clustering of the points is known to be NP-hard. However, there are efficient algorithms that find a k-clustering within a factor of two of the best. We will see that singular value decomposition together with this type of approximate k-means clustering is very useful.

Spectral Clustering of the data into k clusters. The method consists of the following steps:

1. Find the top k right singular vectors of the data matrix A.

2. Project each row of A into the space spanned by these singular vectors to obtain a $n \times d$ matrix $\bar{A}$.

3. Apply an algorithm to find an approximately optimal k-clustering of $\bar{A}$.

It is important to note that the projected points are being clustered, i.e., the rows of $\bar{A}$ and not A itself. Projection offers the obvious advantage of decreasing the dimension of the problem from d to k, making it easier to cluster. The more important advantage of projecting is that it yields cluster centers closer to the true centers than clustering A. This is not so obvious and we demonstrate it here. The formal statement is contained in Theorem 8.7.

We will see how to use the fact that spectral clustering finds centers close to the true centers to get an actual clustering close to the true clustering. But this makes sense only if there is no ambiguity about what the true clustering is. We will develop a notion of a proper clustering that says the clusters are distinct enough so as not to be confused with each other. We will then show that if there is a proper clustering, spectral clustering will find a clustering close to the proper clustering. This is proved in Theorem 8.9.

Consider a spherical Gaussian F in $\mathbf{R}^d$ with mean $\boldsymbol{\mu}$ and variance one in every direction. As we saw in Chapter 2, for a point $\mathbf{x}$ picked according to F, $|\mathbf{x}-\boldsymbol{\mu}|^2$ is likely to be about d. Now suppose we apply an approximate k-means clustering algorithm, which finds a clustering with sum of distances squared at most $(1+\varepsilon)$ times the optimal. With this amount of error, a center $\boldsymbol{\mu'}$ found may have $|\boldsymbol{\mu'}-\boldsymbol{\mu}|^2 \approx \varepsilon d$.

Consider a mixture of two spherical Gaussians in $\mathbf{R}^d$, for large d, each of variance one in every direction. If the inter-center separation between them is say six, which is six standard deviations, then the error of εd would result in confusing the two. So, even in this simple case, approximate optimization does not do a good job. Now consider a mixture of k spherical Gaussians, each of variance one in every direction, We saw in Chapter 4 that the space spanned by the top k singular vectors contains the means of the k Gaussians. Project all data points on to this space. The densities are still Gaussian in the projection with variance again one in every direction. The mean squared distance of projected data points to the projected mean of the respective densities is $O(k)$ and in an approximately best k-means clustering, the cluster centers will be at distance squared at most $O(k)$ from the true means, not $O(d)$. In this example, we assumed that the data points were stochastically generated from a mixture of Gaussians. We show in what follows that this is not necessary. Indeed, we show that the intuitive argument here also holds for any arbitrary set of data points. But first, we have to define an analog of variance for a general set of data points. This is simple. It is just the average squared distance from the cluster center instead of the average distance squared to the mean of the probability density. Now, for spherical Gaussians, the squared distance in every direction is the same, but in general, they are not and we will take the maximum over all directions.

Represent a k-clustering by a $n \times d$ matrix C with each row of C being the cluster center of the cluster the corresponding row of A belongs to. Note that C has only k

distinct rows. Define the *variance* of C, denoted $\sigma^2(C)$, by

$$\sigma^2(C) = \max_{\substack{\mathbf{v} \\ |\mathbf{v}|=1}} \frac{1}{n} |(A - C)\mathbf{v}|^2,$$

which is simply the maximum, in any direction $\mathbf{v}$, of the mean-squared distance of a data point from its cluster center. It is easy to see that $\sigma^2(C) = \frac{1}{n}||A - C||_2^2$.

If we had a stochastic model of data, as for example a mixture of Gaussians generating the data, then there is a true clustering and it is desirable that our algorithm find this clustering or at least come close. In general, we do not assume that there is a stochastic model and so there is no true clustering. Nevertheless, we will be able to show that spectral clustering does nearly as well as any clustering C. Namely, for most data points, the cluster centers found by spectral clustering will be at distance at most $O\left(\sqrt{k}\,\sigma(C)\right)$ of the cluster centers in C. The reader should think about the question: How is it that the one clustering found by the algorithm can do this for every possible clustering? The answer is that if C is a very bad clustering, $\sigma(C)$ is large and the requirement of being within distance $O(\sqrt{k}\sigma(C))$ is very weak. For the theorem below, recall the notation that $\mathbf{a_i}, \mathbf{c_i}$, and $\mathbf{c_i'}$ are respectively the i^{th} row of A, C, and C'.

First, we need two technical lemmas.

Lemma 8.5 *For any two vectors* $\mathbf{u}$ *and* $\mathbf{v}$*,*

$$|\mathbf{u} + \mathbf{v}|^2 \geq \frac{1}{2}|\mathbf{u}|^2 - |\mathbf{v}|^2.$$

Proof:

$$\begin{aligned} |\mathbf{u} + \mathbf{v}|^2 &= (\mathbf{u} + \mathbf{v}) \cdot (\mathbf{u} + \mathbf{v}) = |\mathbf{u}|^2 + |\mathbf{v}|^2 + 2\mathbf{u} \cdot \mathbf{v} \\ &\geq |\mathbf{u}|^2 + |\mathbf{v}|^2 - 2|\mathbf{u}||\mathbf{v}| = (|\mathbf{u}| - |\mathbf{v}|)^2 . \end{aligned}$$

From this and the fact that for any two real numbers a and b,

$$(a - b)^2 \geq (a - b)^2 - \left(\frac{1}{\sqrt{2}}a - \sqrt{2}b\right)^2 = \frac{1}{2}a^2 - b^2,$$

the claim follows. ■

Lemma 8.6 *Suppose A is an $n \times d$ matrix and $\bar{A}$ is the projection of the rows of A onto the subspace spanned by the top k singular vectors of A. Then for any matrix C of rank at most k,*

$$||\bar{A} - C||_F^2 \leq 8k||A - C||_2^2.$$

Proof: Since the rank of $\bar{A} - C$ is at most the sum of the ranks of $\bar{A}$ and C, which is most $2k$,

$$||\bar{A} - C||_F^2 \leq 2k||\bar{A} - C||_2^2 \tag{8.1}$$

by Lemma 4.2. Now

$$||\bar{A} - C||_2 \leq ||\bar{A} - A||_2 + ||A - C||_2 \leq 2||A - C||_2,$$

the last inequality since $\bar{A}$ is the best rank k approximation for the spectral norm and C has rank at most k. Combining this with (8.1), the lemma follows. ■

Theorem 8.7 *Suppose A is a $n \times d$ data matrix and C is any clustering of A and suppose C' (also a $n \times d$ matrix with k distinct rows) is the clustering of $\bar{A}$ found by the spectral clustering algorithm. For all but εn of the data points, we have $|\mathbf{c_i} - \mathbf{c'_i}|^2 < \frac{48k}{\varepsilon}\sigma^2(C)$.*

Proof: Let $\Delta = \frac{48k}{\varepsilon}\sigma^2(C)$ and $B = \{i||\mathbf{c_i} - \mathbf{c'_i}|^2 \geq \Delta\}$ be the bad set of i. We must show that B has at most εn elements.

$$\begin{aligned}
\sum_{i \in B} |\mathbf{\bar{a}_i} - \mathbf{c'_i}|^2 &= \sum_{i \in B} |(\mathbf{c_i} - \mathbf{c'_i}) + (\mathbf{\bar{a}_i} - \mathbf{c_i})|^2 \\
&\geq \frac{1}{2}\sum_{i \in B} |\mathbf{c_i} - \mathbf{c'_i}|^2 - \sum_{i \in B} |\mathbf{\bar{a}_i} - \mathbf{c_i}|^2 \qquad \text{by the Lemma 8.5} \\
&\geq \frac{1}{2}|B|\Delta - \sum_{i=1}^{n} |\mathbf{\bar{a}_i} - \mathbf{c_i}|^2 = \frac{1}{2}|B|\Delta - ||\bar{A} - C||_F^2.
\end{aligned}$$

On the other hand,

$$\sum_{i \in B} |\mathbf{\bar{a}_i} - \mathbf{c'_i}|^2 \leq \sum_{i=1}^{n} |\mathbf{\bar{a}_i} - \mathbf{c'_i}|^2 \leq 2\sum_{i=1}^{n} |\mathbf{\bar{a}_i} - \mathbf{c_i}|^2 = 2||\bar{A} - C||_F^2,$$

since, C' is within a factor of two of being the best k-means clustering of the projected data matrix $\bar{A}$ implies that if we took C as a clustering of $\bar{A}$, then, it is at most a factor of two better than C'. Combining,

$$3||\bar{A} - C||_F^2 \geq \frac{1}{2}|B|\Delta$$

which implies

$$|B| \leq \frac{6\varepsilon||\bar{A} - C||_F^2}{48k\sigma^2(C)}.$$

From Lemma 8.6, $||\bar{A} - C||_F^2 \leq 8k||A - C||_2^2 = 8kn\sigma^2(C)$. Plugging this in, the theorem follows. ■

We need the following lemma which asserts another property of spectral clustering, namely that the clustering it finds has σ which is within a factor of $5\sqrt{k}$ of the best possible σ for any clustering.

Lemma 8.8 *Let C^* be the k-clustering with the minimum σ among all k-clusterings of the data. For the clustering C' found by spectral clustering, we have*

$$\sigma(C') \leq 5\sqrt{k}\sigma(C^*).$$

Proof:

$$\begin{aligned} ||A - C'||_2 &\leq ||A - \bar{A}||_2 + ||\bar{A} - C'||_2 \leq ||A - C^*||_2 + ||\bar{A} - C'||_F \\ &\leq \sqrt{n}\sigma(C^*) + \sqrt{2}||\bar{A} - C^*||_F \leq \sqrt{n}\sigma(C^*) + 4\sqrt{kn}\sigma(C^*), \text{ by Lemma 8.6.} \end{aligned}$$

For the second inequality, we used the fact that since $\bar{A}$ is the best rank k approximation to A in spectral norm, $||A - C^*||_2 \geq ||A - \bar{A}||_2$ and for the third inequality, we used the fact that C' is within a factor of two of the optimal k-means clustering of $\bar{A}$ and in particular, the clustering C^* is not better by a factor of more than two. Now the lemma follows. ■

Now we show that we can use the fact that spectral clustering finds cluster centers close to the true centers to find approximately the true clustering. This makes sense only if there is no ambiguity about what the true clustering is. A necessary condition for a clustering to be unambiguous is that the clusters must be distinct or spatially well-separated. Otherwise, points could be put into either of two nearby clusters without changing the k-means objective function much. We make this more precise with the following definition:

Definition 8.1 *A clustering C^* is said to be proper if*

1. *$\sigma(C^*)$ is least among all k-clustering of the data, and*
2. *the centers of any two clusters in C^* are separated by a distance of at least $70k^2\sigma(C^*)/\sqrt{\varepsilon}$.*

■

Here, ε is any positive real number. Why do we choose this definition of a proper clustering? For the case of spherical Gaussians, the separation required here corresponds to the means of different Gaussians being a constant number of standard deviations apart. A different definition might have insisted on the clusters being distinct enough so that even an approximately best k-means clustering in $\mathbf{R}^d$, would give approximately the true clustering. Since for a spherical Gaussian with variance one in every direction, data points are about $\sqrt{d}$ away from the center, this would intuitively require the means of two such Gaussians involved in a mixture to be at least $\Omega(\sqrt{d})$ apart, which is a stronger requirement than that of being proper. To be proper, a separation of only $\Omega(1)$ is required.

We modify the spectral clustering algorithm by adding a *merge step* at the end.

Merge Step Let C' be the clustering found by spectral clustering. Repeatedly merge any two clusters with cluster centers separated by a distance of at most $14\sqrt{k}\sigma(C')/\sqrt{\varepsilon}$.

Theorem 8.9 *Suppose there is a proper clustering C^* of data points. Then, spectral clustering followed by merge-step produces a clustering $C^{(0)}$ with the property that by reclustering at most εn points, we can get from $C^{(0)}$ to C^*.*

Proof: Let C' be the clustering produced by spectral clustering before the merge step is executed. Let $\Delta = 49k\sigma^2(C')/\varepsilon$. Define $B = \{i : |\mathbf{c'_i} - \mathbf{c^*_i}|^2 > \Delta\}$. Let S be one particular cluster in C^*. For any $i, j \in S \setminus B$, we have $|\mathbf{c'_i} - \mathbf{c^*_i}| \le \sqrt{\Delta}$ and $|\mathbf{c'_j} - \mathbf{c^*_j}| \le \sqrt{\Delta}$. Since $\mathbf{c^*_i} = \mathbf{c^*_j}$, i and j will be in one cluster after the merge step. Now if S and T are two different clusters in C^*, by the definition of proper, for $i \in S \setminus B$ and $j \in T \setminus B$, we have

$$|\mathbf{c^*_i} - \mathbf{c^*_j}| \ge 70k^2\sigma(C^*)/\sqrt{\varepsilon} \ge 14k^{3/2}\sigma(C')/\sqrt{\varepsilon} \ge 2k\sqrt{\Delta},$$

by Lemma 8.8. So $|\mathbf{c'_i} - \mathbf{c'_j}| \ge 2(k-1)\sqrt{\Delta}$ by the definition of B and the merge step (even when repeated $k-1$ times) will not merge i and j into one cluster. Thus, for all $i, j \notin B$, we have that i and j belong to the same cluster in C^* if and only if they belong to the same cluster in $C^{(0)}$. Thus, by reclustering at most $|B|$ points, we can get from $C^{(0)}$ to C^*. ■

8.5 Recursive Clustering Based on Sparse Cuts

Suppose we are given an undirected, connected graph $G(V, E)$ in which an edge indicates the end point vertices are similar. Recursive clustering starts with all vertices in one cluster and recursively splits a cluster into two parts whenever there are not too many edges from one part to the other part of the cluster. For this technique to be effective it is important that the data has an hierarchical clustering. Consider what would happen if one used recursive clustering to find communities of students at an institution hoping that one of the clusters might be computer science students. At the first level one might get four clusters corresponding to freshman, sophomores, juniors, and seniors. At the next level one might get clusters that were majors partitioned into year rather than majors. Another problem would occur if the real top level clusters were overlapping. If one is clustering journals articles, the top level might be mathematics, physics, chemistry, etc. However, there are papers that are related to both mathematics and physics. Such a paper would be put in one cluster or the other and the community that the paper really belonged in would be split and thus never found at lower levels in the clustering.

Formally, for two disjoint sets S and T of vertices, define

$$\Phi(S, T) = \frac{\text{number of edges from } S \text{ to } T}{\text{total number of edges incident to } S \text{ in } G}.$$

$\Phi(S, T)$ measures the relative strength of similarities between S and T. Let $d(i)$ be the degree of vertex i and for $d(S) = \sum_{i \in S} d(i)$. Let m be the total number of edges. The following algorithm cuts only a small fraction of the edges, yet ensures that each cluster is consistent, namely no subset of it has low similarity to the rest of the cluster.

Recursive Clustering Algorithm

If a current cluster W has a subset S with $d(S) \leq \frac{1}{2}d(W)$ and $\Phi(S,T) \leq \varepsilon$, then split W into two clusters: S and W-S. Repeat until no such split is possible.

Theorem 8.10 *At termination of the above algorithm, the total number of edges between vertices in different clusters is at most $O(\varepsilon m \ln n)$.*

Proof: Each edge between two different clusters at the end was "cut up" at some stage by the algorithm. We will "charge" edge cuts to vertices and bound the total charge. When the algorithm partitions a cluster W into S and W-S with $d(S) \leq (1/2)d(W)$, each $k \in S$ is charged $\frac{d(k)}{d(W)}$ times the number of edges being cut. Since $\Phi(S, W\text{-}S) \leq \varepsilon$, the charge added to each $k \in W$ is a most $\varepsilon d(k)$. A vertex is charged only when it is in the smaller part $(d(S) \leq d(W)/2)$ of the cut. So between any two times it is charged, $d(W)$ is reduced by a factor of at least two and so a vertex can be charged at most $\log_2 m \leq O(\ln n)$ times, proving the theorem. ■

To implement the algorithm, we have to compute $\text{Min}_{S \subseteq W} \Phi(S, W\text{-}S)$, an NP-hard problem. So the theorem cannot be implemented right away. Luckily, eigenvalues and eigenvectors, which can be computed fast, give an approximate answer. The connection between eigenvalues and sparsity, known as Cheeger's inequality, is deep with applications to Markov chains among others. We do not discuss this here.

8.6 Kernel Methods

The clustering methods discussed so far work well only when the data satisfy certain conditions. For example, in any distance-based measure like k-means or k-center, once the cluster centers are fixed, the Vornoi diagram of the cluster centers determines which cluster each data point belongs to. Cells of the Vornoi diagram are determined by hyperplane bisectors of line segments joining pairs of centers. This implies that clusters are linearly separable.

Such criteria cannot separate clusters that are not linearly separable in the input space. The chapter on learning had many examples that were not linearly separable in the original space, but were linearly separable when mapped to a higher dimensional space using a nonlinear function called a kernel. An analogous technique can be used in the case of clustering, but with two differences.

1. There may be any number k of clusters, whereas in learning, there were just two classes, the positive and negative examples.

2. There is unlabelled data, i.e., we are not given which cluster each data point belongs to, whereas in the case of learning each data point was labeled. The clustering situation is sometimes called *unsupervised* whereas the labeled learning situation is called *supervised*, the reason being, one imagines a supervisor, human judgement, supplying the labels.

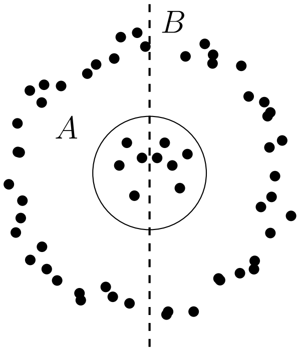


Figure 8.2: Example where 2-median clustering B is not natural.

These two differences do not prevent the application of kernel methods to clustering. Indeed, here too, one could first embed the data in a different space using the Gaussian or other kernel and then run k-means in the embedded space. Again, one need not write down the whole embedding explicitly. In the learning setting, since there were only two classes with a linear separator, we were able to write a convex program to find the normal of the separator. When there are k classes, there could be as many as $\binom{k}{2}$ hyperplanes separating pairs of classes, so the computational problem is harder and there is no simple convex program to solve the problem. However, we can still run the k-means algorithm in the embedded space. The centroid of a cluster is kept as the average of the data points in the cluster. Recall that we only know dot products, not distances in the higher dimensional space, but we can use the relation $|\mathbf{x}-\mathbf{y}|^2 = \mathbf{x}\cdot\mathbf{x}+\mathbf{y}\cdot\mathbf{y}+2\mathbf{x}\cdot\mathbf{y}$ to go from dot products to distances.

There are situations in which high-dimensional data points lie in a lower dimensional manifold. In such situations, a Gaussian kernel is useful. Say we are given a set of n points $S = \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}\}$ in $\mathbf{R}^d$ that we wish to cluster into k subsets. The Gaussian kernel uses an affinity measure that emphasizes closeness of points and drops off exponentially as the points get farther apart. We define the affinity between points i and j by

$$a_{ij} = \begin{cases} e^{-\frac{1}{2\sigma^2}\,\|\mathbf{s_i}-\mathbf{s_j}\|^2} & i \neq j \\ 0 & i = j. \end{cases}$$

The affinity matrix gives a closeness measure for points. The measure drops off exponentially fast with distance and thus favors close points. Points farther apart have their closeness shrink to zero. We give two examples to illustrate briefly the use of Gaussian kernels. The first example is similar to Figure 8.2 of points on two concentric annuli. Suppose the annuli are close together, i.e., the distance between them is $\delta << 1$. Even if we used similarity between objects, rather than say the k-median criterion, it is not clear that we will get the right clusters; namely two separate circles. Instead, suppose the circles are sampled at a rate so that adjacent samples are separated by a distance $\varepsilon << \delta$. Define a Gaussian kernel with variance ε^2. Then, if sample $\mathbf{s_1}$ is on Circle 1 and sample $\mathbf{s_2}$ is on

Circle 2, $e^{-|\mathbf{s_1}-\mathbf{s_2}|^2/2\varepsilon^2} << 1$, so they are very likely to be put in separate clusters as desired.

Our second example has three curves. Suppose two points from two different curves are never closer than δ. If we sample at a high enough rate, every sample will have many other samples from the same curve close to it giving high similarity according to the Gaussian kernel, but no two samples from different curves will have high similarity. This example can be generalized to a situation where the points lie on different "sheets" or low dimensional manifolds.

Two points near each other on the same circle will have a high affinity value, i.e., they will be close together in this metric. For the right sigma value, the two closest points, one from each circle, will be infinitely far apart. Thus, the affinity matrix is a band matrix, consisting of two blocks of data.

8.7 Agglomerative Clustering

Agglomerative clustering is the opposite of recursive clustering. It starts with each point in a separate cluster and then repeatedly merges the two closest clusters into one. There are various criteria to determine which two clusters are merged at any point. They are based on first defining a distance between two clusters in terms of the distance between points. Four of these possibilities are listed below.

1. Nearest neighbor - the distance between clusters C_i and C_j is the distance between the points in C_i and C_j that are closest.

   $$d_{\min}\left(C_i, C_j\right) = \min_{\substack{\mathbf{x}\in C_i\\ \mathbf{y}\in C_j}} |\mathbf{x}-\mathbf{y}|$$

   This measure basically builds the minimal cost spanning tree.

2. Farthest neighbor - the distance between clusters C_i and C_j is the distance between the points in C_i and C_j that are farthest apart.

   $$d_{\max}\left(C_i, C_j\right) = \max_{\substack{\mathbf{x}\in C_i\\ \mathbf{y}\in C_j}} |\mathbf{x}-\mathbf{y}|$$

3. Mean - the distance between two clusters is the distance between the centroids of the clusters.

4. Average - the distance between two clusters is the average distance between points in the two clusters.

Agglomerative clustering in moderate and high dimensions often gives rise to a very unbalanced tree. This section gives some insight into the cause of this phenomenon. We begin by considering agglomerative clustering of 1-dimensional data. The 1-dimensional

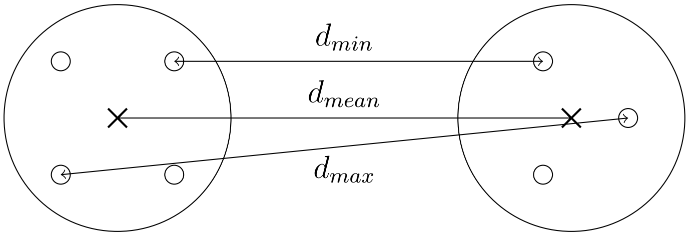


Figure 8.3: Illustration of minimum, maximum and mean distances for a set of points.

case gives rise to a balanced tree.

Consider agglomerative clustering using nearest neighbor of n points on a line. Assume the distances between adjacent points are independent random variables and measure the distance between clusters by the distance of the nearest neighbors. Here it is easy to see that each cluster is always an interval. In this case, any two adjacent intervals are equally likely to be merged. The last merge will occur when the largest distance between two adjacent points is encountered. This is equally likely to be the distance between point i and point $i+1$ for any i, $1 \leq i < n$. The height of the final merge tree will be one more than the maximum of the heights of the two subtrees. Let $h(n)$ be the expected height of a merge tree with n leaves. Then

$$\begin{aligned} h\left(n\right) &= 1 + \tfrac{1}{n}\sum_{i=1}^{n} \max\left\{h\left(i\right), h\left(n-i\right)\right\} \\ &= 1 + \tfrac{2}{n}\sum_{i=\frac{n}{2}+1}^{n} h\left(i\right) \\ &= 1 + \tfrac{2}{n}\sum_{i=\frac{n}{2}+1}^{\frac{3}{4}n} h\left(i\right) + \tfrac{2}{n}\sum_{i=\frac{3}{4}n+1}^{n} h\left(i\right) \end{aligned}$$

Since $h(i)$ is monotonic, for $\frac{n}{2} < i \leq \frac{3}{4}n$, bound $h\left(i\right)$ by $h\left(\frac{3}{4}n\right)$ and for $\frac{3}{4}n < i \leq n$, bound $h\left(i\right)$ by $h\left(n\right)$. Thus,

$$\begin{aligned} h\left(n\right) &\leq 1 + \tfrac{2}{n}\tfrac{n}{4}h\left(\tfrac{3n}{4}\right) + \tfrac{2}{n}\tfrac{n}{4}h\left(n\right) \\ &\leq 1 + \tfrac{1}{2}h\left(\tfrac{3n}{4}\right) + \tfrac{1}{2}h\left(n\right). \end{aligned}$$

This recurrence has a solution $h\left(n\right) \leq b\log n$ for sufficiently large b. Thus, the merge tree has no long path and is bushy.

If the n points are in high dimension rather than constrained to a line, then the distance between any two points, rather than two adjacent points, can be the smallest distance. One can think of edges being added one at a time to the data to form a spanning tree. Only now we have an arbitrary tree rather than a straight line. The order in which the edges are added corresponds to the order in which the connected components are merged by the agglomerative algorithm. Two extreme cases of the spanning tree for the set of points are the straight line which gives a bushy agglomerative tree or a star which gives a skinny agglomerative tree of height n. Note there are two trees involved here, the spanning tree and the agglomerative tree.

The question is what is the shape of the spanning tree? If distance between components is nearest neighbor, the probability of an edge between two components is proportional to the size of the components. Thus, once a large component forms it will swallow up small components giving a more star like spanning tree and hence a tall skinny agglomerative tree. Notice the similarity to the $G(n, p)$ problem.

If we defined distance between two clusters to be the maximum distance between any two points in the clusters and merge the two clusters that are the smallest distance apart, then we are more likely to get a bushy spanning tree and a skinny agglomerative tree. If all distances between points are independent and we have two clusters of size k and a singleton, the maximum distance between the points in the two clusters of size k is likely to give a larger distance than the maximum between the singleton and the k points in a cluster. Thus, the singleton will likely merge into one of the clusters before the two clusters will merge and in general small clusters will combine before larger ones, resulting in a bushy spanning tree and a bushy agglomerative tree.

8.8 Dense Submatrices and Communities

Represent n data points in d-space by the rows of an $n \times d$ matrix A. Assume that A has all nonnegative entries. Examples to keep in mind for this section are the document-term matrix and the customer-product matrix. We address the question of how to define and find efficiently a coherent large subset of rows. To this end, the matrix A can be represented by a bipartite graph. One side has a vertex for each row and the other side a vertex for each column. Between the vertex for row i and the vertex for column j, there is an edge with weight a_{ij}.

We want a subset S of row vertices and a subset T of column vertices so that

$$A(S, T) = \sum_{i \in S, j \in T} a_{ij}$$

is high. This simple definition is not good since $A(S, T)$ will be maximized by taking all rows and columns. We need a balancing criterion that ensures that $A(S, T)$ is high relative to the sizes of S and T. One possibility is to maximize $\frac{A(S,T)}{|S||T|}$. This is not a good

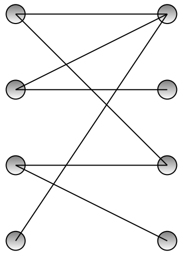


Figure 8.4: Example of a bipartite graph.

measure either, since it is maximized by the single edge of highest weight. The definition we use is the following. Let A be a matrix with nonnegative entries. For a subset S of rows and a subset T of columns, the *density* $d(S,T)$ of S and T is $d(S,T) = \frac{A(S,T)}{\sqrt{|S||T|}}$. The *density* $d(A)$ of A is defined as the maximum value of $d(S,T)$ over all subsets of rows and columns. This definition applies to bipartite as well as non bipartite graphs.

One important case is when A's rows and columns both represent the same set and a_{ij} is the similarity between object i and object j. Here $d(S,S) = \frac{A(S,S)}{|S|}$. If A is an $n \times n$ 0-1 matrix, it can be thought of as the adjacency matrix of an undirected graph, and $d(S,S)$ is the average degree of a vertex in S. The subgraph of maximum average degree in a graph can be found exactly by network flow techniques, as we will show in the next section. We do not know an efficient (polynomial-time) algorithm for finding $d(A)$ exactly in general. However, we show that $d(A)$ is within a $O(\log^2 n)$ factor of the top singular value of A assuming $|a_{ij}| \leq 1$ for all i and j. This is a theoretical result. The gap may be much less than $O(\log^2 n)$ for many problems, making the singular value and singular vector quite useful. Also, S and T with $d(S,T) \geq \Omega(d(A)/\log^2 n)$ can be found algorithmically.

Theorem 8.11 *Let A be an $n \times d$ matrix with entries between 0 and 1. Then*

$$\sigma_1(A) \geq d(A) \geq \frac{\sigma_1(A)}{4 \log n \log d}.$$

Furthermore, subsets S and T satisfying $d(S,T) \geq \frac{\sigma_1(A)}{4 \log n \log d}$ may be found from the top singular vector of A.

Proof: Let S and T be the subsets of rows and columns that achieve $d(A) = d(S,T)$. Consider an n-vector $\mathbf{u}$ which is $\frac{1}{\sqrt{|S|}}$ on S and 0 elsewhere and a d-vector $\mathbf{v}$ which is $\frac{1}{\sqrt{|T|}}$ on T and 0 elsewhere. Then,

$$\sigma_1(A) \geq \mathbf{u}^T A \mathbf{v} = \sum_{ij} u_i v_j a_{ij} = d(S,T) = d(A)$$

establishing the first inequality.

To prove the second inequality, express $\sigma_1(A)$ in terms of the first left and right singular vectors $\mathbf{x}$ and $\mathbf{y}$.

$$\sigma_1(A) = \mathbf{x}^T A\mathbf{y} = \sum_{i,j} x_i a_{ij} y_j, \qquad |\mathbf{x}| = |\mathbf{y}| = 1.$$

Since the entries of A are nonnegative, the components of the first left and right singular vectors must all be nonnegative, that is, $x_i \geq 0$ and $y_j \geq 0$ for all i and j. To bound $\sum\limits_{i,j} x_i a_{ij} y_j$, break the summation into $O(\log n \log d)$ parts. Each part corresponds to a given α and β and consists of all i such that $\alpha \leq x_i < 2\alpha$ and all j such that $\beta \leq y_i < 2\beta$. The $\log n \log d$ parts are defined by breaking the rows into $\log n$ blocks with α equal to $\frac{1}{2}\frac{1}{\sqrt{n}}$, $\frac{1}{\sqrt{n}}$, $2\frac{1}{\sqrt{n}}$, $4\frac{1}{\sqrt{n}}$, ... , 1 and by breaking the columns into $\log d$ blocks with β equal to $\frac{1}{2}\frac{1}{\sqrt{d}}$, $\frac{1}{\sqrt{d}}$, $\frac{2}{\sqrt{d}}$, $\frac{4}{\sqrt{d}}$, ... , 1. The i such that $x_i < \frac{1}{2\sqrt{n}}$ and the j such that $y_j < \frac{1}{2\sqrt{d}}$ will be ignored at a loss of at most $\frac{1}{4}\sigma_1(A)$. [Exercise (8.28) proves the loss is at most this amount.]

Since $\sum\limits_i x_i^2 = 1$, the set $S = \{i|\alpha \leq x_i < 2\alpha\}$ has $|S| \leq \frac{1}{\alpha^2}$ and similarly, $T = \{j|\beta \leq y_j \leq 2\beta\}$ has $|T| \leq \frac{1}{\beta^2}$. Thus

$$\begin{aligned}\sum_{\substack{i\\ \alpha\leq x_i\leq 2\alpha}} \sum_{\substack{j\\ \beta\leq y_j\leq 2\beta}} x_i y_j a_{ij} &\leq 4\alpha\beta A(S,T)\\ &\leq 4\alpha\beta d(S,T)\sqrt{|S||T|}\\ &\leq 4d(S,T)\\ &\leq 4d(A).\end{aligned}$$

From this it follows that

$$\sigma_1(A) \leq 4d(A)\log n \log d$$

or

$$d(A) \geq \tfrac{\sigma_1(A)}{4\log n \log d}$$

proving the second inequality.

It is also clear that for each of the values of (α, β), we can compute $A(S,T)$ and $d(S,T)$ as above and taking the best of these $d(S,T)$'s gives us an algorithm as claimed in the Theorem. ∎

Note that in many cases, the nonzero values of x_i and y_j (after zeroing out the low entries) will only go from $\frac{1}{2}\frac{1}{\sqrt{n}}$ to $\frac{c}{\sqrt{n}}$ for x_i and $\frac{1}{2}\frac{1}{\sqrt{d}}$ to $\frac{c}{\sqrt{d}}$ for y_j, since the singular vectors are likely to be balanced given that a_{ij} are all between 0 and 1. In this case, there will be $O(1)$ groups only and the log factors disappear.

Another measure of density is based on similarities. Recall that the similarity between objects represented by vectors (rows of A) is defined by their dot products. Thus, similarities are entries of the matrix AA^T. Define the average cohesion $f(S)$ of a set S of rows of A to be the sum of all pairwise dot products of rows in S divided by $|S|$. The average cohesion of A is the maximum over all subsets of rows of the average cohesion of the subset.

Since the singular values of AA^T are squares of singular values of A, we expect $f(A)$ to be related to $\sigma_1(A)^2$ and $d(A)^2$. Indeed it is. We state the following without proof.

Lemma 8.12 *$d(A)^2 \leq f(A) \leq d(A) \log n$. Also, $\sigma_1(A)^2 \geq f(A) \geq \frac{c\sigma_1(A)^2}{\log n}$.*

$f(A)$ can be found exactly using flow techniques as we will see later.

In this section, we described how to find a large global community. There is another question, that of finding a small local community including a given vertex. We will visit this question in Section 8.10.

8.9 Flow Methods

Here we consider dense induced subgraphs of a graph. An induced subgraph of a graph consisting of a subset of the vertices of the graph along with all edges of the graph that connect pairs of vertices in the subset of vertices. We show that finding an induced subgraph with maximum average degree can be done by network flow techniques. This is simply maximizing density $d(S, S)$ of Section 8.8 over all subsets S of the graph. First consider the problem of finding a subset of vertices such that the induced subgraph has average degree at least λ for some parameter λ. Then do a binary search on the value of λ until the maximum λ for which there exists a subgraph with average degree at least λ is found.

Given a graph G in which one wants to find a dense subgraph, construct a directed graph H from the given graph and then carry out a flow computation on H. H has a node for each edge of the original graph, a node for each vertex of the original graph, plus two additional nodes s and t. There is a directed edge with capacity one from s to each node corresponding to an edge of the original graph and a directed edge with infinite capacity from each node corresponding to an edge of the original graph to the two nodes corresponding to the vertices the edge connects. Finally, there is a directed edge with capacity λ from each node corresponding to a vertex of the original graph to t.

Notice there are three types of cut sets of the directed graph that have finite capacity. The first cuts all arcs from the source. It has capacity e, the number of edges of the original graph. The second cuts all edges into the sink. It has capacity λv, where v is the number of vertices of the original graph. The third cuts some arcs from s and some arcs into t. It partitions the set of vertices and the set of edges of the original graph into two

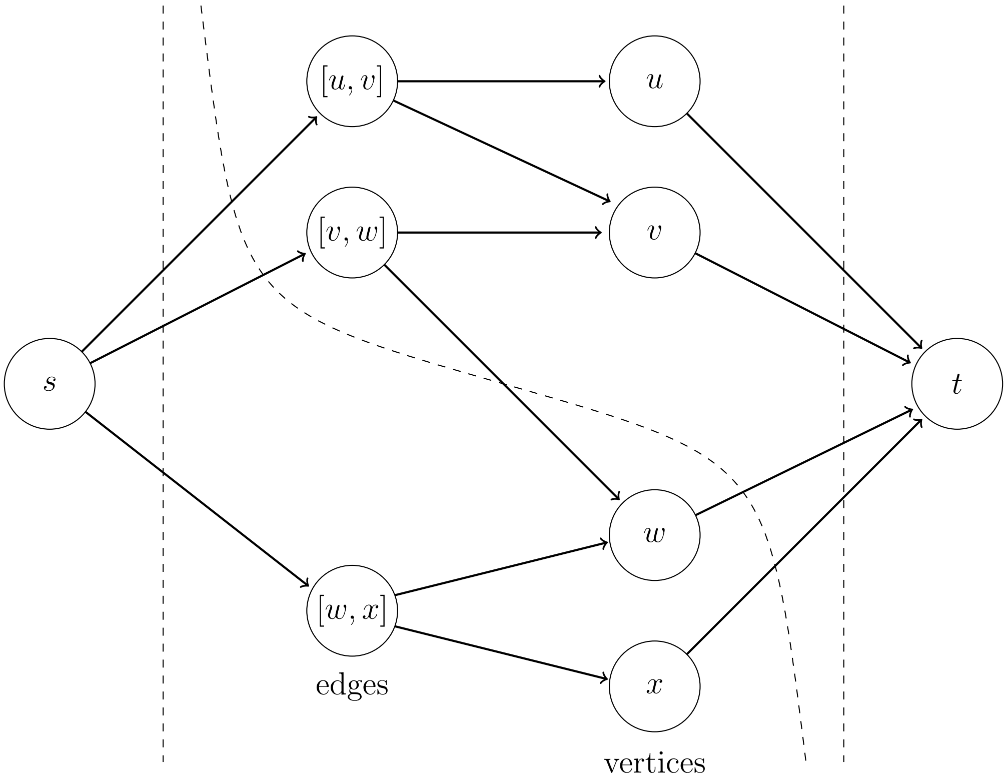


Figure 8.5: The directed graph H used by the flow technique to find a dense subgraph

blocks. The first block contains the source node s, a subset of the edges e_s, and a subset of the vertices v_s defined by the subset of edges. The first block must contain both end points of each edge in e_s; otherwise an infinite arc will be in the cut. The second block contains t and the remaining edges and vertices. The edges in this second block either connect vertices in the second block or have one endpoint in each block. The cut set will cut some infinite arcs from edges not in e_s coming into vertices in v_s. However, these arcs are directed from nodes in the block containing t to nodes in the block containing s. Note that any finite capacity cut that leaves an edge node connected to s must cut the two related vertex nodes from t. Thus, there is a cut of capacity $e - e_s + \lambda v_s$ where v_s and e_s are the vertices and edges of a subgraph. For this cut to be the minimal cut, the quantity $e - e_s + \lambda v_s$ must be minimal over all subsets of vertices of the original graph and the capcity must be less than e and also less than λv.

If there is a subgraph with v_s vertices and e_s edges where the ratio $\frac{e_s}{v_s}$ is sufficiently large so that $\frac{e_S}{v_S} > \frac{e}{v}$, then for λ such that $\frac{e_S}{v_S} > \lambda > \frac{e}{v}$, $e_s - \lambda v_s > 0$ and $e - e_s + \lambda v_s < e$. Similarly $e < \lambda v$ and thus $e - e_s + \lambda v_s < \lambda v$. This implies that the cut $e - e_s + \lambda v_s$ is less than either e or λv and the flow algorithm will find a nontrivial cut and hence a proper subset. For different values of λ in the above range there maybe different nontrivial cuts.

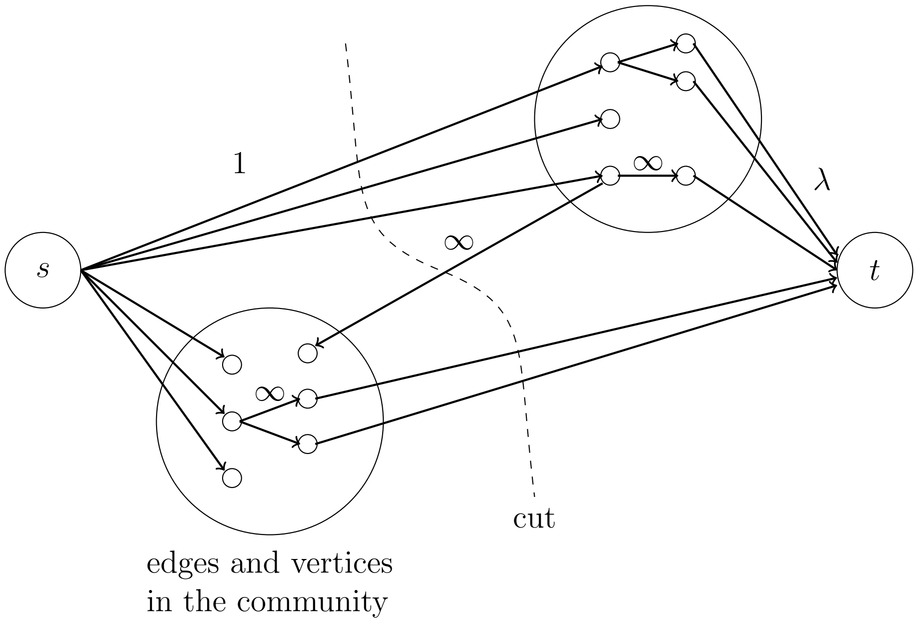


Figure 8.6: Cut in flow graph

Note that for a given density of edges, the number of edges grows as the square of the number of vertices and $\frac{e_s}{v_s}$ is less likely to exceed $\frac{e}{v}$ if v_S is small. Thus, the flow method works well in finding large subsets since it works with $\frac{e_S}{v_S}$. To find small communities one would need to use a method that worked with $\frac{e_S}{v_S^2}$ as the following example illustrates.

Example: Consider finding a dense subgraph of 1,000 vertices and 2,000 internal edges in a graph of 10^6 vertices and $6{\times}10^6$ edges. For concreteness, assume the graph was generated by the following process. First, a 1,000-vertex graph with 2,000 edges was generated as a random regular degree four graph. The 1,000-vertex graph was then augmented to have 10^6 vertices and edges were added at random until all vertices were of degree 12. Note that each vertex among the first 1,000 has four edges to other vertices among the first 1,000 and eight edges to other vertices. The graph on the 1,000 vertices is much denser than the whole graph in some sense. Although the subgraph induced by the 1,000 vertices has four edges per vertex and the full graph has twelve edges per vertex, the probability of two vertices of the 1,000 being connected by an edge is much higher than for the graph as a whole. The probability is given by the ratio of the actual number of edges connecting vertices among the 1,000 to the number of possible edges if the vertices formed a complete graph. $\frac{A(S,S)}{|S|^2}$?]

$$p = \frac{e}{\left(\binom{v}{2}\right)} = \frac{2e}{v(v-1)}$$

For the 1,000 vertices, this number is $p = \frac{2\times 2{,}000}{1{,}000\times 999} \cong 4 \times 10^{-3}$. For the entire graph this number is $p = \frac{2 \times 6 \times 10^6}{10^6 \times 10^6} = 12 \times 10^{-6}$. This difference in probability of two vertices being connected should allow us to find the dense subgraph. ■

In our example, the cut of all arcs out of s is of capacity 6×10^6, the total number of edges in the graph, and the cut of all arcs into t is of capacity λ times the number of vertices or $\lambda \times 10^6$. A cut separating the 1,000 vertices and 2,000 edges would have capacity $6 \times 10^6 - 2,000 + \lambda \times 1,000$. This cut cannot be the minimum cut for any value of λ since $\frac{e_s}{v_s} = 2$ and $\frac{e}{v} = 6$, hence $\frac{e_s}{v_s} < \frac{e}{v}$. The point is that to find the 1,000 vertices, we have to maximize $A(S,S)/|S|^2$ rather than $A(S,S)/|S|$. Note that $A(S,S)/|S|^2$ penalizes large |S| much more and therefore can find the 1,000 node "dense" subgraph.

8.10 Finding a Local Cluster Without Examining the Whole Graph

If one wishes to find the community containing a vertex v in a large graph with say a billion vertices, one would like to find the community in time proportional to the size of the community and independent of the size of the graph. Thus, we would like local methods that do not inspect the entire graph but only the neighborhood around the vertex v. We now give several such algorithms. Throughout this section, we assume the graph is undirected.

Breadth-First Search

The simplest method is to do a breadth first search starting at v. Clearly if there is a small connected component containing v, we will find it in time depending only on the size (number of edges) of the component. In a more subtle situation, each edge may have a weight that is the similarity between the two end points. If there is a small cluster C containing v, with each outgoing edge from C to $\bar{C}$ having weight less than some ε, C could clearly also be found by breadth-first search in time proportional to the size of C. However, in general, it is unlikely that the cluster will have such obvious telltale signs of its boundary and one needs more complex techniques, some of which we describe now.

By max flow

Given a vertex v in a directed graph, we want to find a small set S of vertices whose boundary (set of a few outgoing edges) is very small. Suppose we are looking for a set S whose boundary is of size at most b and whose cardinality is at most k. Clearly, if $\deg(v) < b$ then the problem is trivial, so assume $\deg(v) \geq b$.

Think of a flow problem where v is the source. Put a capacity of one on each edge of the graph. Create a new vertex that is the sink and add an edge of capacity α from each vertex of the original graph to the new sink vertex, where $\alpha = b/k$. If a community of size at most k with boundary at most b containing v exists, then there will be a cut separating v from the sink of size at most $k\alpha + b = 2b$, since the cut will have k edges from the community to the sink and b edges from the community to the remainder of the graph. Conversely, if there is a cut of size at most $2b$, then the community containing v has a boundary of size at most $2b$ and has at most $2k$ vertices since each vertex has an edge to the sink with capacity $\frac{b}{k}$. Thus, to come within a factor of two of the answer, all one needs to do is determine whether there is a cut of size at most $2b$. Since we know that the minimum size of any cut equals the maximum flow, it suffices to find the maximum flow. If the flow algorithm can do more than $2k$ flow augmentations, then the maximum flow and hence the minimum cut is of size more than $2b$. If not, the minimum cut is of size at most $2b$.

In executing the flow algorithm one finds an augmenting path from source to sink and augments the flow. Each time a new vertex not seen before is reached, there is an edge to the sink and the flow can be augmented by α directly on the path from v to the new vertex to the sink. So the amount of work done is a function of b and k, not the total number of vertices in the graph.

Sparsity and Local communities

In this part, we consider another definition of a local community. A local community in an undirected graph $G(V, E)$ is a subset of vertices with strong internal similarities and weak similarities to the outside. Using the same notation as in Section 8.5, we formalize this as follows:

Definition 8.2 *A subset S of vertices is a local community with parameter $\varepsilon > 0$ if it satisfies the following conditions:*

$$\Phi(S, \bar{S}) \leq \varepsilon^3 \tag{8.2}$$

$$\forall T \subseteq S,\ d(T) \leq \frac{1}{2} d(S), \quad \Phi(T, S \setminus T) \geq \varepsilon. \tag{8.3}$$

■

The first condition says that the connections of S to the outside $\bar{S}$ are weak. The second condition requires subsets of S of size as measured by $d(\cdot)$ less than 1/2 of the size of S to be strongly connected to the rest of S. Otherwise, S would not be one community, rather it would split into at least two. Note that for $\varepsilon << 1$, we have $\varepsilon^3 << \varepsilon$ and so the internal connections are required to be much stronger than the external ones. This is intuitively consistent with what we think of as a strong community. However, as opposed to Section 8.5, where, we spent time that grows as a function of $|V|$ since recursive clustering starts with the whole of V as one cluster, here, we will assume that $|S| << |U|$ and would like to find S in time which grows as a function of $|S|$, not $|V|$.

To accomplish this, we do a random walk on the graph starting with a vertex in S

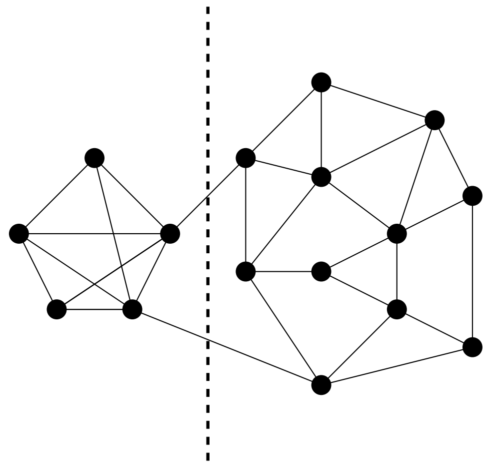


Figure 8.7: Example of a highly connected, low sparsity community.

with transition probability matrix P (See Chapter 5) given by:

$$p_{ij} = \frac{1}{d_i} \quad \text{for } j \text{ adjacent to } i.$$

Recall from Chapter 5 that the Fundamental Theorem of Markov Chains proved that the long-term average probability vector converges to a stationary probability $\boldsymbol{\pi}$ given by

$$\pi_i = \frac{d(i)}{\sum_{j \in V} d(j)},$$

assuming G is connected.

If the Markov Chain is run for long enough, the probabilities will "spread" throughout V in proportion to the degrees. This is not desirable in this context. We would rather have it be essentially confined to S, our local community. Intuitively, since S 's connection to $\bar{S}$ is at most ε^3, if we run the Markov Chain for $O(1/\varepsilon^2)$ steps, we hope to have only ε probability of stepping into $\bar{S}$. Unfortunately, this is not valid. There may be a few boundary vertices in S that have strong connections to $\bar{S}$ and if we happen to start in one of them, we might step into $\bar{S}$ right away. All we can assert is that for most starting points in S, we do not step into $\bar{S}$ in $O(1/\varepsilon^2)$ steps. We now show this.

Lemma 8.13 *Suppose condition (8.2) is satisfied. Then there is a subset S_0 of S with $\pi(S_0) \geq \frac{3}{4}\pi(S)$ such that starting the Markov Chain at any i in S_0 and running it for $t_0 \in O(1/\varepsilon^2)$ steps, the probability that we ever step into $\bar{S}$ is at most $O(\varepsilon)$.*

Proof: For $i \in S$ and $t \in O(1/\varepsilon^2)$, let $f(i,t)$ be the probability that the Markov chain started at vertex i at time 0, walks from S to a vertex in $\bar{S}$ at time t. We would like to upper bound $f(i,t)$. While this is difficult, observe that had the walk started in the stationary distribution $\boldsymbol{\pi}$, then it would remain in the stationary distribution and so the probability that it would step from a vertex in S to a vertex in $\bar{S}$ at time t is precisely $\sum\limits_{j\in S}\sum\limits_{k\in\bar{S}}\pi_j p_{jk} = \Phi(S)\pi(S)$. By linearity, this probability is $\sum\limits_{i\in S}\pi_i f(i,t)$, so

$$\sum_{i\in S}\pi_i f(i,t) = \Phi(S)\pi(S) \leq \varepsilon^3\pi(S)$$

Let f_i be the probability that when started at time 0 at i, we step at least once into $\bar{S}$ in the first $O(1/\varepsilon^2)$ steps of the Markov Chain. We get from above that:

$$\sum_{i\in S}\frac{\pi_i}{\pi(S)}f_i \leq O(\varepsilon).$$

But $\sum_{i\in S}\frac{\pi_i}{\pi(S)}f_i$ is the weighted average of f_i over $i\in S$ with weights $\pi_i/\pi(S)$ and so by Markov inequality, it follows that the weight of the set of i for which $f_i > c\varepsilon$ for a large constant c cannot exceed $1/4$ proving the lemma. ■

To discover S, besides not going out of S, we also need that the walk spreads through all, or at least most, of S which we now show.

Lemma 8.14 *Suppose conditions (8.2) and (8.3) are satisfied. Start the markov Chain in S_0 and run it for t_0 steps. Let S_1 be the set of all $i\in S$ for which the expected number of visits is at least $(3/4)\pi_i/\pi(S)t_0$. Then, $\pi(S_1)\geq(3/4)\pi(S)$.*

Proof: For $i\in S$ and $t\leq t_0$, let

$$\begin{aligned} g_{it} &= \text{Prob(Walk remains in } S \text{ and is at } i \text{ at time } t) \\ h_i &= \frac{1}{t_0}\sum_{t=0}^{t_0-1} g_{it}. \end{aligned}$$

From Lemma 8.13, we have

$$\sum_{i\in S} g_{it} = \text{Prob(Walk remains in } S) \geq 1 - O(\varepsilon) \implies \sum_{i\in S} h_i \geq 1 - O(\varepsilon). \tag{8.4}$$

Let $\tilde{P}$ be the transition probability matrix of a Markov chain with states S obtained from P by redirecting each transition (i,j) from a vertex $i\in S$ to a vertex $j\in\bar{S}$ to be a self-loop at i. More formally, $\tilde{p}_{jk} = p_{jk}$, for $j,k\in S$ and for all $i\in S$, $\tilde{p}_{ii} = 1 - \sum\limits_{j\neq i} p_{ij}$. We still have $\pi_j\tilde{p}_{jk} = \pi_k\tilde{p}_{kj}$ and so (from Chapter 5), we know that the stationary probability of the chain $\tilde{P}$ is $\frac{1}{\pi(S)}\boldsymbol{\pi}$.

Further, the conductance of $\tilde{P}$ is at least ε by (8.3). Let $\mathbf{p^{(t)}}$ denote the probabilities of the chain $\tilde{P}$ at time t and let $\mathbf{a}$ denote the long term average, i.e.,

$$\mathbf{a} = \frac{1}{t_0}\sum_{t=0}^{t_0-1}\mathbf{p^{(t)}}.$$

From Theorem 5.5 of Chapter 5, $\left|\mathbf{a} - \frac{1}{\pi(S)}\boldsymbol{\pi}\right| \leq \frac{1}{100}$, Thus

$$a(S_1) \leq \frac{\pi(S_1)}{\pi(S)} + \frac{1}{100}. \tag{8.5}$$

Now, for each $i \in S$, $g_{it} \leq p_i^{(t)}$, since every run of the walk that never steps out of S has a corresponding run in $\tilde{P}$. This is an inequality rather than an equation because, walks which would have stepped out of S are now redirected via the self-loop we created; they are counted in $p_i^{(t)}$, but not in g_{it}. So, $h_i \leq a_i$ and hence, $h(S_1) \leq a(S_1)$. Using (8.5) and (8.4):

$$h(S \setminus S_1) = h(S) - h(S_1) \geq h(S) - a(S_1) \geq 1 - O(\varepsilon) - \frac{\pi(S_1)}{\pi(S)} - \frac{1}{100} \geq \frac{\pi(S \setminus S_1)}{\pi(S)} - \frac{1}{50}.$$

But, $h(S \setminus S_1) \leq \frac{3}{4}\frac{\pi(S \setminus S_1)}{\pi(S)}$. Thus, $\pi(S \setminus S_1) \leq \frac{4}{50}$ proving the lemma. ∎

Modularity clustering

Another way to partition a graph into communities is based on the concept of modularity. The method is popular for small graphs. Consider the partition of the vertices of a graph into communities. The modularity of the partition is defined to be the fraction of edges that lie within communities minus the expected number of edges that lie within communities in a random graph with the same degree distribution. Let A be the adjacency matrix of a graph, m the total number of edges, d_v the degree of vertex v, and let i index the communities defined by a partition of the vertices. Then, the modularity is given by

$$Q = \tfrac{1}{2m} \sum_i \sum_{v,w \in i} a_{vw} - \sum_i \sum_{v,w \in i} \tfrac{d_v}{2m}\tfrac{d_w}{2m}$$

Let e_{ij} denote the fraction of the total set of edges that connect communities i and j and let a_i denote the fraction of edges with ends in community i. Then

$$e_{ij} = \tfrac{1}{2m} \sum_{\substack{v \in i \\ w \in j}} a_{vw}$$

and

$$a_i = \tfrac{1}{2m} \sum_{v \in i} d_v$$

Write

$$\begin{aligned} Q &= \sum_i \sum_{v,w \in i} \left(\tfrac{1}{2m} a_{vw} - \tfrac{d_v}{2m}\tfrac{d_w}{2m}\right) \\ &= \sum_i \left(e_{ij} - a_i^2\right) \end{aligned}$$

The algorithm for finding communities works as follows. Start with each vertex in a community. Repeatedly merge the pair of communities that maximizes the change in Q. This can be done in time $O\left(m\log^2 n\right)$ where n is the number of vertices in the graph and m is the number of edges where $m \geq n$. The algorithm works well on small graphs but is inefficient for graphs with even a few thousand vertices.

Percolation clustering

Another clustering method that is useful in clustering nodes of a graph is called percolation clustering. Here one selects a value k and creates a new graph whose vertices correspond to k-cliques in the original graph. Edges in this new graph correspond to k-1 cliques connecting the k-cliques in the original graph. The connected components of this new graph are the clusters for the original graph.

8.11 Axioms for Clustering

Each clustering algorithm tries to optimize some criterion, like the sum of squared distances to the nearest cluster center, over all possible clusterings. We have seen many different optimization criteria in this chapter and many more are used. Now, we take a step back and ask what are the desirable properties of a clustering criterion and if there are criteria satisfying these properties. Our first result is negative. We present three seemingly desirable properties of a measure, and then show that no measure satisfies them. Next we argue that these requirements are too stringent and under more reasonable requirements, a slightly modified form of the sum of Euclidean distance squared between all pairs of points inside the same cluster is indeed a measure satisfying the desired properties.

8.11.1 An Impossibility Result

Let $A(d)$ denote the optimal clustering found by the clustering algorithm A using distance function d on a set S. The clusters of the clustering $A(d)$ form a partition Γ of S.

The first desirable property of a clustering algorithm is scale invariance. A clustering algorithm A is *scale invariant* if for any $\alpha > 0$, $A(d) = A(\alpha d)$. That is, multiplying all distances by some scale factor does not change the optimal clustering. In general, there could be ties for what the algorithm returns; in that case, we adopt the convention that $A(d) = A(\alpha d)$ really means for any clustering returned by A on distance d, it can also be returned by A on distance αd.

A clustering algorithm A is *rich* (full/complete) if for every partitioning Γ there exists a distance function d such that $A(d) = \Gamma$. That is, for any desired partitioning, we can find a set of distances so that the clustering algorithm returns the desired partitioning.

A clustering algorithm is *consistent* if increasing the distance between points in different clusters and reducing the distance between points in the same cluster does not change the clusters produced by the clustering algorithm.

If a clustering algorithm is consistent and $A(d) = \Gamma$, one can find a new distance function d' such that $A(d') = \Gamma$ where there are only two distances a and b. Here a is the distance between points within a cluster and b is the distance between points in different clusters. By consistency, we can reduce all distances within clusters and increase all distances between clusters there by getting two distances a and b with $a < b$ where a is the distance between points within a cluster and b is the distance between points in different clusters.

There exist natural clustering algorithms satisfying any two of the three axioms. The *single link clustering algorithm* starts with each point in a cluster by itself and then merges the two clusters that are closest. The process continues until some stopping condition is reached. One can view the process as the points being vertices of a graph and edges being labeled by the distances between vertices. One merges the two vertices that are closest and merges parallel edges taking the distance of the merged edge to be the minimum of the two distances.

Theorem 8.15

1. *The single link clustering algorithm with the k-cluster stopping condition, stop when there are k clusters, satisfies scale-invariance and consistency. We do not get richness since we only get clustering's with k clusters.*

2. *The single link clustering algorithm with scale α stopping condition satisfies scale invariance and richness. The scale α stopping condition is to stop when the closest pair of clusters is of distance greater than or equal to $\alpha d_{\max}$ where $d_{\max}$ is the maximum pair wise distance. Here we do not get consistency. If we select one distance between clusters and increase it significantly until it becomes $d_{\max}$ and in addition $\alpha d_{\max}$ exceeds all other distances, the resulting clustering has just one cluster containing all of the points.*

3. *The single link clustering algorithm with the distance r stopping condition, stop when the inter-cluster distances are all at least r, satisfies richness and consistency; but not scale invariance.*

Proof: (1) Scale-invariance is easy to see. If one scales up all distances by a factor, then at each point in the algorithm, the same pair of clusters will be closest. The argument for consistency is more subtle. Since edges inside clusters of the optimal (final) clustering can only be decreased and since edges between clusters can only be increased, the edges that led to merges between any two clusters are less than any edge between the final clusters. Since the final number of clusters is fixed, these same edges will cause the same merges unless the merge has already occurred due to some other edge that was inside a final cluster having been shortened even more. No edge between two final clusters can cause a merge before all the above edges have been considered. At this time the final number of clusters has been reached and the process of merging has stopped.

(2) and (3) are straight forward. ■

Next, we show that no clustering algorithm can satisfy all three axioms. A distance function d, (a-b)-*conforms* to a partition Γ if all points in a cluster are within distance a of each other and all points in distinct clusters are at least distance b apart. For a clustering algorithm A, the pair of numbers (a, b) is said to force the partition Γ if all distances functions d that (a-b) conform to Γ have $A(d) = \Gamma$.

Associated with a clustering algorithm is a collection of allowable clustering's it can produce using different distance functions. We begin with a theorem stating that scale invariance and consistency imply that no allowable clustering can be a refinement of another allowable clustering.

Theorem 8.16 *If a clustering algorithm satisfies scale-invariance and consistency, then no two clustering's, one of which is a refinement of the other, can both be optimal clustering's returned by the algorithm.*

Proof: Suppose that the range of the clustering algorithm A contains two clustering's, Γ_0 and Γ_1 where Γ_0 is a refinement of Γ_1. Modify the distance functions giving rise to Γ_0 and Γ_1 so that there are only two distinct distances a_0 and b_0 for Γ_0 and a_1 and b_1 for Γ_1. Points within a cluster of Γ_0 are distance a_0 apart and points between clusters of Γ_0 are distance b_0 apart. The distances a_1 and b_1 play similar roles for Γ_1.

Let a_2 be any number less than a_1 and choose ε such that $0 < \varepsilon < a_0 a_2 b_0^{-1}$. Let d be a new distance function where

$$d\left(i,j\right) = \begin{cases} \varepsilon \text{ if } i \text{ and } j \text{ are in the same cluster of } \Gamma_0 \\ a_2 \text{ if } i \text{ and } j \text{ are in differnt clusers of } \Gamma_0 \text{ but the same cluster of } \Gamma_1 \\ b_1 \text{ if } i \text{ and } j \text{ are in different clusters of } \Gamma_1 \end{cases}$$

From $a_0 < b_0$ it follows that $a_0 b_0^{-1} < 1$. Thus $\varepsilon < a_0 a_2 b_0^{-1} < a_2 < a_1$. Since both ε and a_2 are less than a_1, it follows by consistency that $A\left(d\right) = \Gamma_1$. Let $\alpha = b_0 a_2^{-1}$. Since $\varepsilon < a_0 a_2 b_0^{-1}$ and $a_2 < a_1 < b_1$, which implies $a_2^{-1} > a_1^{-1} > b_1^{-1}$, it follows that

$$\alpha d\left(i,j\right) = \begin{cases} b_0 a_2^{-1}\varepsilon < b_0 a_2^{-1} a_0 a_2 b_0^{-1} = a_0 & \text{if } i \text{ and } j \text{ are in the same cluster of } \Gamma_0 \\ \\ b_0 a_2^{-1} a_2 = b_0 & \text{if } i \text{ and } j \text{ are in differnt clusers of } \Gamma_0 \\ & \quad \text{but the same cluster of } \Gamma_1 \\ \\ b_0 a_2^{-1} b_1 > b_0 b_1^{-1} b_1 = b_0 & \text{if } i \text{ and } j \text{ are in different clusters of } \Gamma_1 \end{cases}$$

Thus, by consistency $A(\alpha d) = \Gamma_0$. But by scale invariance $A\left(\alpha d\right) = A\left(d\right) = \Gamma_1$, a contradiction. ■

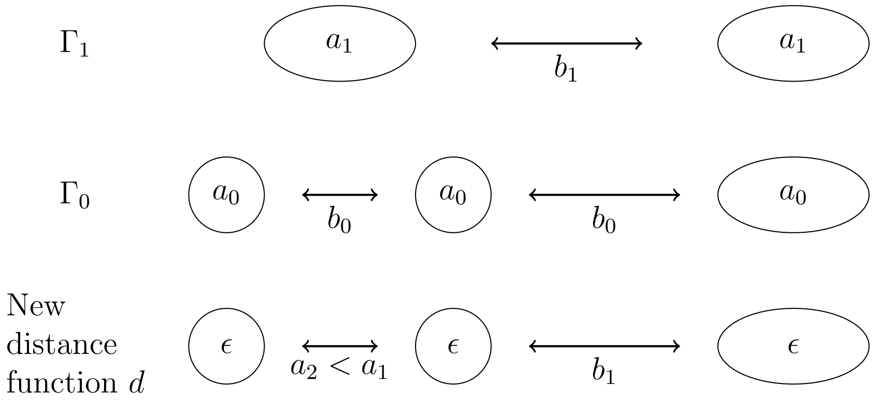


Figure 8.8: Illustration of the sets Γ_0, Γ_1, and those for the distance function d.

Corollary 8.17 *For $n \geq 2$ there is no clustering function f that satisfies scale-invariance, richness, and consistency.*

It turns out that any collection of clustering's in which no clustering is a refinement of any other clustering in the collection is the range of a clustering algorithm satisfying scale invariance and consistency. To demonstrate this, we use the sum of pairs clustering algorithm. Given a collection of clustering's, the *sum of pairs clustering* algorithm finds the clustering that minimizes the sum of all distances between points in the same cluster over all clustering's in the collection.

Theorem 8.18 *Every collection of clustering's in which no clustering is the refinement of another is the range of a clustering algorithm A satisfying scale invariance and consistency.*

Proof: We first show that the sum of pairs clustering algorithm satisfies scale invariance and consistency. Then we show that every collection of clustering's in which no cluster is a refinement of another can be achieved by a sum of pairs clustering algorithm.

Let A be the sum of pairs clustering algorithm. It is clear that A satisfies scale invariance since multiplying all distances by a constant, multiplies the total cost of each cluster by a constant and hence the minimum cost clustering is not changed.

To demonstrate that A satisfies consistency let d be a distance function and Γ the resulting clustering. Increasing the distance between pairs of points in different clusters of Γ does not affect the cost of Γ. If we reduce distances only between pairs of points in clusters of Γ then the cost of Γ is reduced as much or more than the cost of any other clustering. Hence Γ remains the lowest cost clustering.

Consider a collection of clustering's in which no cluster is a refinement of another. It remains to show that every clustering in the collection is in the range of A. In sum of

Figure 8.9: Illustration of the objection to the consistency axiom. Reducing distances between points in a cluster may suggest that the cluster be split into two.

pairs clustering, the minimum is over all clustering's in the collection. We now show for any clustering Γ how to assign distances between pairs of points so that A returns the desired clustering. For pairs of points in the same cluster assign a distance of $1/n^3$. For pairs of points in different clusters assign distance one. The cost of the clustering Γ is less than one. Any clustering that is not a refinement of Γ has cost at least one. Since there are no refinements of Γ in the collection it follows that Γ is the minimum cost clustering. ■

Note that one may question both the consistency axiom and the richness axiom. The following are two possible objections to the consistency axiom. Consider the two clusters in Figure 8.9. If one reduces the distance between points in cluster B, they might get an arrangement that should be three clusters instead of two.

The other objection, which applies to both the consistency and the richness axioms, is that they force many unrealizable distances to exist. For example, suppose the points were in Euclidean d space and distances were Euclidean. Then, there are only nd degrees of freedom. But the abstract distances used here have $O(n^2)$ degrees of freedom since the distances between the $O(n^2)$ pairs of points can be specified arbitrarily. Unless d is about n, the abstract distances are too general. The objection to richness is similar. If for n points in Euclidean d space, the clusters are formed by hyper planes each cluster may be a Voronoi cell or some other polytope, then as we saw in the theory of VC dimensions Section **??** there are only $\binom{n}{d}$ interesting hyper planes each defined by d of the n points. If k clusters are defined by bisecting hyper planes of pairs of points, there are only n^{dk^2} possible clustering's rather than the 2^n demanded by richness. If d and k are significantly less than n, then richness is not reasonable to demand. In the next section, we will see a possibility result to contrast with this impossibility theorem.

The k-means clustering algorithm is one of the most widely used clustering algorithms. We now show that any centroid based algorithm such as k-means does not satisfy the consistency axiom.

Theorem 8.19 *A centroid based clustering such as k-means does not satisfy the consistency axiom.*

Proof: The cost of a cluster is $\sum\limits_i (\mathbf{x_i} - \mathbf{u})^2$, where u is the centroid. An alternative way to compute the cost of the cluster if the distances between pairs of points in the cluster are

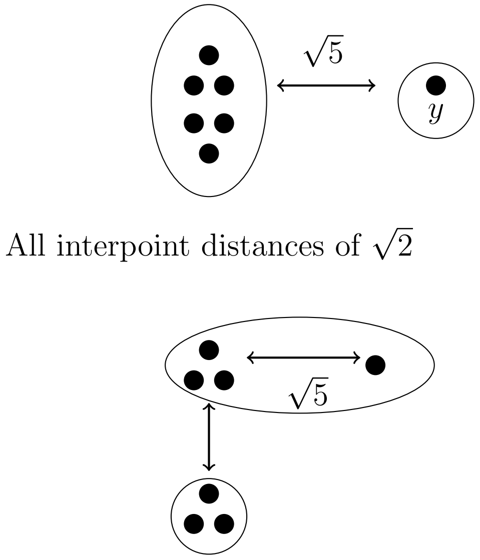


Figure 8.10: Example illustrating k-means does not satisfy the consistency axiom.

known is to compute$\frac{1}{n}\sum\limits_{i\neq j}(\mathbf{x_i}-\mathbf{x_j})^2$ where n is the number of points in the cluster. For a proof see Lemma 8.2. Consider seven points, a point $\mathbf{y}$ and two sets of three points each, called X_0 and X_1. Let the distance from $\mathbf{y}$ to each point in $X_0 \cup X_1$ be $\sqrt{5}$ and let all other distances between pairs of points be $\sqrt{2}$. These distances are achieved by placing each point of X_0 and X_1 a distance one from the origin along a unique coordinate and placing $\mathbf{y}$ at distance two from the origin along another coordinate. Consider a clustering with two clusters (see Figure 8.9). The cost depends only on how many points are grouped with $\mathbf{y}$. Let that number be m. The cost is

$$\frac{1}{m+1}\left[2\binom{m}{2}+5m\right]+\frac{2}{6-m}\binom{6-m}{2}=\frac{8m+5}{m+1}$$

which has its minimum at $m=0$. That is, the point $\mathbf{y}$ is in a cluster by itself and all other points are in a second cluster.

If we now shrink the distances between points in X_0 and points in X_1 to zero, the optimal clustering changes. If the clusters were $X_0 \cup X_1$ and $\mathbf{y}$, then the distance would be $9 \times 2 = 18$ whereas if the clusters are $X_0 \cup \{\mathbf{y}\}$ and X_1, the distance would be only $3 \times 5 = 15$. Thus, the optimal clustering is $X_0 \cup \{\mathbf{y}\}$ and X_1. Hence k-means does not satisfy the consistency axiom since shrinking distances within clusters changes the optimal clustering. ∎

5 **Relaxing the axioms**

Given that no clustering algorithm can satisfy scale invariance, richness, and consistency, one might want to relax the axioms in some way. Then one gets the following results.

1. Single linkage with a distance stopping condition satisfies a relaxed scale-invariance property that states that for $\alpha > 1$, then $f\,(\alpha d)$ is a refinement of $f(d)$.

2. Define *refinement consistency* to be that shrinking distances within a cluster or expanding distances between clusters gives a refinement of the clustering. Single linkage with α stopping condition satisfies scale invariance, refinement consistency and richness except for the trivial clustering of all singletons.

8.11.2 A Satisfiable Set of Axioms

In this section, we propose a different set of axioms that are reasonable for distances between points in Euclidean space and show that the clustering measure, the sum of squared distances between all pairs of points in the same cluster, slightly modified, is consistent with the new axioms. We assume through the section that points are in Euclidean d-space. Our three new axioms follow.

We say that a clustering algorithm satisfies the *consistency condition* if, for the clustering produced by the algorithm on a set of points, moving a point so that its distance to any point in its own cluster is not increased and its distance to any point in a different cluster is not decreased, then the algorithm returns the same clustering after the move.

Remark: Although it is not needed in the sequel, it is easy to see that for an infinitesimal perturbation dx of x, the perturbation is consistent if and only if each point in the cluster containing x lies in the half space through x with dx as the normal and each point in a different cluster lies in the other half space.

An algorithm is *scale-invariant* if multiplying all distances by a positive constant does not change the clustering returned.

An algorithm has the *richness* property if for any set K of k distinct points in the ambient space, there is some placement of a set S of n points to be clustered so that the algorithm returns a clustering with the points in K as centers. So there are k clusters, each cluster consisting of all points of S closest to one particular point of K.

We will show that the following algorithm satisfies these three axioms.

Balanced k-means algorithm

> Among all partitions of the input set of n points into k sets, each of size n/k, return the one that minimizes the sum of squared distances between all pairs of points in the same cluster.

Theorem 8.20 *The balanced k-means algorithm satisfies the consistency condition, scale invariance, and the richness property.*

Proof: Scale invariance is obvious. Richness is also easy to see. Just place n/k points of S to coincide with each point of K. To prove consistency, define the *cost* of a cluster T to be the sum of squared distances of all pairs of points in T.

Suppose $S_1, S_2, \ldots, S_k$ is an optimal clustering of S according to the balanced k-means algorithm. Move a point $x \in S_1$ to z so that its distance to each point in S_1 is non increasing and its distance to each point in $S_2, S_3, \ldots, S_k$ is non decreasing. Suppose $T_1, T_2, \ldots, T_k$ is an optimal clustering after the move. Without loss of generality assume $z \in T_1$. Define $\tilde{T}_1 = (T_1 \setminus \{z\}) \cup \{x\}$ and $\tilde{S}_1 = (S_1 \setminus \{x\}) \cup \{z\}$. Note that $\tilde{T}_1, T_2, \ldots, T_k$ is a clustering before the move, although not necessarily an optimal clustering. Thus

$$\operatorname{cost}\left(\tilde{T}_1\right) + \operatorname{cost}(T_2) + \cdots + \operatorname{cost}(T_k) \geq \operatorname{cost}(S_1) + \operatorname{cost}(S_2) + \cdots + \operatorname{cost}(S_k).$$

If $\operatorname{cost}(T_1) - \operatorname{cost}\left(\tilde{T}_1\right) \geq \operatorname{cost}\left(\tilde{S}_1\right) - \operatorname{cost}(S_1)$ then

$$\operatorname{cost}(T_1) + \operatorname{cost}(T_2) + \cdots + \operatorname{cost}(T_k) \geq \operatorname{cost}\left(\tilde{S}_1\right) + \operatorname{cost}(S_2) + \cdots + \operatorname{cost}(S_k).$$

Since $T_1, T_2, \ldots, T_k$ is an optimal clustering after the move, so also must be $\tilde{S}_1, S_2, \ldots, S_k$ proving the theorem.

It remains to show that $\operatorname{cost}(T_1) - \cos t\left(\tilde{T}_1\right) \geq \operatorname{cost}\left(\tilde{S}_1\right) - \operatorname{cost}(S_1)$. Let u and v stand for elements other than x and z in S_1 and T_1. The terms $|u-v|^2$ are common to T_1 and $\tilde{T}_1$ on the left hand side and cancel out. So too on the right hand side. So we need only prove

$$\sum_{u \in T_1} (|z-u|^2 - |x-u|^2) \geq \sum_{u \in S_1} (|z-u|^2 - |x-u|^2).$$

For $u \in S_1 \cap T_1$, the terms appear on both sides, and we may cancel them, so we are left to prove

$$\sum_{u \in T_1 \setminus S_1} (|z-u|^2 - |x-u|^2) \geq \sum_{u \in S_1 \setminus T_1} (|z-u|^2 - |x-u|^2)$$

which is true because by the movement of x to z, each term on the left hand side is non negative and each term on the right hand side is non positive. ■

8.12 Exercises

Exercise 8.1 *Construct examples where using distances instead of distance squared gives bad results for Gaussian densities. For example, pick samples from two 1-dimensional unit variance Gaussians, with their centers 10 units apart. Cluster these samples by trial and error into two clusters, first according to k-means and then according to the k-median criteria. The k-means clustering should essentially yield the centers of the Gaussians as cluster centers. What cluster centers do you get when you use the k-median criterion?*

Exercise 8.2 *Let* $v = (1, 3)$*. What is the* L_1 *norm of* v*? The* L_2 *norm? The square of the* L_1 *norm?*

Exercise 8.3 *Show that in 1-dimension, the center of a cluster that minimizes the sum of distances of data points to the center is in general not unique. Suppose we now require the center also to be a data point; then show that it is the median element (not the mean). Further in 1-dimension, show that if the center minimizes the sum of squared distances to the data points, then it is unique.*

Exercise 8.4 *Construct a block diagonal matrix* A *with three blocks of size 50. Each matrix element in a block has value* $p = 0.7$ *and each matrix element not in a block has value* $q = 0.3$*. generate a* 150×150 *matrix* B *of random numbers in the range [0,1]. If* $b_{ij} \geq a_{ij}$ *replace* a_{ij} *with the value one. Otherwise replace* a_{ij} *with value zero. The rows of* A *have three natural clusters. Permute the rows and columns of* A *so the first 50 rows do not form the first cluster, the next 50 the second cluster, and the last 50 the third cluster.*

1. *Apply the k-mean algorithm to* A *with* $k = 3$*. Do you find the correct clusters?*
2. *Apply the k-means algorithm to* A *for* $1 \leq k \leq 10$*. Plot the value of the sum of squares to the cluster centers versus* k*. Was three the correct value for* k*?*

Exercise 8.5 *Let* M *be a* $k \times k$ *matrix whose elements are numbers in the range [0,1]. A matrix entry close to one indicates that the row and column of the entry correspond to closely related items and an entry close to zero indicates unrelated entities. Develop an algorithm to match each row with a closely related column where a column can be matched with only one row.*

Exercise 8.6 *The simple greedy algorithm of Section 8.3 assumes that we know the clustering radius* r*. Suppose we do not. Describe how we might arrive at the correct* r*?*

Exercise 8.7 *For the k-median problem, show that there is at most a factor of two ratio between the optimal value when we either require all cluster centers to be data points or allow arbitrary points to be centers.*

Exercise 8.8 *For the k-means problem, show that there is at most a factor of four ratio between the optimal value when we either require all cluster centers to be data points or allow arbitrary points to be centers.*

Exercise 8.9 *Consider clustering points in the plane according to the k-median criterion, where cluster centers are required to be data points. Enumerate all possible clustering's and select the one with the minimum cost. The number of possible ways of labeling n points, each with a label from $\{1, 2, \ldots, k\}$ is k^n which is prohibitive. Show that we can find the optimal clustering in time at most a constant times $\binom{n}{k} + k^2$. Note that $\binom{n}{k} \leq n^k$ which is much smaller than k^n when $k << n$.*

Exercise 8.10 *Suppose in the previous exercise, we allow any point in space (not necessarily data points) to be cluster centers. Show that the optimal clustering may be found in time at most a constant times n^{2k^2}.*

Exercise 8.11 *Corollary 8.3 shows that for a set of points $\{a_1, a_2, \ldots, a_n\}$, there is a unique point x, namely their centroid, which minimizes $\sum\limits_{i=1}^{n} |a_i - x|^2$. Show examples where the x minimizing $\sum\limits_{i=1}^{n} |a_i - x|$ is not unique. (Consider just points on the real line.) Show examples where the x defined as above are far apart from each other.*

Exercise 8.12 *Let $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ be a set of unit vectors in a cluster. Let $\mathbf{c} = \frac{1}{n} \sum\limits_{i=1}^{n} \mathbf{a_i}$ be the cluster centroid. The centroid $\mathbf{c}$ is not in general a unit vector. Define the similarity between two points $\mathbf{a}_i$ and $\mathbf{a}_j$ as their dot product. Show that the average cluster similarity $\frac{1}{n^2} \sum\limits_{i,j} \mathbf{a_i}\mathbf{a_j}^T$ is the same whether it is computed by averaging all pairs or computing the average similarity of each point with the centroid of the cluster.*

Exercise 8.13 *For some synthetic data estimate the number of local minima for k-means by using the birthday estimate. Is your estimate an unbaised estimate of the number? an upper bound? a lower bound? Why?*

Exercise 8.14 *Examine the example in Figure and discuss how to fix it. Optimizing according to the k-center or k-median criteria would seem to produce clustering B while clustering A seems more desirable.*

Exercise 8.15 *Prove that for any two vectors $\mathbf{a}$ and $\mathbf{b}$, $|\mathbf{a} - \mathbf{b}|^2 \geq \frac{1}{2}|\mathbf{a}|^2 - |\mathbf{b}|^2$.*

Exercise 8.16 *Let A be an $n \times d$ data matrix, B its best rank k approximation, and C the optimal centers for k-means clustering of rows of A. How is it possible that $\|A - B\|_F^2 < \|A - C\|_F^2$?*

Exercise 8.17 *Suppose S is a finite set of points in space with centroid $\mu(S)$. If a set T of points is added to S, show that the centroid $\mu(S \cup T)$ of $S \cup T$ is at distance at most $\frac{|T|}{|S|+|T|} |\mu(T) - \mu(S)|$ from μ.*

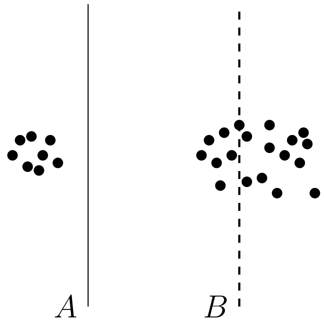


Figure 8.11: insert caption

Exercise 8.18 *What happens if we relax this restriction, for example, if we allow for S, the entire set?*

Exercise 8.19 *Given the graph $G = (V, E)$ of a social network where vertices represent individuals and edges represent relationships of some kind, one would like to define the concept of a community. A number of deferent definitions are possible.*

1. *A subgraph $S = (V_S, E_S)$ whose density $\frac{E_S}{V_S^2}$ is greater than that of the graph $\frac{E}{V^2}$.*

2. *A subgraph S with a low conductance like property such as the number of graph edges leaving the subgraph normalized by the minimum size of S or $V - S$ where size is measured by the sum of degrees of vertices in S or in $V - S$.*

3. *A subgraph that has more internal edges than in a random graph with the same degree distribution.*

Which would you use and why?

Exercise 8.20 *A stochastic matrix is a matrix with non negative entries in which each row sums to one. Show that for a stochastic matrix, the largest eigenvalue is one. Show that the eigenvalue has multiplicity one if and only if the corresponding Markov Chain is connected.*

Exercise 8.21 *Show that if P is a stochastic matrix and π satisfies $\pi_i p_{ij} = \pi_j p_{ji}$, then for any left eigenvector $\mathbf{v}$ of P, the vector $\mathbf{u}$ with components $u_i = \frac{v_i}{\pi_i}$ is a right eigenvector with the same eigenvalue.*

Exercise 8.22 *In Theorem (**??**), how can one clustering $C^{(0)}$ be close to any proper clustering? What if there are several proper clusterings?*

Exercise 8.23 *Give an example of a clustering problem where the clusters are not linearly separable in the original space, but are separable in a higher dimensional space.*
***Hint**: Look at the example for Gaussian kernels in the chapter on learning.*

Exercise 8.24 *The Gaussian kernel maps points to a higher dimensional space. What is this mapping?*

Exercise 8.25 *Agglomerative clustering requires that one calculate the distances between all pairs of points. If the number of points is a million or more, then this is impractical. One might try speeding up the agglomerative clustering algorithm by maintaining a 100 clusters at each unit of time. Start by randomly selecting a hundred points and place each point in a cluster by itself. Each time a pair of clusters is merged randomly select one of the remaining data points and create a new cluster containing that point. Suggest some other alternatives.*

Exercise 8.26 *Let A be the adjacency matrix of an undirected graph. Let $d(S,S) = \frac{A(S,S)}{|S|}$ be the density of the subgraph induced by the set of vertices S. Prove that $d\,(S,S)$ is the average degree of a vertex in S.*

Exercise 8.27 *Suppose A is a matrix with non negative entries. Show that $A(S,T)/(|S||T|)$ is maximized by the single edge with highest a_{ij}.*

Exercise 8.28 *Suppose A is a matrix with non negative entries and*

$$\sigma_1(A) = \mathbf{x}^T A\mathbf{y} = \sum_{i,j} x_i a_{ij} y_j, \qquad |\mathbf{x}| = |\mathbf{y}| = 1.$$

Zero out all x_i less than $1/2\sqrt{n}$ and all y_j less than $1/2\sqrt{d}$. Show that the loss is no more than ${}^1\!/\!_4{}^{th}$ of $\sigma_1(A)$.

Exercise 8.29 *Consider other measures of density such as $\frac{A(S,T)}{|S|^\rho |T|^\rho}$ for different values of ρ. Discuss the significance of the densest subgraph according to these measures.*

Exercise 8.30 *Let A be the adjacency matrix of an undirected graph. Let M be the matrix whose ij^{th} element is $a_{ij} - \frac{d_i d_j}{2m}$. Partition the vertices into two groups S and $\bar{S}$. Let s be the indicator vector for the set S and let $\bar{s}$ be the indicator variable for $\bar{S}$. Then $s^T M s$ is the number of edges in S above the expected number given the degree distribution and $s^T M \bar{s}$ is the number of edges from S to $\bar{S}$ above the expected number given the degree distribution. Prove that if $s^T M s$ is positive $s^T M \bar{s}$ must be negative.*

Exercise 8.31 *Which of the three axioms, scale invariance, richness, and consistency are satisfied by the following clustering algorithms.*

1. *$k-$means*

2. *Spectral Clustering.*

Exercise 8.32 ***(Research Problem)**: What are good measures of density that are also effectively computable? Is there empirical/theoretical evidence that some are better than others?*

9 Topic Models, Hidden Markov Process, Graphical Models, and Belief Propagation

In the chapter on learning and VC dimension, we saw many model-fitting problems. There we were given labeled data and simple classes of functions: half-spaces, support vector machines, etc. The problem was to fit the best model from a class of functions to the data. Model fitting is of course more general and in this chapter we discuss some useful models. These general models are often computationally infeasible, in the sense that they do not admit provably efficient algorithms. Nevertheless, data often falls into special cases of these models that can be solved efficiently.

9.1 Topic Models

A *topic model* is a model for representing a large collection of documents. Each document is viewed as a combination of topics and each topic has a set of word frequencies. For a collection of news articles over a period, the topics may be politics, sports, science, etc. For the topic politics, the words like "president" and "election" may have high frequencies and for the topic sports, words like "batter" and "goal" may have high frequencies. A news item document may be 60% on politics and 40% on sports. The word frequencies in the document will be convex combinations of word frequencies for the topics, politics and sports, with weights 0.6 and 0.4 respectively. We describe this more formally with vectors and matrices.

Each document is viewed as a "bag of words". We disregard the order and context in which each word occurs in the document and instead only list the frequency of occurrences of each term. Frequency is the number of occurrences of the term divided by the total number of all terms in the document. Discarding context information may seem wasteful, but this approach works well in practice and is widely used. Each document is an n-dimensional vector where n is the total number of different terms in all the documents in the collection. Each component of the vector is the frequency of a particular term in the document. Terms are words or phrases. Not all words are chosen as terms; articles, simple verbs, and pronouns like "a", "is", and "it" may be ignored. Represent the collection of documents by a $n \times m$ matrix A, called the *term-document* matrix, with one column per document in the collection. The topic model hypothesizes that there are r topics and each of the m documents is a combination of topics. The number of topics r is usually much smaller than the number of terms n. So corresponding to each document, there is a vector with r components telling us the fraction of the document that is on each of the topics. In the example above, this vector will have 0.6 in the component for politics and 0.4 in the component for sports. Arrange these vectors as the columns of a $r \times m$ matrix C, called the *topic-document* matrix. There is a third matrix B which is $n \times r$. Each column of B corresponds to a topic; each component of the column gives the frequency of a term in that topic. In the simplest model, the term frequencies in documents are exact combinations of term frequencies in the various topics that make up the document. So,

a_{ij}, the frequency of the i^{th} term in the j^{th} document is the sum over all topics l of the fraction of document j which is on topic l times the frequency of term i in topic l. In matrix notation,

$$A = BC.$$

Pictorially, we can represent this as:

$$\begin{array}{cc} & \text{DOCUMENT} \\ \begin{array}{c}\text{T}\\ \text{E}\\ \text{R}\\ \text{M}\end{array} & \left(\begin{array}{c} \\ \\ A \\ n \times m \\ \\ \\ \end{array}\right) \end{array} = \begin{array}{cc} & \text{TOPIC} \\ \begin{array}{c}\text{T}\\ \text{E}\\ \text{R}\\ \text{M}\end{array} & \left(\begin{array}{c} \\ \\ B \\ n \times r \\ \\ \\ \end{array}\right) \end{array} \begin{array}{cc} & \text{DOCUMENT} \\ \begin{array}{c}\text{T}\\ \text{O}\\ \text{P}\\ \text{I}\\ \text{C}\end{array} & \left(\begin{array}{c} C \\ r \times m \end{array}\right) \end{array}$$

This model is too simple to be realistic since the frequency of each term in the document is unlikely to be exactly what is given by the equation $A = BC$. So, a more sophisticated stochastic model is used in practice.

From the document collection we observe the $n \times m$ matrix A. Can we find B and C such that $A = BC$? The top r singular vectors from a singular value decomposition of A give a factorization BC. But there are additional constraints stemming from the fact that frequencies of terms in one particular topic are nonnegative reals summing to one and from the fact that the fraction of each topic a particular document is on are also nonnegative reals summing to one. Altogether the constraints are:

1. $A = BC$ and $\sum\limits_i a_{ij} = 1$.

2. The entries of B and C are all non-negative.

3. Each column of B and each column of C sums to one.

Given the first two conditions, we can achieve the third by multiplying the i^{th} column of B by a positive real number and dividing the i^{th} row of C by the same real number without violating $A = BC$. By doing this, one may assume that each column of B sums to one. Since $\sum_i a_{ij}$ is the total frequency of all terms in document j, $\sum_i a_{ij} = 1$. Now $a_{ij} = \sum_k b_{ik}c_{kj}$ implies $\sum_i a_{ij} = \sum_{i,k} b_{ik}c_{kj} = \sum_k c_{kj} = 1$. Thus, the columns of C also sum to one.

The problem can be posed as one of factoring the given matrix A into the product of two matrices with nonnegative entries called nonnegative matrix factorization.

Nonnegative matrix factorization (NMF) Given an $n \times m$ matrix A and an integer r, determine whether there is a factorization of A into XY where, X is an $n \times r$ matrix with nonnegative entries and Y is $r \times m$ matrix with nonnegative entries and if so, find such a factorization.

Nonnegative matrix factorization is a more general problem than topic modeling and there are many heuristic algorithms to solve the problem. But in general, they suffer from one of two problems, they can get stuck at local optima that are not solutions or take exponential time. In fact, the general NMF problem is NP-hard. In practice, often r is much smaller than n and m. We first show that while the NMF problem as formulated above is a nonlinear problem in $r(n + m)$ unknown entries of X and Y, it can be reformulated as a nonlinear problem with just $2r^2$ unknowns under the simple nondegeneracy assumption that A has rank r. Think of r as say, 25, while n and m are in the tens of thousands to see why this is useful.

Lemma 9.1 *If A has rank r, then the NMF problem can be formulated as a problem with $2r^2$ unknowns.*

Proof: If $A = XY$, then each row of A is a linear combination of the rows of Y. So the space spanned by the rows of A must be contained in the space spanned by the rows of Y. The latter space has dimension at most r, while the former has dimension r. So they must be equal. Thus, every row of Y must be a linear combination of the rows of A. Choose any set of r independent rows of A to form a $r \times m$ matrix A_1. Then $Y = SA_1$ for some $r \times r$ matrix S. By analogous reasoning, if A_2 is a $n \times r$ matrix of r independent columns of A, there is a $r \times r$ matrix T such that $X = A_2T$. Now we can easily cast NMF in terms of the unknowns S and T.

$$A = A_2TSA_1 \qquad (SA_1)_{ij} \geq 0 \qquad (A_2T)_{kl} \geq 0 \qquad \forall i, j, k, l.$$

■

It remains to solve the nonlinear problem in the $2r^2$ variables. There is a classical algorithm that solves such problems in time exponential in r^2 and polynomial in the other parameters. In fact, there is a logical theory, called the theory of reals of which this is a special case and any problem in the theory of reals can be solved in time exponential only in the number of variables. We do not give details here.

Besides the special case when r is small, there is another important case of NMF in the topic modeling application that can be solved. This is the case when there are *anchor terms*. An anchor term for a topic is a term that occurs in the topic and does not occur in any other topic. For example, the term "batter" may an anchor term for the topic baseball and "election" for the topic politics. Consider the case when each topic has an anchor term. In matrix notation, this assumes that for each column of the term-topic matrix B, there is a row whose sole nonzero entry is in that column. In this case, it is easy to see that each row of the topic-document matrix C has a scaled copy of it occurring as a row of the given term-document matrix A. Here is an illustrative diagram:

$$\begin{pmatrix} 0.3 \times c_4 \\ \\ A \\ 0.2 \times c_2 \\ \\ \\ \end{pmatrix} = \begin{array}{r} \text{election} \\ \\ \text{batter} \\ \\ \\ \end{array} \begin{pmatrix} 0 & 0 & 0 & 0.3 \\ & & B & \\ 0 & 0.2 & 0 & 0 \\ & & & \\ & & & \\ \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}.$$

If we knew which rows of A were copies of rows of C, called special rows of A, we could find C. Once C is known, we could solve the linear equations and inequalities, $A = BC$ and $b_{ij} \geq 0$, to get B. The following lemma shows that after making one modification, we can find the rows of A that are special. Suppose a row of C is a nonnegative linear combination of the other rows of C. Eliminate that row of C as well as the corresponding column of B, suitably modifying the other columns of B, maintaining $A = BC$. For example, if row 5 of C equals 4 times row 3 of C plus 3 times row 6 of C, then we can delete row 5 of C, then add 4 times column 5 of B to column 3 of B and add 3 times column 5 of B to column 6 of B and delete column 5 of B.

Repeating this until each row of C is positively independent of the other rows of C, i.e., it cannot be expressed as a nonnegative linear combination of the other rows. We still have a scaled copy of each row of C in A. Further, the other rows of A are all nonnegative linear combinations of rows of C and thus are nonnegative linear combinations of the special rows of A.

Lemma 9.2 *Suppose A has a factorization $A = BC$, where the rows of C are positively independent and for each column of B, there is a row that has its sole nonzero entry in that column. Then there is a scaled copy of each row of C in A and furthermore, the rows of A that are scaled copies of rows of C are precisely the rows of A that are positively independent of other rows of A. These rows can be identified by solving a linear program, one program per row.*

Proof: The set of special rows of A can be identified by solving n linear programming problems. Check each row of A to see if it is positively independent of all other rows. Denote by $\mathbf{a_i}$ the i th row of A. Then, the i^{th} row is positively dependent upon the others if and only if there are real numbers $x_1, x_2, \ldots x_{i-1}, x_{i+1}, \ldots x_n$ such that

$$\sum_{j \neq i} x_j \mathbf{a_j} = \mathbf{a_i}, \quad x_j \geq 0.$$

This is a linear program. ■

As we remarked earlier, the equation $A = BC$ will not hold exactly. A more practical model views A as a matrix of probabilities rather than exact frequencies. In this model, each document is generated by picking its terms in independent trials. Each trial for document j picks term 1 with probability a_{1j}; term 2 with probability a_{2j}, etc. We are not given entire documents; instead we are given s independent trials for each document. Our job is to find B and C. We do not discuss the details of either the model or the algorithms. In this new situation, algorithms are known to find B and C when there exist anchor terms, even with a small number s of trials.

At the heart of such an algorithm is the following problem:

Approximate NMF Given a $n \times m$ matrix A and the promise that there is a $n \times r$ matrix B and a $r \times m$ matrix C, both with nonnegative entries, such that $||A - BC||_F \leq \Delta$, find B' and C' of the same dimensions, with nonnegative entries such that $||A - B'C'||_F \leq \Delta'$.

Here, Δ' is related to Δ and if the promise does not hold, the algorithm is allowed to return any answer.

Now for the case when anchor words exist, this reduces to the problem of finding which rows of A have the property that no point close to the row is positively dependent on other rows. It is easy to write the statement that there is a vector $\mathbf{y}$ close to $\mathbf{a_i}$ which is positively dependent on the other rows as a convex program:

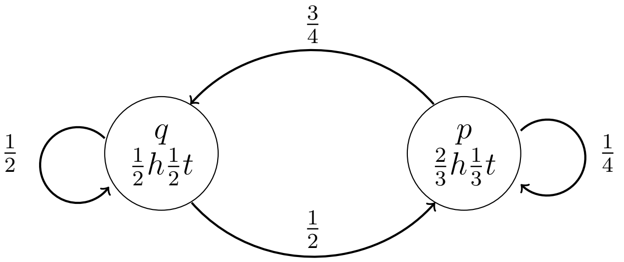
$$\exists x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \text{ such that } \left| \sum_{j \neq i} x_j \mathbf{a_j} - \mathbf{a_i} \right| \leq \varepsilon.$$

$|\sum_{j \neq i} x_j \mathbf{a_j} - \mathbf{a_i}|$ is convex function of x_j and hence this problem can be solved efficiently.

9.2 Hidden Markov Model

A *hidden Markov model*, HMM, consists of a finite set of states with a transition between each pair of states. There is an initial probability distribution α on the states and a transition probability a_{ij} associated with the transition from state i to state j. Each state has a probability distribution $p(O, i)$ giving the probability of outputting the symbol O in state i. A transition consists of two components. A state transition to a new state followed by the output of a symbol. The HMM starts by selecting a start state according to the distribution α and outputting a symbol.

Example: An example of a HMM is the graph with two states q and p illustrated below.



The initial distribution is $\alpha(q) = 1$ and $\alpha(p) = 0$. At each step a change of state occurs followed by the output of heads or tails with probability determined by the new state.

■

We consider three problems in increasing order of difficulty. First, given a HMM what is the probability of a given output sequence? Second, given a HMM and an output sequence, what is the most likely sequence of states? And third, knowing that the HMM has at most n states and given an output sequence, what is the most likely HMM? Only the third problem concerns a "hidden" Markov model. In the other two problems, the model is known and the questions can be answered in polynomial time using dynamic programming. There is no known polynomial time algorithm for the third question.

How probable is an output sequence

Given a HMM, how probable is the output sequence $O = O_0O_1O_2\cdots O_T$ of length $T+1$? To determine this, calculate for each state i and each initial segment of the sequence of observations, $O_0O_1O_2\cdots O_t$ of length $t+1$, the probability of observing $O_0O_1O_2\cdots O_t$ ending in state i. This is done by a dynamic programming algorithm starting with $t = 0$ and increasing t. For $t = 0$ there have been no transitions. Thus, the probability of observing O_0 ending in state i is the initial probability of starting in state i times the probability of observing O_0 in state i. The probability of observing $O_0O_1O_2\cdots O_t$ ending in state i is the sum of the probabilities over all states j of observing $O_0O_1O_2\cdots O_{t-1}$ ending in state j times the probability of going from state j to state i and observing O_t. The time to compute the probability of a sequence of length T when there are n states is $O(n^2T)$. The factor n^2 comes from the calculation for each time unit of the contribution from each possible previous state to the probability of each possible current state. The space complexity is $O(n)$ since one only needs to remember the probability of reaching each state for the most recent value of t.

Algorithm to calculate the probability of the output sequence

The probability, $\text{Prob}(O_0O_1\cdots O_T, i)$ of the output sequence $O_0O_1\cdots O_T$ ending in state i is given by

$\text{Prob}(O_0, i) = \alpha(i)p(O_0, i)$

for $t = 1$ to T

$$\text{Prob}(O_0O_1\cdots O_t, i) = \sum_j \text{Prob}(O_0O_1\cdots O_{t-1}, j)a_{ij}p(O_{t+1}, i)$$

Example: What is the probability of the sequence hhht by the HMM in the above example?

$t = 3$	$\frac{3}{32}\frac{1}{2}\frac{1}{2} + \frac{5}{72}\frac{3}{4}\frac{1}{2} = \frac{19}{384}$	$\frac{3}{32}\frac{1}{2}\frac{1}{3} + \frac{5}{72}\frac{1}{4}\frac{1}{3} = \frac{37}{64\times 27}$
$t = 2$	$\frac{1}{8}\frac{1}{2}\frac{1}{2} + \frac{1}{6}\frac{3}{4}\frac{1}{2} = \frac{3}{32}$	$\frac{1}{8}\frac{1}{2}\frac{2}{3} + \frac{1}{6}\frac{1}{4}\frac{2}{3} = \frac{5}{72}$
$t = 1$	$\frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8}$	$\frac{1}{2}\frac{1}{2}\frac{2}{3} = \frac{1}{6}$
$t = 0$	$\frac{1}{2}$	0
	q	p

For $t = 0$, the q entry is 1/2 since the probability of being in state q is one and the probability of outputting heads is $\frac{1}{2}$. The entry for p is zero since the probability of starting in state p is zero. For $t = 1$, the q entry is $\frac{1}{8}$ since for $t = 0$ the q entry is $\frac{1}{2}$ and in state q the HMM goes to state q with probability $\frac{1}{2}$ and outputs heads with probability $\frac{1}{2}$. The p entry is $\frac{1}{6}$ since for $t = 0$ the q entry is $\frac{1}{2}$ and in state q the HMM goes to state p with probability $\frac{1}{2}$ and outputs heads with probability $\frac{2}{3}$. For $t = 2$, the q entry is $\frac{3}{32}$ which consists of two terms. The first term is the probability of ending in state q at $t = 1$ times the probability of staying in q and outputting h. The second is the probability of ending in state p at $t = 1$ times the probability of going from state p to state q and outputting h.

From the table, the probability of producing the sequence hhht is $\frac{19}{384} + \frac{37}{1728} = 0.0709$.

■

The most likely sequence of states - the Viterbi algorithm

Given a HMM and an observation $O = O_0O_1\cdots O_T$, what is the most likely sequence of states? The solution is given by the Viterbi algorithm, which is a slight modification to the dynamic programming algorithm just given for determining the probability of an output sequence. For $t = 0, 1, 2, \ldots, T$ and for each state i, calculate the probability of the most likely sequence of states to produce the output $O_0O_1O_2\cdots O_t$ ending in state i. For each value of t, calculate the most likely sequence of states by selecting over all states j the most likely sequence producing $O_0O_1O_2\cdots O_t$ and ending in state i consisting of the most likely sequence producing $O_0O_1O_2\cdots O_{t-1}$ ending in state j followed by the transition from j to i producing O_t. Note that in the previous example, we added the probabilities of each possibility together. Now we take the maximum and also record where the maximum came from. The time complexity is $O(n^2T)$ and the space complexity is $O(nT)$. The space complexity bound is argued as follows. In calculating the probability of the most likely sequence of states that produces $O_0O_1\ldots O_t$ ending in state i, we remember the previous state j by putting an arrow with edge label t from i to j. At the end, can find the most likely sequence by tracing backwards as is standard for dynamic programming algorithms.

Example: For the earlier example what is the most likely sequence of states to produce the output hhht?

$t = 3$	$\max\{\frac{1}{48}\frac{1}{2}\frac{1}{2}, \frac{1}{24}\frac{3}{4}\frac{1}{2}\} = \frac{1}{64} \quad q$ or p	$\max\{\frac{3}{48}\frac{1}{2}\frac{1}{3}, \frac{1}{24}\frac{1}{4}\frac{1}{3}\} = \frac{1}{96} \quad q$
$t = 2$	$\max\{\frac{1}{8}\frac{1}{2}\frac{1}{2}, \frac{1}{6}\frac{3}{4}\frac{1}{2}\} = \frac{3}{48} \quad p$	$\max\{\frac{1}{8}\frac{1}{2}\frac{2}{3}, \frac{1}{6}\frac{1}{4}\frac{2}{3}\} = \frac{1}{24} \quad q$
$t = 1$	$\frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8} \quad q$	$\frac{1}{2}\frac{1}{2}\frac{2}{3} = \frac{1}{6} \quad q$
$t = 0$	$\frac{1}{2} \quad q$	0 p

Note that the two sequences of states, *qqpq* and *qpqq*, are tied for the most likely sequences of states. ∎

Determining the underlying hidden Markov model

Given an n-state HMM, how do we adjust the transition probabilities and output probabilities to maximize the probability of an output sequence $O_1 O_2 \cdots O_T$? The assumption is that T is much larger than n. There is no known computationally efficient method for solving this problem. However, there are iterative techniques that converge to a local optimum.

Let a_{ij} be the transition probability from state i to state j and let $b_j(O_k)$ be the probability of output O_k given that the HMM is in state j. Given estimates for the HMM parameters, a_{ij} and b_j, and the output sequence O, we can improve the estimates by calculating for each unit of time the probability that the HMM goes from state i to state j and outputs the symbol O_k.

a_{ij}	transition probability from state i to state j
$b_j(O_{t+1})$	probability of O_{t+1} given that the HMM is in state j at time $t+1$
$\alpha_t(i)$	probability of seeing $O_0O_1\cdots O_t$ and ending in state i at time t
$\beta_{t+1}(j)$	probability of seeing the tail of the sequence $O_{t+2}O_{t+3}\cdots O_T$ given state j at time $t+1$
$\delta(i,j)$	probability of going from state i to state j at time t given the sequence of outputs O
$s_t(i)$	probability of being in state i at time t given the sequence of outputs O
$p(O)$	probability of output sequence O

Given estimates for the HMM parameters, a_{ij} and b_j, and the output sequence O, the probability $\delta_t(i,j)$ of going from state i to state j at time t is given by the probability of producing the output sequence O and going from state i to state j at time t divided by the probability of producing the output sequence O.

$$\delta_t(i,j) = \frac{a_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{p(O)}$$

The probability $p(O)$ is the sum over all pairs of states i and j of the numerator in the above formula for $\delta_t(i,j)$. That is,

$$p(O) = \sum_i \sum_j \alpha_t(j)a_{ij}b_j(O_{t+1})\beta_{t+1}(j).$$

The probability of being in state i at time t is given by

$$s_t(i) = \sum_{j=1}^{n} \delta_t(i,j).$$

Note that $\delta_t(i,j)$ is the probability of being in state i at time t given $O_0O_1O_2\cdots O_t$ but it is not the probability of being in state i at time t given O since it does not take into account the remainder of the sequence O. Summing $s_t(i)$ over all time periods gives the expected number of times state i is visited and the sum of $\delta_t(i,j)$ over all time periods gives the expected number of times edge i to j is traversed.

Given estimates of the HMM parameters $a_{i,j}$ and $b_j(O_k)$, we can calculate by the above formulas estimates for

1. $\sum_{i=1}^{T-1} s_t(i)$, the expected number of times state i is visited and departed from

2. $\sum_{i=1}^{T-1} \delta_t(i,j)$, the expected number of transitions from state i to state j

Using these estimates we can obtain new estimates of the HMM parameters

$$\overline{a_{ij}} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions out of state } i} = \frac{\sum_{t=1}^{T-1} \delta_t(i,j)}{\sum_{t=1}^{T-1} s_t(i)}$$

$$\overline{b_j}(O_k) = \frac{\text{expected number of times in state } j \text{ observing symbol } O_k}{\text{expected number of times in state } j} = \frac{\sum\limits_{\substack{t=1\\ \text{subject to}\\ O_t=O_k}}^{T-1} s_t(j)}{\sum_{t=1}^{T-1} s_t(j)}$$

By iterating the above formulas we can arrive at a local optimum for the HMM parameters $a_{i,j}$ and $b_j(O_k)$.

9.3 Graphical Models, and Belief Propagation

A graphical model is a compact representation of a function of n variables $x_1, x_2, \ldots, x_n$. It consists of a graph, directed or undirected, whose vertices correspond to variables that take on values from some set. In this chapter, we consider the case where the function is a probability distribution and the set of values the variables take on is finite, although graphical models are often used to represent probability distributions with continuous variables. The edges of the graph represent relationships or constraints between the variables.

The directed model represents a joint probability distribution that factors into a product of conditional probabilities.

$$p\left(x_1, x_2, \ldots, x_n\right) = \prod_{i=1}^{n} p\left(x_i | \text{parents of } x_i\right)$$

It is assumed that the directed graph is acyclic. The directed graphical model is called a *Bayesian* or *belief network* and appears frequently in the artificial intelligence and the statistics literature.

The undirected graph model, called a Markov random field, can also represent a joint probability distribution of the random variables at its vertices. In many applications the Markov random field represents a function of the variables at the vertices which is to be optimized by choosing values for the variables.

A third model called the factor model is akin to the Markov random field, but here the dependency sets have a different structure. In the following sections we describe all these models in more detail.

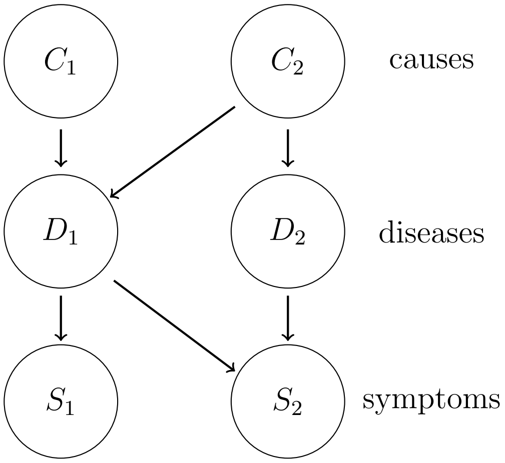


Figure 9.1: A Bayesian network

9.4 Bayesian or Belief Networks

A Bayesian network is a directed acyclic graph where vertices correspond to variables and a directed edge from y to x represents a conditional probability $p(x|y)$. If a vertex x has edges into it from $y_1, y_2, \ldots, y_k$, then the conditional probability is $p\left(x \mid y_1, y_2, \ldots, y_k\right)$. The variable at a vertex with no in edges has an unconditional probability distribution. If the value of a variable at some vertex is known, then the variable is called *evidence*. An important property of a Bayesian network is that the joint probability is given by the product over all nodes of the conditional probability of the node conditioned on all its immediate predecessors.

In the example of Fig. 9.1, a patient is ill and sees a doctor. The doctor ascertains the symptoms of the patient and the possible causes such as whether the patient was in contact with farm animals, whether he had eaten certain foods, or whether the patient has an hereditary predisposition to any diseases. Using the above Bayesian network where the variables are true or false, the doctor may wish to determine one of two things. What is the marginal probability of a given disease or what is the most likely set of diseases. In determining the most likely set of diseases, we are given a T or F assignment to the causes and symptoms and ask what assignment of T or F to the diseases maximizes the joint probability. This latter problem is called the maximum a posteriori probability (MAP).

Given the conditional probabilities and the probabilities $p\left(C_1\right)$ and $p\left(C_2\right)$ in Example 9.1, the joint probability $p\left(C_1, C_2, D_1, \ldots\right)$ can be computed easily for any combination of values of $C_1, C_2, D_1, \ldots$. However, we might wish to find that value of the variables of highest probability (MAP) or we might want one of the marginal probabilities $p\left(D_1\right)$ or $p\left(D_2\right)$. The obvious algorithms for these two problems require evaluating the probability $p\left(C_1, C_2, D_1, \ldots\right)$ over exponentially many input values or summing the probability $p\left(C_1, C_2, D_1, \ldots\right)$ over exponentially many values of the variables other than those for which we want the marginal probability. In certain situations, when the joint probability distribution can be expressed as a product of factors, a belief propagation algorithm can solve the maximum a posteriori problem or compute all marginal probabilities quickly.

9.5 Markov Random Fields

The Markov random field model arose first in statistical mechanics where it was called the Ising model. It is instructive to start with a description of it. The simplest version of the Ising model consists of n particles arranged in a rectangular $\sqrt{n} \times \sqrt{n}$ grid. Each particle can have a spin that is denoted ± 1. The energy of the whole system depends on interactions between pairs of neighboring particles. Let x_i be the spin, ± 1, of the i^{th} particle. Denote by $i \sim j$ the relation that i and j are adjacent in the grid. In the Ising model, the energy of the system is given by

$$f(x_1, x_2, \ldots, x_n) = \exp\left(c \sum_{i \sim j} |x_i - x_j| \right).$$

c is a constant that can be positive or negative. If $c < 0$, then energy is lower if many adjacent pairs have opposite spins and if $c > 0$ the reverse holds. The model was first used to model probabilities of spin configurations. The hypothesis was that for each $\{x_1, x_2, \ldots, x_n\}$ in $\{-1, +1\}^n$, the energy of the configuration with these spins is proportional to $f(x_1, x_2, \ldots, x_n)$.

In most computer science settings, such functions are mainly used as objective functions that are to be optimized subject to some constraints. The problem is to find the minimum energy set of spins under some constraints on the spins. Usually the constraints just specify the spins of some particles. Note that when $c > 0$, this is the problem of minimizing $\sum_l imits_{i \sim j} |x_i - x_j|$ subject to the constraints. The objective function is convex and so this can be done efficiently. If $c < 0$, however, we need to minimize a concave function for which there is no known efficient algorithm. The minimization of a concave function in general is NP-hard.

A second important motivation comes from the area of vision. It has to to do with reconstructing images. Suppose we are given observations of the intensity of light at individual pixels, $x_1, x_2, \ldots, x_n$ and wish to compute the true values, the true intensities, of these variables $y_1, y_2, \ldots, y_n$. There may be two sets of constraints, the first stipulating that the y_i must be close to the corresponding x_i and the second, a term correcting possible observation errors, stipulating that y_i must be close to the values of y_j for $j \sim i$. This can be formulated as

$$\text{Minimize} \sum_i |x_i - y_i| + \sum_{i \sim j} |y_i - y_j|,$$

where the values of x_i are constrained to be the observed values. The objective function is convex and polynomial time minimization algorithms exist. Other objective functions

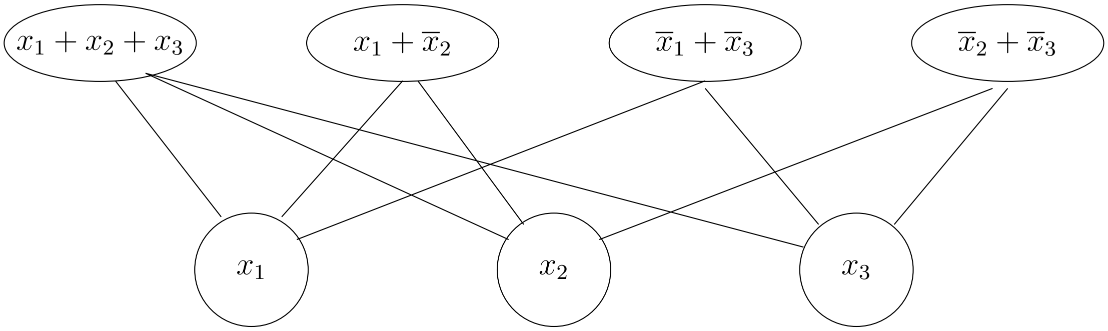

$x_1 + x_2 + x_3$
$x_1 + \overline{x}_2$
$\overline{x}_1 + \overline{x}_3$
$\overline{x}_2 + \overline{x}_3$
x_1
x_2
x_3

Figure 9.2: The factor graph for the function $f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(x_{_1} + x_{_2})(x_{_1} + x_{_3})(x_2 + x_3)$.

using say sum of squares instead of sum of absolute values can be used and thee are polynomial time algorithms as long as the function to be minimized is convex.

More generally, the correction term may depend on all grid points within distance two of each point rather than just immediate neighbors. Even more generally, we may have n variables $y_1, y_2, \ldots y_n$ with the value of some already specified and subsets $S_1, S_2, \ldots S_m$ of these variables constrained in some way. The constraints are accumulated into one objective function which is a product of functions $f_1, f_2, \ldots, f_m$, where function f_i is evaluated on the variables in subset S_i. The problem is to minimize $\prod_{i=1}^{m} f_i(y_j, j \in S_i)$ subject to constrained values. Note that the vision example had a sum instead of a product, but by taking exponentials we can turn the sum into a product as in the Ising model.

In general, the f_i are not convex; indeed they may be discrete. So the minimization cannot be carried out by a known polynomial time algorithm. The most used forms of the Markov random field involve S_i which are cliques of a graph. So we make the following definition.

A *Markov Random Field* consists of an undirected graph and an associated function that factorizes into functions associated with the cliques of the graph. The special case when all the factors correspond to cliques of size one or two is of interest.

9.6 Factor Graphs

Factor graphs arise when we have a function f of a variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ that can be expressed as $f(\mathbf{x}) = \prod\limits_{\alpha} f_\alpha(x_\alpha)$ where each factor depends only on some small number of variables x_α. The difference from Markov random fields is that the variables corresponding to factors do not necessarily form a clique. Associate a bipartite graph where one set of vertices correspond to the factors and the other set to the variables. Place an edge between a variable and a factor if the factor contains that variable. See

Figure 9.2

9.7 Tree Algorithms

Let $f(\mathbf{x})$ be a function that is a product of factors. When the factor graph is a tree there are efficient algorithms for solving certain problems. With slight modifications, the algorithms presented can also solve problems where the function is the sum of terms rather than a product of factors.

The first problem is called *marginalization* and involves evaluating the sum of f over all variables except one. In the case where f is a probability distribution the algorithm computes the marginal probabilities and thus the word marginalization. The second problem involves computing the assignment to the variables that maximizes the function f. When f is a probability distribution, this problem is the maximum a posteriori probability or MAP problem.

If the factor graph is a tree, then there exists an efficient algorithm for solving these problems. Note that there are four problems: the function f is either a product or a sum and we are either marginalizing or finding the maximizing assignment to the variables. All four problems are solved by essentially the same algorithm and we present the algorithm for the marginalization problem when f is a product. Assume we want to "sum out" all the variables except x_1, so we will be left with a function of x_1.

We call the variable node associated with the variable x_i node x_i. First, make the node x_1 the root of the tree. It will be useful to think of the algorithm first as a recursive algorithm and then unravel the recursion. We want to compute the product of all factors occurring in the sub-tree rooted at the root with all variables except the root-variable summed out. Let g_i be the product of all factors occurring in the sub-tree rooted at node x_i with all variables occurring in the subtree except x_i summed out. Since this is a tree, x_1 will not reoccur anywhere except the root. Now, the grandchildren of the root are variable nodes and suppose for recursion, each grandchild x_i of the root, has already computed its g_i. It is easy to see that we can compute g_1 by the following.

Each grandchild x_i of the root passes its g_i to its parent, which is a factor node. Each child of x_1 collects all its children's g_i, multiplies them together with its own factor and sends the product to the root. The root multiplies all the products it gets from its children and sums out all variables except its own variable, namely here x_1.

Unraveling the recursion is also simple, with the convention that a leaf node just receives 1, product of an empty set of factors, from its children. Each node waits until it receives a message from each of its children. After that, if the node is a variable node, it computes the product of all incoming messages, and sums this product function over all assignments to the variables except for the variable of the node. Then, it sends the

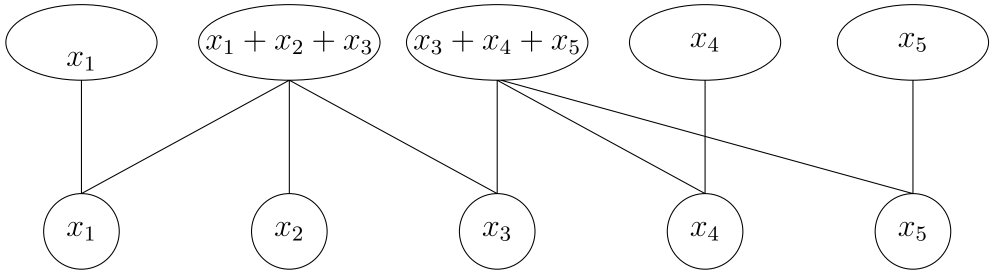


Figure 9.3: The factor graph for the function $f = x_1 (x_1 + x_2 + x_3)(x_3 + x_4 + x_5) x_4 x_5$.

resulting function of one variable out along the edge to its parent. If the node is a factor node, it computes the product of its factor function along with incoming messages from all the children and sends the resulting function out along the edge to its parent.

The reader should prove that the following invariant holds assuming the graph is a tree:

Invariant The message passed by each variable node to its parent is the product of all factors in the subtree under the node with all variables in the subtree except its own summed out.

Consider the following example where

$$f = x_1 (x_1 + x_2 + x_3)(x_3 + x_4 + x_5) x_4 x_5$$

and the variables take on values 0 or 1. Consider marginalizing f by computing

$$f(x_1) = \sum_{x_2 x_3 x_4 x_5} x_1 (x_1 + x_2 + x_3)(x_3 + x_4 + x_5) x_4 x_5,$$

In this case the factor graph is a tree as shown in Figure 9.3. The factor graph as a rooted tree and the messages passed by each node to its parent are shown in Figure 9.4. If instead of computing marginal's, one wanted the variable assignment that maximizes the function f, one would modify the above procedure by replacing the summation by a maximization operation. Obvious modifications handle the situation where $f(\mathbf{x})$ is a sum of products.

$$f(\mathbf{x}) = \sum_{x_1, \ldots, x_n} g(\mathbf{x})$$

9.8 Message Passing in general Graphs

The simple message passing algorithm in the last section gives us the one variable function of x_1 when we sum out all the other variables. For a general graph that is not a tree, we formulate an extension of that algorithm. But unlike the case of trees, there is no proof that the algorithm will converge and even if it does, there is no guarantee

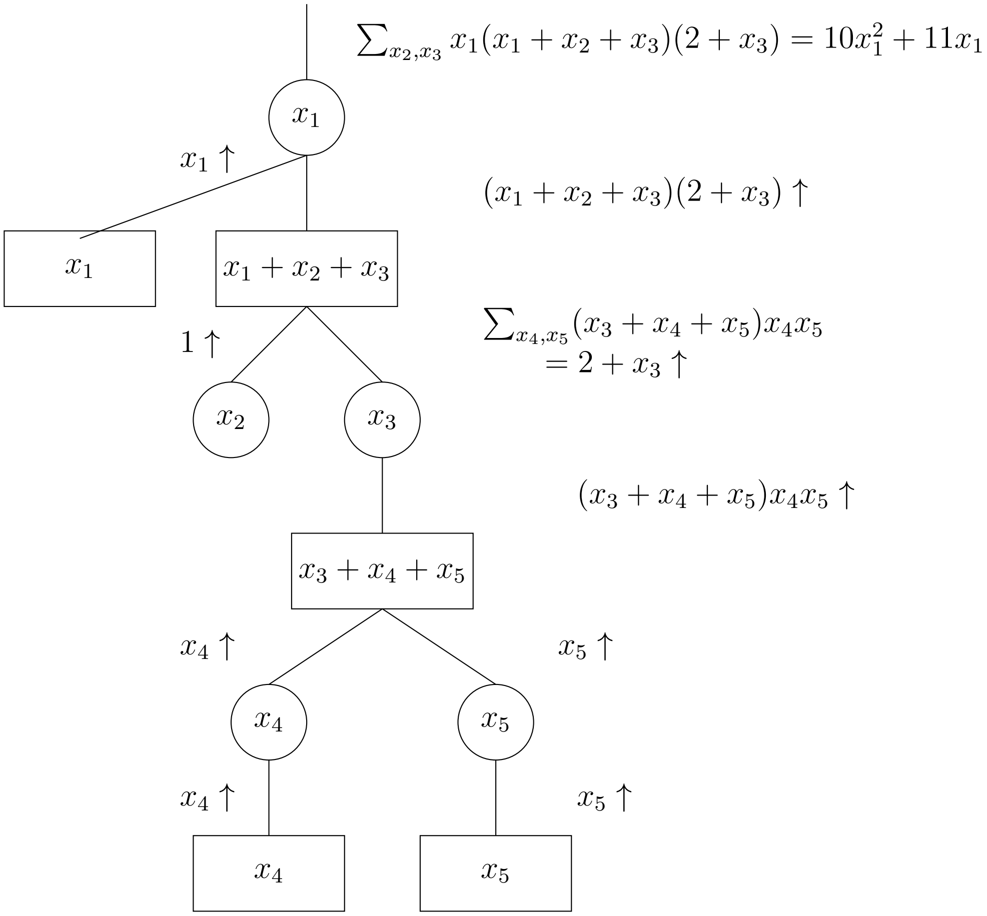


Figure 9.4: Messages.

that the limit is the marginal probability. This has not prevented its usefulness in some applications.

First, lets ask a more general question, just for trees. Suppose we want to compute for each i the one variable function of x_i when we sum out all variables $x_j, j \neq i$. Do we have to repeat what we did for x_1 once for each x_i? Luckily, the answer is no. It will suffice to do a second pass from the root to the leaves of essentially the same message passing algorithm to get all the answers. Recall that in the first pass, each edge of the tree has sent a message "up", from the child to the parent. In the second pass, each edge will send a message from the parent to the child. We start with the root and work downwards for this pass. Each node waits until its parent has sent it a message before sending messages to each of its children. The rules for messages are:

Rule 1 The message from a factor node v to a child x_i, which is the variable node x_i, is the product of all messages received by v in both passes from all nodes other than x_i

times the factor at v itself.

Rule 2 The message from a variable node x_i to a child, a factor node, v is the product of all messages received by x_i in both passes from all nodes except v, with all variables except x_i summed out. The message is a function of x_i alone.

At termination, one can show when the graph is a tree that if we take the product of all messages received in both passes by a variable node x_i and sum out all variables except x_i in this product, what we get is precisely the entire function marginalized to x_i. We do not give the proof here. But the idea is simple. We know from the first pass that the product of the messages coming to a variable node x_i from its children is the product of all factors in the sub-tree rooted at x_i. In the second pass, we claim that the message from the parent v to x_i is the product of all factors which are not in the sub-tree rooted at x_i which one can show either directly or by induction working from the root downwards.

We can apply the same rules 1 and 2 to any general graph. We do not have child and parent relationships and it is not possible to have the two synchronous passes as before. The messages keep flowing and one hopes that after some time, the messages will stabilize, but nothing like that is proven. We state the algorithm for general graphs now:

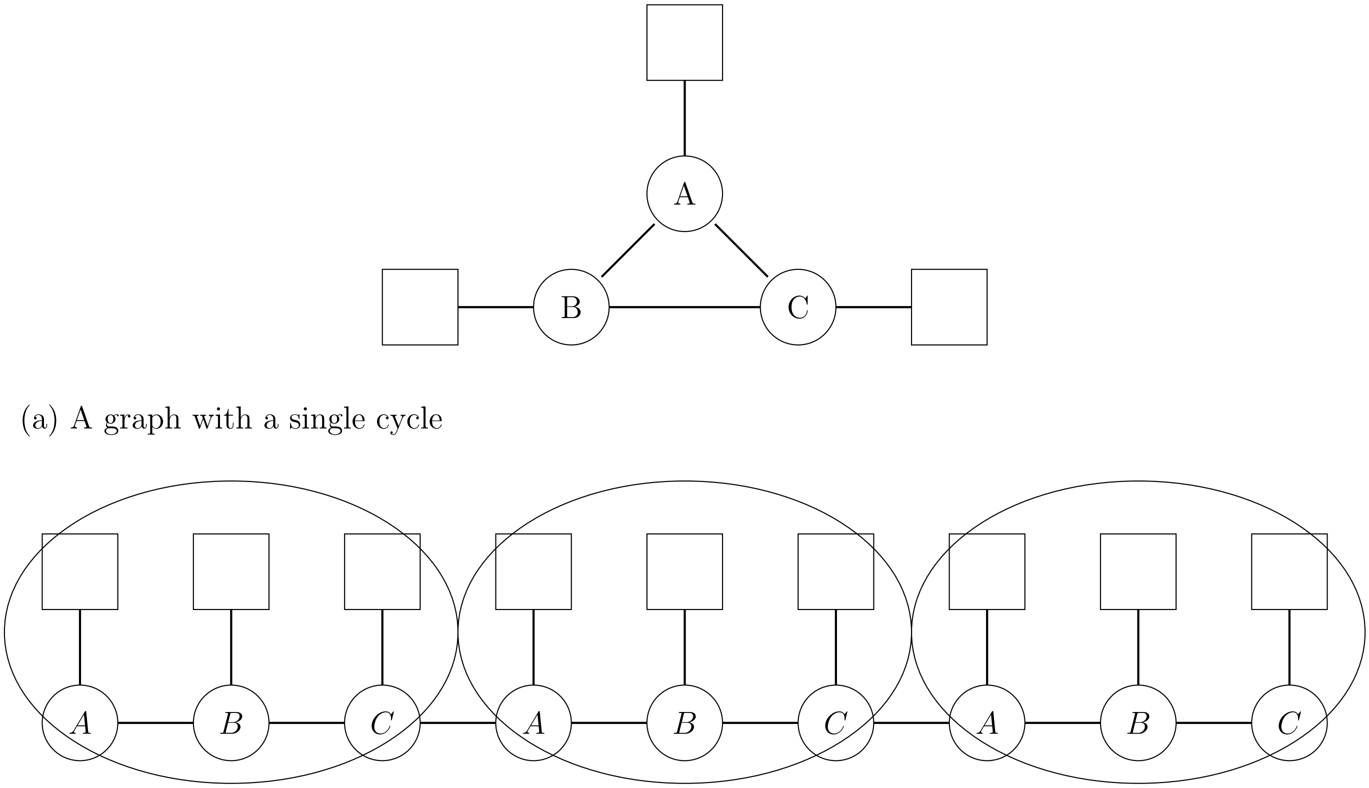
Rule 1 At each time, each factor node v sends a message to each adjacent node x_i. The message is the product of all messages received by v at the previous step except for the one from x_i multiplied by the factor at v itself.

Rule 2 At each time, each variable node x_i sends a message to each adjacent node v. The message is the product of all messages received by x_i at the previous step except the one from v, with all variables except x_i summed out.

9.9 Graphs with a Single Cycle

The message passing algorithm gives the correct answers on trees and on certain other graphs. One such situation is graphs with a single cycle which we treat here. We switch from the marginalization problem to the MAP problem as the proof of correctness is simpler for the MAP problem. Consider the network in the Figure 9.5a below with a single cycle. The message passing scheme will multiply count some evidence. The local evidence at A will get passed around the loop and will come back to A. Thus, A will count the local evidence multiple times. If all evidence is multiply counted in equal amounts, then there is a possibility that all though the numerical values of the marginal probabilities (beliefs) are wrong, the algorithm still converges to the correct maximum a posteriori assignment.

Consider the unwrapped version of the graph in Figure 9.5b. The messages that the loopy version will eventually converge to, assuming convergence, are the same messages that occur in the unwrapped version provided that the nodes are sufficiently far in from



(a) A graph with a single cycle

(b) Segment of unrolled graph

Figure 9.5: Unwrapping a graph with a single cycle

the ends. The beliefs in the unwrapped version are correct for the unwrapped graph since it is a tree. The only question is, how similar are they to the true beliefs in the original network.

Write $p\left(A,B,C\right)=e^{\log p(A,B,C)}=e^{J(A,B,C)}$ where $J\left(A,B,C\right)=\log p\left(A,B,C\right)$. Then the probability for the unwrapped network is of the form $e^{kJ(A,B,C)+J'}$ where the J' is associated with vertices at the ends of the network where the beliefs have not yet stabilized and the $kJ\left(A,B,C\right)$ comes from k inner copies of the cycle where the beliefs have stabilized. Note that the last copy of J in the unwrapped network shares an edge with J' and that edge has an associated Ψ. Thus, changing a variable in J has an impact on the value of J' through that Ψ. Since the algorithm maximizes $J_k=kJ\left(A,B,C\right)+J'$ in the unwrapped network for all k, it must maximize $J\left(A,B,C\right)$. To see this, set the variables A, B, C, so that J_k is maximized. If $J\left(A,B,C\right)$ is not maximized, then change A, B, and C to maximize $J\left(A,B,C\right)$. This increases J_k by some quantity that is proportional to k. However, two of the variables that appear in copies of $J\left(A,B,C\right)$ also appear in J' and thus J' might decrease in value. As long as J' decreases by some finite amount, we can increase J_k by increasing k sufficiently. As long as all Ψ's are nonzero, J' which is

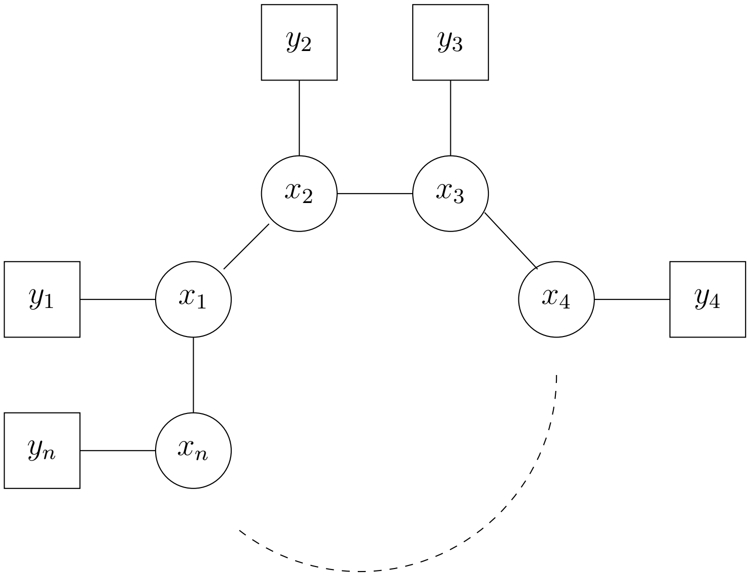


Figure 9.6: A Markov random field with a single loop.

proportional to $\log \Psi$, can change by at most some finite amount. Hence, for a network with a single loop, assuming that the message passing algorithm converges, it converges to the maximum a posteriori assignment.

9.10 Belief Update in Networks with a Single Loop

In the previous section, we showed that when the message passing algorithm converges, it correctly solves the MAP problem for graphs with a single loop. The message passing algorithm can also be used to obtain the correct answer for the marginalization problem. Consider a network consisting of a single loop with variables $x_1, x_2, \ldots, x_n$ and evidence $y_1, y_2, \ldots, y_n$ as shown in Figure 9.6. The x_i and y_i can be represented by vectors having a component for each value x_i can take on. To simplify the discussion assume the x_i take on values $1, 2, \ldots, m$.

Let m_i be the message sent from vertex i to vertex $i+1 \bmod n$. At vertex $i+1$ each component of the message m_i is multiplied by the evidence y_{i+1} and the constraint function Ψ. This is done by forming a diagonal matrix D_{i+1} where the diagonal elements are the evidence and then forming a matrix M_i whose rs^{th} element is $\Psi\left(x_{i+1} = r,\ x_i = s\right)$. The message m_{i+1} is $M_i D_{i+1} m_i$. Multiplication by the diagonal matrix D_{i+1} multiplies the components of the message m_i by the associated evidence. Multiplication by the matrix M_i multiplies each component of the vector by the appropriate value of Ψ and sums over the values producing the vector which is the message m_{i+1}. Once the message has travelled around the loop, the new message m_1' is given by

$$m_1' = M_n D_1 M_{n-1} D_n \cdots M_2 D_3 M_1 D_2 m_1$$

Let $M = M_n D_1 M_{n-1} D_n \cdots M_2 D_3 M_1 D_2 m_1$. Assuming that M's principle eigenvalue is unique, the message passing will converge to the principle vector of M. The rate of convergences depends on the ratio of the first and second eigenvalues.

An argument analogous to the above concerning the messages gong clockwise around the loop applies to messages moving counter clockwise around the loop. To obtain the estimate of the marginal probability $p\left(x_1\right)$, one multiples component wise the two messages arriving at x_1 along with the evidence y_1. This estimate does not give the true marginal probability but the true marginal probability can be computed from the estimate and the rate of convergences by linear algebra.

9.11 Maximum Weight Matching

We have seen that the belief propagation algorithm converges to the correct solution in trees and graphs with a single cycle. It also correctly converges for a number of problems. Here we give one example, the maximum weight matching problem where there is a unique solution.

We apply the belief propagation algorithm to find the maximal weight matching (MWM) in a complete bipartite graph. If the MWM in the bipartite graph is unique, then the belief propagation algorithm will converge to it.

Let $G = (V_1, V_2, E)$ be a complete bipartite graph where $V_1 = \{a_1, \ldots, a_n\}$, $V_2 = \{b_1, \ldots, b_n\}$, and $(a_i, b_j) \in E, \quad 1 \leq i, j \leq n$. Let $\pi = \{\pi(1), \ldots, \pi(n)\}$ be a permutation of $\{1, \ldots, n\}$. The collection of edges $\left\{\left(a_1, b_{\pi(1)}\right), \ldots, \left(a_n, b_{\pi(n)}\right)\right\}$is called a *matching* which is denoted by π. Let w_{ij} be the weight associated with the edge (a_i, b_j). The weight of the matching π is $w_\pi = \sum\limits_{i=1}^{n} w_{i\pi(i)}$. The maximum weight matching π^* is $\pi^* = \arg\max\limits_{\pi}\ w_\pi$

The first step is to create a factor graph corresponding to the MWM problem. Each edge of the bipartite graph is represented by a variable c_{ij} which takes on the values zero or one. The value one means that the edge is present in the matching, the value zero means that the edge is not present in the matching. A set of constraints is used to force the set of edges to be a matching. The constraints are of the form $\sum\limits_{j} c_{ij} = 1$ and $\sum\limits_{i} c_{ij} = 1$. Any assignment of 0,1 to the variables c_{ij} that satisfies all of the constraints defines a matching. In addition, we have constraints for the weights of the edges.

We now construct a factor graph, a portion of which is shown in Fig. 9.10. Associated with the factor graph is a function $f\left(c_{11}, c_{12}, \ldots\right)$ consisting of a set of terms for each c_{ij} enforcing the constraints and summing the weights of the edges of the matching. The terms for c_{12} are

$$-\lambda\left|\left(\sum_i c_{i2}\right)-1\right|-\lambda\left|\left(\sum_j c_{1j}\right)-1\right|+w_{12}c_{12}$$

where λ is a large positive number used to enforce the constraints when we maximize the function. Finding the values of $c_{11}, c_{12}, \ldots$ that maximize f finds the maximum weighted matching for the bipartite graph.

If the factor graph was a tree, then the message from a variable node x to its parent is a message $g(x)$ that gives the maximum value for the sub tree for each value of x. To compute $g(x)$, one sums all messages into the node x. For a constraint node, one sums all messages from sub trees and maximizes the sum over all variables except the variable of the parent node subject to the constraint. The message from a variable x consists of two pieces of information, the value $p\,(x=0)$ and the value $p\,(x=1)$. This information can be encoded into a linear function of x.

$$[p\,(x=1)-p\,(x=0)]\,x+p\,(x=0)$$

Thus, the messages are of the form $ax+b$. To determine the MAP value of x once the algorithm converges, sum all messages into x and take the maximum over x=1 and x=0 to determine the value for x. Since the arg maximum of a linear form ax+b depends only on whether a is positive or negative and since maximizing the output of a constraint depends only on the coefficient of the variable, we can send messages consisting of just the variable coefficient.

To calculate the message to c_{12} from the constraint that node b_2 has exactly one neighbor, add all the messages that flow into the constraint node from the c_{i2}, $i \neq 1$ nodes and maximize subject to the constraint that exactly one variable has value one. If $c_{12}=0$, then one of c_{i2}, $i \neq 1$, will have value one and the message is $\max\limits_{i\neq 1}\alpha\,(i,2)$. If $c_{12}=1$, then the message is zero. Thus, we get

$$-\max_{i\neq 1}\alpha\,(i,2)\,x+\max_{i\neq 1}\alpha\,(i,2)$$

and send the coefficient $-\max\limits_{i\neq 1}\alpha\,(i,2)$. This means that the message from c_{12} to the other constraint node is $\beta(1,2)=w_{12}-\max\limits_{i\neq 1}\alpha\,(i,2)$.

The alpha message is calculated in a similar fashion. If $c_{12}=0$, then one of c_{1j} will have value one and the message is $\max\limits_{j\neq 1}\beta\,(1,j)$. If $c_{12}=1$, then the message is zero. Thus, the coefficient $-\max\limits_{j\neq 1}\alpha\,(1,j)$ is sent. This means that $\alpha(1,2)=w_{12}-\max\limits_{j\neq 1}\alpha\,(1,j)$.

To prove convergence, we enroll the constraint graph to form a tree with a constraint node as the root. In the enrolled graph a variable node such as c_{12} will appear a number of times which depends on how deep a tree is built. Each occurrence of a variable such as c_{12} is deemed to be a distinct variable.

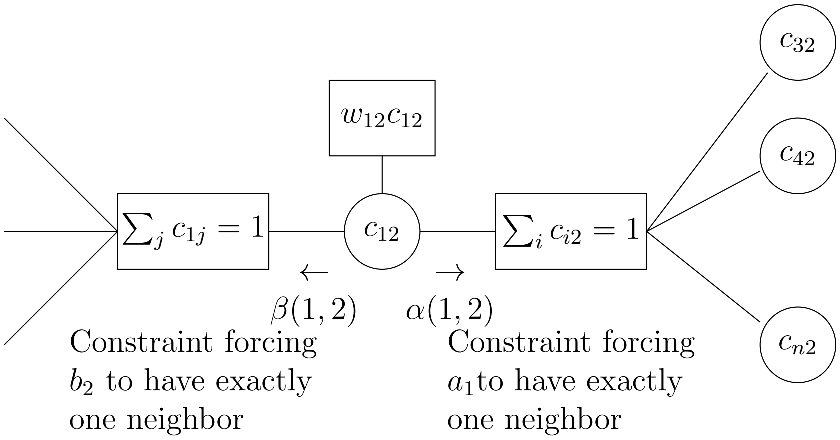


Figure 9.7: Portion of factor graph for the maximum weight matching problem.

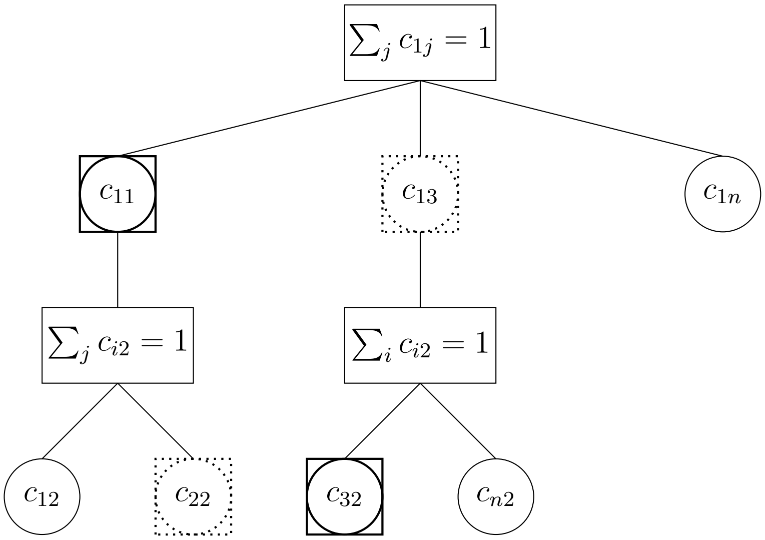


Figure 9.8: Tree for MWM problem.

Lemma 9.3 *If the tree obtained by unrolling the graph is of depth k, then the messages to the root are the same as the messages in the constraint graph after k-iterations.*

Proof: Straight forward. ∎

Define a matching in the tree to be a set of vertices so that there is exactly one variable node of the match adjacent to each constraint. Let Λ denote the vertices of the matching. Heavy circles represent the nodes of the above tree that are in the matching Λ.

Let Π be the vertices corresponding to maximum weight matching edges in the bipartite graph. Recall that vertices in the above tree correspond to edges in the bipartite graph. The vertices of Π are denoted by dotted circles in the above tree.

Consider a set of trees where each tree has a root that corresponds to one of the constraints. If the constraint at each root is satisfied by the edge of the MWM, then we have found the MWM. Suppose that the matching at the root in one of the trees disagrees with the MWM. Then there is an alternating path of vertices of length $2k$ consisting of vertices corresponding to edges in Π and edges in Λ. Map this path onto the bipartite graph. In the bipartite graph the path will consist of a number of cycles plus a simple path. If k is large enough there will be a large number of cycles since no cycle can be of length more than $2n$. Let m be the number of cycles. Then $m \geq \frac{2k}{2n} = \frac{k}{n}$.

Let π^* be the MWM in the bipartite graph. Take one of the cycles and use it as an alternating path to convert the MWM to another matching. Assuming that the MWM is unique and that the next closest matching is ε less, $W_{\pi*} - W_{\pi} > \varepsilon$ where π is the new matching.

Consider the tree matching. Modify the tree matching by using the alternating path of all cycles and the left over simple path. The simple path is converted to a cycle by adding two edges. The cost of the two edges is at most 2w* where w* is the weight of the maximum weight edge. Each time we modify Λ by an alternating cycle, we increase the cost of the matching by at least ε. When we modify Λ by the left over simple path, we increase the cost of the tree matching by $\varepsilon - 2w*$ since the two edges that were used to create a cycle in the bipartite graph are not used. Thus

$$\text{weight of } \Lambda \text{ - weight of } \Lambda' \geq \tfrac{k}{n}\varepsilon - 2w*$$

which must be negative since Λ' is optimal for the tree. However, if k is large enough this becomes positive, an impossibility since Λ' is the best possible. Since we have a tree, there can be no cycles, as messages are passed up the tree, each sub tree is optimal and hence the total tree is optimal. Thus the message passing algorithm must find the maximum weight matching in the weighted complete bipartite graph assuming that the maximum weight matching is unique. Note that applying one of the cycles that makes up the alternating path decreased the bipartite graph match but increases the value of

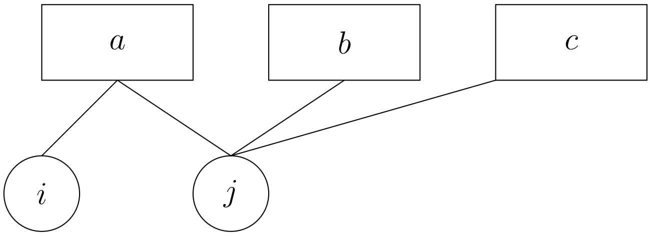


Figure 9.9: warning propagation

the tree. However, it does not give a higher tree matching, which is not possible since we already have the maximum tree matching. The reason for this is that the application of a single cycle does not result in a valid tree matching. One must apply the entire alternating path to go from one matching to another.

9.12 Warning Propagation

Significant progress has been made using methods similar to belief propagation in finding satisfying assignments for 3-CNF formulas. Thus, we include a section on a version of belief propagation, called warning propagation, that is quite effective in finding assignments. Consider a factor graph for a SAT problem. Index the variables by i, j, and k and the factors by a, b, and c. Factor a sends a message m_{ai} to each variable i that appears in the factor a called a warning. The warning is 0 or 1 depending on whether or not factor a believes that the value assigned to i is required for a to be satisfied. A factor a determines the warning to send to variable i by examining all warnings received by other variables in factor a from factors containing them.

For each variable j, sum the warnings from factors containing j that warn j to take value T and subtract the warnings that warn j to take value F. If the difference says that j should take value T or F and this value for variable j does not satisfy a, and this is true for all j, then a sends a warning to i that the value of variable i is critical for factor a.

Start the warning propagation algorithm by assigning 1 to a warning with probability 1/2. Iteratively update the warnings. If the warning propagation algorithm converges, then compute for each variable i the local field h_i and the contradiction number c_i. The local field h_i is the number of clauses containing the variable i that sent messages that i should take value T minus the number that sent messages that i should take value F. The contradiction number c_i is 1 if variable i gets conflicting warnings and 0 otherwise. If the factor graph is a tree, the warning propagation algorithm converges. If one of the warning messages is one, the problem is unsatisfiable; otherwise it is satisfiable.

9.13 Correlation Between Variables

In many situations one is interested in how the correlation between variables drops off with some measure of distance. Consider a factor graph for a 3-CNF formula. Measure the distance between two variables by the shortest path in the factor graph. One might ask if one variable is assigned the value true, what is the percentage of satisfying assignments in which the second variable also is true. If the percentage is the same as when the first variable is assigned false, then we say that the two variables are uncorrelated. How difficult it is to solve a problem is likely to be related to how fast the correlation decreases with distance.

Another illustration of this concept is in counting the number of perfect matchings in a graph. One might ask what is the percentage of matching in which some edge is present and ask how correlated this percentage is with the presences or absence of edges at some distance d. One is interested in whether the correlation drops off with distance. To explore this concept we consider the Ising model studied in physics.

The Ising or ferromagnetic model is a pairwise random Markov field. The underlying graph, usually a lattice, assigns a value of ±1, called spin, to the variable at each vertex. The probability (Gibbs measure) of a given configuration of spins is proportional to $exp(\beta \sum\limits_{(i,j)\in E} x_i x_j) = \prod\limits_{(i,j)\in E} e^{\beta x_i x_j}$ where $x_i = \pm 1$ is the value associated with vertex i. Thus

$$p\left(x_1, x_2, \ldots, x_n\right) = \tfrac{1}{Z} \prod_{(i,j)\in E} exp(\beta x_i x_j) = \tfrac{1}{Z} e^{\beta \sum\limits_{(i,j)\in E} x_i x_j}$$

where Z is a normalization constant.

The value of the summation is simply the difference in the number of edges whose vertices have the same spin minus the number of edges whose vertices have opposite spin. The constant β is viewed as inverse temperature. High temperature corresponds to a low value of β and low temperature corresponds to a high value of β. At high temperature, low β, the spins of adjacent vertices are uncorrelated whereas at low temperature adjacent vertices have identical spins. The reason for this is that the probability of a configuration is proportional to $e^{\beta \sum\limits_{i \sim j} x_i x_j}$. As β is increased, $e^{\beta \sum\limits_{i \sim j} x_i x_j}$ for configurations with a large number of edges whose vertices have identical spins increases more than for configurations whose edges have vertices with non identical spins. When the normalization constant $\frac{1}{Z}$ is adjusted for the new value of β, the highest probability configurations are those where adjacent vertices have identical spins.

Given the above probability distribution, what is the correlation between two variables x_i and x_j. To answer this question, consider the probability that x_i equals plus one as a function of the probability that x_j equals plus one. If the probability that x_i equals plus one is $\frac{1}{2}$ independent of the value of the probability that x_j equals plus one, we say the values are uncorrelated.

Consider the special case where the graph G is a tree. In this case a phase transition occurs at $\beta_0 = \frac{1}{2}\ln\frac{d+1}{d-1}$ where d is the degree of the tree. For a sufficiently tall tree and for $\beta > \beta_0$, the probability that the root has value +1 is bounded away from $^1\!/_2$ and depends on whether the majority of leaves have value +1 or -1. For $\beta < \beta_0$ the probability that the root has value +1 is $^1\!/_2$ independent of the values at the leaves of the tree.

Consider a height one tree of degree d. If i of the leaves have spin +1 and $d-i$ have spin -1, then the probability of the root having spin +1 is proportional to

$$e^{i\beta-(d-i)\beta} = e^{(2i-d)\beta}.$$

If the probability of a leaf being +1 is p, then the probability of i leaves being +1 and $d-i$ being -1 is

$$\binom{d}{i} p^i \left(1-p\right)^{d-i}$$

Thus, the probability of the root being +1 is proportional to

$$A = \sum_{i=1}^{d} \binom{d}{i} p^i (1-p)^{d-i} e^{(2i-d)\beta} = e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} \left(pe^{2\beta}\right)^i (1-p)^{d-i} = e^{-d\beta}\left[pe^{2\beta}+1-p\right]^d$$

and the probability of the root being –1 is proportional to

$$\begin{aligned} B &= \sum_{i=1}^{d} \binom{d}{i} p^i (1-p)^{d-i} e^{-(2i-d)\beta} \\ &= e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} p^i \left[(1-p)e^{-2(i-d)\beta}\right] \\ &= e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} p^i \left[(1-p)e^{2\beta}\right]^{d-i} \\ &= e^{-d\beta}\left[p+(1-p)e^{2\beta}\right]^d. \end{aligned}$$

The probability of the root being +1 is

$$q = \tfrac{A}{A+B} = \frac{\left[pe^{2\beta}+1-p\right]^d}{\left[pe^{2\beta}+1-p\right]^d + \left[p+(1-p)e^{2\beta}\right]^d} = \tfrac{C}{D}$$

where

$$C = \left[pe^{2\beta}+1-p\right]^d$$

and

$$D = \left[pe^{2\beta}+1-p\right]^d + \left[p+\left(1-p\right)e^{2\beta}\right]^d.$$

At high temperature, low β, the probability q of the root of the height one tree being $+1$ in the limit as β goes to zero is

$$q = \frac{p+1-p}{[p+1-p]+[p+1-p]} = \frac{1}{2}$$

independent of p. At low temperature, high β,

$$q \approx \frac{p^d e^{2\beta d}}{p^d e^{2\beta d} + (1-p)^d e^{2\beta d}} = \frac{p^d}{p^d + (1-p)^d} = \left\{ \begin{array}{ll} 0 & p=0 \\ 1 & p=1 \end{array} \right. .$$

q goes from a low probability of $+1$ for p below $1/2$ to high probability of $+1$ for p above $1/2$.

Now consider a very tall tree. If the p is the probability that a root has value $+1$, we can iterate the formula for the height one tree and observe that at low temperature the probability of the root being one converges to some value. At high temperature, the probability of the root being one is $1/2$ independent of p. See Figure 9.10. At the phase transition, the slope of q at p=1/2 is one.

Now the slope of the probability of the root being 1 with respect to the probability of a leaf being 1 in this height one tree is

$$\frac{\partial q}{\partial p} = \frac{D\frac{\partial C}{\partial p} - C\frac{\partial D}{\partial p}}{D^2}$$

Since the slope of the function $q(p)$ at p=1/2 when the phase transition occurs is one, we can solve $\frac{\partial q}{\partial p} = 1$ for the value of β where the phase transition occurs. First, we show that $\left.\frac{\partial D}{\partial p}\right|_{p=\frac{1}{2}} = 0$.

$$\begin{aligned} D &= \left[pe^{2\beta} + 1 - p\right]^d + \left[p + (1-p)\, e^{2\beta}\right]^d \\ \tfrac{\partial D}{\partial p} &= d\left[pe^{2\beta} + 1 - p\right]^{d-1}\left(e^{2\beta} - 1\right) + d\left[p + (1-p)\, e^{2\beta}\right]^{d-1}\left(1 - e^{2\beta}\right) \\ \left.\tfrac{\partial D}{\partial p}\right|_{p=\frac{1}{2}} &= \tfrac{d}{2^{d-1}}\left[e^{2\beta} + 1\right]^{d-1}\left(e^{2\beta} - 1\right) + \tfrac{d}{2^{d-1}}\left[1 + e^{2\beta}\right]^{d-1}\left(1 - e^{2\beta}\right) = 0 \end{aligned}$$

Then

$$\begin{aligned} \left.\frac{\partial q}{\partial p}\right|_{p=\frac{1}{2}} &= \left.\frac{D\frac{\partial C}{\partial p} - C\frac{\partial D}{\partial p}}{D^2}\right|_{p=\frac{1}{2}} = \left.\frac{\frac{\partial C}{\partial p}}{D}\right|_{p=\frac{1}{2}} = \left.\frac{d\left[pe^{2\beta} + 1 - p\right]^{d-1}\left(e^{2\beta} - 1\right)}{\left[pe^{2\beta} + 1 - p\right]^d + \left[p + (1-p)\, e^{2\beta}\right]^d}\right|_{p=\frac{1}{2}} \\ &= \frac{d\left[\frac{1}{2}e^{2\beta} + \frac{1}{2}\right]^{d-1}\left(e^{2\beta} - 1\right)}{\left[\frac{1}{2}e^{2\beta} + \frac{1}{2}\right]^d + \left[\frac{1}{2} + \frac{1}{2}e^{2\beta}\right]^d} = \frac{d\left(e^{2\beta} - 1\right)}{1 + e^{2\beta}} \end{aligned}$$

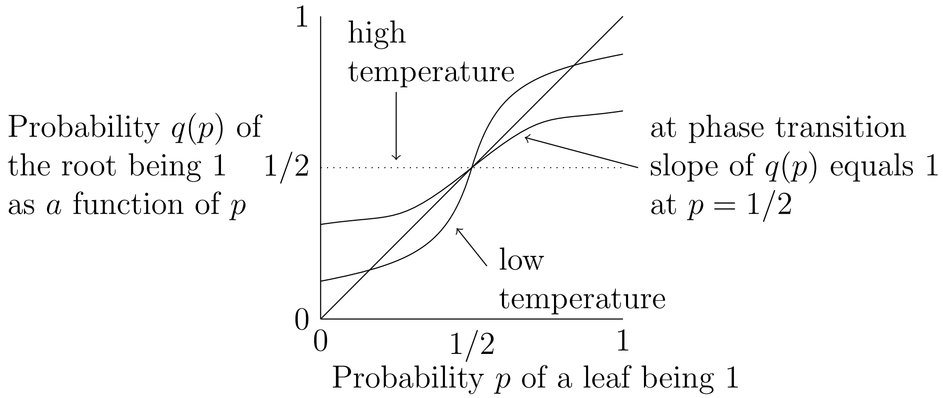


Figure 9.10: Shape of q as a function of p for the height one tree and three values of β corresponding to low temperature, the phase transition temperature, and high temperature.
.

Setting

$$\frac{d\left(e^{2\beta}-1\right)}{1+e^{2\beta}}=1$$

And solving for β yields

$$d\left(e^{2\beta}-1\right)=1+e^{2\beta}$$
$$e^{2\beta}=\tfrac{d+1}{d-1}$$
$$\beta=\tfrac{1}{2}\ln\tfrac{d+1}{d-1}$$

To complete the argument, we need to show that q is a monotonic function of p. To see this, write $q = \frac{1}{1+\frac{B}{A}}$. A is a monotonically increasing function of p and B is monotonically decreasing. From this it follows that q is monotonically increasing.

In the iteration going from p to q, we do not get the true marginal probabilities at each level since we ignored the effect of the portion of the tree above. However, when we get to the root, we do get the true marginal for the root. To get the true marginal's for the interior nodes we need to send messages down from the root.

Note: The joint probability distribution for the tree is of the form $e^{\beta \sum\limits_{(ij)\in E)} x_i x_j} = \prod\limits_{(i,j)\in E} e^{\beta x_i x_j}$. Suppose x_1 has value 1 with probability p. Then define a function φ, called evidence, such that

$$\varphi\left(x_1\right)=\begin{cases} p & \text{for } \mathrm{x}_1 = 1 \\ 1-p & \text{for } \mathrm{x}_1 = -1 \end{cases}$$
$$=\left(p-\tfrac{1}{2}\right)x_1+\tfrac{1}{2}$$

and multiply the joint probability function by φ. Note, however, that the marginal probability of x_1 is not p. In fact, it may be further from p after multiplying the conditional probability function by the function φ.

9.14 Exercises

Exercise 9.1 *Find a nonnegative factorization of the matrix*

$$A = \begin{pmatrix} 4 & 6 & 5 \\ 1 & 2 & 3 \\ 7 & 10 & 7 \\ 6 & 8 & 4 \\ 6 & 10 & 11 \end{pmatrix}$$

Indicate the steps in your method and show the intermediate results.

Exercise 9.2 *Find a nonnegative factorization of each of the following matrices.*

$$(1) \quad \begin{pmatrix} 10 & 9 & 15 & 14 & 13 \\ 2 & 1 & 3 & 3 & 1 \\ 8 & 7 & 13 & 11 & 11 \\ 7 & 5 & 11 & 10 & 7 \\ 5 & 5 & 11 & 6 & 11 \\ 1 & 1 & 3 & 1 & 3 \\ 2 & 2 & 2 & & 2 \end{pmatrix} \qquad (2) \quad \begin{pmatrix} 5 & 5 & 10 & 14 & 17 \\ 2 & 2 & 4 & 4 & 6 \\ 1 & 1 & 2 & 4 & 4 \\ 1 & 1 & 2 & 2 & 3 \\ 3 & 3 & 6 & 8 & 10 \\ 5 & 5 & 10 & 16 & 18 \\ 2 & 2 & 4 & 6 & 7 \end{pmatrix}$$

$$(3) \quad \begin{pmatrix} 4 & 4 & 3 & 3 & 1 & 3 & 4 & 3 \\ 13 & 16 & 13 & 10 & 5 & 13 & 14 & 10 \\ 15 & 24 & 21 & 12 & 9 & 21 & 18 & 12 \\ 7 & 16 & 15 & 6 & 7 & 15 & 10 & 6 \\ 1 & 4 & 4 & 1 & 2 & 4 & 2 & 1 \\ 5 & 8 & 7 & 4 & 3 & 7 & 6 & 4 \\ 3 & 12 & 12 & 3 & 6 & 12 & 6 & 3 \end{pmatrix} \qquad (4) \quad \begin{pmatrix} 1 & 1 & 3 & 4 & 4 & 4 & 1 \\ 9 & 9 & 9 & 12 & 9 & 9 & 3 \\ 6 & 6 & 12 & 16 & 15 & 15 & 4 \\ 3 & 3 & 3 & 4 & 3 & 3 & 1 \end{pmatrix}$$

Exercise 9.3 *Consider the matrix A that is the product of nonnegative matrices B and C.*

$$\begin{pmatrix} 12 & 22 & 41 & 35 \\ 19 & 20 & 13 & 48 \\ 11 & 14 & 16 & 29 \\ 14 & 16 & 14 & 36 \end{pmatrix} = \begin{pmatrix} 10 & 1 \\ 1 & 9 \\ 3 & 4 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 1 & 2 & 4 & 3 \\ 2 & 2 & 1 & 5 \end{pmatrix}$$

Which rows of A are approximate positive linear combinations of other rows of A?
Find an approxiamte nonnegative factorization of A

Exercise 9.4 *What is the probability of heads occurring after a sufficiently long sequence of transitions in Viterbi algorithm example of the most likely sequence of states?*

Exercise 9.5 *Find optimum parameters for a three state HMM and given output sequence. Note the HMM must a strong signature in the output sequence or we probably will not be able to find it. The following example may not be good for that reason.*

	1	*2*	*3*
1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$
2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$
3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

	A	*B*
1	$\frac{3}{4}$	$\frac{1}{4}$
2	$\frac{1}{4}$	$\frac{3}{4}$
3	$\frac{1}{3}$	$\frac{2}{3}$

Exercise 9.6 *In the Ising model for a tree of degree one, a chain of vertices, is there a phase transition where the correlation between the value at the root and the value at the leaves becomes independent? Work out mathematical what happens.*

Exercise 9.7 *For a Boolean function in CNF the marginal probability gives the number of satisfiable assignments with x_1.*

How does one obtain the number of satisfying assignments for a 2-CNF formula? Not completely related to first sentence.

10 Other Topics

10.1 Rankings

Ranking is important. We rank movies, restaurants, students, web pages, and many other items. Ranking has become a multi-billion dollar industry as organizations try to raise the position of their web pages in the display of web pages returned by search engines to relevant queries. Developing a method of ranking that is not manipulative is an important task.

A ranking is a complete ordering in the sense that for every pair of items a and b, either a is preferred to b or b is preferred to a. Furthermore, a ranking is transitive in that $a > b$ and $b > c$ implies $a > c$.

One problem of interest in ranking is that of combining many individual rankings into one global ranking. However, merging ranked lists is nontrivial as the following example illustrates.

Example: Suppose there are three individuals who rank items a, b, and c as illustrated in the following table.

individual	first item	second item	third item
1	a	b	c
2	b	c	a
3	c	a	b

Suppose our algorithm tried to rank the items by first comparing a to b and then comparing b to c. In comparing a to b, two of the three individuals prefer a to b and thus we conclude a is preferable to b. In comparing b to c, again two of the three individuals prefer b to c and we conclude that b is preferable to c. Now by transitivity one would expect that the individuals would prefer a to c, but such is not the case, only one of the individuals prefers a to c and thus c is preferable to a. We come to the illogical conclusion that a is preferable to b, b is preferable to c, and c is preferable to a. ■

Suppose there are a number of individuals or voters and a set of candidates to be ranked. Each voter produces a ranked list of the candidates. From the set of ranked lists can one construct a single ranking of the candidates? Assume the method of producing a global ranking is required to satisfy the following three axioms.

Nondictatorship – The algorithm cannot always simply select one individual's ranking.

Unanimity – If every individual prefers a to b, then the global ranking must prefer a to b.

Independent of irrelevant alternatives – If individuals modify their rankings but keep the order of a and b unchanged, then the global order of a and b should not change.

Arrow showed that no ranking algorithm exists satisfying the above axioms.

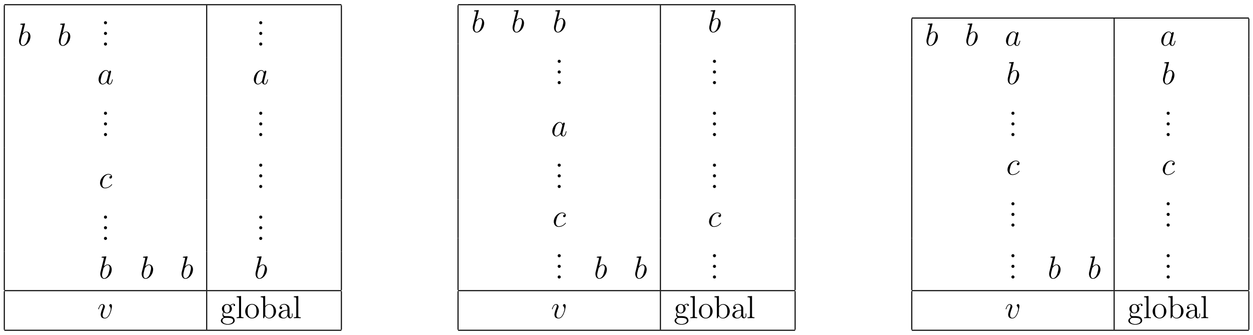
Theorem 10.1 *(**Arrow**) Any algorithm for creating a global ranking from individual rankings of three or more elements in which the global ranking satisfies unanimity and independence of irrelevant alternatives is a dictatorship.*

Proof: Let a, b, and c be distinct items. Consider a set of rankings in which each individual ranks b either first or last. Some individuals may rank b first and others may rank b last. For this set of rankings, the global ranking must put b first or last. Suppose to the contrary that b is not first or last in the global ranking. Then there exist a and c where the global ranking puts $a > b$ and $b > c$. By transitivity, the global ranking puts $a > c$. Note that all individuals can move c above a without affecting the order of b and a or the order of b and c since b was first or last on each list. Thus, by independence of irrelevant alternatives, the global ranking would continue to rank $a > b$ and $b > c$ even if all individuals moved c above a since that would not change the individuals relative order of a and b or the individuals relative order of b and c. But then by unanimity, the global ranking would need to put $c > a$, a contradiction. We conclude that the global ranking puts b first or last.

Consider a set of rankings in which every individual ranks b last. By unanimity, the global ranking must also rank b last. Let the individuals, one by one, move b from bottom to top leaving the other rankings in place. By unanimity, the global ranking must eventually move b from the bottom all the way to the top. When b first moves, it must move all the way to the top by the previous argument. Let v be the first individual whose change causes the global ranking of b to change.

We now argue that v is a dictator. First, we argue that v is a dictator for any pair ac not involving b. We will refer to three rankings of v (see Figure 10.1). The first ranking of v is the ranking prior to v moving b from the bottom to the top and the second is the ranking just after v has moved b to the top. Choose any pair ac where a is above c in v's ranking. The third ranking of v is obtained by moving a above b in the second ranking so that $a > b > c$ in v's ranking. By independence of irrelevant alternatives, the global ranking after v has switched to the third ranking puts $a > b$ since all individual ab votes are the same as just before v moved b to the top of his ranking. At that time the global ranking placed $a > b$. Similarly $b > c$ in the global ranking since all individual bc votes are the same as just after v moved b to the top causing b to move to the top in the global ranking. By transitivity the global ranking must put $a > c$ and thus the global ranking of a and c agrees with v.

Now all individuals except v can modify their rankings arbitrarily while leaving b in its extreme position and by independence of irrelevant alternatives, this does not affect the



first ranking second ranking third ranking

Figure 10.1: The three rankings that are used in the proof of Theorem 10.1.

global ranking of $a > b$ or of $b > c$. Thus, by transitivity this does not affect the global ranking of a and c. Next, all individuals except v can move b to any position without affecting the global ranking of a and c.

At this point we have argued that independent of other individuals' rankings, the global ranking of a and c will agree with v's ranking. Now v can change its ranking arbitrarily, provided it maintains the order of a and c, and by independence of irrelevant alternatives the global ranking of a and c will not change and hence will agree with v. Thus, we conclude that for all a and c, the global ranking agrees with v independent of the other rankings except for the placement of b. But other rankings can move b without changing the global order of other elements. Thus, v is a dictator for the ranking of any pair of elements not involving b.

Note that v changed the relative order of a and b in the global ranking when it moved b from the bottom to the top in the previous argument. We will use this in a moment.

The individual v is also a dictator over every pair ab. Repeat the construction showing that v is a dictator for every pair ac not involving b only this time place c at the bottom. There must be an individual v_c who is a dictator for any pair such as ab not involving c. Since both v and v_c can affect the global ranking of a and b independent of each other, it must be that v_c is actually v. Thus, the global ranking agrees with v no matter how the other voters modify their rankings. ∎

10.2 Hare System for Voting

One voting system would be to have everyone vote for their favorite candidate. If some candidate receives a majority of votes, he or she is declared the winner. If no candidate receives a majority of votes, the candidate with the fewest votes is dropped from the slate and the process is repeated.

The Hare system implements this method by asking each voter to rank all the candidates. Then one counts how many voters ranked each candidate as number one. If no candidate receives a majority, the candidate with the fewest number one votes is dropped from each voters ranking. If the dropped candidate was number one on some voters list, then the number two candidate becomes that voter's number one choice. The process of counting the number one rankings is then repeated.

Although the Hare system is widely used it fails to satisfy Arrow' axioms as all voting systems must. Consider the following situation in which there are 21 voters that fall into four categories. Voters within a category rank individuals in the same order.

Category	Number of voters in category	Preference order
1	7	abcd
2	6	bacd
3	5	cbad
4	3	dcba

The Hare system would first eliminate d since d gets only three rank one votes. Then it would eliminate b since b gets only six rank one votes whereas a gets seven and c gets eight. At this point a is declared the winner since a has thirteen votes to c's eight votes.

Now assume that Category 4 voters who prefer b to a move a up to first place. Then the election proceeds as follows. In round one, d is eliminated since it gets no rank one votes. Then c with five votes is eliminated and b is declared the winner with 11 votes. Note that by moving a up, category 4 voters were able to deny a the election and get b to win, whom they prefer over a.

10.3 Compressed Sensing and Sparse Vectors

Given a function $x(t)$, one can represent the function by the composition of sinusoidal functions. Basically one is representing the time function by its frequency components. The transformation from the time representation of a function to it frequency representation is accomplished by a Fourier transform. The Fourier transform of a function $x(t)$ is given by

$$f(\omega) = \int x(t) e^{-2\pi\omega t} dt$$

Converting the frequency representation back to the time representation is done by the inverse Fourier transformation

$$x(t) = \int f(\omega) e^{-2\pi\omega t} d\omega$$

In the discrete case, $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$ and $\mathbf{f} = [f_0, f_1, \ldots, f_{n-1}]$. The Fourier transform and its inverse are $\mathbf{f} = A\mathbf{x}$ with $a_{ij} = \omega^{ij}$ where ω is the principle n^{th} root of unity.

There are many other transforms such as the Laplace, wavelets, chirplets, etc. In fact, any nonsingular $n \times n$ matrix can be used as a transform.

If one has a discrete time sequence $\mathbf{x}$ of length n, the Nyquist theorem states that n coefficients in the frequency domain are needed to represent the signal $\mathbf{x}$. However, if the signal $\mathbf{x}$ has only s nonzero elements, even though one does not know which elements they are, one can recover the signal by randomly selecting a small subset of the coefficients in the frequency domain. It turns out that one can reconstruct sparse signals with far fewer samples than one might suspect and an area called compressed sampling has emerged with important applications.

Motivation

Let A be an $n \times d$ matrix with n much smaller than d whose elements are generated by independent Gaussian processes. Let $\mathbf{x}$ be a sparse d-dimensional vector with at most s nonzero coordinates, $s << d$. $\mathbf{x}$ is called the signal and A is the "measurement" matrix. What we measure are the components of the n dimensional vector $A\mathbf{x}$. We ask if we can recover the signal $\mathbf{x}$ from measurements $A\mathbf{x}$, where the number n of measurements is much smaller than the dimension d? We have two advantages over an arbitrary system of linear equations. First, the solution $\mathbf{x}$ is known to be sparse and second we have the choice of the measurement matrix A.

In many applications, the signal is sparse in either the time domain or the frequency domain. For images, it is often the case that in the frequency domain very few frequencies have significant amplitude. If we zero out small frequency amplitudes, we get a sparse frequency representation of the signal. It is wasteful to measure each of the d components of the signal $\mathbf{x}$, most of which are zero. Instead, we measure n linear combinations of components, the linear combinations form A. In applications, we choose the matrix A. A usual choice is a matrix whose entries are independent zero mean, unit variance Gaussian random variables. Since we have no control over the signal, our system needs to recover any signal. We will show that n needs to depend essentially only on s, not on d.

10.3.1 Unique Reconstruction of a Sparse Vector

A vector is said to be s-sparse if it has at most s nonzero elements. Let $\mathbf{x}$ be a d-dimensional, s-sparse vector with $s << d$. Consider solving $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ where A is an $n \times d$ matrix with $n < d$. The set of solutions to $A\mathbf{x} = \mathbf{b}$ is a subspace. However, if we restrict ourselves to sparse solutions, under certain conditions on A there is a unique s-sparse solution. Suppose that there were two s-sparse solutions, $\mathbf{x_1}$ and $\mathbf{x_2}$. Then $\mathbf{x_1} - \mathbf{x_2}$

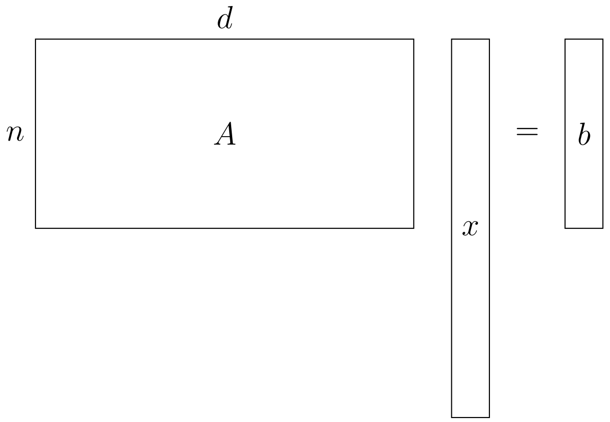


Figure 10.2: $A\mathbf{x} = \mathbf{b}$ has a vector space of solutions but possibly only one sparse solution.

would be a $2s$-sparse solution to the homogeneous system $A\mathbf{x} = \mathbf{0}$. A $2s$-sparse solution to the homogeneous equation $A\mathbf{x} = \mathbf{0}$ requires that some $2s$ columns of A be linearly dependent. Unless A has $2s$ linearly dependent columns there can be only one s-sparse solution.

Now suppose n is $\Omega(s^2)$ and we pick an $n \times d$ matrix A with random independent zero mean, unit variance Gaussian entries. Take any subset of $2s$ columns of A. Since we have already seen in Chapter 2 that each of these $2s$ vectors is likely to be essentially orthogonal to the space spanned by the previous vectors, the sub-matrix is unlikely to be singular. This intuition can be made rigorous.

To find a sparse solution to $A\mathbf{x} = \mathbf{b}$, one would like to minimize the zero norm $\|\mathbf{x}\|_0$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$. This is a computationally hard problem. There are techniques to minimize a convex function over a convex set. But $||\mathbf{x}||_0$ is not a convex function. With no further hypotheses, it is NP-hard. With this in mind, we use the one norm as a proxy for the zero norm and minimize the one norm $\|\mathbf{x}\|_1$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$. Although this problem appears to be nonlinear, it can be solved by linear programming by writing $\mathbf{x} = \mathbf{u} - \mathbf{v}$, $\mathbf{u} \geq 0$, and $\mathbf{v} \geq 0$, and then minimizing the linear function $\sum\limits_i u_i + \sum\limits_i v_i$ subject to $A\mathbf{u}$-$A\mathbf{v}$=$\mathbf{b}$, $\mathbf{u} \geq 0$, and $\mathbf{v} \geq 0$.

Under what conditions will minimizing $\|\mathbf{x}\|_1$ over $\{\mathbf{x}|A\mathbf{x} = \mathbf{b}\}$ recover the s-sparse solution to $A\mathbf{x}$=$\mathbf{b}$? If $g(\mathbf{x})$ is a convex function, then any local minimum of g is a global minimum. If $g(\mathbf{x})$ is differentiable at its minimum, the gradient ∇g must be zero there. However, the 1-norm is not differentiable at its minimum. Thus, we introduce the concept of a subgradient of a convex function. Where the function is differentiable the subgradient is just the gradient. Where the function is not differentiable, the sub gradient is any line touching the function at the point that lies totally below the function. See Figure 10.3.

Subgradients are defined as follows. A *subgradient* of a function g at a point $\mathbf{x_0}$, is a

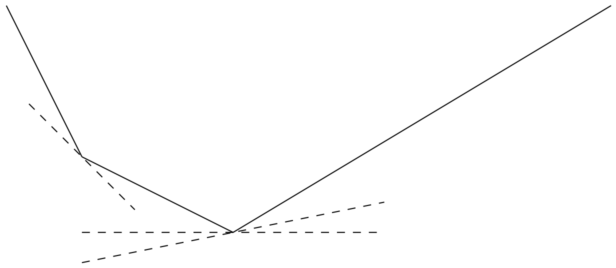


Figure 10.3: Some subgradients for a function that is not everywhere differentiable.

vector $\nabla g(\mathbf{x_0})$ satisfying $g\left(\mathbf{x_0}+\Delta\mathbf{x}\right) \geq g\left(\mathbf{x_0}\right)+\left(\nabla g\right)^T\Delta\mathbf{x}$ for any vector $\Delta\mathbf{x}$. A point is a minimum for a convex function if there is a subgradient at that point with slope zero.

Consider the function $\|x\|_1$, where x is a real variable. For $x<0$, the subgradient equals the gradient and has value -1. For $x>0$, the subgradient equals the gradient and has value 1. At $x=0$, the subgradient can be any value in the range [-1,1]. The following proposition generalizes this example to the 1-norm function in d-space.

Proposition 10.2 *A vector* $\mathbf{v}$ *is a subgradient of the 1-norm function* $||\mathbf{x}||_1$ *at* $\mathbf{x}$ *if and only if it satisfies the three conditions below:*

1. $v_i=-1$ *for all* i *in* I_1 *where,* $I_1=\{i|x_i<0\}$*,*
2. $v_i=1$ *for all* i *in* I_2 *where,* $I_2=\{i|x_i>0\}$*,*
3. *and* v_i *in* $[-1,1]$ *for all* i *in* I_3 *where,* $I_3=\{i|x_i=0\}$*.*

Proof: It is easy to see that for any vector $\mathbf{y}$,

$$||\mathbf{x}+\mathbf{y}||_1-||\mathbf{x}||_1 \geq -\sum_{i\in I_1} y_i+\sum_{i\in I_2} y_i+\sum_{i\in I_3}|y_i|.$$

If i is in I_1, x_i is negative. If y_i is also negative, then $||x_i+y_i||_1=||x_i||_1+||y_i||_1$ and thus $||x_i+y_i||_1-||x_i||_1=||y_i||_1=-y_i$. If y_i is positive and less than $||x_i||_1$, then $||x_i+y_i||_1=||x_i||-y_i$ and thus $||x_i+y_i||_1-||x_i||=-y_i$. If y_i is positive and greater than or equal to $||x_i||_1$, then $||x_i+y_i||_1=y_i-||x_i||_1$ and thus $||x_i+y_i||_1-||x_i||_1=y_i-2||x_i||_1\geq -y_i$. Similar reasoning establishes the case for i in I_2 or I_3.

If $\mathbf{v}$ satisfies the conditions in the proposition, then $||\mathbf{x}+\mathbf{y}||_1 \geq ||\mathbf{x}||_1+\mathbf{v}^T\mathbf{y}$ as required. Now for the converse, suppose that $\mathbf{v}$ is a subgradient. Consider a vector $\mathbf{y}$ that is zero in all components except the first and y_1 is nonzero with $y_1=\pm\varepsilon$ for a small $\varepsilon>0$. If $1\in I_1$, then $||\mathbf{x}+\mathbf{y}||_1-||\mathbf{x}||_1=-y_1$ which implies that $-y_1\geq v_1y_1$. Choosing $y_1=\varepsilon$, gives $-1\geq v_1$ and choosing $y_1=-\varepsilon$, gives $-1\leq v_1$. So $v_1=-1$. Similar reasoning gives the second condition. For the third condition, choose i in I_3 and set $y_i=\pm\varepsilon$ and argue similarly. ■

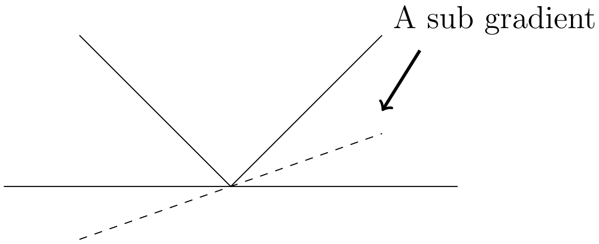


Figure 10.4: Illustration of a subgradient for $|\mathbf{x}|_1$ at $\mathbf{x} = 0$

To characterize the value of $\mathbf{x}$ that minimizes $\|\mathbf{x}\|_1$ subject to $A\mathbf{x}$=$\mathbf{b}$, note that at the minimum $\mathbf{x_0}$, there can be no downhill direction consistent with the constraint $A\mathbf{x}$=$\mathbf{b}$. Thus, if the direction $\Delta\mathbf{x}$ at $\mathbf{x_0}$ is consistent with the constraint $A\mathbf{x}$=$\mathbf{b}$*, that is* $A\Delta\mathbf{x}$*=0* so that $A\left(\mathbf{x_0} + \Delta\mathbf{x}\right) = \mathbf{b}$, any subgradient ∇ for $\|\mathbf{x}\|_1$ at $\mathbf{x_0}$ must satisfy $\nabla^T\Delta\mathbf{x} = 0$.

A sufficient but not necessary condition for $\mathbf{x_0}$ to be a minimum is that there exists some $\mathbf{w}$ such that the sub gradient at $\mathbf{x_0}$ is given by $\nabla = A^T\mathbf{w}$. Then for any $\Delta\mathbf{x}$ such that $A\Delta\mathbf{x} = 0$, $\nabla^T\Delta\mathbf{x} = \mathbf{w}^T A\Delta\mathbf{x} = \mathbf{w}^T \cdot \mathbf{0} = 0$. That is, for any direction consistent with the constraint $A\mathbf{x} = \mathbf{b}$, the subgradient is zero and hence $\mathbf{x_0}$ is a minimum.

10.3.2 The Exact Reconstruction Property

Theorem 10.3 below gives a condition that guarantees that a solution $\mathbf{x_0}$ to $A\mathbf{x} = \mathbf{b}$ is the unique minimum 1-norm solution to $A\mathbf{x} = \mathbf{b}$. This is a sufficient condition, but not necessary condition.

Theorem 10.3 *Suppose* $\mathbf{x_0}$ *satisfies* $A\mathbf{x_0} = \mathbf{b}$*. If there is a subgradient* ∇ *to the 1-norm function at* $\mathbf{x_0}$ *for which there exists a* $\mathbf{w}$ *where* $\nabla = A^T\mathbf{w}$ *and the columns of* A *corresponding to nonzero components of* $\mathbf{x_0}$ *are linearly independent, then* $\mathbf{x_0}$ *minimizes* $\|\mathbf{x}\|_1$ *subject to* $A\mathbf{x}$=$\mathbf{b}$*. Furthermore, these conditions imply that* $\mathbf{x_0}$ *is the unique minimum.*

Proof: We first show that $\mathbf{x_0}$ minimizes $\|\mathbf{x}\|_1$. Suppose $\mathbf{y}$ is another solution to $A\mathbf{x} = \mathbf{b}$. We need to show that $||y||_1 \geq ||x_0||_1$. Let $\mathbf{z} = \mathbf{y} - \mathbf{x_0}$. Then $A\mathbf{z} = A\mathbf{y} - A\mathbf{x_0} = \mathbf{0}$. Hence, $\nabla^T\mathbf{z} = (A^T\mathbf{w})^T\mathbf{z} = \mathbf{w}^T A\mathbf{z} = 0$. Now, since ∇ is a subgradient of the 1-norm function at $\mathbf{x_0}$,

$$||\mathbf{y}||_1 = ||\mathbf{x_0} + \mathbf{z}||_1 \geq ||\mathbf{x_0}||_1 + \nabla^T \cdot \mathbf{z} = ||\mathbf{x}_0||_1$$

and so we have that $||\mathbf{x_0}||_1$ minimizes $||\mathbf{x}||_1$ over all solutions to $A\mathbf{x} = \mathbf{b}$.

Suppose $\tilde{\mathbf{x}}_0$ were another minimum. Then ∇ is also a subgradient at $\tilde{\mathbf{x}}_0$ as it is at $\mathbf{x_0}$. To see this, for $\Delta\mathbf{x}$ such that $A\Delta\mathbf{x} = 0$,

$$\|\tilde{\mathbf{x}}_0 + \Delta\mathbf{x}\|_1 = \left\| \mathbf{x}_0 + \underbrace{\tilde{\mathbf{x}}_\mathbf{0} - \mathbf{x}_0 + \Delta\mathbf{x}}_{\alpha} \right\|_1 \geq \|\mathbf{x_0}\|_1 + \nabla^T\left(\tilde{\mathbf{x}}_\mathbf{0} - \mathbf{x_0} + \Delta\mathbf{x}\right).$$

The above equation follows from the definition of ∇ being a subgradient for the one norm function, $\|\|_1$, at $\mathbf{x_0}$. Thus,

$$\|\tilde{\mathbf{x}}_\mathbf{0} + \Delta\mathbf{x}\|_1 \geq \|\mathbf{x_0}\|_1 + \nabla^T (\tilde{\mathbf{x}}_\mathbf{0} - \mathbf{x_0}) + \nabla^T \Delta\mathbf{x}.$$

But

$$\nabla^T (\tilde{\mathbf{x}}_\mathbf{0} - \mathbf{x_0}) = \mathbf{w}^T A (\tilde{\mathbf{x}}_\mathbf{0} - \mathbf{x_0}) = \mathbf{w}^T (\mathbf{b} - \mathbf{b}) = 0.$$

Hence, since $\tilde{\mathbf{x_0}}$ being a minimum means $||\tilde{\mathbf{x_0}}||_1 = ||\mathbf{x_0}||_1$,

$$\|\tilde{\mathbf{x}}_\mathbf{0} + \Delta\mathbf{x}\|_1 \geq \|\mathbf{x_0}\|_1 + \nabla^T \Delta\mathbf{x} = ||\tilde{\mathbf{x_0}}||_1 + \nabla^T \Delta\mathbf{x}.$$

This implies that ∇ is a sub gradient at $\tilde{\mathbf{x}}_\mathbf{0}$.

Now, ∇ is a subgradient at both $\mathbf{x_0}$ and $\tilde{\mathbf{x}}_0$. By Proposition 10.2, we must have that $(\nabla)_i = \text{sgn}((x_0)_i) = \text{sgn}((\tilde{x}_0)_i)$, whenever either is nonzero and $|(\nabla)_i| < 1$, whenever either is 0. It follows that $\mathbf{x_0}$ and $\tilde{\mathbf{x}}_\mathbf{0}$ have the same sparseness pattern. Since $A\mathbf{x_0} = \mathbf{b}$ and $A\tilde{\mathbf{x}}_\mathbf{0} = \mathbf{b}$ and $\mathbf{x_0}$ and $\tilde{\mathbf{x}}_\mathbf{0}$ are both nonzero on the same coordinates, and by the assumption that the columns of A corresponding to the nonzeros of $\mathbf{x_0}$ and $\tilde{\mathbf{x}}_\mathbf{0}$ are independent, it must be that $\mathbf{x_0} = \tilde{\mathbf{x_0}}$. ■

10.3.3 Restricted Isometry Property

Next we introduce the restricted isometry property that plays a key role in exact reconstruction of sparse vectors. A matrix A satisfies the *restricted isometry property, RIP,* if for any s-sparse $\mathbf{x}$ there exists a δ_s such that

$$(1 - \delta_s) |\mathbf{x}|^2 \leq |A\mathbf{x}|^2 \leq (1 + \delta_s) |\mathbf{x}|^2 . \tag{10.1}$$

Isometry is a mathematical concept; it refers to linear transformations that exactly preserve length such as rotations. If A is an $n \times n$ isometry, all its eigenvalues are ± 1 and it represents a coordinate system. Since a pair of orthogonal vectors are orthogonal in all coordinate system, for an isometry A and two orthogonal vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}^T A^T A\mathbf{y} = 0$. We will prove approximate versions of these properties for matrices A satisfying the restricted isometry property. The approximate versions will be used in the sequel.

A piece of notation will be useful. For a subset S of columns of A, let A_S denote the submatrix of A consisting of the columns of S.

Lemma 10.4 *If A satisfies the restricted isometry property, then*

1. *For any subset S of columns with $|S| = s$, the singular values of A_S are all between $1 - \delta_s$ and $1 + \delta_s$.*

2. *For any two orthogonal vectors $\mathbf{x}$ and $\mathbf{y}$, with supports of size s_1 and s_2 respectively, $|\mathbf{x}^T A^T A\mathbf{y}| \leq 5|\mathbf{x}||\mathbf{y}|(\delta_{s_1} + \delta_{s_2})$.*

Proof: Item 1 follows from the definition. To prove the second item, assume without loss of generality that $|\mathbf{x}| = |\mathbf{y}| = 1$. Since $\mathbf{x}$ and $\mathbf{y}$ are orthogonal, $|\mathbf{x} + \mathbf{y}|^2 = 2$. Consider $|A(\mathbf{x}+\mathbf{y})|^2$. This is between $2(1-\delta_{s_1}+\delta_{s_2})^2$ and $2(1+\delta_{s_1}+\delta_{s_2})^2$ by the restricted isometry property. Also $|A\mathbf{x}|^2$ is between $(1 - \delta_{s_1})^2$ and $(1 + \delta_{s_1})^2$ and $|A\mathbf{y}|^2$ is between $(1 - \delta_{s_2})^2$ and $(1 + \delta_{s_2})^2$. Since

$$\begin{aligned} 2\mathbf{x}^T A^T A\mathbf{y} &= (\mathbf{x} + \mathbf{y})^T A^T A(\mathbf{x} + \mathbf{y}) - \mathbf{x}^T A^T A\mathbf{x} - \mathbf{y}^T A^T A\mathbf{y} \\ &= |A(\mathbf{x} + \mathbf{y})|^2 - |A\mathbf{x}|^2 - |A\mathbf{y}|^2, \end{aligned}$$

it follows that

$$\begin{aligned} |2\mathbf{x}^T A^T A\mathbf{y}| &\leq 2(1 + \delta_{s_1} + \delta_{s_2})^2 - (1 - \delta_{s_1})^2 - (1 - \delta_{s_2})^2 \\ & 6(\delta_{s_1} + \delta_{s_2}) + (\delta_{s_1}^2 + \delta_{s_2}^2 + 4\delta_{s_1} + 4\delta_{s_2}) \leq 9(\delta_{s_1} + \delta_{s_2}). \end{aligned}$$

Thus, for arbitrary $\mathbf{x}$ and $\mathbf{y}$ $|\mathbf{x}^T A^T A\mathbf{y}| \leq (9/2)|\mathbf{x}||\mathbf{y}|(\delta_{s_1} + \delta_{s_2})$. ∎

Theorem 10.5 *Suppose A satisfies the restricted isometry property with*

$$\delta_{s+1} \leq \frac{1}{10\sqrt{s}}.$$

Suppose $\mathbf{x_0}$ has at most s nonzero coordinates and satisfies $A\mathbf{x} = \mathbf{b}$. Then a subgradient $\nabla||(\mathbf{x_0})||_1$ for the 1-norm function exists at $\mathbf{x_0}$ which satisfies the conditions of Theorem 10.3 and so $\mathbf{x_0}$ is the unique minimum 1-norm solution to $A\mathbf{x} = \mathbf{b}$.

Proof: Let

$$S = \{i|(\mathbf{x_0})_\mathbf{i} \neq 0\}$$

be the support of $\mathbf{x_0}$ and let $\bar{S} = \{i|(\mathbf{x_0})_\mathbf{i} = \mathbf{0}\}$ be the complement set of coordinates. To find a subgradient $\mathbf{u}$ at $\mathbf{x_0}$ satisfying Theorem 10.3, search for a $\mathbf{w}$ such that $\mathbf{u} = A^T\mathbf{w}$ where for coordinates in which $\mathbf{x_0} \neq 0$, $\mathbf{u} = sgn\,(\mathbf{x_0})$ and for the remaining coordinates the 2-norm of $\mathbf{u}$ is minimized. Solving for $\mathbf{w}$ is a least squares problem. Let $\mathbf{z}$ be the vector with support S, with $z_i = \text{sgn}(\mathbf{x_0})$ on S. Consider the vector $\mathbf{w}$ defined by

$$\mathbf{w} = A_S \left(A_S^T A_S\right)^{-1} \mathbf{z}.$$

This happens to be the solution of the least squares problem, but we do not need this fact. We only state it to tell the reader how we came up with this expression. Note that A_S has independent columns from the restricted isometry property assumption, and so $A_S^T A_S$ is invertible. We will prove that this $\mathbf{w}$ satisfies the conditions of Theorem 10.3. First, for coordinates in S,

$$(A^T\mathbf{w})_S = (A_S)^T A_S (A_S^T A_S)^{-1}\mathbf{z} = \mathbf{z}$$

as required.

For coordinates in $\bar{S}$, we have

$$(A^T\mathbf{w})_{\bar{S}} = (A_{\bar{S}})^T A_S (A_S^T A_S)^{-1}\mathbf{z}.$$

Now, the eigenvalues of $A_S^T A_S$, which are the squares of the singular values of A_S, are between $(1-\delta_s)^2$ and $(1+\delta_s)^2$. So $||(A_S^T A_S)^{-1}|| \le \frac{1}{(1-\delta_S)^2}$. Letting $\mathbf{p} = (A_S^T A_S)^{-1}\mathbf{z}$, we have $|\mathbf{p}| \le \frac{\sqrt{s}}{(1-\delta_S)^2}$. Write $A_s\mathbf{p}$ as $A\mathbf{q}$, where $\mathbf{q}$ has all coordinates in $\bar{S}$ equal to zero. Now, for $j \in \bar{S}$

$$(A^T\mathbf{w})_j = e_j^T A^T A\mathbf{q}$$

and part (2) of Lemma 10.4 gives $|(A^T\mathbf{w})_j| \le 9\delta_{s+1}\sqrt{s}/(1-\delta_s^2) \le 1/2$ establishing the Theorem 10.3 holds. ■

A Gaussian matrix is a matrix where each element is an independent Gaussian variable. Gaussian matrices satisfy the restricted isometry property. (Exercise **??**)

10.4 Applications

10.4.1 Sparse Vector in Some Coordinate Basis

Consider $A\mathbf{x} = \mathbf{b}$ where A is a square $n \times n$ matrix. The vectors $\mathbf{x}$ and $\mathbf{b}$ can be considered as two representations of the same quantity. For example, $\mathbf{x}$ might be a discrete time sequence with $\mathbf{b}$ the frequency spectrum of $\mathbf{x}$ and the matrix A the Fourier transform. The quantity $\mathbf{x}$ can be represented in the time domain by $\mathbf{x}$ and in the frequency domain by its Fourier transform $\mathbf{b}$. In fact, any orthonormal matrix can be thought of as a transformation and there are many important transformations other than the Fourier transformation.

Consider a transformation A and a signal $\mathbf{x}$ in some standard representation. Then $\mathbf{y} = A\mathbf{x}$ transforms the signal $\mathbf{x}$ to another representation $\mathbf{y}$. If A spreads any sparse signal $\mathbf{x}$ out so that the information contained in each coordinate in the standard basis is spread out to all coordinates in the second basis, then the two representations are said to be *incoherent*. A signal and its Fourier transform are one example of incoherent vectors. This suggests that if $\mathbf{x}$ is sparse, only a few randomly selected coordinates of its Fourier transform are needed to reconstruct $\mathbf{x}$. In the next section we show that a signal cannot be too sparse in both its time domain and its frequency domain.

10.4.2 A Representation Cannot be Sparse in Both Time and Frequency Domains

We now show that there is an uncertainty principle that states that a time signal cannot be sparse in both the time domain and the frequency domain. If the signal is of length n, then the product of the number of nonzero coordinates in the time domain and the number of nonzero coordinates in the frequency domain must be at least n. We first prove two technical lemmas.

In dealing with the Fourier transform it is convenient for indices to run from 0 to $n-1$ rather than from 1 to n. Let $x_0, x_1, \ldots, x_{n-1}$ be a sequence and let $f_0, f_1, \ldots, f_{n-1}$ be its discrete Fourier transform. Let $i = \sqrt{-1}$. Then $f_j = \frac{1}{\sqrt{n}} \sum\limits_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk}, \quad j = 0, \ldots, n-1$. In matrix form $\mathbf{f} = Z\mathbf{x}$ where $z_{jk} = e^{-\frac{2\pi i}{n} jk}$.

$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{2\pi i}{n}} & e^{-\frac{2\pi i}{n}2} & \cdots & e^{-\frac{2\pi i}{n}(n-1)} \\ \vdots & \vdots & & \vdots & \\ 1 & e^{-\frac{2\pi i}{n}(n-1)} & e^{-\frac{2\pi i}{n}2(n-1)} & \cdots & e^{-\frac{2\pi i}{n}(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

If some of the elements of $\mathbf{x}$ are zero, delete the zero elements of $\mathbf{x}$ and the corresponding columns of the matrix. To maintain a square matrix, let n_x be the number of nonzero elements in $\mathbf{x}$ and select n_x consecutive rows of the matrix. Normalize the columns of the resulting submatrix by dividing each element in a column by the column element in the first row. The resulting submatrix is a Vandermonde matrix that looks like

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \\ a^2 & b^2 & c^2 & d^2 \\ a^3 & b^3 & c^3 & d^3 \end{pmatrix}$$

and is nonsingular.

Lemma 10.6 *If $x_0, x_1, \ldots, x_{n-1}$ has n_x nonzero elements, then $f_0, f_1, \ldots, f_{n-1}$ cannot have n_x consecutive zeros.*

Proof: Let $i_1, i_2, \ldots, i_{n_x}$ be the indices of the nonzero elements of $\mathbf{x}$. Then the elements of the Fourier transform in the range $k = m+1, m+2, \ldots, m+n_x$ are

$$f_k = \frac{1}{\sqrt{n}} \sum_{j=1}^{n_x} x_{i_j} e^{\frac{-2\pi i}{n} k i_j}$$

Note the use of i as $\sqrt{-1}$ and the multiplication of the exponent by i_j to account for the actual location of the element in the sequence. Normally, if every element in the sequence was included, we would just multiply by the index of summation.

Convert the equation to matrix form by defining $z_{kj} = \frac{1}{\sqrt{n}} \exp(-\frac{2\pi i}{n} k i_j)$ and write $\mathbf{f} = Z\mathbf{x}$. Actually instead of $\mathbf{x}$, write the vector consisting of the nonzero elements of $\mathbf{x}$. By its definition, $\mathbf{x} \neq 0$. To prove the lemma we need to show that $\mathbf{f}$ is nonzero. This will be true provided Z is nonsingular. If we rescale Z by dividing each column by its leading entry we get the Vandermonde determinant which is nonsingular. ■

$$\begin{array}{ccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & z & z^2 & z^3 & z^4 & z^5 & z^6 & z^7 & z^8 \\
1 & z^2 & z^4 & z^6 & z^8 & z & z^3 & z^5 & z^7 \\
1 & z^3 & z^6 & 1 & z^3 & z^6 & 1 & z^3 & z^6 \\
1 & z^4 & z^8 & z^3 & z^7 & z^2 & z^6 & z & z^5 \\
1 & z^5 & z & z^6 & z^2 & z^7 & z^3 & z^8 & z^4 \\
1 & z^6 & z^3 & 1 & z^6 & z^3 & 1 & z^6 & z^3 \\
1 & z^7 & z^5 & z^3 & z & z^8 & z^6 & z^4 & z^2 \\
1 & z^8 & z^7 & z^6 & z^5 & z^4 & z^3 & z^2 & z
\end{array}$$

Figure 10.5: The matrix Z for n=9.

Theorem 10.7 *Let n_x be the number of nonzero elements in* $\mathbf{x}$ *and let n_f be the number of nonzero elements in the Fourier transform of* $\mathbf{x}$. *Let n_x divide n. Then $n_x n_f \geq n$.*

Proof: If $\mathbf{x}$ has n_x nonzero elements, $\mathbf{f}$ cannot have a consecutive block of n_x zeros. Since n_x divides n there are $\frac{n}{n_x}$ blocks each containing at least one nonzero element. Thus, the product of nonzero elements in $\mathbf{x}$ and $\mathbf{f}$ is at least n. ■

Fourier transform of spikes prove that above bound is tight

To show that the bound in Theorem 10.7 is tight we show that the Fourier transform of the sequence of length n consisting of $\sqrt{n}$ ones, each one separated by $\sqrt{n}-1$ zeros, is the sequence itself. For example, the Fourier transform of the sequence 100100100 is 100100100. Thus, for this class of sequences, $n_x n_f = n$.

Theorem 10.8 *Let $S\left(\sqrt{n}, \sqrt{n}\right)$ be the sequence of 1's and 0's with $\sqrt{n}$ 1's spaced $\sqrt{n}$ apart. The Fourier transform of $S\left(\sqrt{n}, \sqrt{n}\right)$ is itself.*

Proof: Consider the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, \left(\sqrt{n}-1\right)\sqrt{n}$. These are the columns for which $S\left(\sqrt{n}, \sqrt{n}\right)$ has value 1. The element of the matrix Z in the row $j\sqrt{n}$ of column $k\sqrt{n}$, $0 \leq k < \sqrt{n}$ is $z^{nkj} = 1$. Thus, for these rows Z times the vector $S\left(\sqrt{n}, \sqrt{n}\right) = \sqrt{n}$ and the $1/\sqrt{n}$ normalization yields $f_{j\sqrt{n}} = 1$.

For rows whose index is not of the form $j\sqrt{n}$, the row b, $b \neq j\sqrt{n}$, $j \in \{0, \sqrt{n}, \ldots, \sqrt{n}-1\}$, the elements in row b in the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, \left(\sqrt{n}-1\right)\sqrt{n}$ are $1, z^b, z^{2b}, \ldots, z^{\left(\sqrt{n}-1\right)b}$ and thus $f_b = \frac{1}{\sqrt{n}}\left(1 + z^b + z^{2b} \cdots + z^{\left(\sqrt{n}-1\right)b}\right) = \frac{1}{\sqrt{n}}\frac{z^{\sqrt{n}b}-1}{z-1} = 0$ since $z^{b\sqrt{n}} = 1$ and $z \neq 1$.

Uniqueness of l_1 optimization

Consider a redundant representation for a sequence. One such representation would be representing a sequence as the concatenation of two sequences, one specified by its coordinates and the other by its Fourier transform. Suppose some sequence could be represented as a sequence of coordinates and Fourier coefficients sparsely in two different ways. Then by subtraction, the zero sequence could be represented by a sparse sequence. The representation of the zero sequence cannot be solely coordinates or Fourier coefficients. If y is the coordinate sequence in the representation of the zero sequence, then the Fourier portion of the representation must represent $-y$. Thus y and its Fourier transform would have sparse representations contradicting $n_x n_f \geq n$. Notice that a factor of two comes in when we subtract the two representations.

Suppose two sparse signals had Fourier transforms that agreed in almost all of their coordinates. Then the difference would be a sparse signal with a sparse transform. This is not possible. Thus, if one selects $\log n$ elements of their transform these elements should distinguish between these two signals.

10.4.3 Biological

There are many areas where linear systems arise in which a sparse solution is unique. One is in plant breading. Consider a breeder who has a number of apple trees and for each tree observes the strength of some desirable feature. He wishes to determine which genes are responsible for the feature so he can cross bread to obtain a tree that better expresses the desirable feature. This gives rise to a set of equations $A\mathbf{x} = \mathbf{b}$ where each row of the matrix A corresponds to a tree and each column to a position on the genone. See Figure 10.6. The vector $\mathbf{b}$ corresponds to the strength of the desired feature in each tree. The solution $\mathbf{x}$ tells us the position on the genone corresponding to the genes that account for the feature. It would be surprising if there were two small independent sets of genes that accounted for the desired feature. Thus, the matrix must have a property that allows only one sparse solution.

10.4.4 Finding Overlapping Cliques or Communities

Consider a graph that consists of several cliques. Suppose we can observe only low level information such as edges and we wish to identify the cliques. An instance of this problem is the task of identifying which of ten players belongs to which of two teams of five players each when one can only observe interactions between pairs of individuals. There is an interaction between two players if and only if they are on the same team. In this situation we have a matrix A with $\binom{10}{5}$ columns and $\binom{10}{2}$ rows. The columns represent possible teams and the rows represent pairs of individuals. Let $\mathbf{b}$ be the $\binom{10}{2}$ dimensional vector of observed interactions. Let $\mathbf{x}$ be a solution to $A\mathbf{x} = \mathbf{b}$. There is a sparse solution $\mathbf{x}$ where $\mathbf{x}$ is all zeros except for the two 1's for 12345 and 678910 where the two teams are {1,2,3,4,5} and {6,7,8,9,10}. The question is can we recover $\mathbf{x}$ from $\mathbf{b}$. If the matrix A had satisfied the restricted isometry condition, then we could surely do this. Although A does not satisfy the restricted isometry condition which guarantees recover of all sparse vectors, we can recover the sparse vector in the case where the teams are non overlapping or almost non overlapping. If A satisfied the restricted isometry property we would minimize $\|\mathbf{x}\|_1$ subject to $A\mathbf{x} = \mathbf{b}$. Instead, we minimize $\|\mathbf{x}\|_1$ subject to $\|A\mathbf{x} - \mathbf{b}\|_\infty \leq \varepsilon$ where we bound the largest error.

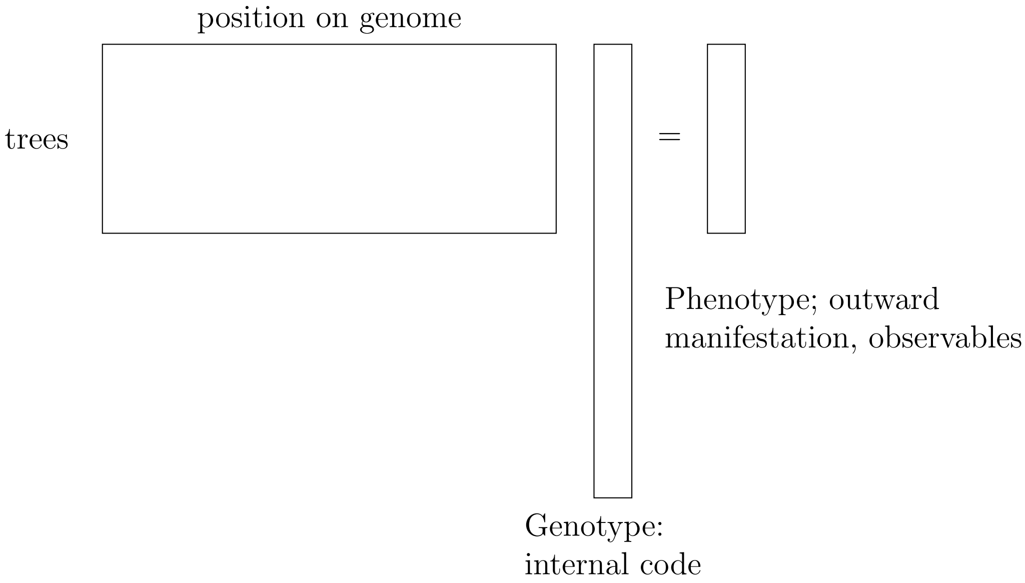


Figure 10.6: The system of linear equations used to find the internal code for some observable phenomenon.

10.4.5 Low Rank Matrices

Suppose L is a low rank matrix that has been corrupted by noise. That is, $M = L + R$. If the R is Gaussian, then principle component analysis will recover L from M. However, if L has been corrupted by several missing entries or several entries have a large noise added to them and they become outliers, then principle component analysis may be far off. However, if L is low rank and R is sparse, then L can be recovered effectively from $L + R$. To do this, find the L and R that minimize $\|L\|_* + \lambda \|R\|_1$. Here $\|L\|_*$ is the sum of the singular values of L. A small value of $\|L\|_*$ indicates a low rank matrix. Notice that we do not need to know the rank of L or the elements that were corrupted. All we need is that the low rank matrix L is not sparse and that the sparse matrix R is not low rank. We leave the proof as an exercise.

An example where low rank matrices that have been corrupted might occur is aerial photographs of an intersection. Given a long sequence of such photographs, they will be the same except for cars and people. If each photo is converted to a vector and the vector used to make a column of a matrix, then the matrix will be low rank corrupted by the traffic. Finding the original low rank matrix will separate the cars and people from the back ground.

10.5 Gradient

The gradient of a function $f(\mathbf{x})$ of d variables, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, at a point $\mathbf{x_0}$ is denoted $\bigtriangledown f(\mathbf{x_0})$. It is a d-dimensional vector with components $\frac{\partial f}{\partial x_1}(\mathbf{x_0})$, $\frac{\partial f}{\partial x_2}(\mathbf{x_0})$, $\ldots \frac{\partial f}{\partial x_d}(\mathbf{x_0})$, where $\frac{\partial f}{\partial x_i}$ are partial derivatives. Without explicitly stating we assume that the derivatives referred to exist. The rate of increase of the function f as we move from $\mathbf{x_0}$ in a direction $\mathbf{u}$ is easily seen to be $\bigtriangledown f(\mathbf{x_0}) \cdot \mathbf{u}$. So the direction of steepest descent is $-\bigtriangledown f(\mathbf{x_0})$; this is a natural direction to move in if we wish to minimize f. But by how much should we move? A large move may overshoot the minimum. [See figure (10.7).] A simple fix is to minimize f on the line from $\mathbf{x_0}$ in the direction of steepest descent by solving a one dimensional minimization problem. This gets us the next iterate $\mathbf{x_1}$ and we may repeat. Here, we will not discuss the issue of step-size any further. Instead, we focus on "infinitesimal" gradient descent, where, the algorithm makes infinitesimal moves in the $-\bigtriangledown f(\mathbf{x_0})$ direction. Whenever $\bigtriangledown \mathbf{f}$ is not the zero vector, we strictly decrease the function in the direction $-\bigtriangledown \mathbf{f}$, so the current point is not a minimum of the function f. Conversely, a point $\mathbf{x}$ where $\bigtriangledown \mathbf{f} = \mathbf{0}$ is called a *first-order local optimum* of f. In general, local minima do not have to be global minima (see (10.7)) and gradient descent may converge to a local minimum which is not a global minimum. In the special case when the function f is convex, this is not the case. A function f of a single variable x is said to be convex if for any two points x and y, the line joining $f(x)$ and $f(y)$ is above the curve $f(\cdot)$. A function of many variables is convex if on any line segment in its domain, it acts as a convex function of one variable on the line segment.

Definition 10.1 *A function f over a convex domain is a convex function if for any two points $\mathbf{x}, \mathbf{y}$ in the domain, and any λ in $[0,1]$ we have*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}).$$

The function is concave if the inequality is satisfied with $\geq$ instead of $\leq$.

Theorem 10.9 *Suppose f is a convex, differentiable, function defined on a closed bounded convex domain. Then any first-order local minimum is also a global minimum. Thus, infinitesimal gradient descent always reaches the global minimum.*

Proof: We will prove that if at a point $\mathbf{x}$, is a local minimum, then it must be a global minimum. If not, consider a global minimum point $\mathbf{y} \neq \mathbf{x}$. But on the line joining $\mathbf{x}$ and $\mathbf{y}$, the function must not go above the line joining $f(\mathbf{x})$ and $f(\mathbf{y})$. This means for an infinitesimal $\varepsilon > 0$, if we move distance ε from $\mathbf{x}$ towards $\mathbf{y}$, the function must decrease, so we cannot have $\bigtriangledown \mathbf{f}(\mathbf{x}) = \mathbf{0}$, contradicting the assumption that $\mathbf{x}$ is a local minimum. ∎

The second derivatives $\frac{\partial^2}{\partial x_i \partial x_j}$ form a matrix, called the Hessian, denoted $H(f(\mathbf{x}))$. The Hessian of f at $\mathbf{x}$ is a symmetric $d \times d$ matrix with (i,j) th entry $\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})$. The second derivative of f at $\mathbf{x}$ in the direction $\mathbf{u}$ is the rate of change of the first derivative as we move along $\mathbf{u}$ from $\mathbf{x}$. It is easy to see that it equals

$$\mathbf{u}^T\ H(f(\mathbf{x}))\mathbf{u}.$$

To see this, note that the second derivative of f along $\mathbf{u}$ is

$$\begin{aligned}\sum_j u_j \frac{\partial}{\partial x_j}\left(\bigtriangledown f(\mathbf{x})\cdot\mathbf{u}\right) &= \sum_j u_j \sum_i \frac{\partial}{\partial x_j}\left(u_i \frac{\partial f}{\partial x_i}\right)\\ &= \sum_{j,i} u_j u_i \frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}).\end{aligned}$$

Theorem 10.10 *Suppose f is a function from a closed convex domain D in $\mathbf{R}^d$ to the reals and the Hessian of f exists everywhere in D. Then f is convex (concave) on D if and only if the Hessian of f is positive (negative) semi-definite everywhere on D.*

Gradient descent requires the gradient to exist. But, even if the gradient is not always defined, one can minimize a convex function over a convex domain efficiently, i.e., in polynomial time. Here, the quote is added because of the lack of rigor in the statement, one can only find an approximate minimum and the time really depends on the error parameter as well as the presentation of the convex set. We do not go into these details here. But, in principle we can minimize a convex function over a convex domain. We can also maximize a concave function over a convex domain. However, in general, we do not have efficient procedures to maximize a convex function over a convex set. It is easy to see that at a first-order local minimum of a possibly non-convex function, the gradient vanishes. But second-order local decrease of the function may be possible. The steepest second-order decrease is in the direction of $\pm\mathbf{v}$, where, $\mathbf{v}$ is the eigenvector of the Hessian corresponding to the largest absolute valued eigenvalue.

10.6 Linear Programming

An optimization problem which has been carefully studied and is immensely useful is linear programming. We consider linear programming problem in the following form where A is an $m \times n$ matrix of rank m , $\mathbf{c}$ is $1 \times n$, $\mathbf{b}$ is $m \times 1$ and $\mathbf{x}$ is $n \times 1$) :

$$\max \mathbf{c}\cdot\mathbf{x} \qquad : \qquad A\mathbf{x} = \mathbf{b} \qquad ; \qquad x \geq 0.$$

Inequality constraints can be converted to this form by adding slack variables. Also, we can do Gaussian elimination on A and if it does not have rank m, we either find that the system of equations has no solution, whence we may stop or we can find and discard redundant equations. After this preprocessing, we may assume that A 's rows are independent.

The simplex algorithm is a classical method to solve linear programming problems. It is a vast subject and is well discussed in many texts. Here, we will discuss the ellipsoid algorithm which is in a sense based more on continuous mathematics and is closer to the spirit of this book.

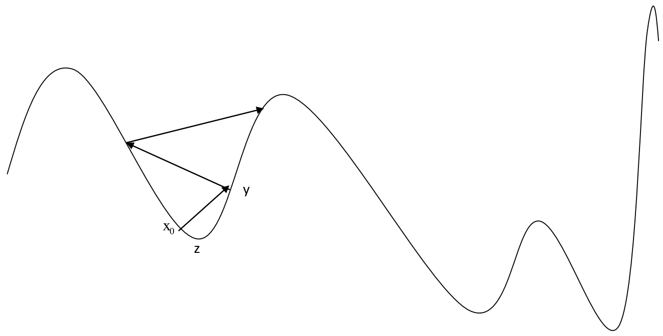


Figure 10.7

10.6.1 The Ellipsoid Algorithm

The first polynomial time algorithm for Linear Programming was developed by Khachian based on work of Iudin, Nemirovsky and Shor and is called the ellipsoid algorithm. The algorithm is best stated for the seemingly simpler problem of determining whether there is a solution to $A\mathbf{x} \leq \mathbf{b}$ and if so finding one. The ellipsoid algorithm starts with a large ball in d-space which is guaranteed to contain the polyhedron $A\mathbf{x} \leq \mathbf{b}$. Even though we do not yet know if the polyhedron is empty or non-empty, such a ball can be found. It checks if the center of the ball is in the polyhedron, if it is, we have achieved our objective. If not, we know from convex geometry (in particular, the Separating Hyperplane Theorem) that there is a hyperplane called the separating hyperplane through the center of the ball such that the whole polytope lies in one half space.

We then find an ellipsoid which contains the ball intersected with this half-space See Figure (10.8. The ellipsoid is guaranteed to contain $Ax \leq b$ as was the ball earlier. We now check if the center of the ellipsoid satisfies the inequalities. If not, there is a separating hyper plane again and we may repeat the process. After a suitable number of steps, either we find a solution to the original $A\mathbf{x} \leq \mathbf{b}$ or, we end up with a very small ellipsoid. Now if the original A and $\mathbf{b}$ had integer entries, one can ensure that the set $A\mathbf{x} \leq \mathbf{b}$, after a slight perturbation which preserves its emptiness/non-emptiness, has a volume of at least some $\epsilon > 0$ and if our ellipsoid has shrunk to a volume of less than this ϵ, then we know there is no solution and we can stop. Clearly this must happen within $\log_\rho V_0/\epsilon = O((V_0 d)/\epsilon)$, where V_0 is an upper bound on the initial volume and ρ is the factor by which the volume shrinks in each step. We do not go into details of how to get a value for V_0 here, but the important points are that (i) V_0 only occurs under the logarithm and (ii) the dependence on d is linear. These features ensure a polynomial time algorithm.

The main difficulty in proving fast convergence is to show that the volume of the ellipsoid shrinks by a certain factor in each step. Thus, the question can be phrased as suppose E is an ellipsoid with center $\mathbf{x_0}$ and consider the half-ellipsoid E' defined by

$$E' = \{\mathbf{x} : \mathbf{x} \in E \ ; \ \mathbf{a} \cdot (\mathbf{x} - \mathbf{x_0}) \geq 0\},$$

where, $\mathbf{a}$ is some unit length vector. Let $\hat{E}$ be the smallest volume ellipsoid containing E'. Show that

$$\frac{\mathrm{Vol}(\hat{E})}{\mathrm{Vol}(E)} \leq 1 - \rho$$

for some $\rho > 0$. A sequence of geometric reductions transforms this into a simple problem. First, observe that we can translate the entire picture and assume that $\mathbf{x_0} = 0$. Next, rotate the coordinate axes so that $\mathbf{a}$ is replaced by $(1, 0, 0, \ldots, 0)$. Finally, make a nonsingular linear transformation τ so that $\tau E = B = \{\mathbf{x} : |\mathbf{x}| = 1\}$, the unit sphere. The important point is that a nonsingular linear transformation τ multiplies the volumes of all sets by $|\det(\tau)|$, so that $\frac{\mathrm{Vol}(\hat{E})}{\mathrm{Vol}(E)} = \frac{\mathrm{Vol}(\tau(\hat{E}))}{\mathrm{Vol}(\tau(E))}$. Now, the following lemma answers the question raised.

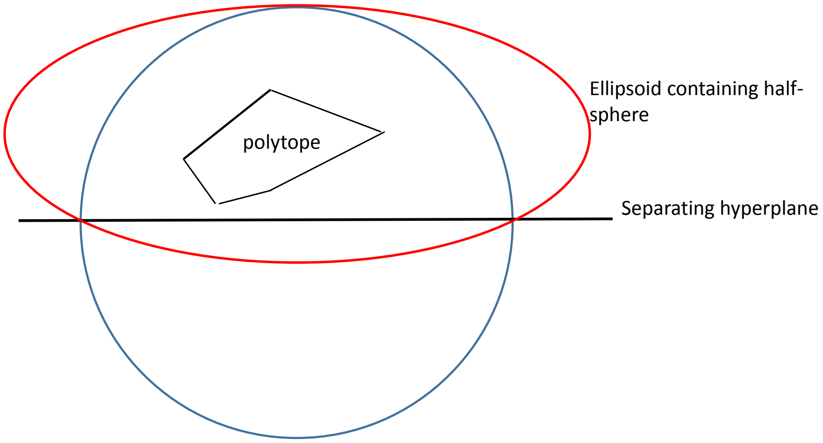


Figure 10.8: Ellipsoid Algorithm

Lemma 10.11 *Consider the half-sphere $B' = \{\mathbf{x} : x_1 \geq 0\ ;\ |\mathbf{x}| \leq 1\}$. The following ellipsoid $\hat{E}$ contains B':*

$$\hat{E} = \left\{ \mathbf{x} \middle| \left(\frac{d+1}{d}\right)^2 \left(x_1 - \frac{1}{d+1}\right)^2 + \left(\frac{d^2-1}{d^2}\right)\left(x_2^2 + x_3^2 + \ldots + x_d^2\right) \leq 1 \right\}.$$

Further,

$$\frac{Vol(\hat{E})}{Vol(B)} = \left(\frac{d}{d+1}\right)\left(\frac{d^2}{d^2-1}\right)^{(d-1)/2} \leq 1 - \frac{1}{4d}.$$

Proof: See Exercise (10.3).

10.7 Integer Optimization

The problem of maximizing a linear function subject to linear inequality constraints, but with the variables constrained to be integers is called integer programming:

$$\text{Max } \mathbf{c} \cdot \mathbf{x} \ \text{ subject to } A\mathbf{x} \leq \mathbf{b} \ \ x_i \text{ integers .}$$

This problem is NP-hard. One way to handle the hardness is to relax the integer constraints, solve the linear program in polynomial time and round the fractional values to integers. The simplest rounding, round each variable which is 1/2 or more to 1, the rest to 0, yields sensible results in some cases. The vertex cover problem is one of them. The problem is to choose a subset of vertices so that each edge is covered, at least one of its end points is in the subset. The integer program is:

$$\text{Min} \sum_i x_i \text{ subject to } x_i + x_j \geq 1 \; \forall \text{ edges } (i,j) \;\; ; x_i \text{ integers} .$$

Solve the Linear program. At least one variable for each edge must be at least 1/2 and the simple rounding rounds it up to 1. So the integer solution we get is still feasible. It clearly at most doubles the objective function from the Linear Programming solution and since the LP solution value is at most the optimal integer programming solution value, we get within a factor of 2 of the optimal.

10.8 Semi-Definite Programming

Semi-definite programs are special cases of convex programs. Recall that a $n \times n$ matrix A is positive semi-definite if and only if (i) A is symmetric and (ii) for all $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x}^T A\mathbf{x} \geq 0$. There are many equivalent characterizations of positive semi-definite matrices. We mention one. A symmetric matrix A is positive semi-definite if and only if it can be expressed as $A = BB^T$ for a possibly rectangular matrix B.

An semi-definite program (SDP) is the problem of minimizing a linear function $\mathbf{c}^T\mathbf{x}$ subject to a constraint that $F = F_0 + F_1x_1 + F_2x_2 + \cdots + F_dx_d$ is positive semi-definite. Here $F_0, F_1, \ldots, F_d$ are given symmetric matrices.

This is a convex program since the set of $\mathbf{x}$ satisfying the constraint is a convex set: To see this, just note that if $F(\mathbf{x}) = F_0 + F_1x_1 + F_2x_2 + \cdots + F_dx_d$ and $F(\mathbf{y}) = F_0 + F_1y_1 + F_2y_2 + \cdots + F_dy_d$ are positive semi-definite, then so is $F(\alpha\mathbf{x} + (1-\alpha)\mathbf{y})$ for all $\alpha \in [0,1]$. So SDP's can be solved in polynomial time in principle. It turns out that there are more efficient algorithms for SDP's than general convex programs and also that many interesting problems can be formulated as SDP's. We discuss the latter aspect here.

First note that linear programs are special cases of SDP's. For any vector $\mathbf{v}$, let diag($\mathbf{v}$) denote a diagonal matrix with the components of $\mathbf{v}$ on the diagonal. Then it is easy to see that the constraints $\mathbf{v} \geq \mathbf{0}$ are equivalent to the constraint diag($\mathbf{v}$) is positive semi-definite. Consider now the linear rogram:

Minimize $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$. Rewrite $A\mathbf{x} = \mathbf{b}$ as $A\mathbf{x} - \mathbf{b} \geq \mathbf{0}$; $\mathbf{b} - A\mathbf{x} \geq \mathbf{0}$ and use the idea of diagonal matrices above to formulate this as an SDP.

A second interesting example is that of quadratic programs of the form:
Minimize $\frac{(\mathbf{c}^T\mathbf{x})^2}{\mathbf{d}^T\mathbf{x}}$ subject to $A\mathbf{x} + \mathbf{b} \geq \mathbf{0}$.

First, this is equivalent to : Minimize t subject to $A\mathbf{x} + \mathbf{b} \geq \mathbf{0}$ and $t \geq \frac{(\mathbf{c}^T\mathbf{x})^2}{\mathbf{d}^T\mathbf{x}}$. Now this is in turn equivalent to the SDP:

Minimize t subject to the following matrix being positive semi-definite:

$$\begin{pmatrix} \mathrm{diag}(A\mathbf{x}+\mathbf{b}) & 0 & 0 \\ 0 & t & \mathbf{c}^T\mathbf{x} \\ 0 & \mathbf{c}^T\mathbf{x} & \mathbf{d}^T\mathbf{x} \end{pmatrix}.$$

An exciting area of application of SDP is to solve some integer problems. The central idea is best illustrated by its early application in a breakthrough due to Goemans and Williamson ([**?**]) for the maximum cut problem which given a graph $G(V, E)$ asks for the cut $S, \bar{S}$ maximizing the number of edges going across the cut from S to $\bar{S}$. For each $i \in V$, let x_i be an integer variable assuming values ± 1 depending on whether $i \in S$ or $i \in \bar{S}$ respectively. Then the max-cut problem can be posed as

$$\text{Maximize } \textstyle\sum_{(i,j)\in E}(1 - x_i x_j) \text{ subject to the constraints } x_i \in \{-1, +1\}.$$

The integrality constraint on the x_i makes the problem NP-hard. Instead replace the integer constraints by allowing the $\mathbf{x_i}$ to be unit length vectors. This enlarges the set of feasible solutions since ± 1 are just 1-dimensional vectors of length 1. The relaxed problem is an SDP and can be solved in polynomial time. To see that it is an SDP, consider $\mathbf{x_i}$ as the rows of a matrix X. The variables of our SDP are not X, but actually $Y = XX^T$, which is a positive definite matrix. The SDP is

$$\text{Maximize } \textstyle\sum_{(i,j)\in E}(1 - y_{ij}) \text{ subject to } Y \text{ positive semi definte.}$$

This can be solved in polynomial time. From the solution Y, find X satisfying $Y = XX^T$. Now, instead of a ± 1 label on each vertex, we have vector labels, namely the rows of X. We need to round the vectors to ± 1 to get an S. One natural way to do this is to pick a random vector $\mathbf{v}$ and if for vertex i, $\mathbf{x_i} \cdot \mathbf{v}$ is positive, put i in S, otherwise put it in $\bar{S}$. Goemans and Wiiliamson showed that this method produces a cut guaranteed to be at least 0.878 times the maximum. The .878 factor is a big improvement on the previous best factor of 0.5 which is easy to get by putting each vertex into S with probability 1/2.

Exercise 10.1

1. *Suppose for a univariate convex function f and a finite interval D, $|f''(x)| \leq \delta |f'(x)|$ for every x. Then, what is a good step size to choose for gradient descent? Derive a bound on the number of steps needed to get an approximate minimum of f in terms of as few parameters as possible.*

2. *Generalize the statement and proof to convex functions of d variables.*

Exercise 10.2 *Prove that the maximum of a convex function over a polytope is attained at one of its vertices.*

Exercise 10.3 *Prove Lemma 10.11.*

10.9 Exercises

Exercise 10.4 *Select a method that you believe is good for combining individual rankings into a global ranking. Consider a set of rankings where each individual ranks b last. One by one move b from the bottom to the top leaving the other rankings in place. Does there exist a v_b as in Theorem 10.1 where v_b is the ranking that causes b to move from the bottom to the top in the global ranking. If not, does your method of combing individual rankings satisfy the axioms of unanimity and independence of irrelevant alternatives.*

Exercise 10.5 *Show that the three axioms: non dictator, unanimity, and independence of irrelevant alternatives are independent.*

Exercise 10.6 *Does the axiom of independence of irrelevant alternatives make sense? What if there were three rankings of five items. In the first two rankings, A is number one and B is number two. In the third ranking, B is number one and A is number five. One might compute an average score where a low score is good. A gets a score of 1+1+5=7 and B gets a score of 2+2+1=5 and B is ranked number one in the global raking. Now if the third ranker moves A up to the second position, A's score becomes 1+1+2=4 and the global ranking of A and B changes even though no individual ranking of A and B changed. Is there some alternative axiom to replace independence of irrelevant alternatives? Write a paragraph on your thoughts on this issue.*

Exercise 10.7 *Prove that the global ranking agrees with column v_b even if b is moved down through the column.*

Exercise 10.8 *Create a random 100 by 100 orthonormal matrix A and a sparse 100-dimensional vector* $\mathbf{x}$*. Compute* $A\mathbf{x} = \mathbf{b}$*. Randomly select a few coordinates of* $\mathbf{b}$ *and reconstruct* $\mathbf{x}$ *from the samples of* $\mathbf{b}$ *using the minimization of 1-norm technique of Section 10.3.1. Did you get* $\mathbf{x}$ *back?*

Exercise 10.9 *Let A be a low rank $n \times m$ matrix. Let r be the rank of A. Let $\tilde{A}$ be A corrupted by Gaussian noise. Prove that the rank r SVD approximation to $\tilde{A}$ minimizes* $\left|A - \tilde{A}\right|_F^2$.

Exercise 10.10 *Prove that minimizing $||x||_0$ subject to $Ax = b$ is NP-complete.*

Exercise 10.11 *Let A be a Gaussian matrix where each element is a random Gauussian variable with zero mean and variance one. Prove that A has the restricted isometry property.*

Exercise 10.12 *Generate* 100×100 *matrices of rank 20, 40, 60 80, and 100. In each matrix randomly delete 50, 100, 200, or 400 entries. In each case try to recover the original matrix. How well do you do?*

Exercise 10.13 *Repeat the previous exercise but instead of deleting elements, corrupt the elements by adding a reasonable size corruption to the randomly selected matrix entires.*

Exercise 10.14 *Compute the Fourier transform of the sequence 1000010000.*

Exercise 10.15 *What is the Fourier transform of a cyclic shift?*

Exercise 10.16 *Let $S(i, j)$ be the sequence of i blocks each of length j where each block of symbols is a 1 followed by $i-1$ 0's. The number n=6 is factorable but not a perfect square. What is Fourier transform of $S\,(2, 3)$= 100100?*

Exercise 10.17 *Let Z be the n root of unity. Prove that $\left\{z^{bi}|0 \leq i < n\right\} = \{z^i|0 \leq i < n\}$ provide that b does not divide n.*

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ a & b & \cdots & c \\ a^2 & b^2 & \cdots & c^2 \\ \vdots & \vdots & & \vdots \\ a^d & b^d & \cdots & c^d \end{pmatrix}$$

Show that if the elements in the second row of the Vandermonde matrix are distinct, then the Vandermonde matrix is nonsingular by using the fact that specifying the value of an n^{th} degree polynomial at $n+1$ points uniquely determines the polynomial.

Exercise 10.18 *Many problems can be formulated as finding $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b} + \mathbf{r}$ where $\mathbf{r}$ is some residual error. Discuss the advantages and disadvantages of each of the following three versions of the problem.*

1. *Set $\mathbf{r}$=0 and find $\mathbf{x}$= argmin $\|\mathbf{x}\|_1$ satisfying $A\mathbf{x} = \mathbf{b}$*
2. *Lasso: find $\mathbf{x}$= argmin $\left(\|\mathbf{x}\|_1 + \alpha \|\mathbf{r}\|_2^2\right)$ satisfying $A\mathbf{x} = \mathbf{b}$*
3. *find $\underline{x}$=argmin $\|\mathbf{x}\|_1$ such that $\|\mathbf{r}\|_2 < \varepsilon$*

Exercise 10.19 *Create a graph of overlapping communities as follows. Let n=1,000. Partition the integers into ten blocks each of size 100. The first block is $\{1, 2, \ldots, 100\}$. The second is $\{100, 101, \ldots, 200\}$, and so on. Add edges to the graph so that the vertices in each block form a clique. Now randomly permute the indices and partition the sequence into ten blocks of 100 vertices each. Again add edges so that these new blocks are cliques. Randomly permute the indices a second time and repeat the process of adding edges. The result is a graph in which each vertex is in three cliques. Explain how to find the cliques given the graph.*

Exercise 10.20 *Repeat the above exercise but instead of adding edges to form cliques, use each block to form a G(100,p) graph. For how small a p can you recover the blocks? What if you add G(1,000,q) to the graph for some small value of q.*

Exercise 10.21 *Construct an $n \times m$ matrix A where each of the m columns is a 0-1 indicator vector with approximately 1/4 entries being 1. Then $B = AA^T$ is a symmetric matrix that can be viewed as the adjacency matrix of an n vertex graph. Some edges will have weight greater than one. The graph consists of a number of possibly over lapping cliques. Your task given B is to find the cliques by the following technique of finding a 0-1 vector in the column space of B by the following linear program for finding b and x.*

$$b = argmin||b||_1$$

subject to

$$\begin{aligned} &Bx = b \\ &b_1 = 1 \\ &0 \leq b_i \leq 1 \quad 2 \leq i \leq n \end{aligned}$$

Then subtract bb^T from B and repeat.

Exercise 10.22 *Construct an example of a matrix A satisfying the following conditions*

1. *The columns of A are 0-1 vectors where the support of no two columns overlap by 50% or more.*
2. *No column's support is totally within the support of another column.*
3. *The minimum 1-norm vector in the column space of A is not a 0-1 vector.*

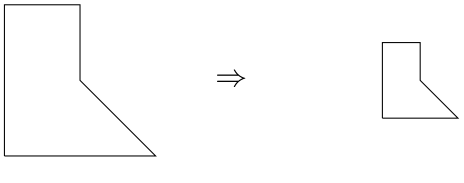
Exercise 10.23 *Let $M = L+R$ where L is a low rank matrix corrupted by a sparse noise matrix R. Why can we not recover L from M if R is low rank or if L is sparse?*

11 Wavelets

Given a vector space of functions, one would like an orthonormal set of basis functions that span the space. The Fourier transform provides a set of basis functions based on sines and cosines. Often we are dealing with functions that have finite support in which case we would like the basis vectors to have finite support. Also we would like to have an efficient algorithm for computing the coefficients of the expansion of a function in the basis.

11.1 Dilation

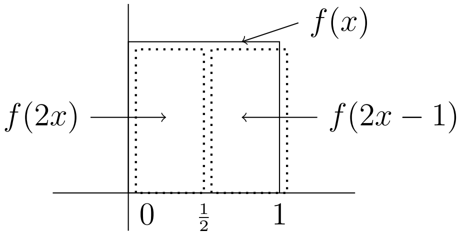
We begin our development of wavelets by first introducing dilation. A *dilation* is a mapping that scales all distances by the same factor.



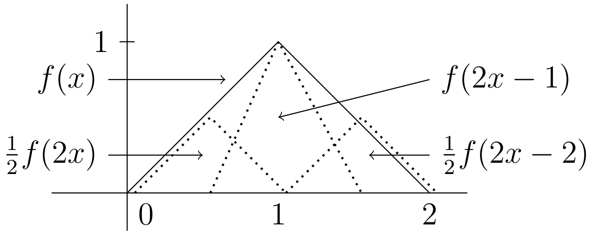
A dilation equation is an equation where a function is defined in terms of a linear combination of scaled, shifted versions of itself. For example,

$$f(x) = \sum_{k=0}^{d-1} c_k f(2x-k).$$

An example is $f(x) = f(2x) + f(2x-1)$ which has a solution $f(x)$ equal one for $0 \le x < 1$ and is zero elsewhere. The equation is illustrated in the figure below. The solid rectangle is $f(x)$ and the dotted rectangles are $f(2x)$ and $f(2x-1)$.



Another example is $f(x) = \frac{1}{2}f(2x) + f(2x-1) + \frac{1}{2}f(2x-2)$. A solution is illustrated in the figure below. The function $f(x)$ is indicated by solid lines. The functions $\frac{1}{2}f(2x)$, $f(2x+1)$, and $\frac{1}{2}f(2x-2)$ are indicated by dotted lines.



Lemma 11.1 *If a dilation equation in which all the dilations are a factor of two reduction has a solution, then either the coefficients on the right hand side of the equation sum to two or the integral $\int_{-\infty}^{\infty} f(x)dx$ of the solution is zero.*

Proof: Integrate both sides of the dilation equation from $-\infty$ to $+\infty$.

$$\begin{aligned} \int_{-\infty}^{\infty} f(x)dx &= \int_{-\infty}^{\infty} \sum_{k=0}^{d-1} c_k f(2x-k)dx = \sum_{k=0}^{d-1} c_k \int_{-\infty}^{\infty} f(2x-k)dx \\ &= \sum_{k=0}^{d-1} c_k \int_{-\infty}^{\infty} f(2x)dx = \frac{1}{2}\sum_{k=0}^{d-1} c_k \int_{-\infty}^{\infty} f(x)dx \end{aligned}$$

If $\int_{-\infty}^{\infty} f(x)dx \neq 0$, then dividing both sides by $\int\limits_{-\infty}^{\infty} f(x)dx$ gives $\sum\limits_{k=0}^{d-1} c_k = 2$ ∎

The above proof interchanged the order of the summation and the integral. This is valid provided the 1-norm of the function is finite. Also note that there are nonzero solutions to dilation equations in which all dilations are a factor of two reduction where the coefficients do not sum to two such as

$$f(x) = f(2x) + f(2x-1) + f(2x-2) + f(2x-3)$$

or

$$f(x) = f(2x) + 2f(2x-1) + 2f(2x-2) + 2f(2x-3) + f(2x-4).$$

In these examples $f(x)$ takes on both positive and negative values and $\int_{-\infty}^{\infty} f(x)dx = 0$.

11.2 The Haar wavelet

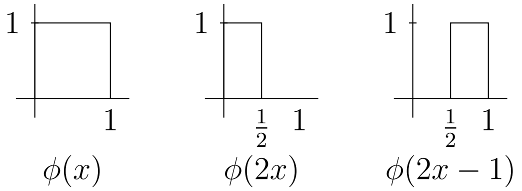
Let $\phi(x)$ be a solution to the dilation equation $f(x) = f(2x)+f(2x-1)$. The function ϕ is called a *scale function* or *scale vector* and is used to generate the two dimensional family of functions, $\phi_{jk} = \phi(2^j x - k)$. Other authors scale $\phi_{jk} = \phi(2^j x - k)$ by $2^{\frac{j}{2}}$ so that the 2-norm, $\int_{-\infty}^{\infty} \phi_{jk}^2(t)dt$, is 1. However, for educational purposes, simplifying the notation for ease of understanding was preferred.

For a given value of j, the shifted versions, $\{\phi_{jk}|k \geq 0\}$, span a space V_j. The spaces $V_0, V_1, V_2, \ldots$ are larger and larger spaces and allow better and better approximations to a function. The fact that $\phi(x)$ is the solution of a dilation equation implies that for for fixed j ϕ_{jk} is a linear combination of the $\{\phi_{j+1,k}|k \geq 0\}$ and this ensures that $V_j \subseteq V_{j+1}$. It is for this reason that it is desirable in designing a wavelet system for the scale function to satisfy a dilation equation. For a given value of j, the shifted ϕ_{jk} are orthogonal in the sense that $\int_x \phi_{jk}(x)\phi_{jl}(x)dx = 0$ for $k \neq l$.

Note that for each j, the set of functions ϕ_{jk}, $k = 0, 1, 2\ldots$, form a basis for a vector space V_j and are orthogonal. The set of basis vectors ϕ_{jk}, for all j and k, form an over complete basis and for different values of j are not orthogonal. Since ϕ_{jk}, $\phi_{j+1,2k}$, and $\phi_{j+1,2k+1}$ are linearly dependent, for each value of j delete $\phi_{j+1,k}$ for odd values of k to get a linearly independent set of basis vectors. To get an orthogonal set of basis vectors, define

$$\psi_{jk}(x) = \begin{cases} 1 & \frac{2k}{2^j} \leq x < \frac{2k+1}{2^j} \\ \\ -1 & \frac{2k+1}{2^j} \leq x < \frac{2k+2}{2^j} \\ \\ 0 & \text{otherwise} \end{cases}$$

and replace $\phi_{j,2k}$ with $\psi_{j+1,2k}$. Basically, replace the three functions



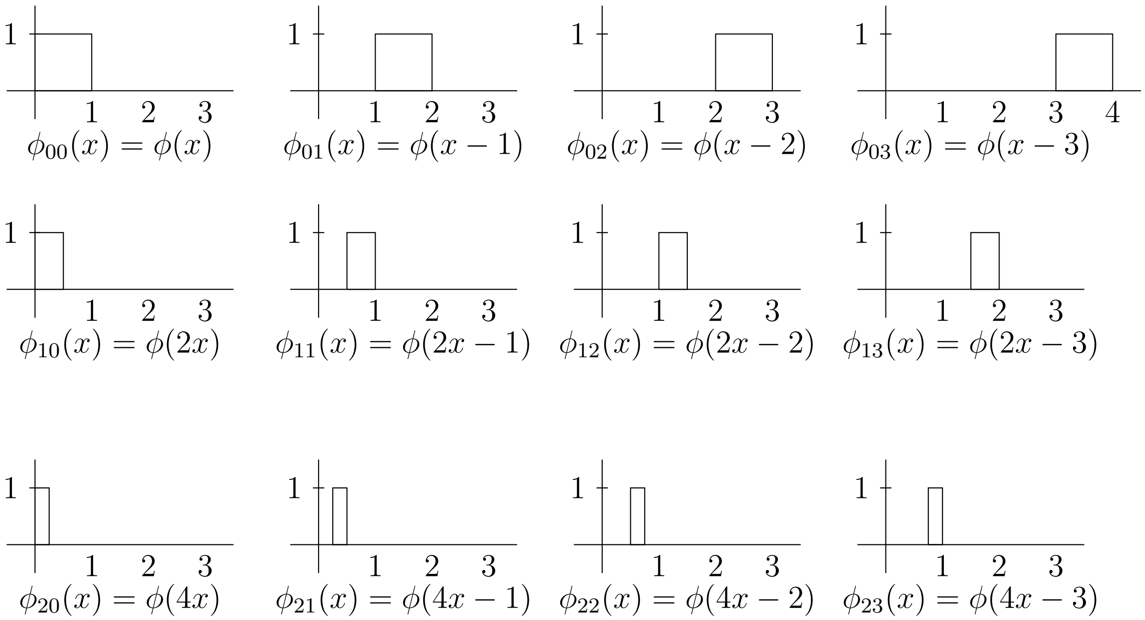
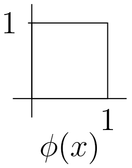
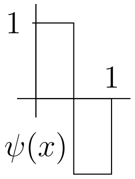


Figure 11.1: Set of scale functions associated with the Haar wavelet.

by the two functions





The basis set becomes

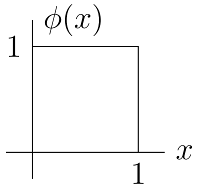
$$\begin{array}{llllllll}
\phi_{00} & \psi_{10} \\
\psi_{20} & \psi_{22} \\
\psi_{30} & \psi_{32} & \psi_{34} & \psi_{36} \\
\psi_{40} & \psi_{42} & \psi_{44} & \psi_{46} & \psi_{48} & \psi_{4,10} & \psi_{4,12} & \psi_{4,14}
\end{array}$$

To find a basis for a function that has only finite support, select a scale vector $\phi(x)$ whose scale is that of the support of the function to be represented. Next approximate the function by the set of scale functions $\phi(2^j x - k)$, $k = 0, 1, \ldots$, for some fixed value of j. The value of j is determined by the desired accuracy of the approximation. Basically the x axis has been divided into intervals of size 2^{-j} and in each interval the function is approximated by a fixed value. It is this approximation of the function that is expressed as a linear combination of the basis functions.

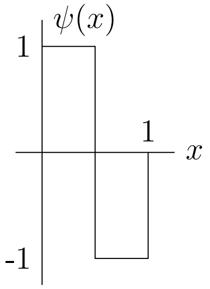
Once the value of j has been selected, the function is sampled at 2^j points, one in each interval of width 2^{-j}. Let the sample values be $s_0, s_1, \ldots$. The approximation to the

The Haar Wavelet

$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ \\ 0 & \text{otherwise} \end{cases}$$



$$\psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ \\ -1 & \frac{1}{2} \le x < 1 \\ \\ 0 & \text{otherwise} \end{cases}$$



function is $\sum_{k=0}^{2^j-1} s_k \phi(2^j x - k)$ and is represented by the vector $(s_0, s_1 \ldots, s_{2^j-1})$. The problem now is to represent the approximation to the function using the basis vectors rather than the non orthogonal set of scale functions $\phi_{jk}(x)$. This is illustrated in the following example.

To represent the function corresponding to a vector such as $\begin{pmatrix} 3 & 1 & 4 & 8 & 3 & 5 & 7 & 9 \end{pmatrix}$, one needs to find the c_i such that

$$\begin{pmatrix} 3 \\ 1 \\ 4 \\ 8 \\ 3 \\ 5 \\ 7 \\ 9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix}.$$

The first column represents the scale function $\phi(x)$ and subsequent columns the ψ's. The tree in Figure 11.2 illustrates an efficient way to find the coefficients representing the vector $\begin{pmatrix} 3 & 1 & 4 & 8 & 3 & 5 & 7 & 9 \end{pmatrix}$ in the basis. Each vertex in the tree contains the average of the quantities of its two children. The root gives the average of the elements in the vector, which is 5 in this example. This average is the coefficient of the basis vector in the first column of the above matrix. The second basis vector converts the average of the eight elements into the average of the first four elements, which is 4, and the last four elements, which is 6, with a coefficient of -1. Working up the tree determines the coefficients for each basis vector.

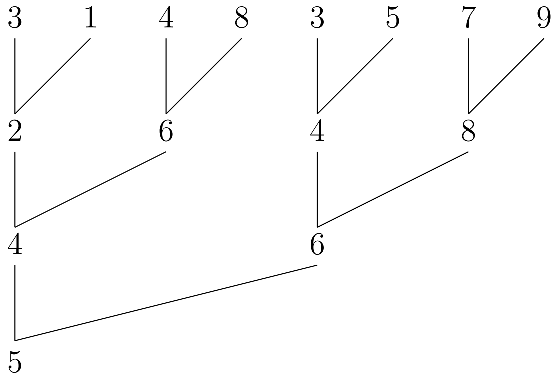


Figure 11.2: Tree of function averages

$$\begin{pmatrix} 3\\1\\4\\8\\3\\5\\7\\9 \end{pmatrix} = 5\begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix} -1\begin{pmatrix} 1\\1\\1\\1\\-1\\-1\\-1\\-1 \end{pmatrix} -2\begin{pmatrix} 1\\1\\-1\\-1\\0\\0\\0\\0 \end{pmatrix} -2\begin{pmatrix} 0\\0\\0\\0\\1\\1\\-1\\-1 \end{pmatrix} +1\begin{pmatrix} 1\\-1\\0\\0\\0\\0\\0\\0 \end{pmatrix} -2\begin{pmatrix} 0\\0\\1\\-1\\0\\0\\0\\0 \end{pmatrix} -1\begin{pmatrix} 0\\0\\0\\0\\1\\-1\\0\\0 \end{pmatrix} -1\begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\-1 \end{pmatrix}$$

11.3 Wavelet systems

So far we have explained wavelets using the simple to understand Haar wavelet. We now consider general wavelet systems. A wavelet system is built from a basic scaling function $\phi(x)$, which comes from a dilation equation. Scaling and shifting of the basic scaling function gives a two dimensional set of scaling functions ϕ_{jk} where

$$\phi_{jk}(x) = \phi(2^j x - k).$$

For a fixed value of j, the ϕ_{jk} span a space V_j. If $\phi(x)$ satisfies a dilation equation

$$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x - k),$$

then ϕ_{jk} is a linear combination of the $\phi_{j+1,k}$'s and this implies that $V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \cdots$.

11.4 Solving the dilation equation

Consider solving a dilation equation

$$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x - k)$$

to obtain the scale function for a wavelet system. Perhaps the easiest way is to assume a solution and then calculate the scale function by successive approximation as in the following program for

$$\phi(x) = \tfrac{1+\sqrt{3}}{4}\phi(2x) + \tfrac{3+\sqrt{3}}{4}\phi(2x-1) + \tfrac{3-\sqrt{3}}{4}\phi(2x-2) + \tfrac{1-\sqrt{3}}{4}\phi(2x-3),$$

a Daubechies scale function. The solution will actually be samples of $\phi(x)$ at some desired resolution.

```
Set the initial approximation to $\phi(x)$ by generating a vector whose components approximate the samples of $\phi(x)$ at equally spaced values of $x$.
Calculate the coefficients of the dilation equation.
    $c_1 = \frac{1+\sqrt{3}}{4}$    $c_2 = \frac{3+\sqrt{3}}{4}$    $c_3 = \frac{3-\sqrt{3}}{4}$    $c_4 = \frac{1-\sqrt{3}}{4}$
Execute the following loop until the values for $\phi(x)$ converge.
begin
    Calculate $\phi(2x)$ by averaging successive values of $\phi(x)$ together. Then fill out the remaining half of the vector representing $\phi(2x)$ with zeros.
    Calculate $\phi(2x-1)$, $\phi(2x-2)$, and $\phi(2x-3)$ by shifting the contents of $\phi(2x)$ the appropriate distance, discard the zeros that move off the right end and add zeros at the left end.
    Calculate the new approximation for $\phi(x)$ using the above values for $\phi(2x-1)$, $\phi(2x-2)$, and $\phi(2x-3)$ in the dilation equation for $\phi(2x)$.
end
```

The convergence of the iterative procedure for computing is fast if the eigenvectors of a certain matrix are unity.

Another approach to solving the dilation equation

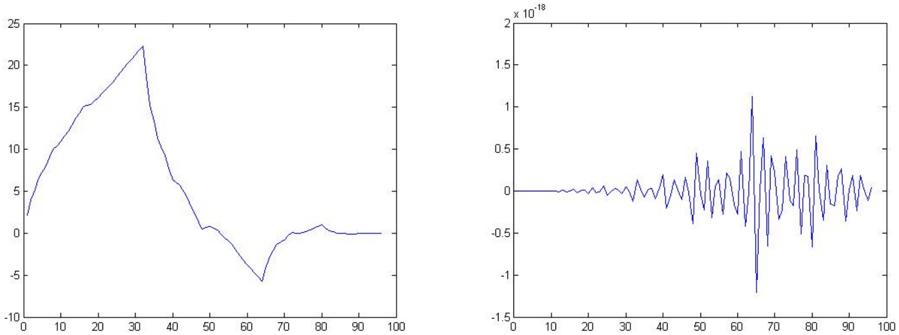


Figure 11.3: Daubechies scale function and associated wavelet

Consider the dilation equation $\phi(x) = \frac{1}{2}f(2x) + f(2x-1) + \frac{1}{2}f(2x-2)$ and consider continuous solutions with support in $0 \le x < 2$.

$$\begin{array}{ll}
\phi(0) = \frac{1}{2}\phi(0) + \phi(-1) + \phi(-2) = \frac{1}{2}\phi(0) + 0 + 0 & \phi(0) = 0 \\
\phi(2) = \frac{1}{2}\phi(4) + \phi(3) + \phi(2) = \frac{1}{2}\phi(2) + 0 + 0 & \phi(2) = 0 \\
\phi(1) = \frac{1}{2}\phi(2) + \phi(1) + \phi(0) = 0 + \phi(1) + 0 & \phi(1) \quad \text{arbitrary}
\end{array}$$

Set $\phi(1) = 1$. Then

$$\phi(\tfrac{1}{2}) = \tfrac{1}{2}\phi(1) + \phi(0) + \tfrac{1}{2}\phi(-1) = \tfrac{1}{2}$$

$$\phi(\tfrac{3}{2}) = \tfrac{1}{2}\phi(3) + \phi(2) + \tfrac{1}{2}\phi(1) = \tfrac{1}{2}$$

$$\phi(\tfrac{1}{4}) = \tfrac{1}{2}\phi(\tfrac{1}{2}) + \phi(-\tfrac{1}{2}) + \tfrac{1}{2}\phi(-\tfrac{3}{2}) = \tfrac{1}{4}$$

Etc.

11.5 Conditions on the dilation equation

We would like a basis for a vector space of functions where each basis vector has finite support and the basis vectors are orthogonal. This is achieved by a wavelet system consisting of a shifted version of a scale function that satisfies a dilation equation along with a set of wavelets of various scales and shifts. For the scale function to have a nonzero integral, Lemma 11.1 requires that the coefficients of the dilation equation sum to two. Although the scale function $\phi(x)$ for the Haar system has the property that $\phi(x)$ and $\phi(x-k)$, $k > 0$, are orthogonal, this is not true for the scale function for the dilation equation $\phi(x) = \frac{1}{2}\phi(2x) + \phi(2x-1) + \frac{1}{2}\phi(2x-2)$. The conditions that integer shifts of the scale function be orthogonal and that the scale function has finite support puts additional conditions on the coefficients of the dilation equation. These conditions are developed in the next two lemmas.

Lemma 11.2 *Let*

$$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x-k).$$

If $\phi(x)$ and $\phi(x-k)$ are orthogonal for $k \neq 0$ and $\phi(x)$ has been normalized so that $\int_{-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$, then $\sum_{i=0}^{d-1} c_i c_{i-2k} = 2\delta(k)$.

Proof: Assume $\phi(x)$ has been normalized so that $\int_{-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$. Then

$$\begin{aligned}\int_{x=-\infty}^{\infty} \phi(x)\phi(x-k)dx &= \int_{x=-\infty}^{\infty} \sum_{i=0}^{d-1} c_i\phi(2x-i) \sum_{j=0}^{d-1} c_j\phi(2x-2k-j)dx \\ &= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} c_i c_j \int_{x=-\infty}^{\infty} \phi(2x-i)\phi(2x-2k-j)dx\end{aligned}$$

Since

$$\begin{aligned}\int_{x=-\infty}^{\infty} \phi(2x-i)\phi(2x-2k-j)dx &= \frac{1}{2}\int_{x=-\infty}^{\infty} \phi(y-i)\phi(y-2k-j)dy \\ &= \frac{1}{2}\int_{x=-\infty}^{\infty} \phi(y)\phi(y+i-2k-j)dy \\ &= \frac{1}{2}\delta(2k+j-i),\end{aligned}$$

$\displaystyle\int_{x=-\infty}^{\infty} \phi(x)\phi(x-k)dx = \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} c_i c_j \frac{1}{2}\delta(2k+j-i) = \frac{1}{2}\sum_{i=0}^{d-1} c_i c_{i-2k}$. Since $\phi(x)$ was normalized so that
$\displaystyle\int_{-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$, it follows that $\displaystyle\sum_{i=0}^{d-1} c_i c_{i-2k} = 2\delta(k)$. ■

Lemma 11.2 provides a necessary but not sufficient condition on the coefficients of the dilation equation for shifts of the scale function to be orthogonal. One should note that the conditions of Lemma 11.2 are not true for the triangular or piecewise quadratic solutions to

$$\phi(x) = \frac{1}{2}\phi(2x) + \phi(2x-1) + \frac{1}{2}\phi(2x-2)$$

and

$$\phi(x) = \frac{1}{4}\phi(2x) + \frac{3}{4}\phi(2x-1) + \frac{3}{4}\phi(2x-2) + \frac{1}{4}\phi(2x-3)$$

which overlap and are not orthogonal.

For $\phi(x)$ to have finite support the dilation equation can have only a finite number of terms. This is proved in the following lemma.

Lemma 11.3 *If $0 \leq x < d$ is the support of $\phi(x)$, and the set of integer shifts, $\{\phi(x-k) | k \geq 0\}$, are linearly independent, then $c_k = 0$ unless $0 \leq k \leq d-1$.*

Scale and wavelet coefficients equations

$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x-k)$

$$\int\limits_{-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$$

$$\sum_{j=0}^{d-1} c_j = 2$$

$$\sum_{j=0}^{d-1} c_j c_{j-2k} = 2\delta(k)$$

$c_k = 0$ unless $0 \leq k \leq d-1$

d even

$$\sum_{j=0}^{d-1} c_{2j} = \sum_{j=0}^{d-1} c_{2j+1}$$

$$\psi(x) = \sum_{k=0}^{d-1} b_k \phi(x-k)$$

$$\int\limits_{x=-\infty}^{\infty} \phi(x)\psi(x-k) = 0$$

$$\int\limits_{x=-\infty}^{\infty} \psi(x)dx = 0$$

$$\int\limits_{x=-\infty}^{\infty} \psi(x)\psi(x-k)dx = \delta(k)$$

$$\sum_{i=0}^{d-1} (-1)^k b_i b_{i-2k} = 2\delta(k)$$

$$\sum_{j=0}^{d-1} c_j b_{j-2k} = 0$$

$$\sum_{j=0}^{d-1} b_j = 0$$

$b_k = (-1)^k c_{d-1-k}$

One designs wavelet systems so the above conditions are satisfied.

Proof: If the support of $\phi(x)$ is $0 \leq x < d$, then the support of $\phi(2x)$ is $0 \leq x < \frac{d}{2}$. If

$$\phi(x) = \sum_{k=-\infty}^{\infty} c_k \phi(2x-k)$$

the support of both sides of the equation must be the same. Since the $\phi(x-k)$ are linearly independent the limits of the summation are actually $k=0$ to $d-1$ and

$$\phi(x) = \sum_{k=0}^{d-1} c_k \phi(2x-k).$$

It follows that $c_k = 0$ unless $0 \leq k \leq d-1$.

The condition that the integer shifts are linearly independent is essential to the proof and the lemma is not true without this condition. ■

One should also note that $\sum\limits_{i=0}^{d-1} c_i c_{i-2k} = 0$ for $k \neq 0$ implies that d is even since for d odd and $k = \frac{d-1}{2}$

$$\sum_{i=0}^{d-1} c_i c_{i-2k} = \sum_{i=0}^{d-1} c_i c_{i-d+1} = c_{d-1}c_0.$$

For $c_{d-1}c_0$ to be zero either c_{d-1} or c_0 must be zero. Since either $c_0 = 0$ or $c_{d-1} = 0$, there are only $d-1$ nonzero coefficients. From here on we assume that d is even. If the dilation equation has d terms and the coefficients satisfy the linear equation $\sum_{k=0}^{d-1} c_k = 2$ and the $\frac{d}{2}$ quadratic equations $\sum_{i=0}^{d-1} c_i c_{i-2k} = 2\delta(k)$ for $1 \leq k \leq \frac{d-1}{2}$, then for $d > 2$ there are $\frac{d}{2} - 1$ coefficients that can be used to design the wavelet system to achieve desired properties.

11.6 Derivation of the wavelets from the scaling function

In a wavelet system one develops a mother wavelet as a linear combination of integer shifts of a scaled version of the scale function $\phi(x)$. Let the mother wavelet $\psi(x)$ be given by $\psi(x) = \sum\limits_{k=0}^{d-1} b_k \phi(2x-k)$. One wants integer shifts of the mother wavelet $\psi(x-k)$ to be orthogonal and also for integer shifts of the mother wavelet to be orthogonal to the scaling function $\phi(x)$. These conditions place restrictions on the coefficients b_k which are the subject matter of the next two lemmas.

Lemma 11.4 *(Orthogonality of $\psi(x)$ and $\psi(x-k)$) Let $\psi(x) = \sum\limits_{k=0}^{d-1} b_k \phi(2x-k)$. If $\psi(x)$ and $\psi(x-k)$ are orthogonal for $k \neq 0$ and $\psi(x)$ has been normalized so that $\int_{-\infty}^{\infty} \psi(x)\psi(x-k)dx = \delta(k)$, then*

$$\sum_{i=0}^{d-1} (-1)^k b_i b_{i-2k} = 2\delta(k).$$

Proof: Analogous to Lemma 11.2. ∎

Lemma 11.5 *(Orthogonality of $\phi(x)$ and $\psi(x-k)$) Let $\phi(x) = \sum\limits_{k=0}^{d-1} c_k \phi(2x-k)$ and $\psi(x) = \sum\limits_{k=0}^{d-1} b_k \phi(2x-k)$. If $\int\limits_{x=-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$ and $\int\limits_{x=-\infty}^{\infty} \phi(x)\psi(x-k)dx = 0$ for all k, then $\sum\limits_{i=0}^{d-1} c_i b_{i-2k} = 0$ for all k.*

Proof:

$$\int_{x=-\infty}^{\infty} \phi(x)\psi(x-k)dx = \int_{x=-\infty}^{\infty} \sum_{i=0}^{d-1} c_i \phi(2x-i) \sum_{j=1}^{d-1} b_j \phi(2x-2k-j)dx = 0.$$

Interchanging the order of integration and summation

$$\sum_{i=0}^{d-1} \sum_{j=0}^{d-1} c_i b_j \int_{x=-\infty}^{\infty} \phi(2x-i)\phi(2x-2k-j)dx = 0$$

Substituting $y = 2x - i$ yields

$$\frac{1}{2}\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} c_i b_j \int_{y=-\infty}^{\infty} \phi(y)\phi(y-2k-j+i)dy = 0$$

Thus,

$$\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} c_i b_j \delta(2k+j-i) = 0$$

Summing over j gives

$$\sum_{i=0}^{d-1} c_i b_{i-2k} = 0$$

■

Lemma 11.5 gave a condition on the coefficients in the equations for $\phi(x)$ and $\psi(x)$ if integer shifts of the mother wavelet are to be orthogonal to the scale function. In addition, for integer shifts of the mother wavelet to be orthogonal to the scale function requires that $b_k = (-1)^k c_{d-1-k}$.

Lemma 11.6 *Let the scale function* $\phi(x)$ *equal* $\sum\limits_{k=0}^{d-1} c_k\phi(2x-k)$ *and let the wavelet function* $\psi(x)$ *equal* $\sum\limits_{k=0}^{d-1} b_k\phi(2x-k)$. *If the scale functions are orthogonal*

$$\int_{-\infty}^{\infty} \phi(x)\phi(x-k)dx = \delta(k)$$

and the wavelet functions are orthogonal with the scale function

$$\int\limits_{x=-\infty}^{\infty} \phi(x)\psi(x-k)dx = 0$$

for all k, *then* $b_k = (-1)^k c_{d-1-k}$.

Proof: By Lemma 11.5, $\sum\limits_{j=0}^{d-1} c_j b_{j-2k} = 0$ for all k. Separating $\sum\limits_{j=0}^{d-1} c_j b_{j-2k} = 0$ into odd and even indices gives

$$\sum_{j=0}^{\frac{d}{2}-1} c_{2j}b_{2j-2k} + \sum_{j=0}^{\frac{d}{2}-1} c_{2j+1}b_{2j+1-2k} = 0 \tag{11.1}$$

for all k.

$$\begin{aligned}
c_0b_0 + c_2b_2 + c_4b_4 + \cdots + c_1b_1 + c_3b_3 + c_5b_5 + \cdots &= 0 \qquad k=0\\
c_2b_0 + c_4b_2 + \cdots \qquad + c_3b_1 + c_5b_3 + \cdots &= 0 \qquad k=1\\
c_4b_0 + \cdots \qquad + c_5b_1 + \cdots &= 0 \qquad k=2
\end{aligned}$$

By Lemmas 11.2 and 11.4, $\sum\limits_{j=0}^{d-1} c_j c_{j-2k} = 2\delta(k)$ and $\sum\limits_{j=0}^{d-1} b_j b_{j-2k} = 2\delta(k)$ and for all k. Separating odd and even terms,

$$\sum_{j=0}^{\frac{d}{2}-1} c_{2j}c_{2j-2k} + \sum_{j=0}^{\frac{d}{2}-1} c_{2j+1}c_{2j+1-2k} = 2\delta(k) \tag{11.2}$$

and

$$\sum_{j=0}^{\frac{d}{2}-1} b_{2j}b_{2j-2k} + \sum_{j=0}^{\frac{d}{2}-1}(-1)^j b_{2j+1}b_{2j+1-2k} = 2\delta(k) \tag{11.3}$$

for all k.

$$\begin{aligned} c_0c_0 + c_2c_2 + c_4c_4 + \cdots + c_1c_1 + c_3c_3 + c_5c_5 + \cdots &= 2 && k=0 \\ c_2c_0 + c_4c_2 + \cdots \qquad + c_3c_1 + c_5c_3 + \cdots &= 0 && k=1 \\ c_4c_0 + \cdots \qquad\qquad + c_5c_1 + \cdots &= 0 && k=2 \end{aligned}$$

$$\begin{aligned} b_0b_0 + b_2b_2 + b_4b_4 + \cdots + b_1b_1 - b_3b_3 + b_5b_5 - \cdots &= 2 && k=0 \\ b_2b_0 + b_4b_2 + \cdots \qquad - b_3b_1 + b_5b_3 - \cdots &= 0 && k=1 \\ b_4b_0 + \cdots \qquad\qquad + b_5b_1 - \cdots &= 0 && k=2 \end{aligned}$$

Let $C_e = (c_0, c_2, \ldots, c_{d-2})$, $C_o = (c_1, c_3, \ldots, c_{d-1})$, $B_e = (b_0, b_2, \ldots, b_{d-2})$, and $B_o = (b_1, b_3, \ldots, b_{d-1})$. Equations 12.1, 12.2, and 11.3 can be expressed as convolutions[22] of these sequences. Equation 12.1 is $C_e * B_e^R + C_o * B_o^R = 0$, 12.2 is $C_e * C_e^R + C_o * C_o^R = \delta(k)$, and 11.3 is $B_e * B_e^R + B_o * B_o^R = \delta(k)$, where the superscript R stands for reversal of the sequence. These equations can be written in matrix format as

$$\begin{pmatrix} C_e & C_o \\ B_e & B_o \end{pmatrix} * \begin{pmatrix} C_e^R & B_e^R \\ C_o^R & B_o^R \end{pmatrix} = \begin{pmatrix} 2\delta & 0 \\ 0 & 2\delta \end{pmatrix}$$

Taking the Fourier or z-transform yields

$$\begin{pmatrix} F(C_e) & F(C_o) \\ F(B_e) & F(B_o) \end{pmatrix} \begin{pmatrix} F(C_e^R) & F(B_e^R) \\ F(C_o^R) & F(B_o^R) \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

where F denotes the transform. Taking the determinant yields

$$\Big(F(C_e)F(B_o) - F(B_e)F(C_o)\Big)\Big(F(C_e)F(B_o) - F(C_o)F(B_e)\Big) = 4$$

Thus $F(C_e)F(B_o) - F(C_o)F(B_e) = 2$ and the inverse transform yields

$$C_e * B_o - C_o * B_e = 2\delta(k).$$

[22] The convolution of $(a_0, a_1, \ldots, a_{d-1})$ and $(b_0, b_1, \ldots, b_{d-1})$ denoted $(a_0, a_1, \ldots, a_{d-1}) * (b_0, b_1, \ldots, b_{d-1})$ is the sequence $(a_0b_{d-1}, a_0b_{d-2} + a_1b_{d-1}, a_0b_{d-3} + a_1b_{d-2} + a_3b_{d-1} \ldots, a_{d-1}b_0)$.

Convolution by C_e^R yields

$$C_e^R * C_e * B_o - C_e^R * B_e * C_o = C_e^R * 2\delta(k)$$

Now $\sum\limits_{j=0}^{d-1} c_j b_{j-2k} = 0$ so $-C_e^R * B_e = C_o^R * B_o$. Thus

$$\begin{aligned} C_e^R * C_e * B_o + C_o^R * B_o * C_o &= 2C_e^R * \delta(k) \\ (C_e^R * C_e + C_o^R * C_o) * B_o &= 2C_e^R * \delta(k) \\ 2\delta(k) * B_o &= 2C_e^R * \delta(k) \\ C_e &= B_o^R \end{aligned}$$

Thus, $c_i = 2b_{d-1-i}$ for even i. By a similar argument, convolution by C_0^R yields

$$C_0^R * C_e * B_0 - C_0^R * C_0 * B_e = 2C_0^R \delta(k)$$

Since $C_)^R * B_0 = -C_0^R * B_e$

$$\begin{aligned} -C_e^R * C_e^R * B_e - C_0^R * C_0 * B_e &= 2C_0^R \delta(k) \\ -(C_e * C_e^R + C_0^R * C_0) * B_e &= 2C_0^R \delta(k) \\ -2\delta(k) B_e &= 2C_0^R \delta(k) \\ -B_e &= C_0^R \end{aligned}$$

Thus, $c_i = -2b_{d-1-i}$ for all odd i and hence $c_i = (-1)^i 2b_{d-1-i}$ for all i. ■

11.7 Sufficient conditions for the wavelets to be orthogonal

Section 11.6 gave necessary conditions on the b_k and c_k in the definitions of the scale function and wavelets for certain orthogonality properties. In this section we show that these conditions are also sufficient for certain orthogonality conditions. One would like a wavelet system to satisfy certain conditions.

1. Wavelets, $\psi_j(2^j x - k)$, at all scales and shifts to be orthogonal to the scale function $\phi(x)$.

2. All wavelets to be orthogonal. That is

$$\int_{-\infty}^{\infty} \psi_j(2^j x - k)\psi_l(2^l x - m)dx = \delta(j - l)\delta(k - m)$$

3. $\phi(x)$ and ψ_{jk}, $j \le l$ and all k, to span V_l, the space spanned by $\phi(2^l x - k)$ for all k.

These items are proved in the following lemmas. The first lemma gives sufficient conditions on the wavelet coefficients b_k in the definition

$$\psi(x) = \sum_k b_k \psi(2x-k)$$

for the mother wavelet so that the wavelets will be orthogonal to the scale function. That is, if the wavelet coefficients equal the scale coefficients in reverse order with alternating negative signs, then the wavelets will be orthogonal to the scale function.

Lemma 11.7 *If $b_k = (-1)^k c_{d-1-k}$, then $\int_{-\infty}^{\infty} \phi(x)\psi(2^j x - l)dx = 0$ for all j and l.*

Proof: Assume that $b_k = (-1)^k c_{d-1-k}$. We first show that $\phi(x)$ and $\psi(x-k)$ are orthogonal for all values of k. Then we modify the proof to show that $\phi(x)$ and $\psi(2^j x - k)$ are orthogonal for all j and k.

Assume $b_k = (-1)^k c_{d-1-k}$. Then

$$\begin{aligned}
\int_{-\infty}^{\infty} \phi(x)\psi(x-k) &= \int_{-\infty}^{\infty} \sum_{i=0}^{d-1} c_i \phi(2x-i) \sum_{j=0}^{d-1} b_j \phi(2x-2k-j)dx \\
&= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} c_i (-1)^j c_{d-1-j} \int_{-\infty}^{\infty} \phi(2x-i)\phi(2x-2k-j)dx \\
&= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} (-1)^j c_i c_{d-1-j} \delta(i-2k-j) \\
&= \sum_{j=0}^{d-1} (-1)^j c_{2k+j} c_{d-1-j} \\
&= c_{2k}c_{d-1} - c_{2k+1}c_{d-2} + \cdots + c_{d-2}c_{2k-1} - c_{d-1}c_{2k} \\
&= 0
\end{aligned}$$

The last step requires that d be even which we have assumed for all scale functions.

For the case where the wavelet is $\psi(2^j - l)$, first express $\phi(x)$ as a linear combination of $\phi(2^{j-1}x - n)$. Now for each these terms

$$\int_{-\infty}^{\infty} \phi(2^{j-1}x - m)\psi(2^j x - k)dx = 0$$

To see this, substitute $y = 2^{j-1}x$. Then

$$\int_{-\infty}^{\infty} \phi(2^j x - m)\psi(2^j x - k)dx = \frac{1}{2^{j-1}} \int_{-\infty}^{\infty} \phi(y-m)\psi(2y-k)dy$$

which by the previous argument is zero. ■

The next lemma gives conditions on the coefficients b_k that are sufficient for the wavelets to be orthogonal.

Lemma 11.8 *If* $b_k = (-1)^k c_{d-1-k}$, *then*

$$\int_{-\infty}^{\infty} \frac{1}{2^j}\psi_j(2^j x - k)\frac{1}{2^k}\psi_l(2^l x - m)dx = \delta(j-l)\delta(k-m).$$

Proof: The first level wavelets are orthogonal.

$$\begin{aligned}
\int_{-\infty}^{\infty} \psi(x)\psi(x-k)dx &= \int_{-\infty}^{\infty} \sum_{i=0}^{d-1} b_i\phi(2x-i)\sum_{j=0}^{d-1} b_j\phi(2x-2k-j)dx \\
&= \sum_{i=0}^{d-1} b_i \sum_{j=0}^{d-1} b_j \int_{-\infty}^{\infty} \phi(2x-i)\phi(2x-2k-j)dx \\
&= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} b_i b_j \delta(i-2k-j) \\
&= \sum_{i=0}^{d-1} bib_{i-2k} \\
&= \sum_{i=0}^{d-1} (-1)^i c_{d-1-i}(-1)^{i-2k}c_{d-1-i+2k} \\
&= \sum_{i=0}^{d-1} (-1)^{2i-2k} c_{d-1-i}c_{d-1-i+2k}
\end{aligned}$$

Substituting j for $d-1-i$ yields

$$\sum_{j=0}^{d-1} c_j c_{j+2k} = 2\delta(k)$$

Example of orthogonality when wavelets are of different scale.

$$\begin{aligned}
\int_{-\infty}^{\infty} \psi(2x)\psi(x-k)dx &= \int_{-\infty}^{\infty} \sum_{i=0}^{d-1} b_i\phi(4x-i)\sum_{j=0}^{d-1} b_j\phi(2x-2k-j)dx \\
&= \sum_{i=0}^{d-1}\sum_{i=0}^{d-1} b_i b_j \int_{-\infty}^{\infty} \phi(4x-i)\phi(2x-2k-j)dx
\end{aligned}$$

Since $\phi(2x-2k-j) = \sum\limits_{l=0}^{d-1} c_l \phi(4x-4k-2j-l)$

$$\begin{aligned}
\int_{-\infty}^{\infty} \psi(2x)\psi(x-k)dx &= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1}\sum_{l=0}^{d-1} b_i b_j c_l \int_{-\infty}^{\infty} \psi(4x-i)\phi(4x-4k-2j-l)dx \\
&= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1}\sum_{l=0}^{d-1} b_i b_j c_l \delta(i-4k-2j-l) \\
&= \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} b_i b_j c_{i-4k-2j}
\end{aligned}$$

Since $\sum\limits_{j=0}^{d-1} c_j b_{j-2k} = 0$, $\sum\limits_{i=0}^{d-1} b_i c_{i-4k-2j} = \delta(j-2k)$ Thus

$$\int_{-\infty}^{\infty} \psi(2x)\psi(x-k)dx = \sum_{j=0}^{d-1} b_j \delta(j-2k) = 0.$$

Orthogonality of scale function with wavelet of different scale.

$$\begin{aligned}
\int_{-\infty}^{\infty} \phi(x)\psi(2x-k)dx &= \int_{-\infty}^{\infty} \sum_{j=0}^{d-1} c_j \phi(2x-j)\psi(2x-k)dx \\
&= \sum_{j=0}^{d-1} c_j \int_{-\infty}^{\infty} \phi(2x-j)\psi(2x-k)dx \\
&= \frac{1}{2}\sum_{j=0}^{d-1} c_j \int_{-\infty}^{\infty} \phi(y-j)\psi(y-k)dy \\
&= 0
\end{aligned}$$

If ψ was of scale 2^j, ϕ would be expanded as a linear combination of ϕ of scale 2^j all of which would be orthogonal to ψ. ■

11.8 Expressing a function in terms of wavelets

Given a wavelet system with scale function ϕ and mother wavelet ψ we wish to express a function $f(x)$ in terms of an orthonormal basis of the wavelet system. First we will express $f(x)$ in terms of scale functions $\phi_{jk}(x) = \phi(2^j x - k)$. To do this we will build a tree similar to that in Figure 11.2 for the Haar system only computing the coefficients will be much more complex. Recall that the coefficients at a level in the tree are the coefficients to represent $f(x)$ using scale functions with the precision of the level.

Let $f(x) = \sum_{k=0}^{\infty} a_{jk}\phi_j(x-k)$ where the a_{jk} are the coefficients in the expansion of $f(x)$ using level j scale functions. Since the $\phi_j(x-k)$ are orthogonal

$$a_{jk} = \int_{x=-\infty}^{\infty} f(x)\phi_j(x-k)dx.$$

Expanding ϕ_j in terms of ϕ_{j+1} yields

$$\begin{aligned}
a_{jk} &= \int_{x=-\infty}^{\infty} f(x)\sum_{m=0}^{d-1} c_m\phi_{j+1}(2x-2k-m)dx \\
&= \sum_{m=0}^{d-1} c_m \int_{x=-\infty}^{\infty} f(x)\phi_{j+1}(2x-2k-m)dx \\
&= \sum_{m=0}^{d-1} c_m a_{j+1,2k+m}
\end{aligned}$$

Let $n = 2k+m$. Now $m = n-2k$. Then

$$a_{jk} = \sum_{n=2k}^{d-1} c_{n-2k}a_{j+1,n} \tag{11.4}$$

In construction the tree similar to that in Figure 11.2, the values at the leaves are the values of the function sampled in the intervals of size 2^{-j}. Equation 11.4 is used to compute values as one moves up the tree. The coefficients in the tree could be used if we wanted to represent $f(x)$ using scale functions. However, we want to represent $f(x)$ using one scale function whose scale is the support of $f(x)$ along with wavelets which gives us an orthogonal set of basis functions. To do this we need to calculate the coefficients for the wavelets.. The value at the root of the tree is the coefficient for the scale function. We then move down the tree calculating the coefficients for the wavelets.

Finish by calculating wavelet coefficients
maybe add material on jpeg

Example: Add example using D_4.
Maybe example using sinc

11.9 Designing a wavelet system

In designing a wavelet system there are a number of parameters in the dilation equation. If one uses d terms in the dilation equation, one degree of freedom can be used to satisfy

$$\sum_{i=0}^{d-1} c_i = 2$$

which insures the existence of a solution with a nonzero mean. Another $\frac{d}{2}$ degrees of freedom are used to satisfy

$$\sum_{i=0}^{d-1} c_i c_{i-2k} = \delta(k)$$

which insures the orthogonal properties. The remaining $\frac{d}{2} - 1$ degrees of freedom can be used to obtain some desirable properties such as smoothness. Smoothness appears to be related to vanishing moments of the scaling function. Material on the design of systems is beyond the scope of this book and can be found in the literature.

Exercises

Exercise 11.1 *What is the solution to the dilation equation $f(x) = f(2x) + f(2x-k)$ for k an integer?*

Exercise 11.2 *Are there solutions to $f(x) = f(2x) + f(2x-1)$ other than a constant multiple of*

$$f(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \textit{otherwise} \end{cases} ?$$

Exercise 11.3 *Is there a solution to $f(x) = \frac{1}{2}f(2x) + f(2x-1) + \frac{1}{2}f(2x-2)$ with $f(0) = f(1) = 1$ and $f(2) = 0$?*

Exercise 11.4 *What is the solution to the dilation equation*

$$f(x) = f(2x) + f(2x-1) + f(2x-2) + f(2x-3).$$

Exercise 11.5 *Consider the dilation equation*

$$f(x) = f(2x) + 2f(2x-1) + 2f(2x-2) + 2f(2x-3) + f(2x-4)$$

1. *What is the solution to the dilation equation?*

2. *What is the value of $\int_{-\infty}^{\infty} f(x)dx$?*

Exercise 11.6 *What are the solutions to the following families of dilation equations.*

1.

$$\begin{aligned}
f(x) =& f(2x) + f(2x-1) \\
f(x) =& \frac{1}{2}f(2x) + \frac{1}{2}f(2x-1) + \frac{1}{2}f(2x-2) + \frac{1}{2}f(2x-3) \\
f(x) =& \frac{1}{4}f(2x) + \frac{1}{4}f(2x-1) + \frac{1}{4}f(2x-2) + \frac{1}{4}f(2x-3) + \frac{1}{4}f(2x-4) + \frac{1}{4}f(2x-5) + \frac{1}{4}f(2x-6) + \\
f(x) =& \frac{1}{k}f(2x) + \frac{1}{k}f(2x) + \cdots + \frac{1}{k}f(2x)
\end{aligned}$$

2.

$$\begin{aligned}
f(x) =& \frac{1}{3}f(2x) + \frac{2}{3}f(2x-1) + \frac{2}{3}f(2x-2) + \frac{1}{3}f(2x-3) \\
f(x) =& \frac{1}{4}f(2x) + \frac{3}{4}f(2x-1) + \frac{3}{4}f(2x-2) + \frac{1}{4}f(2x-3) \\
f(x) =& \frac{1}{5}f(2x) + \frac{4}{5}f(2x-1) + \frac{4}{5}f(2x-2) + \frac{1}{5}f(2x-3) \\
f(x) =& \frac{1}{k}f(2x) + \frac{k-1}{k}f(2x-1) + \frac{k-1}{k}f(2x-2) + \frac{1}{k}f(2x-3)
\end{aligned}$$

3.

$$
\begin{aligned}
f(x) =&\frac{1}{2}f(2x) + \frac{1}{2}f(2x-1) + \frac{1}{2}f(2x-2) + \frac{1}{2}f(2x-3)\\
f(x) =&\frac{3}{2}f(2x) - \frac{1}{2}f(2x-1) + \frac{3}{2}f(2x-2) - \frac{1}{2}f(2x-3)\\
f(x) =&\frac{5}{2}f(2x) - \frac{3}{2}f(2x-1) + \frac{5}{2}f(2x-2) - \frac{3}{2}f(2x-3)\\
f(x) =&\frac{1+2k}{2}f(2x) - \frac{2k-1}{2}f(2x-1) + \frac{1+2k}{2}f(2x-2) - \frac{2k-1}{2}f(2x-3)
\end{aligned}
$$

4.

$$
\begin{aligned}
f(x) =&\frac{1}{3}f(2x) + \frac{2}{3}f(2x-1) + \frac{2}{3}f(2x-2) + \frac{1}{3}f(2x-3)\\
f(x) =&\frac{4}{3}f(2x) - \frac{1}{3}f(2x-1) + \frac{5}{3}f(2x-2) - \frac{2}{3}f(2x-3)\\
f(x) =&\frac{7}{3}f(2x) - \frac{4}{3}f(2x-1) + \frac{8}{3}f(2x-2) - \frac{5}{3}f(2x-3)\\
f(x) =&\frac{1+3k}{3}f(2x) - \frac{2-3k}{3}f(2x-1) + \frac{2+3k}{3}f(2x-2) - \frac{1-3k}{3}f(2x-3)
\end{aligned}
$$

Exercise 11.7

Solution: $\blacksquare$

Exercise 11.8

1. *What is the solution to the dilation equation $f(x) = \frac{1}{2}f(2x) + \frac{3}{2}f(2x-1)$? Hint: Write a program to see what the solution looks like.*

2. *How does the solution change when the equation is changed to $f(x) = \frac{1}{3}f(2x) + \frac{5}{3}f(2x-1)$?*

3. *How does the solution change if the coefficients no longer sum to two as in $f(x) = f(2x) + 3f(2x-1)$?*

Exercise 11.9 *If $f(x)$ is frequency limited by 2π, prove that*

$$f(x) = \sum_{k=0}^{\infty} f(k)\frac{\sin(\pi(x-k))}{\pi(x-k)}.$$

Hint: Use the Nyquist sampling theorem which states that a function frequency limited by 2π is completely determined by samples spaced one unit apart. Note that this result means that

$$f(k) = \int_{-\infty}^{\infty} f(x)\frac{\sin(\pi(x-k))}{\pi(x-k)}dx$$

Exercise 11.10 *Compute an approximation to the scaling function that comes from the dilation equation*

$$\phi(x) = \frac{1+\sqrt{3}}{4}\phi(2x) + \frac{3+\sqrt{3}}{4}\phi(2x-1) + \frac{3-\sqrt{3}}{4}\phi(2x-2) + \frac{1\sqrt{3}}{4}\phi(2x-3).$$

Exercise 11.11 *Assume* $\phi(x) = \sum_{i=0}^{d-1} c_i\phi(2x-i)$, $\psi(x) = \sum_{i=0}^{d-1} b_k\phi(2x-i)$, $b_k = (-1)^k c_{d-1-k}$ *and* $\sum(-1)c_k =$ ***FINISH*** *Prove that* $\int_{-\infty}^{\infty}\psi(x)dx = 0$.
Add conditions $\psi(x) = \sum_{k=0}^{d-1}\phi(x)$, *etc.*

Exercise 11.12 *Consider* $f(x)$ *to consist of the semi circle* $(x-\frac{1}{2})^2 + y^2 = \frac{1}{4}$ *and* $y \geq 0$ *for* $0 \leq x \leq 1$ *and 0 otherwise.*

1. *Using precision* $j = 4$ *find the coefficients for the scale functions and the wavelets for* D_4 *defined by the dilation equation*

$$\phi(x) = \frac{1+\sqrt{3}}{4}\phi(2x) + \frac{3+\sqrt{3}}{4}\phi(2x-1) + \frac{3-\sqrt{3}}{4}\phi(2x-2) + \frac{1\sqrt{3}}{4}\phi(2x-3)$$

2. *Graph the approximation to the semi circle for precision* $j = 4$.

Exercise 11.13 *What is the set of all solutions to the dilation equation*

$$\phi(x) = \frac{1+\sqrt{3}}{4}\phi(2x) + \frac{3+\sqrt{3}}{4}\phi(2x-1) + \frac{3-\sqrt{3}}{4}\phi(2x-2) + \frac{1\sqrt{3}}{4}\phi(2x-3)$$

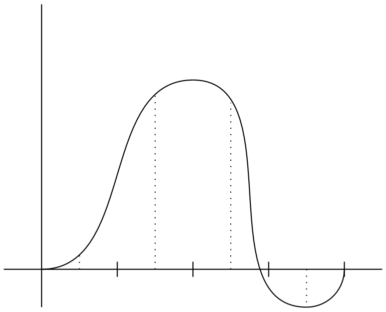
Exercise 11.14 *Prove that if scale functions defined by a dilation equation are orthogonal, then the sum of the even coefficients must equal the sum of the odd coefficients in the dilation equation. That is,* $\sum_k c_{2k} = \sum_k c_{2k+1}$.

```
function  = wavelets

acc=32;  %accuracy of computation
phit=[1:acc zeros(1,3*acc)];

c1=(1+3^0.5)/4; c2=(3+3^0.5)/4; c3=(3-3^0.5)/4; c4=(1-3^0.5)/4;

for i=1:10
    temp=(phit(1:2:4*acc)+phit(2:2:4*acc))/2;
    phi2t=[temp zeros(1,3*acc)];

    phi2tshift1=[ zeros(1,acc) temp zeros(1,2*acc)];
    phi2tshift2=[ zeros(1,2*acc) temp zeros(1,acc)];
    phi2tshift3=[ zeros(1,3*acc) temp ];

    phit=c1*phi2t+c2*phi2tshift1+c3*phi2tshift2+c4*phi2tshift3;

    plot(phit)
    figure(gcf)
    pause

end plot(phit) figure(gcf) end
```

]

To Do

1. Continuous wavelet transform $F(a,b) = \frac{1}{\sqrt{|b|}} \int f(x)\psi(\frac{x-a}{b} dx$
2. References
   (a) Introduction to Wavelets and Wavelet Transforms A primer, Sidney Burrus etal
   (b) Introduction to wavelets, Nimrod Peleg
   (c) A Survey on Wavelet Applications in Data Mining Tao Li et al
   (d) An Introduction to Wavelets (Amara Graps)
   (e) A Really Friendly Guide to Wavelets (C. Valens, 1999)
   (f) Wavelets and their Applications in Databases (Daniel Keim, Martin Heczko)
   (g) David Colella & Christopher Heil Characterizations of scaling functions: continuous solutions
3. image compression jpeg
4. W_2 the orthogonal component of V_1 in V_2.
5. did we prove that integral of wavelet is zero? p252
6. figure



12 Appendix

12.1 Asymptotic Notation

We introduce the big O notation here. The motivating example is the analysis of the running time of an algorithm. The running time may be a complicated function of the input length n such as $5n^3+25n^2\ln n-6n+22$. Asymptotic analysis is concerned with the behavior as $n\to\infty$ where the higher order term $5n^3$ dominates. Further, the coefficient 5 of $5n^3$ is not of interest since its value varies depending on the machine model. So we say that the function is $O(n^3)$. The big O notation applies to functions on the positive integers taking on positive real values.

Definition 12.1 *For functions f and g from the natural numbers to the positive reals, $f(n)$ is $O(g(n))$ if there exists a constant $c>0$ such that for all n, $f(n)\le cg(n)$.* ■

Thus, $f(n)=5n^3+25n^2\ln n-6n+22$ is $O(n^3)$. The upper bound need not be tight. Not only is $f(n)$, $O(n^3)$, it is also $O(n^4)$. Note $g(n)$ must be strictly greater than 0 for all n.

To say that the function $f(n)$ grows at least as fast as $g(n)$, one uses a notation called omega of n. For positive real valued f and g, $f(n)$ is $\Omega(g(n))$ if there exists a constant $c>0$ such that for all n, $f(n)\ge cg(n)$. If $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$, then $f(n)$ is $\Theta(g(n))$. Theta of n is used when the two functions have the same asymptotic growth rate.

Many times one wishes to bound the low order terms. To do this, a notation called little o of n is used. We say $f(n)$ is $o(g(n))$ if $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$. Note that $f(n)$ being $O(g(n))$ means that asymptotically $f(n)$ does not grow faster than $g(n)$, whereas $f(n)$ being $o(g(n))$ means that asymptotically $f(n)/g(n)$ goes to zero. If $f(n)=2n+\sqrt{n}$, then

asymptotic upper bound
$f(n)$ is $O(g(n))$ if for all n, $f(n)\le cg(n)$ for some constant $c>0$. $\le$

asymptotic lower bound
$f(n)$ is $\Omega(g(n))$ if for all n, $f(n)\ge cg(n)$ for some constant $c>0$. $\ge$

asymptotic equality
$f(n)$ is $\Theta(g(n))$ if it is both $O(g(n))$ and $\Omega(g(n))$. $=$

$f(n)$ is $o(g(n))$ if $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$. $<$

$f(n)\sim g(n)$ if $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=1$. $=$

$f(n)$ is $\omega\left(g\left(n\right)\right)$ **if** $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=\infty$. $>$

$f(n)$ is $O(n)$ but in bounding the lower order term, we write $f(n) = 2n + o(n)$. Finally, we write $f(n) \sim g(n)$ if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = 1$ and say $f(n)$ is $\omega(g(n))$ if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$. The difference between $f(n)$ being $\Theta(g(n))$ and $f(n) \sim g(n)$ is that in the first case $f(n)$ and $g(n)$ may differ by a multiplicative constant factor.

12.2 Useful relations

Summations

$$\sum_{i=0}^{n} a^i = 1 + a + a^2 + \cdots = \frac{1 - a^{n+1}}{1 - a}, \quad a \neq 1$$

$$\sum_{i=0}^{\infty} a^i = 1 + a + a^2 + \cdots = \frac{1}{1 - a}, \quad |a| < 1$$

$$\sum_{i=0}^{\infty} ia^i = a + 2a^2 + 3a^3 \cdots = \frac{a}{(1 - a)^2}, \quad |a| < 1$$

$$\sum_{i=0}^{\infty} i^2 a^i = a + 4a^2 + 9a^3 \cdots = \frac{a(1 + a)}{(1 - a)^3}, \quad |a| < 1$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$

We prove one equality.

$$\sum_{i=0}^{\infty} ia^i = a + 2a^2 + 3a^3 \cdots = \frac{a}{(1 - a)^2}, \text{ provided } |a| < 1.$$

Write $S = \sum\limits_{i=0}^{\infty} ia^i$.

$$aS = \sum_{i=0}^{\infty} ia^{i+1} = \sum_{i=1}^{\infty} (i - 1)a^i.$$

Thus,

$$S - aS = \sum_{i=1}^{\infty} ia^i - \sum_{i=1}^{\infty} (i - 1)a^i = \sum_{i=1}^{\infty} a^i = \frac{a}{1 - a},$$

from which the equality follows. The sum $\sum\limits_{i} i^2 a^i$ can also be done by an extension of this method (left to the reader). Using generating functions, we will see another proof of both these equalities by derivatives.

$$\sum_{i=1}^{\infty}\frac{1}{i}=1+\tfrac{1}{2}+\left(\tfrac{1}{3}+\tfrac{1}{4}\right)+\left(\tfrac{1}{5}+\tfrac{1}{6}+\tfrac{1}{7}+\tfrac{1}{8}\right)+\cdots\geq 1+\tfrac{1}{2}+\tfrac{1}{2}+\cdots \text{ and thus diverges.}$$

The summation $\sum\limits_{i=1}^{n}\frac{1}{i}$ grows as $\ln n$ since $\sum\limits_{i=1}^{n}\frac{1}{i}\approx\int_{x=1}^{n}\frac{1}{x}\,dx$. In fact, $\lim\limits_{i\to\infty}\left(\sum\limits_{i=1}^{n}\frac{1}{i}-\ln(n)\right)=\gamma$ where $\gamma\cong 0.5772$ is Euler's constant. Thus, $\sum\limits_{i=1}^{n}\frac{1}{i}\cong\ln(n)+\gamma$ for large n.

Truncated Taylor series

If all the derivatives of a function $f(x)$ exist, then we can write

$$f(x)=f(0)+f'(0)x+f''(0)\frac{x^2}{2}+\cdots.$$

The series can be truncated. In fact, there exists some y between 0 and x such that

$$f(x)=f(0)+f'(y)x.$$

Also, there exists some z between 0 and x such that

$$f(x)=f(0)+f'(0)x+f''(z)\frac{x^2}{2}$$

and so on for higher derivatives. This can be used to derive inequalities. For example, if $f(x)=\ln(1+x)$, then its derivatives are

$$f'(x)=\frac{1}{1+x}\ ;\ f''(x)=-\frac{1}{(1+x)^2}\ ;\ f'''(x)=\frac{2}{(1+x)^3}.$$

For any z, $f''(z)<0$ and thus for any x, $f(x)\leq f(0)+f'(0)x$ hence, $\ln(1+x)\leq x$, which also follows from the inequality $1+x\leq e^x$. Also using

$$f(x)=f(0)+f'(0)x+f''(0)\frac{x^2}{2}+f'''(z)\frac{x^3}{3!}$$

for $z>-1$, $f'''(z)>0$, and so for $x>-1$,

$$\ln(1+x)>x-\frac{x^2}{2}.$$

Exponentials and logs

$$a^{\log b}=b^{\log a}$$

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots\qquad e=2.7182\qquad \tfrac{1}{e}=0.3679$$

Setting $x=1$ in the equation $e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$ yields $e=\sum\limits_{i=0}^{\infty}\frac{1}{i!}$.

$$\lim_{n\to\infty}\left(1+\tfrac{a}{n}\right)^n = e^a$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 \cdots \quad |x| < 1$$

The above expression with $-x$ substituted for x gives rise to the approximations

$$\ln(1-x) < -x$$

which also follows from $1-x \leq e^{-x}$, since $\ln(1-x)$ is a monotone function for $x \in (0,1)$.

For $0 < x < 0.69$, $\ln(1-x) > -x - x^2$.

Trigonometric identities

$$\begin{array}{l}
e^{ix} = \cos(x) + i\sin(x) \\
\cos(x) = \frac{1}{2}\left(e^{ix} + e^{-ix}\right) \\
\sin(x) = \frac{1}{2i}\left(e^{ix} - e^{-ix}\right) \\
\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y) \\
\cos(x \pm y) = cos(x)\cos(y) \mp \sin(x)\sin(y) \\
\cos\left(2\theta\right) = \cos^2\theta - \sin^2\theta = 1 - 2\sin^2\theta \\
\sin\left(2\theta\right) = 2\sin\theta\cos\theta \\
\sin^2\frac{\theta}{2} = \frac{1}{2}\left(1 - \cos\theta\right) \\
\cos^2\frac{\theta}{2} = \frac{1}{2}\left(1 + \cos\theta\right)
\end{array}$$

Gaussian and related integrals

$$\int xe^{ax^2}dx = \frac{1}{2a}e^{ax^2}$$

$$\int \tfrac{1}{a^2+x^2}dx = \tfrac{1}{a}\tan^{-1}\tfrac{x}{a} \text{ thus } \int\limits_{-\infty}^{\infty} \tfrac{1}{a^2+x^2}dx = \tfrac{\pi}{a}$$

$$\int\limits_{-\infty}^{\infty} e^{-\frac{a^2x^2}{2}}dx = \frac{\sqrt{2\pi}}{a} \text{ thus } \frac{a}{\sqrt{2\pi}}\int\limits_{-\infty}^{\infty} e^{-\frac{a^2x^2}{2}}dx = 1$$

$$\int\limits_{0}^{\infty} x^2e^{-ax^2}dx = \frac{1}{4a}\sqrt{\frac{\pi}{a}}$$

$$\int\limits_{0}^{\infty} x^{2n}e^{-\frac{x^2}{a^2}}dx = \sqrt{\pi}\frac{1\cdot 3\cdot 5\cdots(2n-1)}{2^{n+1}}a^{2n-1} = \sqrt{\pi}\frac{(2n)!}{n!}\left(\frac{a}{2}\right)^{2n+1}$$

$$\int_0^{\infty} x^{2n+1}e^{-\frac{x^2}{a^2}}dx = \frac{n!}{2}a^{2n+2}$$

$$\int\limits_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi}$$

To verify $\int\limits_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi}$, consider $\left(\int\limits_{-\infty}^{\infty} e^{-x^2}dx\right)^2 = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} e^{-\left(x^2+y^2\right)}dxdy$. Let $x = r\cos\theta$ and $y = r\sin\theta$. The Jacobian of this transformation of variables is

$$J(r,\theta) = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r$$

Thus,

$$\begin{aligned}\left(\int\limits_{-\infty}^{\infty} e^{-x^2}dx\right)^2 &= \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} e^{-\left(x^2+y^2\right)}dxdy = \int\limits_{0}^{\infty}\int\limits_{0}^{2\pi} e^{-r^2}J(r,\theta)\,drd\theta \\ &= \int\limits_{0}^{\infty} e^{-r^2}rdr\int\limits_{0}^{2\pi} d\theta \\ &= -2\pi\left[\tfrac{e^{-r^2}}{2}\right]_0^{\infty} = \pi\end{aligned}$$

Thus, $\int\limits_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi}$.

Miscellaneous integrals

$$\int_{x=0}^{1} x^{\alpha-1}(1-\alpha)^{\beta-1}dx = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

For definition of the gamma function see Section 12.3 **Binomial coefficients**

The binomial coefficient $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ is the number of ways of choosing k items from n. The number of ways of choosing $d+1$ items from $n+1$ items equals the number of ways of choosing the $d+1$ items from the first n items plus the number of ways of choosing d of the items from the first n items with the other item being the last of the $n+1$ items.

$$\binom{n}{d} + \binom{n}{d+1} = \binom{n+1}{d+1}.$$

The observation that the number of ways of choosing k items from $2n$ equals the number of ways of choosing i items from the first n and choosing $k-i$ items from the second n summed over all i, $0 \le i \le k$ yields the identity

$$\sum_{i=0}^{k} \binom{n}{i}\binom{n}{k-i} = \binom{2n}{k}.$$

Setting $k = n$ in the above formula and observing that $\binom{n}{i} = \binom{n}{n-i}$ yields

$$\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n}.$$

More generally $\sum\limits_{i=0}^{k} \binom{n}{i}\binom{m}{k-i} = \binom{n+m}{k}$ by a similar derivation.

12.3 Useful Inequalities

$1 + x \le e^x$ for all real x.

One often establishes an inequality such as $1 + x \le e^x$ by showing that the difference of the two sides, namely $e^x - (1+x)$, is always positive. This can be done by taking derivatives. The first and second derivatives are $e^x - 1$ and e^x. Since e^x is always positive, $e^x - 1$ is monotonic and $e^x - (1+x)$ is convex. Since $e^x - 1$ is monotonic, it can be zero only once and is zero at $x = 0$. Thus, $e^x - (1+x)$ takes on its minimum at $x = 0$ where it is zero establishing the inequality.

$(1-x)^n \ge 1 - nx$ for $0 \le x \le 1$

$1 + x \leq e^x$ for all real x

$(1-x)^n \geq 1 - nx$ for $0 \leq x \leq 1$

$(x+y)^2 \leq 2x^2 + 2y^2$

Triangle Inequality $|\mathbf{x}+\mathbf{y}| \leq |\mathbf{x}| + |\mathbf{y}|$.

Cauchy-Schwartz Inequality $|\mathbf{x}||\mathbf{y}| \geq \mathbf{x}^T\mathbf{y}$

Young's Inequality For positive real numbers p and q where $\frac{1}{p} + \frac{1}{q} = 1$ and positive reals x and y,

$$xy \leq \frac{1}{p}x^p + \frac{1}{q}y^q.$$

Hölder's inequalityHölder's inequality For positive real numbers p and q with $\frac{1}{p} + \frac{1}{q} = 1$,

$$\sum_{i=1}^{n} |x_i y_i| \ \leq \ \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \left(\sum_{i=1}^{n} |y_i|^q\right)^{1/q}.$$

Jensen's inequality For a convex function f,

$$f\left(\sum_{i=1}^{n} \alpha_i x_i\right) \leq \sum_{i=1}^{n} \alpha_i f(x_i),$$

Let $g(x) = (1-x)^n - (1-nx)$. We establish $g(x) \geq 0$ for x in $[0,1]$ by taking the derivative.

$$g'(x) = -n(1-x)^{n-1} + n = n\big(1-(1-x)^{n-1}\big) \geq 0$$

for $0 \leq x \leq 1$. Thus, g takes on its minimum for x in $[0,1]$ at $x=0$ where $g(0)=0$ proving the inequality.

$(x+y)^2 \leq 2x^2 + 2y^2$

The inequality follows from $(x+y)^2 + (x-y)^2 = 2x^2 + 2y^2$.

Lemma 12.1 *For any nonnegative reals $a_1, a_2, \ldots, a_n$ and any $\rho \in [0,1]$, $\left(\sum_{i=1}^n a_i\right)^\rho \leq \sum_{i=1}^n a_i^\rho$.*

Proof: We will see that we can reduce the proof of the lemma to the case when only one of the a_i is nonzero and the rest are zero. To this end, suppose a_1 and a_2 are both positive and without loss of generality, assume $a_1 \geq a_2$. Add an infinitesimal positive amount ϵ to a_1 and subtract the same amount from a_2. This does not alter the left hand side. We claim it does not increase the right hand side. To see this, note that

$$(a_1 + \epsilon)^\rho + (a_2 - \epsilon)^\rho - a_1^\rho - a_2^\rho = \rho(a_1^{\rho-1} - a_2^{\rho-1})\epsilon + O(\epsilon^2),$$

and since $\rho - 1 \leq 0$, we have $a_1^{\rho-1} - a_2^{\rho-1} \leq 0$, proving the claim. Now by repeating this process, we can make $a_2 = 0$ (at that time a_1 will equal the sum of the original a_1 and a_2). Now repeating on all pairs of a_i, we can make all but one of them zero and in the process, we have left the left hand side the same, but have not increased the right hand side. So it suffices to prove the inequality at the end which clearly holds. This method of proof is called the variational method. ∎

The Triangle Inequality

For any two vectors $\mathbf{x}$ and $\mathbf{y}$, $|\mathbf{x}+\mathbf{y}| \leq |\mathbf{x}| + |\mathbf{y}|$. Since $\mathbf{x} \cdot \mathbf{y} \leq |\mathbf{x}||\mathbf{y}|$,

$$|\mathbf{x}+\mathbf{y}|^2 = (\mathbf{x}+\mathbf{y})^T \cdot (\mathbf{x}+\mathbf{y}) = |\mathbf{x}|^2 + |\mathbf{y}|^2 + 2\mathbf{x}^T \cdot \mathbf{y} \leq |\mathbf{x}|^2 + |\mathbf{y}|^2 + 2|\mathbf{x}||\mathbf{y}| = (|\mathbf{x}| + |\mathbf{y}|)^2.$$

The inequality follows by taking square roots.

Stirling approximation

$$n! \cong \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \qquad \binom{2n}{n} \cong \frac{1}{\sqrt{\pi n}} 2^{2n}$$

$$\sqrt{2\pi n}\frac{n^n}{e^n} < n! < \sqrt{2\pi n}\frac{n^n}{e^n}\left(1 + \frac{1}{12n-1}\right)$$

We prove the inequalities, except for constant factors. Namely, we prove that

$$1.4 \left(\frac{n}{e}\right)^n \sqrt{n} \leq n! \leq e\left(\frac{n}{e}\right)^n \sqrt{n}.$$

Write $\ln(n!) = \ln 1 + \ln 2 + \cdots + \ln n$. This sum is approximately $\int_{x=1}^n \ln x \, dx$. The indefinite integral $\int \ln x \, dx = (x \ln x - x)$ gives an approximation, but without the $\sqrt{n}$ term. To get the $\sqrt{n}$, differentiate twice and note that $\ln x$ is a concave function. This means that for any positive x_0,

$$\frac{\ln x_0 + \ln(x_0 + 1)}{2} \leq \int_{x=x_0}^{x_0+1} \ln x \, dx,$$

since for $x \in [x_0, x_0 + 1]$, the curve $\ln x$ is always above the spline joining $(x_0, \ln x_0)$ and $(x_0 + 1, \ln(x_0 + 1))$. Thus,

$$\begin{aligned}
\ln(n!) &= \frac{\ln 1}{2} + \frac{\ln 1 + \ln 2}{2} + \frac{\ln 2 + \ln 3}{2} + \cdots + \frac{\ln(n-1) + \ln n}{2} + \frac{\ln n}{2} \\
&\leq \int_{x=1}^n \ln x \, dx + \frac{\ln n}{2} = [x \ln x - x]_1^n + \frac{\ln n}{2} \\
&= n \ln n - n + 1 + \frac{\ln n}{2}.
\end{aligned}$$

Thus, $n! \le n^n e^{-n} \sqrt{n} e$. For the lower bound on $n!$, start with the fact that for any $x_0 \ge 1/2$ and any real ρ

$$\ln x_0 \ge \frac{1}{2}(\ln(x_0+\rho) + \ln(x_0-\rho)) \quad \text{implies} \quad \ln x_0 \ge \int_{x=x_0-0.5}^{x_0+.5} \ln x \, dx.$$

Thus,

$$\ln(n!) = \ln 2 + \ln 3 + \cdots + \ln n \ge \int_{x=1.5}^{n+.5} \ln x \, dx,$$

from which one can derive a lower bound with a calculation.

Stirling approximation for the binomial coefficient

$$\binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

Using the Stirling approximation for $k!$,

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} \le \frac{n^k}{k!} \cong \left(\frac{en}{k}\right)^k.$$

The gamma function

For $a > 0$

$$\Gamma(a) = \int\limits_0^\infty x^{a-1} e^{-x} dx$$

$$\Gamma\left(\tfrac{1}{2}\right) = \sqrt{\pi}, \quad \Gamma(1) = \Gamma(2) = 1, \text{ and for } n \ge 2, \quad \Gamma(n) = (n-1)\Gamma(n-1).$$

The last statement is proved by induction on n. It is easy to see that $\Gamma(1) = 1$. For $n \ge 2$, we use integration by parts.

$$\int f(x)\, g'(x)\, dx = f(x)\, g(x) - \int f'(x)\, g(x)\, dx$$

Write $\Gamma(n) = \int_{x=0}^{\infty} f(x)g'(x)\, dx$, where, $f(x) = x^{n-1}$ and $g'(x) = e^{-x}$. Thus,

$$\Gamma(n) = [f(x)g(x)]_{x=0}^{\infty} + \int_{x=0}^{\infty} (n-1)x^{n-2}e^{-x}\, dx = (n-1)\Gamma(n-1),$$

as claimed.

Cauchy-Schwartz Inequality

$$\left(\sum_{i=1}^n x_i^2\right)\left(\sum_{i=1}^n y_i^2\right) \ge \left(\sum_{i=1}^n x_i y_i\right)^2$$

In vector form, $|\mathbf{x}||\mathbf{y}| \geq \mathbf{x}^T\mathbf{y}$, the inequality states that the dot product of two vectors is at most the product of their lengths. The Cauchy-Schwartz inequality is a special case of Hölder's inequality with $p = q = 2$.

Young's inequality

For positive real numbers p and q where $\frac{1}{p} + \frac{1}{q} = 1$ and positive reals x and y,

$$\frac{1}{p}x^p + \frac{1}{q}y^q \geq xy.$$

The left hand side of Young's inequality, $\frac{1}{p}x^p + \frac{1}{q}y^q$, is a convex combination of x^p and y^q since $\frac{1}{p}$ and $\frac{1}{q}$ sum to 1. $\ln(x)$ is a concave function for $x > 0$ and so the ln of the convex combination of the two elements is greater than or equal to the convex combination of the ln of the two elements

$$\ln(\frac{1}{p}x^p + \frac{1}{q}y^p) \geq \frac{1}{p}\ln(x^p) + \frac{1}{q}\ln(y^q) = \ln(xy).$$

Since for $x \geq 0$, $\ln x$ is a monotone increasing function, $\frac{1}{p}x^p + \frac{1}{q}y^q \geq xy.$.

Hölder's inequalityHölder's inequality

For positive real numbers p and q with $\frac{1}{p} + \frac{1}{q} = 1$,

$$\sum_{i=1}^{n} |x_i y_i| \ \leq \ \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \left(\sum_{i=1}^{n} |y_i|^q\right)^{1/q}.$$

Let $x_i' = x_i \ / \ (\sum_{i=1}^n |x_i|^p)^{1/p}$ and $y_i' = y_i \ / \ (\sum_{i=1}^n |y_i|^q)^{1/q}$. Replacing x_i by x_i' and y_i by y_i' does not change the inequality. Now $\sum_{i=1}^n |x_i'|^p = \sum_{i=1}^n |y_i'|^q = 1$, so it suffices to prove $\sum_{i=1}^n |x_i'y_i'| \leq 1$. Apply Young's inequality to get $|x_i'y_i'| \leq \frac{|x_i'|^p}{p} + \frac{|y_i'|^q}{q}$. Summing over i, the right hand side sums to $\frac{1}{p} + \frac{1}{q} = 1$ finishing the proof.

For $a_1, a_2, \ldots, a_n$ real and k a positive integer,

$$(a_1 + a_2 + \cdots + a_n)^k \leq n^{k-1}(|a_1|^k + |a_2|^k + \cdots + |a_n|^k).$$

Using Hölder's inequality with $p = k$ and $q = k/(k-1)$,

$$\begin{aligned}|a_1 + a_2 + \cdots + a_n| &\leq |a_1 \cdot 1| + |a_2 \cdot 1| + \cdots + |a_n \cdot 1| \\ &\leq \left(\sum_{i=1}^{n} |a_i|^k\right)^{1/k} (1 + 1 + \cdots + 1)^{(k-1)/k},\end{aligned}$$

from which the current inequality follows.

Arithmetic and geometric means

The arithmetic mean of a set of nonnegative reals is at least their geometric mean. For $a_1, a_2, \ldots, a_n > 0$,

$$\frac{1}{n}\sum_{i=1}^{n} a_i \geq \sqrt[n]{a_1 a_2 \cdots a_n}.$$

Assume that $a_1 \geq a_2 \geq \ldots \geq a_n$. We reduce the proof to the case when all the a_i are equal using the variational method; in this case the inequality holds with equality. Suppose $a_1 > a_2$. Let ε be a positive infinitesimal. Add ε to a_2 and subtract ε from a_1 to get closer to the case when they are equal. The left hand side $\frac{1}{n}\sum_{i=1}^{n} a_i$ does not change.

$$\begin{aligned}(a_1-\varepsilon)(a_2+\varepsilon)a_3a_4\cdots a_n &= a_1a_2\cdots a_n + \varepsilon(a_1-a_2)a_3a_4\cdots a_n + O(\varepsilon^2)\\ &> a_1a_2\cdots a_n\end{aligned}$$

for small enough $\varepsilon > 0$. Thus, the change has increased $\sqrt[n]{a_1a_2\cdots a_n}$. So if the inequality holds after the change, it must hold before. By continuing this process, one can make all the a_i equal.

Approximating sums by integrals

For monotonic decreasing $f(x)$,

$$\int\limits_{x=m}^{n+1} f(x)dx \leq \sum_{i=m}^{n} f(i) \leq \int\limits_{x=m-1}^{n} f(x)dx.$$

See Fig. 12.1. Thus,

$$\int\limits_{x=2}^{n+1} \tfrac{1}{x^2}dx \leq \sum_{i=2}^{n} \tfrac{1}{i^2} = \tfrac{1}{4} + \tfrac{1}{9} + \cdots + \tfrac{1}{n^2} \leq \int\limits_{x=1}^{n} \tfrac{1}{x^2}dx$$

and hence $\frac{3}{2} - \frac{1}{n+1} \leq \sum\limits_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$.

Jensen's Inequality

For a convex function f,

$$f\left(\frac{1}{2}(x_1+x_2)\right) \leq \frac{1}{2}\left(f(x_1)+f(x_2)\right).$$

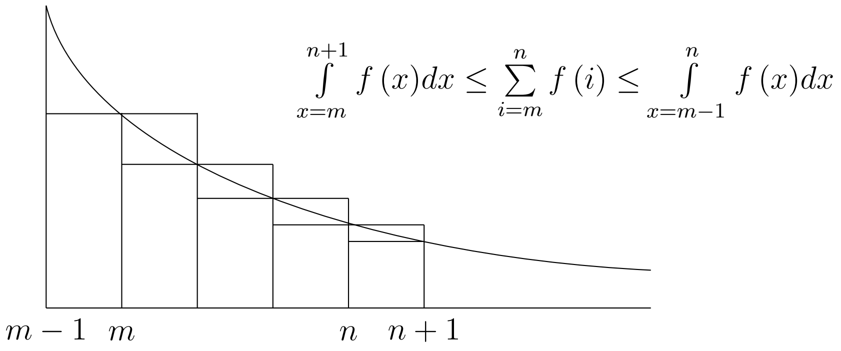

$\int_{x=m}^{n+1} f(x)dx \leq \sum_{i=m}^{n} f(i) \leq \int_{x=m-1}^{n} f(x)dx$
$m-1$
m
n
$n+1$

Figure 12.1: Approximating sums by integrals

More generally for any convex function f,

$$f\left(\sum_{i=1}^{n} \alpha_i x_i\right) \leq \sum_{i=1}^{n} \alpha_i f(x_i),$$

where $0 \leq \alpha_i \leq 1$ and $\sum_{i=1}^{n} \alpha_i = 1$. From this, it follows that for any convex function f and random variable x,

$$E(f(x)) \geq f(E(x)).$$

We prove this for a discrete random variable x taking on values $a_1, a_2, \ldots$ with $\text{Prob}(x = a_i) = \alpha_i$:

$$E(f(x)) = \sum_i \alpha_i f(a_i) \geq f\left(\sum_i \alpha_i a_i\right) = f(E(x)).$$

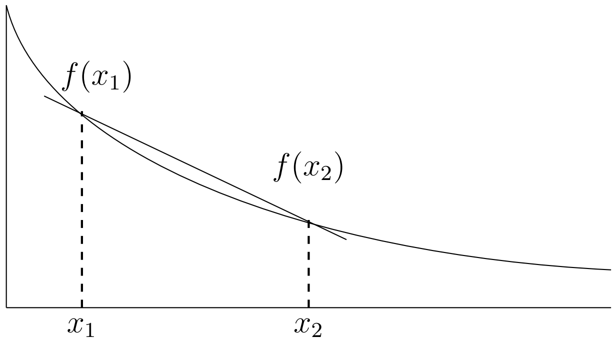


Figure 12.2: For a convex function f, $f\left(\frac{x_1+x_2}{2}\right) \leq \frac{1}{2}\left(f(x_1) + f(x_2)\right)$.

Example: Let $f(x) = x^k$ for k an even positive integer. Then, $f''(x) = k(k-1)x^{k-2}$ which since $k-2$ is even is nonnegative for all x implying that f is convex. Thus,

$$E(x) \le \sqrt[k]{E(x^k)},$$

since $t^{\frac{1}{k}}$ is a monotone function of t, $t > 0$. It is easy to see that this inequality does not necessarily hold when k is odd; indeed for odd k, x^k is not a convex function. ■

Tails of Gaussian

For bounding the tails of Gaussian densities, the following inequality is useful. The proof uses a technique useful in many contexts. For $t > 0$,

$$\int_{x=t}^{\infty} e^{-x^2}\, dx \le \frac{e^{-t^2}}{2t}.$$

In proof, first write: $\int_{x=t}^{\infty} e^{-x^2}\, dx \le \int_{x=t}^{\infty} \frac{x}{t} e^{-x^2}\, dx$, using the fact that $x \ge t$ in the range of integration. The latter expression is integrable in closed form since $d(e^{-x^2}) = (-2x)e^{-x^2}$ yielding the claimed bound.

A similar technique yields an upper bound on

$$\int_{x=\beta}^{1} (1-x^2)^{\alpha}\, dx,$$

for $\beta \in [0,1]$ and $\alpha > 0$. Just use $(1-x^2)^{\alpha} \le \frac{x}{\beta}(1-x^2)^{\alpha}$ over the range and integrate in closed form the last expression.

$$\begin{aligned} \int_{x=\beta}^{1} (1-x^2)^{\alpha} dx \le \int_{x=\beta}^{1} \frac{x}{\beta}(1-x^2)^{\alpha} dx = \frac{-1}{2\beta(\alpha+1)}(1-x^2)^{\alpha+1}\bigg|_{x=\beta}^{1} \\ = \frac{(1-\beta^2)^{\alpha+1}}{2\beta(\alpha+1)} \end{aligned}$$

12.4 Probability

Consider an experiment such as flipping a coin whose outcome is determined by chance. To talk about the outcome of a particular experiment, we introduce the notion of a *random variable* whose value is the outcome of the experiment. The set of possible outcomes is called the *sample space*. If the sample space is finite, we can assign a probability of occurrence to each outcome. In some situations where the sample space is infinite, we can assign a probability of occurrence. The probability $p(i) = \frac{6}{\pi^2}\frac{1}{i^2}$ for i an integer greater than or equal to one is such an example. The function assigning the probabilities is called a *probability distribution function.*

In many situations, a probability distribution function does not exist. For example, for the uniform probability on the interval [0,1], the probability of any specific value is zero. What we can do is define a *probability density function* $p(x)$ such that

$$\text{Prob}(a < x < b) = \int\limits_a^b p(x)dx$$

If x is a continuous random variable for which a density function exists, then the *cumulative distribution function* $f(a)$ is defined by

$$f(a) = \int_{-\infty}^{a} p(x)dx$$

which gives the probability that $x \leq a$.

12.4.1 Sample Space, Events, Independence

There may be more than one relevant random variable in a situation. For example, if one tosses n coins, there are n random variables, $x_1, x_2, \ldots, x_n$, taking on values 0 and 1, a 1 for heads and a 0 for tails. The set of possible outcomes, the sample space, is $\{0, 1\}^n$. An *event* is a subset of the sample space. The event of an odd number of heads, consists of all elements of $\{0, 1\}^n$ with an odd number of 1's.

Let A and B be two events. The joint occurrence of the two events is denoted by $(A \wedge B)$. The *conditional probability* of event A given that event B has occurred is denoted by $\text{Prob}(A|B)$and is given by

$$\text{Prob}(A|B) = \frac{\text{Prob}(A \wedge B)}{\text{Prob}(B)}.$$

Events A and B are *independent* if the occurrence of one event has no influence on the probability of the other. That is, $\text{Prob}(A|B) = \text{Prob}(A)$ or equivalently, $\text{Prob}(A \wedge B) = \text{Prob}(A)\text{Prob}(B)$. Two random variables x and y are *independent* if for every possible set A of values for x and every possible set B of values for y, the events x in A and y in B are independent.

A collection of n random variables $x_1, x_2, \ldots, x_n$ is *mutually independent* if for all possible sets $A_1, A_2, \ldots, A_n$ of values of $x_1, x_2, \ldots, x_n$,

$$\text{Prob}(x_1 \in A_1, x_2 \in A_2, \ldots, x_n \in A_n) = \text{Prob}(x_1 \in A_1)\text{Prob}(x_2 \in A_2) \cdots \text{Prob}(x_n \in A_n).$$

If the random variables are discrete, it would suffice to say that for any real numbers $a_1, a_2, \ldots, a_n$

$$\text{Prob}(x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n) = \text{Prob}(x_1 = a_1)\text{Prob}(x_2 = a_2) \cdots \text{Prob}(x_n = a_n).$$

Random variables $x_1, x_2, \ldots, x_n$ are pairwise independent if for any a_i and a_j, $i \neq j$, $\text{Prob}(x_i = a_i, x_j = a_j) = \text{Prob}(x_i = a_i)\text{Prob}(x_j = a_j)$. Mutual independence is much stronger than requiring that the variables are pairwise independent. Consider the example of 2-universal hash functions discussed in Chapter **??**.

If (x, y) is a random vector and one normalizes it to a unit vector $\left(\frac{x}{\sqrt{x^2+y^2}}, \frac{y}{\sqrt{x^2+y^2}}\right)$ the coordinates are no longer independent since knowing the value of one coordinate uniquely determines the value of the other.

12.4.2 Linearity of Expectation

An important concept is that of the expectation of a random variable. The *expected value*, $E(x)$, of a random variable x is $E(x) = \sum\limits_x xp(x)$ in the discrete case and $E(x) = \int\limits_{-\infty}^{\infty} xp(x)dx$ in the continuous case. The expectation of a sum of random variables is equal to the sum of their expectations. The linearity of expectation follows directly from the definition and does not require independence.

12.4.3 Union Bound

Let $A_1, A_2, \ldots, A_n$ be events. The actual probability of the union of events is given by Boole's formula.

$$\text{Prob}(A_1 \cup A_2 \cup \cdots A_n) = \sum_{i=1}^{n} \text{Prob}(A_i) - \sum_{ij} \text{Prob}(A_i \wedge A_j) + \sum_{ijk} \text{Prob}(A_i \wedge A_j \wedge A_k) - \cdots$$

Often we only need an upper bound on the probability of the union and use

$$\text{Prob}(A_1 \cup A_2 \cup \cdots A_n) \leq \sum_{i=1}^{n} \text{Prob}(A_i)$$

This upper bound is called the *union bound.*

12.4.4 Indicator Variables

A useful tool is that of an indicator variable that takes on value 0 or 1 to indicate whether some quantity is present or not. The indicator variable is useful in determining the expected size of a subset. Given a random subset of the integers $\{1, 2, \ldots, n\}$, the expected size of the subset is the expected value of $x_1 + x_2 + \cdots + x_n$ where x_i is the indicator variable that takes on value 1 if i is in the subset.

Example: Consider a random permutation of n integers. Define the indicator function $x_i = 1$ if the i^{th} integer in the permutation is i. The expected number of fixed points is given by

$$E\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} E(x_i) = n\frac{1}{n} = 1.$$

Note that the x_i are not independent. But, linearity of expectation still applies. ■

Example: Consider the expected number of vertices of degree d in a random graph $G(n,p)$. The number of vertices of degree d is the sum of n indicator random variables, one for each vertex, with value one if the vertex has degree d. The expectation is the sum of the expectations of the n indicator random variables and this is just n times the expectation of one of them. Thus, the expected number of degree d vertices is $n\binom{n}{d}p^d(1-p)^{n-d}$. ■

12.4.5 Variance

In addition to the expected value of a random variable, another important parameter is the variance. The *variance* of a random variable x, denoted var(x) or often $\sigma^2(x)$ is $E\left(x - E\left(x\right)\right)^2$ and measures how close to the expected value the random variable is likely to be. The *standard deviation* σ is the square root of the variance. The units of σ are the same as those of x.

By linearity of expectation

$$\sigma^2 = E\left(x - E\left(x\right)\right)^2 = E(x^2) - 2E(x)E(x) + E^2(x) = E\left(x^2\right) - E^2\left(x\right).$$

12.4.6 Variance of the Sum of Independent Random Variables

In general, the variance of the sum is not equal to the sum of the variances. However, if x and y are independent, then $E\left(xy\right) = E\left(x\right)E\left(y\right)$ and

$$var(x+y) = var\left(x\right) + var\left(y\right).$$

To see this

$$\begin{aligned} var(x+y) &= E\left((x+y)^2\right) - E^2(x+y) \\ &= E(x^2) + 2E(xy) + E(y^2) - E^2(x) - 2E(x)E(y) - E^2(y). \end{aligned}$$

From independence, $2E(xy) - 2E(x)E(y) = 0$ and

$$\begin{aligned} var(x+y) &= E(x^2) - E^2(x) + E(y^2) - E^2(y) \\ &= var(x) + var(y). \end{aligned}$$

More generally, if $x_1, x_2, \ldots, x_n$ are pairwise independent random variables, then

$$var(x_1 + x_2 + \cdots + x_n) = var(x_1) + var(x_2) + \cdots + var(x_n).$$

For the variance of the sum to be the sum of the variances only requires pairwise independence not full independence.

12.4.7 Median

One often calculates the average value of a random variable to get a feeling for the magnitude of the variable. This is reasonable when the probability distribution of the variable is Gaussian, or has a small variance. However, if there are outliers, then the average may be distorted by outliers. An alternative to calculating the expected value is to calculate the median, the value for which half of the probability is above and half is below.

12.4.8 The Central Limit Theorem

Let $s = x_1 + x_2 + \cdots + x_n$ be a sum of n independent random variables where each x_i has probability distribution

$$x_i = \begin{cases} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{cases}.$$

The expected value of each x_i is $1/2$ with variance

$$\sigma_i^2 = \left(\frac{1}{2} - 0\right)^2 \frac{1}{2} + \left(\frac{1}{2} - 1\right)^2 \frac{1}{2} = \frac{1}{4}.$$

The expected value of s is $n/2$ and since the variables are independent, the variance of the sum is the sum of the variances and hence is $n/4$. How concentrated s is around its mean depends on the standard deviation of s which is $\frac{\sqrt{n}}{2}$. For n equal 100 the expected value of s is 50 with a standard deviation of 5 which is 10% of the mean. For $n = 10,000$ the expected value of s is 5,000 with a standard deviation of 50 which is 1% of the mean. Note that as n increases, the standard deviation increases, but the ratio of the standard deviation to the mean goes to zero. More generally, if x_i are independent and identically distributed, each with standard deviation σ, then the standard deviation of $x_1 + x_2 + \cdots + x_n$ is $\sqrt{n}\sigma$. So, $\frac{x_1+x_2+\cdots+x_n}{\sqrt{n}}$ has standard deviation σ. The central limit theorem makes a stronger assertion that in fact $\frac{x_1+x_2+\cdots+x_n}{\sqrt{n}}$ has Gaussian distribution with standard deviation σ.

Theorem 12.2 *Suppose $x_1, x_2, \ldots, x_n$ is a sequence of identically distributed independent random variables, each with mean μ and variance σ^2. The distribution of the random variable*

$$\frac{1}{\sqrt{n}}\left(x_1 + x_2 + \cdots + x_n - n\mu\right)$$

converges to the distribution of the Gaussian with mean 0 and variance σ^2.

12.4.9 Probability Distributions

The Gaussian or normal distribution

The normal distribution is

$$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\frac{(x-m)^2}{\sigma^2}}$$

where m is the mean and σ^2 is the variance. The coefficient $\frac{1}{\sqrt{2\pi}\sigma}$ makes the integral of the distribution be one. If we measure distance in units of the standard deviation σ from the mean, then

$$\phi(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$$

Standard tables give values of the integral

$$\int\limits_0^t \phi(x)dx$$

and from these values one can compute probability integrals for a normal distribution with mean m and variance σ^2.

General Gaussians

So far we have seen spherical Gaussian densities in $\mathbf{R}^d$. The word spherical indicates that the level curves of the density are spheres. If a random vector $\mathbf{y}$ in $\mathbf{R}^d$ has a spherical Gaussian density with zero mean, then y_i and y_j, $i \neq j$, are independent. However, in many situations the variables are correlated. To model these Gaussians, level curves that are ellipsoids rather than spheres are used.

For a random vector $\mathbf{x}$, the covariance of x_i and x_j is $E((x_i - \mu_i)(x_j - \mu_j))$. We list the covariances in a matrix called the *covariance matrix*, denoted Σ.[23] Since $\mathbf{x}$ and $\boldsymbol{\mu}$ are column vectors, $(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T$ is a $d \times d$ matrix. Expectation of a matrix or vector means componentwise expectation.

$$\Sigma = E\big((\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\big).$$

The general Gaussian density with mean $\boldsymbol{\mu}$ and positive definite covariance matrix Σ is

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right).$$

To compute the covariance matrix of the Gaussian, substitute $\mathbf{y} = \Sigma^{-1/2}(\mathbf{x} - \boldsymbol{\mu})$. Noting that a positive definite symmetric matrix has a square root:

$$\begin{aligned} E((\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T &= E(\Sigma^{1/2}\mathbf{y}\mathbf{y}^T\Sigma^{1/2}) \\ &= \Sigma^{1/2}\left(E(\mathbf{y}\mathbf{y}^T)\right)\Sigma^{1/2} = \Sigma. \end{aligned}$$

[23] Σ is the standard notation for the covariance matrix. We will use it sparingly so as not to confuse with the summation sign.

The density of $\mathbf{y}$ is the unit variance, zero mean Gaussian, thus $E(yy^T) = I$.

Bernoulli trials and the binomial distribution

A Bernoulli trial has two possible outcomes, called success or failure, with probabilities p and $1-p$, respectively. If there are n independent Bernoulli trials, the probability of exactly k successes is given by the *binomial distribution*

$$B\left(n,p\right) = \binom{n}{k} p^k (1-p)^{n-k}$$

The mean and variance of the binomial distribution $B(n,p)$ are np and $np(1-p)$, respectively. The mean of the binomial distribution is np, by linearity of expectations. The variance is $np(1-p)$ since the variance of a sum of independent random variables is the sum of their variances.

Let x_1 be the number of successes in n_1 trials and let x_2 be the number of successes in n_2 trials. The probability distribution of the sum of the successes, $x_1 + x_2$, is the same as the distribution of $x_1 + x_2$ successes in $n_1 + n_2$ trials. Thus, $B\left(n_1, p\right) + B\left(n_2, p\right) = B\left(n_1 + n_2, p\right)$.
When p is a constant, the expected degree of vertices in $G\left(n,p\right)$ increases with n. For example, in $G\left(n, \frac{1}{2}\right)$, the expected degree of a vertex is $n/2$. In many real applications, we will be concerned with $G\left(n,p\right)$ where $p = d/n$, for d a constant; i.e., graphs whose expected degree is a constant d independent of n. Holding $d = np$ constant as n goes to infinity, the binomial distribution

$$\text{Prob}\left(k\right) = \binom{n}{k} p^k \left(1-p\right)^{n-k}$$

approaches the Poisson distribution

$$\text{Prob}(k) = \frac{(np)^k}{k!} e^{-np} = \frac{d^k}{k!} e^{-d}.$$

move text beginning here to appendix
To see this, assume $k = o(n)$ and use the approximations $n - k \cong n$, $\binom{n}{k} \cong \frac{n^k}{k!}$, and $\left(1 - \frac{1}{n}\right)^{n-k} \cong e^{-1}$ to approximate the binomial distribution by

$$\lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} = \frac{n^k}{k!} \left(\frac{d}{n}\right)^k (1 - \frac{d}{n})^n = \frac{d^k}{k!} e^{-d}.$$

Note that for $p = \frac{d}{n}$, where d is a constant independent of n, the probability of the binomial distribution falls off rapidly for $k > d$, and is essentially zero for all but some finite number of values of k. This justifies the $k = o(n)$ assumption. Thus, the Poisson distribution is a good approximation.

end of material to move

Poisson distribution

The Poisson distribution describes the probability of k events happening in a unit of time when the average rate per unit of time is λ. Divide the unit of time into n segments. When n is large enough, each segment is sufficiently small so that the probability of two events happening in the same segment is negligible. The Poisson distribution gives the probability of k events happening in a unit of time and can be derived from the binomial distribution by taking the limit as $n \to \infty$.

Let $p = \frac{\lambda}{n}$. Then

$$\begin{aligned}
\text{Prob}(k \text{ successes in a unit of time}) &= \lim_{n\to\infty} \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1-\frac{\lambda}{n}\right)^{n-k} \\
&= \lim_{n\to\infty} \frac{n\,(n-1)\cdots(n-k+1)}{k!} \left(\frac{\lambda}{n}\right)^k \left(1-\frac{\lambda}{n}\right)^n \left(1-\frac{\lambda}{n}\right)^{-k} \\
&= \lim_{n\to\infty} \frac{\lambda^k}{k!} e^{-\lambda}
\end{aligned}$$

In the limit as n goes to infinity the binomial distribution $p\,(k) = \binom{n}{k} p^k \,(1-p)^{n-k}$ becomes the Poisson distribution $p\,(k) = e^{-\lambda}\frac{\lambda^k}{k!}$. The mean and the variance of the Poisson distribution have value λ. If x and y are both Poisson random variables from distributions with means λ_1 and λ_2 respectively, then $x+y$ is Poisson with mean $m_1 + m_2$. For large n and small p the binomial distribution can be approximated with the Poisson distribution.

The binomial distribution with mean np and variance $np(1-p)$ can be approximated by the normal distribution with mean np and variance $np(1-p)$. The central limit theorem tells us that there is such an approximation in the limit. The approximation is good if both np and $n(1-p)$ are greater than 10 provided k is not extreme. Thus,

$$\binom{n}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} \cong \frac{1}{\sqrt{\pi n/2}} e^{-\frac{(n/2-k)^2}{\frac{1}{2}n}}.$$

This approximation is excellent provided k is $\Theta(n)$. The Poisson approximation

$$\binom{n}{k} p^k \,(1-p)^k \cong e^{-np}\frac{(np)^k}{k!}$$

is off for central values and tail values even for $p = 1/2$. The approximation

$$\binom{n}{k} p^k \,(1-p)^{n-k} \cong \frac{1}{\sqrt{\pi p n}} e^{-\frac{(pn-k)^2}{pn}}$$

is good for $p = 1/2$ but is off for other values of p.

Generation of random numbers according to a given probability distribution

Suppose one wanted to generate a random variable with probability density $p(x)$ where $p(x)$ is continuous. Let $P(x)$ be the cumulative distribution function for x and let u be a random variable with uniform probability density over the interval [0,1]. Then the random variable $x = P^{-1}(u)$ has probability density $p(x)$.

Example: For a Cauchy density function the cumulative distribution function is

$$P(x) = \int\limits_{t=-\infty}^{x} \frac{1}{\pi}\frac{1}{1+t^2}dt = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}(x).$$

Setting $u = P(x)$ and solving for x yields $x = \tan\left(\pi\left(u - \frac{1}{2}\right)\right)$. Thus, to generate a random number $x \geq 0$ using the Cauchy distribution, generate u, $0 \leq u \leq 1$, uniformly and calculate $x = \tan\left(\pi\left(u - \frac{1}{2}\right)\right)$. The value of x varies from $-\infty$ to ∞ with $x = 0$ for $u = 1/2$. ■

12.4.10 Bayes Rule and Estimators

Bayes rule

Bayes rule relates the conditional probability of A given B to the conditional probability of B given A.

$$\text{Prob}(A|B) = \frac{\text{Prob}(B|A)\,\text{Prob}(A)}{\text{Prob}(B)}$$

Suppose one knows the probability of A and wants to know how this probability changes if we know that B has occurred. Prob(A) is called the prior probability. The conditional probability Prob($A|B$) is called the posterior probability because it is the probability of A after we know that B has occurred.

The example below illustrates that if a situation is rare, a highly accurate test will often give the wrong answer.

Example: Let A be the event that a product is defective and let B be the event that a test says a product is defective. Let Prob($B|A$) be the probability that the test says a product is defective assuming the product is defective and let Prob $\left(B|\bar{A}\right)$ be the probability that the test says a product is defective if it is not actually defective.

What is the probability Prob$(A|B)$ that the product is defective if the test say it is defective? Suppose Prob$(A) = 0.001$, Prob$(B|A) = 0.99$, and Prob $\left(B|\bar{A}\right) = 0.02$. Then

$$\begin{aligned}\text{Prob}\,(B) &= \text{Prob}\,(B|A)\,\text{Prob}\,(A) + \text{Prob}\,\left(B|\bar{A}\right)\text{Prob}\,\left(\bar{A}\right)\\ &= 0.99\times 0.001 + 0.02\times 0.999\\ &= 0.02087\end{aligned}$$

and

$$\text{Prob}\,(A|B) = \frac{\text{Prob}\,(B|A)\,\text{Prob}\,(A)}{\text{Prob}\,(B)} \approx \frac{0.99\times 0.001}{0.0210} = 0.0471$$

Even though the test fails to detect a defective product only 1% of the time when it is defective and claims that it is defective when it is not only 2% of the time, the test is correct only 4.7% of the time when it says a product is defective. This comes about because of the low frequencies of defective products. ∎

The words prior, a posteriori, and likelihood come from Bayes theorem.

$$\text{a posteriori} = \frac{\text{likelihood} \ \times\ \text{prior}}{\text{normalizing constant}}$$

$$\text{Prob}\,(A|B) = \frac{\text{Prob}\,(B|A)\,\text{Prob}\,(A)}{\text{Prob}\,(B)}$$

The a posteriori probability is the conditional probability of A given B. The likelihood is the conditional probability Prob$(B|A)$.

Unbiased Estimators

Consider n samples $x_1, x_2, \ldots, x_n$ from a Gaussian distribution of mean μ and variance σ^2. For this distribution, $m = \frac{x_1+x_2+\cdots+x_n}{n}$ is an unbiased estimator of μ, which means that $E(m) = \mu$ and $\frac{1}{n}\sum\limits_{i=1}^{n}(x_i-\mu)^2$ is an unbiased estimator of σ^2. However, if μ is not known and is approximated by m, then $\frac{1}{n-1}\sum\limits_{i=1}^{n}(x_i-m)^2$ is an unbiased estimator of σ^2.

Maximum Likelihood Estimation MLE

Suppose the probability distribution of a random variable x depends on a parameter r. With slight abuse of notation, since r is a parameter rather than a random variable, we denote the probability distribution of x as $p\,(x|r)$. This is the likelihood of observing x if r was in fact the parameter value. The job of the maximum likelihood estimator, MLE, is to find the best r after observing values of the random variable x. The likelihood of r being the parameter value given that we have observed x is denoted $L(r|x)$. This is again not a probability since r is a parameter, not a random variable. However, if we were to apply Bayes' rule as if this was a conditional probability, we get

$$L(r|x) = \frac{\text{Prob}(x|r)\text{Prob}(r)}{\text{Prob}(x)}.$$

Now, assume Prob(r) is the same for all r. The denominator Prob(x) is the absolute probability of observing x and is independent of r. So to maximize $L(r|x)$, we just maximize Prob$(x|r)$. In some situations, one has a prior guess as to the distribution Prob(r). This is then called the "prior" and in that case, we call Prob$(x|r)$ the posterior which we try to maximize.

Example: Consider flipping a coin 100 times. Suppose 62 heads and 38 tails occur. What is the most likely value of the probability of the coin to come down heads when the coin is flipped? In this case, it is $r = 0.62$. The probability that we get 62 heads if the unknown probability of heads in one trial is r is

$$\text{Prob}\,(62\ \ \text{heads}|r) = \binom{100}{62} r^{62}(1-r)^{38}.$$

This quantity is maximized when $r = 0.62$. To see this take the logarithm, which as a function of r is $\ln \binom{100}{62} + 62\ln r + 38\ln(1-r)$. The derivative with respect to r is zero at $r = 0.62$ and the second derivative is negative indicating a maximum. Thus, $r = 0.62$ is the maximum likelihood estimator of the probability of heads in a trial. ■

12.4.11 Tail Bounds and Chernoff inequalities

Markov's inequality bounds the probability that a nonnegative random variable exceeds a value a.

$$p(x \geq a) \leq \frac{E(x)}{a}.$$

or

$$p\left(x \geq aE(x)\right) \leq \frac{1}{a}$$

If one also knows the variance, σ^2, then using Chebyshev's inequality one can bound the probability that a random variable differs from its expected value by more than a standard deviations.

$$p(|x-m| \geq a\sigma) \leq \frac{1}{a^2}$$

If a random variable s is the sum of n independent random variables $x_1, x_2, \ldots, x_n$ of finite variance, then better bounds are possible. For any $\delta > 0$,

$$\text{Prob}(s > (1+\delta)m) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^m$$

and for $0 < \gamma \leq 1$,

$$\text{Prob}\left(s < (1-\gamma)m\right) < \left[\frac{e^{-\gamma}}{(1+\gamma)^{(1+\gamma)}}\right]^m < e^{-\frac{\gamma^2 m}{2}}$$

Chernoff inequalities

Chebyshev's inequality bounds the probability that a random variable will deviate from its mean by more than a given amount. Chebyshev's inequality holds for any probability distribution. For some distributions we can get much tighter bounds. For example, the probability that a Gaussian random variable deviates from its mean falls off exponentially with the distance from the mean. Here we shall be concerned with the situation where we have a random variable that is the sum of n independent random variables. This is another situation in which we can derive a tighter bound than that given by the Chebyshev inequality. We consider the case where the n independent variables are binomial but similar results can be shown for independent random variables from any distribution that has a finite variance.

Let $x_1, x_2, \ldots, x_n$ be independent random variables where

$$x_i = \begin{cases} 0 & \text{Prob } 1-p \\ 1 & \text{Prob } \quad p \end{cases} .$$

Consider the sum $s = \sum\limits_{i=1}^{n} x_i$. Here the expected value of each x_i is p and by linearity of expectation, the expected value of the sum is $m{=}np$. Theorem 2.9 bounds the probability that the sum s exceeds $(1+\delta)\, m$.

Theorem 12.3 *For any* $\delta > 0$, *Prob* $\big(s > (1+\delta)m\big) < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{m}$

Proof: For any $\lambda > 0$, the function $e^{\lambda x}$ is monotone. Thus,

$$\text{Prob}\left(s > (1+\delta)m\right) = \text{Prob}\left(e^{\lambda s} > e^{\lambda(1+\delta)m}\right).$$

$e^{\lambda x}$ is nonnegative for all x, so we can apply Markov's inequality to get

$$\text{Prob}\left(e^{\lambda s} > e^{\lambda(1+\delta)m}\right) \le e^{-\lambda(1+\delta)m} E\left(e^{\lambda s}\right).$$

Since the x_i are independent,

$$\begin{aligned} E\left(e^{\lambda s}\right) &= E\left(e^{\lambda \sum\limits_{i=1}^{n} x_i}\right) = E\left(\prod_{i=1}^{n} e^{\lambda x_i}\right) = \prod_{i=1}^{n} E\left(e^{\lambda x_i}\right) \\ &= \prod_{i=1}^{n}\left(e^{\lambda} p + 1 - p\right) = \prod_{i=1}^{n}\left(p(e^{\lambda}-1)+1\right). \end{aligned}$$

Using the inequality $1 + x < e^{x}$ with $x = p(e^{\lambda} - 1)$ yields

$$E\left(e^{\lambda s}\right) < \prod_{i=1}^{n} e^{p(e^{\lambda}-1)}.$$

Thus, for all $\lambda > 0$

$$\begin{aligned}
\text{Prob}\left(s > (1+\delta)m\right) &\leq \text{Prob}\left(e^{\lambda s} > e^{\lambda(1+\delta)m}\right) \\
&\leq e^{-\lambda(1+\delta)m} E\left(e^{\lambda s}\right) \\
&\leq e^{-\lambda(1+\delta)m} \prod_{i=1}^{n} e^{p(e^{\lambda}-1)}.
\end{aligned}$$

Setting $\lambda = \ln(1+\delta)$

$$\begin{aligned}
\text{Prob}\left(s > (1+\delta)m\right) &\leq \left(e^{-\ln(1+\delta)}\right)^{(1+\delta)m} \prod_{i=1}^{n} e^{p(e^{\ln(1+\delta)}-1)} \\
&\leq \left(\frac{1}{1+\delta}\right)^{(1+\delta)m} \prod_{i=1}^{n} e^{p\delta} \\
&\leq \left(\frac{1}{(1+\delta)}\right)^{(1+\delta)m} e^{np\delta} \\
&\leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{m}.
\end{aligned}$$

■

To simplify the bound of Theorem 12.3, observe that

$$(1+\delta)\ln(1+\delta) = \delta + \frac{\delta^2}{2} - \frac{\delta^3}{6} + \frac{\delta^4}{12} - \cdots.$$

Therefore

$$(1+\delta)^{(1+\delta)} = e^{\delta + \frac{\delta^2}{2} - \frac{\delta^3}{6} + \frac{\delta^4}{12} - \cdots}$$

and hence

$$\tfrac{e^{\delta}}{(1+\delta)^{(1+\delta)}} = e^{-\frac{\delta^2}{2} + \frac{\delta^3}{6} - \cdots}.$$

Thus, the bound simplifies to

$$\text{Prob}\left(s < (1+\delta)\,m\right) \leq e^{-\frac{\delta^2}{2}m + \frac{\delta^3}{6}m - \cdots}.$$

For small δ the probability drops exponentially with δ^2.

When δ is large another simplification is possible. First

$$\text{Prob}\left(s > (1+\delta)\,m\right) \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{m} \leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)m}$$

If $\delta > 2e-1$, substituting $2e-1$ for δ in the denominator yields

$$\text{Prob}(s > (1+\delta)\, m) \le 2^{-(1+\delta)m}.$$

Theorem 12.3 gives a bound on the probability of the sum being greater than the mean. We now bound the probability that the sum will be less than its mean.

Theorem 12.4 *Let* $0 < \gamma \le 1$*, then* $\Pr ob\big(s < (1-\gamma)m\big) < \left(\frac{e^{-\gamma}}{(1+\gamma)^{(1+\gamma)}}\right)^m < e^{-\frac{\gamma^2 m}{2}}$.

Proof: For any $\lambda > 0$

$$\text{Prob}\,\big(s < (1-\gamma)m\big) = \text{Prob}\,\big(-s > -(1-\gamma)m\big) = \text{Prob}\,\big(e^{-\lambda s} > e^{-\lambda(1-\gamma)m}\big)\,.$$

Applying Markov's inequality

$$\text{Prob}\,\big(s < (1-\gamma)m\big) < \frac{E(e^{-\lambda x})}{e^{-\lambda(1-\gamma)m}} < \frac{\prod\limits_{i=1}^{n} E(e^{-\lambda X_i})}{e^{-\lambda(1-\gamma)m}}.$$

Now

$$E(e^{-\lambda x_i}) = pe^{-\lambda} + 1 - p = 1 + p(e^{-\lambda} - 1) + 1.$$

Thus,

$$\text{Prob}(s < (1-\gamma)m) < \frac{\prod\limits_{i=1}^{n} [1 + p(e^{-\lambda} - 1)]}{e^{-\lambda(1-\gamma)m}}.$$

Since $1 + x < e^x$

$$\text{Prob}\,\big(s < (1-\gamma)m\big) < \frac{e^{np(e^{-\lambda}-1)}}{e^{-\lambda(1-\gamma)m}}.$$

Setting $\lambda = \ln \frac{1}{1-\gamma}$

$$\begin{aligned}\text{Prob}\,\big(s < (1-\gamma)m\big) &< \frac{e^{np(1-\gamma-1)}}{(1-\gamma)^{(1-\gamma)m}} \\ &< \left(\frac{e^{-\gamma}}{(1-\gamma)^{(1-\gamma)}}\right)^m.\end{aligned}$$

But for $0 < \gamma \le 1$, $(1-\gamma)^{(1-\gamma)} > e^{-\gamma + \frac{\gamma^2}{2}}$. To see this note that

$$\begin{aligned}(1-\gamma)\ln(1-\gamma) &= (1-\gamma)\left(-\gamma - \frac{\gamma^2}{2} - \frac{\gamma^3}{3} - \cdots\right) \\ &= -\gamma - \frac{\gamma^2}{2} - \frac{\gamma^3}{3} - \cdots + \gamma^2 + \frac{\gamma^3}{2} + \frac{\gamma^4}{3} + \cdots \\ &= -\gamma + \left(\gamma^2 - \frac{\gamma^2}{2}\right) + \left(\frac{\gamma^3}{2} - \frac{\gamma^3}{3}\right) + \cdots \\ &= -\gamma + \frac{\gamma^2}{2} + \frac{\gamma^3}{6} + \cdots \\ &\ge -\gamma + \frac{\gamma^2}{2}.\end{aligned}$$

It then follows that

$$\text{Prob}\left(s<(1-\gamma)m\right)<\left(\frac{e^{-\gamma}}{(1-\gamma)^{(1-\gamma)}}\right)^m<e^{-\frac{m\gamma^2}{2}}.$$

■

12.5 Bounds on Tail Probability

We now prove the tail inequality 2.9 used to prove the Gaussian Annulus Theorem.

Proof (Theorem 2.9): We first prove an upper bound on $E(x^r)$ for any even positive integer $r\le s$ and then use Markov's inequality as discussed earlier. Expand $(x_1+x_2+\cdots+x_n)^r$.

$$\begin{aligned}(x_1+x_2+\cdots+x_n)^r&=\sum\binom{r}{r_1,r_2,\ldots,r_n}x_1^{r_1}x_2^{r_2}\cdots x_n^{r_n}\\&=\sum\frac{r!}{r_1!r_2!\cdots r_n!}x_1^{r_1}x_2^{r_2}\cdots x_n^{r_n}\end{aligned}$$

where the r_i range over all nonnegative integers summing to r. By independence

$$E(x^r)=\sum\frac{r!}{r_1!r_2!\cdots r_n!}E(x_1^{r_1})E(x_2^{r_2})\cdots E(x_n^{r_n}).$$

If in a term, any $r_i=1$, the term is zero since $E(x_i)=0$. Assume henceforth that $(r_1,r_2,\ldots,r_n)$ runs over sets of nonzero r_i summing to r where each nonzero r_i is at least two. Let

$$J=\{(r_1,r_2,\ldots,r_n):r_i\in\{0,2,3,\ldots\}\,;\,\sum_{i=1}^n r_i=r\}.$$

Since $|E(x_i^{r_i})|\le\sigma^2 r_i!$,

$$E(x^r)\le r!\sum_{(r_1,r_2,\ldots,r_n)\in J}\sigma^{2(\text{ number of nonzero } r_i \text{ in set})}.$$

Collect terms of the summation with t nonzero r_i for $t=1,2,\ldots,r/2$. Let

$$J_t=\{(r_1,r_2,\ldots,r_n)\in J:\ \text{number of non-zero } r_i\ =t\}.$$

So,

$$E(x^r)=r!\sum_{t=1}^{r/2}|J_t|\sigma^{2t}.$$

We now bound $|J_t|$. There are $\binom{n}{t}$ subsets of $\{1,2,\ldots,n\}$ of cardinality t. Once a subset is fixed as the set of t values of i with nonzero r_i, set each of the $r_i\ge 2$. That is, allocate two to each of the r_i and then allocate the remaining $r-2t$ to the t r_i arbitrarily. The number of such allocations is just $\binom{r-2t+t-1}{t-1}=\binom{r-t-1}{t-1}$. So,

$$|J_t| \leq \binom{n}{t}\binom{r-t-1}{t-1}$$

$$E(x^r) \leq r!\sum_{t=1}^{r/2}\binom{n}{t}\binom{r-t-1}{t-1}\sigma^{2t} \leq r!\sum_t \frac{(n\sigma^2)^t}{t!}2^{r-t-1}.$$

Let $h(t)=\frac{(n\sigma^2)^t}{t!}2^{r-t-1}$. In the hypotheses of the theorem $a \leq \sqrt{2}\, n\sigma^2$ and $s \leq \frac{a^2}{4n\sigma^2}$. Thus, r is at most $n\sigma^2/2$. For $t \leq r/2$, increasing t by one, increases $h(t)$ by at least $n\sigma^2/(2t)$, which is at least two. This gives

$$E(x^r)=r!\sum_{t=1}^{r/2}h(t) \leq r!h(r/2)(1+\frac{1}{2}+\frac{1}{4}+\cdots) \leq \frac{r!}{(r/2)!}2^{r/2}(n\sigma^2)^{r/2}.$$

Applying Markov inequality,

$$\text{Prob}(|x|>a)=\text{Prob}(|x|^r>a^r) \leq \frac{r!(n\sigma^2)^{r/2}2^{r/2}}{(r/2)!a^r} \leq \left(r\frac{2n\sigma^2}{a^2}\right)^{r/2}.$$

The bound applies for any $r \leq s$. Take r to be the largest even integer less than or equal to $a^2/(6n\sigma^2)$. [By Calculus, we see that the function $f(x)=(cx)^{x/2}$ is minimized at $x=1/ec$ (just differentiate $\ln(f(x))$). So, $r=a^2/(2en\sigma^2)$ minimizes the upper bound. Our choice here replaces $2e$ by 6.] The tail probability is at most $e^{-r/2}$, which is at most $e\cdot e^{-a^2/(12n\sigma^2)} \leq 3\cdot e^{-a^2/(12n\sigma^2)}$, proving the theorem. ∎

12.6 Eigenvalues and Eigenvectors

12.6.1 Eigenvalues and Eigenvectors

Let A be an $n\times n$ real matrix. The scalar λ is called an eigenvalue of A if there exists a nonzero vector $\mathbf{x}$ satisfying the equation $A\mathbf{x}=\lambda\mathbf{x}$. The vector $\mathbf{x}$ is called the eigenvector of A associated with λ. The set of all eigenvectors associated with a given eigenvalue form a subspace as seen from the fact that if $A\mathbf{x}=\lambda\mathbf{x}$ and $A\mathbf{y}=\lambda\mathbf{y}$, then for any scalers c and d, $A(c\mathbf{x}+d\mathbf{y})=\lambda(c\mathbf{x}+d\mathbf{y})$. The equation $A\mathbf{x}=\lambda\mathbf{x}$ has a nontrivial solution only if $\det(A-\lambda I)=0$. The equation $\det(A-\lambda I)=0$ is called the *characteristic* equation and has n not necessarily distinct roots.

Matrices A and B are similar if there is an invertible matrix P such that $A=P^{-1}BP$.

Theorem 12.5 *If A and B are similar, then they have the same eigenvalues.*

Proof: Let A and B be similar matrices. Then there exists an invertible matrix P such that $A=P^{-1}BP$. For an eigenvector $\mathbf{x}$ of A with eigenvalue λ, $A\mathbf{x}=\lambda\mathbf{x}$, which implies $P^{-1}BP\mathbf{x}=\lambda\mathbf{x}$ or $B(P\mathbf{x})=\lambda(P\mathbf{x})$. So, $P\mathbf{x}$ is an eigenvector of B with the same eigenvalue λ. Since the reverse also holds, the theorem follows. ∎

Even though two similar matrices, A and B, have the same eigenvalues, their eigenvectors are in general different.

The matrix A is *diagonalizable* if A is similar to a diagonal matrix.

Theorem 12.6 *A is diagonalizable if and only if A has n linearly independent eigenvectors.*

Proof:

(only if) Assume A is diagonalizable. Then there exists an invertible matrix P and a diagonal matrix D such that $D = P^{-1}AP$. Thus, $PD = AP$. Let the diagonal elements of D be $\lambda_1, \lambda_2, \ldots, \lambda_n$ and let $\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}$ be the columns of P. Then $AP = [A\mathbf{p_1}, A\mathbf{p_2}, \ldots, A\mathbf{p_n}]$ and $PD = [\lambda_1\mathbf{p_1}, \lambda_2\mathbf{p_2}, \ldots, \lambda_n\mathbf{p_n}]$. Hence $A\mathbf{p_i} = \lambda_i\mathbf{p_i}$. That is, the λ_i are the eigenvalues of A and the $\mathbf{p_i}$ are the corresponding eigenvectors. Since P is invertible, the $\mathbf{p_i}$ are linearly independent.

(if) Assume that A has n linearly independent eigenvectors $\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Then $A\mathbf{p_i} = \lambda_i\mathbf{p_i}$ and reversing the above steps

$$AP = [A\mathbf{p_1}, A\mathbf{p_2}, \ldots, A\mathbf{p_n}] = [\lambda_1\mathbf{p_1}, \lambda_2\mathbf{p_2}, \ldots \lambda_n\mathbf{p_n}] = PD.$$

Thus, $AP = DP$. Since the $\mathbf{p_i}$ are linearly independent, P is invertible and hence $A = P^{-1}DP$. Thus, A is diagonalizable. ∎

It follows from the proof of the theorem that if A is diagonalizable and has eigenvalue λ with multiplicity k, then there are k linearly independent eigenvectors associated with λ.

A matrix P is *orthogonal* if it is invertible and $P^{-1} = P^T$. A matrix A is *orthogonally diagonalizable* if there exists an orthogonal matrix P such that $P^{-1}AP = D$ is diagonal. If A is orthogonally diagonalizable, then $A = PDP^T$ and $AP = PD$. Thus, the columns of P are the eigenvectors of A and the diagonal elements of D are the corresponding eigenvalues.

If P is an orthogonal matrix, then P^TAP and A are both representations of the same linear transformation with respect to different bases. To see this, note that if $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n}$ is the standard basis, then a_{ij} is the component of $A\mathbf{e_j}$ along the direction $\mathbf{e_i}$, namely, $a_{ij} = \mathbf{e_i}^TA\mathbf{e_j}$. Thus, A defines a linear transformation by specifying the image under the transformation of each basis vector. Denote by $\mathbf{p_j}$ the j^{th} column of P. It is easy to see that $(P^TAP)_{ij}$ is the component of $A\mathbf{p_j}$ along the direction $\mathbf{p_i}$, namely, $(P^TAP)_{ij} = \mathbf{p_i}^TA\mathbf{p_j}$. Since P is orthogonal, the $\mathbf{p_j}$ form a basis of the space and so P^TAP represents the same linear transformation as A, but in the basis $p_1, p_2, \ldots, p_n$.

Another remark is in order. Check that

$$A = PDP^T = \sum_{i=1}^{n} d_{ii}\mathbf{p_i}\mathbf{p_i}^T.$$

Compare this with the singular value decomposition where

$$A = \sum_{i=1}^{n} \sigma_i \mathbf{u_i}\mathbf{v_i}^T,$$

the only difference being that $\mathbf{u_i}$ and $\mathbf{v_i}$ can be different and indeed if A is not square, they will certainly be.

12.6.2 Symmetric Matrices

For an arbitrary matrix, some of the eigenvalues may be complex. However, for a symmetric matrix with real entries, all eigenvalues are real. The number of eigenvalues of a symmetric matrix, counting multiplicities, equals the dimension of the matrix. The set of eigenvectors associated with a given eigenvalue form a vector space. For a nonsymmetric matrix, the dimension of this space may be less than the multiplicity of the eigenvalue. Thus, a nonsymmetric matrix may not be diagonalizable. However, for a symmetric matrix the eigenvectors associated with a given eigenvalue form a vector space of dimension equal to the multiplicity of the eigenvalue. Thus, all symmetric matrices are diagonalizable. The above facts for symmetric matrices are summarized in the following theorem.

Theorem 12.7 (Real Spectral Theorem) *Let A be a real symmetric matrix. Then*

1. *The eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_n$, are real, as are the components of the corresponding eigenvectors, $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$.*
2. **(Spectral Decomposition)** *A is orthogonally diagonalizable and indeed*

$$A = VDV^T = \sum_{i=1}^{n} \lambda_i \mathbf{v_i}\mathbf{v_i}^T,$$

where V is the matrix with columns $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$, $|\mathbf{v_i}| = 1$ and D is a diagonal matrix with entries $\lambda_1, \lambda_2, \ldots, \lambda_n$.

Proof: $A\mathbf{v_i} = \lambda_i \mathbf{v_i}$ and $\mathbf{v_i}^c A\mathbf{v_i} = \lambda_i \mathbf{v_i}^c\mathbf{v_i}$. Here the c superscript means conjugate transpose. Then

$$\lambda_i = \mathbf{v_i}^c A\mathbf{v_i} = (\mathbf{v_i}^c A\mathbf{v_i})^{cc} = (\mathbf{v_i}^c A^c\mathbf{v_i})^c = (\mathbf{v_i}^c A\mathbf{v_i})^c = \lambda_i^c$$

and hence λ_i is real.

Since λ_i is real, a nontrivial solution to $(A - \lambda_i I)\,\mathbf{x} = 0$ has real components.

Let P be a real symmetric matrix such that $P\mathbf{v_1} = \mathbf{e_1}$ where $\mathbf{e_1} = (1, 0, 0, \ldots, 0)^T$ and $P^{-1} = P^T$. We will construct such a P shortly. Since $A\mathbf{v_1} = \lambda_1 \mathbf{v_1}$,

$$PAP^T\mathbf{e_1} = PAv_1 = \lambda P v_1 = \lambda_1 \mathbf{e_1}.$$

The condition $PAP^T\mathbf{e_1} = \lambda_1\mathbf{e_1}$ plus symmetry implies that $PAP^T = \begin{pmatrix} \lambda_1 & 0 \\ 0 & A' \end{pmatrix}$ where A' is $n-1$ by $n-1$ and symmetric. By induction, A' is orthogonally diagonalizable. Let Q be the orthogonal matrix with $QA'Q^T = D'$, a diagonal matrix. Q is $(n-1) \times (n-1)$. Augment Q to an $n \times n$ matrix by putting 1 in the $(1, 1)$ position and 0 elsewhere in the first row and column. Call the resulting matrix R. R is orthogonal too.

$$R\begin{pmatrix} \lambda_1 & 0 \\ 0 & A' \end{pmatrix} R^T = \begin{pmatrix} \lambda_1 & 0 \\ 0 & D' \end{pmatrix} \quad \Longrightarrow \quad RPAP^TR^T = \begin{pmatrix} \lambda_1 & 0 \\ 0 & D' \end{pmatrix}.$$

Since the product of two orthogonal matrices is orthogonal, this finishes the proof of (2) except it remains to construct P. For this, take an orthonormal basis of space containing $\mathbf{v_1}$. Suppose the basis is $\{\mathbf{v_1}, \mathbf{w_2}, \mathbf{w_3}, \ldots\}$ and V is the matrix with these basis vectors as its columns. Then $P = V^T$ will do. ∎

Theorem 12.8 (The fundamental theorem of symmetric matrices) *A real matrix A is orthogonally diagonalizable if and only if A is symmetric.*

Proof: (if) Assume A is orthogonally diagonalizable. Then there exists P such that $D = P^{-1}AP$. Since $P^{-1} = P^T$, we get

$$A = PDP^{-1} = PDP^T$$

which implies

$$A^T = (PDP^T)^T = PDP^T = A$$

and hence A is symmetric.
(only if) Already roved. ∎

Note that a nonsymmetric matrix may not be diagonalizable, it may have eigenvalues that are not real, and the number of linearly independent eigenvectors corresponding to an eigenvalue may be less than its multiplicity. For example, the matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

has eigenvalues 2, $\frac{1}{2} + i\frac{\sqrt{3}}{2}$, and $\frac{1}{2} - i\frac{\sqrt{3}}{2}$. The matrix $\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ has characteristic equation $(1 - \lambda)^2 = 0$ and thus has eigenvalue 1 with multiplicity 2 but has only one linearly independent eigenvector associated with the eigenvalue 1, namely $\mathbf{x} = c\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad c \neq 0$. Neither of these situations is possible for a symmetric matrix.

12.6.3 Relationship between SVD and Eigen Decomposition

The singular value decomposition exists for any $n \times d$ matrix whereas the eigenvalue decomposition exists only for certain square matrices. For symmetric matrices the decompositions are essentially the same.

The singular values of a matrix are always positive since they are the sum of squares of the projection of a row of a matrix onto a singular vector. Given a symmetric matrix, the eigenvalues can be positive or negative. If A is a symmetric matrix with eigenvalue decomposition $A = V_E D_E V_E^T$ and singular value decomposition $A = U_S D_S V_S^T$, what is the relationship between D_E and D_S, and between V_E and V_S, and between U_S and V_E? Observe that if A can be expressed as QDQ^T where Q is orthonormal and D is diagonal, then $AQ = QD$. That is, each column of Q is an eigenvector and the elements of D are the eigenvalues. Thus, if the eigenvalues of A are distinct, then Q is unique up to a permutation of columns. If an eigenvalue has multiplicity k, then the space spanned the k columns is unique. In the following we will use the term essentially unique to capture this situation. Now $AA^T = U_S D_S^2 U_S^T$ and $A^T A = V_S D_S^2 V_S^T$. By an argument similar to the one above, U_S and V_S are essentially unique and are the eigenvectors or negatives of the eigenvectors of A and A^T. The eigenvalues of AA^T or $A^T A$ are the squares of the eigenvalues of A. If A is not positive semi definite and has negative eigenvalues, then in the singular value decomposition $A = U_S D_S V_S$, some of the left singular vectors are the negatives of the eigenvectors. Let S be a diagonal matrix with $\pm 1's$ on the diagonal depending on whether the corresponding eigenvalue is positive or negative. Then $A = (U_S S)(S D_S) V_S$ where $U_S S = V_E$ and $S D_S = D_E$.

12.6.4 Extremal Properties of Eigenvalues

In this section we derive a min max characterization of eigenvalues that implies that the largest eigenvalue of a symmetric matrix A has a value equal to the maximum of $\mathbf{x}^T A\mathbf{x}$ over all vectors $\mathbf{x}$ of unit length. That is, the largest eigenvalue of A equals the 2-norm of A. If A is a real symmetric matrix there exists an orthogonal matrix P that diagonalizes A. Thus

$$P^T AP = D$$

where D is a diagonal matrix with the eigenvalues of A, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, on its diagonal. Rather than working with A, it is easier to work with the diagonal matrix D. This will be an important technique that will simplify many proofs.

Consider maximizing $\mathbf{x}^T A\mathbf{x}$ subject to the conditions

1. $\sum\limits_{i=1}^{n} x_i^2 = 1$

2. $\mathbf{r}_i^T \mathbf{x} = 0, \quad 1 \leq i \leq s$

where the r_i are any set of nonzero vectors. We ask over all possible sets $\{r_i|1 \leq i \leq s\}$ of s vectors, what is the minimum value assumed by this maximum.

Theorem 12.9 (Min max theorem) *For a symmetric matrix A, $\min\limits_{\mathbf{r}_1,\ldots,\mathbf{r}_s} \max\limits_{\substack{\mathbf{x} \\ \mathbf{r}_i \perp \mathbf{x}}}(\mathbf{x}^t A\mathbf{x}) = \lambda_{s+1}$ where the minimum is over all sets $\{r_1, r_2, \ldots, r_s\}$ of s nonzero vectors and the maximum is over all unit vectors x orthogonal to the s nonzero vectors.*

Proof: A is orthogonally diagonalizable. Let P satisfy $P^TP = I$ and $P^TAP = D$, D diagonal. Let $\mathbf{y} = P^T\mathbf{x}$. Then $\mathbf{x} = P\mathbf{y}$ and

$$\mathbf{x}^T A\mathbf{x} = \mathbf{y}^T P^T AP\mathbf{y} = \mathbf{y}^T D\mathbf{y} = \sum_{i=1}^{n} \lambda_i y_i^2$$

Since there is a one-to-one correspondence between unit vectors $\mathbf{x}$ and $\mathbf{y}$, maximizing $\mathbf{x}^T A\mathbf{x}$ subject to $\sum x_i^2 = 1$ is equivalent to maximizing $\sum\limits_{i=1}^{n} \lambda_i y_i^2$ subject to $\sum y_i^2 = 1$. Since $\lambda_1 \geq \lambda_i$, $2 \leq i \leq n$, $\mathbf{y} = (1,\ 0,\ \ldots,\ 0)$ maximizes $\sum\limits_{i=1}^{n} \lambda_i y_i^2$ at λ_1. Then $\mathbf{x} = P\mathbf{y}$ is the first column of P and is the first eigenvector of A. Similarly λ_n is the minimum value of $\mathbf{x}^T A\mathbf{x}$ subject to the same conditions.

Now consider maximizing $\mathbf{x}^T A\mathbf{x}$ subject to the conditions

1. $\sum x_i^2 = 1$
2. $\mathbf{r}_i^T\mathbf{x} = 0$

where the $\mathbf{r}_i$ are any set of nonzero vectors. We ask over all possible choices of s vectors what is the minimum value assumed by this maximum.

$$\min_{\mathbf{r}_1,\ldots,\mathbf{r}_s} \max_{\substack{\mathbf{x} \\ \mathbf{r}_i^T\mathbf{x}=0}} \mathbf{x}^T A\mathbf{x}$$

As above, we may work with $\mathbf{y}$. The conditions are

1. $\sum y_i^2 = 1$
2. $\mathbf{q}_i^T\mathbf{y} = 0$ where, $\mathbf{q}_i^T = \mathbf{r}_i^T P$

Consider any choice for the vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s$. This gives a corresponding set of $\mathbf{q}_i$. The $\mathbf{y}_i$ therefore satisfy s linear homogeneous equations. If we add $y_{s+2} = y_{s+3} = \cdots y_n = 0$ we have $n-1$ homogeneous equations in n unknowns $y_1, \ldots, y_n$. There is at least one solution that can be normalized so that $\sum y_i^2 = 1$. With this choice of $\mathbf{y}$

$$\mathbf{y}^T D\mathbf{y} = \sum \lambda_i y_i^2 \geq \lambda_{s+1}$$

since coefficients greater than or equal to $s+1$ are zero. Thus, for any choice of $\mathbf{r_i}$ there will be a $\mathbf{y}$ such that

$$\max_{\substack{\mathbf{y} \\ \mathbf{r}_i^T\mathbf{y}=0}} (\mathbf{y}^T P^T A P \mathbf{y}) \geq \lambda_{s+1}$$

and hence

$$\min_{\mathbf{r_1},\mathbf{r_2},\ldots,\mathbf{r_s}} \max_{\substack{\mathbf{y} \\ \mathbf{r}_i^T\mathbf{y}=0}} (\mathbf{y}^T P^T A P \mathbf{y}) \geq \lambda_{s+1}.$$

However, there is a set of s constraints for which the minimum is less than or equal to λ_{s+1}. Fix the relations to be $y_i = 0, \quad 1 \leq i \leq s$. There are s equations in n unknowns and for any $\mathbf{y}$ subject to these relations

$$\mathbf{y}^T D \mathbf{y} = \sum_{s+1}^{n} \lambda_i y_i^2 \leq \lambda_{s+1}.$$

Combining the two inequalities, $\min\max \mathbf{y}^T D \mathbf{y} = \lambda_{s+1}$. ■

The above theorem tells us that the maximum of $\mathbf{x}^T A \mathbf{x}$ subject to the constraint that $|\mathbf{x}|^2 = 1$ is λ_1. Consider the problem of maximizing $\mathbf{x}^T A \mathbf{x}$ subject to the additional restriction that $\mathbf{x}$ is orthogonal to the first eigenvector. This is equivalent to maximizing $\mathbf{y}^t P^t A P \mathbf{y}$ subject to $\mathbf{y}$ being orthogonal to (1,0,...,0), i.e. the first component of $\mathbf{y}$ being 0. This maximum is clearly λ_2 and occurs for $\mathbf{y} = (0, 1, 0, \ldots, 0)$. The corresponding $\mathbf{x}$ is the second column of P or the second eigenvector of A.

Similarly the maximum of $\mathbf{x}^T A \mathbf{x}$ for $\mathbf{p_1}^T\mathbf{x} = \mathbf{p_2}^T\mathbf{x} = \cdots \mathbf{p_s}^T\mathbf{x} = 0$ is λ_{s+1} and is obtained for $\mathbf{x} = \mathbf{p_{s+1}}$.

12.6.5 Eigenvalues of the Sum of Two Symmetric Matrices

The min max theorem is useful in proving many other results. The following theorem shows how adding a matrix B to a matrix A changes the eigenvalues of A. The theorem is useful for determining the effect of a small perturbation on the eigenvalues of A.

Theorem 12.10 *Let A and B be $n \times n$ symmetric matrices. Let C=A+B. Let α_i, β_i, and γ_i denote the eigenvalues of A, B, and C respectively, where $\alpha_1 \geq \alpha_2 \geq \ldots \alpha_n$ and similarly for β_i, γ_i. Then $\alpha_s + \beta_1 \geq \gamma_s \geq \alpha_s + \beta_n$.*

Proof: By the min max theorem we have

$$\alpha_s = \min_{\mathbf{r}_1,\ldots,\mathbf{r}_{s-1}} \max_{\substack{\mathbf{x} \\ \mathbf{r}_i \perp \mathbf{x}}} (\mathbf{x}^T A \mathbf{x}).$$

Suppose $\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_{s-1}}$ attain the minimum in the expression. Then using the min max theorem on C,

$$\begin{aligned}\gamma_s &\leq \max_{\mathbf{x} \perp \mathbf{r_1}, \mathbf{r_2}, \ldots \mathbf{r_{s-1}}} \left(\mathbf{x}^T (A+B)\mathbf{x}\right) \\ &\leq \max_{\mathbf{x} \perp \mathbf{r_1}, \mathbf{r_2}, \ldots \mathbf{r_{s-1}}} (\mathbf{x}^T A\mathbf{x}) + \max_{\mathbf{x} \perp \mathbf{r_1}, \mathbf{r_2}, \ldots \mathbf{r_{s-1}}} (\mathbf{x}^T B\mathbf{x}) \\ &\leq \alpha_s + \max_{\mathbf{x}} (\mathbf{x}^T B\mathbf{x}) \leq \alpha_s + \beta_1.\end{aligned}$$

Therefore, $\gamma_s \leq \alpha_s + \beta_1$.

An application of the result to $A = C + (-B)$, gives $\alpha_s \leq \gamma_s - \beta_n$. The eigenvalues of -$B$ are minus the eigenvalues of B and thus $-\beta_n$ is the largest eigenvalue. Hence $\gamma_s \geq \alpha_s + \beta_n$ and combining inequalities yields $\alpha_s + \beta_1 \geq \gamma_s \geq \alpha_s + \beta_n$. ∎

Lemma 12.11 *Let A and B be $n \times n$ symmetric matrices. Let C=A+B. Let α_i, β_i, and γ_i denote the eigenvalues of A, B, and C respectively, where $\alpha_1 \geq \alpha_2 \geq \ldots \alpha_n$ and similarly for β_i, γ_i. Then $\gamma_{r+s-1} \leq \alpha_r + \beta_s$.*

Proof: There is a set of $r-1$ relations such that over all $\mathbf{x}$ satisfying the $r-1$ relationships

$$\max(\mathbf{x}^T A\mathbf{x}) = \alpha_r.$$

And a set of $s-1$ relations such that over all $\mathbf{x}$ satisfying the $s-1$ relationships

$$\max(\mathbf{x}^T B\mathbf{x}) = \beta_s.$$

Consider $\mathbf{x}$ satisfying all these $r+s-2$ relations. For any such $\mathbf{x}$

$$\mathbf{x}^T C\mathbf{x} = \mathbf{x}^T A\mathbf{x} + \mathbf{x}^T B\mathbf{x}x \leq \alpha_r + \beta_s$$

and hence over all the $\mathbf{x}$

$$\max(\mathbf{x}^T C\mathbf{x}) \leq \alpha_s + \beta_r$$

Taking the minimum over all sets of $r+s-2$ relations

$$\gamma_{r+s-1} = \min\max(\mathbf{x}^T C\mathbf{x}) \leq \alpha_r + \beta_s$$

∎

12.6.6 Norms

A set of vectors $\{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ is *orthogonal* if $\mathbf{x_i}^T\mathbf{x_j} = 0$ for $i \neq j$ and is *orthonormal* if in addition $|\mathbf{x_i}| = 1$ for all i. A matrix A is *orthonormal* if $A^T A = I$. If A is a square orthonormal matrix, then rows as well as columns are orthogonal. In other words, if A is square orthonormal, then A^T is also. In the case of matrices over the complexes, the concept of an orthonormal matrix is replaced by that of a unitary matrix. A^* is the conjugate transpose of A if $a_{ij}^* = \bar{a}_{ji}$ where a_{ij}^* is the ij^{th} entry of A^* and $\bar{a}_{ij}^*$ is the complex conjugate of the ij^{th} element of A. A matrix A over the field of complex numbers is **unitary** if $AA^* = I$.

Norms

A **norm** on $\mathbf{R}^n$ is a function $f : \mathbf{R}^n \to \mathbf{R}$ satisfying the following three axioms:

1. $f(\mathbf{x}) \geq 0$,
2. $f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$, and
3. $f(\alpha\mathbf{x}) = |\alpha| f(\mathbf{x})$.

A norm on a vector space provides a distance function where

$$\text{distance}(\mathbf{x}, \mathbf{y}) = norm(\mathbf{x} - \mathbf{y}).$$

An important class of norms for vectors is the p-norms defined for $p > 0$ by

$$|\mathbf{x}|_p = (|\mathbf{x_1}|^p + \cdots + |\mathbf{x_n}|^p)^{\frac{1}{p}}.$$

Important special cases are

$$\begin{aligned} &|\mathbf{x}|_0 \text{ the number of non zero entries} \\ &|\mathbf{x}|_1 = |x_1| + \cdots + |x_n| \\ &|\mathbf{x}|_2 = \sqrt{|x_1|^2 + \cdots + |x_n|^2} \\ &|\mathbf{x}|_\infty = \max |x_i|. \end{aligned}$$

Lemma 12.12 *For any* $1 \leq p < q$, $|\mathbf{x}|_q \leq |\mathbf{x}|_p$.

Proof:

$$|\mathbf{x}|_q^q = \sum_i |x_i|^q.$$

Let $a_i = |x_i|^q$ and $\rho = p/q$. Using Jensen's inequality (see Section 12.3) that for any nonnegative reals $a_1, a_2, \ldots, a_n$ and any $\rho \in (0, 1)$, we have $(\sum_{i=1}^n a_i)^\rho \leq \sum_{i=1}^n a_i^\rho$, the lemma is proved. ■

There are two important matrix norms, the matrix p-norm

$$||A||_p = \max_{|\mathbf{x}|=1} \|A\mathbf{x}\|_p$$

and the Frobenius norm

$$||A||_F = \sqrt{\sum_{ij} a_{ij}^2}.$$

Let $\mathbf{a_i}$ be the i^{th} column of A. Then $\|A\|_F^2 = \sum_i \mathbf{a_i}^T \mathbf{a_i} = tr\left(A^T A\right)$. A similar argument on the rows yields $\|A\|_F^2 = tr\left(AA^T\right)$. Thus, $\|A\|_F^2 = tr\left(A^T A\right) = tr\left(AA^T\right)$.
If A is symmetric and rank k

$$||A||_2^2 \leq ||A||_F^2 \leq k\,||A||_2^2\,.$$

12.6.7 Important Norms and Their Properties

Lemma 12.13 $||AB||_2 \leq ||A||_2\,||B||_2$

Proof: $||AB||_2 = \max\limits_{|\mathbf{x}|=1} |AB\mathbf{x}|$. Let $\mathbf{y}$ be the value of $\mathbf{x}$ that achieves the maximum and let $\mathbf{z} = B\mathbf{y}$. Then

$$||AB||_2 = |AB\mathbf{y}| = |A\mathbf{z}| = \left|A\frac{\mathbf{z}}{|\mathbf{z}|}\right| |\mathbf{z}|$$

But $\left|A\frac{\mathbf{z}}{|\mathbf{z}|}\right| \leq \max\limits_{|\mathbf{x}|=1} |A\mathbf{x}| = ||A||_2$ and $|\mathbf{z}| \leq \max\limits_{|\mathbf{x}|=1} |B\mathbf{x}| = ||B||_2$. Thus $||AB||_2 \leq ||A||_2\,||B||_2$. ■

Let Q be an orthonormal matrix.

Lemma 12.14 *For all* $\mathbf{x}$, $|Q\mathbf{x}| = |\mathbf{x}|$.

Proof: $|Q\mathbf{x}|_2^2 = \mathbf{x}^T Q^T Q\mathbf{x} = \mathbf{x}^T\mathbf{x} = |\mathbf{x}|_2^2$. ■

Lemma 12.15 $||QA||_2 = ||A||_2$

Proof: For all $\mathbf{x}$, $|Q\mathbf{x}| = |\mathbf{x}|$. Replacing $\mathbf{x}$ by $A\mathbf{x}$, $|QA\mathbf{x}| = |A\mathbf{x}|$ and thus $\max\limits_{|\mathbf{x}|=1} |QA\mathbf{x}| = \max\limits_{|\mathbf{x}|=1} |A\mathbf{x}|$ ■

Lemma 12.16 $||AB||_F^2 \leq ||A||_F^2\,||B||_F^2$

Proof: Let $\mathbf{a_i}$ be the i^{th} column of A and let $\mathbf{b_j}$ be the j^{th} column of B. By the Cauchy-Schwartz inequality $\left\|\mathbf{a_i}^T\mathbf{b_j}\right\| \leq \|\mathbf{a_i}\|\,\|\mathbf{b_j}\|$. Thus $||AB||_F^2 = \sum\limits_i\sum\limits_j \left|\mathbf{a_i}^T\mathbf{b_j}\right|^2 \leq \sum\limits_i\sum\limits_j \|\mathbf{a_i}\|^2\,\|\mathbf{b_j}\|^2 = \sum\limits_i \|\mathbf{a_i}\|^2 \sum\limits_j \|\mathbf{b_j}\|^2 = ||A||_F^2\,||B||_F^2$ ■

Lemma 12.17 $||QA||_F = ||A||_F$

Proof: $||QA||_F^2 = \mathrm{Tr}(A^T Q^T QA) = \mathrm{Tr}(A^T A) = ||A||_F^2$. ■

Lemma 12.18 *For real, symmetric matrix A with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$, $\|A\|_2^2 = \max(\lambda_1^2, \lambda_n^2)$ and $\|A\|_F^2 = \lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2$*

Proof: Suppose the spectral decomposition of A is PDP^T, where P is an orthogonal matrix and D is diagonal. We saw that $||P^T A||_2 = ||A||_2$. Applying this again, $||P^T AP||_2 = ||A||_2$. But, $P^T AP = D$ and clearly for a diagonal matrix D, $||D||_2$ is the largest absolute value diagonal entry from which the first equation follows. The proof of the second is analogous. ∎

If A is real and symmetric and of rank k then $||A||_2^2 \leq ||A||_F^2 \leq k\,||A||_2^2$

Theorem 12.19 $||A||_2^2 \leq ||A||_F^2 \leq k\,||A||_2^2$

Proof: It is obvious for diagonal matrices that $||D||_2^2 \leq ||D||_F^2 \leq k\,||D||_2^2$. Let $D = Q^t AQ$ where Q is orthonormal. The result follows immediately since for Q orthonormal, $||QA||_2 = ||A||_2$ and $||QA||_F = ||A||_F$. ∎

Real and symmetric are necessary for some of these theorems. This condition was needed to express $\Sigma = Q^T AQ$. For example, in Theorem 12.19 suppose A is the $n \times n$ matrix

$$A = \begin{pmatrix} 1 & 1 & \\ 1 & 1 & \\ \vdots & \vdots & 0 \\ 1 & 1 & \end{pmatrix}.$$

$||A||_2 = 2$ and $||A||_F = \sqrt{2n}$. But A is rank 2 and $||A||_F > 2\,||A||_2$ for $n > 8$.

Lemma 12.20 *Let A be a symmetric matrix. Then $\|A\|_2 = \max\limits_{|\mathbf{x}|=1} \left|\mathbf{x}^T A\mathbf{x}\right|$.*

Proof: By definition, the 2-norm of A is $\|A\|_2 = \max\limits_{|\mathbf{x}|=1} |A\mathbf{x}|$. Thus,

$$\|A\|_2 = \max_{|\mathbf{x}|=1} |A\mathbf{x}| = \max_{|\mathbf{x}|=1} \sqrt{\mathbf{x}^T A^T A\mathbf{x}} = \sqrt{\lambda_1^2} = \lambda_1 = \max_{|\mathbf{x}|=1} \left|\mathbf{x}^T A\mathbf{x}\right|$$

∎

The two norm of a matrix A is greater than or equal to the 2-norm of any of its columns. Let $\mathbf{a_u}$ be a column of A.

Lemma 12.21 $|\mathbf{a_u}| \leq \|A\|_2$

Proof: Let $\mathbf{e_u}$ be the unit vector with a 1 in position u and all other entries zero. Note $\lambda = \max\limits_{|x|=1} |Ax|$. Let $\mathbf{x} = \mathbf{e_u}$ where $\mathbf{a_u}$ is row u. Then $|\mathbf{a_u}| = |A\mathbf{e_u}| \leq \max\limits_{|\mathbf{x}|=1} |A\mathbf{x}| = \lambda$ ∎

12.6.8 Linear Algebra

Lemma 12.22 *Let A be an $n \times n$ symmetric matrix. Then* $\det(A) = \lambda_1 \lambda_2 \cdots \lambda_n$.

Proof: The $\det(A - \lambda I)$ is a polynomial in λ of degree n. The coefficient of λ^n will be ± 1 depending on whether n is odd or even. Let the roots of this polynomial be $\lambda_1, \lambda_2, \ldots, \lambda_n$. Then $\det(A - \lambda I) = (-1)^n \prod_{i=1}^{n} (\lambda - \lambda_i)$. Thus

$$\det(A) = \det(A - \lambda I)|_{\lambda=0} = (-1)^n \prod_{i=1}^{n} (\lambda - \lambda_i) \Big|_{\lambda=0} = \lambda_1 \lambda_2 \cdots \lambda_n$$

■

The trace of a matrix is defined to be the sum of its diagonal elements. That is, $\operatorname{tr}(A) = a_{11} + a_{22} + \cdots + a_{nn}$.

Lemma 12.23 $tr(A) = \lambda_1 + \ \lambda_2 + \ \cdots + \ \lambda_n$.

Proof: Consider the coefficient of λ^{n-1} in $\det(A - \lambda I) = (-1)^n \prod_{i=1}^{n} (\lambda - \lambda_i)$. Write

$$A - \lambda I = \begin{pmatrix} a_{11} - \lambda & a_{12} & \cdots \\ a_{21} & a_{22} - \lambda & \cdots \\ \vdots & \vdots & \vdots \end{pmatrix}.$$

Calculate $\det(A - \lambda I)$ by expanding along the first row. Each term in the expansion involves a determinant of size $n-1$ which is a polynomial in λ of deg $n-2$ except for the principal minor which is of deg $n-1$. Thus the term of deg $n-1$ comes from

$$(a_{11} - \lambda)(a_{22} - \lambda) \cdots (a_{nn} - \lambda)$$

and has coefficient $(-1)^{n-1}(a_{11} + a_{22} + \cdots + a_{nn})$. Now

$$\begin{aligned}(-1)^n \prod_{i=1}^{n} (\lambda - \lambda_i) &= (-1)^n (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n) \\ &= (-1)^n \left(\lambda^n - (\lambda_1 + \lambda_2 + \cdots + \lambda_n)\lambda^{n-1} + \cdots \right)\end{aligned}$$

Therefore equating coefficients $\lambda_1 + \lambda_2 + \cdots + \lambda_n = a_{11} + a_{22} + \cdots + a_{nn} = tr(A)$

Note that $(tr(A))^2 \neq tr(A^2)$. For example $A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ has trace 3, $A^2 = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$ has trace 5 $\neq$9. However $tr(A^2) = \lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2$. To see this, observe that $A^2 = (V^T D V)^2 = V^T D^2 V$. Thus, the eigenvalues of A^2 are the squares of the eigenvalues for A.

■

Alternative proof that $tr(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. Suppose the spectral decomposition of A is $A = PDP^T$. We have

$$\operatorname{tr}(A) = tr\left(PDP^T\right) = \operatorname{tr}\left(DP^TP\right) = \operatorname{tr}(D) = \lambda_1 + \lambda_2 + \cdots + \lambda_n.$$

Lemma 12.24 *If A is $n \times m$ and B is a $m \times n$ matrix, then tr(AB)=tr(BA).*

$$\operatorname{tr}(AB) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ji} = \sum_{j=1}^{n}\sum_{i=1}^{n} b_{ji}a_{ij} = \operatorname{tr}(BA)$$

Pseudo inverse

Let A be an $n \times m$ rank r matrix and let $A = U\Sigma V^T$ be the singular value decomposition of A. Let $\Sigma' = diag\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right)$ where $\sigma_1, \ldots, \sigma_r$ are the nonzero singular values of A. Then $A' = V\Sigma' U^T$ is the pseudo inverse of A. It is the unique X that minimizes $\|AX - I\|_F$.

Second eigenvector

Suppose the eigenvalues of a matrix are $\lambda_1 \geq \lambda_2 \geq \cdots$. The second eigenvalue, λ_2, plays an important role for matrices representing graphs. It may be the case that $|\lambda_n| > |\lambda_2|$.

Why is the second eigenvalue so important? Consider partitioning the vertices of a regular degree d graph $G = (V, E)$ into two blocks of equal size so as to minimize the number of edges between the two blocks. Assign value +1 to the vertices in one block and -1 to the vertices in the other block. Let $\mathbf{x}$ be the vector whose components are the ±1 values assigned to the vertices. If two vertices, i and j, are in the same block, then x_i and x_j are both +1 or both –1 and $(x_i - x_j)^2 = 0$. If vertices i and j are in different blocks then $(x_i - x_j)^2 = 4$. Thus, partitioning the vertices into two blocks so as to minimize the edges between vertices in different blocks is equivalent to finding a vector $\mathbf{x}$ with coordinates ±1 of which half of its coordinates are +1 and half of which are –1 that minimizes

$$E_{cut} = \frac{1}{4}\sum_{(i,j)\in E}(x_i - x_j)^2$$

Let A be the adjacency matrix of G. Then

$$\begin{aligned}
\mathbf{x}^TA\mathbf{x} &= \sum_{ij} a_{ij}x_ix_j = 2\sum_{edges} x_ix_j \\
&= 2 \times \begin{pmatrix}\text{number of edges}\\ \text{within components}\end{pmatrix} - 2 \times \begin{pmatrix}\text{number of edges}\\ \text{between components}\end{pmatrix} \\
&= 2 \times \begin{pmatrix}\text{total number}\\ \text{of edges}\end{pmatrix} - 4 \times \begin{pmatrix}\text{number of edges}\\ \text{between components}\end{pmatrix}
\end{aligned}$$

Maximizing $\mathbf{x}^T A\mathbf{x}$ over all $\mathbf{x}$ whose coordinates are ±1 and half of whose coordinates are +1 is equivalent to minimizing the number of edges between components.

Since finding such an $\mathbf{x}$ is computational difficult, replace the integer condition on the components of $\mathbf{x}$ and the condition that half of the components are positive and half of the components are negative with the conditions $\sum\limits_{i=1}^{n} x_i^2 = 1$ and $\sum\limits_{i=1}^{n} x_i = 0$. Then finding the optimal $\mathbf{x}$ gives us the second eigenvalue since it is easy to see that the first eigenvector Is along $\mathbf{1}$

$$\lambda_2 = \max_{\mathbf{x}\perp\mathbf{v_1}} \frac{\mathbf{x}^T A\mathbf{x}}{\sum x_i^2}$$

Actually we should use $\sum\limits_{i=1}^{n} x_i^2 = n$ not $\sum\limits_{i=1}^{n} x_i^2 = 1$. Thus $n\lambda_2$ must be greater than $2 \times \left(\begin{array}{l}\text{total number}\\ \text{of edges}\end{array}\right) - 4 \times \left(\begin{array}{l}\text{number of edges}\\ \text{between components}\end{array}\right)$ since the maximum is taken over a larger set of $\mathbf{x}$. The fact that λ_2 gives us a bound on the minimum number of cross edges is what makes it so important.

12.6.9 Distance between subspaces

Suppose S_1 and S_2 are two subspaces. Choose a basis of S_1 and arrange the basis vectors as the columns of a matrix X_1; similarly choose a basis of S_2 and arrange the basis vectors as the columns of a matrix X_2. Note that S_1 and S_2 can have different dimensions. Define the square of the distance between two subspaces by

$$dist^2(S_1, S_2) = dist^2(X_1, X_2) = ||X_1 - X_2X_2^T X_1||_F^2$$

Since $X_1 - X_2X_2^T X_1$ and $X_2X_2^T X_1$ are orthogonal

$$\|X_1\|_F^2 = \left\|X_1 - X_2X_2^T X_1\right\|_F^2 + \left\|X_2X_2^T X_1\right\|_F^2$$

and hence

$$dist^2\left(X_1, X_2\right) = \|X_1\|_F^2 - \left\|X_2X_2^T X_1\right\|_F^2.$$

Intuitively, the distance between X_1 and X_2 is the Frobenius norm of the component of X_1 not in the space spanned by the columns of X_2.

If X_1 and X_2 are 1-dimensional unit length vectors, $dist^2\left(X_1, X_2\right)$ is the sin squared of the angle between the spaces.

Example: Consider two subspaces in four dimensions

$$X_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \qquad X_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Here

$$dist^2\left(X_1,X_2\right)=\left\|\begin{pmatrix}\frac{1}{\sqrt{2}} & 0\\ 0 & \frac{1}{\sqrt{3}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}}\\ 0 & \frac{1}{\sqrt{3}}\end{pmatrix}-\begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}} & 0\\ 0 & \frac{1}{\sqrt{3}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}}\\ 0 & \frac{1}{\sqrt{3}}\end{pmatrix}\right\|_F^2$$

$$=\left\|\begin{pmatrix}0 & 0\\ 0 & 0\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}}\\ 0 & \frac{1}{\sqrt{3}}\end{pmatrix}\right\|_F^2=\frac{7}{6}$$

In essence, we projected each column vector of X_1 onto X_2 and computed the Frobenius norm of X_1 minus the projection. The Frobenius norm of each column is the sin squared of the angle between the original column of X_1 and the space spanned by the columns of X_2.

■

12.7 Generating Functions

A sequence $a_0, a_1, \ldots$, can be represented by a generating function $g(x)=\sum\limits_{i=0}^{\infty} a_i x^i$. The advantage of the generating function is that it captures the entire sequence in a closed form that can be manipulated as an entity. For example, if $g(x)$ is the generating function for the sequence $a_0, a_1, \ldots$, then $x\frac{d}{dx}g(x)$ is the generating function for the sequence $0,\ a_1,\ 2a_{2,}\ 3a_3, \ldots$ and $x^2g''(x)+xg'(x)$ is the generating function for the sequence for $0,\ a_1,\ 4a_2,\ 9a_3, \ldots$

Example: The generating function for the sequence $1, 1, \ldots$ is $\sum\limits_{i=0}^{\infty} x^i=\frac{1}{1-x}$. The generating function for the sequence $0, 1, 2, 3, \ldots$ is

$$\sum_{i=0}^{\infty} ix^i=\sum_{i=0}^{\infty} x\tfrac{d}{dx}x^i=x\tfrac{d}{dx}\sum_{i=0}^{\infty} x^i=x\tfrac{d}{dx}\tfrac{1}{1-x}=\tfrac{x}{(1-x)^2}.$$

■

Example: If A can be selected 0 or 1 times and B can be selected 0, 1, or 2 times and C can be selected 0, 1, 2, or 3 times, in how many ways can five objects be selected. Consider the generating function for the number of ways to select objects. The generating function for the number of ways of selecting objects, selecting only A's is $1+x$, only B's is $1+x+x^2$, and only C's is $1+x+x^2+x^3$. The generating function when selecting A's, B's, and C's is the product.

$$(1+x)(1+x+x^2)(1+x+x^2+x^3)=1+3x+5x^2+6x^3+5x^4+3x^5+x^6$$

The coefficient of x^5 is 3 and hence we can select five objects in three ways: ABBCC, ABCCC, or BBCCC.

■

The generating functions for the sum of random variables

Let $f(x) = \sum\limits_{i=0}^{\infty} p_i x^i$ be the generating function for an integer valued random variable where p_i is the probability that the random variable takes on value i. Let $g(x) = \sum\limits_{i=0}^{\infty} q_i x^i$ be the generating function of an independent integer valued random variable where q_i is the probability that the random variable takes on the value i. The sum of these two random variables has the generating function $f(x)g(x)$. This is because the coefficient of x^i in the product $f(x)g(x)$ is $\sum_{k=0}^{i} p_k q_{k-i}$ and this is also the probability that the sum of the random variables is i. Repeating this, the generating function of a sum of independent nonnegative integer valued random variables is the product of their generating functions.

12.7.1 Generating Functions for Sequences Defined by Recurrence Relationships

Consider the Fibonacci sequence

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ \ldots$$

defined by the recurrence relationship

$$f_0 = 0 \qquad f_1 = 1 \qquad f_i = f_{i-1} + f_{i-2} \quad i \geq 2$$

Multiply each side of the recurrence by x^i and sum from i equals two to infinity.

$$\begin{aligned}
\sum_{i=2}^{\infty} f_i x^i &= \sum_{i=2}^{\infty} f_{i-1} x^i + \sum_{i=2}^{\infty} f_{i-2} x^i \\
f_2 x^2 + f_3 x^3 + \cdots &= f_1 x^2 + f_2 x^3 + \cdots + f_0 x^2 + f_1 x^3 + \cdots \\
&= x\left(f_1 x + f_2 x^2 + \cdots\right) + x^2\left(f_0 + f_1 x + \cdots\right) \qquad (12.1)
\end{aligned}$$

Let

$$f(x) = \sum_{i=0}^{\infty} f_i x^i. \qquad (12.2)$$

Substituting (12.2) into (12.1) yields

$$\begin{aligned}
&f(x) - f_0 - f_1 x = x\left(f(x) - f_0\right) + x^2 f(x) \\
&f(x) - x = x f(x) + x^2 f(x) \\
&f(x)(1 - x - x^2) = x
\end{aligned}$$

Thus, $f(x) = \frac{x}{1-x-x^2}$ is the generating function for the Fibonacci sequence.

Note that generating functions are formal manipulations and do not necessarily converge outside some region of convergence. Consider the generating function $f\left(x\right) = \sum\limits_{i=0}^{\infty} f_i x^i = \frac{x}{1-x-x^2}$ for the Fibonacci sequence. Using $\sum\limits_{i=0}^{\infty} f_i x^i$,

$$f(1) = f_0 + f_1 + f_2 + \cdots = \infty$$

and using $f(x) = \frac{x}{1-x-x^2}$

$$f(1) = \frac{1}{1-1-1} = -1.$$

Asymptotic behavior

To determine the asymptotic behavior of the Fibonacci sequence write

$$f\left(x\right) = \frac{x}{1-x-x^2} = \frac{\frac{\sqrt{5}}{5}}{1-\phi_1 x} + \frac{-\frac{\sqrt{5}}{5}}{1-\phi_2 x}$$

where $\phi_1 = \frac{1+\sqrt{5}}{2}$ and $\phi_1 = \frac{1-\sqrt{5}}{2}$ are the reciprocals of the two roots of the quadratic $1 - x - x^2 = 0$.

Then

$$f\left(x\right) = \frac{\sqrt{5}}{5}\Big(1 + \phi_1 x + \left(\phi_1 x\right)^2 + \cdots - \big(1 + \phi_2 x + \left(\phi_2 x\right)^2 + \cdots\big)\Big).$$

Thus,

$$f_n = \frac{\sqrt{5}}{5}\left(\phi_1^n - \phi_2^n\right).$$

Since $\phi_2 < 1$ and $\phi_1 > 1$, for large n, $f_n \cong \frac{\sqrt{5}}{5}\phi_1^n$. In fact, since $f_n = \frac{\sqrt{5}}{5}\left(\phi_1^n - \phi_2^n\right)$ is an integer and $\phi_2 < 1$, it must be the case that $f_n = \left\lfloor f_n + \frac{\sqrt{5}}{2}\phi_2^n \right\rfloor$. Hence $f_n = \left\lfloor \frac{\sqrt{5}}{5}\phi_1^n \right\rfloor$ for all n.

Means and standard deviations of sequences

Generating functions are useful for calculating the mean and standard deviation of a sequence. Let z be an integral valued random variable where p_i is the probability that z equals i. The expected value of z is given by $m = \sum\limits_{i=0}^{\infty} ip_i$. Let $p(x) = \sum\limits_{i=0}^{\infty} p_i x^i$ be the generating function for the sequence $p_1, p_2, \ldots$. The generating function for the sequence $p_1, 2p_2, 3p_3, \ldots$ is

$$x\frac{d}{dx}p(x) = \sum_{i=0}^{\infty} ip_i x^i.$$

Thus, the expected value of the random variable z is $m = xp'(x)|_{x=1} = p'(1)$. If p was not a probability function, its average value would be $\frac{p'(1)}{p(1)}$ since we would need to normalize the area under p to one.

The second moment of z, is $E(z^2) - E^2(z)$ and can be obtained as follows.

$$\begin{aligned} x^2 \frac{d}{dx} p(x)\bigg|_{x=1} &= \sum_{i=0}^{\infty} i(i-1)x^i p(x)\bigg|_{x=1} \\ &= \sum_{i=0}^{\infty} i^2 x^i p(x)\bigg|_{x=1} - \sum_{i=0}^{\infty} i x^i p(x)\bigg|_{x=1} \\ &= E(z^2) - E(z). \end{aligned}$$

Thus, $\sigma^2 = E(z^2) - E^2(z) = E(z^2) - E(z) + E(z) - E^2(z) = p^{"}(1) + p'(1) - \big(p'(1)\big)^2$.

12.7.2 The Exponential Generating Function and the Moment Generating Function

Besides the ordinary generating function there are a number of other types of generating functions. One of these is the exponential generating function. Given a sequence $a_0, a_1, \ldots$, the associated *exponential generating function* is $g(x) = \sum\limits_{i=0}^{\infty} a_i \frac{x^i}{i!}$.

Moment generating functions

The k^{th} moment of a random variable x around the point b is given by $E((x-b)^k)$. Usually the word moment is used to denote the moment around the value 0 or around the mean. In the following, we use moment to mean the moment about the origin.

The *moment generating function* of a random variable x is defined by

$$\Psi(t) = E(e^{tx}) = \int\limits_{-\infty}^{\infty} e^{tx} p(x) dx$$

Replacing e^{tx} by its power series expansion $1 + tx + \frac{(tx)^2}{2!} \cdots$ gives

$$\Psi(t) = \int\limits_{-\infty}^{\infty} \left(1 + tx + \frac{(tx)^2}{2!} + \cdots\right) p(x) dx$$

Thus, the k^{th} moment of x about the origin is $k!$ times the coefficient of t^k in the power series expansion of the moment generating function. Hence, the moment generating function is the exponential generating function for the sequence of moments about the origin.

The moment generating function transforms the probability distribution $p(x)$ into a function $\Psi(t)$ of t. Note $\Psi(0) = 1$ and is the area or integral of $p(x)$. The moment generating function is closely related to the *characteristic function* which is obtained by replacing e^{tx} by e^{itx} in the above integral where $i = \sqrt{-1}$ and is related to the *Fourier* *transform* which is obtained by replacing e^{tx} by e^{-itx}.

$\Psi(t)$ is closely related to the Fourier transform and its properties are essentially the same. In particular, $p(x)$ can be uniquely recovered by an inverse transform from $\Psi(t)$. More specifically, if all the moments m_i are finite and the sum $\sum\limits_{i=0}^{\infty} \frac{m_i}{i!} t^i$ converges absolutely in a region around the origin, then $p(x)$ is uniquely determined.

The Gaussian probability distribution with zero mean and unit variance is given by $p\left(x\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$. Its moments are given by

$$u_n = \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{\infty} x^n e^{-\frac{x^2}{2}} dx$$
$$= \begin{cases} \frac{n!}{2^{\frac{n}{2}}\left(\frac{n}{2}\right)!} & \text{n even} \\ 0 & \text{n odd} \end{cases}$$

To derive the above, use integration by parts to get $u_n = \left(n-1\right) u_{n-2}$ and combine this with $u_0 = 1$ and $u_1 = 0$. The steps are as follows. Let $u = e^{-\frac{x^2}{2}}$ and $v = x^{n-1}$. Then $u' = -xe^{-\frac{x^2}{2}}$ and $v' = \left(n-1\right) x^{n-2}$. Now $uv = \int u'v + \int uv'$ or

$$e^{-\frac{x^2}{2}} x^{n-1} = \int x^n e^{-\frac{x^2}{2}} dx + \int \left(n-1\right) x^{n-2} e^{-\frac{x^2}{2}} dx.$$

From which

$$\int x^n e^{-\frac{x^2}{2}} dx = \left(n-1\right) \int x^{n-2} e^{-\frac{x^2}{2}} dx - e^{-\frac{x^2}{2}} x^{n-1}$$
$$\int\limits_{-\infty}^{\infty} x^n e^{-\frac{x^2}{2}} dx = \left(n-1\right) \int\limits_{-\infty}^{\infty} x^{n-2} e^{-\frac{x^2}{2}} dx$$

Thus, $u_n = \left(n-1\right) u_{n-2}$.

The moment generating function is given by

$$g\left(s\right) = \sum_{n=0}^{\infty} \frac{u_n s^n}{n!} = \sum_{\substack{n=0 \\ n \text{ even}}}^{\infty} \frac{n!}{2^{\frac{n}{2}} \frac{n}{2}!} \frac{s^n}{n!} = \sum_{i=0}^{\infty} \frac{s^{2i}}{2^i i!} = \sum_{i=0}^{\infty} \frac{1}{i!} \left(\frac{s^2}{2}\right)^i = e^{\frac{s^2}{2}}.$$

For the general Gaussian, the moment generating function is

$$g\left(s\right) = e^{su + \left(\frac{\sigma^2}{2}\right) s^2}$$

Thus, given two independent Gaussians with mean u_1 and u_2 and variances σ_1^2 and σ_2^2, the product of their moment generating functions is

$$e^{s(u_1+u_2) + \left(\sigma_1^2 + \sigma_2^2\right) s^2},$$

the moment generating function for a Gaussian with mean $u_1 + u_2$ and variance $\sigma_1^2 + \sigma_2^2$. Thus, the convolution of two Gaussians is a Gaussian and the sum of two random variables that are both Gaussian is a Gaussian random variable.

12.8 Miscellaneous

12.8.1 Lagrange multipliers

Lagrange multipliers are used to convert a constrained optimization problem into an unconstrained optimization. Suppose we wished to maximize a function $f(\mathbf{x})$ subject to a constraint $g(\mathbf{x}) = c$. The value of $f(\mathbf{x})$ along the constraint $g(\mathbf{x}) = c$ might increase for a while and then start to decrease. At the point where $f(\mathbf{x})$ stops increasing and starts to decrease, the contour line for $f(\mathbf{x})$ is tangent to the curve of the constraint $g(\mathbf{x}) = c$. Stated another way the gradient of $f(\mathbf{x})$ and the gradient of $g(\mathbf{x})$ are parallel.

By introducing a new variable λ we can express the condition by $\nabla_{\mathbf{x}} f = \lambda \nabla_{\mathbf{x}} g$ and $g = c$. These two conditions hold if and only if

$$\nabla_{\mathbf{x}\lambda} \big(f\left(\mathbf{x}\right) + \lambda \left(g\left(\mathbf{x}\right) - c \right) \big) = 0$$

The partial with respect to λ establishes that $g(\mathbf{x}) = c$. We have converted the constrained optimization problem in x to an unconstrained problem with variables $\mathbf{x}$ and λ.

12.8.2 Finite Fields

For a prime p and integer n there is a unique finite field with p^n elements. In Section 4.6 we used the field GF(2^n), which consists of polynomials of degree less than n with coefficients over the field GF(2). In GF(2^8)

$$(x^7 + x^5 + x) + (x^6 + x^5 + x^4) = x^7 + x^6 = x^4 = x$$

Multiplication is modulo an irreducible polynomial. Thus

$$\begin{aligned}(x^7 + x^5 + x)(x^6 + x^5 + x^4) &= x^{13} + x^{12} + x^{11} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^5 \\ &= x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5 \\ &= x^6 + x^4 + x^3 + x^2 \qquad \text{mod } x^8 + x^4 + x^3 + x + 1\end{aligned}$$

Division of $x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5$ by $x^6 + x^4 + x^3 + x^2$ is illustrated below.

$$\begin{array}{rccccccccccc}
 & x^{13} & +x^{12} & +x^{10} & +x^9 & & +x^7 & +x^6 & +x^5 & & & \\
-x^5(x^8 + x^4 + x^3 + x^2 + 1) = & x^{13} & & & +x^9 & +x^8 & & +x^6 & +x^5 & & & \\
\hline
 & & x^{12} & +x^{10} & & +x^8 & +x^7 & & & & & \\
-x^4(x^8 + x^4 + x^3 + x^2 + 1) = & & x^{12} & & & +x^8 & +x^7 & & +x^5 & +x^4 & & \\
\hline
 & & & x^{10} & & & & & +x^5 & x^4 & & \\
-x^2(x^8 + x^4 + x^3 + x^2 + 1) = & & & x^{10} & & & & x^6 & +x^5 & & x^3 & x^2 \\
\hline
 & & & & & & & x^6 & & +x^4 & +x^3 & +x^2
\end{array}$$

12.8.3 Hash Functions

Universal Hash Families

ADD PARAGRAPH ON MOTIVATION integrate material with Chapter

Let $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$ where $m \geq n$. A family of hash functions $H = \{h | h : M \to N\}$ is said to be 2-universal if for all x and y, $x \neq y$, and for h chosen uniformly at random from H,

$$Prob\left[h\left(x\right) = h\left(y\right)\right] \leq \frac{1}{n}$$

Note that if H is the set of all possible mappings from M to N, then H is 2-universal. In fact $Prob\left[h\left(x\right) = h\left(y\right)\right] = \frac{1}{n}$. The difficulty in letting H consist of all possible functions is that a random h from H has no short representation. What we want is a small set H where each $h \in H$ has a short representation and is easy to compute.

Note that for a 2-universal H, for any two elements x and y, $h(x)$ and $h(y)$ behave as independent random variables. For a random f and any set X the set $\{f\left(x\right) | x \in X\}$ is a set of independent random variables.

12.8.4 Application of Mean Value Theorem

The mean value theorem states that if $f(x)$ is continuous and differentiable on the interval $[a, b]$, then there exists c, $a \leq c \leq b$ such that $f'(c) = \frac{f(b)-f(a)}{b-a}$. That is, at some point between a and b the derivative of f equals the slope of the line from $f(a)$ to $f(b)$. See Figure 12.8.4.

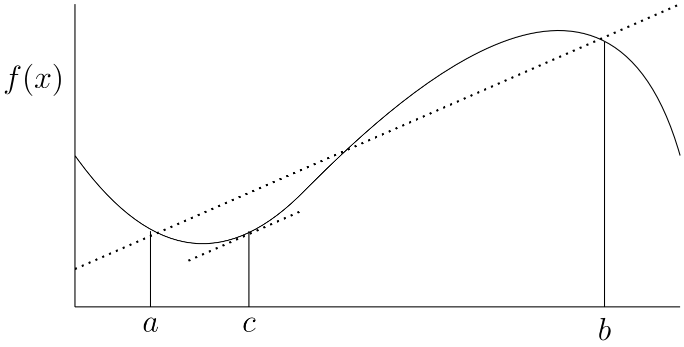


Figure 12.3: Illustration of the mean value theorem.

One application of the mean value theorem is with the Taylor expansion of a function. The Taylor expansion about the origin of $f(x)$ is

$$f(x) = f(0) + f'(0)x + \frac{1}{2!}f''(0)x^2 + \frac{1}{3!}f'''(0)x^3 + \cdots \tag{12.3}$$

By the mean value theorem there exists c, $0 \leq c \leq x$, such that $f'(c) = \frac{f(x)-f(0)}{x}$ or $f(x) - f(0) = xf'(c)$. Thus

$$xf'(c) = f'(0)x + \frac{1}{2!}f''(0)x^2 + \frac{1}{3!}f'''(0)x^3 + \cdots$$

and

$$f(x) = f(0) + xf'(c).$$

One could apply the mean value theorem to $f'(x)$ in

$$f'(x) = f'(0) + f''(0)x + \frac{1}{2!}f'''(0)x^2 + \cdots$$

Then there exists d, $0 \leq d \leq x$ such that

$$xf''(d) = f''(0)x + \frac{1}{2!}f'''(0)x^2 + \cdots$$

Integrating

$$\frac{1}{2}x^2 f''(d) = \frac{1}{2!}f''(0)x + \frac{1}{3!}f'''(0)x^3 + \cdots$$

Substituting into Eq(12.3)

$$f(x) = f(0) + f'(0)x + \frac{1}{2}x^2 f''(d).$$

12.8.5 Sperner's Lemma

Consider a triangulation of a 2-dimensional simplex. Let the vertices of the simplex be colored R, B, and G. If the vertices on each edge of the simplex are colored only with the two colors at the endpoints then the triangulation must have a triangle whose vertices are three different colors. In fact, it must have an odd number of such vertices. A generalization of the lemma to higher dimensions also holds.

Create a graph whose vertices correspond to the triangles of the triangulation plus an additional vertex corresponding to the outside region. Connect two vertices of the graph by an edge if the triangles corresponding to the two vertices share a common edge that is color R and B. The edge of the original simplex must have an odd number of such triangular edges. Thus, the outside vertex of the graph must be of odd degree. The graph must have an even number of odd degree vertices. Each odd vertex is of degree 0, 1, or 2. The vertices of odd degree, i.e. degree one, correspond to triangles which have all three colors.

12.8.6 Prüfer

Here we prove that the number of labeled trees with n vertices is n^{n-2}. By a labeled tree we mean a tree with n vertices and n distinct labels, each label assigned to one vertex.

Theorem 12.25 *The number of labeled trees with n vertices is n^{n-2}.*

Proof: (Prüfer sequence) There is a one-to-one correspondence between labeled trees and sequences of length $n-2$ of integers between 1 and n. An integer may repeat in the sequence. The number of such sequences is clearly n^{n-2}. Although each vertex of the tree has a unique integer label the corresponding sequence has repeating labels. The reason for this is that the labels in the sequence refer to interior vertices of the tree and the number of times the integer corresponding to an interior vertex occurs in the sequence is related to the degree of the vertex. Integers corresponding to leaves do not appear in the sequence.

To see the one-to-one correspondence, first convert a tree to a sequence by deleting the lowest numbered leaf. If the lowest numbered leaf is i and its parent is j, append j to the tail of the sequence. Repeating the process until only two vertices remain yields the sequence. Clearly a labeled tree gives rise to only one sequence.

It remains to show how to construct a unique tree from a sequence. The proof is by induction on n. For $n = 1$ or 2 the induction hypothesis is trivially true. Assume the induction hypothesis true for $n-1$. Certain numbers from 1 to n do not appear in the sequence and these numbers correspond to vertices that are leaves. Let i be the lowest number not appearing in the sequence and let j be the first integer in the sequence. Then i corresponds to a leaf connected to vertex j. Delete the integer j from the sequence. By the induction hypothesis there is a unique labeled tree with integer labels $1, \ldots, i-1, i+1, \ldots, n$. Add the leaf i by connecting the leaf to vertex j. We need to argue that no other sequence can give rise to the same tree. Suppose some other sequence did. Then the i^{th} integer in the sequence must be j. By the induction hypothesis the sequence with j removed is unique.

```
Algorithm
    Create leaf list - the list of labels not appearing in the Prüfer sequence. $n$ is the
          length of the Prüfer list plus two.
    while Prüfer sequence is non empty do
    begin
          $p$ =first integer in Prüfer sequence
          $e$ =smallest label in leaf list
          Add edge $(p, e)$
          Delete e from leaf list
          Delete $p$ from Prüfer sequence
          If $p$ no longer appears in Prüfer sequence add $p$ to leaf list
    end
    There are two vertices $e$ and $f$ on leaf list, add edge $(e, f)$
```

12.9 Exercises

Exercise 12.1 *What is the difference between saying $f(n)$ is $O\left(n^3\right)$ and $f(n)$ is $o\left(n^3\right)$?*

Exercise 12.2 *If $f\left(n\right) \sim g\left(n\right)$ what can we say about $f(n) + g(n)$ and $f(n) - g(n)$?*

Exercise 12.3 *What is the difference between $\sim$ and Θ?*

Exercise 12.4 *If $f\left(n\right)$ is $O\left(g\left(n\right)\right)$ does this imply that $g\left(n\right)$ is $\Omega\left(f\left(n\right)\right)$?*

Exercise 12.5 *What is $\lim\limits_{k\to\infty}\left(\frac{k-1}{k-2}\right)^{k-2}$.*

Exercise 12.6 *Select a, b, and c uniformly at random from $[0,1]$. The probability that $b < a$ is $1/2$. The probability that c<a is $1/2$. However, the probability that both b and c are less than a is $\frac{1}{3}$ not $1/4$. Why is this? Note that the six possible permutations abc, acb, bac, cab, bca, and cba, are all equally likely. Assume that a, b, and c are drawn from the interval (0,1]. Given that $b < a$, what is the probability that $c < a$?*

Exercise 12.7 *Let $A_1, A_2, \ldots, A_n$ be events. Prove that Prob$(A_1 \cup A_2 \cup \cdots A_n) \leq \sum\limits_{i=1}^{n}$ Prob(A_i)*

Exercise 12.8 *Give an example of three random variables that are pairwise independent but not fully independent.*

Exercise 12.9 *Give examples of nonnegative valued random variables with median >> mean. Can we have median << mean?*

Exercise 12.10 *Consider n samples $x_1, x_2, \ldots, x_n$ from a Gaussian distribution of mean μ and variance σ. For this distribution $m = \frac{x_1+x_2+\cdots+x_n}{n}$ is an unbiased estimator of μ. If μ is known then $\frac{1}{n}\sum\limits_{i=1}^{n}\left(x_i-\mu\right)^2$ is an unbiased estimator of σ^2. Prove that if we approximate μ by m, then $\frac{1}{n-1}\sum\limits_{i=1}^{n}\left(x_i-m\right)^2$ is an unbiased estimator of σ^2.*

Exercise 12.11 *Given the distribution $\frac{1}{\sqrt{2\pi}3}e^{-\frac{1}{2}\left(\frac{x}{3}\right)^2}$ what is the probability that x >1?*

Exercise 12.12 $e^{-\frac{x^2}{2}}$ *has value 1 at $x = 0$ and drops off very fast as x increases. Suppose we wished to approximate $e^{-\frac{x^2}{2}}$ by a function $f(x)$ where*

$$f\left(x\right) = \left\{\begin{array}{ll} 1 & |x| \leq a \\ 0 & |x| > a \end{array}\right. .$$

What value of a should we use? What is the integral of the error between $f(x)$ and $e^{-\frac{x^2}{2}}$?

Exercise 12.13 *Given two sets of red and black balls with the number of red and black balls in each set shown in the table below.*

	red	*black*
Set 1	*40*	*60*
Set 2	*50*	*50*

Randomly draw a ball from one of the sets. Suppose that it turns out to be red. What is the probability that it was drawn from Set 1?

Exercise 12.14 *Why cannot one prove an analogous type of theorem that states* $p\,(x \le a) \le \frac{E(x)}{a}$ *?*

Exercise 12.15 *Compare the Markov and Chebyshev bounds for the following probability distributions*

1. $p(x) = \begin{cases} 1 & x = 1 \\ 0 & \textit{otherwise} \end{cases}$

2. $p(x) = \begin{cases} 1/2 & 0 \le x \le 2 \\ 0 & \textit{otherwise} \end{cases}$

Exercise 12.16 *Let* s *be the sum of* n *independent random variables* $x_1, x_2, \ldots, x_n$ *where for each* i

$$x_i = \begin{cases} 0 & \textit{Prob} \quad p \\ 1 & \textit{Prob} \quad 1-p \end{cases}$$

1. *How large must* δ *be if we wish to have* $\textit{Prob}\left(s < (1-\delta)\,m\right) < \varepsilon$*?*

2. *If we wish to have* $\textit{Prob}\left(s > (1+\delta)\,m\right) < \varepsilon$*?*

Exercise 12.17 *What is the expected number of flips of a coin until a head is reached? Assume p is probability of a head on an individual flip. What is value if p=1/2?*

Exercise 12.18 *Given the joint probability*

P(A,B)	A=0	A=1
B=0	1/16	1/8
B=1	1/4	9/16

1. *What is the marginal probability of* A*? of* B*?*

2. *What is the conditional probability of* B *given* A*?*

Exercise 12.19 *Consider independent random variables x_1, x_2, and x_3, each equal to zero with probability $\frac{1}{2}$. Let $S = x_1 + x_2 + x_3$ and let F be event that $S \in \{1, 2\}$. Conditioning on F, the variables x_1, x_2, and x_3 are still each zero with probability $\frac{1}{2.}$ Are they still independent?*

Exercise 12.20 *Consider rolling two dice A and B. What is the probability that the sum S will add to nine? What is the probability that the sum will be 9 if the roll of A is 3?*

Exercise 12.21 *Write the generating function for the number of ways of producing chains using only pennies, nickels, and dines. In how many ways can you produce 23 cents?*

Exercise 12.22 *A dice has six faces, each face of the dice having one of the numbers 1 though 6. The result of a role of the dice is the integer on the top face. Consider two roles of the dice. In how many ways can an integer be the sum of two roles of the dice.*

Exercise 12.23 *If a(x) is the generating function for the sequence $a_0, a_1, a_2, \ldots$, for what sequence is a(x)(1-x) the generating function.*

Exercise 12.24 *How many ways can one draw n $a's$ and $b's$ with an even number of $a's$.*

Exercise 12.25 *Find the generating function for the recurrence $a_i = 2a_{i-1} + i$ where $a_0 = 1$.*

Exercise 12.26 *Find a closed form for the generating function for the infinite sequence of prefect squares 1, 4, 9, 16, 25, ...*

Exercise 12.27 *Given that $\frac{1}{1-x}$ is the generating function for the sequence $1, 1, \ldots$, for what sequence is $\frac{1}{1-2x}$ the generating function?*

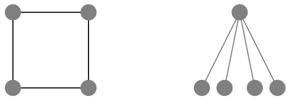
Exercise 12.28 *Find a closed form for the exponential generating function for the infinite sequence of prefect squares 1, 4, 9, 16, 25, ...*

Exercise 12.29 *Prove that the L_2 norm of $(a_1, a_2, \ldots, a_n)$ is less than or equal to the L_1 norm of $(a_1, a_2, \ldots, a_n)$.*

Exercise 12.30 *Prove that there exists a y, $0 \leq y \leq x$, such that $f(x) = f(0) + f'(y)x$.*

Exercise 12.31 *Show that the eigenvectors of a matrix A are not a continuous function of changes to the matrix.*

Exercise 12.32 *What are the eigenvalues of the two graphs shown below? What does this say about using eigenvalues to determine if two graphs are isomorphic.*



Exercise 12.33 *Let A be the adjacency matrix of an undirected graph G. Prove that eigenvalue λ_1 of A is at least the average degree of G.*

Exercise 12.34 *Show that if A is a symmetric matrix and λ_1 and λ_2 are distinct eigenvalues then their corresponding eigenvectors x_1 and x_2 are orthogonal.*
Hint:

Exercise 12.35 *Show that a matrix is rank k if and only if it has k nonzero eigenvalues and eigenvalue 0 of rank n-k.*

Exercise 12.36 *Prove that maximizing $\frac{x^TAx}{x^Tx}$ is equivalent to maximizing x^TAx subject to the condition that x be of unit length.*

Exercise 12.37 *Let A be a symmetric matrix with smallest eigenvalue $\lambda_{\min}$. Give a bound on the largest element of A^{-1}.*

Exercise 12.38 *Let A be the adjacency matrix of an n vertex clique with no self loops. Thus, each row of A is all ones except for the diagonal entry which is zero. What is the spectrum of A.*

Exercise 12.39 *Let A be the adjacency matrix of an undirect graph G. Prove that the eigenvalue λ_1 of A is at least the average degree of G.*

Exercise 12.40 *We are given the probability distribution for two random vectors x and y and we wish to stretch space to maximize the expected distance between them. Thus, we will multiply each coordinate by some quantity a_i. We restrict $\sum\limits_{i=1}^{d} a_i^2 = d$. Thus, if we increase some coordinate by $a_i > 1$, some other coordinate must shrink. Given random vectors $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$ how should we select a_i to maximize $E\left(|x-y|^2\right)$? The a_i stretch different coordinates. Assume*

$$y_i = \left\{ \begin{array}{cc} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{array} \right.$$

and that x_i has some arbitrary distribution.

$$\begin{array}{c} E\left(|x-y|^2\right) = E\sum\limits_{i=1}^{d}\left[a_i^2\left(x_i-y_i\right)^2\right] = \sum\limits_{i=1}^{d} a_i^2 E\left(x_i^2 - 2x_iy_i + y_i^2\right) \\ = \sum\limits_{i=1}^{d} a_i^2 E\left(x_i^2 - x_i + \frac{1}{2}\right) \end{array}$$

Since $E\left(x_i^2\right) = E\left(x_i\right)$ we get . Thus, weighting the coordinates has no effect assuming $\sum\limits_{i=1}^{d} a_i^2 = 1$. Why is this? Since $E\left(y_i\right) = \frac{1}{2}$.

$E\left(|x-y|^2\right)$ is independent of the value of x_i hence its distribution.

What if $y_i = \left\{ \begin{array}{cc} 0 & \frac{3}{4} \\ 1 & \frac{1}{4} \end{array} \right.$ *and* $E(y_i) = \frac{1}{4}$. *Then*

$$\begin{array}{c} E\left(|x-y|^2\right) = \sum\limits_{i=1}^{d} a_i^2 E\left(x_i^2 - 2x_iy_i + y_i^2\right) = \sum\limits_{i=1}^{d} a_i^2 E\left(x_i - \frac{1}{2}x_i + \frac{1}{4}\right) \\ = \sum\limits_{i=1}^{d} a_i^2 \left(\frac{1}{2}E(x_i) + \frac{1}{4}\right) \end{array} .$$

To maximize put all weight on the coordinate of x with highest probability of one. What if we used 1-norm instead of the two norm?

$$E(|x-y|) = E\sum_{i=1}^{d} a_i |x_i - y_i| = \sum_{i=1}^{d} a_i E|x_i - y_i| = \sum_{i=1}^{d} a_i b_i$$

where $b_i = E(x_i - y_i)$. *If* $\sum\limits_{i=1}^{d} a_i^2 = 1$, *then to maximize let* $a_i = \frac{b_i}{b}$. *Taking the dot product of a and b is maximized when both are in the same direction.*

Exercise 12.41 *Maximize x+y subject to the constraint that* $x^2 + y^2 = 1$.

Exercise 12.42 *Draw a tree with 10 vertices and label each vertex with a unique integer from 1 to 10. Construct the Prfer sequence for the tree. Given the Prfer sequence recreate the tree.*

Exercise 12.43 *Construct the tree corresponding to the following Prfer sequences*

1. *113663*
2. *552833226*

Index

References

[ABC$^+$08] Reid Andersen, Christian Borgs, Jennifer T. Chayes, John E. Hopcroft, Vahab S. Mirrokni, and Shang-Hua Teng. Local computation of pagerank contributions. *Internet Mathematics*, 5(1):23–45, 2008.

[AF] David Aldous and James Fill. *Reversible Markov Chains and Random Walks on Graphs*. http://www.stat.berkeley.edu/ aldous/RWG/book.html.

[AK] Sanjeev Arora and Ravindran Kannan. Learning mixtures of separated nonspherical gaussians. *Annals of Applied Probability*, 15(1A):6992.

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986.

[AM05] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *COLT*, pages 458–469, 2005.

[AN72] Krishna Athreya and P. E. Ney. *Branching Processes*, volume 107. Springer, Berlin, 1972.

[AP03] Dimitris Achlioptas and Yuval Peres. The threshold for random k-sat is 2^{k} (ln 2 - o(k)). In *STOC*, pages 223–231, 2003.

[Aro11] Multiplicative weights method: a meta-algorithm and its applications. *Theory of Computing journal - to appear*, 2011.

[AS08] Noga Alon and Joel H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., Hoboken, NJ, third edition, 2008. With an appendix on the life and work of Paul ErdHos.

[BA] Albert-Lszl Barabsi and Rka Albert. Emergence of scaling in random networks. *Science*, 286(5439).

[BEHW] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the Association for Computing Machinary.*

[BGG97] C Sidney Burrus, Ramesh A Gopinath, and Haitao Guo. Introduction to wavelets and wavelet transforms: a primer. 1997.

[Ble12] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.

[BMPW98] Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21:37–47, 1998.

[Bol01] Béla Bollobás. *Random Graphs.* Cambridge University Press, 2001.

[BT87] Béla Bollobás and Andrew Thomason. Threshold functions. *Combinatorica*, 7(1):35–38, 1987.

[CF86] Ming-Te Chao and John V. Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM J. Comput.*, 15(4):1106–1118, 1986.

[CGTS99] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 1–10, New York, NY, USA, 1999. ACM.

[CHK$^+$] Duncan S. Callaway, John E. Hopcroft, Jon M. Kleinberg, M. E. J. Newman, and Steven H. Strogatz. Are randomly grown graphs really random?

[Chv92] *33rd Annual Symposium on Foundations of Computer Science, 24-27 October 1992, Pittsburgh, Pennsylvania, USA.* IEEE, 1992.

[CLMW11] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11, 2011.

[DFK91] Martin Dyer, Alan Frieze, and Ravindran Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the Association for Computing Machinary*, 1991.

[DFK$^+$99] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. In *SODA*, pages 291–299, 1999.

[DG99] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. 99(006), 1999.

[DS84] Peter G. Doyle and J. Laurie Snell. *Random walks and electric networks*, volume 22 of *Carus Mathematical Monographs.* Mathematical Association of America, Washington, DC, 1984.

[DS07] Sanjoy Dasgupta and Leonard J. Schulman. A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226, 2007.

[ER60] Paul Erdös and Alfred Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

[FK99] Alan M. Frieze and Ravindan Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[Fri99] Friedgut. Sharp thresholds of graph properties and the k-sat problem. *Journal of the American Math. Soc.*, 12, no 4:1017–1054, 1999.

[FS96] Alan M. Frieze and Stephen Suen. Analysis of two simple heuristics on a random instance of k-sat. *J. Algorithms*, 20(2):312–355, 1996.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete mathematics - a foundation for computer science (2. ed.)*. Addison-Wesley, 1994.

[GvL96] Gene H. Golub and Charles F. van Loan. *Matrix computations (3. ed.)*. Johns Hopkins University Press, 1996.

[HBB10] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online learning for latent dirichlet allocation. In *NIPS*, pages 856–864, 2010.

[Jer98] Mark Jerrum. Mathematical foundations of the markov chain monte carlo method. In Dorit Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, 1998.

[JKLP93] Svante Janson, Donald E. Knuth, Tomasz Luczak, and Boris Pittel. The birth of the giant component. *Random Struct. Algorithms*, 4(3):233–359, 1993.

[JLR00] Svante Janson, Tomasz Ĺuczak, and Andrzej Ruciński. *Random Graphs*. John Wiley and Sons, Inc, 2000.

[Kan09] Ravindran Kannan. A new probability inequality using typical moments and concentration results. In *FOCS*, pages 211–220, 2009.

[Kar90] Richard M. Karp. The transitive closure of a random digraph. *Random Structures and Algorithms*, 1(1):73–94, 1990.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *JOURNAL OF THE ACM*, 46(5):604–632, 1999.

[Kle00] Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, pages 163–170, 2000.

[Kle02] Jon M. Kleinberg. An impossibility theorem for clustering. In *NIPS*, pages 446–453, 2002.

[KV95] Michael Kearns and Umesh Vazirani. *An introduction to Computational Learning Theory*. MIT Press, 1995.

[KV09] Ravi Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009.

[Liu01] Jun Liu. *Monte Carlo Strategies in Scientific Computing.* Springer, 2001.

[Mat10] Jiří Matoušek. *Geometric discrepancy*, volume 18 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2010. An illustrated guide, Revised paperback reprint of the 1999 original.

[Mit97] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[MR95a] Michael Molloy and Bruce A. Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms*, 6(2/3):161–180, 1995.

[MR95b] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[MR99] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. In *Algorithms and theory of computation handbook*, pages 15–1–15–23. CRC, Boca Raton, FL, 1999.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

[MV10] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *FOCS*, pages 93–102, 2010.

[Pal85] Edgar M. Palmer. *Graphical evolution.* Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons Ltd., Chichester, 1985. An introduction to the theory of random graphs, A Wiley-Interscience Publication.

[Par98] Beresford N. Parlett. *The symmetric eigenvalue problem*, volume 20 of *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Corrected reprint of the 1980 original.

[per10] *Markov Chains and Mixing Times.* American Mathematical Society, 2010.

[Sch90] Rob Schapire. Strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[SJ] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation.*

[Sly10] Allan Sly. Computational transition at the uniqueness threshold. In *FOCS*, pages 287–296, 2010.

[SS01] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

[Val84] Leslie G. Valiant. A theory of the learnable. In *STOC*, pages 436–445, 1984.

[Val13] L. Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World.* Basic Books, 2013.

[VC71] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[Vem04] Santosh Vempala. *The Random Projection Method.* DIMACS, 2004.

[VW02] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixtures of distributions. *Journal of Computer and System Sciences*, pages 113–123, 2002.

[Wil06] H.S. Wilf. *Generatingfunctionology.* Ak Peters Series. A K Peters, 2006.

[WS98a] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393 (6684), 1998.

[WS98b] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393, 1998.

[WW96] E. T. Whittaker and G. N. Watson. *A course of modern analysis.* Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1996. An introduction to the general theory of infinite processes and of analytic functions; with an account of the principal transcendental functions, Reprint of the fourth (1927) edition.