4.5 APPLICATIONS OF SINGULAR VALUE DECOMPOSITION

4.5.1 Principal Component Analysis

The traditional use of SVD is in Principal Component Analysis (PCA). A typical use of PCA is illustrated by customer-product data where there are n customers buying d products. One wishes to construct a matrix A where the element a_{ij} of the matrix A indicates the probability of customer i purchasing product j. Here n and d are very large, on the order of millions, and if A were a full rank matrix there is probably little one could do to estimate or even store A. One hypothesizes that there are really only k underlying basic factors (like age, income, family size, etc.) that determine a customer's purchase behavior. An individual customer's behavior is determined by some weighted combination of these underlying factors. That is, a customer's purchase behavior can be characterized by a k-dimensional vector where k is much less that n and d. The components of the vector are weights for each of the basic factors. Associated with each basic factor is a vector of probabilities, each component of which is the probability of purchasing a given product by someone whose behavior depends only on that factor. More abstractly, A is an $n \times d$ matrix that can be expressed as the product of two matrices U and V where U is an $n \times k$ matrix expressing the factor weights for each customer and V is a $k \times d$ matrix expressing the purchase probabilities of products that correspond to that factor.

The original use of PCA was given A and k find U and V. The dimensions of A were such that one could work with the full matrix. The matrix was obtained from real data and hence was noisy. Thus, one wanted to find U and V so as to minimize $\left|A - U^T V\right|_F^2$. This is done by finding the first k singular vectors of the singular value decomposition.

At present, the scale of data has increased so that A is too large to be stored. One version of the problem is given a few elements of A find U and V. This area is called collaborative filtering and one of its uses is to target an ad to a customer based on one or two purchases. Clearly if A were of full rank one could not estimate A by a few entries. However, if one has a low rank model for A it is possible to estimate A based on less information.

4.5.2 Clustering a Mixture of Spherical Gaussians

It is relatively easy to cluster points in two or three dimensions. However, clustering is not so easy in higher dimensions. Many problems have high dimensional data and clustering problems are no exception. Clustering problems tend to be NP-hard so we do not have polynomial time algorithms to solve them. For this reason, the application of stochastic models of input data to clustering problems has become important.

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 $w_1F_1 + w_2F_2 + \cdots + w_kF_k$ where $F_1, F_2, \ldots F_k$ are the basic densities and $w_1, w_2, \ldots w_k$ are positive real numbers called weights that add up to one. Clearly, $w_1F_1 + w_2F_2 + \ldots w_kF_k$ is a probability density, it integrates to one.

The model fitting problem is to fit a mixture of k basic densities to n samples, each sample drawn according to the same mixture distribution. The class of basic densities is known, but the components of the mixture are not.

Here, we only deal with the case where the basic densities are all spherical Gaussians. The samples are generated by picking an integer i from the set $\{1, 2, \ldots k\}$ with probabilities $w_1, w_2, \ldots w_k$, respectively. Then, picking a sample according to F_i and repeating the process n times. This process generates n samples according to the mixture. In this process, the set of samples is naturally partitioned into k sets, each set corresponding to an F_i.

The model-fitting problem consists of two sub problems. The first sub problem is to cluster the sample into k subsets where each subset was picked according to one component density. The second sub problem is to fit a distribution to each subset. We discuss only the clustering problem here. The problem of fitting a single Gaussian to a set of data points was discussed in Section 2.3 of Chapter 2.

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 then 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 10 times the maximum of their standard deviations, then they hardly overlap. What is the analog of this statement in higher dimensions?

For a d dimensional spherical Gaussian, with standard deviation σ in each direction[1], it is easy to see that the expected distance squared from the center is $d\sigma^2$. Define the radius r of the Gaussian to be the square root of the average distance squared from the center; so r is $\sqrt{d}\sigma$. If the inter-center separation between two spherical Gaussians, both of radius r is at least $2r = 2\sqrt{d}\sigma$, then it is easy to see that the densities hardly overlap. But this separation requirement grows with d and is too large to be useful in many cases. For a way out of such a strong requirement, imagine projecting the two Gaussians onto the one-dimensional line joining their centers. The projection is also a Gaussian with the same standard deviation as shown in the following lemma.

Lemma 4.5: Suppose F is a d-dimensional spherical Gaussian with centre μ and standard deviation σ. The density of F projected onto an arbitrary k dimensional subspace V is a spherical Gaussian with the same standard deviation.

[1] Since a spherical Gaussian has the same standard deviation in every direction, We call it the standard deviation of the Gaussian henceforth.

Proof: Since *F* is spherical, the projection is independent of the k dimensional subspace. Pick *V* to be the subspace spanned by the first *k* coordinate vectors. For a point $x=(x_1,x_2,\ldots x_d)$, we will use the notation $x'=\left(x_{k+1},x_{k+2},\cdots,x_n\right)$ and $x''=(x_1,x_2,\ldots x_k)$. The density of the projected Gaussian at the point $\left(x_1,x_2,\cdots,x_k\right)$ is

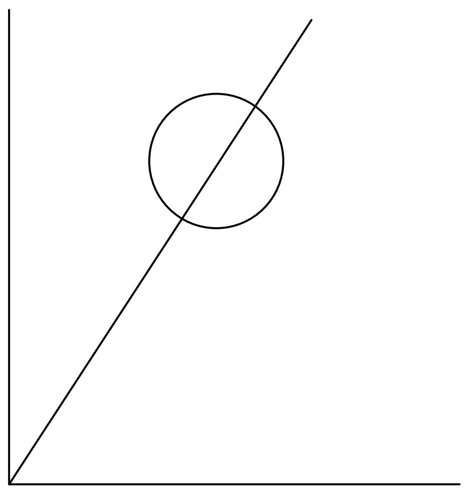
$$ce^{-\frac{|x''-\mu''|^2}{2\sigma^2}}\int_{x'} e^{-\frac{|x'-\mu'|^2}{2\sigma^2}}dx'=c'e^{-\frac{|x''-\mu''|^2}{2\sigma^2}}.$$

This clearly implies the lemma.

■

Once projected onto the line joining the two centers, the two Gaussians are just one-dimensional Gaussians with standard deviation σ and so a separation distance of a certain number of σ (independent of *d*) should suffice. This leads to the question how to find the space spanned by the two centers from the samples. More generally, if there are *k* Gaussians, how does one find the *k* dimensional subspace spanned by the *k* centers? A simple argument shows that the best-fit subspace to the date points gives the answer.

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, the best fit 1-dimensional subspace is the line though the center and the origin. Next we show that the best fit *k*-dimensional subspace for a single Gaussian is any *k* dimensional subspace containing the line through its 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. 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. 3. The best fit k-dimensional subspace for k Gaussians is the subspace containing their centers.

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

Defintion 4.1: If *F* is a probability density in *d* space, the best fit line for *F* is the line *l* passing through the origin which minimizes the expected squared (perpendicular) distance to the line, namely,

$$\int \text{dist}(x,l)^2 F(x)\,dx.$$

■

Recall that a *k* dimensional subspace is the best-fit subspace if the sum of squared distances to it is minimized or equivalently, the sum of squared lengths of projections onto it is maximized. This was defined for a set of points, but again it can be extended to a density as above:

Definition: If F is a probability density in *d*-space and *V* is a sub-space, then the expected squared perpendicular distance of *V* to F, denoted $f(V,F)$, is given by

$$f(V,F) = \int \text{dist}^2(x,V)\,F(x)\,dx,$$

where $\text{dist}(x,V)$ denotes the perpendicular distance from x to V.

For the uniform density on the unit circle centered at the origin, it is easy to see that any line passing through the origin is a best fit line for the probability distribution.

Lemma 4.6: Let the probability density *F* be a spherical Gaussian with centre μ. The best fit 1-dimensional subspace is the line passing through μ and the origin.

Proof: For a randomly chosen ***x*** (according to *F*) and a fixed unit length vector v,

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

since $E\left[\left(v^T(x-\mu)\right)^2\right]$ is the variance in the direction ***v*** and $E\left(v^T(x-\mu)\right)=0$. The lemma follows from the fact that the best fit line ***v*** is the one that maximizes $E\left(\left(v^T x\right)^2\right)$ which is maximized when ***v*** is aligned with the center μ.

■

Lemma 4.7: For a spherical Gaussian with centre μ, a *k* dimensional subspace is a best fit subspace if and only if it contains μ.

Proof: By symmetry, it is clear that every *k* dimensional subspace through μ has the same sum of distances squared to the density. Now by the SVD procedure, we know that the best-fit *k* dimensional subspace contains the best fit line, i.e., contains μ. Thus, the lemma follows.

■

This immediately leads to the following theorem.

Theorem 4.6: If F is a mixture of *k* spherical Gaussians whose centers span a *k* dimensional subspace, then its unique best fit *k* dimensional subspace is the one containing the centers.

Proof: Let F= $w_1F_1 + w_2F_2 + \cdots + w_kF_k$. Let *V* be any sub-space of dimension *k* or less. Then, we have

$$\begin{aligned} f(V,F) &= \int \text{dist}^2(x,V)F(x)dx \\ &= \sum_{i=1}^{k} w_i \int \text{dist}^2(x,V)F_i(x)dx \\ &\geq \sum_{i=1}^{k} w_i \text{ (distance squared of } \text{F}_i \text{ to its best fit subspace)} \end{aligned}$$

Now if we choose V to be the space spanned by the centers of the densities F_i , then the last inequality becomes an equality by the previous lemma proving the theorem.

■

If we had an infinite set of points drawn according to the mixture, then the *k* dimensional SVD subspace gives us exactly the space of the centers. In reality, we have only a large number of samples drawn according to the mixture. 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.

4.3 An Application of SVD to a Discrete Optimization Problem

In the last example, SVD was used as a dimension reduction technique. It found a *k*-dimensional sub-space of a *d*-dimensional space (the space of centers) 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 examples. 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 node set V of the 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 if $i \in S$ and 0 if $i \in \bar{S}$. The vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ 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 max-cut problem can be posed as the optimization problem

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

In matrix notation,

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

where $\mathbf{1}$ denotes the vector of all 1's. So, we can restate the problem as

$$\text{Maximize } \mathbf{x}^T A(1-\mathbf{x}) \text{ subject to } \quad x_i \in \{0,1\} \qquad \text{Eq. 4.1}$$

The SVD is used to solve this problem approximately by computing the SVD of A and replacing A by $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ in Eq. 4.1 to get

$$\text{Maximize } \mathbf{x}^T A_k(1-\mathbf{x}) \quad \text{subject to} \quad x_i \in \{0,1\} \qquad \text{Eq. 4.2}$$

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

We will show:

(i) For each 0-1 vector $\mathbf{x}$, $\mathbf{x}^T A_k(1-\mathbf{x})$ and $\mathbf{x}^T A(1-\mathbf{x})$ differ by at most $\frac{n^2}{\sqrt{k+1}}$. Thus, the maxima in Eq. 4.1 and Eq. 4.2 differ by at most this amount.

(ii) A near optimal $\mathbf{x}$ for Eq. 4.2 can be found by exploiting the low rank of A_k, which by (i) is near optimal for Eq. 4.1. where near Optimal means with (additive) error of at most $\frac{n^2}{\sqrt{k+1}}$.

First, we prove (i). Since $\mathbf{x}$ and $\mathbf{1}-\mathbf{x}$ are 0-1 n-vectors, each has length at most $\sqrt{n}$. By the definition of 2-norm, $|(A-A_k)(\mathbf{1}-\mathbf{x})| \le \sqrt{\mathbf{n}} \, \| A-A_k \|_2$. 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 **???** $\| A-A_k \|_2 = \sigma_{k+1}(A)$. Also,

$$\begin{aligned} \sigma_1^2+\sigma_2^2+\ldots\sigma_{k+1}^2 &\le \| A \|_F^2 \\ \sigma_1^2+\sigma_2^2+\ldots\sigma_{k+1}^2 &\ge (k+1)\sigma_{k+1}^2 \\ \| A \|_F^2 = \sum_{i,j} a_{ij}^2 &\le n^2 \end{aligned}$$

imply that $\| A-A_k \|_2 \le \dfrac{n}{\sqrt{k+1}}$ proving (i).

Next we focus on item (ii). 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 u and v are 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 Max Cut problem approximately.

Item (ii) is solved by dynamic programming. A sub problem consists of maximizing the sum restricted to the first j coordinates of u and v. Start with $j=1$ and increase the j one at a time. As j increases, we compute

$$\max_{S \subseteq V} \sum_{i=1}^{k} \sigma_i \mathbf{u_i}(S)\mathbf{v_i}(\bar{S})$$

where $\mathbf{u_i}(S)$ denotes the sum of the first j coordinates of the vector $\mathbf{u_i}$ that belong to S and $\mathbf{v_i}(\bar{S})$ denotes the sum of the first j coordinates of the vector $\mathbf{v_i}$ that belong to $\bar{S}$. To do this, it suffices to find the pairs of values of $(\mathbf{u}_i(S), \mathbf{v}_i(\bar{S}))$ for all possible S to some granularity, i.e., we just need these values to the nearest integer multiple of some δ, which will be specified later. When $j=1$, there are only two pairs of values, one where the coordinate is in S and one where it is in $\bar{S}$. Each time the j is increased by one, each element in the earlier level can add at most two elements depending on whether the new coordinate is added to S or not. We try both possibilities and round each coordinate to the nearest multiple of δ. Then we prune redundant pairs. With $\delta \le O(\sqrt{n}/\sqrt{k})$, each rounded $\mathbf{u_i}(S)$ can have at most $O(\sqrt{k})$ possible values. So in all, we have at most $O(k^k)$ possible pairs.

It is not difficult to show (Exercise) that the running time of the whole algorithm is only $O(nk^k)$. It is exponential in k, but polynomial in n. We summarize what we have accomplished:

Theorem 4.7: Given a directed graph $G(V,E)$, we can compute a cut in it of size at least the maximum cut minus $\frac{2n^2}{\sqrt{k}}$ in time k^k times a polynomial in n.

■

It would be quite a surprise to have an algorithm which actually achieves the same accuracy in time polynomial in *n* and *k* because then, we could get an exact max cut in polynomial time. (Why?)

4.4 SVD as a Compression Algorithm

Suppose A is the pixel intensity matrix of a large image. The entry a_{ij} gives the intensity of the ij^{th} pixel. If A is $n \times n$, the transmission of A requires transmitting $O(n^2)$ real numbers. Instead, one could send A_k, that is, the top *k* singular values $\sigma_1, \sigma_2, \ldots \sigma_k$ along with the left and right singular vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, and $\mathbf{v_1}, \mathbf{v}_2, \ldots, \mathbf{v_k}$. This would require sending $O(kn)$ real numbers instead of $O(n^2)$ real numbers. If *k* is much smaller than *n*, this results in savings. Indeed, it has been argued that for many images, a *k* much smaller than *n* is sufficient to (visually) reconstruct the image. Thus, one could use SVD as a compression method.

It turns out that for pictures there is a fixed basis so that the top several hundred singular vectors is sufficient to represent any picture approximately. This means that the space spanned by the top several hundred singular vectors is not too different than the space spanned by the top two hundred singular vectors of a given picture. Thus, compressing pictures by this standard basis can save substantially since we can transmit the standard basis once and whenever we want to transmit a picture we send only the top several hundred singular values for the standard basis.

Eigenvalues, Eigenvectors and the Spectral Decomposition

John: I propose that we postpone writing this section for now until we see what all we need of eigenvalues and eigenvectors. [They are needed for Markov Chains and a bunch of other things..So for now, I have not revised this.]

Let B be a square matrix. If the Vector x and scalar λ are such that $Bx = \lambda x$, then x is an *eigenvector* of the matrix B and λ is the corresponding *eigenvalue*. We present here a spectral decomposition theorem for the special case where B is of the form $B = AA^T$ for some (possibly rectangular) matrix A. If A is a real valued matrix, then B is symmetric and positive definite. That is, $x^T Bx > 0$ for all non zero Vectors x. The spectral decomposition theorem holds more generally and the interested reader should consult a Linear Algebra book.

Theorem (Spectral Decomposition): If $B = AA^T$ then $B = \sum_i \sigma_i^2 u_i u_i^T$ where $A = \sum_i \sigma_i u_i v_i^T$ is the singular valued decomposition of *A*.
Proof: First

$$\begin{aligned} B = AA^T &= \left(\sum_i \sigma_i u_i v_i^T\right)\left(\sum_j \sigma_j u_j v_j^T\right)^T \\ &= \sum_i \sum_j \sigma_i \sigma_j u_i v_i^T v_j u_j^T \\ &= \sum_i \sigma_i^2 u_i u_i^T. \end{aligned}$$

■

When the σ_i are all distinct, the u_i are all of the eigenvectors of *B* and the σ_i^2 are the corresponding eigenvalues. If the σ_i are not distinct, then any Vector that is a linear combination of those u_i with the same eigenvalue is an eigenvector of B.

Theorem: If σ_i is an eigenvalue of multiplicity *k*, then σ_i has k linearly independent eigenvectors. That is, the space of eigenvectors is of dimension *k*.

Proof: We show that u is an eigenvector of B if and only if it is a linear combination of u_i with equal eigenvalues σ_i. Consider a candidate eigenvector *u* which is a linear combination of the u_i, say $u = \sum_i a_i u_i$. We must have

$$Bu = \sum_i a_i \sigma_i^2 u_i = \lambda \sum_i a_i u_i$$

for *u* to be an eigenVector. This implies that for any i, with $a_i \neq 0$, We must have $\lambda = \sigma_i^2$. This prowes that a_i can be non-zero only on a group of *i* with σ_i equal. Conversely, for such a group, any $u = \sum_i a_i u_i$ with a_i non-zero only on that group is an eigenvector with $\lambda = \sigma_i^2$.

■

The Power Iteration, Hubs and Authorities and PageRank

The power iteration has many applications, two of which we now illustrate. The first is to the definition of *Hubs* and *Authorities*. The World Wide Web can be represented by a directed graph whose nodes correspond to Web pages and directed edges to hyperlinks between pages. Some web pages, called *authorities*, are the most prominent sources for information on a given topic. Other pages called *hubs*, are ones which identify the so called authorities on a topic. Authority pages are pointed to by many hub pages and hubs are pages that 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 pointed to by many hubs.

One would like to assign hub weights and authority weights to each node of the web. If there are *d* nodes, the hub weights form a *d*-dimensional vector ***u*** and the authority weights form a *d*-dimensional vector ***v***. Suppose *A* is the matrix representing the directed graph. Then a_{ij} is 1 if node *i* points to node *j* and 0 otherwise. If we had an accurate hub vector ***u***, we could compute the authority vector ***v*** by the formula

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

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

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

Similarly, if we had an accurate authority vector ***v***, then we could calculate a hub vector ***u*** by $u = Av$. Of course, at the start, we have neither vector accurately. But the above immediately suggests a power iteration. Start with any ***v***. Set $u = Av$; then set $v = A^T u$ and repeat the process. This is the basis of the HITS algorithm.

Another important use of the power method is in using random walks to compute what is known as the *Page Rank* of a web page. The page rank value of a web page is used to rank the pages of the web.

A random walk on the web goes from web page *i* to an adjacent web page *j*. Associated with each page *i* is a probability distribution which for each adjacent page *j* gives the probability p_{ij} of selecting page *j* for the next step in the walk It is easy to see that if we represent the probabilities of being in each state at time *t* by a vector $p(t)$, then we have the probability of being in state *j* at time *t*+1 given by the equation

$$p_j(t+1) = \sum_i p_i(t)\, p_{ij}\ .$$

.

If we represent the transition probabilities by a matrix *P*, then

$$p^T(t+1) = p^T(t) P$$

and thus

$$p^T(t) = p^T(0) P^t\ .$$

Thus, computing $p(t)$ can be done by computing *P* to the power *t*. It turns out that under some conditions, the random walk has a steady state probability vector that we can think of as $p(\infty)$. It has turned out to be very useful to rank pages in decreasing order of $p_j(\infty)$ in essence saying that the web pages with the highest steady state probabilities are the most important. The first attempts were based just on going from *i* to one of the web pages pointed to by *i* picked uniformly at random, but there have been many improvements. The ranks are then computed by finding $p(t)$ for a large enough *t*.

A slightly more sophisticate random walk is used for several reasons. First, a web page might not contain any links and thus there is nowhere for the walk to go. Second, a page that has no in links will never be reached. Even if every node had at least one in link and one out link, the graph might not be strongly connected and the walk would eventually end up in some strongly connected component of the graph. A fourth difficulty occurs when the graph is periodic, that is, the greatest common divisor of all cycle lengths of the graph is greater than one. In this case, the random walk does not converge to a stationary probability distribution but rather oscillates between some set of probability distributions. We will consider this topic further in Chapter XXX.

Exercises

Exercise 4.: Generate a number of samples according to a mixture of 1-dimensional Gaussians. See what happens as the centers get closer. Alternatively, keep the centers fixed and vary the standard deviation and see what happens.

■

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

Exercise : Write pseudo-code based on the description of the algorithm above.

Exercise: Read in a photo and convert to a matrix. Perform a singular value decomposition. Reconstruct the photo using only 10%, 25%, 50% of the singular values. (a) Print the reconstructed photo. How good is the quality of the reconstructed photo?
(b) 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. 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: Find a collection of something and try the compression technique. Note that for non pictures it may not work since pictures may have a property that the eye can see only certain frequencies. Pictures or sounds might work.

■

Exercise: Create a 100×100 matrix A 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 100 singular vectors? How many singular vectors are required to capture 95% of the Frobenius norm?

Exercise: Generate a 100 such random matrices and find the first 100 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: Let A be a real valued matrix. Prove that $B = AA^T$ is positive definite.

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

Exercise:

Computational Problem : Compute the SVD of ????. Or write a program to implement the power method and SVD (run the power

method for a fixed number of steps…)………….More Details

Exercise: Let *A* be a square invertible matrix with SVD $A = \sum_i \sigma_i u_i v_i^T$. Prove that the inverse of *A* is $\sum_i \frac{1}{\sigma_i} v_i u_i^T$.

Exercise: Suppose *A* is square, but not necessarily invertible with SVD $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$. Let $B = \sum_i \frac{1}{\sigma_i} v_i u_i^T$. Show that $Bv = v$ for all *v* 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 (a) For any matrix *A*, show that $\sigma_k \le \frac{\|A\|_F}{\sqrt{k}}$.

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

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

Exercise: Let *A* be an $n \times d$ matrix. Suppose you can pre-process *A*. Then given a number of *d*-vectors $x_1, x_2, \ldots, x_m$. For each x_i find the vector Ax_i approximately, in the sense that you find a vector u_i satisfying $|u_i - Ax_i| \le \varepsilon \|A\|_F |x_i|$ (here ε >0 is a given error bound). Describe an algorithm that accomplishes this in time $O\left(\frac{d+n}{\varepsilon^2}\right)$ per x_i (not counting the pre-processing time).

Exercise: Given *A*,*b,* and *m*, use SVD to find a vector *x* with |x|<m minimizing $|Ax - b|$.

Constrained Least Squares Problem using SVD – Golub and van Loan, Chapter 12 :

(More explanation or algorithm as in GL chapter 12 should be given…..)

Exercise: Add exercise on min cut