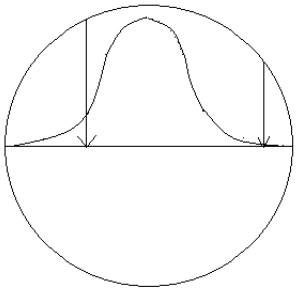
CS 683: Advanced Design & Analysis of Algorithms April 9, 2008

Lecture 32: High Dimensions and Large Data Sets

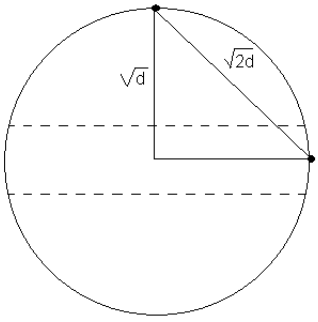
Lecturer: John Hopcroft *Scribe: Sucheta Soundarajan, Chris Provan*

1 High Dimensional Data

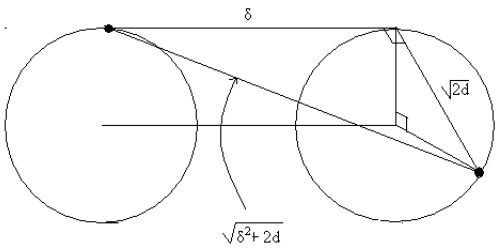
Consider the surface area of a hypersphere. If we divide the surface area into tiny squares, and project down onto a line through the center of the sphere, we will get a Gaussian distribution.



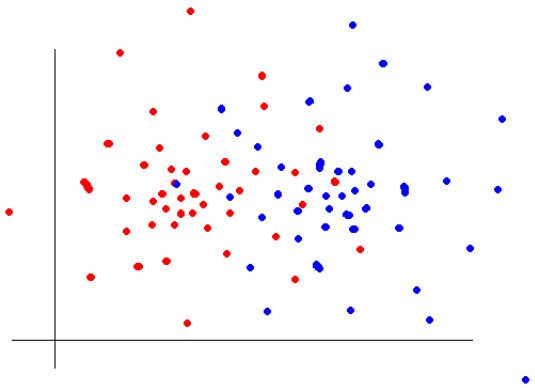
Because the distribution is Gaussian, the vast majority of points appear within 3 standard deviations of this center line. This means that most of the points occur within the volume bounded by two hyperplanes perpendicular to that center line. Then what is the distance between two random points on a sphere?



Now what if we consider two different spheres? The distance between two points on the same sphere is $\sqrt{2d}$, but what about the distance between two points on different spheres? Suppose that the spheres are separated by distance δ. The distance between the two points is $\sqrt{\delta^2 + 2d}$.



We can use this knowledge to solve an interesting problem. Suppose we have two Gaussian processes creating data in high dimensions. Suppose further that the centers of the processes are close enough that there is significant overlap in the areas where their points fall.



We know that the distance between two points created by the same process is about $\sqrt{2d}$, and the distance between two points created by different processes is about $\sqrt{\delta^2 + 2d}$. We must include an error term γ, since these numbers are only exact as the dimensions go to ∞. Then in order to distinguish points from different processes, we need:

$$\sqrt{\delta^2 + 2d} \geq \sqrt{2d} + \gamma$$

$$\sqrt{2d}(1 + \tfrac{1}{2}\tfrac{\delta^2}{2d} + ...) \geq \sqrt{2d} + \gamma$$

$$\tfrac{\delta^2}{2\sqrt{2d}} \geq \gamma$$

$$\delta^2 \geq 2\sqrt{2d}\gamma$$

$$\delta \geq (2\sqrt{2d}\gamma)^{\frac{1}{2}} d^{\frac{1}{4}}$$

So suppose we have d = 10000. Then we need the processes to be separated by about $\delta = 10$ in order to distinguish them.

Exercise: Try and see if this works!

Can we do anything to further reduce δ? Suppose we have k Gaussian processes. We can use the SVD to remove noise, and project down onto k dimensions.

$[x_1, x_2, ..., x_d] \rightarrow [x_1, x_2, ..., x_k]$ (with change of coordinates). We project onto k dimensions because k is the minimum number which will allow us to distinguish the centers, since there are k centers. Then this gives us a $\frac{k}{d}$ reduction, so now all we need is $\delta \geq (2\sqrt{2d}\gamma)^{\frac{1}{2}} k^{\frac{1}{4}}$

Some possible portfolio exercises:

- Consider a hypercube in d dimensions with a line connecting an opposite pair of vertices. Where do the other vertices fall if they are projected onto this line?
- For $d = 1000$, how many data points do I need to accurately pinpoint the center of a cluster of points?
- Is there an efficient algorithm for finding the nearest neighbor of a data point? This remains an outstanding research problem. To illustrate the difficulty, suppose we project the points into a single dimension with the hopes of checking some number of the nearest neighbors in this dimension. By our above analysis, the points will all be clustered about the one-dimensional origin, so this method will not work.

2 Large Data Sets

2.1 Sketches

Suppose we would like to store the content of a large number of webpages in such a way that, in an offline setting, we can compare an original document (such as a student paper) to the web documents to check for similarities. Clearly storing the documents in their intact form is not feasible. If each document is a sequence of 1000 numbers (representing sets or sequences of characters) then we could select a small subset, perhaps ten, of these numbers to compare. So we sort the numbers for each document (using a random sort order to avoid comparing any common software prefixes or attempts to game the system), select the 10 lowest numbers, and compare.

How can we convert from a sequence to a set and back? Use a sliding window to grab the elements of the set. A four character sliding window, for example, would encode the first sentence of this paragraph as

'How_' 'ow_c' 'w_ca' '_can' 'can_' 'an_w' etc.

We can then reconstruct the original document by matching the sequences included in the set representing the document. An interesting question is how long should the segments be in order to uniquely reconstruct the original document? This issue is complicated when there are long repeated sequences in the original document.

2.2 Data Streams

Suppose now that we would like to count the number of occurrences of a given symbol or sequence in a string of n 0's and 1's. An exact solution requires $\log n$ bits of storage since there are at most n occurrences.

The storage requirement could be reduced to $\log \log n$ if we only needed to store the logarithm of the number of occurrences. This is accomplished by a random algorithm as follows:

- Set $k = 1$ after the first occurrence.
- After each subsequent occurrence, increment k with probability $\left(\frac{1}{2}\right)^k$.

Then 2^k is approximately the number of occurrences in the data stream.