CS 683: Advanced Algorithms	**Nearest Neighbor Search**
DATE 04/17/2001	**Scribe: Elliot Anshelevich, Anirban Dasgupta**

1 Applications of Nearest Neighbor

We now discuss an important geometric problem — nearest-neighbor search — and an approach to solving it through an application of VC-dimension. Given a set of points of high dimension, we want to find out the nearest (or the k nearest) neighbors of any given query point. We give two areas where this application comes in handy.

Example 1.1 *In image-processing, often digital images are stored as just a string of bits. One possibility could be chopping the entire image into regions and computing the average values of the colour intensities over each region and storing these averages for all the regions. So, the image could now be represented as a vector over R^d where d is the number of regions, and is typically very large. In this case, Euclidean distance between two image-vectors gives us some notion of similarity.*

Example 1.2 *This example relates to the vector space model of image retrieval developed by Salton. Suppose we have a reasonable vocabulary of the English language. We represent documents in English by a vector with one coordinate for each word in the vocabulary. Document i is mapped to a vector $v_i \in R^d$. Typically d is of the order of $50K$ to $100K$ for a reasonable information retrieval system. The j^{th} coordinate of v_i stores the number of times the j^{th} word from the vocabulary has appeared in the document. Or we may take the vectors to be just boolean vectors, the j^{th} coordinate representing whether the j^{th} word has at all appeared in the document. Again, in this case, the notion of similarity of two vectors under the L_1 or L_2 norm gives us some idea of similarity between documents.*

1.1 Nearest Neighbor Search

Suppose we have a set of points P in R^d. Our aim is to preprocess it so that we can provide answers to nearest neighbor queries in reasonable time. That is, given a query point $q \in R^d$, we shall return the nearest neighbor $p_i \in P$ achieving $\min_{p \in P} d(p, q)$, or we might need the k-nearest points to q. In our discussion we shall restrict ourselves to the nearest-neighbor problem only.

The complexity of this problem consists of two parts: the preprocessing and storage time, and the actual query time. For the case of $d = 1$, i.e. when all the points are on a line, all we need to do is sort the points and store the midpoints between successive points. Then, given a query point, all we need to do is do a binary search on the stored midpoints to find in which interval it lies. This directly gives us the nearest neighbor. Thus the preprocessing time is $O(n \log n)$ and the storage requirement is $O(n)$. The query time complexity is $O(\log n)$.

For two dimensions, the analogue of the above algorithm is finding out Voronoi diagrams. A Voronoi diagram of a set of points $\{p_1, \ldots, p_n\}$ is defined as a partition of the plane into

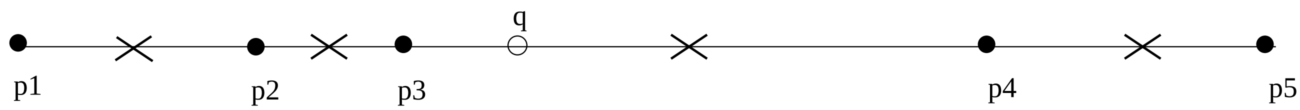

Figure 1: One dimensional nearest neighbor

cells (closed or open polygons) such that the point p_i lies in cell i and all points in R^2 lying in cell-i are nearer to p_i than to any other point from the given set. It is known that Voronoi diagrams in R^2 can be found in time $O(n \log n)$ and require linear storage. Also, given a query point, we just need to find out which cell it lies in. This can possibly be done by constructing projections on both the coordinates and doing binary search (not sure).

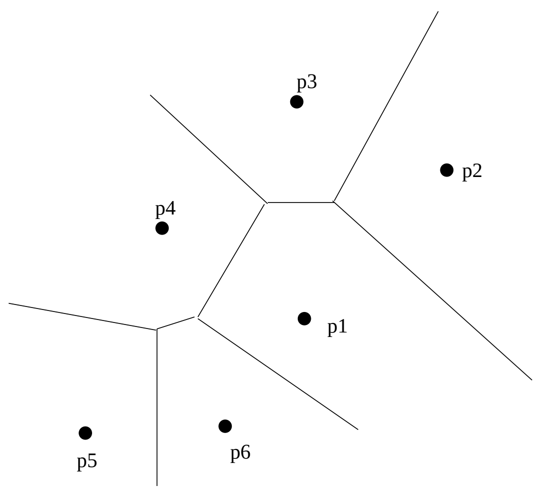

Figure 2: Voronoi diagrams

This technique does not scale well to higher dimensions. For larger d, Clarkson (1987) improved on a long sequence of previous work with an algorithm for nearest-neighbor search that has pre-processing and storage requirement $O(n^{\lceil d/2(1+\varepsilon) \rceil})$ The query time is $O(c^d \log n)$ for a constant c. As d gets near or as high as $\log n$, this query time is not really doing better than the brute force search algorithm which takes no preprocessing time, linear storage and linear query time.

1.2 High dimensional nearest neighbor search

We introduce the notion of approximate nearest neighbor.

Definition 1.1 *Given a initial set of points $P = \{p_1, \ldots, p_n\}$, and a query point q, call p_i to be the ε-approximate nearest neighbor to q if $d(p_i, q) \leq (1 + \varepsilon) \min_{p \in P} d(p, q)$.*

Due to a result by Arya et al. we get that ε-approximate nearest neighbor can be solved in preprocessing time $O(n \log n)$, storage space $O(n \log n)$, and query time $O(c^d \varepsilon^{-d} \log n)$. Our goal is the following

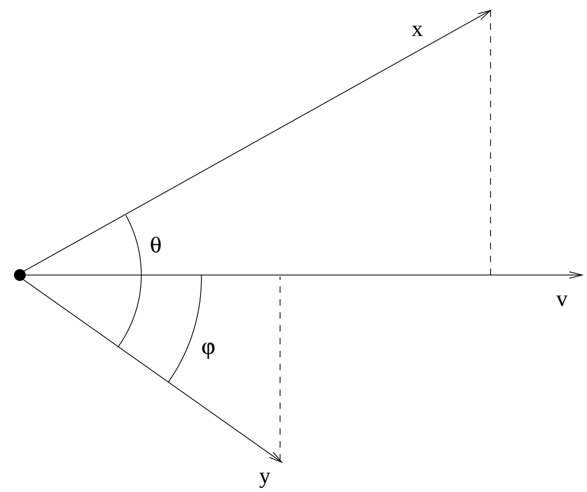

Figure 3: Random projection

Goal. Suppose we are willing to pay high preprocessing and storage costs. Then can we achieve a query time cost of $O(poly(\frac{1}{\varepsilon})poly(d)\log n)$?

It turns out that this is possible by using an idea based on random projections (Kleinberg 1997). The intuition behind the algorithm is that relative distances between the query point and the initial set P should preserve their relation under projection on random vectors.

1.3 Random Projections

Suppose we have points p_i and p_j from the initial set of points. Let q be the query point. Let us take a random line L and project the vectors $\overline{p_i q}$ and $\overline{p_j q}$ on the line L. We need to analyze the probability that the shorter vector actually has a shorter projection. We present a general lemma for this.

Lemma 1.2 *Suppose* $0 < \gamma \leq \frac{1}{2}$*,* $x, y \in R^d$ *such that* $\|x\|(1+\gamma) \leq \|y\|$ *(i.e.* x *is sufficiently smaller than* y*), Now, we choose a vector* v *uniformly at random from the unit sphere* S^{d-1} *in* R^d*. Then* $Pr[|v \cdot x| < |v \cdot y|] \geq \frac{1}{2} + \frac{\gamma}{5}$*.*

Proof. It is enough to think about the 2-dimensional space spanned by x and y. Let $r = \frac{\|x\|}{\|y\|} \leq \frac{1}{1+\gamma}$. Suppose the angle between v and x is φ and the angle between x and y is θ. So we need to know if $\cos^2(\theta - \varphi) \leq r^2 \cos^2 \varphi$. This being true is a bad outcome for us, and hence we want to upper bound the probability for this event. It is easily seen that the worst case occurs when the vectors x and y are orthogonal to each other, i.e. $\theta = \frac{\pi}{2}$. Actually the probability of the "bad" event occurring increases from 0 to $\frac{\pi}{2}$ and decreases again from $\frac{\pi}{2}$ to π, and so on. So we need to see when $\cos^2(\frac{\pi}{2} - \varphi) = \sin^2 \varphi \leq r^2 \cos^2 \varphi$ i.e. need upper bound $Pr[|\tan\varphi| \leq r]$. Hence, $Pr[\text{bad event}] = Pr[|\tan\varphi| \leq r] = \frac{2\tan^{-1} r}{\pi} < \frac{1}{2}$ as from Figure 4. If we do a Taylor expansion of $\tan^{-1} r$, we can show that that the above probability is less that $(\frac{1}{2} - \frac{\gamma}{5})$. Hence, $Pr[\text{bad event}] < \frac{1}{2} - \frac{\gamma}{5}$. Hence the lemma follows. ∎

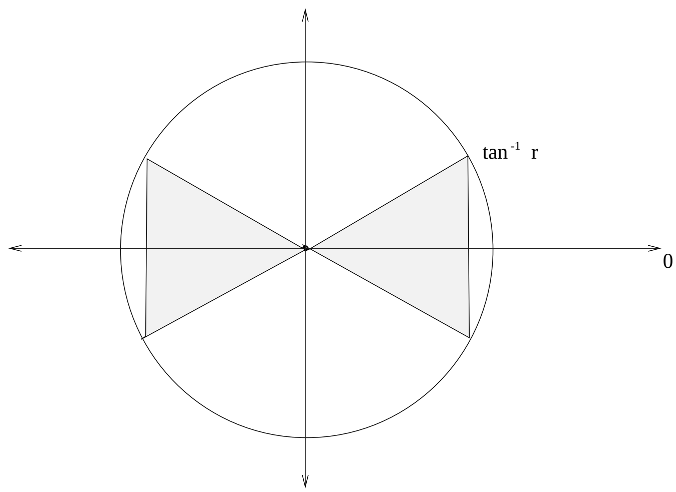

Figure 4: Wedge for "bad" event

From the lemma, we get the following corollary.

Corollary 1.3 *Given x and y, the set $W_{x,y}$ of vectors from S^{d-1}, that give rise to the bad event (i.e. the projection of x exceeds projection of y, under the conditions of the above lemma), is a wedge of hyperplanes of probability measure $< \frac{1}{2} - \frac{\gamma}{5}$.*

Definition 1.4 *A distinguishing set is a finite set of points V on the unit sphere in $\mathbb{R}^d$ so that no wedge $W_{x,y}$ of measure $< \frac{1}{2} - \frac{\gamma}{5}$ (for any x and y producing such a wedge) has at least half of V.*

The point of a distinguishing set is that V gives a correct length comparison for any $x, y \in \mathbb{R}^d$ differing by $\geq 1 + \gamma$ factor, by majority vote.
Question: How big must V be? This is actually a VC-dimension question. The ground set here is the unit sphere (denoted S^{d-1}), and $\mathcal{C}$ are all the wedges.
Fact: An ε-sample for the infinite set system (S^{d-1},wedges) with $\varepsilon = \frac{\gamma}{5}$ is a distinguishing set. This is clear from the definitions of distinguishing set and ε-sample. So, if $d' =$VC-dim(S^{d-1},wedges), then we can take a random sample from S^{d-1} of size $|V| = O(\frac{d'}{\gamma^2} \log \frac{d'}{\gamma} + \frac{1}{\gamma^2} \log \frac{1}{\delta})$, since this would form an $\frac{\gamma}{5}$-sample with probability $\geq (1 - \delta)$. This is in terms of d', however, so we need to bound d'.

Claim 1.5 $d' = O(d \log d)$

To prove this, notice that all wedges are a result of a Boolean function of four halfspaces, specifically the function $f(A_1, A_2, A_3, A_4)$ which takes four halfspaces and produces $(A_1 \cap A_2) \cup (A_3 \cap A_4)$. (The Boolean operations here are $\cap$ and $\cup$). Therefore, Claim 1.5 results from the following Lemma:

Lemma 1.6 *Let f be a Boolean function on h inputs, each input a set. Let $(U, \mathcal{R})$ be a set system of VC-dimension d. Let $(U, \mathcal{R}_f)$ be the new set system, where $\mathcal{R}_f = f($all combinations of $\mathcal{R})$. Then the VC-dimension of $(U, \mathcal{R}_f)$= $O(dh \log dh)$ if $h = O(d)$.*

Proof. If $A \subseteq U$ has size n, at most $\sum_{i=0}^{d} \binom{n}{i}$ subsets of A can be realized by intersections with members of $\mathcal{R}$, by definition of VC-dimension. How many different intersections of A can there be with members of $\mathcal{R}_f$? We know that $\sum_{i=0}^{d} \binom{n}{i} \leq n^d$, so there are at most n^d sets in $\mathcal{R}$ which result in different intersections with A. There are $\leq \binom{n^d}{h} \leq n^{dh}$ combinations of these sets of $\mathcal{R}$ that can be given to f as inputs, so there are at most n^{dh} sets of $\mathcal{R}_f$ which yield different intersections with A. So if A is shattered by $(U, \mathcal{R}_f)$, then $n^{dh} \geq 2^n$, by definition of "shattered". This means that $dh \log n \geq n$, so $n/\log n \leq dh$. Therefore, $n = O(dh \log dh)$ would be enough for A not to be shattered, and so VC-dim$(U, \mathcal{R}_f)$= $O(dh \log dh)$. ∎