Lecture #6

Last Time: The Johnson-Lindenstrauss lemma, and sparse / fast variants

unbiased estimator of ℓ_2-distance via Gaussians

Today: Nearest neighbor search and LSH

NN Setup: Given a set P of n points in d-dimesions

(1) construct data structure

(2) be able to answer query point(s) q, with closest point in P

e.g. spam classification

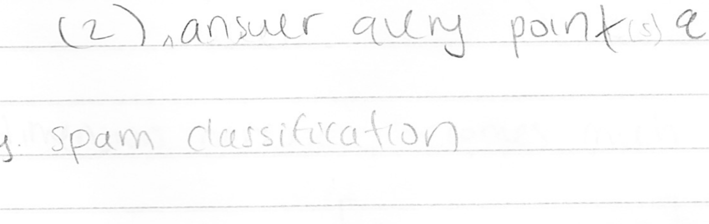
Some initial ideas:

Approach #1: No preprocessing, on query, linearly search through P

space: $O(dn)$ query time: $O(dn)$

Can we improve the query time? And at what cost?

Approach #2 ($d=1$): Binary search tree



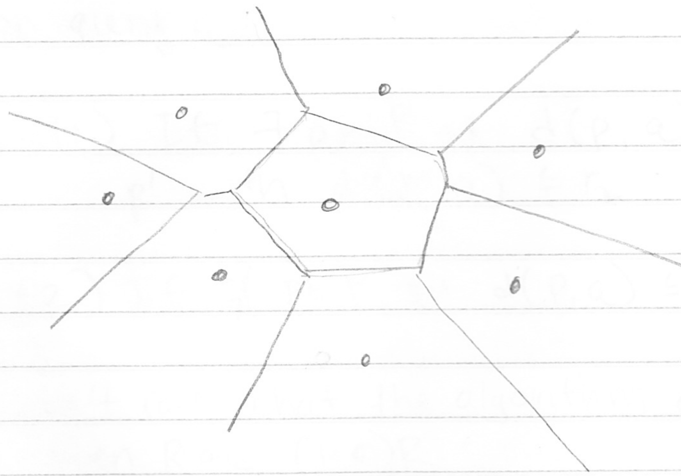
more generally, think of P as partitioning space

construct binary search tree on P, on query search for q

space: $O(n)$ query time: $O(\log n)$

Approach #2 ($d=2$) Voronoi Diagram

partition of plane into regions based on which queries map to given point



As dimension increases, becomes much more complex

Other kd-trees [Bentley, 1975]

All approaches in high-dimension have exponential space or query time or both!

"curse of dimensionality" computational geometry coined by Bellman

we will study relaxation

c-ANN

Setup: Given a set P of n points in d-dimensions construct data structure s.t. on query q:

Return a point $p \in P$ with

$$d(p,q) \le c \min_{p' \in P} d(p',q)$$ ← c-approx nearest neighbor

Even easier:

(r_1, r_2)-PLEB

point location in equal balls

Setup: Given a set P of n points in d-dimensions and radii r_1, r_2, construct data structure s.t. on query q

(1) If $\exists p \in P$ s.t. $d(p,q) \le r_1$ return YES and any p' with $d(p',q) \le r_2$

(2) If $\nexists p \in P$ s.t. $d(p,q) \le r_2$ return No

we don't care what the algorithm does if closest point is between R and $(1+\varepsilon)R$

Let D_{max}/D_{min} be max/min interpoint distances in P

Lemma: If for every r there is a data structure with space S and query time T that solves $(r, (1+\varepsilon)r)$-PLEB then there is an algorithm for $(1+\varepsilon)^2$-ANN with space $O(S \log_{1+\varepsilon} \frac{D_{max}}{D_{min}})$ and query time $O(T \log\log_{1+\varepsilon} \frac{D_{max}}{D_{min}})$

Pf: Construct a data structure for $(r, (1+\varepsilon)r)$-PLEB for radii

$$\frac{D_{min}}{2}, (1+\varepsilon)\frac{D_{min}}{2}, (1+\varepsilon)^2 \frac{D_{min}}{2}, \ldots \approx D_{max}$$

Use binary search to find minimum r s.t. $(r, (1+\varepsilon)r)$-PLEB returns YES, let p be returned point and let r^* be radius. Then

(1) $d(p,q) \le (1+\varepsilon) r^*$ since $(r^*, (1+\varepsilon)r^*)$-PLEB said YES

(2) $\forall p', \ d(p',q) \ge \frac{r^*}{(1+\varepsilon)}$ since $\left(\frac{r^*}{(1+\varepsilon)}, r^*\right)$-PLEB said NO

Thus p is a $(1+\varepsilon)^2$-ANN $\boxtimes$

ring-cover trees

More efficient reductions in [Indyk, Motwani 1998] and [Har-Peled, 2001] $(r, (1+\varepsilon)r)$-PLEB $\Rightarrow$ $(1+\varepsilon)$-ANN

So far collisions = bad, but today we will exploit them

<u>Locality Sensitive Hashing</u>: [Indyk, Motwani 1998]
similar items are <u>more</u> likely to map to same bucket

<u>Setup</u>: Hash family $\mathcal{H} = \{h : U \to S\}$ is called (r_1, r_2, P_1, P_2)-locality sensitive if for any $p, p' \in U$

(1) If $d(p, p') \le r_1$ then $\Pr_{h \in \mathcal{H}}[h(p) = h(p')] \ge P_1$

(2) If $d(p, p') \ge r_2$ the $\Pr_{h \in \mathcal{H}}[h(p) = h(p')] \le P_2$

we will always work with $P_1 > P_2$ and $r_1 < r_2$

<u>Theorem</u> [Indyk, Motwani]: Suppose there is a (r_1, r_2, P_1, P_2)-locality sensitive hash family $\mathcal{H}$. Then there is an algorithm for (r_1, r_2)-PLEB which uses

<u>space</u>: $O(dn + n^{1+\rho})$
<u>query time</u>: $O(n^{\rho})$ evaluations of hash fctn

where $\rho = \frac{\ln 1/P_1}{\ln 1/P_2}$, and succeeds with constant probability (for fixed P, q)

The space is polynomial, but query time is <u>sublinear</u>

Let k, ℓ be parameters chosen later. Let

amplification

$$\mathcal{G} = \{g : U \to \underbrace{S^k}_{\text{new bucket}}\}$$ where

$g = (h_1(p), h_2(p), \ldots h_k(p))$ and each $h_i \in \mathcal{H}$

Preprocessing: (1) Choose $g_1, g_2 \ldots g_\ell$ independently, uniformly at random

(2) For each $p \in P$, store it in buckets $g_1(p), g_2(p) \ldots g_\ell(p)$

(3) Discard empty buckets

On query: Search $g_1(q), g_2(q) \ldots g_\ell(q)$ and find first 2ℓ points.
return any point p found with $d(p,q) \le r_2$ and YES. Else return NO.

g_i, p, q

Analysis: We want to show, for fixed P, q, with constant probability over choice of g_i's, the following events hold:

(1) If $\exists p \in P$ with $d(p,q) \le r_1$ then $g_j(p) = g_j(q)$ for some j

(2) There are at most 2ℓ points $p \in P$ with $d(p,q) \ge r_2$ and $g_j(p) = g_j(q)$ for some j

Fix j:

Let $k = \log_{1/P_2} n$, then the expected number of points satisfying the conditions in (2) is at most

$$n (P_2)^k = n (P_2)^{\log_{1/P_2} n} \le 1$$

Thus by Markov, the probability (2) does not hold is at most $\frac{1}{2}$

Fix j: The probability of $g_j(p) = g_j(q)$ in (1) is bounded from below by

$$P_1^k = P_1^{\log_{1/P_2} n} = \underbrace{\left(\left(\frac{1}{P_2}\right)^{\log_{1/P_2} P_1} \right)}_{P_2^{\frac{\log 1/P_1}{\log 1/P_2}}}^{\log_{1/P_2} n} = n^{\frac{-\log 1/P_1}{\log 1/P_2}} = n^{-\rho}$$

$\ln 1+x = x - \frac{x^3}{3!} \cdots$, thus

$$\rho = \frac{\ln \frac{1}{1-\frac{r}{d}}}{\ln \frac{1}{1-\frac{cr}{d}}} \approx \frac{\ln 1+\frac{r}{d}}{\ln 1+\frac{cr}{d}} \approx \frac{\frac{r}{d}}{\frac{cr}{d}} = \frac{1}{c}$$

thus the probability that (1) holds is at least

$$1-(1-n^{-\rho})^{\ell} \geq 1 - 1/e > \frac{1}{2}$$

and setting $\ell = n^{\rho}$ gives.

thus with constant probability both (1) and (2) hold, which implies the algorithm succeeds in solving (r_1, r_2)-PLEB $\boxtimes$

Some applications

Let $\mathbb{H}^d = \{0,1\}^d$ be the d-dimensional Hypercube, then

$$\mathcal{H} = \{ h_i \mid h_i(\overbrace{b_1, b_2 \ldots b_d}^{\text{bits}}) = b_i \text{ for } i=1,2\ldots d\} \text{ is}$$

$(r, cr, 1-\frac{r}{d}, 1-\frac{cr}{d})$-locality sensitive, for the Hamming distance.

<u>Corollary</u>: There is an algorithm for (r,cr)-PLEB in $\mathbb{H}^d$ that uses space $O(dn + n^{1+\frac{1}{c}})$ and for each query needs $O(n^{\frac{1}{c}})$ evaluations of the hash function, each of which takes $O(d)$ time.

i.e. for the Hamming distance, $\rho \leq 1/c$

<u>Theorem</u> [Andoni, Indyk, 2006] for Euclidean distance, $\rho \leq 1/c^2$

These bounds are tight [O'Donnell, Wu, Zhou 2011]