Approximate Nearest Neighbor
(Hamming Distance)

Vahab S. Mirrokni

Massachusetts Institute of Technology

Overview

- Definitions
- Motivations and previous algorithms
- An algorithm with fast query, large storage[KOR98]
- An algorithm with sublinear query, moderate storage[IM98]

Nearest Neighbor Search

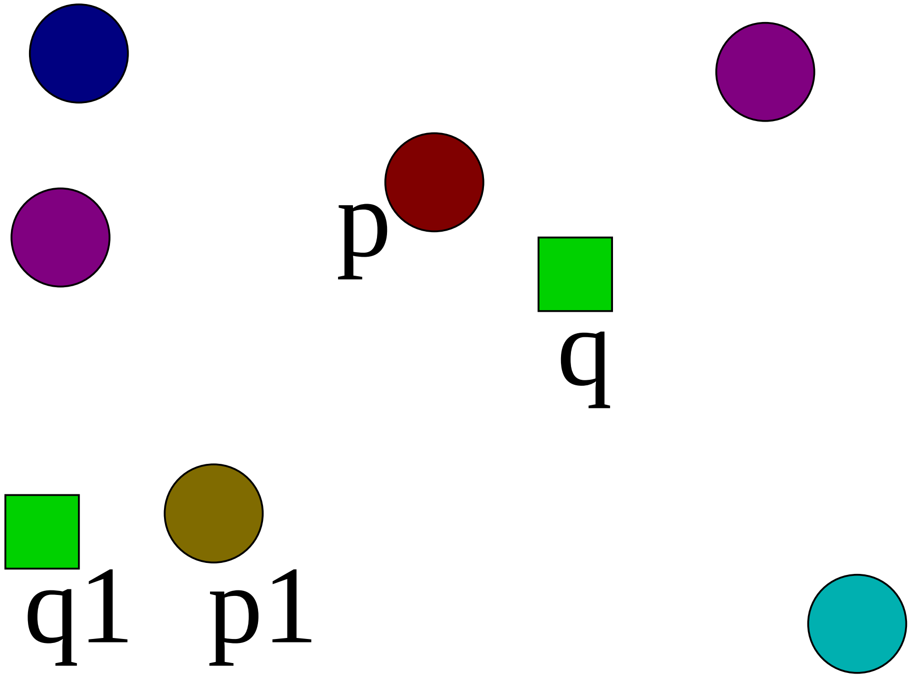
Given

- Metric space $D = (X, d)$, n **points** $P \subset X$

Goal

- A data structure which for any query $q \in X$ returns the point(s) in P **closest to** q.

Example: NN



Metric Space: Hamming Distance

Metric Space $D = (X, d)$

- Metric space examples: l_1, l_2, l_p, l_∞, Hamming distance

- p, q are two binary strings of length d or points in binary space of dimension d,

- $D(p, q) =$ the number of different bits in two strings

- Using **embedding**, we can have similar algorithms for other metrics.

Motivations and Applications

- Searching biological databases or genetic sequences.
- Discretizing other spaces to $\{0,1\}$ sequences.
- Core algorithms for proximity problems
    - closest pair, diameter, MST, matching.

Previous Works and Goals

Previous works

- Sublinear query time, Space exponential in d

Goals

- Sublinear query time, Moderate storage.

PLEB (Point Location in Equal Balls)

Given

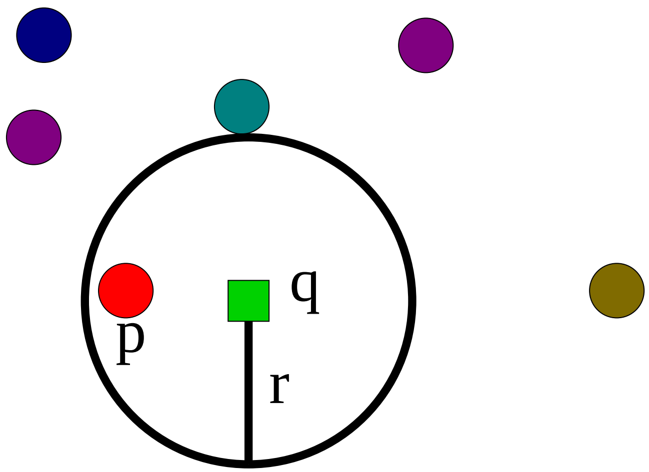
Metric space, n-set P$\subset X$ of points, $r > 0$

Goal

A data structure which for any query $q \in X$

- if there is $p \in P$ s.t. $D(q, p) \leq r$: returns p
- otherwise: returns NONE

Example



Ball with radius r around q

- ϵ-NN can be reduced to PLEB problem in $O(n^{\frac{\log n}{\epsilon^d}} \log \frac{n}{\epsilon})$ space and $O(\log \frac{n}{\epsilon})$ query.

Simple Algorithm

- if d is **small** ($d = O(\lg n)$): for all queries keep the answer in a table

Simple Algorithm

- if d is **small** ($d = O(\lg n)$): for all queries keep the answer in a table

PLEB-PreProcessing0(P, d, r)

Construct a table in which we have an answer for all queries

PLEB-Query0(q, r)

return answer for q from the table

Simple Algorithm

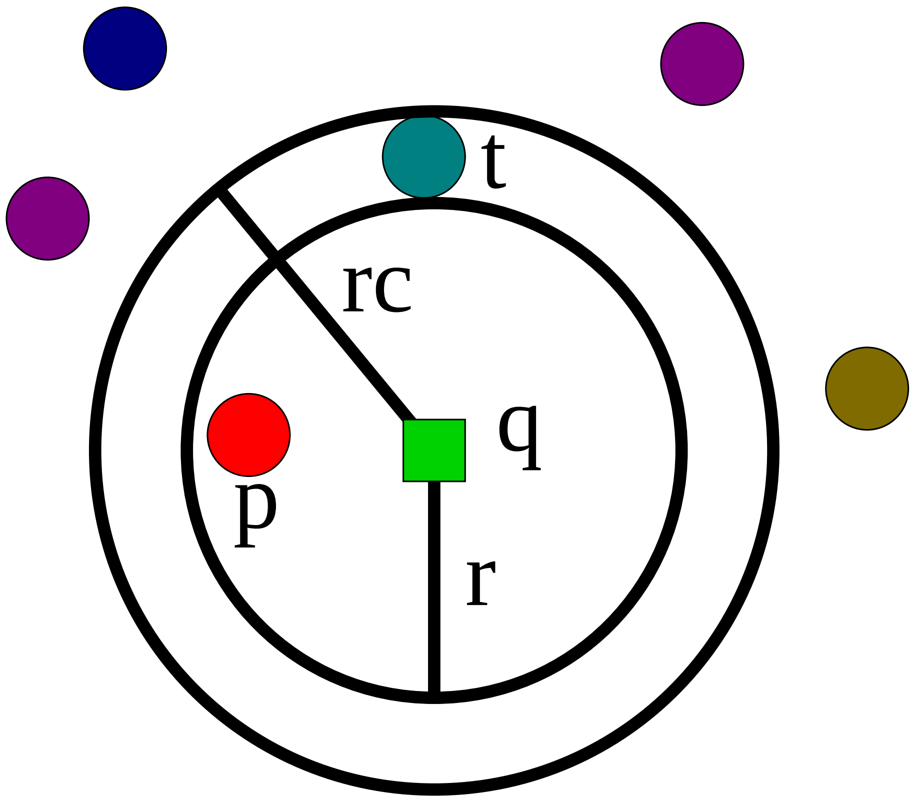
- if d is **small** ($d = O(\lg n)$): for all queries keep the answer in a table

> PLEB-PreProcessing0(P, d, r)
>
> Construct a table in which we have an answer for all queries
>
> PLEB-Query0(q, r)
>
> return answer for q from the table

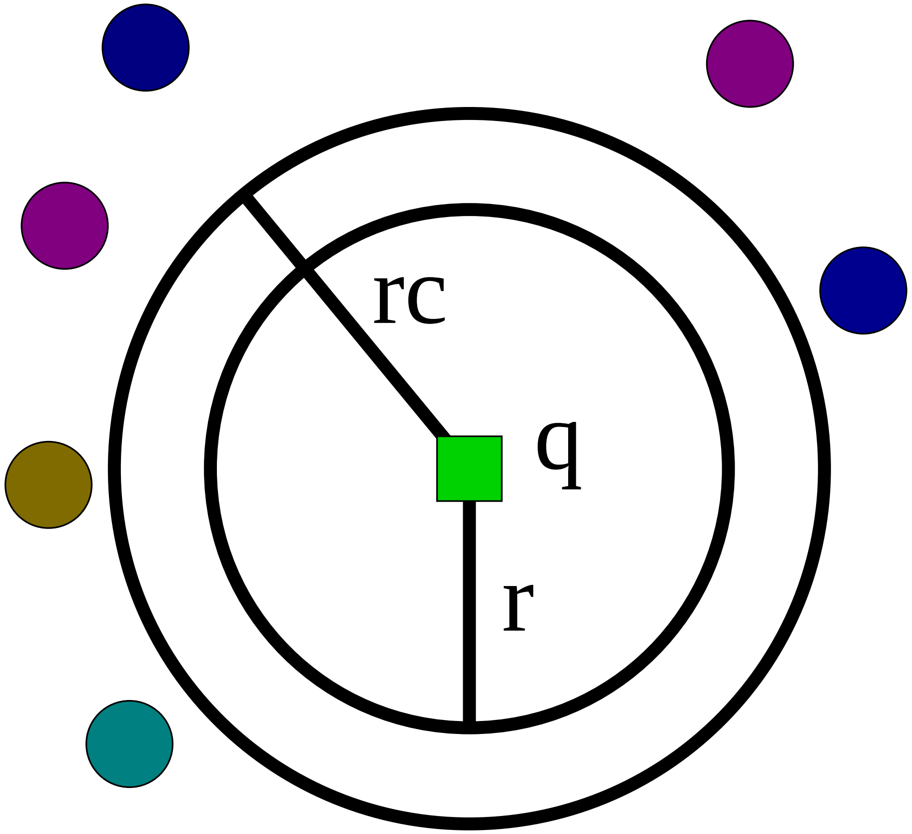
- Space: 2^d Query: $O(d)$
- How about **large** d?

Goal
A data structure that for any query $q \in X$

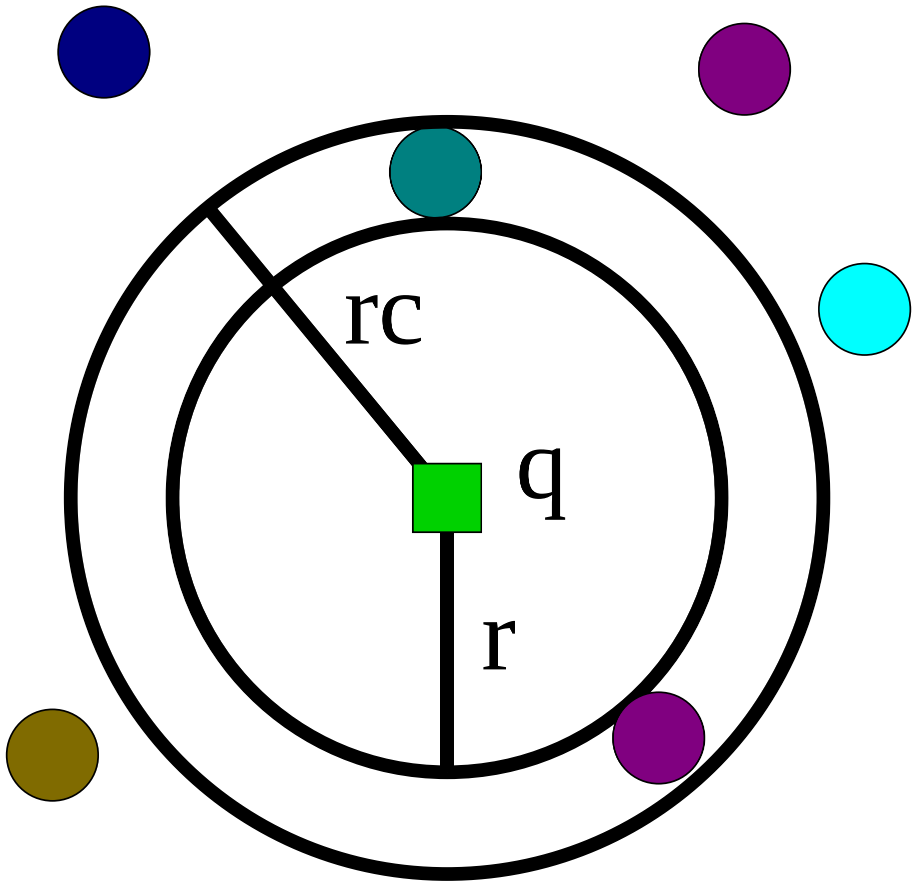
- if $D(q, P) \leq r$, return p' s.t. $D(q, p') \leq r(1 + \epsilon)$
- if $D(q, P) > (1 + \epsilon)r$: return NONE.
- otherwise: return anything.
- $c = 1 + \epsilon$: (r, c)-PLEB problem.



return t (or p).



return NONE



return anything

Reduce dimension from d to d' so that:

- d' is small enough ($2^{d'}$ can be stored)

Randomized Dimensionality Reduction

Reduce dimension from d to d' so that:

- d' is small enough ($2^{d'}$ can be stored)

- distance between query points and the data set points won't change more than a factor of $1 + \epsilon$ (with new distance r')

- solve PLEB in the new data sets and r' (not approximately) to solve ϵ-PLEB for the first data set

First Algorithm

PLEB-PreProcessing1(P, d, r)

Construct a mapping g from dimension d to d'

PLEB-PreProcessing0$(g(P), d', r')$

PLEB-Query1(q, r)

return PLEB-Query0$(g(q), r')$

- Space at least $2^{d'}$

$g, d', r' = ?$

Properties of Mapping g ($\epsilon < 1$)

Mapping $g : \{0,1\}^d \rightarrow \{0,1\}^{d'}$ such that for fixed points p and q,

- $d' = O(\frac{\lg n}{\epsilon^2})$(small enough)
- $D(p,q) \leq r \Rightarrow Pr[D(g(p), g(q)) \leq r'] \geq \frac{1}{2}$
- $D(p,q) \geq (1+\epsilon)r \Rightarrow Pr[D(g(p), g(q)) \leq r'] \leq \frac{1}{3n}$

How to use mapping?

- If there exist $p : D(p,q) \leq r$
$$\Rightarrow Pr[D(g(p), g(q)) > r'] \leq \frac{1}{2}$$
- If $D(p,q) \geq (1+\epsilon)r$
$$\Rightarrow Pr[D(g(p), g(q)) \leq r'] \leq \frac{1}{3n}$$
- if there is no $p' : D(p', q) \leq (1+\epsilon)r$: then
$$Pr[\exists p : D(g(p), g(q)) \leq r'] \leq \frac{1}{3}$$

.

How to use mapping?(ctd')

Bad Events

- E_1 :There is a point $p^* : D(q, p^*) \leq r$, but $D(g(q), g(p^*)) > r'$ $\leftarrow Pr[E_1] < \frac{1}{2}$

- E_2 : There is a point $p^* : D(q, p^*) \leq r$, but for another point $p : D(p, q) > rc$, we have $D(g(q), g(p)) \leq r' \leftarrow Pr[E_2] < \frac{1}{3}$

- Thus the probability of incorrectness is at most

$$\boxed{\frac{1}{2} + \frac{1}{3} = \frac{5}{6} < 1}$$

Construction

- $p|I_i$ = projection of p on set I_i

- $g(p)$ is concatenation of d' bits.

Construction of ith bit

- random function of sample k bits from binary string

How to find g(p)

- $I_i \subset \{1,\ldots,d\}, |I_i| = k (i = 1,\ldots,d')$
- $f_i : \{0,1\}^{|I_i|} \to \{0,1\}$ random functions.
- elements of I_i are chosen independently and uniformly at random from $\{1,\ldots,d\}$ ($|I_i| = k = \lceil \frac{d}{r} \rceil$).
- $g_i(p) = f_i(p|I_i) \Leftarrow$ a random function of sample of k random bits.
- $g(p) = (g_1(p),\ldots,g_{d'}(p))\Leftarrow$ concatenation d' of above bits

Example

- $d = 10,\ d' = 2,\ r = 5 \rightarrow (5, c)$-PLEB $\Rightarrow k = \frac{d}{r} = 2$
- $I_1 = \{5, 7\}, I_2 = \{4, 9\}$

$p = 1101101101$

$$\begin{array}{cc|c||cc|c}
f_1 & & & f_2 & & \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0
\end{array} \Rightarrow \begin{array}{l} g_1(p) = f_1(11) = 1 \\ \\ g_2(p) = f_2(10) = 0 \end{array} \Rightarrow g(p) = (10)$$

Intuition

- $Pr[p|I_i = q|I_i]$ will depend on $|I_i|$ (number of sample bits) and $D(p, q)$

- f just returns a random 0 or 1

How to implement

- maintain random f by keeping values of f which have been evaluated so far.

- for evaluating $f_i(p)$, if there is no $f_i(p)$ choose $f_i(p)$ at random and store $(f_i(p), p)$, otherwise report $f_i(p)$.

- using hashing $\rightarrow$ negligible time

- to compute g: it takes $O(dd')$.

Proof Ideas

Suppose $D(p,q) = \delta r$

- $P(\delta) = Pr[g_i(p) = g_i(q)]$
- $g_i(p) = g_i(q) \Leftrightarrow (p|I_i = q|I_i$ or $((p|I_i \neq q|I_i) \wedge (f_i(p|I_i) = f_i(q|I_i))$
- $Q(\delta) = Pr[p|I_i = q|I_i]$
- f is random $\Leftrightarrow P(\delta) = \frac{1}{2}(1 - Q(\delta)) + Q(\delta) = \frac{1}{2}(1 + Q(\delta))$

$$\boxed{Q(\delta) = (1 - \frac{\delta r}{d})^k \simeq e^{-\delta}}$$

Proof Ideas

- $P(\delta) = Pr[g_i(p) = g_i(q)] : P(\delta) \simeq \frac{1}{2}(e^{-\delta} + 1)$
- The expected distance of mapping two points of distance r is $\lceil (1 - P(1))d' \rceil$

$$\boxed{r' = \lceil (1 - P(1))d' \rceil}$$

We can prove desired conditions using Chernoff bounds

First Algorithm

PLEB-PreProcessing1(P, d, r)

Construct a mapping g from dimension d to d'

PLEB-PreProcessing0$(g(P), d', r')$

PLEB-Query1(q, r)

return PLEB-Query0$(g(q), r')$

$$d' = O(\frac{\log n}{\epsilon^2}), r' = \frac{1}{2}d'(1 - e^{-1})$$

- $\epsilon < 1$; Space $n^{O(\frac{1}{\epsilon^2})}$ Query time $O(\frac{d \lg n}{\epsilon^2})$
- In general; Space $n^{O(\frac{1}{\epsilon^2} + \frac{\lg(1+\epsilon)}{1+\epsilon})}$ Query time $O(\frac{d \lg n}{\min(\epsilon^2, 1)})$
- storage overhead $\rightarrow$ sublinear for large enough ϵ.

Locality-Sensitive Hashing

Given

distance measure D

Family $\mathcal{H} = \{h : X \rightarrow Y\}$ is (r_1, r_2, P_1, P_2)-sensitive for D if for all $p, q \in X$

- if $D(p,q) \leq r_1$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$
- if $D(p,q) > r_2$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$

Assume: $P_1 > P_2$ and $r_1 < r_2$

Example for Hamming metric

- $\mathcal{H} = \{h(b_1, \ldots, b_d) = b_i$, for $i = 1, \ldots, d\}$

- Above function is $(r, r(1+\epsilon), 1 - \frac{r}{d}, 1 - \frac{r(1+\epsilon)}{d})$-sensitive for Hamming metric[AMRT96]

- $Pr_{\mathcal{H}}[h(q) = h(p)] = 1 - \frac{D(p,q)}{d}$

Main Idea of the Second Algorithm

- Keeping all points in some hash tables such that with high probability closer points lie in the same bucket

- for a query point, only check the points that are in the same bucket in a hash tables

To solve ϵ-PLEB:

- Set $r_1 \leftarrow r$
- Set $r_2 \leftarrow (1+\epsilon)r$
- Define $\mathcal{C} = \{g | g(p) = h_1(p)h_2(p)\ldots h_k(p)\}$
- Hamming metric $\rightarrow$ sample of k random bits of p

Example: Hash Functions

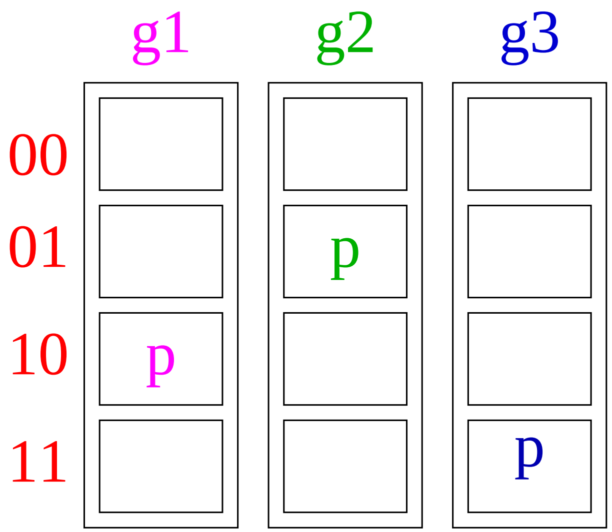
d=5, k=2, l =3

p = 01110

g1=sample bits 3,5   p = 011 10

g2=sample bits 1,2   p = 01110

g3=sample bits 3,4   p = 01110



Second Algorithm

PLEB-PreProcessing2(P, d, r)

Construct family of hash functions $g_i (=$ sample k bits)

$\forall p \in P$, store p at $g_1(p), g_2(p), \ldots, g_l(p)$

PLEB-Query2(q, r)

Find points $p_1, \ldots, p_t$ at $g_1(q), \ldots, g_l(q)$ stop after $3l$ points

Check all points p_i and return p_k if $D(p_k, q) < rc$

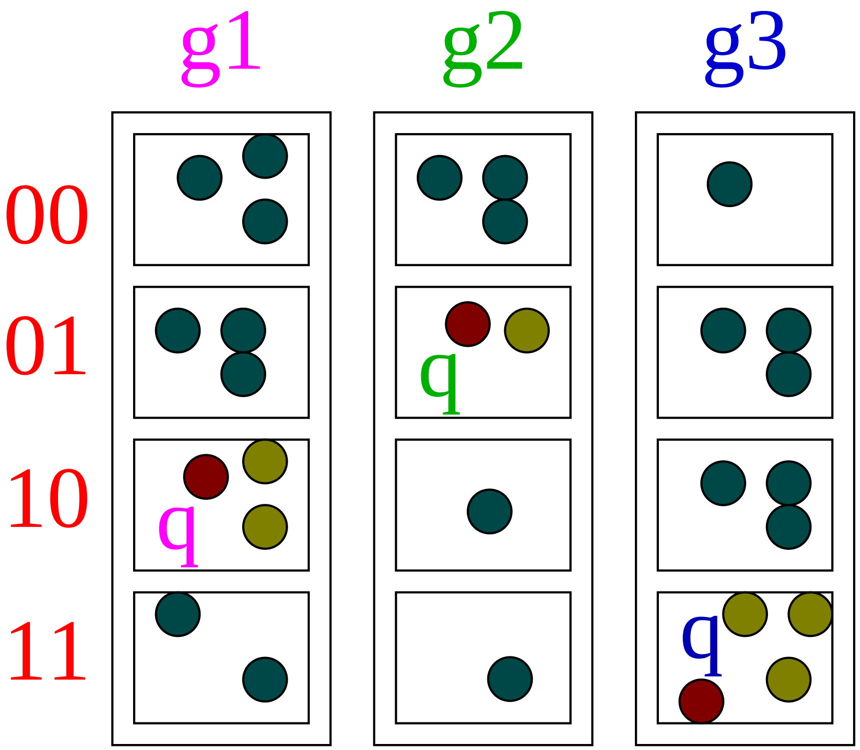
Storage: $O(dn + nl)$ Query: $O(dl)$

$k, l = ?$

Collisions

Query point q

Yellow = Collision



- Bad Collisions: collisions with distance more than rc
- Good Collisions: collisions with distance less than rc

Requirement of correctness

Define:

- p^*: a point s.t. $D(q, p^*) \leq r$
- $\text{FAR}(q) = \{p | D(q,p) > (1+\epsilon)r\}$
- $\text{BUCKET}_j(q) = \{p | g_j(p) = g_j(q)\}$

Events:

- $E_1 : \sum_{j=1}^{l} |\ \text{FAR}(q) \cap \text{BUCKET}_j(q)| \leq 3l$ $\leftarrow$ Bad Collisions
- E_2: $g_j(p^*) = g_j(q)$ for some g_j, $1 \leq j \leq l$ $\leftarrow$ Good Collision

Requirement: $Pr[E_1 \wedge E_2] > 0$

LSH Theorem

- Question: How many indices l and how many hash functions k do we need for correctness?

- Theorem: Setting $l = n^{\rho}$ for $\rho = \frac{\log 1/P_1}{\log 1/P_2}$ and $k = \log_{1/P_2} n$ is sufficient with constant probability(i.e. $Pr[E_1 \wedge E_2] > 0$).

- $\rho \simeq \frac{1}{1+\epsilon}$ for $P_1 = 1 - \frac{r}{d}$ and $P_2 = 1 - \frac{rc}{d}$ in Hamming metric.

Proof

- Claim 1: $Pr[E_1] > \frac{2}{3}$ (i.e. the number of bad collisions $\leq 3l$ with high probability)

- Claim 2: $Pr[E_2] > \frac{1}{2}$ (i.e there is a good collision with high probability)

- Thus, $Pr[E_1 \wedge E_2] > \frac{2}{3} + \frac{1}{2} - 1 > 0$

LSH: Bad Collisions

If $p \in \text{FAR}(q) \Rightarrow$

$$Pr[p \in \text{BUCKET}_j(q)] \leq P_2^k$$

For $k = \log_{1/P_2} n$

$$Pr[p \in \text{BUCKET}_j(q)] \leq P_2^{\log_{\frac{1}{P_2}} n} = \frac{1}{n}$$

$$E[|\text{FAR}(q) \cap \text{BUCKET}_j(q)|] \leq n.\frac{1}{n} = 1$$

LSH: Bad Collisions(cont')

$E[\sum_{j=1}^{l} |\ \text{FAR}(q) \cap \text{BUCKET}_j(q)|] \leq l$

By Markov inequality

$Pr[\sum_{j=1}^{l} |\ \text{FAR}(q) \cap \text{BUCKET}_j(q)| > 3l] \leq \frac{1}{3}$

$E_1 : \sum_{j=1}^{l} |\ \text{FAR}(q) \cap \text{BUCKET}_j(q)| \leq 3l$ $\leftarrow$ Bad Collisions

$\Rightarrow Pr[E_1] > \frac{2}{3}$

Good collision

- E_2: $g_j(p^*) = g_j(q)$ for some g_j, $1 \le j \le l$ $\quad\leftarrow$ Good Collision
- For any $g_j, 1 \le j \le l$

$$Pr[g_j(p^*) = g_j(q)] \ge P_1^k = P_1^{\log_{1/P_2} n} = n^{-\frac{\log 1/P_1}{\log 1/P_2}} = n^{-\rho}$$

Thus,

$$\begin{aligned} Pr[E_2] &\ge 1 - (1 - Pr[g_j(p^*) = g_j(q)])^l \\ &\ge 1 - (1 - n^{-\rho})^{n^\rho} \\ &\ge 1 - 1/e > \tfrac{1}{2} \end{aligned}$$

Second Algorithm

PLEB-PreProcessing2(P, d, r)

Construct family of hash functions $g_i (=$ sample k bits)

$\forall p \in P$, store p at $g_1(p), g_2(p), \ldots, g_l(p)$

PLEB-Query2(q, r)

Find points $p_1, \ldots, p_t$ at $g_1(q), \ldots, g_l(q)$ stop after $3l$ points

Check all points p_i and return p_k if $D(p_k, q) < rc$

Storage: $O(dn + n^{1+\frac{1}{1+\epsilon}})$ Query: $O(dn^{\frac{1}{1+\epsilon}})$

$l = n^{\frac{1}{1+\epsilon}}$

Relation to other topics

- Embedding: using this method for other metrics.

- Reducing other problems to the ϵ-PLEB and use these algorithms