Nearest Neighbor Search

$X_1 = [1\,1\,0\,0 \cdots 0\,1]$ ($\overleftrightarrow{\ d\ }$)

$X_2 = [0\,1\,0\,1\,0 \cdots 1\,0]$

$\vdots \qquad \cdots$

$X_n = [1\,0\,1\,1\,0 \cdots 1\,0]$

Given

a query vector

$q = [1\,1\,0 \cdots 1]$ ($\overleftrightarrow{\ d\ }$)

Objective:

find x_i

s.t. $\quad x_i = \underset{\forall x_j}{\operatorname{argmin}} \ dist(q, x_j)$

(assume Hamming distance is Considered)

Brute force: $O(nd)$

Question: what Type of Datastructures (indices) to build, So that queries Can be answered faster?

For Hamming Dist.: Range Trees.

Euclidean dist: Voronoi Diag.

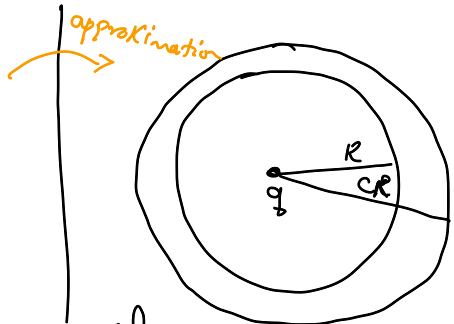
↳ $O(d \log n)$

challenge: These indices work for low-D i.e., d is a Small Constant

---

How to answer NN queries **in High D Spaces**?

↳ d is not Small



if $\exists\, x_i$ s.t. $dist(q, x_i) \le R$

return any Point x_j where

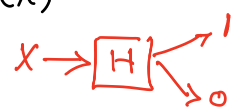
$dist(q, x_j) \le cR$

* $i \in [1, n]$

$j \in [1, n]$

Randomized Index

- A Binary hash $H: X \to \{0, 1\}$ is a function that given an input X maps it to a Binary value $H(X)$



- Given a System (X, d)

X: Universe of Possible inputs

d: distance function

a binary hash belongs to the class **Local Sensitive hash (LSH)** if

$\forall\, dist(x, y) \le R$:

$$P(H(x) = H(y)) \ge P_1$$

$\forall\, dist(x, y) \ge cR$

$$P(H(x) = H(y)) \le P_2$$

AND **$P_1 > P_2$**

for binary vectors:

$H(X) = X_\ell$: The ℓ-th bit of vector X

ℓ ↳ a random index

e.g.

$$\begin{array}{cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ X = & [1 & 0 & 0 & 1 & 0 & 1 & 1] \\ y = & [0 & 0 & 1 & 1 & 0 & 1 & 1] \end{array}$$

$H(X) = X_3$

$H(X) = 0 \quad , \quad H(Y) = 1$

$$P(H(x) \neq H(Y)) = \frac{dist(X,y)}{d}$$

$$\Rightarrow P(H(X) = H(y)) = 1 - \frac{dist(X,y)}{d}$$

$\forall\ dist(X,y) \leq R$

$$\frac{dist(X,y)}{d} \leq R/d$$

$$P(H(X) = H(y)) = 1 - \frac{dist(X,y)}{d} \geq 1 - R/d \leftarrow P_1$$

$\forall\ dist(X,y) \geq CR$

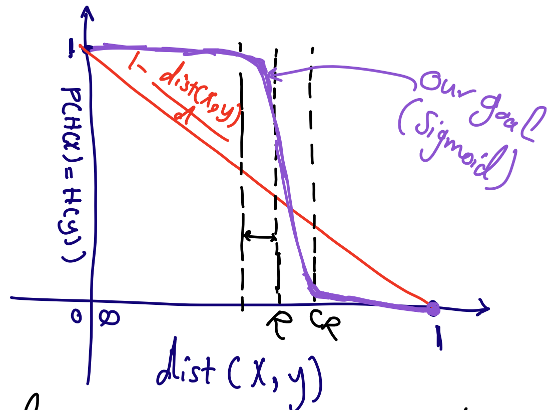
$$P(H(X) = H(y)) \leq 1 - \frac{CR}{d} \leftarrow P_2$$

$$P_1 - P_2 \overset{?}{>} 0$$

$$P_1 - P_2 = 1 - R/d - 1 + \frac{CR}{d}$$

$$= \frac{(C-1)R}{d} > 0 \quad ✓$$

$H \in LSH$



Goal: Using binary hashes from the class of LSH, build a (non-binary) hash that satisfies:

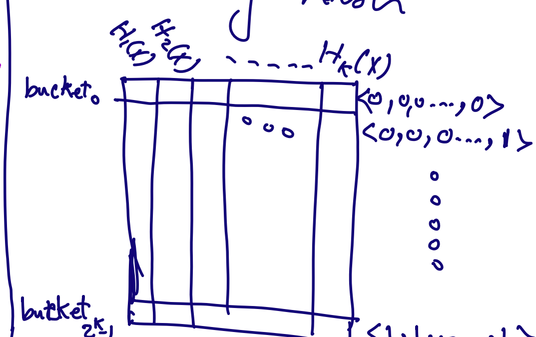
$\forall\ dist(X,y) \leq R$

$$P(H(X) = H(y)) \simeq 1$$

$\forall\ dist(X,y) \geq CR$

$$P(H(X) = H(y)) \simeq 0$$

Step 1: Build a bucket of binary hash



$$H(X) = \langle H_1(X), H_2(X), \ldots, H_K(X) \rangle$$

e.g.

$$\begin{array}{cccccccccc} & 1 & 2 & 3 & & & & & & \\ X = & [0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1] \\ y = & [0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0] \end{array}$$

$H_1 = X_2 \quad , \quad H_2 = X_5$

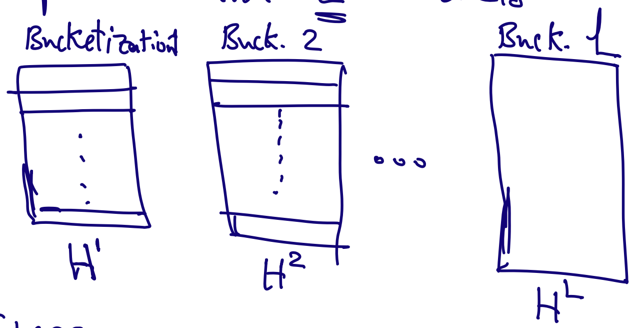
$H(X) = \langle X_2, X_5 \rangle = \langle 0, 1 \rangle$

$H(y) = \langle y_2, y_5 \rangle = \langle 1, 0 \rangle$

$P(x \text{ and } y \text{ fall into the same bucket}) = P(H(y) = H(x))$

$$= \left(1 - \frac{dist(x,y)}{d}\right)^k$$

Step 2: Build $\underline{L}$ Buckets



Step 3:

given the query point q, return any x_j that falls into the bucket of $\underline{q}$ in at least one of the bucketizations

$P(\exists B_i \text{ s.t. } H^i(x) = H^i(y)) =$

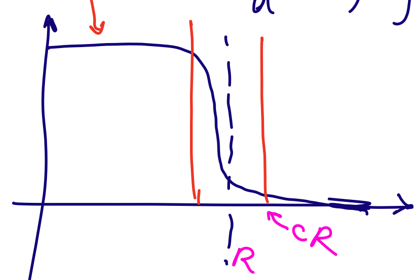
$P((H^1(x) = H^1(y)) \vee (H^2(x) = H^2(y)) \vee \cdots (H^L(x) = H^L(y)))$

$$= 1 - P\left(\bigwedge_{i=1}^{L} (H^i(x) \neq H^i(\cancel{x}))\right)$$

$$= 1 - \left(1 - \left(1 - \frac{dist(x,y)}{d}\right)^k\right)^L$$

A Sigmoid function

$$1 - \left(1 - \left(1 - \frac{dist(x,y)}{d}\right)^k\right)^L$$

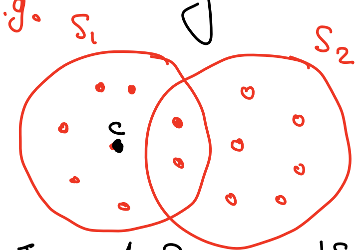


find Proper k, L

Binary Vectors: LSH-B-Hash: a random bit of vector

Set Similarity (distance)

e.g.



$$\text{Jaccard Sim.} = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

$$\text{Jaccard dist.} = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

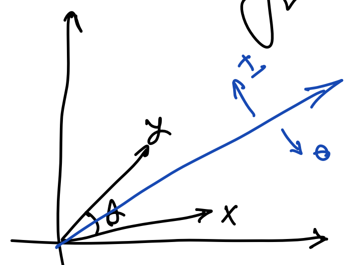
LSH-B-Hash: a random element from the Union

$$H_e(S) = \begin{cases} 1 & e \in S \\ 0 & e \notin S \end{cases}$$

e.g.: $H_e(S_1) = 1, \quad H_e(S_2) = 0$

$$P(H_e(S_1) = H_e(S_2)) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

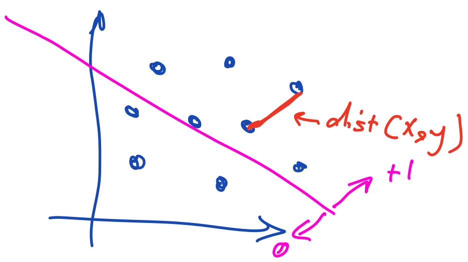
Cosine Similarity



$Sim(x,y) = Cos(\theta)$

LSH-B-Hash: a random vector, Partitioning the Space in two "half Spaces"

Euclidean Distance



LSH-B-Hash: A random Line (HyperPlane) in the Space

LSH for Dimension Reduction

$$\mathbb{R}^d \xrightarrow{t} \mathbb{R}^k$$

$x, y \in \mathbb{R}^d$

$$dist(x,y) \overset{\Delta}{\simeq} dist(t(x), t(y))$$

→ Construct a LSH of k binary hash

Intuitively: Since LSH maintains the distances relatively fixed