Nearest Neighbor Search

$X_1 = [110\ 11001]$
$\longleftrightarrow$ d bits

$\vdots$

$X_n = [0100\ 0010]$

query vector $q = [0011101]$
$\longleftrightarrow$ d bits

find x_i

s.t. $x_i = \text{argmin}\ \overset{?}{dist}(X, q)$

Hamming dist: # diff. bits.

$$dist(X_1, q) = \begin{array}{l} X_1 [1\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \\ q\ [0\ 0\ 1\ 1\ 1\ 1\ 0\ 1] \end{array}$$

$= 4$

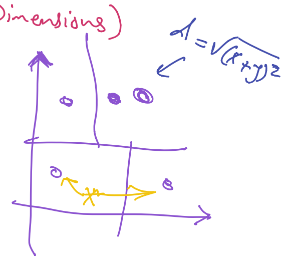
Find the nearest: $O(n\ d)$

Can we do it faster:
what data structures / Preprocessing can help?

Range Trees (low Dimensions)

$\hookrightarrow$ works for low dimensions

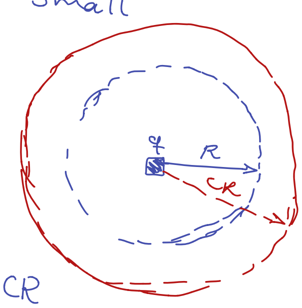
$d = O(1)$
$\hookrightarrow$ is small



How about High Dimensions (HD)

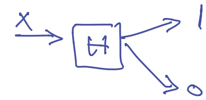
* d is not Small

if $\exists x_i$ s.t.
$dist(q, x_i) \leq R$
return
x_j s.t.
$dist(q, x_j) \leq CR$



(X, d)
$\hookrightarrow$ distance: e.g. Hamming dist.
$\hookrightarrow$ Points: e.g. Binary vectors

A (Binary) hash maps X to a (Binary) number (Bucket)



$H(X) = \begin{cases} 1 \\ 0 \end{cases}$

e.g.: for binary vectors

$$H(X) = \begin{cases} 1 & \text{if } X[0] = 1 \\ 0 & \text{if } X[0] = 0 \end{cases}$$

Given a System (X, d), a hashing H belongs to the class of **Local Sensitive Hashing (LSH)** if the followings hold for a pair of points (x, y)

if $d(x, y) \leq R$
$P(H(x) = H(y)) \geq P_1$

if $d(x, y) \geq CR$
$P(H(X) = H(y)) \leq P_2$

$P_1 > P_2$

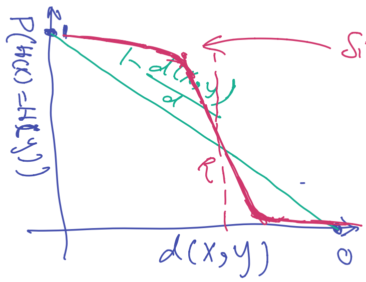
e.g.
$H(X) = X_j$: The j-th bit of X

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

if $d(x, y) \leq R$
$P(H(X) = H(y)) \geq 1 - R/d = P_1$

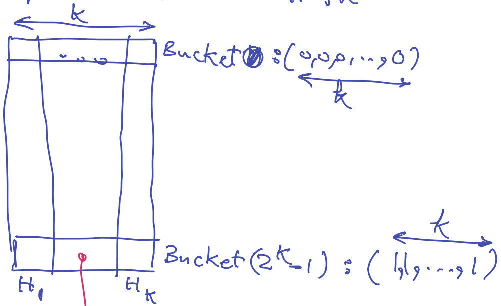
if $d(x, y) \geq CR$
$P(H(y) = H(X)) \leq 1 - CR/d = P_2$

$P_1 > P_2$ ✓ $\in$ LSH



Sigmoid function

Step 1: define K-Hash



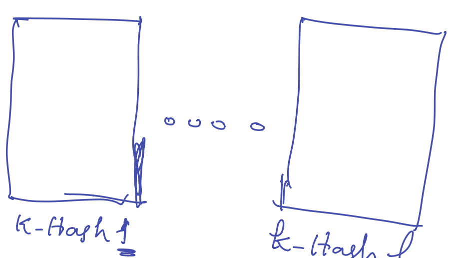
H_i is an indep. Hash function

$$P(\text{Bucket}(X) = \text{Bucket}(y)) = \left(1 - \frac{d(X,y)}{d}\right)^K$$

→ Reduced the chance of false Positive

→ Increased the " " false Negative

Step 2: Consider ℓ K-Hash Buckets



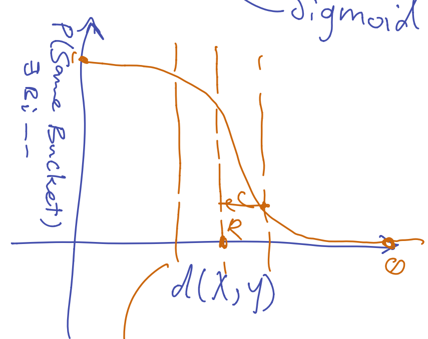
The chance of Similar points to fall into the same bucket in at least one of K-Hashes is high

$$P(B_i(X) = B_i(y)) = \left(1 - \frac{d(X,y)}{d}\right)^K$$

$$P(\nexists B_i \text{ s.t. } B_i(X) = B_i(y)) = \left(1 - \left(1 - \frac{d(X,y)}{d}\right)^K\right)^\ell$$

$$\Rightarrow P(\exists B_i \text{ s.t. } B_i(X) = B_i(y)) = 1 - \left(1 - \left(1 - \frac{d(X,y)}{d}\right)^K\right)^\ell$$

Sigmoid function

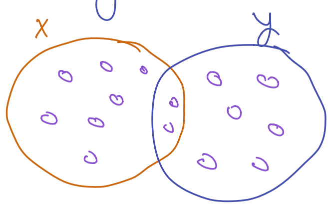


identify proper (K, ℓ) for (R, c)

Space Complexity:

$O(n\,\ell)$

Set Similarity



→ Jakard Similari-

$$S(X,y) = \frac{|X \cap y|}{|X \cup y|} \leftarrow \text{J-Sim}$$

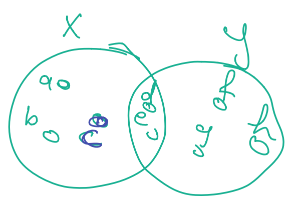
$$d(X,y) = 1 - \frac{|X \cap y|}{|X \cup y|} \leftarrow \text{J-dist.}$$

estimate the distance (Similarity) of two sets (X, y)

Hashing: Randomly Shuffle the elements (Take a random order of elements)

⇒ The index of the first element in a set is its hash

e.g.



H:

H:	c	d	h	f	a	g	b
	0	1	2	3	4	5	6

random order

$H(X) = H(c) = 0$

$H(y) = H(d) = 1$

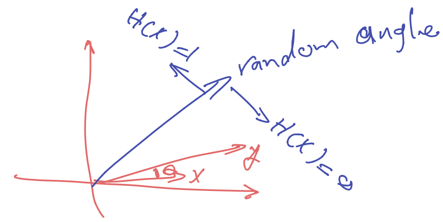
$\boxed{\text{min-hashing}}$

$$P(H(X) = H(y)) = \frac{|X \cap y|}{|X \cup y|} : \text{J-Sim}$$

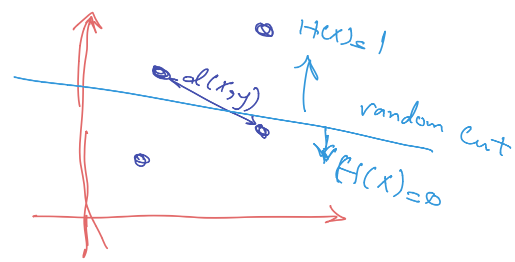
... → LSH

$\boxed{\text{Every k-hash for sets is called bottom-k Sketches}}$

dist: Cosine Sim.



dist: ℓ_2 (Euclidean dist.)



LSH for dimension Reduction

$R^d \to R^k$ s.t. distances are preserved.

→ Simply create a k-hash from a LSH family