# CS60021: Scalable Data Mining

# Similarity Search and Hashing

Sourangshu Bhattacharya

# GENERALIZATION OF LSH

# Locality Sensitive Hashing

[Indyk Motwani]

- Hash family $H$ is *locality sensitive* if

$$\Pr[h(x) = h(y)] \text{ is high if } x \text{ is close to } y$$

$$\Pr[h(x) = h(y)] \text{ is low if } x \text{ is far from } y$$

- Not clear such functions exist for all distance functions

# Locality sensitive hashing

- Originally defined in terms of a similarity function [C'02]

- Given universe $U$ and a similarity $s: U \times U \to [0,1]$ , does there exist a prob distribution over some hash family $H$ such that

$$\Pr_{h \in H}[h(x) = h(y)] = s(x,y)$$

$$s(x,y) = 1 \;\to x = y$$
$$s(x,y) = s(y,x)$$

# Hamming distance

- Points are bit strings of length $d$
- $H(x, y) = |\{i, x_i \neq y_i\}|$

# Hamming distance

- Points are bit strings of length $d$
- $H(x,y) = |\{i, x_i \neq y_i\}|$ $\quad S_H(x,y) = 1 - \frac{H(x,y)}{d}$
  - $x = 1011010001, y = 0111010101$
  - $H(x,y) = 3 \quad S_H(x,y) = 1 - \frac{3}{10} = 0.7$

# Hamming distance

- Points are bit strings of length $d$
- $H(x, y) = |\{i, x_i \neq y_i\}|$ $S_H(x, y) = 1 - \frac{H(x,y)}{d}$
- Define a hash function $h$ by sampling a set of positions
  - $x = 1011010001, y = 0111010101$
  - $S = \{1,5,7\}$
  - $h(x) = 100, h(y) = 100$

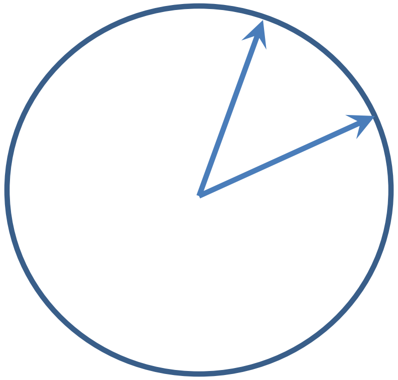
# Existence of LSH

- The above hash family is locality sensitive, $k = |S|$

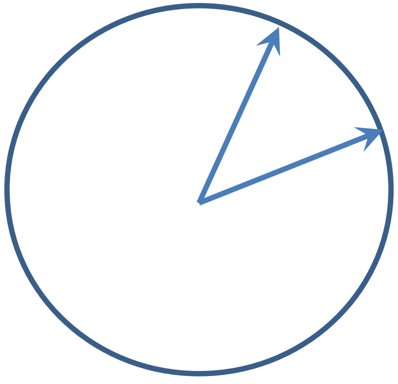
$$\Pr[h(x) = h(y)] = \left(1 - \frac{H(x, y)}{d}\right)^k$$

# LSH for angle distance

- $x, y$ are unit norm vectors
- $d(x, y) = \cos^{-1}(x \cdot y) = \theta$
- $S(x, y) = 1 - \theta/\pi$



# LSH for angle distance

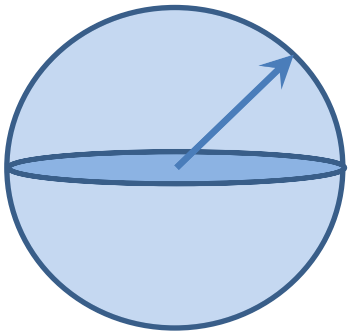


- $x, y$ are unit norm vectors
- $d(x, y) = \cos^{-1}(x \cdot y) = \theta$
- $S(x, y) = 1 - \theta/\pi$

- Choose direction $v$ uniformly at random
  - $h_v(x) = sign(v \cdot x)$
  - $\Pr[h_v(x) = h_v(y)] = 1 - \theta/\pi$

# Aside: picking a direction u.a.r.

- How to sample a vector $x \in R^d, |x|_2 = 1$ and the direction is uniform among all possible directions



- Generate $x = (x_1, \ldots . x_d), x_i \sim N(0,1)$ iid
- Normalize $\frac{x}{|x|_2}$
  - By writing the pdf of the d-dimensional Gaussian in polar form, easy to see that this is uniform direction on unit sphere

# Jaccard distance: minhashing

- Pick a uniform permutation of the element universe $U$
- For any set $S$,
  - $h(S) = min_{x \in S} h(x)$

- Often easier to visualize if we think of the collection of sets as a $\{0,1\}$ matrix

# Example

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| A | 1 | 0 | 1 | 0 |
| B | 1 | 0 | 0 | 1 |
| C | 0 | 1 | 0 | 1 |
| D | 0 | 1 | 0 | 1 |
| E | 0 | 1 | 0 | 1 |
| F | 1 | 0 | 1 | 0 |
| G | 1 | 0 | 1 | 0 |

| |
|---|
| A |
| C |
| G |
| F |
| B |
| E |
| D |

| | | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| 1 | A | 1 | 0 | 1 | 0 |
| 2 | C | 0 | 1 | 0 | 1 |
| 3 | G | 1 | 0 | 1 | 0 |
| 4 | F | 1 | 0 | 1 | 0 |
| 5 | B | 1 | 0 | 0 | 1 |
| 6 | E | 0 | 1 | 0 | 1 |
| 7 | D | 0 | 1 | 0 | 1 |

| | | | |
|---|---|---|---|
| 1 | 2 | 1 | 2 |

[Slide from Evimaria Terzi]

# Example

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| **A** | 1 | 0 | 1 | 0 |
| **B** | 1 | 0 | 0 | 1 |
| **C** | 0 | 1 | 0 | 1 |
| **D** | 0 | 1 | 0 | 1 |
| **E** | 0 | 1 | 0 | 1 |
| **F** | 1 | 0 | 1 | 0 |
| **G** | 1 | 0 | 1 | 0 |

→

| |
|---|
| **D** |
| **B** |
| **A** |
| **C** |
| **F** |
| **G** |
| **E** |

→

| | | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|---|
| 1 | **D** | 0 | 1 | 0 | 1 |
| 2 | **B** | 1 | 0 | 0 | 1 |
| 3 | **A** | 1 | 0 | 1 | 0 |
| 4 | **C** | 0 | 1 | 0 | 1 |
| 5 | **F** | 1 | 0 | 1 | 0 |
| 6 | **G** | 1 | 0 | 1 | 0 |
| 7 | **E** | 0 | 1 | 0 | 1 |
| | | 2 | 1 | 3 | 1 |

# Why is this LSH?

- For sets $S$ and $T$,
  - The first row where one of the two has a 1 belong to $S \cup T$
  - We have equality $h(S) = h(T)$, only if both the rows contain 1
  - This means that this row belongs to $S \cap T$

- Hence, the event $h(S) = h(T)$ is same as the event that a row in $S \cap T$ appears first among all rows in $S \cup T$

$$\Pr[h(S) = h(T)] = \frac{|S \cap T|}{|S \cup T|}$$

# Aside: How to choose random permutations

- Picking a uniform at random permutation is expensive
- In theory, need to choose from a family of min-wise independent permutations

- In practice, can use standard hash functions, hash all the values and then sort

# Which similarities admit LSH?

- There are various similarities and distance that are used in scientific literature
  - Encyclopedia of distances DL'11

- Will there be an LSH for each one of them?
  - Similarity is LSHable if there exists an LSH for it

[slide courtesy R. Kumar]

# LSHable similarities

- <u>Thm</u>: S is LSHable → 1 – S is a metric

$$d(x, y) = 0 \;\rightarrow x = y$$
$$d(x, y) = d(y, x)$$
$$d(x, y) + d(y, z) \geq d(x, z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A, B) = [h(A) \neq h(B)]$$
$$1 \; - \mathrm{S}(\mathrm{A}, \mathrm{B}) = \Pr_h[\Delta_h(A, B)]$$

# LSHable similarities

- Thm: S is LSHable → 1 – S is a metric

$$d(x,y) = 0 \;\rightarrow x = y$$
$$d(x,y) = d(y,x)$$
$$d(x,y) + d(y,z) \geq d(x,z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A,B) = [h(A) \neq h(B)]$$

# LSHable similarities

- Thm: S is LSHable → 1 – S is a metric

$$d(x,y) = 0 \rightarrow x = y$$
$$d(x,y) = d(y,x)$$
$$d(x,y) + d(y,z) \geq d(x,z)$$

Fix hash function $h \in H$ and define

$$\Delta_h(A,B) = [h(A) \neq h(B)]$$
$$1 - \mathrm{S}(\mathrm{A},\mathrm{B}) = \Pr_h[\Delta_h(A,B)]$$

Also

$$\Delta_h(A,B) + \Delta_h(B,C) \geq \Delta_h(A,C)$$

# Example of non-LSHable similarities

- $d(A, B) = 1 - s(A, B)$
- Sorenson-Dice : $s(A, B) = \frac{2|A \cap B|}{|A| + |B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a, b\}$
  - $s(A, B) = 0, s(B, C) = s(A, C) = 2/3$

# Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a,b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

# Example of non-LSHable similarities

- $d(A,B) = 1 - s(A,B)$
- Sorenson-Dice : $s(A,B) = \frac{2|A \cap B|}{|A|+|B|}$
  - Ex: $A = \{a\}, B = \{b\}, C = \{a, b\}$
  - $s(A,B) = 0, s(B,C) = s(A,C) = \frac{2}{3}$
- Overlap: $s(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)}$
  - $s(A,B) = 0, s(A,C) = 1 = s(B,C)$

**These similarities are not LSHable**

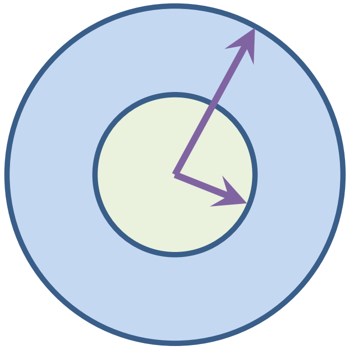
# Gap Definition of LSH

IMRS'97, IM'98, GIM'99

- A family is $(r, R, p, P)$ LSH if

$$\Pr_{h \in H}[h(x) = h(y)] \geq p \; if \; d(x, y) \leq r$$

$$\Pr_{h \in H}[h(x) = h(y)] \leq P \; if \; d(x, y) \geq R$$



# Gap LSH

- All the previous constructions satisfy the gap definition
  - Ex: for $JS(S,T) = \frac{|S \cap T|}{|S \cup T|}$

$$JD(S,T) \leq r \rightarrow JS(S,T) \geq 1 - r \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \geq 1 - r$$

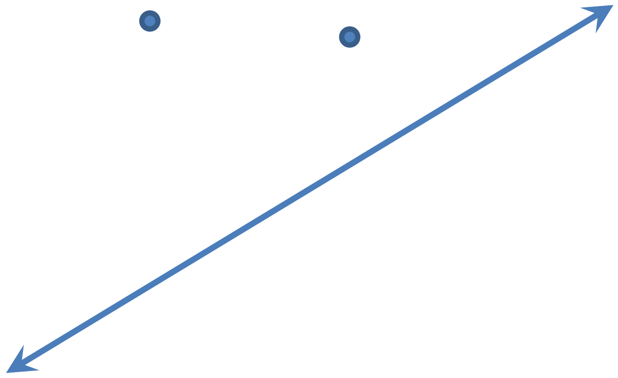
$$JD(S,T) \geq R \rightarrow JS(S,T) \leq 1 - R \rightarrow \Pr[h(S) = h(T)] = JS(S,T) \leq 1 - R$$

Hence is a $(r, R, 1 - r, 1 - R)$ LSH

# L2 norm

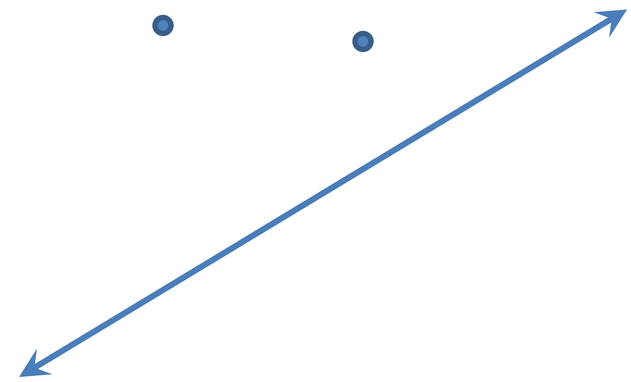
- $d(x, y) = \sqrt{(\sum_i (x_i - y_i)^2}$
- $u =$ random unit norm vector, $w \in R$ parameter, $b \sim Unif[0, w]$

- $h(x) = \lfloor \frac{u \cdot x + b}{w} \rfloor$



# L2 norm

- $d(x, y) = \sqrt{(\sum_i (x_i - y_i)^2}$
- $u$ = random unit norm vector, $w \in R$ parameter, $b \sim Unif[0, w]$
- $h(x) = \lfloor \frac{u \cdot x + b}{w} \rfloor$
- If $|x - y|_2 < \frac{w}{2}$, $\Pr[h(x) = h(y)] \geq \frac{1}{3}$
- If $|x - y|_2 > 4w$, $\Pr[h(x) = h(y)] \leq \frac{1}{4}$



# Solving the near neighbour

- $(r, c)$ $-$near neighbour problem
  - Given query point $q$, return all points $p$ such that $d(p, q) < r$ and none such that $d(p, q) > cr$
  - Solving this gives a subroutine to solve the "nearest neighbour", by building a data-structure for each $r$ , in powers of $(1 + \epsilon)$

# How to actually use it?

- Need to amplify the probability of collisions for “near” points

# Band construction

- AND-ing of LSH
  - Define a composite function $H(x) = (h_1(x), \dots h_k(x))$
  - $\Pr[H(x) = H(y)] = \Pi_{\mathrm{i}} \Pr[h_i(x) = h_i(y)] = \Pr[h_1(x) = h_1(y)]^k$

# Band construction

- AND-ing of LSH
  - Define a composite function $H(x) = (h_1(x), \dots h_k(x))$
  - $\Pr[H(x) = H(y)] = \Pi_{\mathrm{i}} \Pr[h_i(x) = h_i(y)] = \Pr[h_1(x) = h_1(y)]^k$

- OR-ing
  - Create $L$ independent hash-tables for $H_1, H_2, \dots H_L$
  - Given query $q$, search in $\cup_j H_j(q)$

# Example

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|---|---|---|---|---|
| **A** | 1 | 0 | 1 | 0 |
| **B** | 1 | 0 | 0 | 1 |
| **C** | 0 | 1 | 0 | 1 |
| **D** | 0 | 1 | 0 | 1 |
| **E** | 0 | 1 | 0 | 1 |
| **F** | 1 | 0 | 1 | 0 |
| **G** | 1 | 0 | 1 | 0 |

| | S1 | S2 | S3 | S3 |
|---|---|---|---|---|
| h1 | 1 | 2 | 1 | 2 |
| h2 | 2 | 1 | 3 | 1 |

| | S1 | S2 | S3 | S3 |
|---|---|---|---|---|
| h3 | 3 | 1 | 2 | 1 |
| h4 | 1 | 3 | 2 | 2 |

# Why is this better?

- Consider $q, y$ with $\Pr[h(q) = h(y)] = 1 - d(x, y)$
- Probability of not finding $y$ as one of the candidates in $\cup_j H_j(q)$

$$1 - \left(1 - (1 - d)^k\right)^L$$

# Creating an LSH

- If we have a $(r, cr, p, q)$ LSH
- For any $y$, with $|q - y| < r$,
  - Prob of $y$ as candidate in $\cup_j H_j(q) \geq 1 - \left(1 - p^k\right)^L$
- For any $z$, $|q - z| > cr$,
  - Prob of $z$ as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k$

# Creating an LSH

$\rho = \frac{\log(p)}{\log(q)}$ $L = n^{\rho}$ $k = \log(n)/\log\left(\frac{1}{q}\right)$

- If we have a $(r, cr, p, q)$ LSH
- For any $y$, with $|q - y| < r$,
  - Prob of $y$ as candidate in $\cup_j H_j(q) \geq 1 - \left(1 - p^k\right)^L \geq 1 - \frac{1}{e}$
- For any $z$, $|q - z| > cr$,
  - Prob of $z$ as candidate in any fixed $H_j(q) \leq q^k$
  - Expected number of such $z \leq Lq^k \leq L = n^{\rho}$

# Runtime

- Space used = $n^{1+\rho}$
- Query time = $n^{\rho}$

- We can show that for Hamming, angle etc, $\rho \approx \frac{1}{c}$
  - Can get 2-approx near neighbors in $O\left(\sqrt{n}\right)$ query time

# LSH: theory vs practice

- In order to design LSH in practice, the theoretical parameter values are only a guidance
  - Typically need to search over the parameter space to find a good operating point
  - Data statistics can provide some guidance (will see in next class)

# Summary

- Locality sensitive hashing is a powerful tool for near neighbour problems
- Trades off space with query time
- Practical for medium to large datasets with fairly large number of dimensions
  - However, doesn’t really work very well for sparse, very very high dimensional datasets
- LSH and extensions are an area of active research and practice

# References:

- Primary references for this lecture
  - Modern Massive Datasets, Rajaraman, Leskovec, Ullman.
  - Survey by Andoni et al. (CACM 2008) available at www.mit.edu/~andoni/LSH