Locality Sensitive Hashing

- Locality Sensitive Hashing for proximity problems
- LSH for Hamming distance
- AND-OR and OR-AND Composition of LSH
- LSH Scheme and the 'S' curve
- Non-LSH-able distance measures
- LSH for Jaccard distance
- LSH for Cosine distance
- LSH for Euclidean distance
- Data dependent LSH

Imdad ullah Khan

LSH for Proximity Problems

Dictionary ADT

Dictionary: Abstract Data Type

Given n items pre-process and store to support INSERT, SEARCH, DELETE

Varying operations wise complexity with different implementations

- Array
- Sorted Array
- Linked List
- Sorted Linked List
- Binary Search Tree
- Balanced Binary Search Tree
- Hash functions

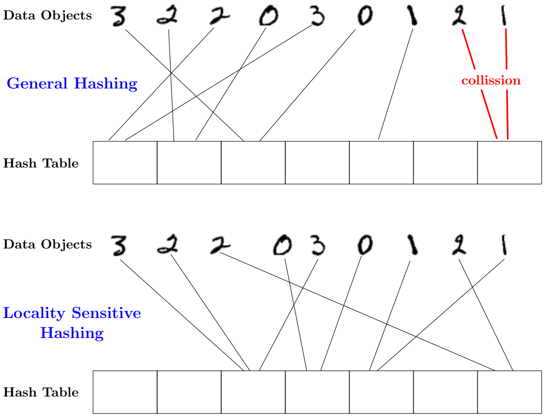
Approaches for nearest neighbor

Hashing works best for duplicate detection not for near duplicate detection

- Array ▷ works for $m = 1$
- Sorted Array ▷ works for $m = 1$
- Voronoi Diagram ▷ works for $m = 2$
- *kd*-tree ▷ works for $m \leq 10$ or 12

Locality Sensitive Hashing

- Need hash functions where **meaningful collisions are desired**
- Want similar objects hash to same buckets

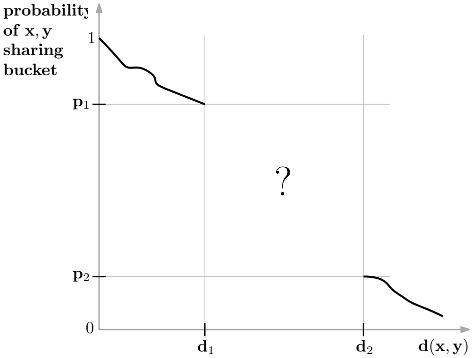


Locality Sensitive Hashing

A family $\mathcal{F} = \{h_1, h_2, \ldots, \}$ is a (d_1, d_2, p_1, p_2)-family of LSH functions, if

For a randomly chosen function h from $\mathcal{F}$, for objects $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x}, \mathbf{y}) \leq d_1$**, then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq p_1$
- **If** $d(\mathbf{x}, \mathbf{y}) \geq d_2$**, then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq p_2$



Using LSH for nearest neighbor query

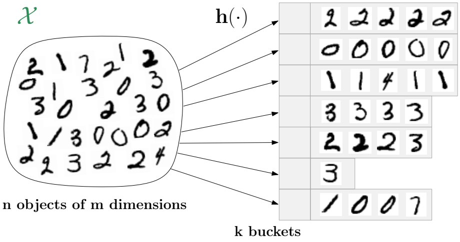
A family $\mathcal{F} = \{h_1, h_2, \ldots, \}$ is a (d_1, d_2, p_1, p_2)-family of LSH functions, if

For a randomly chosen function h from $\mathcal{F}$, for objects $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x}, \mathbf{y}) \leq d_1$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq p_1$
- **If** $d(\mathbf{x}, \mathbf{y}) \geq d_2$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq p_2$

Find k-NN of $\mathbf{q}$ in a dataset X

- Pick a random h from $\mathcal{F}$ and compute $h(\mathbf{x})$ for all $\mathbf{x} \in X$
- Compute $h(\mathbf{q})$ and find $NN(\mathbf{q})$ among objects in bucket $h(\mathbf{q})$



Using LSH for nearest neighbor query

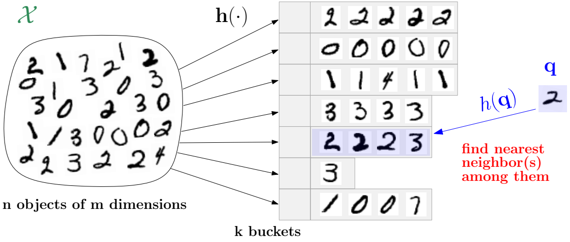
A family $\mathcal{F} = \{h_1, h_2, \dots, \}$ is a (d_1, d_2, p_1, p_2)-family of LSH functions, if

For a randomly chosen function h from $\mathcal{F}$, for objects $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x}, \mathbf{y}) \leq d_1$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq p_1$
- **If** $d(\mathbf{x}, \mathbf{y}) \geq d_2$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq p_2$

Find docs within distance r of $\mathbf{q}$ in a dataset X

- Pick a random h from $\mathcal{F}$ and compute $h(\mathbf{x})$ for all $\mathbf{x} \in X$
- Compute $h(\mathbf{q})$ and find $NN(\mathbf{q})$ among objects in bucket $h(\mathbf{q})$

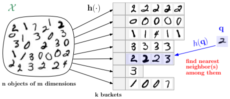


Using LSH for nearest neighbor

- $1M$ docs each of length 1000 (e.g. TF-IDF)
- For a query $\mathbf{q}$ find docs with $d(\bullet, \mathbf{q}) \leq .1$ ▷ Naive approach: $\sim 10^9$ ops
- Use random h from $\mathcal{F}$ of $(.15, .4, .8, .2)$-LSH family ▷ Naive approach on $h(\mathbf{q})$ only

For two docs $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x}, \mathbf{y}) \leq .15$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq .8$
- **If** $d(\mathbf{x}, \mathbf{y}) \geq .40$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq .2$



- False negatives (FN): $d(\bullet, \mathbf{q}) < 0.1 \;\wedge\; h(\bullet) \neq h(\mathbf{q})$
  ▷ qualitative error, missed near neighbor
- False positives (FP): $d(\bullet, \mathbf{q}) > 0.1 \;\wedge\; h(\bullet) = h(\mathbf{q})$
  ▷ wasted/unnecessary distance computation

- $E\big[FN\big] \;<\; E\big[|\{(\mathbf{x}, \mathbf{y}) \;:\; d(\mathbf{x}, \mathbf{y}) \leq .15 \;\wedge\; h(\mathbf{x}) \neq h(\mathbf{y})\}|\big] \;\leq\; 20\%$
- $E\big[|\{(\mathbf{x}, \mathbf{y}) \;:\; d(\mathbf{x}, \mathbf{y}) \geq .4 \;\wedge\; h(\mathbf{x}) = h(\mathbf{y})|\}\big] \;\leq\; 20\%$
- On average $\leq 20\%$ missed near nbrs and hopefully small wasted computation

Using LSH for near duplicates detection

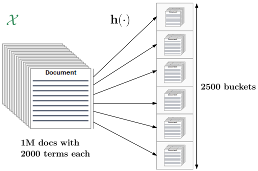
- $1M$ docs each of length 2000 (e.g. TF-IDF)
- Find near duplicates: $sim(\cdot,\cdot) \geq 0.9$ $\left[d(\cdot,\cdot) \leq .1\right]$ ▷ bruteforce $\binom{1M}{2}$ $d()$ $\sim 10^{15}$ ops
- Use random h from $\mathcal{F}$ of $(.15, .4, .8, .2)$-LSH family

For two docs $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x},\mathbf{y}) \leq .15$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq .8$
- **If** $d(\mathbf{x},\mathbf{y}) \geq .40$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq .2$

- Assume all functions in $\mathcal{H}$ are of the form $h : \mathbb{R}^n \mapsto [2500]$ ▷ bucket IDs
- Assume functions in $\mathcal{H}$ maps docs to the 2500 buckets almost uniformly

▷ unrealistic assumption, LSH gives no such guarantee



Algorithm:

Compute distance b/w pairs in each bucket

Output the pair if distance $< .1$

Runtime:

$2500 \times \binom{400}{2}$ $d(\cdot,\cdot)$ computation ▷ $2500\times$ faster

Using LSH for near duplicates detection

- $1M$ docs each of length 2000 (e.g. TF-IDF)
- Find near duplicates pairs with $sim(\cdot,\cdot) \geq 90\% = 0.9$ $\left[d(\cdot,\cdot) \leq .1\right]$
- Use random h from $\mathcal{F}$ of $(.15, .4, .8, .2)$-family of LSH functions

For two docs $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x},\mathbf{y}) \leq .15$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq .8$
- **If** $d(\mathbf{x},\mathbf{y}) \geq .40$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq .2$

- Assume all function in $\mathcal{H}$ are of the form $h : \mathbb{R}^n \mapsto [2500]$ ▷ bucket IDs
- Assume functions in $\mathcal{H}$ maps docs to the 2500 buckets almost uniformly
- Naive approach $\rightarrow \sim 10^{15}$ ops LSH approach $\rightarrow 4 \times 10^{11}$ ops
- False positives (FP): $d(\mathbf{x},\mathbf{y}) > 0.1 \ \wedge \ h(\mathbf{x}) = h(\mathbf{y})$ ▷ wasted comput.
- False negatives (FN): $d(\mathbf{x},\mathbf{x}) \leq 0.1 \ \wedge \ h(\mathbf{x}) \neq h(\mathbf{y})$ ▷ qualitative error

- $E\left[FN\right] < E\left[\left|\{(\mathbf{x},\mathbf{y}) : d(\mathbf{x},\mathbf{y}) \leq .15 \wedge h(\mathbf{x}) \neq h(\mathbf{y})\}\right|\right] \leq 20\%$
- $E\left[\left|\{(\mathbf{x},\mathbf{y}) : d(\mathbf{x},\mathbf{y}) \geq .4 \wedge h(\mathbf{x}) = h(\mathbf{y})\right|\}\right] \leq 20\%$
- On average $\leq 20\%$ missed near dups and hopefully small wasted computation

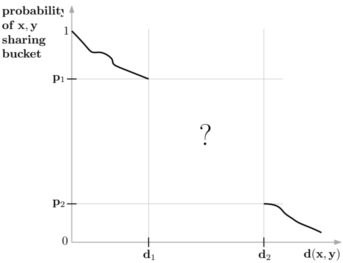
Locality Sensitive Hashing

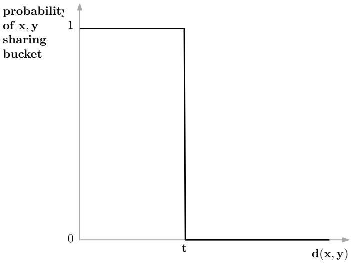
A family $\mathcal{H} = \{h_1, h_2, \dots, \}$ is a (d_1, d_2, p_1, p_2)-family of LSH functions, if

For a randomly chosen function h from $\mathcal{H}$, for objects $\mathbf{x}$ and $\mathbf{y}$

- **If** $d(\mathbf{x}, \mathbf{y}) \leq d_1$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq p_1$
  - We want p_1 to be close to 1 ▷ to reduce false negative
- **If** $d(\mathbf{x}, \mathbf{y}) \geq d_2$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq p_2$
  - We want p_2 to be close to 0 ▷ to reduce false positive
- We want d_1 and d_2 both to be close to t (near duplicates threshold)

▷ to reduce the range of distances with no guarantees



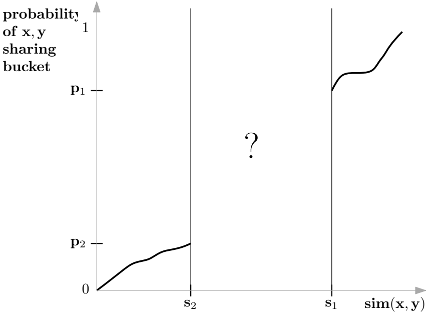


Equivalent definition of LSH functions in terms of similarity

A family $\mathcal{H} = \{h_1, h_2, \ldots, \}$ is a (s_1, s_2, p_1, p_2)-family of LSH functions, if

For a randomly chosen function h from $\mathcal{H}$, for objects **x** and **y**

- **If** $sim(\mathbf{x}, \mathbf{y}) \geq s_1$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq p_1$
- **If** $sim(\mathbf{x}, \mathbf{y}) \leq s_2$, **then** $Pr[h(\mathbf{x}) = h(\mathbf{y})] \leq p_2$



Bit-Sampling: LSH for Hamming distance

Hamming Distance and Similarity

- Hamming distance: used for fixed-length character vectors
- Coordinates values from a finite (usually small) set called **alphabet**
- Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ between two n-vectors $\mathbf{x}$ and $\mathbf{y}$ is the number of coordinates in which they differ
- $0 \leq d_H(\mathbf{x}, \mathbf{y}) \leq n$ and it is a distance metric
- Hamming similarity: $s_H = n - d_H(x, y)$

We use $d_H(x, y) = \dfrac{\text{number of coordinates different in } \mathbf{x} \text{ and } \mathbf{y}}{n \text{ (total number of bits in } \mathbf{x} \text{ and } \mathbf{y})}$

- Similarity in this setting $s_H(\mathbf{x}, \mathbf{y}) = 1 - d_H(\mathbf{x}, \mathbf{y})$
- When contextually clear, we drop subscript from $s_H(\mathbf{x}, \mathbf{y})$ and $d_H(\mathbf{x}, \mathbf{y})$

bit-sampling: LSH for Hamming distance

- $\mathcal{F}$: a family of LSH functions for $d_H(\cdot,\cdot)$ between n-bits strings
- Each $h \in \mathcal{F}$ is of the form $h : \{0,1\}^n \mapsto \{0,1\}$
- $\mathcal{F} \ = \ \{h_i : \ 1 \leq i \leq n\}$ $\qquad \triangleright \ |\mathcal{F}| = n$

$$h_i(\mathbf{x}) \ := \ h_i\left(b_1, b_2, \ldots, b_n\right) \ := \ b_i$$

$h_1(10101011) = 1 \quad h_1(00110011) = 0 \quad h_2(10101011) = 0 \quad h_3(10101011) = 1$

	1	2	3	4	5	6	7	8	9
$\mathbf{x}$	1	0	1	1	0	1	1	0	0

$\mathbf{h}_1(\mathbf{x}) = 1$ $\quad$ $\mathbf{h}_5(\mathbf{x}) = 0$ $\quad$ $\mathbf{h}_8(\mathbf{x}) = 0$

$\mathbf{h}_1(\mathbf{y}) = 0$ $\quad$ $\mathbf{h}_5(\mathbf{y}) = 0$ $\quad$ $\mathbf{h}_8(\mathbf{y}) = 1$

	1	2	3	4	5	6	7	8	9
$\mathbf{y}$	0	1	1	0	0	1	0	1	0

bit-sampling: LSH for Hamming distance

- $\mathcal{F}$: a family of LSH functions for $d_H(\cdot,\cdot)$ between n-bits strings
- Each $h \in \mathcal{F}$ is of the form $h : \{0,1\}^n \mapsto \{0,1\}$
- $\mathcal{F} \ = \ \{h_i :\ 1 \leq i \leq n\}$ $\triangleright\ |\mathcal{F}| = n$

$$h_i(\mathbf{x}) \ := \ h_i\left(b_1, b_2, \ldots, b_n\right) \ := \ b_i$$

$\mathcal{F}$ is a $(r_1, r_2, 1-r_1, 1-r_2)$-LSH family

- Choose a random function form $\mathcal{F} \leftrightarrow$ choose a random index from $[n]$
- $d(\mathbf{x},\mathbf{y}) \leq r_1$ means that $\mathbf{x}$ and $\mathbf{y}$ agree on $\geq (1-r_1)n$ bits
- $Pr[\text{choose } h_i \text{ such that } \mathbf{x}_i = \mathbf{y}_i] \ \geq \ \frac{(1-r_1)n}{n} \ = \ 1 - r_1$
- $d(\mathbf{x},\mathbf{y}) \geq r_2$ means that $\mathbf{x}$ and $\mathbf{y}$ agree on $\leq (1-r_2)n$ bits
- $Pr[\text{choose } h_i \text{ such that } \mathbf{x}_i = \mathbf{y}_i] \ \leq \ \frac{(1-r_2)n}{n} \ = \ 1 - r_2$

Theory of LSH and LSH Scheme

LSH Working

- Candidate pair: Two data items that hash to the same buckets
- Working of a LSH function:
    - Input: $\mathbf{x}$ and $\mathbf{y}$
    - Output: **Yes** a candidate pair or **No**
- $h(\mathbf{x}) = h(\mathbf{y})$ means h declares $\mathbf{x}$ and $\mathbf{y}$ a candidate pair
- We will not go into the detail of how it computes the value
- Values of $h(\mathbf{x})$ and $h(\mathbf{y})$ (bucket IDs) are irrelevant ▷ just check equality
- False negative (FN): $d(\mathbf{x}, \mathbf{y}) \leq t$ (nearest neighbors) but $h(\mathbf{x}) \neq h(\mathbf{y})$
- False positive (FP): $d(\mathbf{x}, \mathbf{y}) > t$ (not nearest neighbors) but $h(\mathbf{x}) = h(\mathbf{y})$
- Also no notion of absolute locality sensitive hashing ▷ only parametric

LSH: Probability Amplification

- Parameters of a LSH family may not be good enough for application
- Use probability amplification (independent trials) to adjust parameters
- Manipulate $\mathcal{H}$ to bound number of FP and FN into desired range
- Dealing with False Positives:
  - Use many independent hash functions from $\mathcal{H}$
  - Consider pairs that are declared candidate by **ALL** of them $\triangleright$ **AND**
  - Dissimilar vectors are less likely to become candidate pair
- Dealing with False Negatives:
  - Use many independent hash functions from $\mathcal{H}$
  - Consider pairs that are declared candidate by **ANY** of them $\triangleright$ **OR**
  - Similar vectors are more likely to become candidate pair

Constructing new LSH families from old

Applying the AND construction to (s_1, s_2, p_1, p_2)-LSH family $\mathcal{F}$:

- Each h' in new family $\mathcal{F}'$ consists of r functions $h_{i1}, h_{i2}, \ldots, h_{ir}$ from $\mathcal{F}$
- $h'_i = \{h_{i1}, h_{i2}, \ldots, h_{ir}\} \in \mathcal{F}'$ works as follows $\triangleright$ $|\mathcal{F}'| = \binom{n}{r}$

$$h'_i(\mathbf{x}) = h'_i(\mathbf{y}) \iff h_{i1}(\mathbf{x}) = h_{i1}(\mathbf{y}) \wedge h_{i2}(\mathbf{x}) = h_{i2}(\mathbf{y}) \wedge \ldots \wedge h_{ir}(\mathbf{x}) = h_{ir}(\mathbf{y})$$

- $h' \in \mathcal{F}'$ only declares a candidate pair if all r functions from $\mathcal{F}$ do

	1	2	3	4	5	6	7	8	9
$\mathbf{x}$	1	0	1	1	0	1	1	0	0

$\mathbf{h}'_1 = \{\mathbf{h}_2, \mathbf{h}_5, \mathbf{h}_7\}$

$\mathbf{h}'_2 = \{\mathbf{h}_1, \mathbf{h}_4, \mathbf{h}_8\}$ $\mathbf{h}'_1(\mathbf{x}) \neq \mathbf{h}'_1(\mathbf{y})$ $\mathbf{h}'_2(\mathbf{x}) \neq \mathbf{h}'_2(\mathbf{y})$ $\mathbf{h}'_3(\mathbf{x}) = \mathbf{h}'_3(\mathbf{y})$

$\mathbf{h}'_3 = \{\mathbf{h}_6, \mathbf{h}_9\}$

	1	2	3	4	5	6	7	8	9
$\mathbf{y}$	0	1	1	0	0	1	0	1	0

Constructing new LSH families from old

Applying the AND construction to (s_1, s_2, p_1, p_2)-LSH family $\mathcal{F}$:

- Each h' in new family $\mathcal{F}'$ consists of r functions $h_{i1}, h_{i2}, \ldots, h_{ir}$ from $\mathcal{F}$
- $h'_i = \{h_{i1}, h_{i2}, \ldots, h_{ir}\} \in \mathcal{F}'$ works as follows $\triangleright$ $|\mathcal{F}'| = \binom{n}{r}$

$$h'_i(\mathbf{x}) = h'_i(\mathbf{y}) \iff h_{i1}(\mathbf{x}) = h_{i1}(\mathbf{y}) \wedge h_{i2}(\mathbf{x}) = h_{i2}(\mathbf{y}) \wedge \ldots \wedge h_{ir}(\mathbf{x}) = h_{ir}(\mathbf{y})$$

- $h' \in \mathcal{F}'$ only declares a candidate pair if all r functions from $\mathcal{F}$ do

$\mathcal{F}'$ is a (s_1, s_2, p_1^r, p_2^r)-family of LSH functions

- Choose $h'_i \in \mathcal{F}' \iff$ Choose r functions $\{h_{i1}, h_{i2}, \ldots, h_{ir}\}$ in $\mathcal{F}$
- $s(\mathbf{x}, \mathbf{y}) \geq s_1 \implies Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \geq p_1$ for $h_{ij} \in \mathcal{F}$
  - $\therefore \quad Pr[h'_i(\mathbf{x}) = h'_i(\mathbf{y})] = \prod_{j=1}^{r} Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \geq p_1^r$
- $s(\mathbf{x}, \mathbf{y}) \leq s_2 \implies Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \leq p_2$ for $h_{ij} \in \mathcal{F}$
  - $\therefore \quad Pr[h'_i(\mathbf{x}) = h'_i(\mathbf{y})] = \prod_{j=1}^{r} Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \leq p_2^r$

Constructing new LSH families from old

Applying the OR construction to (s_1, s_2, p_1, p_2)-LSH family $\mathcal{F}$:

- Each h'' in new family $\mathcal{F}''$ consists of b functions $h_{i1}, h_{i2}, \ldots, h_{ib}$ from $\mathcal{F}$
- $h''_i = \{h_{i1}, h_{i2}, \ldots, h_{ib}\} \in \mathcal{F}''$ works as follows $\triangleright\ |\mathcal{F}'| = \binom{n}{b}$

$$h''_i(\mathbf{x}) = h''_i(\mathbf{y}) \iff h_{i1}(\mathbf{x}) = h_{i1}(\mathbf{y}) \vee h_{i2}(x) = h_{i2}(\mathbf{y}) \vee \ldots \vee h_{ib}(\mathbf{x}) = h_{ib}(\mathbf{y})$$

  - $h'' \in \mathcal{F}''$ only declares a candidate pair if any of b functions from $\mathcal{F}$ do

	1	2	3	4	5	6	7	8	9
$\mathbf{x}$	1	0	1	1	0	1	1	0	0

$\mathbf{h''_1} = \{\mathbf{h_2}, \mathbf{h_5}, \mathbf{h_7}\}$

$\mathbf{h''_2} = \{\mathbf{h_1}, \mathbf{h_4}, \mathbf{h_8}\}$ $\quad \mathbf{h''_1}(\mathbf{x}) = \mathbf{h''_1}(\mathbf{y}) \quad \mathbf{h''_2}(\mathbf{x}) \neq \mathbf{h''_2}(\mathbf{y}) \quad \mathbf{h''_3}(\mathbf{x}) = \mathbf{h''_3}(\mathbf{y})$

$\mathbf{h''_3} = \{\mathbf{h_6}, \mathbf{h_9}\}$

	1	2	3	4	5	6	7	8	9
$\mathbf{y}$	0	1	1	0	0	1	0	1	0

Constructing new LSH families from old

Applying the OR construction to (s_1, s_2, p_1, p_2)-LSH family $\mathcal{F}$:

- Each h'' in new family $\mathcal{F}''$ consists of b functions $h_{i1}, h_{i2}, \ldots, h_{ib}$ from $\mathcal{F}$
- $h''_i = \{h_{i1}, h_{i2}, \ldots, h_{ib}\} \in \mathcal{F}''$ works as follows $\triangleright$ $|\mathcal{F}'| = \binom{n}{b}$

$$h''_i(\mathbf{x}) = h''_i(\mathbf{y}) \iff h_{i1}(\mathbf{x}) = h_{i1}(\mathbf{y}) \vee h_{i2}(\mathbf{x}) = h_{i2}(\mathbf{y}) \vee \ldots \vee h_{ib}(\mathbf{x}) = h_{ib}(\mathbf{y})$$

  - $h'' \in \mathcal{F}''$ only declares a candidate pair if any of b functions from $\mathcal{F}$ do

$\mathcal{F}''$ is a $(s_1, s_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-family of LSH functions

- Choose $h'_i \in \mathcal{F}'' \iff$ Choose b functions $\{h_{i1}, h_{i2}, \ldots, h_{ib}\}$ in $\mathcal{F}$
- $s(\mathbf{x},\mathbf{y}) \geq s_1 \implies Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \geq p_1$ for $h_{ij} \in \mathcal{F}$
  - $\therefore \quad Pr[h''_i(\mathbf{x}) = h''_i(\mathbf{y})] = 1 - \prod_{j=1}^{b} Pr[h_{ij}(\mathbf{x}) \neq h_{ij}(\mathbf{y})] \geq 1-(1-p_1)^b$
- $s(\mathbf{x},\mathbf{y}) \leq s_2 \implies Pr[h_{ij}(\mathbf{x}) = h_{ij}(\mathbf{y})] \leq p_2$ for $h_{ij} \in \mathcal{F}$
  - $\therefore \quad Pr[h''_i(\mathbf{x}) = h''_i(\mathbf{y})] = 1 - \prod_{j=1}^{b} Pr[h_{ij}(\mathbf{x}) \neq h_{ij}(\mathbf{y})] \leq 1-(1-p_2)^b$

Constructing new LSH families from old

Choosing b and r

- Let $\mathcal{F}$ be a (s_1, s_2, p_1, p_2)-LSH family $\triangleright$ $p_1 > p_2$
- Using r-wise AND construction, from $\mathcal{F}$ we get a LSH family, $\mathcal{F}'$
- $\mathcal{F}'$ is (s_1, s_2, p_1^r, p_2^r)-family of LSH functions both probabilities smaller

  $\triangleright$ Our goal was to make only p_2 smaller
- Choose r so p_2^r becomes very small (~ 0) but p_1^r is not very small
- Using b-wise OR construction, from $\mathcal{F}$, we get a LSH family, $\mathcal{F}''$
- $\mathcal{F}''$ is $(s_1, s_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-family, both probabilities larger

  $\triangleright$ Our goal was to make only p_1 larger
- Choose b so $1-(1-p_1)^b$ becomes very large (~ 1) but $1-(1-p_2)^b$ doesn't grow too much

LSH Scheme: AND-OR Compositon

$\mathcal{F} : (s_1, s_2, p_1, p_2)$-family $\xrightarrow{r\text{-wise AND}} \mathcal{F}' : (s_1, s_2, p_1^r, p_2^r)$-family

$\mathcal{F}' : (s_1, s_2, p_1^r, p_2^r)$-family $\xrightarrow{b\text{-wise OR}} \mathcal{F}" : (s_1, s_2, 1-(1-p_1^r)^b, 1-(1-p_2^r)^b)$-family

- Choose b collections of r independent random functions from $\mathcal{F}$
- b meta functions $f_1, \ldots, f_b$ from $\mathcal{F}'$
  - each an AND of r functions in $\mathcal{F}$

x and **y** is a **candidate pair** if

$$[f_1(\mathbf{x}) = f_1(\mathbf{y})] \text{ OR } [f_2(\mathbf{x}) = f_2(\mathbf{y})] \text{ OR } \ldots \text{ OR } [f_b(\mathbf{x}) = f_b(\mathbf{y})]$$

- Visualize this as bands of $b \times r$ signature matrix
- AND-OR Construction: r-way AND followed by b-way OR construction
- Denoted by (r, b) AND-OR construction

LSH Scheme: OR-AND Composition

$\mathcal{F} : (s_1, s_2, p_1, p_2)$-family $\xrightarrow{b\text{-wise OR}}$ $\mathcal{F}' : (s_1, s_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-family

$\mathcal{F}' : (s_1, s_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-family $\xrightarrow{r\text{-wise AND}}$
$\mathcal{F}'' : (s_1, s_2, (1-(1-p_1)^b)^r, (1-(1-p_2)^b)^r)$-family

- Choose r collections of b independent random functions from $\mathcal{F}$
- r meta functions $f_1, \ldots, f_r$ from $\mathcal{F}'$
  - each an OR of b functions from $\mathcal{F}$

x and **y** is a **candidate pair** if

$$[f_1(\mathbf{x}) = f_1(\mathbf{y})] \text{ AND } [f_2(\mathbf{x}) = f_2(\mathbf{y})] \text{ AND } \ldots \text{ AND } [f_b(\mathbf{x}) = f_b(\mathbf{y})]$$

- Visualize this as bands of $b \times r$ signature matrix
- OR-AND Construction: b-way OR followed by r-way AND construction
- denoted by (b, r) OR-AND construction

LSH Scheme: AND-OR Compositon

Effect of construction and values of b and r of steepness of the S-curve

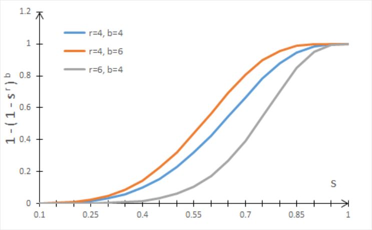
	$(r,b)=(4,4)$	$(r,b)=(4,6)$	$(r,b)=(6,4)$
p	$1-(1-p^r)^b$	$1-(1-p^r)^b$	$1-(1-p^r)^b$
0.1	0.0004	0.0006	0
0.2	0.00638	0.00956	0.00026
0.3	0.03201	0.04763	0.00291
0.4	0.09853	0.1441	0.01628
0.5	0.22752	0.32107	0.06105
0.6	0.42605	0.56518	0.17396
0.7	0.66655	0.80745	0.39387
0.8	0.8785	0.95765	0.70359
0.9	0.98601	0.99835	0.9518

A $(s_1, s_2, .2, .8)$ family is converted by

- $(r,b)=(4,4)$ AND-OR construction to a $(s_1, s_2, 0.00638, 0.8785)$
- $(r,b)=(4,6)$ AND-OR construction to a $(s_1, s_2, 0.00956, 0.95765)$
- $(r,b)=(6,4)$ AND-OR construction to a $(s_1, s_2, 0.00026, 0.70359)$

LSH Scheme and the S cruve

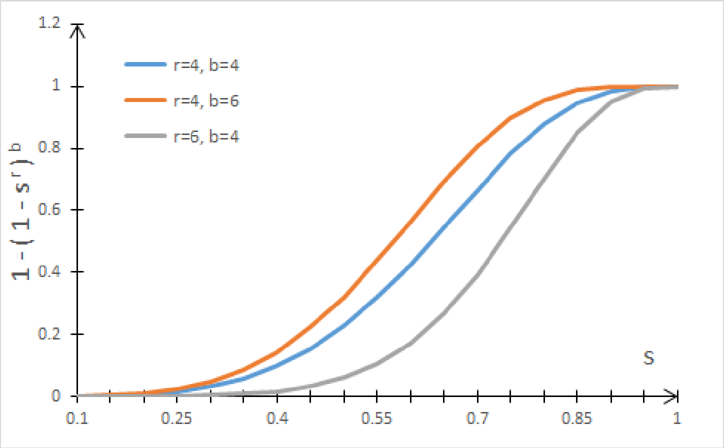
- Plot of $(1-(1-p^r)^b)$ is an S-shaped curve for every b and r



- There is a small range where the probability sharply decrease (for small values of p) or increase (for larger values of p)
- This is exactly what we want (recall our goal of the step function)
- Choose b and r (for AND-OR construction) so p_2 is in the "right interval", and the p_1 is on the left portion of the curve
- Any S-curve has a fixed-point, i.e. $\exists$ p satisfying $p = 1-(1-p^r)^b$, above this p prob. of candid. $(1-(1-p^r)^b)$ increases and vice-versa

LSH Scheme and the S cruve

Effect of construction and values of b and r of steepness of the S-curve



LSH Scheme: OR-AND Composition

Effect of construction and values of b and r of steepness of the S-curve

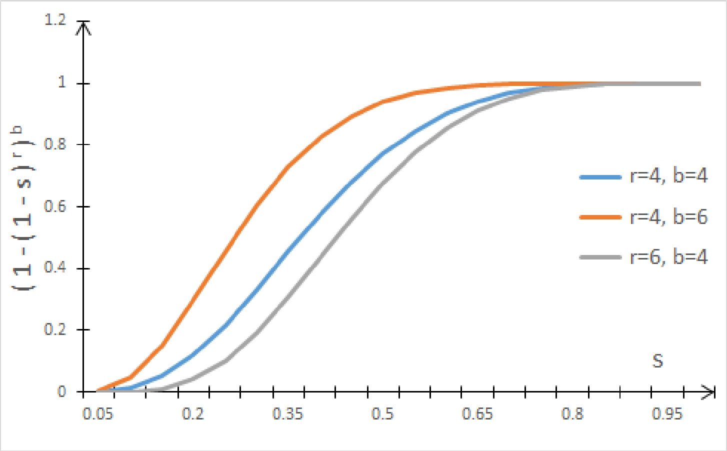
	$(b,r)=(4,4)$	$(b,r)=(4,6)$	$(b,r)=(6,4)$
p	$(1-(1-p)^b)^r$	$(1-(1-p)^b)^r$	$(1-(1-p)^b)^r$
0.1	0.01399	0.0482	0.00165
0.2	0.1215	0.29641	0.04235
0.3	0.33345	0.60613	0.19255
0.4	0.57395	0.82604	0.43482
0.5	0.77248	0.93895	0.67893
0.6	0.90147	0.98372	0.8559
0.7	0.96799	0.99709	0.95237
0.8	0.99362	0.99974	0.99044
0.9	0.9996	1	0.9994

A $(s_1, s_2, .2, .8)$ family is converted by

- $(b,r)=(4,4)$ OR-AND construction to $(s_1, s_2, 0.1215, 0.99362)$
- $(b,r)=(4,6)$ OR-AND construction to $(s_1, s_2, 0.29641, 0.99974)$
- $(b,r)=(6,4)$ OR-AND construction to $(s_1, s_2, 0.04235, 0.99044)$

LSH Scheme and the S cruve

Effect of construction and values of b and r of steepness of the S-curve



LSH Scheme

Create a cascade of multiple AND-OR or OR-AND constructions with varying values of r and b depending on the requirements

LSH for other distances

LSH for other distances

We gave a LSH family for Hamming distance. Next we consider Jaccard, Cosine, Euclidean distances

- We only need a basic (d_1, d_2, p_1, p_2)-LSH family $\mathcal{F}$
- Here d_1 and d_2 are w.r.t other (than Hamming) distance measures
- We want for a random $h \in \mathcal{F}$
  1. if $sim(\mathbf{x}, \mathbf{y})$ is high, then with high probability $h(\mathbf{x}) = h(\mathbf{y})$
  2. if $sim(\mathbf{x}, \mathbf{y})$ is low, then with high probability $h(\mathbf{x}) \neq h(\mathbf{y})$
- With amplification we can adjust the parameters
- Clearly such hash functions will depend on the particular similarity
- We know that not all similarities have such suitable LSH families

Non-LSH-able distances

Known that no LSH scheme exists for certain distance measures

1. Sørensen-Dice: A similarity measure between sets

For two sets X and Y $\quad sim_{sd}(X,Y) = \dfrac{2|X \cap Y|}{|X| + |Y|}$

$X = \{a\}, Y = \{b\},$ and $Z = \{a,b\}$

$sim_{sd}(X,Y) = 0 \qquad sim_{sd}(X,Z) = 2/3 \qquad sim_{sd}(Y,Z) = 2/3$

2. Overlap Similarity: A similarity measure between sets

For two sets X and Y $\quad sim_{ov}(X,Y) = \dfrac{|X \cap Y|}{\min\{|X|, |Y|\}}$

$X = \{a\}, Y = \{b\},$ and $Z = \{a,b\}$

$sim_{ov}(X,Y) = 0 \qquad sim_{ov}(X,Z) = 1 \qquad sim_{ov}(Y,Z) = 1$

In both cases distances are defined as $1 - sim_*(\cdot,\cdot)$

Non-(yet)-LSH-able distances

Open question to design -LSH-able scheme for certain distance measures

1. Anderberg: A similarity measure between sets

$$\text{For } X \text{ and } Y \qquad sim_{an}(X,Y) = \frac{|X \cap Y|}{|X \cap Y| + 2|X \oplus Y|}$$

   Compute this similarity for pairs of
   $X = \{a\}, Y = \{b\},$ and $Z = \{a, b\}$

2. Rogers-Tanimoto A similarity measure between sets

$$\text{For } X \text{ and } Y \qquad sim_{rt}(X,Y) = \frac{|X \cap Y| + |X \cup Y|}{|X \cap Y| + \overline{X \cup Y} + 2|X \oplus Y|}$$

   Compute this similarity for pairs of
   $X = \{a\}, Y = \{b\},$ and $Z = \{a, b\}$

MinHash: LSH for Jaccard distance

LSH for Jaccard distance (Minhashing)

- LSH family for Jaccard distance called Minhashes or Min-wise hashing
- Suppose all sets are subsets of a universal set U
  - If sets are documents, then U could be the English lexicon
- $\mathcal{F}$: set of all permutations of elements in U
- We will show that $\mathcal{F}$ is a family of LSH function
- For a permutation π of elements in U the hash function h_π
  - h_π is of the form $h_\pi : \mathcal{P}(U) \mapsto U$ $\triangleright$ $\mathcal{P}(U)$: all possible subsets
  - Takes as input a subset of U and returns an element of U
  - h_π maps a set $S \subseteq U$ as follows:
  - $h_\pi(S)$ is the first element of S in the order of π
- $|\mathcal{F}| = |U|!$

Minhashing

- Let $U = \{w_0, w_1, w_2, w_3, w_4\}$
- Given four sets S_1, S_2, S_3, S_4
- Let the permutation $\pi = (w_1, w_4, w_0, w_3, w_2)$

elem.id	S_1	S_2	S_3	S_4
w_0	1	0	0	1
w_1	0	0	1	0
w_2	0	1	0	1
w_3	1	0	1	1
w_4	0	0	1	0

Given Sets

elem.id	S_1	S_2	S_3	S_4
w_1	0	0	1	0
w_4	0	0	1	0
w_0	1	0	0	1
w_3	1	0	1	1
w_2	0	1	0	1

Sets reordered according to π

$h_\pi(S_1) = w_0$ $h_\pi(S_2) = w_2$ $h_\pi(S_3) = w_1$ $h_\pi(S_4) = w_0$

- $h_\pi(S)$ is the index of row (elem.id) with first 1 in the order π
- Called minhashing because of this first index (minimum index)

Minhashing

Let $|U| = n$, all sets (vectors) are n-dimensional $\implies$ $n!$ functions in $\mathcal{F}$

$\mathcal{F}$ is a $(d_1, d_2, (1 - d_1), (1 - d_2))$-family of LSH functions

Choose h_π at random from $\mathcal{F}$ $\iff$ Choose a random permutation π of U

Let S and T be two arbitrary subsets of U

- Suppose $d(S, T) \leq d_1$
- Picture S and T as two columns with rows ordered by π
- $h_\pi(S) = h_\pi(T)$ is event that first element in order of π is same in S and T
  - i.e. we get a [1 1] row before any [1 0] and [0 1] row (ignore [0 0] rows)
- Since π is a random permutation the probability of this happening is

$$\frac{\text{No. of } [1\ \ 1] \text{ rows}}{\text{No. of } [1\ \ 1], [1\ \ 0], [0\ \ 1] \text{ rows}} = \frac{|S \cap T|}{|S \cup T|}$$

- Thus $Pr[h_\pi(S) = h_\pi(T)] \geq 1 - d_1$
- The other bound is obtained analogously

Approximate Minhashing

Approximate minhash using universal hash function

1. To pick a random permutation is not easy
2. Finding minhashes of sets is expensive, need sorting by π and find the first 1
3. For large U, all columns would have many 0's (sparse matrix)
4. Approximation: Use universal hash functions instead
5. permutation is of the form $\pi : [n] \mapsto [n]$ (bijection no collisions)
6. Take a universal hash function $h : [n] \mapsto [n]$ or even better $[n] \mapsto [2n]$
7. Will have few collisions; order of $w_i, w_j \in U$ by $h(w_i) <^? h(w_j)$
8. By the randomness of h we get that either order is equally likely
9. The (approximate) minhash value is then computed as follows:

$$minhash(S) = \arg\min_{w \in S} h(w)$$

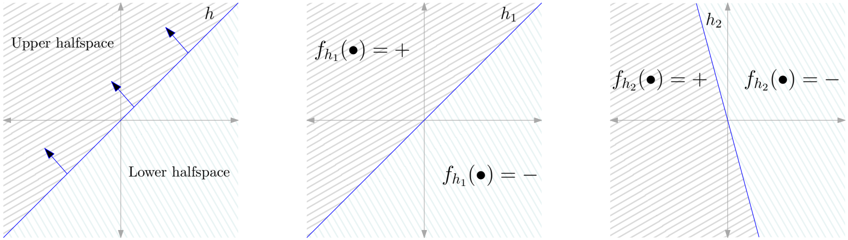
10. With a universal hash function, only need to compute the minimum of elements that are in S (ignore 0 rows in column of S)

SimHash: LSH for Cosine distance

LSH for Cosine distance

A LSH family $\mathcal{F}$ for cosine distance for points in $\mathbb{R}^m$ ▷ simHash

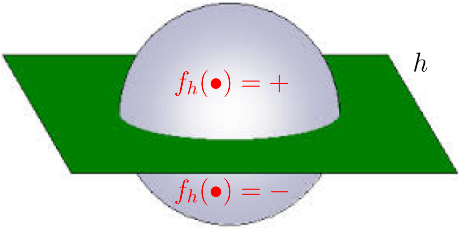
- Choose a hyperplane h in $\mathbb{R}^m$
  - a line in $2d$, a plane in $3d$, a $d-1$ dimensional subspace of $\mathbb{R}^d$
  - h divides the space in two half-spaces (upper/+ve and lower/−ve)
- $\mathcal{F}$ contains functions f_h corresponding to hyperplanes
- f_h maps vectors in the upper half-space to bucket $+$ and vectors in the lower half-space to bucket $-$



u and **v** is a candidate pair if $f_h(\mathbf{u}) = f_h(\mathbf{v})$ else they are not

LSH for Cosine distance

The same concept applies to higher dimensions

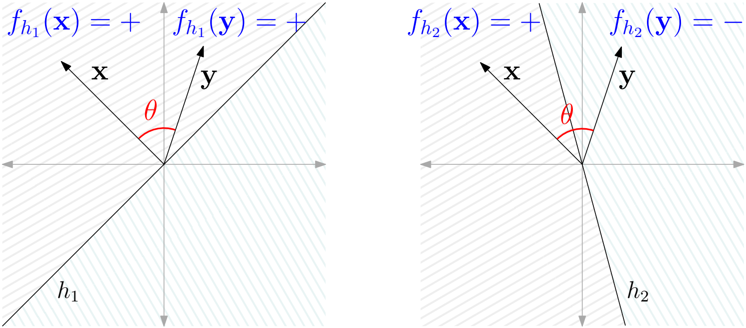


- A hyperplane (a 2d plane) splits the 3d space into two half spaces
- We show only a sphere, as WLOG we consider only unit vectors (surface of unit ball in $\mathbb{R}^d$), as our concern is angles between vectors
- Vectors in the upper half-space are mapped to $+$ by the function corresponding to the given hyperplane h
- Vectors in the lower half-space are mapped to $-$

LSH for Cosine distance

Let **x** and **y** be two vectors with angle θ between them

Probability that a random hyperplane h goes between them is exactly $\theta/180^\circ$



- f_{h_1} and f_{h_2} in $\mathcal{F}$ corresponding to hyperplanes h_1 and h_2
- $f_{h_1}(\mathbf{x}) = f_{h_1}(\mathbf{y}) \implies$ **x** and **y** is a candidate pair under f_{h_1}
- Under f_{h_2}, **x** and **y** is not a candidate pair

LSH for Cosine distance

$\mathcal{F}$: corresponding to $(m-1)$-dim hyperplanes (passing through $\mathbf{0}$ in $\mathbb{R}^m$)

$\mathcal{F}$ is a $(d_1, d_2, {}^{(180-d_1)}/_{180}, {}^{(180-d_2)}/_{180})$-family of LSH functions

Choose random $f_h \in \mathcal{F} \iff$ Choose random hyperplane h

- $d_{cos}(\mathbf{x},\mathbf{y}) \leq d_1 \implies \geq {}^{(1-d_1)}/_{180}$ chance h does not separate $\mathbf{x}$ and $\mathbf{y}$
- $d_{cos}(\mathbf{x},\mathbf{y}) \geq d_2 \implies \leq {}^{(1-d_2)}/_{180}$ chance h does not separate $\mathbf{x}$ and $\mathbf{y}$
- Combining the above two statements we get the theorem $\square$

- We can amplify this as we wish
- $\mathcal{F}$ has infinitely many functions, unlike
- LSH for Hamming similarity (only n functions in the base family) and
- Jaccard similarity ("only" $n!$ functions in the base family)

LSH for Cosine distance: Computation

- Not easy to find the half-space where a vector $\mathbf{x}$ lies
- Pick a unit vector $\mathbf{v}$ and consider hyperplane to which $\mathbf{v}$ is normal
- The unit vector $\mathbf{v}$ "uniquely" represents the hyperplane
  - Infinitely many normal vectors to a hyperplane – all scalings of $\mathbf{v}$
  - But only two unit vectors ($\mathbf{v}$ and $-1\mathbf{v}$) pegged at origin
- The hyperplane with $\mathbf{v}$ as its normal is the family of vectors (the $n-1$ dimensional subspace) whose dot-product with $\mathbf{v}$ is 0
- Upper half-space: vectors whose dot-product with $\mathbf{v}$ is positive (> 0)
- Lower lower half-space: vectors whose dot-product with $\mathbf{v}$ is negative

$f_h(\mathbf{x})$ is computed as follows. Let $\mathbf{v}$ be a normal to h, then

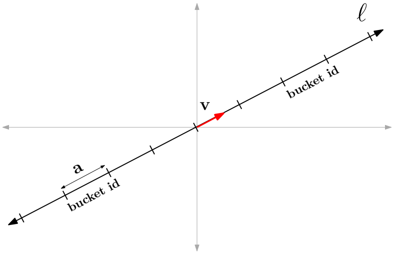
$$f_h(\mathbf{x}) = sign(\mathbf{v} \cdot \mathbf{x}), \quad \text{where} \quad sign(a) = \begin{cases} + & \text{if } a \geq 0 \\ - & \text{otherwise} \end{cases}$$

Random Projection: LSH for Euclidean distance

LSH for Euclidean distance

Overall idea of LSH for Euclidean distance:
Projections of "close-by" vectors in $\mathbb{R}^m$ onto a vector should be "close"

- ℓ : a line in $\mathbb{R}^m$ passing through $\mathbf{0}$
- $\mathbf{v}$: unit vector in direction of ℓ
- Divide ℓ into segments of length a (a fixed constant)
- Segments are buckets for the hash function corresponding to ℓ



Function $h_{\mathbf{v}} = h_\ell$ (corresponding to ℓ or $\mathbf{v}$) maps $\mathbf{x}$ to segment where projection of $\mathbf{x}$ on ℓ lies

$$h_{\mathbf{v}}(\mathbf{x}) = \left\lfloor \frac{\langle \mathbf{x}, \mathbf{v} \rangle}{a} \right\rfloor$$

$h_{\mathbf{v}}$ projects $\mathbf{x}$ onto $\mathbf{v}$ and discretize the projection into a multiple of a

LSH for Euclidean distance

LSH family $\mathcal{F}$ contains functions corresponding to unit vectors in $\mathbb{R}^m$

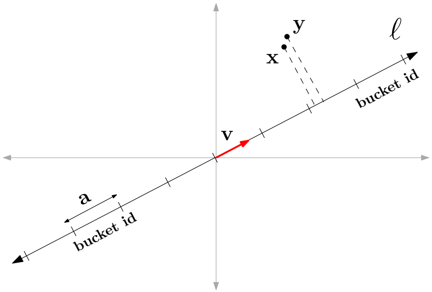
$\mathcal{F}$ has infinitely many functions

Locality sensitivity of $\mathcal{F}$

- Intuitively, close by vectors are likely to fall into the same bucket
- Far vectors are less likely to fall into the same bucket (tricky part)

LSH for Euclidean distance

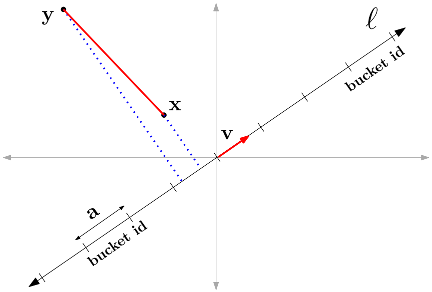
- $Pr[h_{\mathbf{v}}(\mathbf{x}) = h_{\mathbf{v}}(\mathbf{y})] \;\propto\; d(\mathbf{x}, \mathbf{y})$
- It also depends on angle between ℓ and line-segment joining **x** and **y**



- If $d(\mathbf{x}, \mathbf{y})$ is small compared to a, **x** and **y** will likely fall in same bucket
- Though **x** and **y** may fall close to boundary of two adjacent buckets

LSH for Euclidean distance

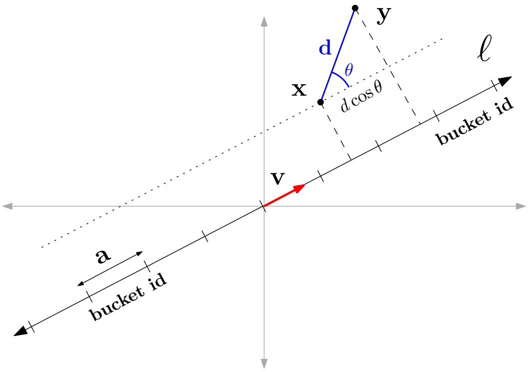
- $Pr[h_{\mathbf{v}}(\mathbf{x}) = h_{\mathbf{v}}(\mathbf{y})] \propto d(\mathbf{x}, \mathbf{y})$
- It also depends on angle between ℓ and line-segment joining **x** and **y**



- If $d(\mathbf{x}, \mathbf{y})$ is large compared to a, **x** and **y** unlikely to fall in one bucket
- If d is large but line segment joining **x** and **y** is almost perpendicular to ℓ, still they are likely to fall in same bucket

LSH for Euclidean distance

- Dependence of event $h_{\mathbf{v}}(\mathbf{x}) = h_{\mathbf{v}}(\mathbf{y})$ and angle between $\overline{xy}$ and ℓ
- If $h_{\mathbf{v}}(\mathbf{x}) = h_{\mathbf{v}}(\mathbf{y})$, then $d\cos\theta_{xy} < a$
- This is only necessary condition, not sufficient
- i.e. even if $d\cos\theta \ll a$, $\mathbf{x}$ and $\mathbf{y}$ may still go to different buckets



LSH for Euclidean distance

$\mathcal{F}$ is a $(a/2, 2a, 1/2, 1/3)$-family of LSH functions

Pick a random h in $\mathcal{F}$ $\iff$ Pick a random line ℓ in $\mathbb{R}^n$

The angle θ between ℓ and the line through $\mathbf{x}$ and $\mathbf{y}$ is random

- Suppose $d = d(\mathbf{x}, \mathbf{y}) < a/2$
- Since $d < a/2$, $\mathbf{x}$ and $\mathbf{y}$ either fall in the same or consecutive buckets
- Even if $\mathbf{x}$ falls on the bucket border, there is $\geq 50\%$ chance that $\mathbf{y}$ falls in the same bucket. Thus $Pr[h_\ell(\mathbf{x}) = h_\ell(\mathbf{y})] \geq 1/2$
- Suppose $d = d(\mathbf{x}, \mathbf{y}) < 2a$
- d' : distance between projections of $\mathbf{x}$ and $\mathbf{y}$ on ℓ ($d' = d\cos\theta$)

$$h_\ell(\mathbf{x}) = h_\ell(\mathbf{y}) \Rightarrow d' < a \Rightarrow d\cos\theta < a \Rightarrow 2a\cos\theta < a \Rightarrow \cos\theta < \frac{1}{2} \Rightarrow \theta \in [60^\circ, 90^\circ]$$

- $Pr\big(\theta \in [60^\circ, 90^\circ]\big)$ is $1/3$ $\quad\triangleright$ θ is random
- Thus $Pr[h_\ell(\mathbf{x}) = h_\ell(\mathbf{y})] \geq 1/3$

LSH for Euclidean distance

- Note difference between $\mathcal{F}$ for ℓ_2 distance and those other distances
- For others we got
  for any d_1 and d_2 and the probabilities $(1 - d_1)$ and $(1 - d_2)$
- Here for any distance $d_1 < d_2$, all we get is $p_1 > p_2$
- This will require more functions for amplification to desired values
- We have infinitely many functions though

LSH Computational Issues

Memory Requirement of LSH and implementation trick

Given that the resulting hash tables have at most n non-zero entries, one can reduce the amount of memory used per hash table to $O(n)$ using universal hash functions

Data Dependent LSH

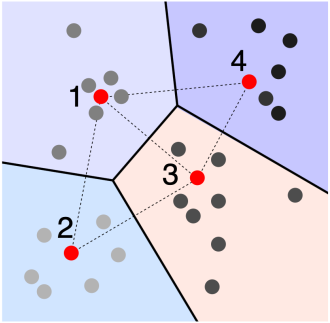
All LSH we discussed are sensitive to specific distance measure

They are all data oblivious (they do not look at the data)

Clustering LSH ▷ a data dependent LSH scheme

Cluster datasets into k clusters (using some method and proximity)

Bucket ID of each point is it's cluster id



https://randorithms.com/