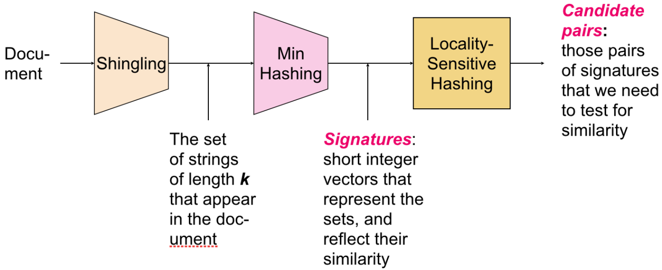
Finding Similar Items II

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 21, 2022

Summary of Current Status



From `mmds.org`

- *Shingling:* turning text files into sets ☞ Done!
- *Minhashing:* computing similarity for large sets ☞ Done!
- *Locality Sensitive Hashing:* avoids $O(N^2)$ comparisons by determining candidate pairs ☞ today!

Current Status: Issues Still Remaining

- Minhashing enabled to compute similarity between two sets very fast
- Shingling enabled to turn documents into sets such that minhashing could be applied
- But if number of items N is too large, $O(N^2)$ similarity computations are infeasible, even using minhashing
- *Idea:* Browse through items and determine *candidate pairs*:
  - Number of candidate pairs is much smaller than $O(N^2)$
  - One performs minhashing only for candidate pairs
  - Candidate pairs can be determined with a very fast procedure
- *Solution: Locality Sensitive Hashing (a.k.a. Near Neighbor Search)*

Current Status: Issues Still Remaining

- Minhashing enabled to compute similarity between two sets very fast
- Shingling enabled to turn documents into sets such that minhashing could be applied
- But if number of items N is too large, $O(N^2)$ similarity computations are infeasible, even using minhashing
- *Idea:* Browse through items and determine *candidate pairs*:
  - Number of candidate pairs is much smaller than $O(N^2)$
  - One performs minhashing only for candidate pairs
  - Candidate pairs can be determined with a very fast procedure
- *Solution: Locality Sensitive Hashing (a.k.a. Near Neighbor Search)*

Current Status: Issues Still Remaining

- Minhashing enabled to compute similarity between two sets very fast
- Shingling enabled to turn documents into sets such that minhashing could be applied
- But if number of items N is too large, $O(N^2)$ similarity computations are infeasible, even using minhashing
- *Idea:* Browse through items and determine *candidate pairs*:
  - Number of candidate pairs is much smaller than $O(N^2)$
  - One performs minhashing only for candidate pairs
  - Candidate pairs can be determined with a very fast procedure
- *Solution: Locality Sensitive Hashing (a.k.a. Near Neighbor Search)*

Learning Goals Today

- Understand the technique of *Locality Sensitive Hashing (LSH)*
- Understand the theory supporting it

Locality Sensitive Hashing

Locality Sensitive Hashing: Idea

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Here: $m = 5, n = 2$
- Originally: each set is from $\{0,1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m-1\}^n$
- Much reduced representation, because $n << m$

$\Rightarrow n \cdot \log_2 m < m$

$\Rightarrow m^n < 2$

Locality Sensitive Hashing: Idea

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Here: $m = 5, n = 2$
- Originally: each set is from $\{0, 1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m - 1\}^n$
- Much reduced representation, because $n << m$

Idea:

- Hash items (columns in *SIG*) several times (b times)
- *Candidate pair:* pair of columns hashed to the same bucket, by any of the hash functions
- *Runtime:* Hashing all columns is $O(N)$, examining buckets requires little time

Locality Sensitive Hashing: Idea

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Here: $m = 5, n = 2$
- Originally: each set is from $\{0, 1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m-1\}^n$
- Much reduced representation, because $n << m$

Idea:

- Hash items (columns in *SIG*) several times (b times)
- *Candidate pair:* pair of columns hashed to the same bucket, by any of the hash functions
- *Runtime:* Hashing all columns is $O(N)$, examining buckets requires little time

Locality Sensitive Hashing: Idea

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Here: $m = 5, n = 2$
- Originally: each set is from $\{0, 1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m-1\}^n$
- Much reduced representation, because $n << m$

Idea:

- Hash items (columns in *SIG*) several times (b times)
- *Candidate pair:* pair of columns hashed to the same bucket, by any of the hash functions
- *Runtime:* Hashing all columns is $O(N)$, examining buckets requires little time

Motivation:

- *False Positive:* dissimilar pair hashing to the same bucket
- *False Negative:* similar pair never hashing to the same bucket
- *Motivation:* limit both false positives and negatives

Locality Sensitive Hashing: Idea

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Here: $m = 5, n = 2$
- Originally: each set is from $\{0, 1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m-1\}^n$
- Much reduced representation, because $n << m$

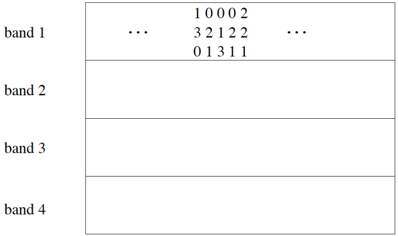
Idea:

- Hash items (columns in *SIG*) several times (b times)
- *Candidate pair:* pair of columns hashed to the same bucket, by any of the hash functions
- *Runtime:* Hashing all columns is $O(N)$, examining buckets requires little time

Motivation:

- *False Positive:* dissimilar pair hashing to the same bucket
- *False Negative:* similar pair never hashing to the same bucket
- *Motivation:* limit both false positives and negatives

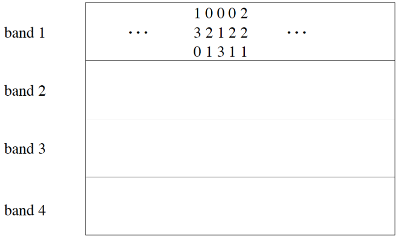
Locality Sensitive Hashing: Banding Technique



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- Divide rows of signature matrix into b bands of r rows each
- For each band, a hash function hashes r integers to buckets
- Number of buckets is large to avoid collisions
- *Candidate pair:* a pair of columns hashed to the same bucket, in any band

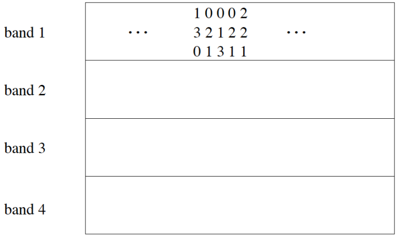
Locality Sensitive Hashing: Banding Technique



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- Divide rows of signature matrix into b bands of r rows each
- For each band, a hash function hashes r integers to buckets
- Number of buckets is large to avoid collisions
- *Candidate pair:* a pair of columns hashed to the same bucket, in any band

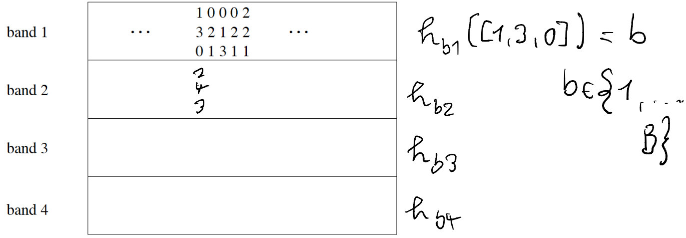
Locality Sensitive Hashing: Banding Technique



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- Divide rows of signature matrix into b bands of r rows each
- For each band, a hash function hashes r integers to buckets
- Number of buckets is large to avoid collisions
- *Candidate pair:* a pair of columns hashed to the same bucket, in any band

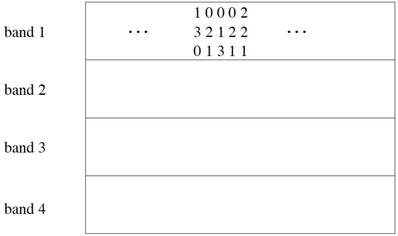
BANDING TECHNIQUE: EXAMPLE



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- The columns showing $[0, 2, 1]$ in band 1 are declared a candidate pair
- Other pairs of columns shown are not declared candidate pairs as per the hash function of the first band
  - apart from collisions occurring ☞ which was designed to happen very rarely
- Pairs of columns may be hashed to the same bucket in another band, so may be declared candidate pairs

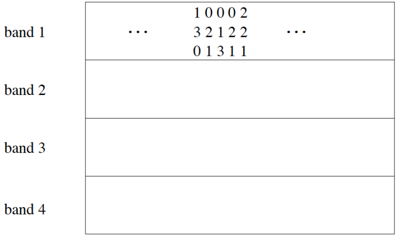
Banding Technique: Example



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- The columns showing $[0, 2, 1]$ in band 1 are declared a candidate pair
- Other pairs of columns shown are not declared candidate pairs as per the hash function of the first band
  - apart from collisions occurring ☞ which was designed to happen very rarely
- Pairs of columns may be hashed to the same bucket in another band, so may be declared candidate pairs

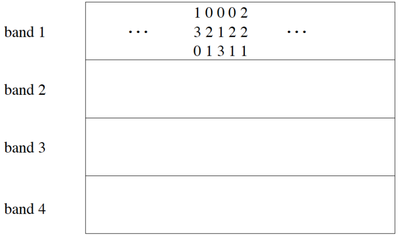
BANDING TECHNIQUE: EXAMPLE



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- The columns showing $[0, 2, 1]$ in band 1 are declared a candidate pair
- Other pairs of columns shown are not declared candidate pairs as per the hash function of the first band
    - apart from collisions occurring ☞ which was designed to happen very rarely
- Pairs of columns may be hashed to the same bucket in another band, so may be declared candidate pairs

BANDING TECHNIQUE: EXAMPLE



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- The columns showing $[0, 2, 1]$ in band 1 are declared a candidate pair
- Other pairs of columns shown are not declared candidate pairs as per the hash function of the first band
  - apart from collisions occurring ☞ which was designed to happen very rarely
- Pairs of columns may be hashed to the same bucket in another band, so may be declared candidate pairs

Banding Technique: Theorem

Let *SIG* be a signature matrix grouped into

- b bands of
- r rows each

and consider

- a pair of columns of Jaccard similarity s

Theorem [LSH Candidate Pair]:
The probability that the pair of columns becomes a candidate pair is

$$1 - (1 - s^r)^b \tag{1}$$

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem “Minhash and Jaccard Similarity” of last lecture, they agree in that row with probability s

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem "Minhash and Jaccard Similarity" of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

- agree in all rows of one band is s^r,
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem “Minhash and Jaccard Similarity” of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

- agree in all rows of one band is s^r,
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem "Minhash and Jaccard Similarity" of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

- agree in all rows of one band is s^r,
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem "Minhash and Jaccard Similarity" of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

- agree in all rows of one band is s^r,
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

Banding Technique: Proof of Theorem

Proof.
Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem “Minhash and Jaccard Similarity” of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

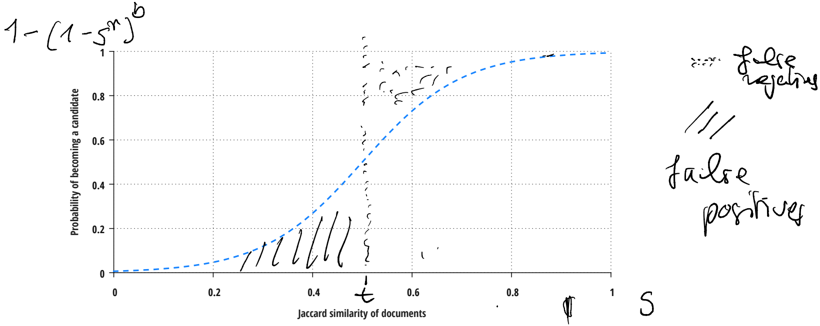
- agree in all rows of one band is s^r,
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

□

Banding Technique: The S-Curve

Definition: [S-Curve]
For given b and r, the *S-curve* is defined by the prescription

$$s \mapsto 1 - (1 - s^r)^b \qquad (2)$$



Exemplary S-curve

BANDING TECHNIQUE: THE S-CURVE

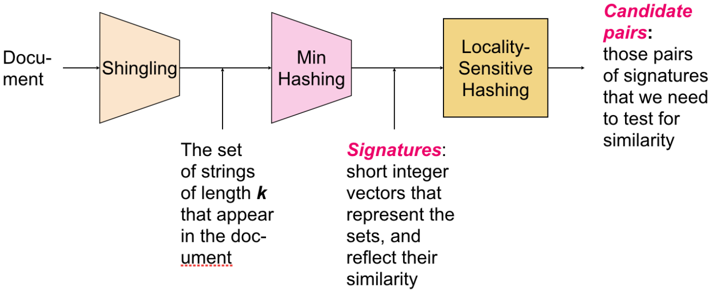
DEFINITION: [S-CURVE]
For given b and r, the *S-curve* is defined by the prescription

$$s \mapsto 1 - (1 - s^r)^b \tag{3}$$

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

Table: Values for S-curve with $b = 20$ and $r = 5$

Finding Similar Documents: Overall Workflow



From `mmds.org`

- Shingling: Done!
- Minhashing: Done!
- Locality-Sensitive Hashing: Done!

Locality Sensitive Hashing: Guidelines

- One needs to determine b, r
- One needs to determine threshold t:
  - $s \geq t$: candidate pair
  - $s < t$: no candidate pair
- bands times rows is number of rows of signature matrix ☞ $br = n$
- t corresponds with point of steepest rise on S-curve: approximately $(1/b)^{(1/r)}$

Locality Sensitive Hashing: Guidelines

- One needs to determine b, r
- One needs to determine threshold t:
  - $s \geq t$: candidate pair
  - $s < t$: no candidate pair
- bands times rows is number of rows of signature matrix ☞ $br = n$
- t corresponds with point of steepest rise on S-curve: approximately $(1/b)^{(1/r)}$

Locality Sensitive Hashing: Guidelines

- One needs to determine b, r
- One needs to determine threshold t:
  - $s \geq t$: candidate pair
  - $s < t$: no candidate pair
- bands times rows is number of rows of signature matrix ☞ $br = n$
- t corresponds with point of steepest rise on S-curve: approximately $(1/b)^{(1/r)}$

LOCALITY SENSITIVE HASHING: GUIDELINES

- One needs to determine b, r
- One needs to determine threshold t:
  - $s \geq t$: candidate pair
  - $s < t$: no candidate pair
- bands times rows is number of rows of signature matrix ☞ $br = n$
- t corresponds with point of steepest rise on S-curve: approximately $(1/b)^{(1/r)}$

FINDING SIMILAR DOCUMENTS: SUMMARY

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

FINDING SIMILAR DOCUMENTS: SUMMARY

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick threshold t, number of bands b and rows r
   - Avoiding false negatives: choose t, b, r such that $t \approx (1/b)^{1/r}$ is low
   - If avoiding false positives, or speed is important, choose t, b, r such that $t \approx (1/b)^{1/r}$ is large
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Distance Measures

Distance Measure: Definition

Definition: [Distance Measure]

Consider a set of objects. A *distance measure* is a function $d(x, y)$ that maps two objects x, y to a number such that

1. $d(x, y) \geq 0$ [d is *non-negative*]
2. $d(x, y) = 0$ implies $x = y$ [only if two objects are identical, the distance is zero; strictly positive otherwise]
3. $d(x, y) = d(y, x)$ [distance is *symmetric*]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [*triangle inequality*]

Distance Measure: Definition

Definition: [Distance Measure]

Consider a set of objects. A *distance measure* is a function $d(x, y)$ that maps two objects x, y to a number such that

1. $d(x, y) \geq 0$ [d is *non-negative*]
2. $d(x, y) = 0$ implies $x = y$ [only if two objects are identical, the distance is zero; strictly positive otherwise]
3. $d(x, y) = d(y, x)$ [distance is *symmetric*]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [*triangle inequality*]

Distance Measure: Definition

Definition: [Distance Measure]

Consider a set of objects. A *distance measure* is a function $d(x, y)$ that maps two objects x, y to a number such that

1. $d(x, y) \geq 0$ [d is *non-negative*]
2. $d(x, y) = 0$ implies $x = y$ [only if two objects are identical, the distance is zero; strictly positive otherwise]
3. $d(x, y) = d(y, x)$ [distance is *symmetric*]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [*triangle inequality*]

Distance Measure: Definition

Definition: [Distance Measure]

Consider a set of objects. A *distance measure* is a function $d(x, y)$ that maps two objects x, y to a number such that

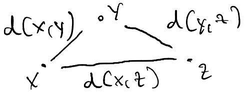
1. $d(x, y) \geq 0$ [d is *non-negative*]
2. $d(x, y) = 0$ implies $x = y$ [only if two objects are identical, the distance is zero; strictly positive otherwise]
3. $d(x, y) = d(y, x)$ [distance is *symmetric*]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [*triangle inequality*]

Distance Measure: Definition

Definition: [Distance Measure]

Consider a set of objects. A *distance measure* is a function $d(x, y)$ that maps two objects x, y to a number such that

1. $d(x, y) \geq 0$ [d is *non-negative*]
2. $d(x, y) = 0$ implies $x = y$ [only if two objects are identical, the distance is zero; strictly positive otherwise]
3. $d(x, y) = d(y, x)$ [distance is *symmetric*]
4. $d(x, z) \leq d(x, y) + d(y, z)$ [*triangle inequality*]



DISTANCE MEASURES: EXAMPLES

- In an *n-dimensional Euclidean space*, points are vectors of length n of real numbers
- The L_r-distance, defined to be

$$d([x_1, ..., x_n], [y_1, ..., y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r} \tag{4}$$

  is a distance measure
- A particular example is the Euclidean distance, defined as the L_2-distance
- *Cosine:* Let $||x||_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ be the L_2-*norm* of a point in Euclidean space. The *cosine similarity* for two points $[x_1, ..., x_n], [y_1, ..., y_n]$ is defined to be

$$\frac{\sum_{i=1}^n x_i y_i}{||x||_2 ||y||_2} \tag{5}$$

  - Measures the *angle* between two vectors x and y
  - Gives rise to distance measure between lines that pass through origin

Distance Measures: Examples

- In an *n-dimensional Euclidean space*, points are vectors of length n of real numbers
- The L_r-distance, defined to be

$$d([x_1, ..., x_n], [y_1, ..., y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r} \tag{4}$$

  is a distance measure
- A particular example is the Euclidean distance, defined as the L_2-distance
- *Cosine:* Let $||x||_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ be the L_2*-norm* of a point in Euclidean space. The *cosine similarity* for two points $[x_1, ..., x_n], [y_1, ..., y_n]$ is defined to be

$$\frac{\sum_{i=1}^n x_i y_i}{||x||_2 ||y||_2} \tag{5}$$

  - Measures the *angle* between two vectors x and y
  - Gives rise to distance measure between lines that pass through origin

DISTANCE MEASURES: EXAMPLES

- Let $\mathrm{SIM}(x, y)$ be the Jaccard similarity between two sets x, y. The quantity
$$1 - \mathrm{SIM}(x, y) \tag{6}$$
can be proven to be a distance measure.
- *Edit distance:* Objects are strings. The edit distance between two strings $x = x_1...x_m, y = y_1...y_n$ is the smallest number of insertions and deletions of single characters to be applied to turn x into y.
- *Hamming Distance:* For $[x_1, ..., x_n], [y_1, ..., y_n]$, the Hamming distance is the number of positions $i \in [1, ..., n]$ where $x_i \neq y_i$

Distance Measures: Examples

- Let $\mathrm{SIM}(x, y)$ be the Jaccard similarity between two sets x, y. The quantity
$$1 - \mathrm{SIM}(x, y) \tag{6}$$
can be proven to be a distance measure.
- *Edit distance:* Objects are strings. The edit distance between two strings $x = x_1 ... x_m, y = y_1 ... y_n$ is the smallest number of insertions and deletions of single characters to be applied to turn x into y.
- *Hamming Distance:* For $[x_1, ..., x_n], [y_1, ..., y_n]$, the Hamming distance is the number of positions $i \in [1, ..., n]$ where $x_i \neq y_i$

Distance Measures: Examples

- Let $\text{SIM}(x, y)$ be the Jaccard similarity between two sets x, y. The quantity

$$1 - \text{SIM}(x, y) \tag{6}$$

  can be proven to be a distance measure.

- *Edit distance:* Objects are strings. The edit distance between two strings $x = x_1 ... x_m, y = y_1 ... y_n$ is the smallest number of insertions and deletions of single characters to be applied to turn x into y.

- *Hamming Distance:* For $[x_1, ..., x_n], [y_1, ..., y_n]$, the Hamming distance is the number of positions $i \in [1, ..., n]$ where $x_i \neq y_i$

Edit / Hamming Distance: Example

Edit Distance D_E:
Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Edit / Hamming Distance: Example

Edit Distance D_E:

Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Edit / Hamming Distance: Example

Edit Distance D_E:
Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Edit / Hamming Distance: Example

Edit Distance D_E:

Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Edit / Hamming Distance: Example

Edit Distance D_E:
Consider $x = $ "*abcde*", $y = $ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Edit / Hamming Distance: Example

Edit Distance D_E:

Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

UNIVERSITÄT BIELEFELD

Edit / Hamming Distance: Example

Edit Distance D_E:

Consider $x =$ "*abcde*", $y =$ "*acfdeg*". Claim: $D_E(x, y) = 3$.

- For proving $D_E(x, y) \leq 3$, consider edit sequence
  1. Delete b
  2. Insert f after c
  3. Insert g after e
- For $D_E(x, y) \geq 3$, consider that x contains b, which y does not, which holds vice versa for f, g. This implies that 3 edit operations are necessary at least.

Hamming Distance D_H:

Consider $x = 10101, y = 11110$:

$$D_H(x, y) = 3$$

because disagreeing in 3 positions (of five overall).

Locality Sensitive Functions

Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \text{SIM}(x, y)$ is the Jaccard distance

Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \mathrm{SIM}(x, y)$ is the Jaccard distance

Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \text{SIM}(x, y)$ is the Jaccard distance

Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \text{SIM}(x, y)$ is the Jaccard distance

Definition: [Locality Sensitive (LS) Family of Functions]
A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-*sensitive* if for each $f \in \mathcal{F}$:

1. $d(x, y) \leq d_1$ implies that the probability that $f(x) = f(y)$ is at least p_1
2. $d(x, y) \geq d_2$ implies that the probability that $f(x) = f(y)$ is at most p_2

UNIVERSITÄT BIELEFELD

Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \text{SIM}(x, y)$ is the Jaccard distance

Definition: [Locality Sensitive (LS) Family of Functions]
A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-*sensitive* if for each $f \in \mathcal{F}$:

1. $d(x, y) \leq d_1$ implies that the probability that $f(x) = f(y)$ is at least p_1
2. $d(x, y) \geq d_2$ implies that the probability that $f(x) = f(y)$ is at most p_2

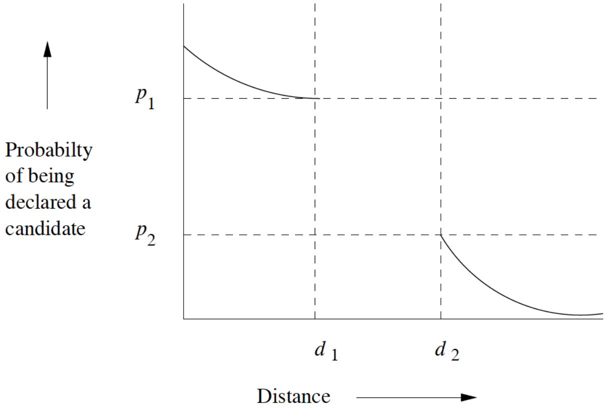
Locality Sensitive Family of Functions: Definition

- Consider functions f that hash items. The notation $f(x) = f(y)$ means that x and y form a candidate pair.
- A collection $\mathcal{F}$ of functions f of this form is called a *family of functions*
- Unless stated otherwise, $d(x, y) = 1 - \text{SIM}(x, y)$ is the Jaccard distance

Definition: [Locality Sensitive (LS) Family of Functions]
A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-*sensitive* if for each $f \in \mathcal{F}$:

1. $d(x, y) \leq d_1$ implies that the probability that $f(x) = f(y)$ is at least p_1
2. $d(x, y) \geq d_2$ implies that the probability that $f(x) = f(y)$ is at most p_2

LS Family of Function: Illustration



Behaviour of any member of a (d_1, d_2, p_1, p_2)-sensitive family of function
From `mmds.org`

LS Family of Functions: Example

Consider minhash functions.

Reminder: Minhash functions map a column in the characteristic matrix to the minimum value the rows, in which there are 1's in the column, get hashed to.

LS Family of Functions: Example

Consider minhash functions.

Reminder: Minhash functions map a column in the characteristic matrix to the minimum value the rows, in which there are 1's in the column, get hashed to.

Example: LS Family of Minhash Functions

- Consider $d(x,y) = 1 - \text{SIM}(x,y)$ to measure the distance between two sets x, y.
- Then it holds that the family of minhash functions is a $(d_1, d_2, 1-d_1, 1-d_2)$-sensitive family for any $0 \leq d_1 < d_2 \leq 1$.

LS Family of Functions: Example

Consider minhash functions.

Reminder: Minhash functions map a column in the characteristic matrix to the minimum value the rows, in which there are 1's in the column, get hashed to.

Example: LS Family of Minhash Functions

- Consider $d(x, y) = 1 - \text{SIM}(x, y)$ to measure the distance between two sets x, y.
- Then it holds that the family of minhash functions is a $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive family for any $0 \leq d_1 < d_2 \leq 1$.

LS Family of Functions: Example

Consider minhash functions.

Reminder: Minhash functions map a column in the characteristic matrix to the minimum value the rows, in which there are 1's in the column, get hashed to.

Example: LS Family of Minhash Functions

- Consider $d(x, y) = 1 - \text{SIM}(x, y)$ to measure the distance between two sets x, y.
- Then it holds that the family of minhash functions is a $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive family for any $0 \leq d_1 < d_2 \leq 1$.

Proof: By definition, $d(x, y) \leq d_1$ implies $\text{SIM}(x, y) = 1 - d(x, y) \geq 1 - d_1$. If, on the other hand, $d(x, y) \geq d_2$, we obtain $\text{SIM}(x, y) = 1 - d(x, y) \leq 1 - d_2$

Amplifying LS Families of Functions: AND-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{r,AND}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{r,AND}$ is based on r members $f_1, ..., f_r$ of $\mathcal{F}$.
-

$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for all } i = 1, ..., r \qquad (7)$$

Amplifying LS Families of Functions: AND-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{r,AND}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{r,AND}$ is based on r members $f_1, ..., f_r$ of $\mathcal{F}$.
-

$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for all } i = 1, ..., r \tag{7}$$

Amplifying LS Families of Functions: AND-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{r,AND}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{r,AND}$ is based on r members $f_1, ..., f_r$ of $\mathcal{F}$.
-
$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for all } i = 1, ..., r \tag{7}$$

Amplifying LS Families of Functions: AND-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{r,AND}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{r,AND}$ is based on r members $f_1, ..., f_r$ of $\mathcal{F}$.
- $$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for all } i = 1, ..., r \tag{7}$$

Example: Consider the members of one band of size r when applying the banding technique.

AMPLIFYING LS FAMILIES OF FUNCTIONS: AND-CONSTRUCTION

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{r,AND}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{r,AND}$ is based on r members $f_1, ..., f_r$ of $\mathcal{F}$.
- $$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for all } i = 1, ..., r \tag{7}$$

Example: Consider the members of one band of size r when applying the banding technique.
Fact: It is easy to show (consider yourself!) that $\mathcal{F}_{r,AND}$ is a $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive family of functions

AMPLIFYING LS FAMILIES OF FUNCTIONS: OR-CONSTRUCTION

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{b,OR}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{b,OR}$ is based on b members $f_1, ..., f_b$ of $\mathcal{F}$.
-

$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for one } i = 1, ..., r \tag{8}$$

Amplifying LS Families of Functions: OR-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{b,OR}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{b,OR}$ is based on b members $f_1, ..., f_b$ of $\mathcal{F}$.
- $$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for one } i = 1, ..., r \qquad (8)$$

Amplifying LS Families of Functions: OR-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{b,OR}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{b,OR}$ is based on b members $f_1, ..., f_b$ of $\mathcal{F}$.
-
$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for one } i = 1, ..., r \tag{8}$$

Amplifying LS Families of Functions: OR-Construction

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{b,OR}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{b,OR}$ is based on b members $f_1, ..., f_b$ of $\mathcal{F}$.
- $$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for one } i = 1, ..., r \tag{8}$$

Example: The OR-construction reflects the effect of combining several bands when applying the banding technique.

AMPLIFYING LS FAMILIES OF FUNCTIONS: OR-CONSTRUCTION

Consider a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$. We construct a new family $\mathcal{F}_{b,OR}$ by the following principle:

- Each single member of $f \in \mathcal{F}_{b,OR}$ is based on b members $f_1, ..., f_b$ of $\mathcal{F}$.
-
$$f(x) = f(y) \quad \Leftrightarrow \quad f_i(x) = f_i(y) \text{ for one } i = 1, ..., r \tag{8}$$

Example: The OR-construction reflects the effect of combining several bands when applying the banding technique.
Fact: It is easy to show (consider yourself again!) that $\mathcal{F}_{b,OR}$ is a $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive family of functions.

Amplifying LS Families of Functions: Locality Sensitive Hashing

Example: Applying the OR-construction to $\mathcal{F}_{r,AND}$, yielding $(\mathcal{F}_{r,AND})_{b,OR}$ reflects applying the banding technique altogether, and varying p_1, p_2 reflects reproducing the S-curve.

This justifies to study LS families of functions as a useful thing to do. For example:

- How does behaviour change when varying r and b?
  ☞ S-curve
- What happens when exhanging AND and OR?

Amplifying LS Families of Functions: Locality Sensitive Hashing

Example: Applying the OR-construction to $\mathcal{F}_{r,AND}$, yielding $(\mathcal{F}_{r,AND})_{b,OR}$ reflects applying the banding technique altogether, and varying p_1, p_2 reflects reproducing the S-curve.

This justifies to study LS families of functions as a useful thing to do. For example:

- How does behaviour change when varying r and b?
  ☞ S-curve
- What happens when exhanging AND and OR?

Amplifying LS Families of Functions: Locality Sensitive Hashing

Example: Applying the OR-construction to $\mathcal{F}_{r,AND}$, yielding $(\mathcal{F}_{r,AND})_{b,OR}$ reflects applying the banding technique altogether, and varying p_1, p_2 reflects reproducing the S-curve.

This justifies to study LS families of functions as a useful thing to do. For example:

- How does behaviour change when varying r and b?
  ☞ S-curve
- What happens when exhanging AND and OR?

Amplifying LS Families of Functions: Locality Sensitive Hashing

p	$1-(1-p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

p	$\left(1-(1-p)^4\right)^4$
0.1	0.0140
0.2	0.1215
0.3	0.3334
0.4	0.5740
0.5	0.7725
0.6	0.9015
0.7	0.9680
0.8	0.9936

Original family $\mathcal{F}$ is $(0.2, 0.6, 0.8, 0.4)$-sensitive.

Left: Applying first the AND- and then the OR-construction, reflecting locality sensitive hashing, yields a $(0.2, 0.6, 0.8785, 0.0985)$-sensitive family.

Right: Applying first the OR- and then the AND-construction, yields a $(0.2, 0.6, 0.9936, 0.5740)$-sensitive family.

LS Families for Other Distance Measures

LS Families for Hamming Distance

LS Families for Hamming Distance

- Assume we have a d-dimensional vector space V
- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- Let $f_i(x) := x_i$ be the entry of x at the i-th position
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- For randomly chosen x, y, the probability that $f_i(x) = f_i(y)$ is

$$\frac{d - h(x,y)}{d} = 1 - \frac{h(x,y)}{d}$$

  the fraction of positions in which x and y agree
- Thus, the family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is

$$(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) - \text{sensitive}$$

  for any $d_1 < d_2$

LS Families for Hamming Distance

- Assume we have a d-dimensional vector space V
- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- Let $f_i(x) := x_i$ be the entry of x at the i-th position
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- For randomly chosen x, y, the probability that $f_i(x) = f_i(y)$ is

$$\frac{d - h(x, y)}{d} = 1 - \frac{h(x, y)}{d}$$

  the fraction of positions in which x and y agree
- Thus, the family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is

$$(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) - \text{sensitive}$$

  for any $d_1 < d_2$

LS Families for Hamming Distance

- Assume we have a d-dimensional vector space V
- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- Let $f_i(x) := x_i$ be the entry of x at the i-th position
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- For randomly chosen x, y, the probability that $f_i(x) = f_i(y)$ is

$$\frac{d - h(x, y)}{d} = 1 - \frac{h(x, y)}{d}$$

  the fraction of positions in which x and y agree
- Thus, the family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is

$$(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) - \text{sensitive}$$

  for any $d_1 < d_2$

LS Families for Hamming Distance

- Assume we have a d-dimensional vector space V
- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- Let $f_i(x) := x_i$ be the entry of x at the i-th position
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- For randomly chosen x, y, the probability that $f_i(x) = f_i(y)$ is

$$\frac{d - h(x, y)}{d} = 1 - \frac{h(x, y)}{d}$$

  the fraction of positions in which x and y agree
- Thus, the family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is

$$(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) - \text{sensitive}$$

  for any $d_1 < d_2$

LS FAMILIES FOR HAMMING DISTANCE

- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- The family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is $(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) -$ sensitive for any $d_1 < d_2$

DIFFERENCES

- Jaccard distance runs from 0 to 1, Hamming distance from 0 to d: need to scale with $1/d$
- There is an unlimited number of minhash functions, but size of $\mathcal{F}$ is only d
- The limited size of $\mathcal{F}$ puts limits to AND/OR constructions

LS Families for Hamming Distance

- Let $h(x, y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- The family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is $(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d}) -$ sensitive for any $d_1 < d_2$

Differences

- Jaccard distance runs from 0 to 1, Hamming distance from 0 to d: need to scale with $1/d$
- There is an unlimited number of minhash functions, but size of $\mathcal{F}$ is only d
- The limited size of $\mathcal{F}$ puts limits to AND/OR constructions

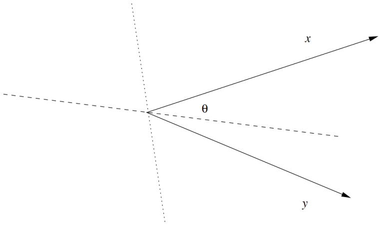
LS Families for Hamming Distance

- Let $h(x,y)$ be the Hamming distance between vectors $x = (x_1, ..., x_d), y = (y_1, ..., y_d) \in V$
- So $f_i(x) = f_i(y)$ if and only if $x_i = y_i$
- The family $\mathcal{F}$ of $\{f_1, ..., f_d\}$ is $(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d})$ − sensitive for any $d_1 < d_2$

Differences

- Jaccard distance runs from 0 to 1, Hamming distance from 0 to d: need to scale with $1/d$
- There is an unlimited number of minhash functions, but size of $\mathcal{F}$ is only d
- The limited size of $\mathcal{F}$ puts limits to AND/OR constructions

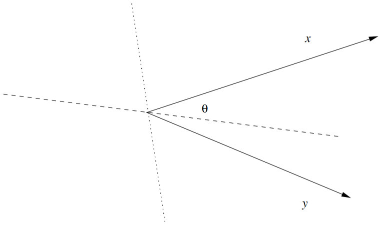
LS Families for Cosine Distance



Two vectors making an angle θ
From mmds.org

- Cosine distance for $x, y \in V$ corresponds with the angle $\theta(x, y) \in [0, 180]$ between them
- Whatever the dimension $d = \dim V$, two vectors x, y span a plane $V(x, y)$ (so $\dim V(x, y) = 2$)
- Angle θ is measured in that plane $V(x, y)$

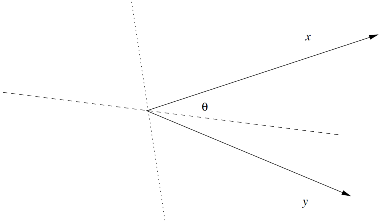
LS FAMILIES FOR COSINE DISTANCE



Two vectors making an angle θ
From mmds.org

- Cosine distance for $x, y \in V$ corresponds with the angle $\theta(x, y) \in [0, 180]$ between them
- Whatever the dimension $d = \dim V$, two vectors x, y span a plane $V(x, y)$ (so $\dim V(x, y) = 2$)
- Angle θ is measured in that plane $V(x, y)$

LS Families for Cosine Distance: Random Hyperplanes

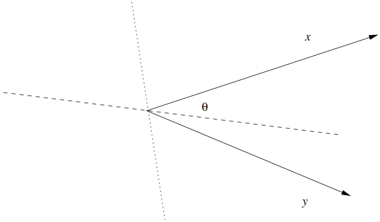


Two vectors making an angle θ
From mmds.org

- Any hyperplane (dimension $\dim V - 1$) intersects $V(x, y)$ in a line
- Figure: two hyperplanes, indicated by dotted and dashed line
- Determine hyperplanes U by picking normal vectors v
- That is

$$U = \{u \in V \mid \langle u, v \rangle = 0\}$$

LS Families for Cosine Distance: Random Hyperplanes

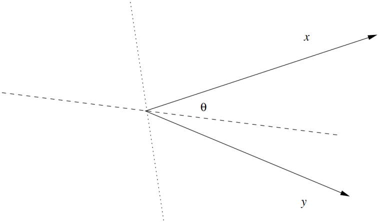


Two vectors making an angle θ
From mmds.org

- Any hyperplane (dimension $\dim V - 1$) intersects $V(x, y)$ in a line
- Figure: two hyperplanes, indicated by dotted and dashed line
- Determine hyperplanes U by picking normal vectors v
- That is

$$U = \{u \in V \mid \langle u, v \rangle = 0\}$$

LS Families for Cosine Distance: Random Hyperplanes



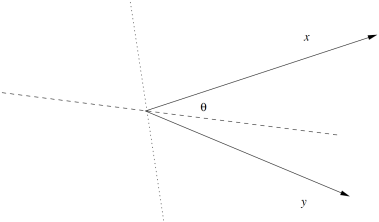
Two vectors making an angle θ
From mmds.org

- Consider dashed line hyperplane U: x and y on different sides
- Let v be normal vector of U:

$$\operatorname{sgn}\langle x, v\rangle \neq \operatorname{sgn}\langle y, v\rangle$$

so one scalar product is positive and the other one is negative

LS Families for Cosine Distance: Random Hyperplanes



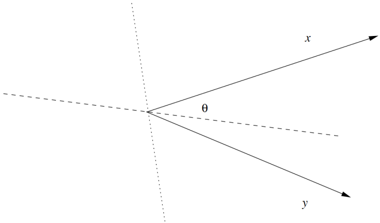
Two vectors making an angle θ
From mmds.org

- Consider dashed line hyperplane U: x and y on different sides
- Let v be normal vector of U:

$$\mathrm{sgn}\langle x, v\rangle \neq \mathrm{sgn}\langle y, v\rangle$$

so one scalar product is positive and the other one is negative

LS Families for Cosine Distance: Random Hyperplanes



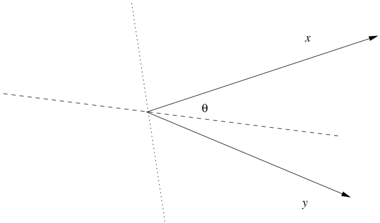
Two vectors making an angle θ
From mmds.org

- Consider dotted line hyperplane U: x and y on the same side
- Let v be normal vector of U:

$$\operatorname{sgn}\langle x, v\rangle = \operatorname{sgn}\langle y, v\rangle$$

so both scalar products positive or both negative

LS Families for Cosine Distance: Random Hyperplanes



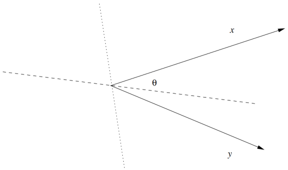
Two vectors making an angle θ
From mmds.org

- Consider dotted line hyperplane U: x and y on the same side
- Let v be normal vector of U:

$$\mathrm{sgn}\langle x, v\rangle = \mathrm{sgn}\langle y, v\rangle$$

so both scalar products positive or both negative

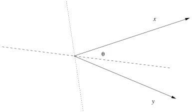
LS Families for Cosine Distance: Random Hyperplanes



Two vectors making an angle θ
From mmds.org

- Probability to choose x, y at an angle $\theta(x, y)$ and
  - hyperplane like dashed line: $\theta(x, y)/180$
  - hyperplane like dotted line: $(180 - \theta(x, y))/180$
- Consider hash functions f corresponding to randomly picked normal vectors v_f

LS Families for Cosine Distance: Random Hyperplanes



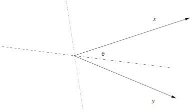
Two vectors making an angle θ
From mmds.org

- Consider family $\mathcal{F}$ of hash functions f corresponding to randomly picked hyperplanes, represented by their normal vectors v_f
- For $x, y \in V$, let

$$f(x) = f(y) \quad \text{if and only if} \quad \text{sgn}\langle v_f, x\rangle = \text{sgn}\langle v_f, y\rangle$$

- $\mathcal{F}$ is $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-sensitive
- One can amplify the family as desired
  - Apart from rescaling by 180, $\mathcal{F}$ is just like minhash family

LS Families for Cosine Distance: Random Hyperplanes



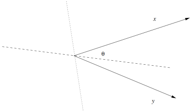
Two vectors making an angle θ
From mmds.org

- Consider family $\mathcal{F}$ of hash functions f corresponding to randomly picked hyperplanes, represented by their normal vectors v_f
- For $x, y \in V$, let

$$f(x) = f(y) \quad \text{if and only if} \quad \operatorname{sgn}\langle v_f, x\rangle = \operatorname{sgn}\langle v_f, y\rangle$$

- $\mathcal{F}$ is $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-sensitive
- One can amplify the family as desired
    - Apart from rescaling by 180, $\mathcal{F}$ is just like minhash family

LS Families for Cosine Distance: Random Hyperplanes



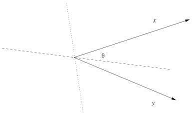
Two vectors making an angle θ
From mmds.org

- Consider family $\mathcal{F}$ of hash functions f corresponding to randomly picked hyperplanes, represented by their normal vectors v_f
- For $x, y \in V$, let

$$f(x) = f(y) \quad \text{if and only if} \quad \operatorname{sgn}\langle v_f, x\rangle = \operatorname{sgn}\langle v_f, y\rangle$$

- $\mathcal{F}$ is $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-sensitive
- One can amplify the family as desired
  - Apart from rescaling by 180, $\mathcal{F}$ is just like minhash family

LS Families for Cosine Distance: Random Hyperplanes



Two vectors making an angle θ
From mmds.org

- Consider family $\mathcal{F}$ of hash functions f corresponding to randomly picked hyperplanes, represented by their normal vectors v_f
- For $x, y \in V$, let

$$f(x) = f(y) \quad \text{if and only if} \quad \text{sgn}\langle v_f, x\rangle = \text{sgn}\langle v_f, y\rangle$$

- $\mathcal{F}$ is $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-sensitive
- One can amplify the family as desired
  - Apart from rescaling by 180, $\mathcal{F}$ is just like minhash family

Sampling Random Normal Vectors: Sketches

- When determining normal vectors of random hyperplanes, it can be shown that it suffices to pick random vectors with entries either -1 or $+1$
- Let $v_1, ..., v_n$ be such random vectors
- For a vector x, the array

$$[\mathrm{sgn}\langle v_1, x\rangle, ..., \mathrm{sgn}\langle v_n, x\rangle] \in [-1, +1]^n \quad (9)$$

is said to be the *sketch* of x

Sampling Random Normal Vectors: Sketches

- When determining normal vectors of random hyperplanes, it can be shown that it suffices to pick random vectors with entries either -1 or $+1$
- Let $v_1, ..., v_n$ be such random vectors
- For a vector x, the array

$$[\text{sgn}\langle v_1, x\rangle, ..., \text{sgn}\langle v_n, x\rangle] \in [-1, +1]^n \tag{9}$$

is said to be the *sketch* of x

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

UNIVERSITÄT BIELEFELD

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

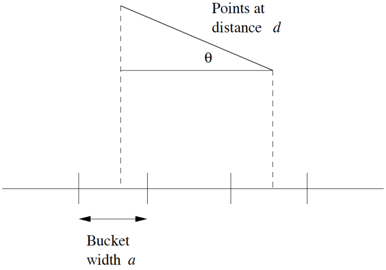
Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

Sketches: Example

- Let $x = [3, 4, 5, 6], y = [4, 3, 2, 1]$
- Let $v_1 = [+1, -1, +1, +1], v_2 = [-1, +1, -1, +1], v_3 = [+1, +1, -1, -1]$
- Then
  - Sketch of x is $[+1, +1, -1]$
  - Sketch of y is $[+1, -1, +1]$
  - Sketches of x, y agree in 1 out of 3 positions: we estimate $\widehat{\theta(x, y)} = 120$
  - However true $\theta(x, y) = 38$
- There are 16 different vectors with $+1, -1$ (cardinality of $\{-1, +1\}^4$ is 16)
- Computing sketches based on all of them yields estimate $\widehat{\theta(x, y)} = 45$

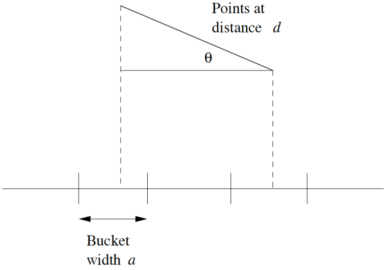
LS FAMILIES FOR EUCLIDEAN DISTANCE



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- Let us consider 2-dimensional space V
- Each member f of family $\mathcal{F}$ is associated with line in V
- Line is divided into buckets (segments) of length a
- Points $x, y \in V$ are "hashed" to buckets
- $f(x) = f(y)$ when hashed to the same segment

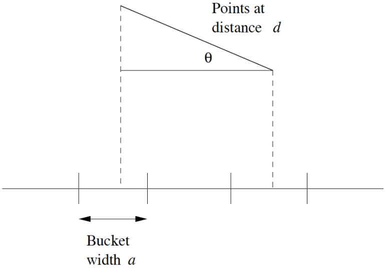
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- Let us consider 2-dimensional space V
- Each member f of family $\mathcal{F}$ is associated with line in V
- Line is divided into buckets (segments) of length a
- Points $x, y \in V$ are "hashed" to buckets
- $f(x) = f(y)$ when hashed to the same segment

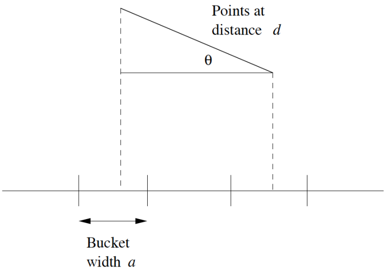
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- Let us consider 2-dimensional space V
- Each member f of family $\mathcal{F}$ is associated with line in V
- Line is divided into buckets (segments) of length a
- Points $x, y \in V$ are "hashed" to buckets
- $f(x) = f(y)$ when hashed to the same segment

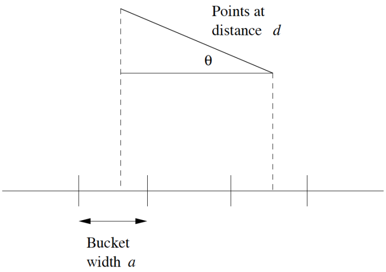
LS FAMILIES FOR EUCLIDEAN DISTANCE



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- Let us consider 2-dimensional space V
- Each member f of family $\mathcal{F}$ is associated with line in V
- Line is divided into buckets (segments) of length a
- Points $x, y \in V$ are "hashed" to buckets
- $f(x) = f(y)$ when hashed to the same segment

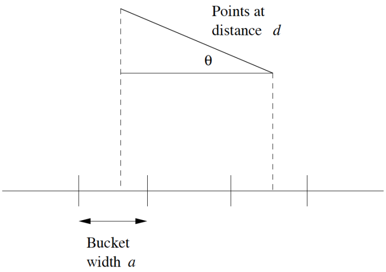
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- If Euclidean distance $d(x, y) \leq a/2$, then probability to hash x, y to same segment is at least $1/2$
  - Distance between x, y after projecting is $d(x, y) \cos\theta \leq d(x, y) \leq a/2$

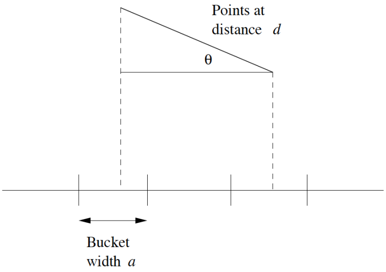
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- If Euclidean distance $d(x, y) \leq a/2$, then probability to hash x, y to same segment is at least $1/2$
  - Distance between x, y after projecting is $d(x, y) \cos \theta \leq d(x, y) \leq a/2$

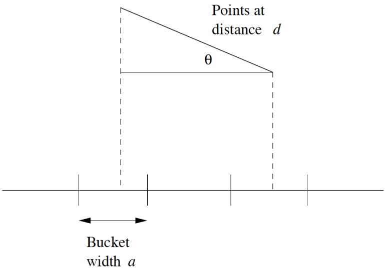
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- If distance between x, y after projecting is greater than a, they will be hashed to different buckets
- So, if $d(x,y) \geq 2a$, we have that $d(x,y) \cos\theta > a$ for $\theta \in [0, 60]$
- It holds that $\theta \in [0, 60]$ with probability $2/3$ (note: here $\theta \in [0, 90]$)

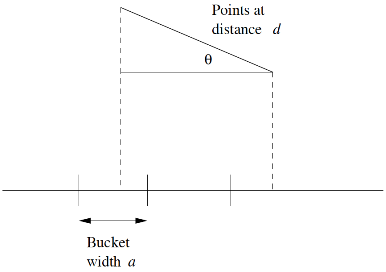
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- If distance between x, y after projecting is greater than a, they will be hashed to different buckets
- So, if $d(x, y) \geq 2a$, we have that $d(x, y) \cos\theta > a$ for $\theta \in [0, 60]$
- It holds that $\theta \in [0, 60]$ with probability $2/3$ (note: here $\theta \in [0, 90]$)

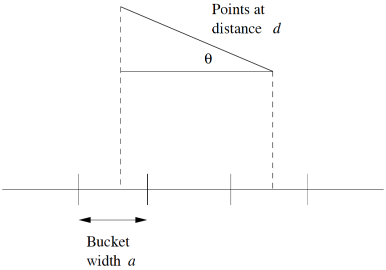
LS Families for Euclidean Distance



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- If distance between x, y after projecting is greater than a, they will be hashed to different buckets
- So, if $d(x,y) \geq 2a$, we have that $d(x,y)\cos\theta > a$ for $\theta \in [0, 60]$
- It holds that $\theta \in [0, 60]$ with probability $2/3$ (note: here $\theta \in [0, 90]$)

LS Families for Euclidean Distance



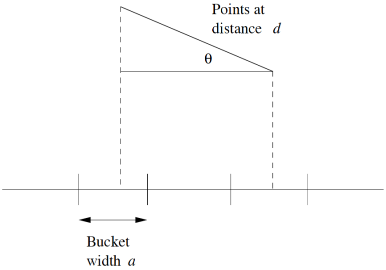
Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- In conclusion, the family described has been

$$(a/2, 2a, 1/2, 1/3) - \text{sensitive}$$

- Family can be amplified as desired
- If families for arbitrary $d_1 < d_2$ (and not just $d_1 = a/2, d_2 = 2a$), and also for arbitrary-dimensional vector spaces are desired, special efforts are required

LS FAMILIES FOR EUCLIDEAN DISTANCE



Two points at distance $d >> a$ are hashed to identical bucket with small probability
From mmds.org

- In conclusion, the family described has been

$$(a/2, 2a, 1/2, 1/3) - \text{sensitive}$$

- Family can be amplified as desired
- If families for arbitrary $d_1 < d_2$ (and not just $d_1 = a/2, d_2 = 2a$), and also for arbitrary-dimensional vector spaces are desired, special efforts are required

Materials / Outlook

- See *Mining of Massive Datasets*, chapter 3.4–3.7
- See `http://www.mmds.org/` for further resources
- Next lecture: “Map Reduce / Workflow Systems I”
  - See *Mining of Massive Datasets* 2.1–2.4