Similarity Search

Stony Brook University
CSE545, Fall 2016

Finding Similar Items

- Applications
  - Document Similarity:
    - Mirrored web-pages
    - Plagiarism; Similar News
  - Recommendations:
    - Online purchases
    - Movie ratings
  - Entity Resolution
  - Fingerprint Matching

Finding Similar Items: What we will cover

- Set Similarity
  - Shingling
  - Minhashing
  - Locality-sensitive hashing

- Embeddings

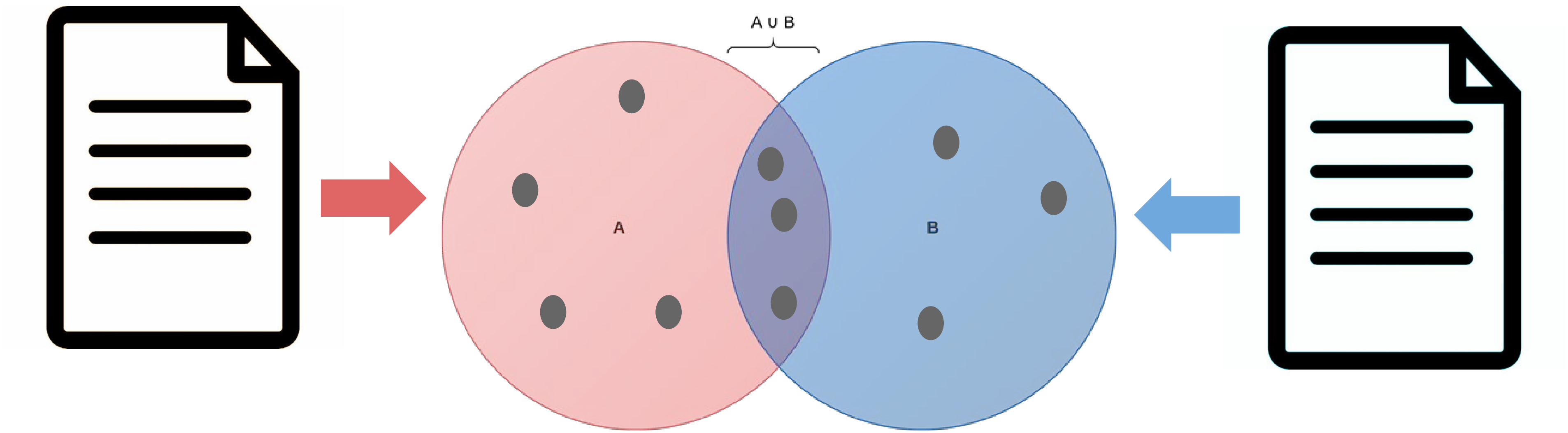
- Distance Metrics
- High-Degree of Similarity

Document Similarity

Challenge: How to represent the document in a way that can be efficiently encoded and compared?

Shingles

Goal: Convert documents to sets



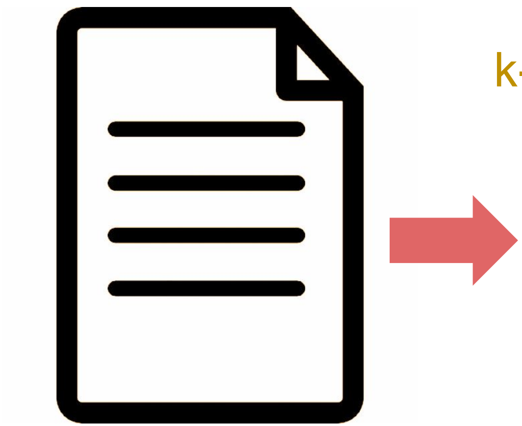
Shingles

Goal: Convert documents to sets

k-shingles (aka "character n-grams") - sequence of *k* characters

Shingles

Goal: Convert documents to sets

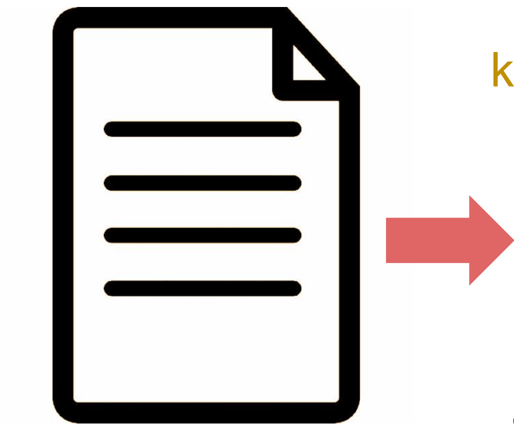


k-shingles (aka “character n-grams”) - sequence of *k* characters

E.g. k=2 doc=”abcdabd”
singles(doc, 2) = {ab, bc, cd, da, bd}

Shingles

Goal: Convert documents to sets



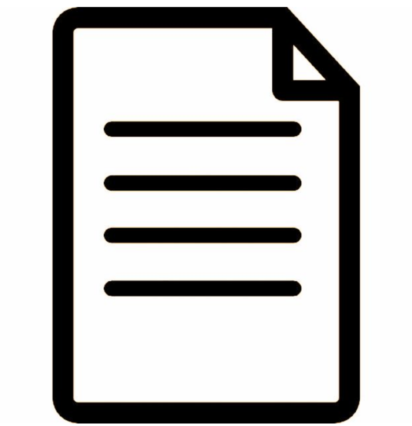
k-shingles (aka “character n-grams”) - sequence of *k* characters

E.g. k=2 doc=”abcdabd”
singles(doc, 2) = {ab, bc, cd, da, bd}

- Similar documents will have many common shingles
- Changing words or order has minimal effect.
- In practice use 5 < k < 10

Shingles

Goal: Convert documents to sets



Large enough that any given shingle appearing a document is highly unlikely (e.g. < .1% chance)

Can hash large singles to smaller (e.g. 9-shingles into 4 bytes)

Can also use words (aka n-grams).

- In practice use $5 < k < 10$

Shingles

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Minhashing

Goal: Convert sets to shorter ids, signatures

Minhashing - Background

Goal: Convert sets to shorter ids, signatures

Characteristic Matrix:

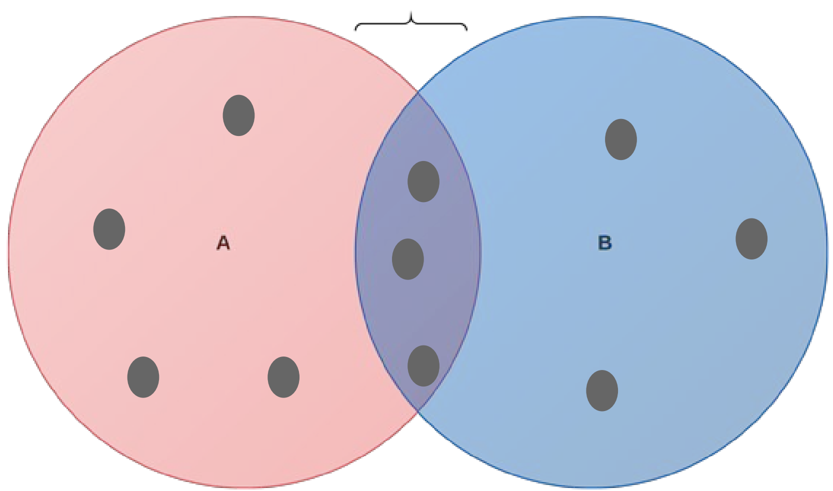
Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

....

(Leskovec at al., 2014; http://www.mmds.org/)

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$



Minhashing - Background

Goal: Convert sets to shorter ids, signatures

Characteristic Matrix:

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

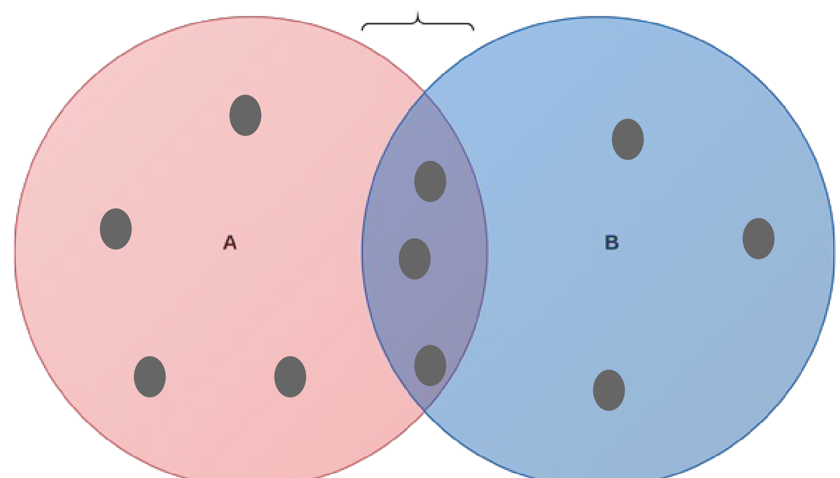
....

(Leskovec at al., 2014; http://www.mmds.org/)

often very sparse! (lots of zeros)

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$



Minhashing - Background

Characteristic Matrix:

	S_1	S_2
ab	1	1
bc	0	1
de	1	0
ah	1	1
ha	0	0
ed	1	1
ca	0	1

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

$sim(S_{1,} S_2) = 3 / 6$ (# both have / # at least one has)

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

How many different rows are possible?

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

How many different rows are possible?

1, 1 -- type a

1, 0 -- type b

0, 1 -- type c

0, 0 -- type d

$sim(S_{1,} S_2) = a / (a+b+c)$

Shingles

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ =

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ =

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ = ha #permuted row 1

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2
$h_1(S_2)$ = ha #permuted row 1
$h_1(S_3)$ = ed #permuted row 2
$h_1(S_4)$ = ha #permuted row 1

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ = ha #permuted row 1

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ = ha #permuted row 1

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Characteristic Matrix:

				S_1	S_2	[illegible]	S_4
1	4	3	ab	1	0	[illegible]	0
3	2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	3	6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Estimate with a random sample of permutations (i.e. ~100)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0		0
3	2						
7							
6	3	6	ah				
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Estimate with a random sample of permutations (i.e. ~100)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

To implement

Problem:

- Can’t actually do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Minhashing

To implement

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Solution: Use “random” hash functions.

- Setup:
    - Pick ~100 hash functions, hashes
    - Store M[i][s] = a potential minimum $h_i(r)$ *#initialized to infinity (num hashs x num sets)*

Minhashing

To implement

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Solution: Use “random” hash functions.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$ *#initialized to infinity (num hashs x num sets)*
- Algorithm:

```
for r in rows of cm: #cm is characteristic matrix
    compute h_i(r) for all i in hashes #produces 100 precomputed values
    for each set s in row r:
        if cm[r][s] == 1:
            for i in hashes: #check which hash produces smallest value
                h_i(r) < M[i][s]: M[i][s] = h_i(r)
```

Minhashing

To implement

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Solution: Use “random” hash functions.

Known as “efficient minhashing”.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$ *#initialized to infinity (num hashs x num sets)*
- Algorithm:

```
for r in rows of cm: #cm is characteristic matrix
    compute h_i(r) for all i in hashes #produces 100 precomputed values
    for each set s in row r:
        if cm[r][s] == 1:
            for i in hashes: #check which hash produces smallest value
                h_i(r) < M[i][s]: M[i][s] = h_i(r)
```

Minhashing

What hash functions to use?

Start with a decent function

E.g. $h_1(x)$ = ascii(string) % large_prime_number

Add a random multiple and addition

E.g. $h_2(x) = (a$*ascii(string) + b) % large_prime_number

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

New Problem: Even if the size of signatures are small, it can be computationally expensive to find similar pairs.

E.g. 1m documents; 1,000,000 choose 2 = 500,000,000,000 pairs

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

If we wanted the similarity for all pairs of documents, could anything be done?

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Approach: Hash multiple times: similar items are likely in the same bucket.

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Approach: Hash multiple times: similar items are likely in the same bucket.

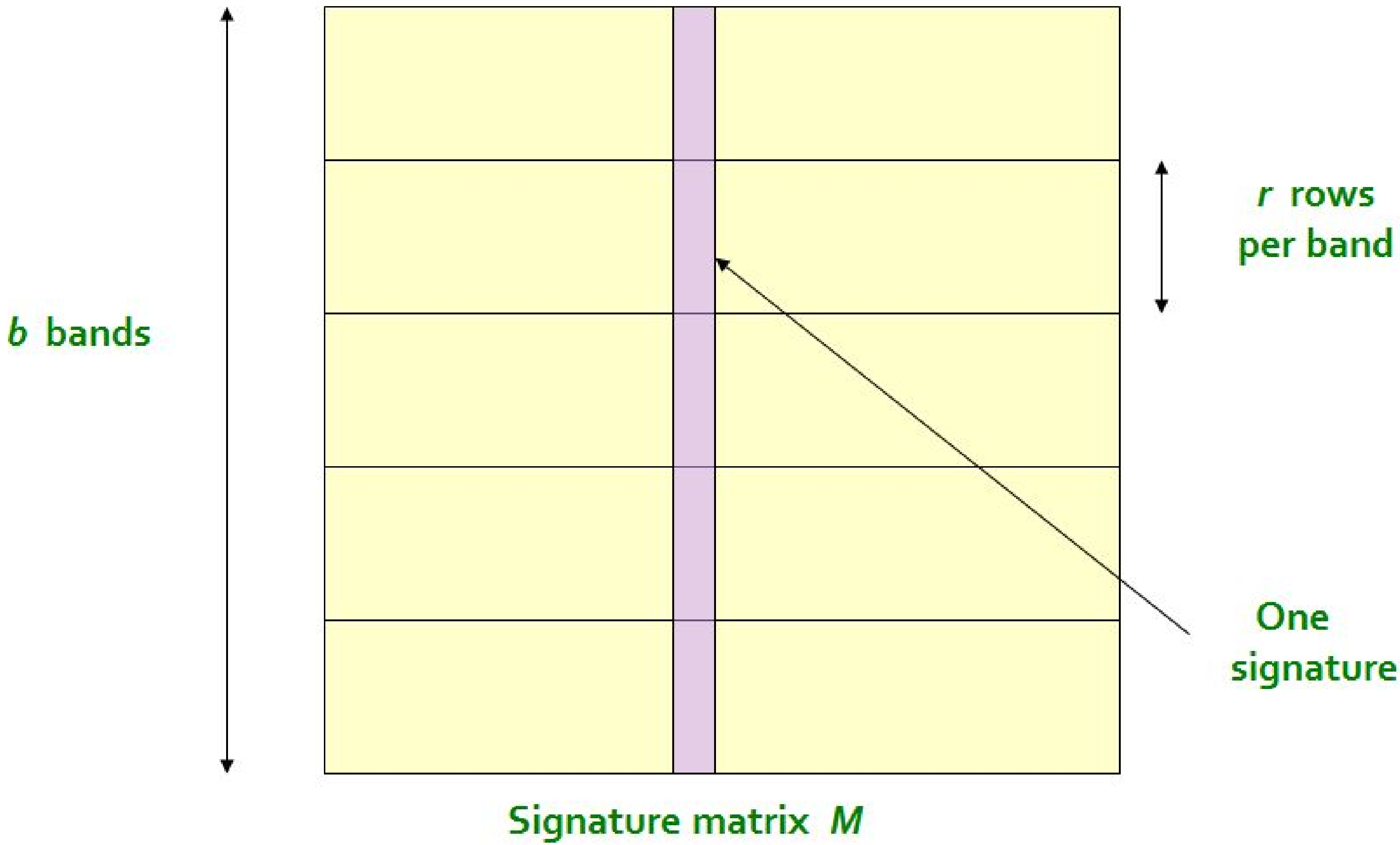
Approach from MinHash: Hash columns of signature matrix

⟹ Candidate pairs end up in the same bucket.

(*LSH* is a type of *near-neighbor search)*

Step 1: Add bands

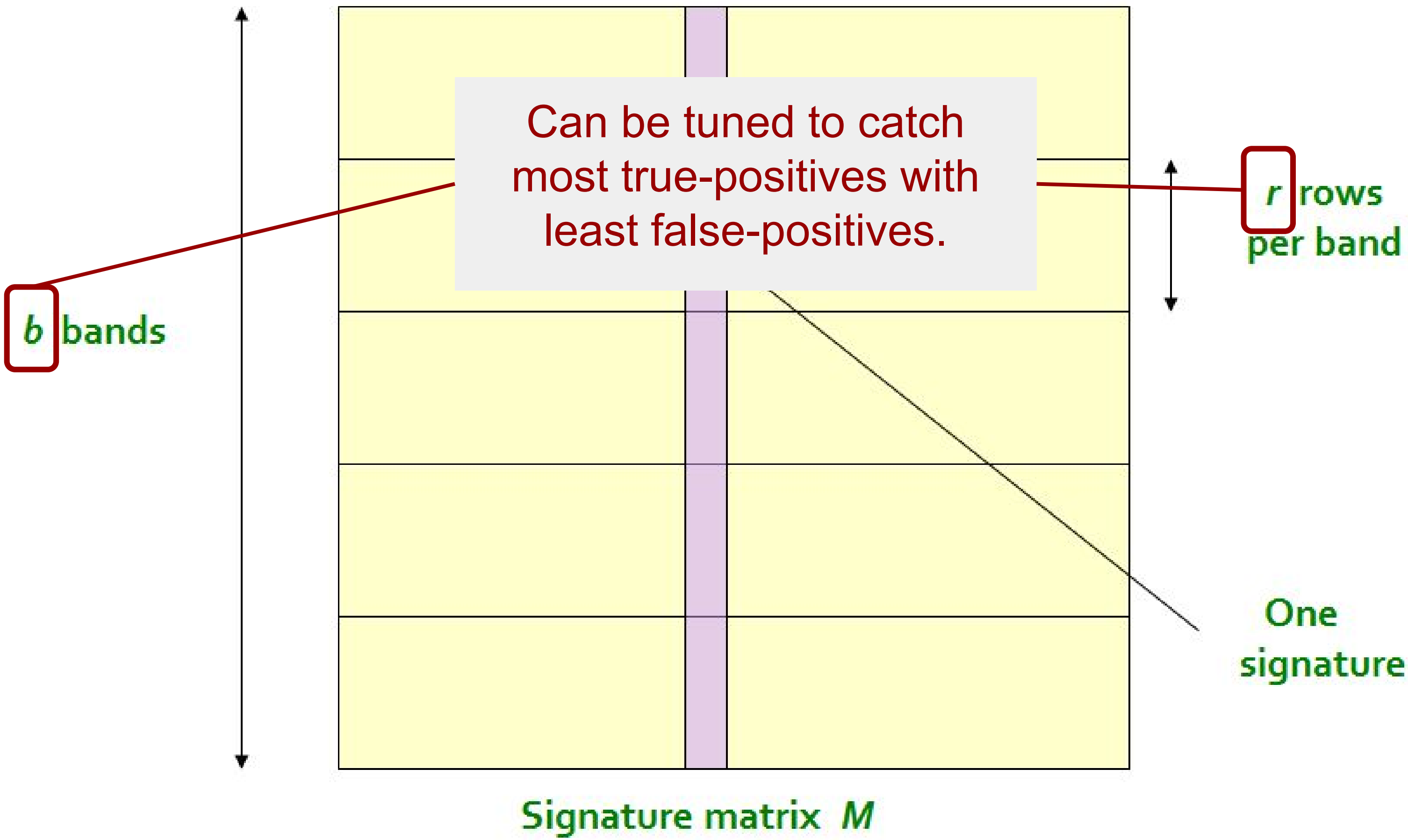
Locality-Sensitive Hashing



(Leskovec at al., 2014; http://www.mmds.org/)

Step 1: Add bands

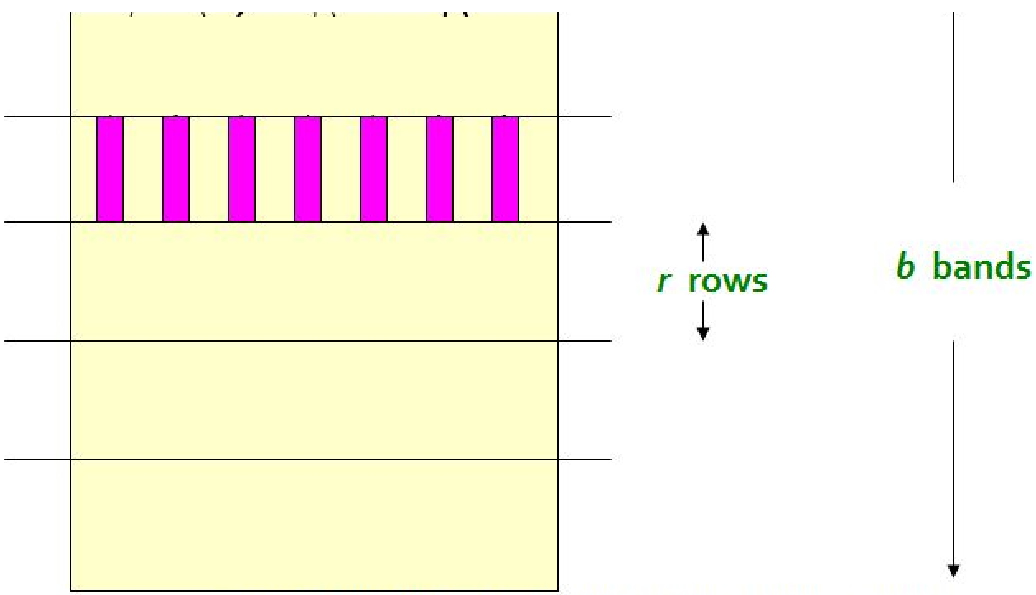
Locality-Sensitive Hashing



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

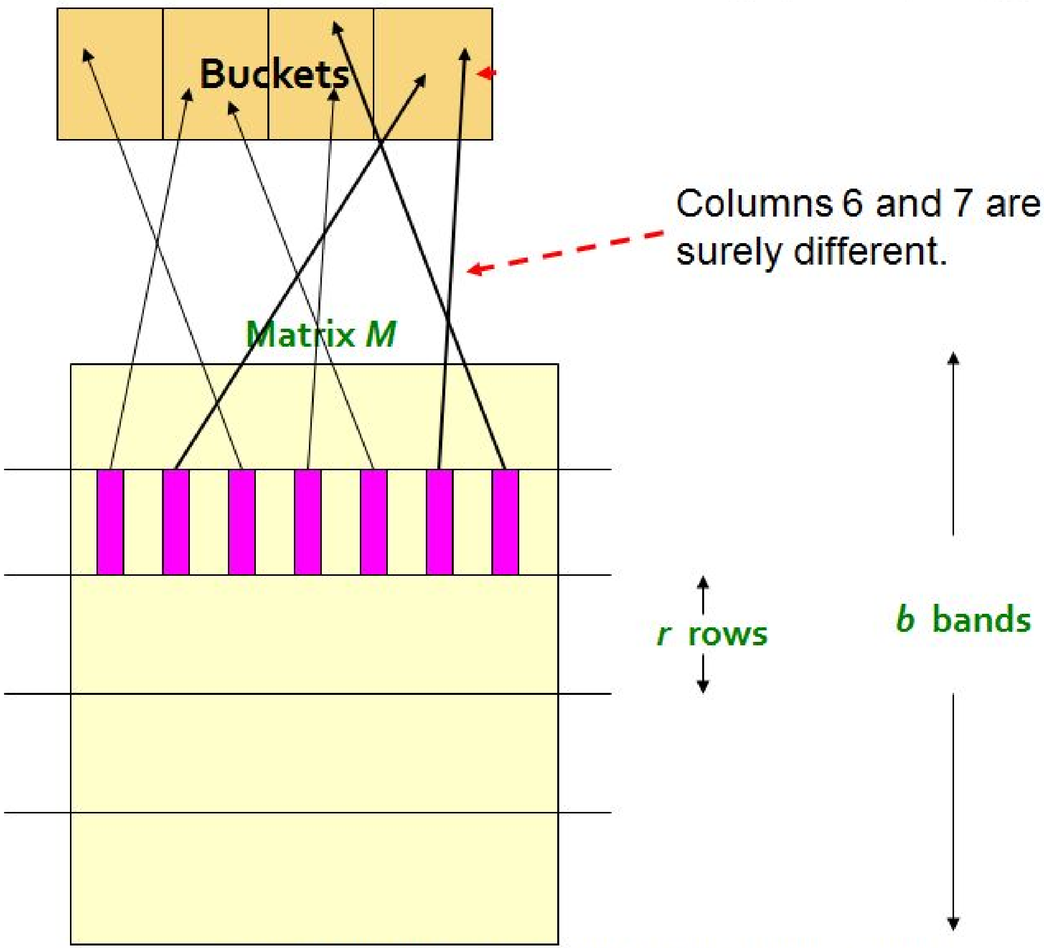
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

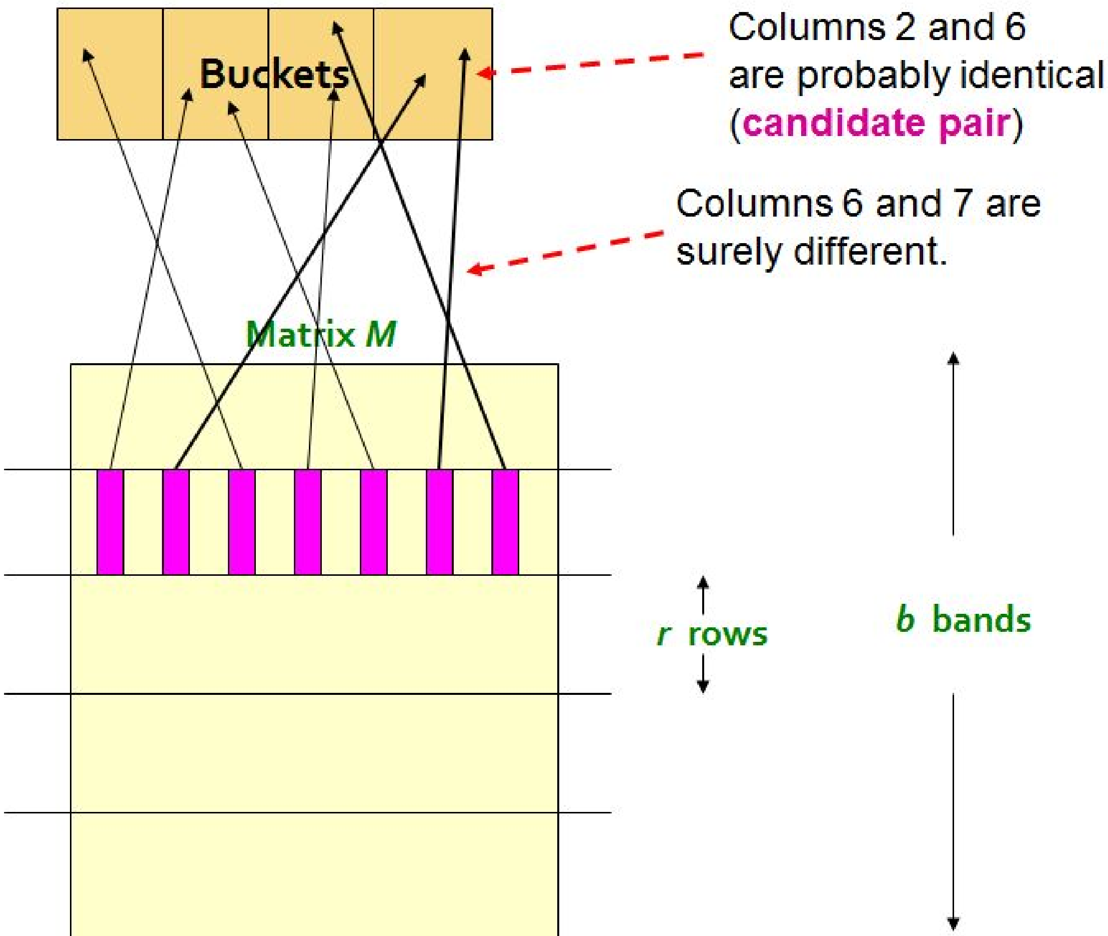
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

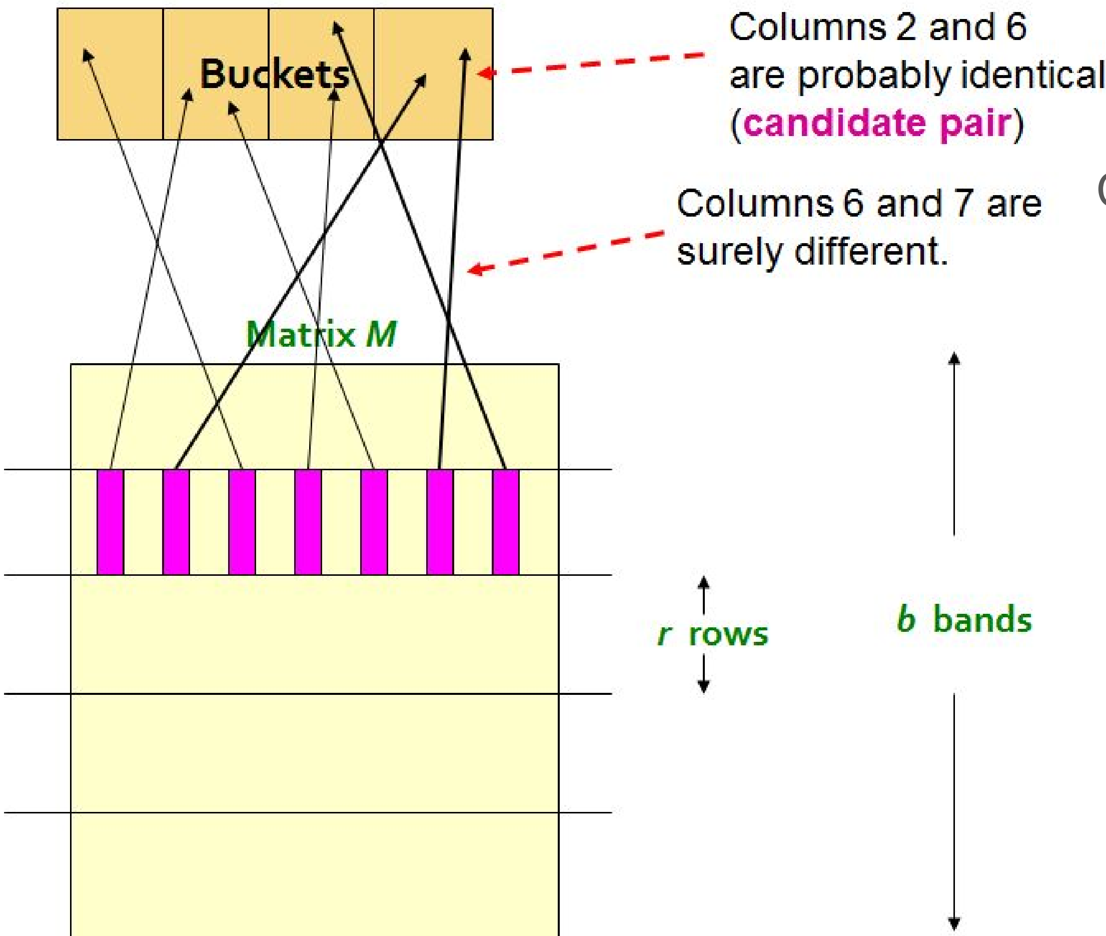
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

Step 2: Hash columns within bands



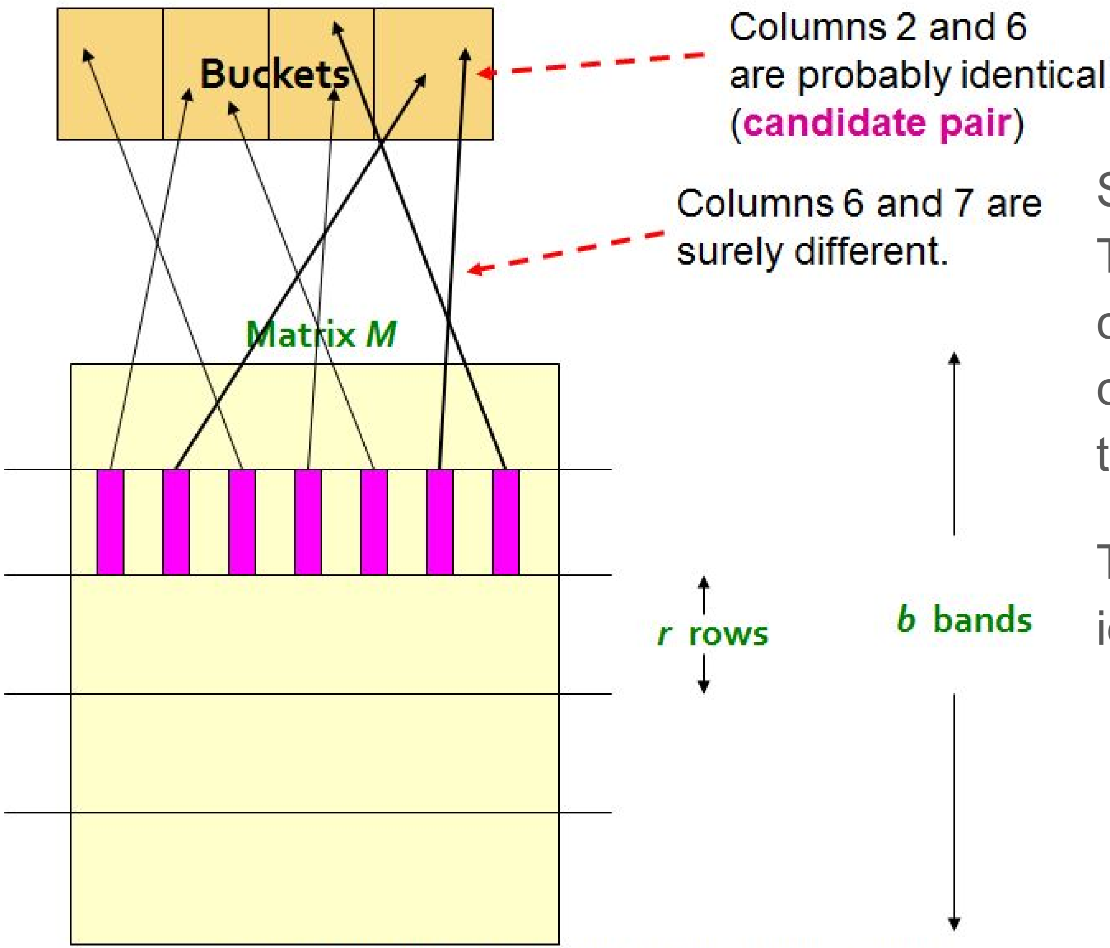
Criteria for being candidate pair:

- They end up in same bucket for at least 1 band.

(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

Step 2: Hash columns within bands



Simplification:
There are enough buckets compared to rows per band that columns must be identical in order to hash to the same bucket.

Thus, we only need to check if identical within a band.

(Leskovec at al., 2014; http://www.mmds.org/)

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity

$P(S_1 == S_2 \mid b)$: probability S1 and S2 agree within a given band

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{==}S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad => \quad P(S_1 != S_2 \mid b) = 1-.328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1==S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad => \quad P(S_1 != S_2 \mid b) = 1-.328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

$= .672^{20} = .00035$

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1 == S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1 - .328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

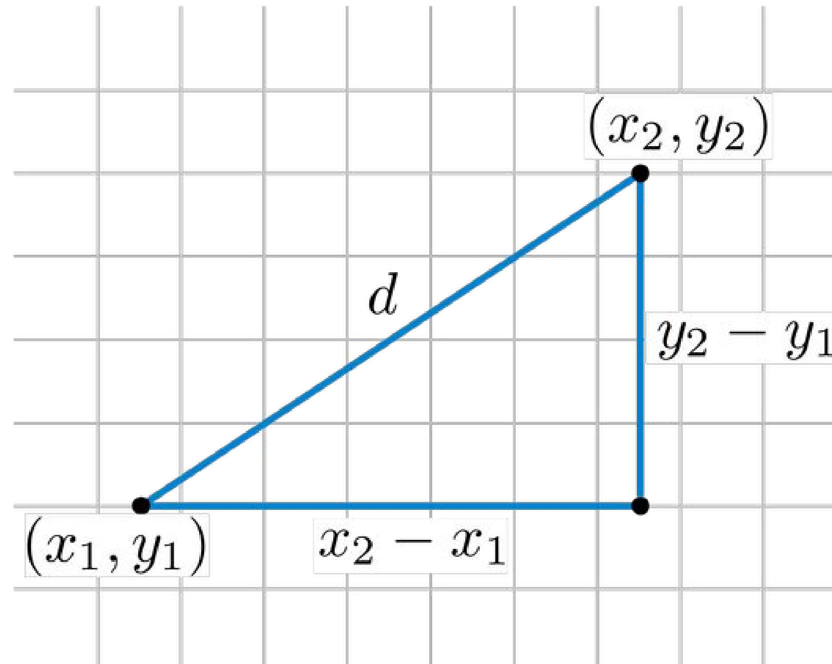
$= .672^{20} = .00035$

What if wanting 40% Jaccard Similarity?

Document Similarity Pipeline

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

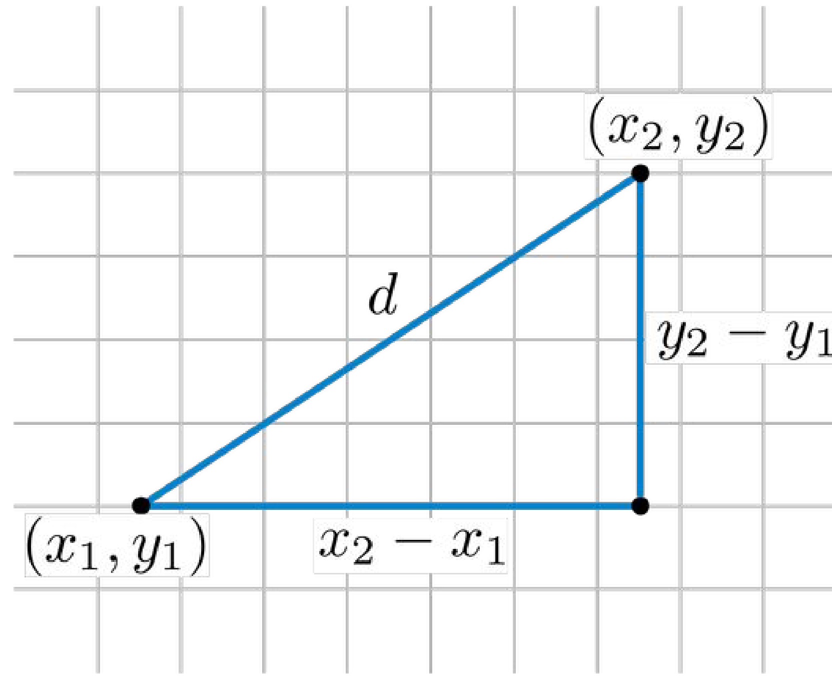
Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

Typical properties, d: distance metric

d(x, x) = 0

d(x, y) = d(y, x)

d(x, y) ≤ d(x,z) + d(z,y)



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance
- Cosine Distance

  …

- Edit Distance
- Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance

$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")

- Cosine Distance

…

Edit Distance

Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

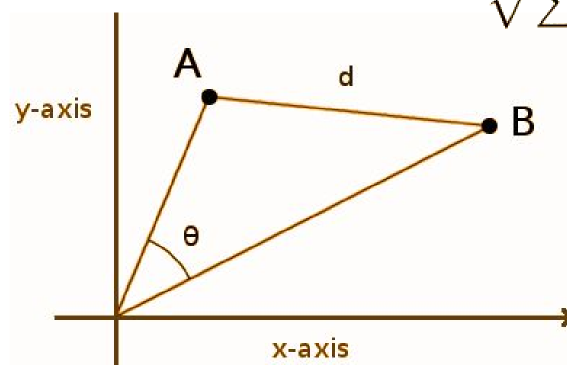
- Euclidean Distance

$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$

("L2 Norm")

- Cosine Distance

$$distance(X,Y) = 1 - \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$



…

Edit Distance

Hamming Distance

Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

E.g. for euclidean distance:

- Choose random lines (analogous to hash functions in minhashing)
- Project the two points onto each line; match if two points within an interval