Data Mining and Analysis

Finding similar items

CSCE 676 :: Fall 2019
Texas A&M University
Department of Computer Science & Engineering
Prof. James Caverlee

Resources

MMDS Chapter 3 + slides

- http://i.stanford.edu/~ullman/mmds/ch3n.pdf
- http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

Carlos Castillo course on Data Mining [https://github.com/chatox/data-mining-course]

How to avoid permuting rows?

Generating lots of permutations for min-hashing is expensive.

Instead, hash rows —> one-pass implementation

Single-pass method

```
for each row r
    for each hash function h_i
        compute h_i(r)
    for each column c
        if c has 1 in row r
            for each hash function h_i
                if h_i(r) < M(i,c)
                then M(i,c) <— h_i(r)
```

Single-pass method: Example

Row	C1	C2	h(x)	g(x)
1	1	0	1	3
2	0	1	2	0
3	1	1	3	2
4	1	0	4	4
5	0	1	0	1

$h(x) = x \bmod 5$

$g(x) = (2x+1) \bmod 5$

	M(i,C1)	M(i,C2)
initial	Int.MAX	Int.MAX
initial	Int.MAX	Int.MAX
h(1)=1	**1**	Int.MAX
g(1)=3	**3**	Int.MAX
h(2)=2	1	**2**
g(2)=0	3	**0**
h(3)=3	1	2
g(3)=2	**2**	0
h(4)=4	1	2
g(4)=4	2	0
h(5)=0	1	**0**
g(5)=1	2	0

Row	C1	C2	h(x)	g(x)
1	1	0	1	3
2	0	1	2	0
3	1	1	3	2
4	1	0	4	4
5	0	1	0	1

Locality-sensitive hashing

(Focus on pairs of signatures likely to be from similar documents)

So far …

We have converted documents into sets of shingles

We have transformed these sets into signatures using min-hash

where the signatures preserve the similarity in the original shingle space

But, we still need to compare all pairs of signatures to find similar items!

Today: LSH to focus on pairs of signatures likely to be from similar documents

LSH: first idea

- **Goal:** Find documents with Jaccard similarity at least ***s*** (for some similarity threshold, e.g., ***s***=0.8)

- **LSH – General idea:** Use a function ***f(x,y)*** that tells whether ***x*** and ***y*** is a ***candidate pair***: a pair of elements whose similarity must be evaluated

- **For Min-Hash matrices:**
  - Hash columns of signature matrix ***M*** to many buckets
  - Each pair of documents that hashes into the same bucket is a **candidate pair**

Signature matrix M

2	1	4	1
1	2	1	2
2	1	2	1

Selecting Candidates

- **Pick a similarity threshold *s* (0 < s < 1)**
- Columns ***x*** and ***y*** of ***M*** are a **candidate pair** if their signatures (***M* (*i*, *x*) = *M* (*i*, *y*)**) agree on at least fraction ***s*** of their rows
- We expect documents ***x*** and ***y*** to have the same (Jaccard) similarity as their signatures

Signature matrix M

2	1	4	1
1	2	1	2
2	1	2	1

Creating buckets of similar documents

- **Big idea: Hash columns of signature matrix *M* several times**

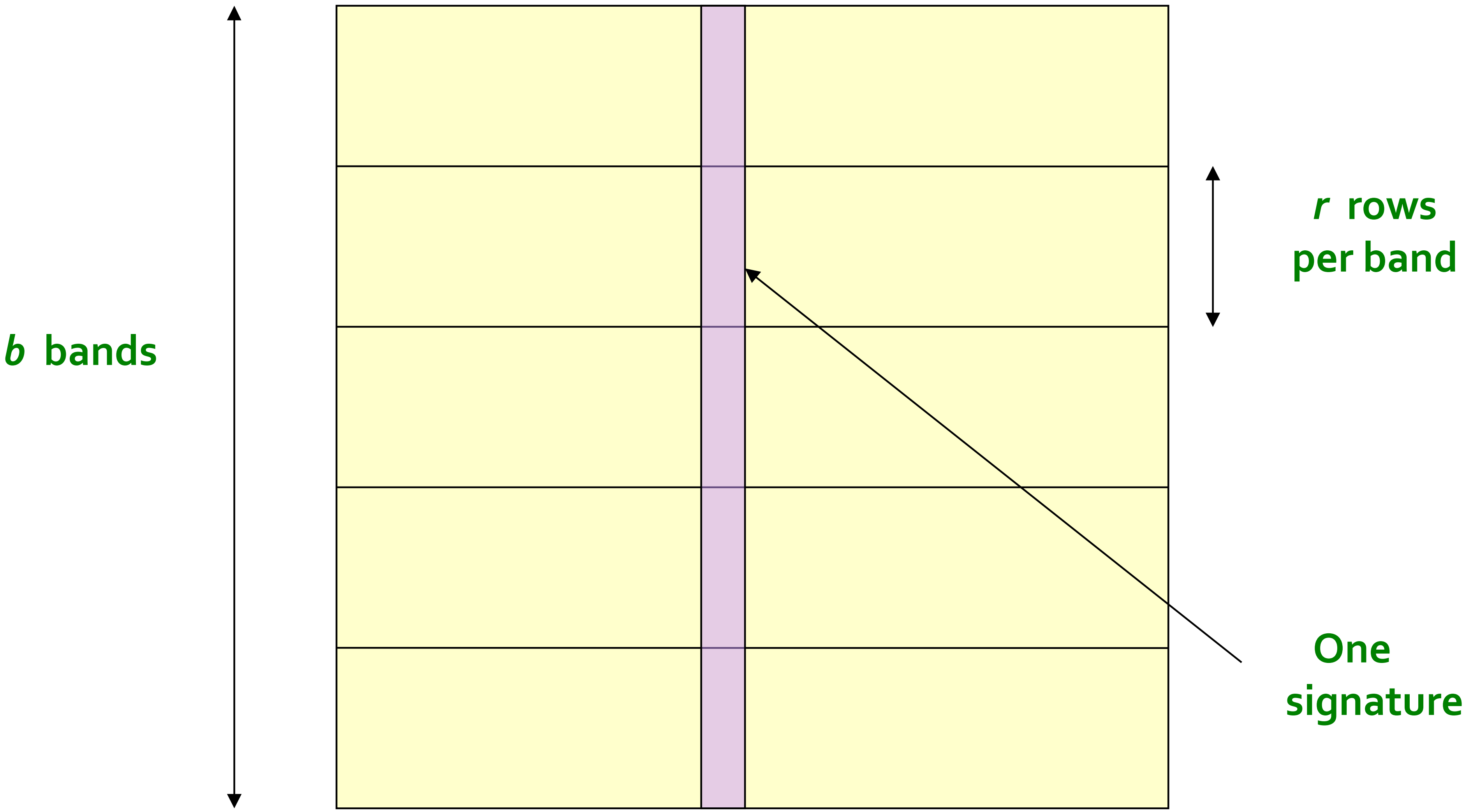
- Arrange that (only) **similar columns** are likely to **hash to the same bucket**, with high probability

- **Candidate pairs are those that hash to the same bucket**

Signature matrix M

2	1	4	1
1	2	1	2
2	1	2	1

Partition M into b bands of size r

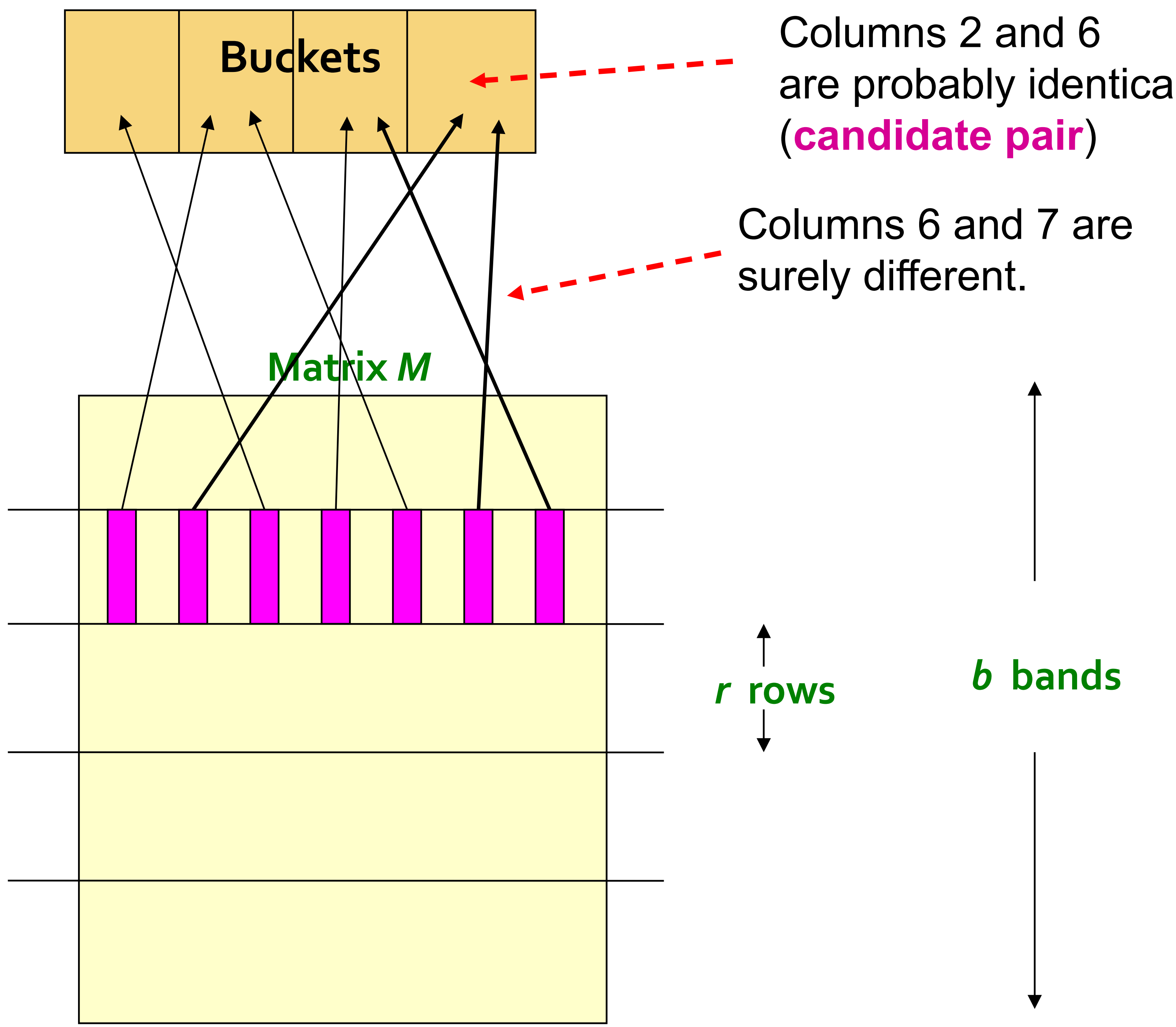


Signature matrix ***M***

Partition M into b bands of size r

- Partition matrix ***M*** into ***b*** bands of ***r*** rows
- For each band, hash its portion of each column to a hash table with ***k*** bucket
  - Make ***k*** as large as possible
- ***Candidate*** column pairs are those that hash to the same bucket for ≥ **1** band
- Tune ***b*** and ***r*** to catch most similar pairs, but few non-similar pairs

Hashing bands



Simplifying assumption: no collisions (no false positives)

- We assume there are **enough buckets** that columns are unlikely to hash to the same bucket unless they are **identical** in a particular band

- Hereafter, we assume that "**same bucket**" means "**identical in that band**"

- Assumption needed only to simplify analysis, not for correctness of algorithm

Example

- **Assume the following case:**
  - Suppose 100,000 columns of ***M*** (100k docs)
  - Signatures of 100 integers (rows) (Therefore, signatures take 40Mb)
  - Choose ***b*** = 20 bands of ***r*** = 5 integers/band

- **Goal:** Find pairs of documents that are at least ***s*** = *0.8* similar

Example: Suppose sim(C_1,C_2) = 0.8

- **Find pairs of** ≥ ***s***=0.8 similarity, set **b**=20, **r**=5
- Since sim(C_1, C_2) ≥ s, we want C_1, C_2 to be a candidate pair:
  - We want them to hash to at least 1 common bucket (at least one band is identical)
- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$
- Probability C_1, C_2 are ***not*** similar in all of the 20 bands: $(1-0.328)^{20} = 0.00035$
  - i.e., about 1/3000th of the 80%-similar column pairs are **false negatives** (we miss them)
  - **We would find 99.965% pairs of truly similar documents**

Example: Suppose sim(C_1,C_2) = 0.3

- **Find pairs of ≥ *s***=0.8 similarity, set **b**=20, **r**=5
- Since sim(C_1, C_2) < s we want C_1, C_2 to hash to NO common buckets (all bands should be different)
- Probability C_1, C_2 identical in one particular band: $(0.3)^5 = 0.00243$
- Probability C_1, C_2 identical in at least 1 of 20 bands: $1 - (1 - 0.00243)^{20} = 0.0474$
- In other words, approximately 4.74% pairs of docs with similarity 0.3% end up becoming **candidate pairs**
- They are **false positives** since we will have to examine them (they are candidate pairs) but then it will turn out their similarity is below threshold **s**

LSH involves a trade-off

- **Pick:**
  - The number of Min-Hashes (rows of *M*)
  - The number of bands ***b***, and
  - The number of rows ***r*** per band to balance false positives/negatives

- **Example:** If we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up