Mining Massive Datasets

Lecture 8

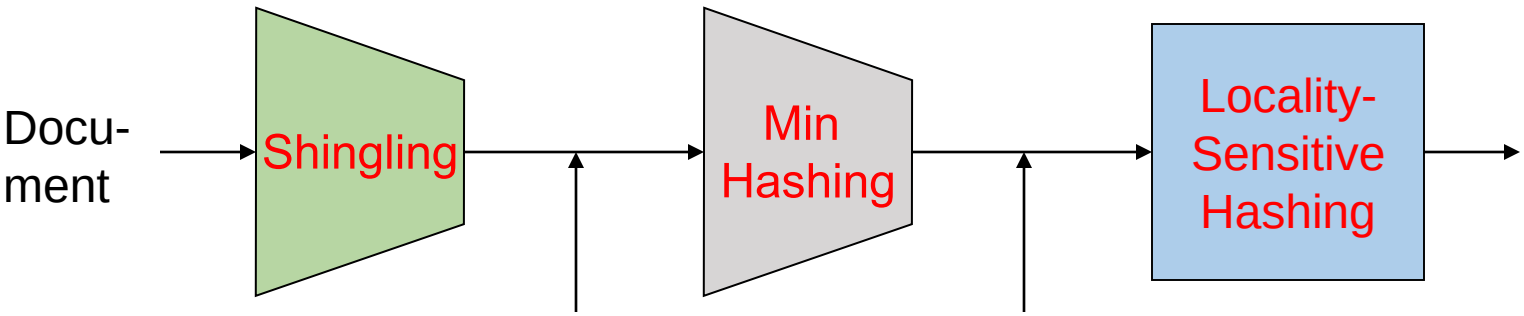
Artur Andrzejak
http://pvs.ifi.uni-heidelberg.de

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Note on Slides

A substantial part of these slides come (either verbatim or in a modified form) from the book Mining of Massive Datasets
by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University).
For more information, see the website accompanying the book: http://www.mmds.org.

Recall



The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Candidate pairs: those pairs of signatures that we need to test for similarity

From Docs to Sets to 0/1-Vectors

- A ***k*-shingle** (or ***k*-gram**) for a document is a sequence of *k* (consecutive) tokens that appears in the doc
- **Example:** k=2; document D_1 = abcab
  - Set of 2-shingles: **S(D_1)** = {ab, bc, ca}
- 1. So now a document $\mathbf{D_1}$ becomes a set of shingles **S(D_1)**

- 2. Represent **S(D_1)** as 0/1-vector (column) **C(D_1)**:

  A. Define for each possible shingle its unique row id, e.g. by sorting

  B. Set 1's in all rows s.t. the corresp. shingle is present in **S(D_1)**

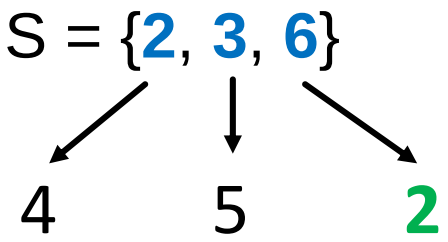
Shingles id = row id	Document's sets		
aaa	1	1	0
aab	1	0	1
aac	1	0	1
…	0	0	1
zzx	0	0	1
zzy	1	1	0
zzz	0	1	0

MinHashing

Repetition and how to compute

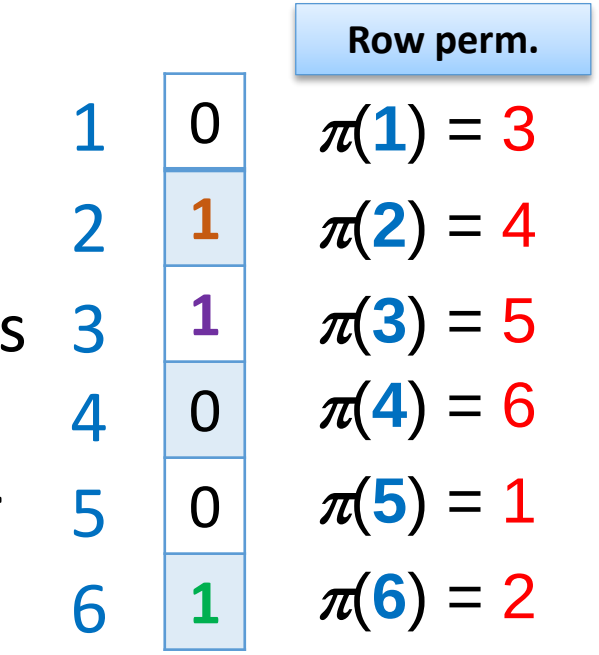
MinHash: Repetition

- Def.: Let **h** be a hash function that maps the members of S to distinct integers, then for any set S define $\mathbf{MinHash_h}(S) = \mathbf{h_{min}}(S)$ to be the minimum value of **h**(x)
  - **Example**:
    - Assume S = {**2**, **3**, **6**} and
    - **h**(**2**) = 4, **h**(**3**) = 5, **h**(**6**) = 2
  - $\mathbf{h_{min}}(S) = 2$

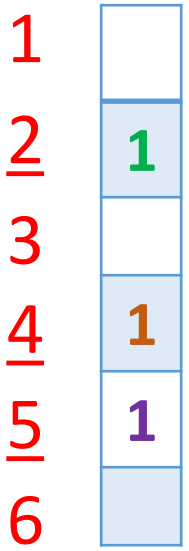


Min-Hashing: Easy Interpretation

- We represent each set as a Boolean vector **C** (here: S = {**2**, **3**, **6**})
- Assume that some hash function **h** is given by a (random) permutation π of the rows of the Boolean vector
  - **h** fulfills: "… maps the members of S to distinct integers"

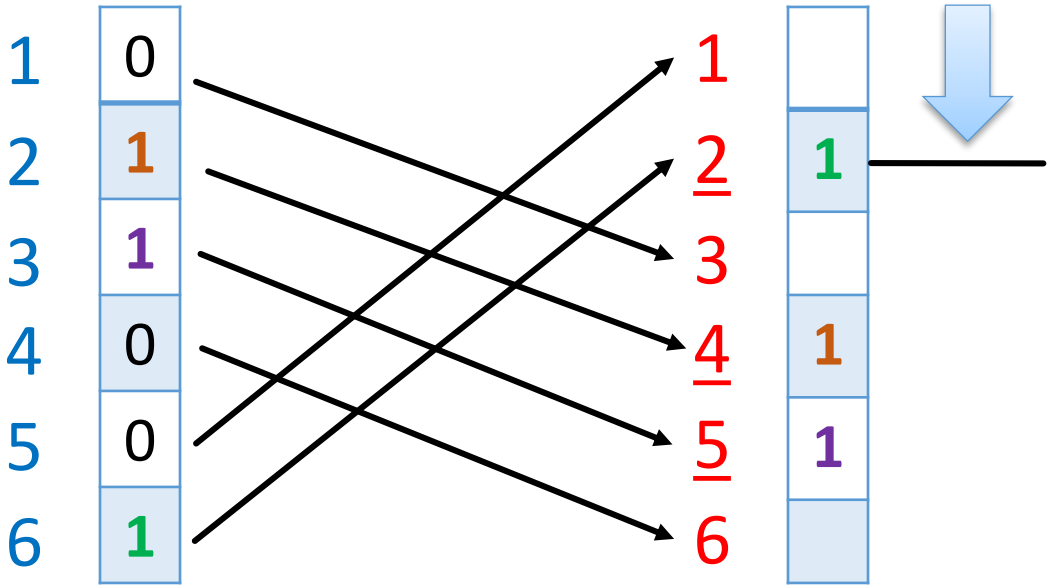


- Then $\mathbf{MinHash_h}(S)$ is the index of the **first row** of the permuted column ***C*** with value **1**
- Depends on **h**; we write $\mathbf{h_{\pi,min}}(S) = 2$



Min-Hashing: Easy Interpretation

- .. **MinHash$_h$**(S) is the index of the **first row** of the <u>permuted</u> column ***C*** with value **1**

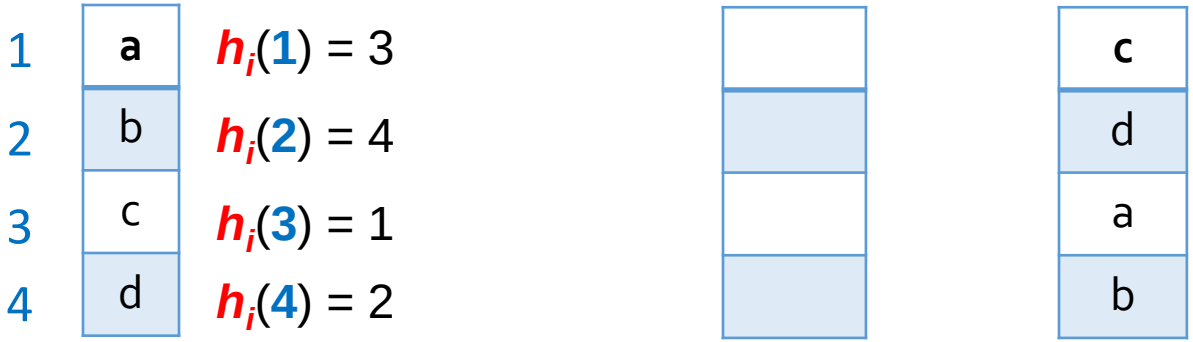


Row perm.
$\pi(1) = 3$
$\pi(2) = 4$
$\pi(3) = 5$
$\pi(4) = 6$
$\pi(5) = 1$
$\pi(6) = 2$

Original C(D_1), i.e. o/1-vector representing set of shingles for document D_1

Implementation /1

- Permuting rows even once is very expensive
- Approximate permutation by a hash function h_i
  - $h_i(x) = [((a \cdot x + b) \bmod p) \bmod N] + 1$
  - a, b: random integers;
    p: a prime ($p > N$); N: #rows in the matrix
  - h_i is possibly not injective, but errors are rare => OK
- Pick about **K = 100** such hash functions h_i



Implementation /2

- Pick about **K = 100** such hash functions h_i

Intuition: for fixed C and h_i, find the smallest value $h_i(r)$ over all rows r with C(r) = 1

<u>One-pass implementation</u>:

- For each column ***C*** and hash-function h_i prepare a "slot" (variable) ***sig(C)[i]*** for the min-hash value
- Initialize all ***sig(C)[i]*** = ∞
- **Scan rows looking for 1s**
  - If row ***q*** has 1 in column ***C***, then for each h_i (i=1..100):
    - If $h_i(q) < sig(C)[i]$, then $sig(C)[i] \leftarrow h_i(q)$

Implementation /3

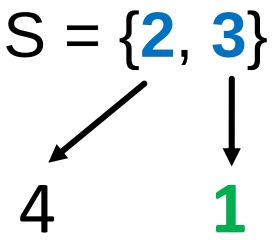
For fixed C and h_i:

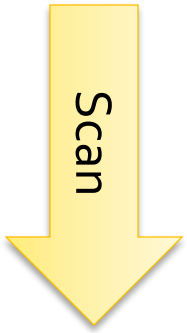
Find the smallest value $h_i(r)$ over all rows r with C(r) = 1

Scan rows looking for 1s

- If row ***q*** has 1 in column ***C***, then for each h_i :
  - If $h_i(q) < sig(C)[i]$, then $sig(C)[i] \leftarrow h_i(q)$

Example: fixed ***C*** and h_i





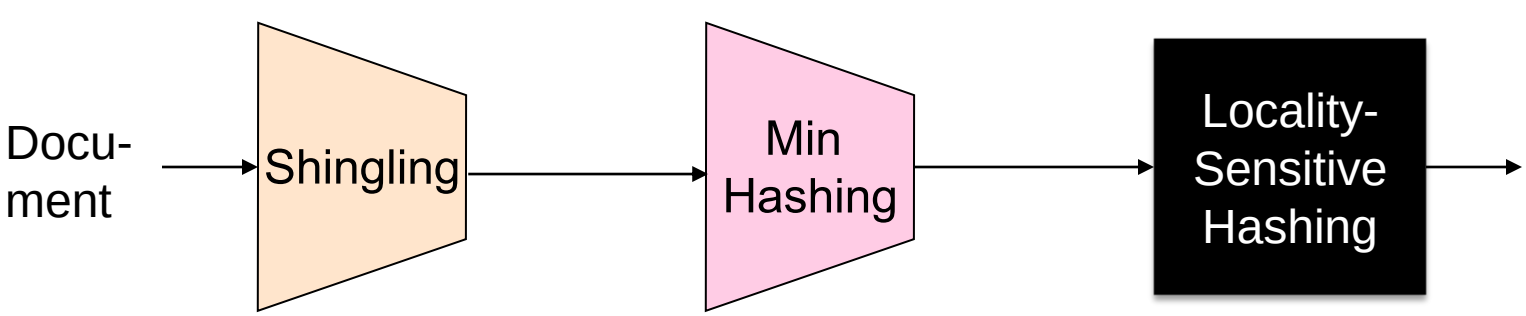
	C	
1	0	$h_i(1) = 3$
2	1	$h_i(2) = 4$
3	1	$h_i(3) = 1$
4	0	$h_i(4) = 2$

sig(C)[i]

sig(C)[i]
∞
4
1
1

Locality Sensitive Hashing

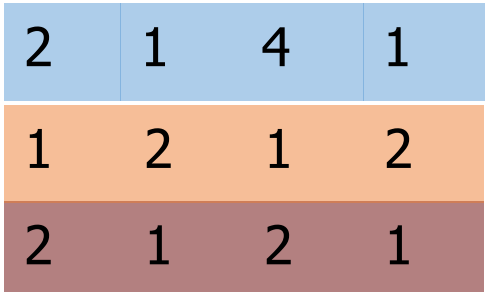
Locality Sensitive Hashing



***Candidate pairs*:**
those pairs of signatures that we need to test for similarity

Step 3: **Locality-Sensitive Hashing**:
Focus on pairs of signatures likely to be from similar documents

LSH: First Cut



- **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

Candidates from Min-Hash

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

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

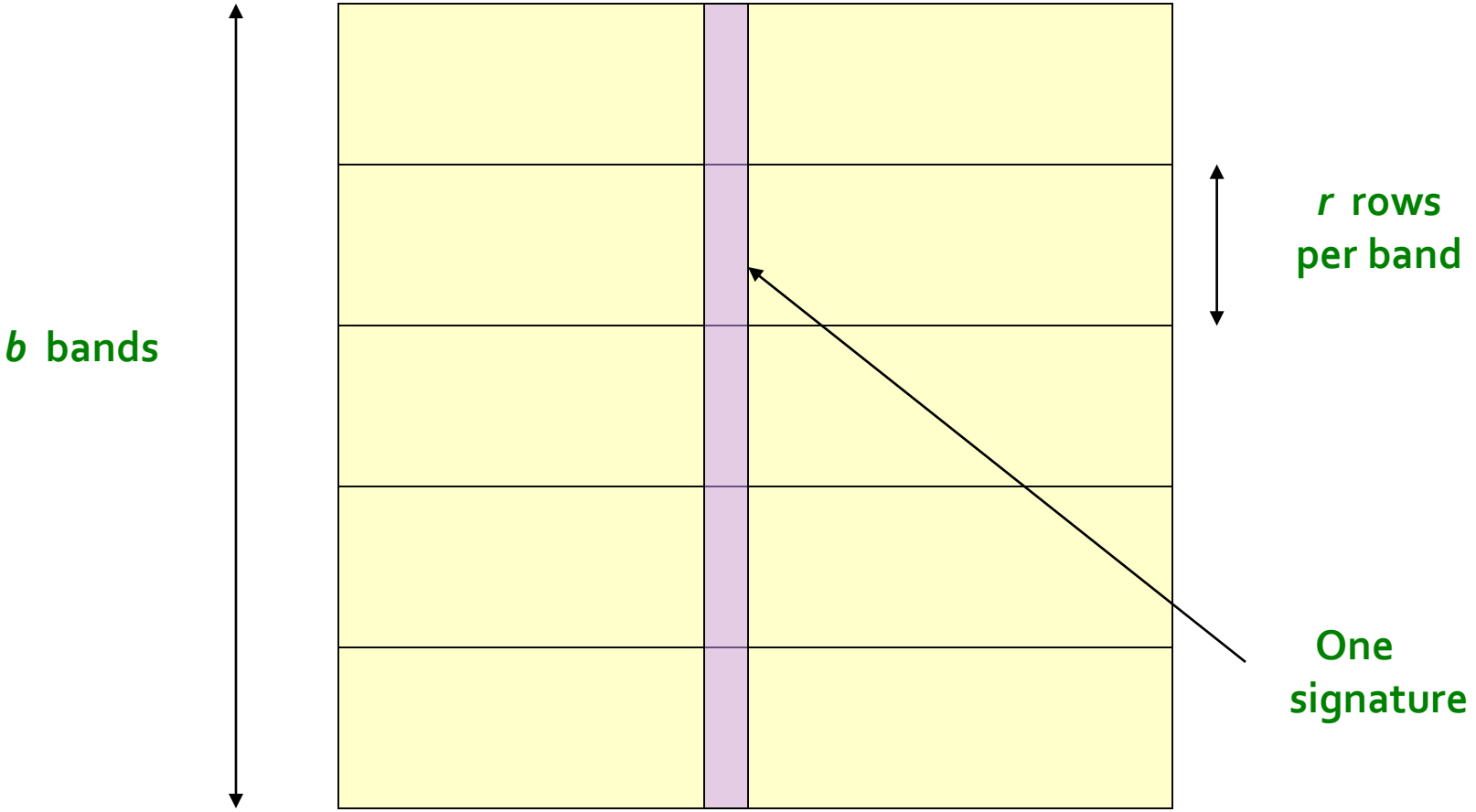
LSH for Min-Hash

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

- **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**

Partition *M* into *b* Bands

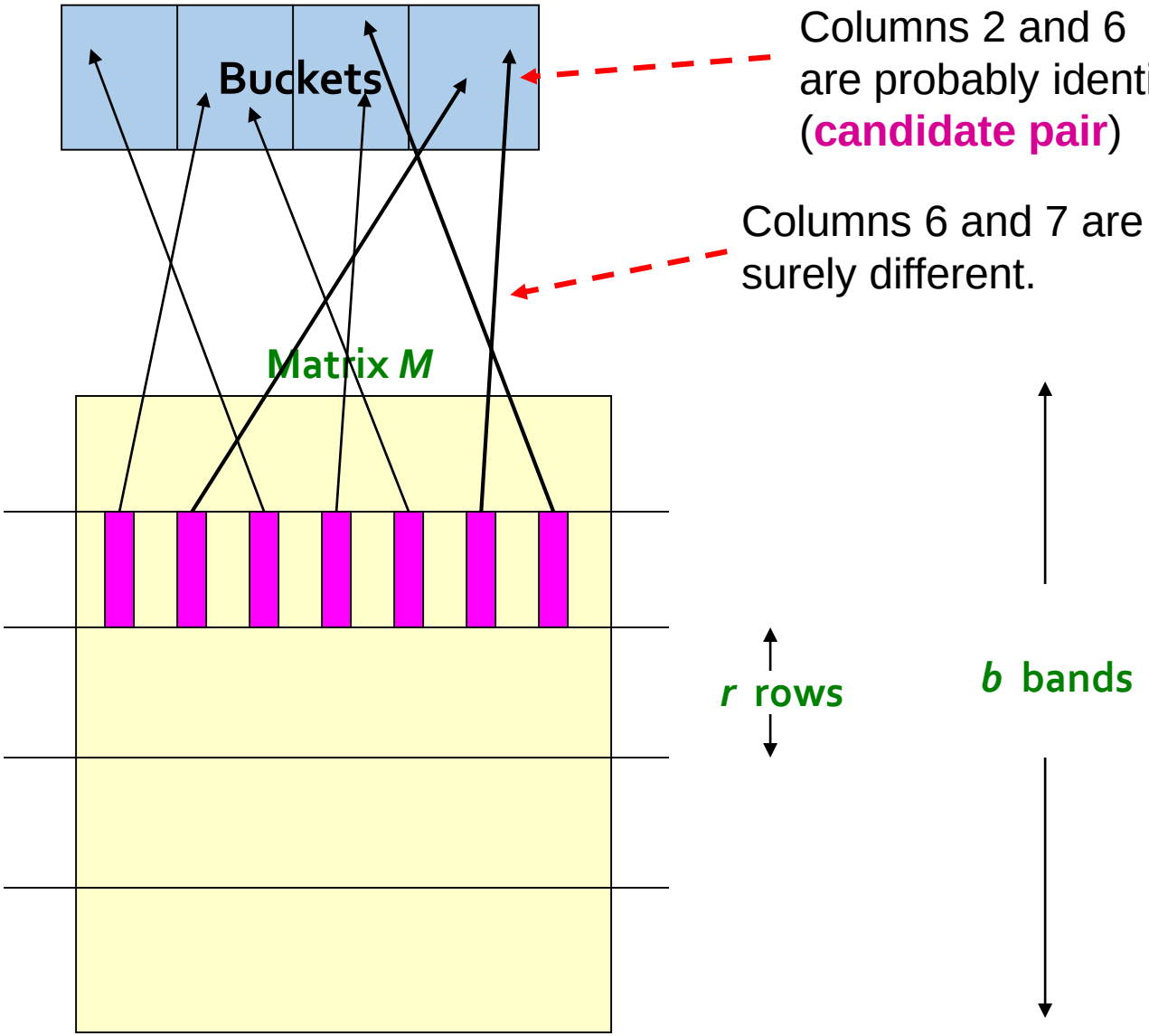
2	1	4	1
1	2	1	2
2	1	2	1



Partition M into Bands

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

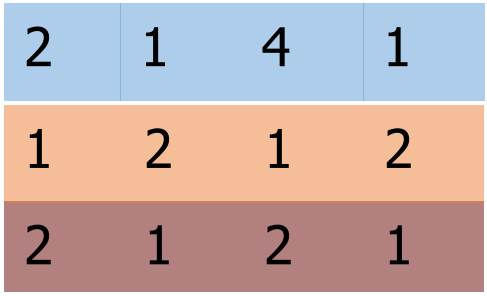
Hashing Bands



Simplifying Assumption

- 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 of Bands



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

C_1, C_2 are 80% Similar

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

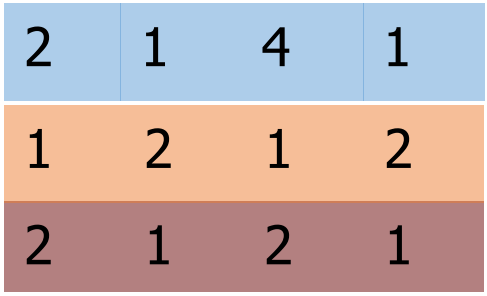
- **Find pairs of $\geq s$=0.8 similarity, set b=20, r=5**
- **Assume:** sim(C_1, C_2) = 0.8
  - Since sim(C_1, C_2) $\geq$ **s**, we want C_1, C_2 to be a **candidate pair**: We want them to hash to at **least 1 common bucket** (at least one band is identical)
- **Probability C_1, C_2 identical in one 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**

C_1, C_2 are 30% Similar

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

- **Find pairs of $\geq s$=0.8 similarity, set b=20, r=5**
- **Assume:** sim(C_1, C_2) = 0.3
  - 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 Tradeoff

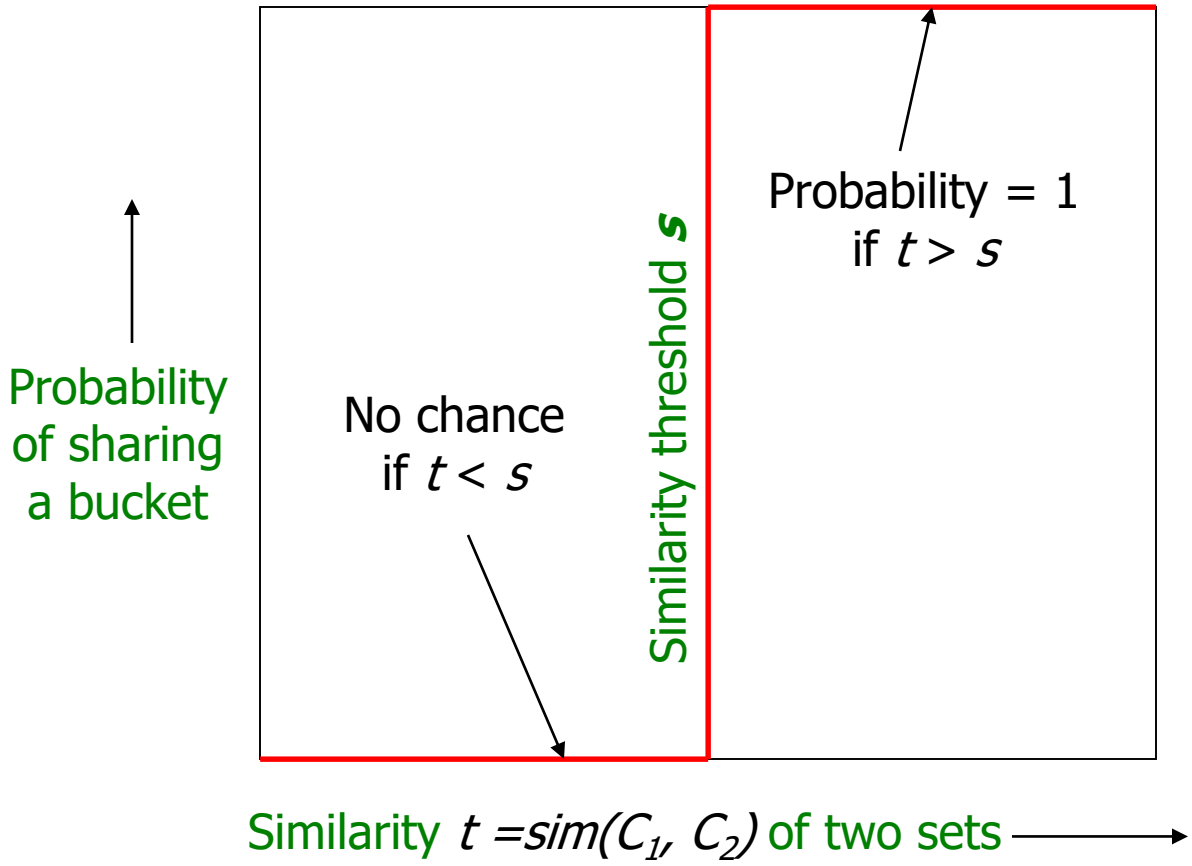


- **Pick:**
  - The number of Min-Hashes (rows of ***M***)
  - The number of bands ***b***, and
  - The number of rows ***r*** per band

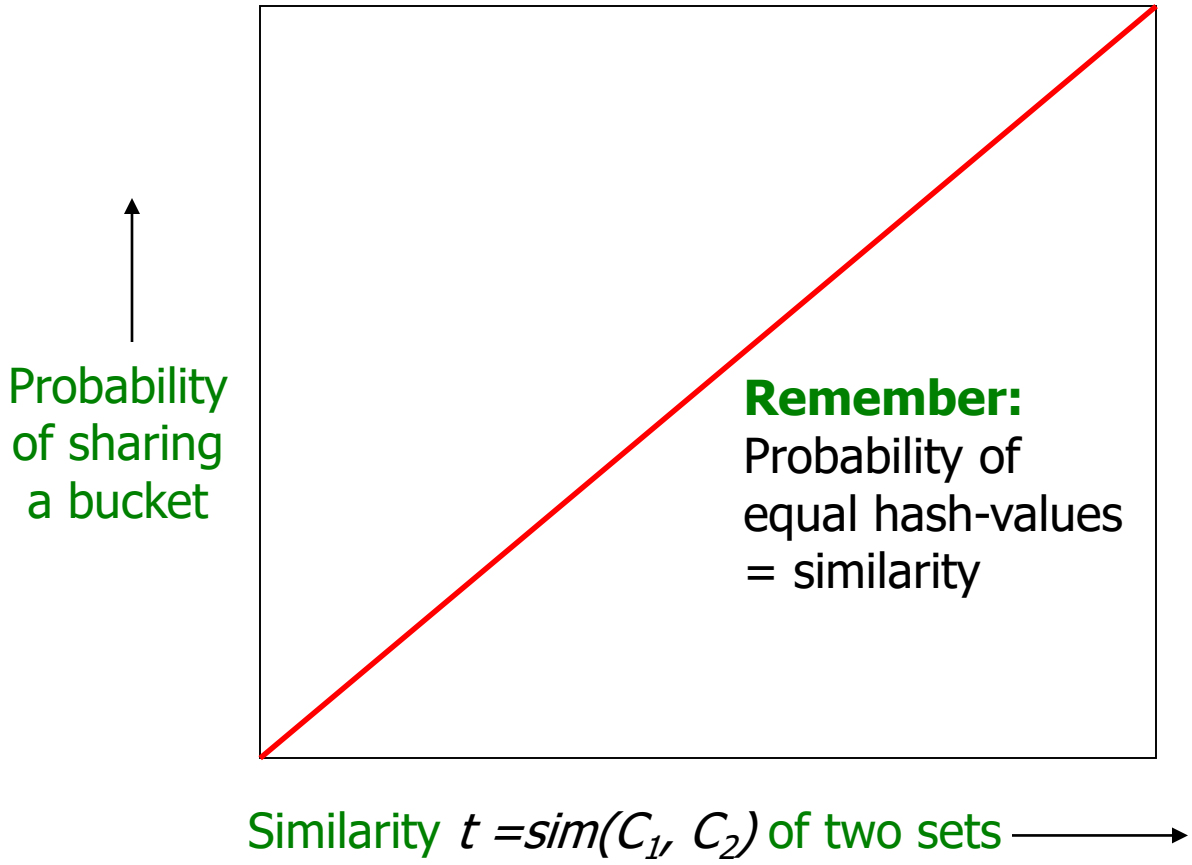
  to balance false positives/negatives

- **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

Analysis of LSH – What We Want



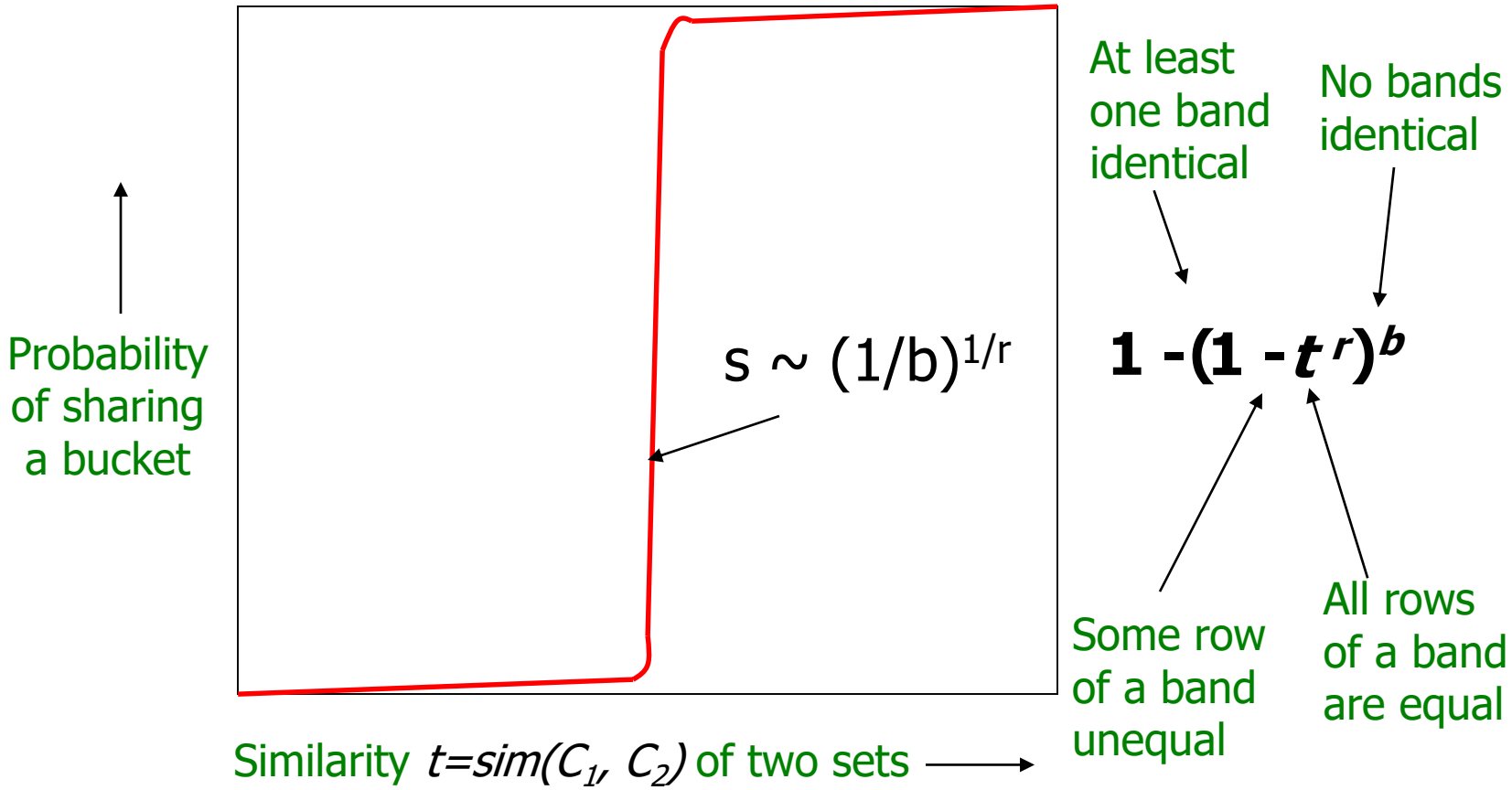
What 1 Band of 1 Row Gives You



b bands, *r* rows/band

- Columns C_1 and C_2 have similarity ***t***
- Pick any band (***r*** rows)
  - Prob. that all rows in band equal = t^r
  - Prob. that some row in band unequal = $1 - t^r$
- Prob. that no band identical = $(1 - t^r)^b$
- Prob. that at least 1 band identical = $1 - (1 - t^r)^b$

What *b* Bands of *r* Rows Gives You



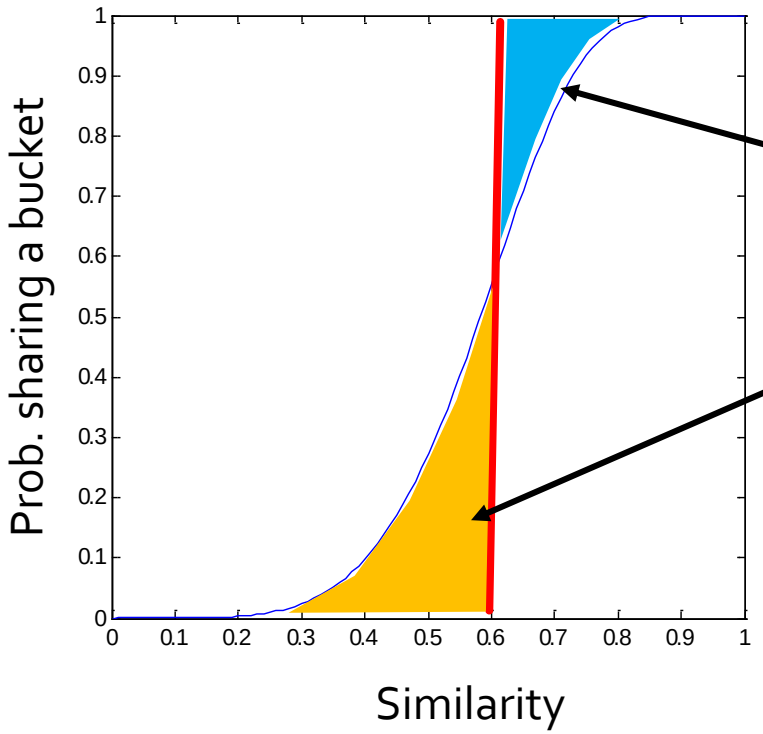
Example: $b = 20$; $r = 5$

- **Similarity threshold s**
- **Prob. that at least 1 band is identical:**

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

Picking *r* and *b*: The S-curve

- **Picking *r* and *b* to get the best S-curve**
  - 50 hash-functions (r=5, b=10)



Blue area: False Negative rate

Orange area: False Positive rate

LSH Summary

- Tune ***M, b, r*** to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures
- Check in main memory that **candidate pairs** really do have **similar signatures**
- **Optional:** In another pass through data, check that the remaining candidate pairs really represent similar documents

Summary: 3 Steps

- **Shingling:** Convert documents to sets
  - We used hashing to assign each shingle an ID
- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
  - We used **similarity preserving hashing** to generate signatures with property $\mathbf{Pr}[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
  - We used hashing to get around generating random permutations
- **Locality-Sensitive Hashing:** Focus on pairs of signatures likely to be from similar documents
  - We used hashing to find **candidate pairs** of similarity $\geq$ **s**

There is more on LSH

- Alternative approaches:
  - Just sample bits of your query / keys
- Applications
  - kNN – k nearest neighbors
  - Computationally efficient “execution” of neural nets
    - For an activation vector of a NN layer, quickly find similar weights (similar to the activations) to the next NN layer

Thank you.

Questions?