MSBD5001 Foundations of Data Analytics

Min-hashing and Locality-sensitive Hashing

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China

Section 1

Finding Similar Documents

Slides for Section 1 to 3 are adopted from:
Chapter 3: Finding Similar Items.
In “Mining Massive Dataset”.
Jure Leskovec, Anand Rajaraman, Jeff Ullman.
Standford University

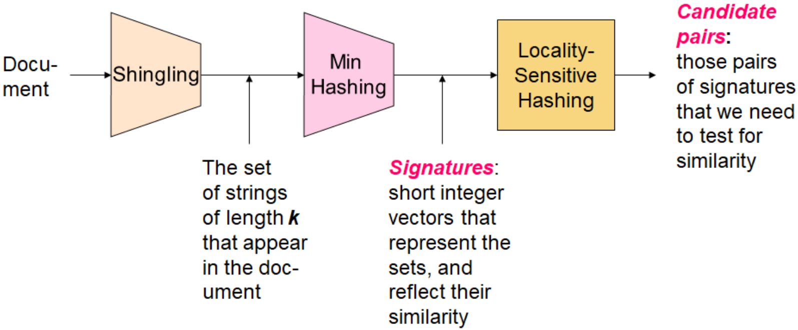
Finding Similar Documents

- Goal: Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs
- Applications:
  - Mirror websites, or approximate mirrors
    - Don't want to show both in search results
  - Similar news articles at many news sites
    - Cluster articles by "same story"
- Problems:
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large or so many that they cannot fit in main memory

Three Essential Steps

1. Shingling: Convert documents to sets
2. Min-Hashing: Convert large sets to short signatures, while preserving similarity
3. Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents
    - Candidate pairs!

The Big Picture



Docu-ment

Shingling

Min Hashing

Locality-Sensitive Hashing

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

Section 2

Shingling: Convert document to sets

Documents as High-Dimensional Data

Step 1: Shingling: Convert documents to sets

- Simple approaches:
  - Document = set of words appearing in document
  - Document = set of "important" words
  - Don't work well for this application. Why?
- Need to account for ordering of words!
- A different way: Shingles!

Define Shingles

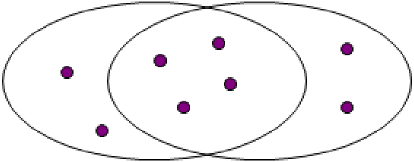
- A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the document
    - Tokens can be characters, words or something else, depending on the application
    - Assume tokens = characters for examples
    - Example: $k = 2$; document $D_1 = abcab$
      Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$
        - ★ Option: Shingles as a bag (multiset), count *ab* twice: $S'(D_1) = \{ab, bc, ca, ab\}$

Compressing Shingles

- To compress long shingles, we can hash them to (say) 4 bytes
- Represent a document by the set of hash values of its k-shingles
  - Idea: Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared
  - Example: $k = 2$; document $D_1 = abcab$
    Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$
    Hash the shingles: $h(D_1) = \{1, 5, 7\}$

Similarity Metric for Shingles

- Document D_1 is a set of its k-shingles $S_1 = S(D_1)$
- Equivalently, each document is a $0/1$ vector in the space of k-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- A natural similarity measure is the Jaccard similarity: $sim(D_1, D_2) = |S_1 \cap S_2| \, / \, |S_1 \cup S_2|$



Working Assumption

- Documents that have lots of shingles in common have similar text, even if the text appears in different order
- Caveat: You must pick k large enough, or most documents will have most shingles
  - $k = 5$ is OK for short documents
  - $k = 10$ is better for long documents

Motivation for Minhashing/Locality-sensitive Hashing

- Suppose we need to find near-duplicate documents among $N = 1$ million documents
- Naively, we would have to compute pairwise jaccard similarities for every pair of documents
  - $N(N-1)/2 \approx 5 * 10^{11}$ comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take 5 days
- For $N = 10$ million, it takes more than a year ...

Section 3

Min-Hashing

Min-Hashing

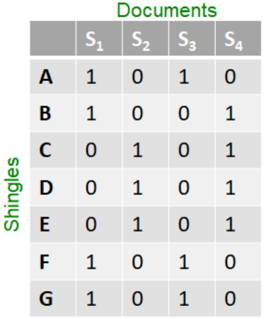
- Step 2: Minhashing: Convert large sets to short signatures, while preserving similarity

Encoding Sets as Bit Vectors

- Many similarity problems can be formalized as finding subsets that have significant intersection
- Encode sets using 0/1 (bit, boolean) vectors
  - One dimension per element in the universal set
  - Interpret set intersection as bitwise AND, and set union as bitwise OR
- Example: $S_1 = 10111$; $S_2 = 10011$
  Size of intersection $= 3$; size of union $= 4$,
  Jaccard similarity (not distance) $= 3/4$
  Distance: $d(S_1, S_2) = 1 - (Jaccard\ similarity) = 1/4$

From Sets to Boolean Matrices

- Rows = elements (shingles)
- Columns = sets (documents)
- 1 in row *e* and column *s* if and only if *e* is a member of *s*
- Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
- Typical matrix is sparse!
- Each document is a column.
- For example, $sim(S_1, S_2) = ?$
  - Size of intersection = 0; size of union = 7, Jaccard similarity (not distance) = 0/7 = 0, Distance: $d(S_1, S_2) = 1 - (Jaccard\ similarity) = 1$
- $sim(S_1, S_3) = ?$
  - Size of intersection = 3; size of union = 4, Jaccard similarity = 3/4 = 0.75, Distance: $d(S_1, S_3) = 1/4$



Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

Outline: Finding Similar Columns

So far:

- Documents → Sets of shingles
- Represent sets as boolean vectors in a matrix
- Next goal: Find similar columns while computing small signatures
- Similarity of columns $==$ similarity of signatures

Outline: Finding Similar Columns

- Next Goal: Find similar columns, Small signatures
- Naive approach:
  1. Signatures of columns: small summaries of columns
  2. Examine pairs of signatures to find similar columns
     - ⋆ Essential: Similarities of signatures and columns are related
  3. Optional: Check that columns with similar signatures are really similar
- Warnings:
  - Comparing all pairs may take too much time: Job for LSH
  - These methods can produce false negatives, and even false positives (if the optional check is not made)
    - ⋆ False negatives: Similar items deemed as non-similar
    - ⋆ False positives: Non-similar items deemed as similar

Hashing Columns (Signatures)

- Key idea: "hash" each column S to a small signature $sig(S)$, such that:
  1. $sig(S)$ is small enough that the signature fits in RAM
  2. $sim(S_1, S_2)$ is almost the same as the "similarity" of signatures $sig(S_1)$ and $sig(S_2)$
- Goal: Find a hash function $h(\cdot)$ such that:
  - If $sim(S_1, S_2)$ is high, then with high probability $sig(S_1) = sig(S_2)$
  - If $sim(S_1, S_2)$ is low, then with high probability $sig(S_1) \neq sig(S_2)$
- Hash documents into buckets. Expect that "most" pairs of near duplicate documents hash into the same bucket!

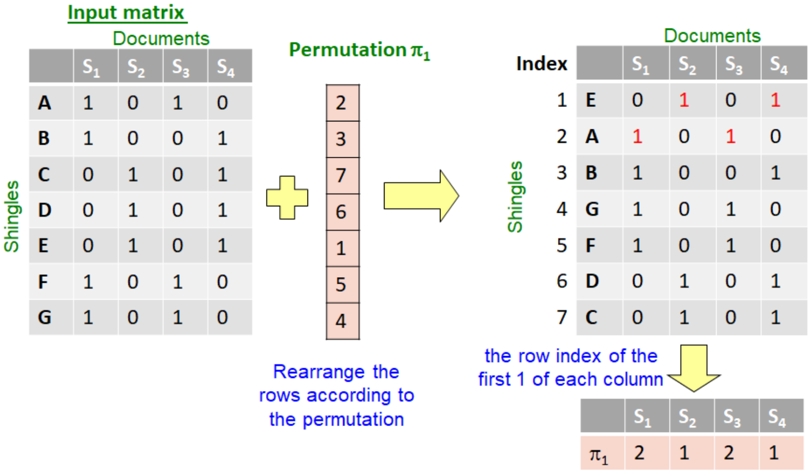
Min-Hashing

- Goal: Find a hash function $h(\cdot)$ such that:
  - If $sim(S_1, S_2)$ is high, then with high probability $sig(S_1) = sig(S_2)$
  - If $sim(S_1, S_2)$ is low, then with high probability $sig(S_1) \neq sig(S_2)$
- Clearly, the hash function depends on the similarity metric:
  - Not all similarity metrics have a suitable hash function
- There is a suitable hash function for the Jaccard similarity: It is called Min-Hashing.

Min-Hashing

- Imagine the rows of the boolean matrix permuted under random permutation π
  - $\pi(C)$ represents the set of indexes where the rows in column C under permutation π has a value 1.
- Define a "hash" function $h_\pi(C) =$ the index of the first (in the permuted order π) row in which column C has value 1: $h_\pi(C) = min_\pi \pi(C)$
- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column

Min-Hashing Example



Input matrix

Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

Permutation π_1

2
3
7
6
1
5
4

Rearrange the rows according to the permutation

Documents

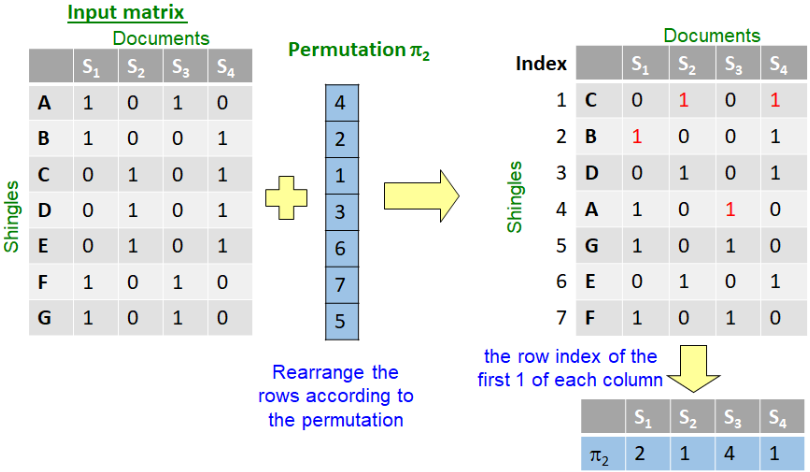
Shingles

Index		S_1	S_2	S_3	S_4
1	**E**	0	1	0	1
2	**A**	1	0	1	0
3	**B**	1	0	0	1
4	**G**	1	0	1	0
5	**F**	1	0	1	0
6	**D**	0	1	0	1
7	**C**	0	1	0	1

the row index of the first 1 of each column

	S_1	S_2	S_3	S_4
π_1	2	1	2	1

Min-Hashing Example



Input matrix

Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

Permutation π_2

4
2
1
3
6
7
5

Rearrange the rows according to the permutation

Documents

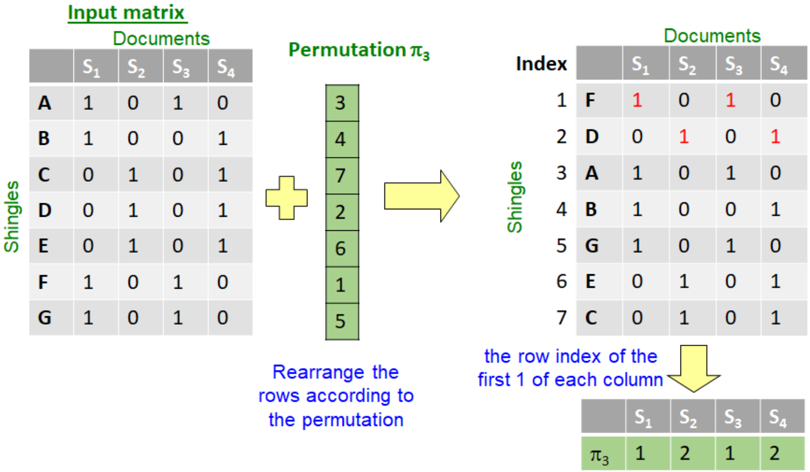
Shingles

Index		S_1	S_2	S_3	S_4
1	**C**	0	1	0	1
2	**B**	1	0	0	1
3	**D**	0	1	0	1
4	**A**	1	0	1	0
5	**G**	1	0	1	0
6	**E**	0	1	0	1
7	**F**	1	0	1	0

the row index of the first 1 of each column

	S_1	S_2	S_3	S_4
π_2	2	1	4	1

Min-Hashing Example



Input matrix

Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

Permutation π_3

3
4
7
2
6
1
5

Rearrange the rows according to the permutation

Documents

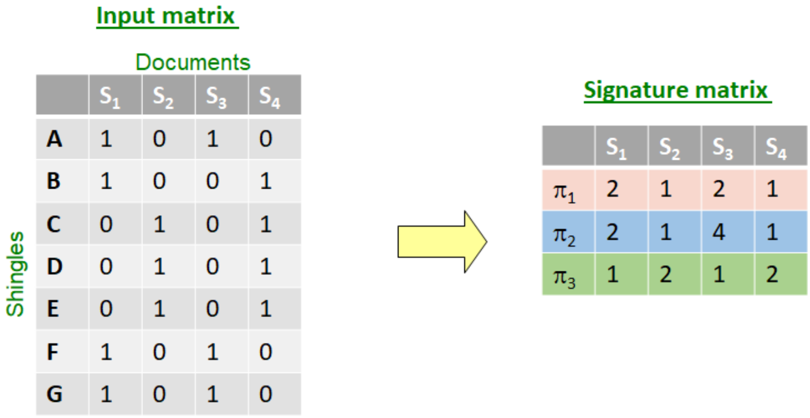
Shingles

Index		S_1	S_2	S_3	S_4
1	**F**	1	0	1	0
2	**D**	0	1	0	1
3	**A**	1	0	1	0
4	**B**	1	0	0	1
5	**G**	1	0	1	0
6	**E**	0	1	0	1
7	**C**	0	1	0	1

the row index of the first 1 of each column

	S_1	S_2	S_3	S_4
π_3	1	2	1	2

Min-Hashing Example



Input matrix

Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0

Signature matrix

	S_1	S_2	S_3	S_4
π_1	2	1	2	1
π_2	2	1	4	1
π_3	1	2	1	2

The Min-Hash Property

- Choose a random permutation π
- Claim: $Pr[h_\pi(S_1) = h_\pi(S_2)] = sim(S_1, S_2)$
- Why?
  - Let X be a document (set of shingles), $y \in X$ is a shingle
  - Then: $Pr[\pi(y) = min(\pi(X))] = 1/\,|X|$
  - It is equally likely that any $y \in X$ is mapped to the min element
  - Let y be s.t. $\pi(y) = min(\pi(S_1 \cup S_2))$
  - Then either:
    $\pi(y) = min(\pi(S_1))$ if $y \in S_1$, or
    $\pi(y) = min(\pi(S_2))$ if $y \in S_2$
    (one of the two columns had to have 1 at position y)
  - So the probability that both are true is the probability $y \in S_1 \cap S_2$
  - $Pr[min(\pi(S_1)) = min(\pi(S_2))] = |S_1 \cap S_2| \,/\, |S_1 \cup S_2| = sim(S_1, S_2)$

Four Types of Rows

- Given cols S_1 and S_2, rows may be classified as:

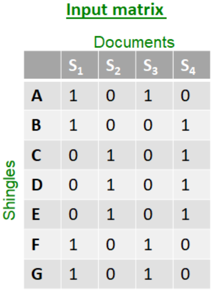
	S_1	S_2
A	1	1
B	1	0
C	0	1
D	0	0

- $a =$ number of rows of type A, etc.
- Note: $sim(S_1, S_2) = a/(a+b+c)$
- Then: $Pr[h(S_1) = h(S_2)] = Sim(S_1, S_2)$
- Look down the cols S_1 and S_2 until we see a 1
- If it's a type-A row, then $h(S_1) = h(S_2)$
  if a type-B or type-C row, then not

Similarity of Signatures

- We know: $Pr[h_\pi(S_1) = h_\pi(S_2)] = sim(S_1, S_2)$
- Now generalize to multiple hash functions
- The similarity of two signatures is the fraction of the hash functions in which they agree
- Note: Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures

Similarity of Signature: Min-Hashing Example

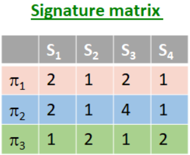


Input matrix

Documents

Shingles

	S_1	S_2	S_3	S_4
A	1	0	1	0
B	1	0	0	1
C	0	1	0	1
D	0	1	0	1
E	0	1	0	1
F	1	0	1	0
G	1	0	1	0



Signature matrix

	S_1	S_2	S_3	S_4
π_1	2	1	2	1
π_2	2	1	4	1
π_3	1	2	1	2

	Actual (Input Matrix)	Signature Matrix
(S_1, S_2)	0	0
(S_1, S_3)	$3/4$	$2/3$
(S_1, S_4)	$1/7$	0
(S_2, S_3)	0	0
(S_2, S_4)	$3/4$	1
(S_3, S_4)	0	0

Min-Hashing Signatures

- Pick $K = 100$ random permutations of the rows
- Think of $sig(S)$ as a column vector
- $sig(i, S) =$ according to the i-th permutation, the index of the first row that has a 1 in column S
  $sig(i, S) = min(\pi(S))$
- Note: The sketch (signature) of document S is small ~ 100 bytes!
- We achieved our goal! We "compressed" long bit vectors into short signatures

Feasibility

- Assume a billion rows
- Hard to pick a random permutation of 1 billion
- Even representing a random permutation requires 1 billion entries!!!
- How about accessing rows in permutated order?
- i.e. It is not feasible!

Implementation Trick

- Approximating row permutation
  - Row hashing!
    - Pick $K = 100$ hash functions h_i
    - Ordering under h_i gives a random row permutation!

One-pass implementation

- Pick k hash functions $h_1, h_2, ..., h_k$
- Let $Sig(i, S_j)$ be the element of the signature matrix for the i-th hash function and document S_j.
- Initialize the signature matrix by setting all $Sig(i, S_j) = \infty$
-

```
For each row $r$
    Compute $h_1(r), h_2(r), ..., h_k(r)$
    For each column $S_j$
        If it has 1 in row $r$
            For each hash function $h_i$
                If $h_i(r)$ is smaller than $Sig(i, S_j)$ then
                    $Sig(i, S_j) = h_i(r)$
```

- Note: How to pick a random hash function $h(x)$?
  - Assume there are N rows
  - $h_{a,b}(x) = ((a * x + b) mod\ p) mod\ N$
  - where a, b are random integers,
  - and p is a prime number ($p > N$)

Section 4

Locality-Sensitive Hashing

Locality-Sensitive Hashing

- Problem: Find all pairs of documents with similarity at least t, e.g. $t = 0.8$
- While the signatures of all columns may fit in main memory, comparing the signatures of all pairs of columns is quadratic in the number of columns.
- Solution: Focus on the pairs of signatures that are likely to be from similar documents

Locality-Sensitive Hashing

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

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

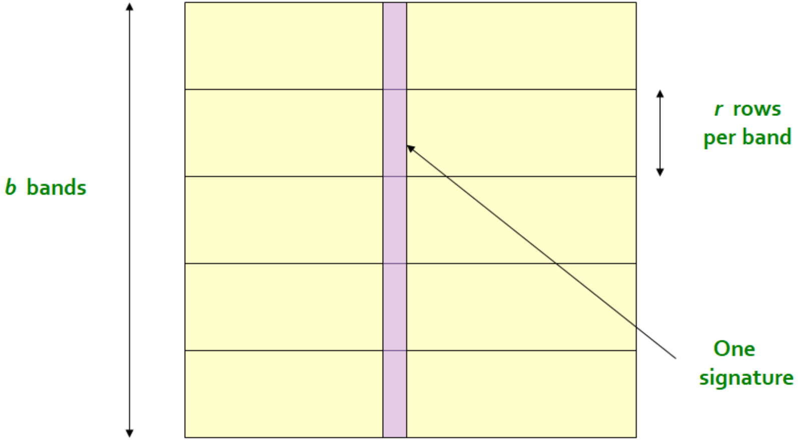
LSH for Min-Hashing

- 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 bands

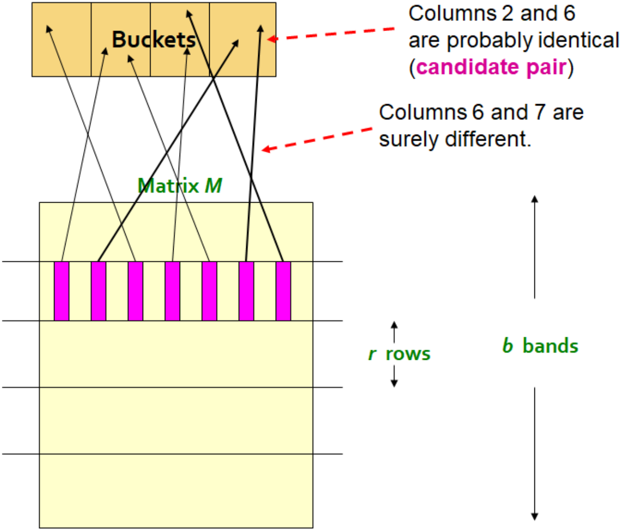
- Divide matrix M into b bands of r rows
- For each band, hash its portion of each column to a hash table with k buckets
- Make k as large as possible
- Candidate column pairs are those that hash to the same bucket for ≤ 1 band
- Tune b and r to catch most similar pairs, but few non-similar pairs

Partition M into bands



Signature matrix *M*

Hashing Bands



Columns 2 and 6 are probably identical (**candidate pair**)

Columns 6 and 7 are surely different.

LSH Analysis

- 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
  - Assume the following case:
    - Suppose 100,000 columns of M (100k documentss)
    - 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 $t = 0.8$ similar

LSH Analysis – Example

Case 1: Documents S_1 and S_2 are 80% similar

- Find pairs of documents with similarity $\geq t$ where $t = 0.8$, set $b = 20$, $r = 5$
- Assume: $sim(S_1, S_2) = 0.8$
- Since $sim(S_1, S_2) \geq t$, we want S_1 and S_2 to be a candidate pair:
  - We want them to hash to at least 1 common bucket (at least one band is identical)
- Probability that S_1 and S_2 are identical in one particular band: $(0.8)^5 = 0.328$
- Probability that S_1 and S_2 are not similar in all of the 20 bands: $(1 - 0.328)^{20} = 0.00035$
- i.e., about $1/3000$th of the 80%-similar column pairs are false negatives (we miss them)
- We would find 99.965% pairs of truly similar documents

LSH Analysis – Example

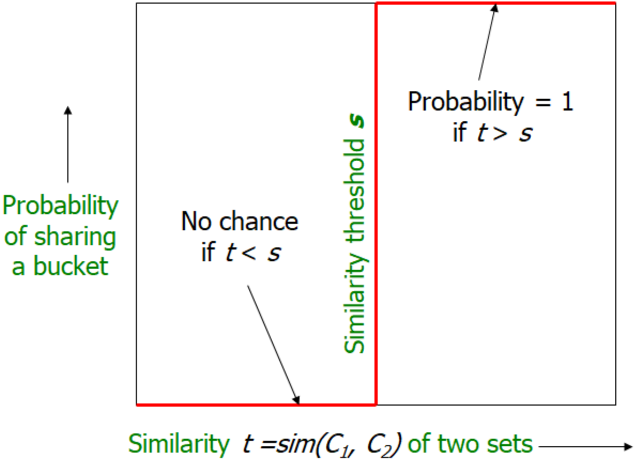
Case 2: Documents S_1 and S_2 are 30% similar

- Find pairs of documents with similarity $\geq t$ where $t = 0.8$, set $b = 20$, $r = 5$
- Assume: $sim(S_1, S_2) = 0.3$
- Since $sim(S_1, S_2) < t$,
    - We want them to hash to NO common bucket (all bands should be different)
- Probability that S_1 and S_2 are identical in one particular band: $(0.3)^5 = 0.00243$
- Probability that S_1 and S_2 are identical in at least 1 of the 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 t

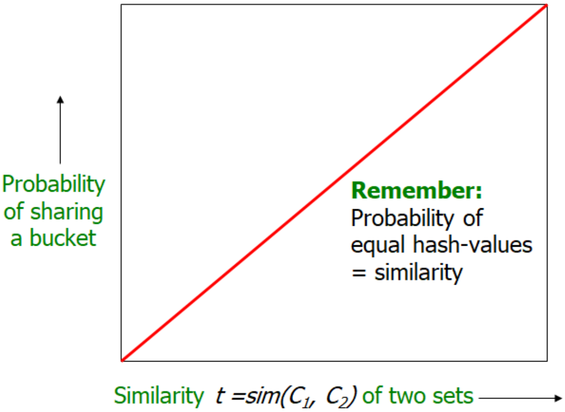
LSH – a Trade-off

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

LSH Analysis – What we want



1 Band of 1 Row



Probability of sharing a bucket

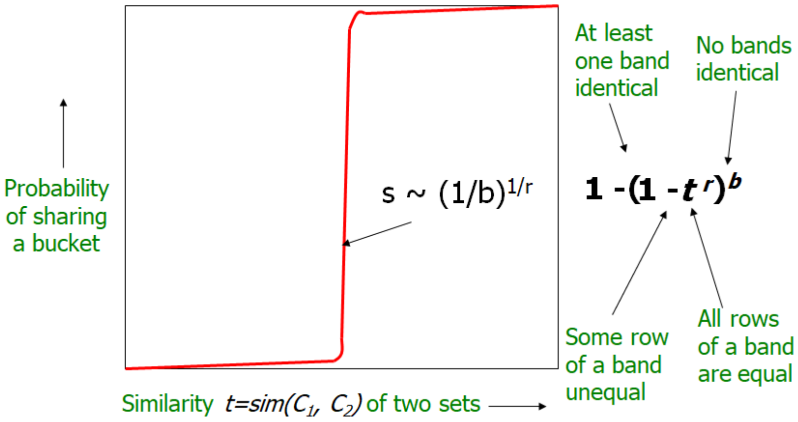
Remember: Probability of equal hash-values = similarity

Similarity $t = sim(C_1, C_2)$ of two sets

b bands, r rows per band

- Columns S_1 and S_2 have similarity s
- Pick any band (r rows)
  - Probability that all rows in band equal $= s^r$
  - Probability that some row in band unequal $= 1 - s^r$
- Probability that no band identical $= (1 - s^r)^b$
- Probability that at least 1 band identical $= 1 - (1 - s^r)^b$

b Band of r Rows



Probability of sharing a bucket

$s \sim (1/b)^{1/r}$

Similarity $t = sim(C_1, C_2)$ of two sets

At least one band identical

No bands identical

$$\mathbf{1 - (1 - t^r)^b}$$

Some row of a band unequal

All rows of a band are equal

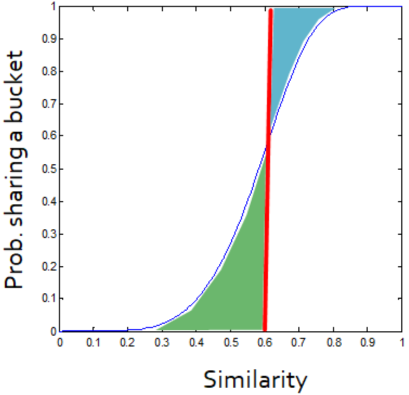
Example: $b = 20$, $r = 5$

- Similarity threshold t
- Probability that at least 1 band is identical

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

Picking r and b: The S-curve

For example: 50 hash-functions (r=5, b=10)



Blue area: False Negative rate

Green area: False Positive rate

References

- Chapter 3: Finding Similar Items.
  In “Mining Massive Dataset”.
  Jure Leskovec, Anand Rajaraman, Je Ullman.
  Standford University