universidade de aveiro

Mining Large Scale Datasets

Finding Similar Items – Locality-Sensitive Hashing

(Adapted from CS246@Starford.edu; http://www.mmds.org)

Sérgio Matos - aleixomatos@ua.pt

Motivation

- Many problems can be expressed as finding "similar" sets

  $\hookrightarrow$ Find near-neighbors in high-dimensional space

- Some examples:
  - Documents with similar content

    Mirror pages, plagiarism
  - E-commerce

    'Similar' products; 'similar' costumers
  - Recommendation and search

    Netflix movies
  - Entity resolution

    FB and LinkedIn profiles

Motivation

- Many problems can be expressed as finding "similar" sets
  $\hookrightarrow$ Find near-neighbors in high-dimensional space

- Some examples:
  - Documents with similar content
    Mirror pages, plagiarism
  - E-commerce
    'Similar' products; 'similar' costumers
  - Recommendation and search
    Netflix movies
  - Entity resolution
    FB and LinkedIn profiles

Note: NOT the same as finding exactly equal items

Motivation

- Given:

  High dimensional data points $x_1, x_2, \ldots$

  Example: images are long vectors of pixel colors

  Some distance function $d(x_1, x_2)$

- Find all pairs of data points that are within a distance threshold

$$d(x_1, x_2) \leq t$$

Naïve approach is $O(N^2)$

Naïve approach requires looking at every pair of items.
Even a "small" dataset of a million items gives half a trillion pairs to examine.

Motivation

- Suppose we need to find near-duplicate documents among N = 1 million documents
- If we compute pairwise similarities for every pair of docs

  $N(N-1)/2 \approx 5 \times 10^{11}$ comparisons

  At 10^6 comparisons/sec, it would take >**5 days**

- For N = 10 million, it would take more than a year!

Motivation

- Suppose we need to find near-duplicate documents among N = 1 million documents
- If we compute pairwise similarities for every pair of docs

  $N(N-1)/2 \approx 5 \times 10^{11}$ comparisons

  At 10^6 comparisons/sec, it would take > **5 days**

- For N = 10 million, it would take more than a year!

$\hookrightarrow$ Can be done in $O(N)$ ☺

Locality-Sensitive Hashing

- Family of related techniques
- Allows to only examine pairs that are likely to be similar

  Avoids quadratic growth in computation time

LSH: general idea

- Hash items into buckets using many different hash functions

  ↪ Functions are designed so that similar items are more likely to hash into same bucket

- Only pairs that share a bucket for at least one of the hash functions need to be examined

  ↪ There may be false negatives – pairs of similar items may not be considered at all

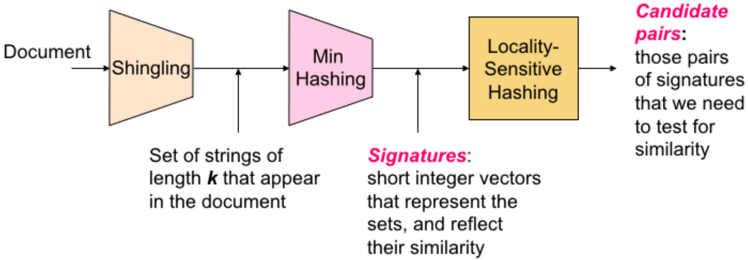
  ↪ There may be false positives – pairs of items may be erroneously found as similar

Application example: similar documents

- Find documents that share a lot of common text
  - Mirror pages
  - Plagiarism
  - Similar news articles

- Transform documents into sets
- Convert sets into smaller *signatures*
- Compare signatures using Jaccard similarity

Similar Documents: Steps



- **Shingling**: Converts a document into a set representation
- **Min-Hashing**: Convert large sets to short signatures, while preserving similarity
- **Locality-Sensitive Hashing**: Focus on pairs of signatures likely to be from similar documents

Shingling: Convert documents into sets

- A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the doc

  Tokens may be chars, words or something else, depending on the application
- k is application dependent, but should be large enough so that most shingles do not appear in a given document

  $k = 8$, 9, or 10 is often used in practice

  first char 9-shingles of slide title:

  “shingling”, “hingling:”, “ingling: ”, “ngling: c”, “gling: co”
- Long shingles can be compressed through hashing

  Use 4 byte integers for example, instead of 9 bytes
- A document is represented by the set of (hash values of) its k-shingles

Shingling: Convert documents into sets

Example: document D1="abcdabd"

Set of 2-shingles: S(D1) = ab, bc, cd, da, bd
Hash the shingles: h(S(D1)) = 1, 5, 7, 8, 11

Benefits of shingles:

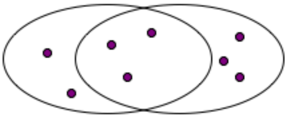
- Similar documents will have many shingles in common
- Changing a word only affects k-shingles within distance $k-1$ from the word
- Reordering paragraphs only affects the shingles that cross paragraph boundaries

Comparing sets: Jaccard similarity

Document D1 is a set of its k-shingles C1=S(D1)

A natural similarity measure is the Jaccard similarity:

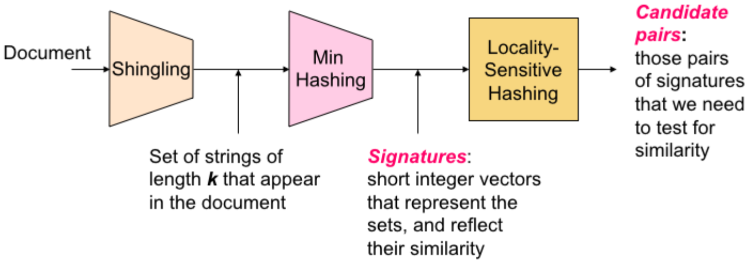
$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



intersection = 3
union = 8
Jaccard similarity = 3/8

Jaccard distance = $1 - sim(D_1, D_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$

Similar Documents: Steps



- Shingling: Converts a document into a set representation
- **Min-Hashing: Convert large sets to short signatures, while preserving similarity**
- Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents

Summarizing sets

- Shingling:

  Documents as sets of shingles

  Represented as boolean/bit vectors in a matrix

- Sets of shingles are large (possibly 4x document size)

  Compute small *signatures*, so that

  Similarity of signatures $\approx$ similarity of documents

$\hookrightarrow$ Remember: Comparing all pairs takes too much time
We will see how to handle this later (LSH)

Summarizing sets: signatures

Key idea

"Hash" each column C to a small signature $h(C)$, such that:
$sim(C1, C2)$ is the same as the "similarity" of $h(C1)$ and $h(C2)$

Find a hash function $h(\cdot)$ such that:

if $sim(C1, C2)$ is high, then with high prob. $h(C1) = h(C2)$

if $sim(C1, C2)$ is low, then with high prob. $h(C1) \neq h(C2)$

Summarizing sets: signatures

Key idea

"Hash" each column C to a small signature $h(C)$, such that:
$sim(C1, C2)$ is the same as the "similarity" of $h(C1)$ and $h(C2)$

Find a hash function $h(\cdot)$ such that:

if $sim(C1, C2)$ is high, then with high prob. $h(C1) = h(C2)$

if $sim(C1, C2)$ is low, then with high prob. $h(C1) \neq h(C2)$

The hash function depends on the similarity metric:

Not all similarity metrics have a suitable hash function

Suitable hash function for the Jaccard similarity: **Min-Hashing**

Characteristic matrix

Encode the collection of sets using bit vectors

- 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

  What is $sim(C_1, C_2)$?

Documents

Shingles

1	1	1	0
1	1	0	1
0	1	0	1
0	0	0	1
1	0	0	1
1	1	1	0
1	0	1	0

Matrix is usually sparse!
$\hookrightarrow$ Not actually constructed; only used to help 'visualize'

Min-Hashing

Definition

To minhash a set represented by a column of the characteristic matrix, perform a random permutation of the rows.

The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1.

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

Create a signature of length 3, using three random permutations

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

	1		1

For the first permutation, select the row with index 1, and assign that number as the signature value to columns with 1 in the characteristic matrix

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1

Next, select row with index 2, and assign that number as the signature value to columns with 1 in the characteristic matrix

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1
	1		1

Now for the second permutation, select the row with index 1, and assign that number as the signature value to columns with 1 in the characteristic matrix

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1
2	1		1

Repeat for row with index 2. Do not change signature value for C4, since the value is already set (and is lower)

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1
2	1		1

Repeat for row with index 3. In this case we don't do any changes since C1 and C4 already have signature values

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1
2	1	4	1

Repeat for row with index 4

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

2	1	2	1
2	1	4	1

Your turn!

Min-Hashing

Each minhash function $h_\pi(\cdot)$ is associated with a (*virtual*) permutation of the rows of the characteristic matrix

$h_\pi(C) =$ the number of the first row (in the permuted order) in which column C has value 1

$h_\pi(C) = min_\pi \pi(C)$

Apply, to all columns, several randomly chosen permutations to create a signature for each column

Result is a signature matrix

columns = sets, rows = minhash values

Min-Hashing and Jaccard similarity

The Min-Hash Property

For a random permutation of rows, the probability that $h(C_1) = h(C_2)$ equals $sim(C_1, C_2)$, the Jaccard similarity of those sets.

$$p\Big(h(C_1) = h(C_2)\Big) = sim(C_1, C_2)$$

Min-Hashing and Jaccard similarity

C_1	C_2	
1	1	a
1	0	b
0	1	c
0	0	d

- Characteristic matrix has rows of types a, b, c, d
- Jaccard similarity between C_1 and C_2

$$sim(C_1, C_2) = a/(a + b + c)$$

(here, a represents the number of lines of type a)

Min-Hashing and Jaccard similarity

Consider the permutations

C_1	C_2
0	0
0	0
0	0
0	0
1	1

...

C_1	C_2
0	0
0	0
0	0
0	0
1	0

...

C_1	C_2
0	0
0	0
0	0
0	0
0	1

...

First case corresponds to finding an a type row, and $h(C_1) = h(C_2)$
Other cases correspond to b and c type rows, and $h(C_1) \neq h(C_2)$

So, the probability of $h(C_1) = h(C_2)$ is equal to the probability of finding an a type row first, or $a/(a+b+c)$

Similarity of signatures

- The similarity of two signatures is the fraction of the hash functions in which they agree
- Thus, the expected similarity of two signatures equals the Jaccard similarity of the columns or sets that the signatures represent
- The longer the signatures, the smaller will be the expected error

Min-Hashing: example

Permutations

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Characteristic matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix

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

Similarities

1-2	1-3	2-4	3-4	
0	.75	.75	0	Original
0	.67	1	0	Signatures

Min-Hashing: Implementation

- Permuting rows is **prohibitive!!**
- Consider 1 billion rows
  - Picking a random permutation of 1 billion is highly expensive
  - Representing a random permutation requires 1 billion entries (4 Gb!)
  - Accessing rows in permuted order represents too many disks accesses (thrashing)

Min-Hashing: Implementation

- Solution: row hashing

  Pick K hash functions h_i (e.g. $K = 100$)

  h_i "permutes" r to position $h_i(r)$ in the permuted order

  For each column c and hash function h_i, the signature value will be given by the smallest value of $h_i(r)$ for which column c has a 1 in row r

- Which hash function to use?

  Universal hashing:

  $h_{a,b}(x) = ((a \cdot x + b) \mathrm{mod} p) \mathrm{mod} N$

  a, b: random integers, $a \neq 0$

  p: prime number, $p > N$

  N: number of shingles

Min-Hashing: Implementation

```
initialize M[i, c]  // matrix of signature values
                    // for each hash function i
                    // and column c
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]
                    M(i,c) = h_i(r)
```

Min-Hashing: Implementation

Row	C_1	C_2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

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

	$M(i, C_1)$	$M(i, C_2)$
$h(1) = 1$	1	∞
$g(1) = 3$	3	∞
$h(2) = 2$	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

Signature matrix *M*

Min-Hashing: Speedup

- Apply only to first m rows

  Some columns may have only zeros in all m initial rows

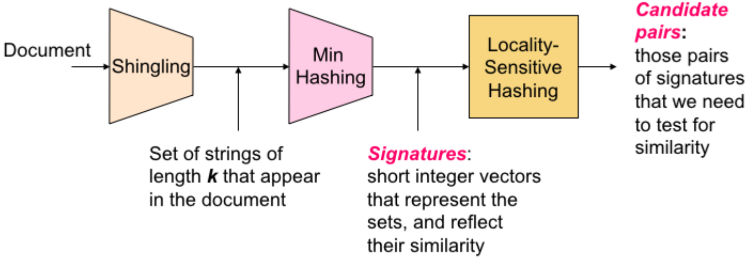
- Divide matrix M into k/m blocks

  Apply minhashing to each block

  Gives k/m minhash values from a single hash function and a single pass over all the rows of M

  Allows using less hash functions

Similar Documents: Steps



- Shingling: Converts a document into a set representation
- Min-Hashing: Convert large sets to short signatures, while preserving similarity
- **Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents**

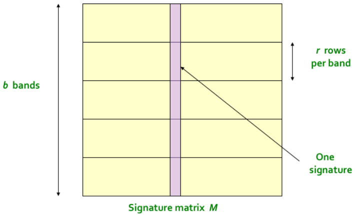
Locality-Sensitive Hashing

- Goal: Find documents with Jaccard similarity at least s_{thresh}
- We what columns C_1 and C_2 of M to be a candidate pair if $M(i, C_1) = M(i, C_2)$ <u>for at least fraction s_{thresh} of rows i</u>

For Min-Hash matrices

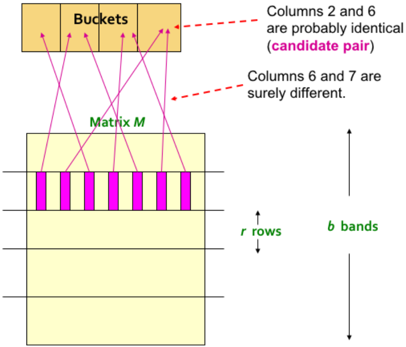
Hash columns of signature matrix M to many buckets
Each pair of documents that hashes into the same bucket is a candidate pair

Locality-Sensitive Hashing



- Divide M into b bands of r rows
- For each band, hash its portion of each column into k buckets
  - Make k as large as possible
- Column pairs that hash to the same bucket for at least 1 band are candidate pairs
- Tune b and r to catch most similar pairs, but few non-similar pairs

Locality-Sensitive Hashing



Locality-Sensitive Hashing

Assumption

There are enough buckets so that columns are unlikely to hash to the same bucket unless they are identical in a particular band

- From this, we can consider that "same bucket" means "identical in that band"
- Assumption needed only to simplify analysis that follows, not for correctness of algorithm

Locality-Sensitive Hashing: example

Consider signatures of 100 integers, choose $b = 20$ and $r = 5$
We want to find pairs with similarity $s \geq 0.8$

Assume C_1 and C_2 are 80% similar ($s = 0.8$)

Since $sim(C1, C2) \geq s$, we want C_1, C_2 to be a candidate pair
$\hookrightarrow$ C_1 and C_2 should hash to at least 1 common bucket

Probability C_1, C_2 are identical in a given band: $s^r = (0.8)^5 = 0.328$
Probability C_1, C_2 are not similar in all of the 20 bands:
$(1 - s^r)^b = (1 - 0.328)^{20} = 0.00035$

$\hookrightarrow$ Misses about 1/3000th of the 80%-similar docs (false negatives)
$\hookrightarrow$ Finds 99.965% pairs of documents with similarity $s \geq 0.8$

Locality-Sensitive Hashing: example

Consider signatures of 100 integers, choose $b = 20$ and $r = 5$
We want to find pairs with similarity $s \geq 0.8$

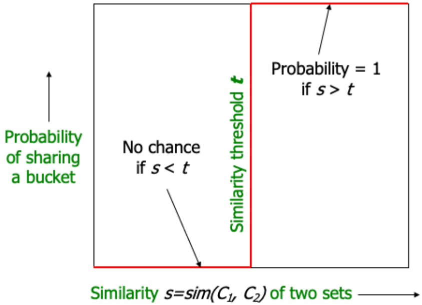
Assume C_1 and C_2 are 30% similar ($s = 0.3$)

Since $sim(C1, C2) < s$, we want C_1, C_2 to NOT be a candidate pair
$\hookrightarrow$ C_1 and C_2 should hash to NO common buckets

Probability C_1, C_2 are identical in a given band:
$s^r = (0.3)^5 = 0.0024$
Probability C_1, C_2 identical in at least 1 of 20 bands:
$1 - (1 - s^r)^b = 1 - (1 - 0.00243)^{20} = 0.0474$

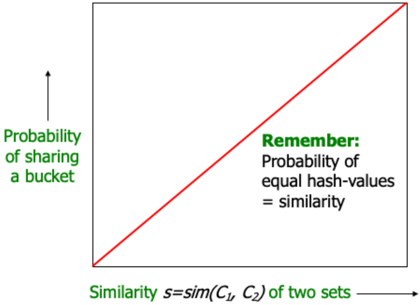
$\hookrightarrow$ Approximately 4.74% pairs of docs with similarity 0.3 end up becoming candidate pairs (false positives)

Locality-Sensitive Hashing



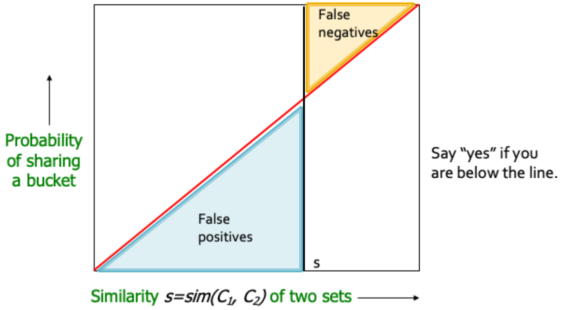
LSH – Optimal scenario: only pairs of sets with similarity $> \; s_{thresh}$ are selected as candidates

Locality-Sensitive Hashing



LSH – One band of one row

Locality-Sensitive Hashing

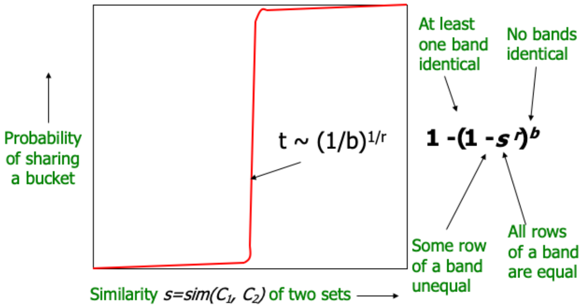


LSH – One band of one row: False negatives and False positives

Locality-Sensitive Hashing

- b bands, r rows/band
- Consider columns C1 and C2 with similarity s
- For any band (r rows):
  - Probability that all rows in band are equal $= s^r$
  - Probability that some row in band is unequal $= 1 - s^r$
  - Probability that no band identical $= (1 - s^r)^b$
  - Probability that at least 1 band identical $= 1 - (1 - s^r)^b$

Locality-Sensitive Hashing



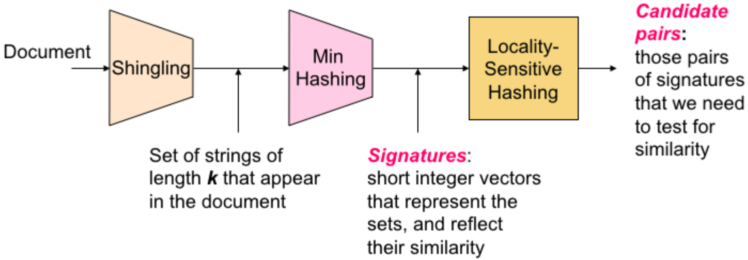
LSH – *b* bands of *r* rows: S-curve

Finding similar documents: pipeline

1. Pick a value of k and construct from each document the set of k-shingles. Optionally, hash shingles to shorter bucket numbers
2. Sort the document-shingle pairs to order them by shingle
3. Pick a length n and compute the minhash signatures for all documents
4. Choose a similarity threshold s_{thresh}. Pick a number of bands b and a number of rows r such that $b \cdot r = n$, and the threshold s_{thresh} is approximately $(1/b)^{1/r}$

   To avoid false negatives, select b and r to get a threshold lower than s_{thresh}; to limit false positives, select b and r to produce a higher threshold
5. Construct candidate pairs by applying the LSH technique
6. Examine each candidate pair's signatures and determine whether the fraction of components in which they agree is at least s_{thresh}
7. Optionally, check the original documents

Similar Documents: Overview



- **Shingling**: Converts a document into a set representation
- **Min-Hashing**: Convert large sets to short signatures, while preserving similarity
- **Locality-Sensitive Hashing**: Focus on pairs of signatures likely to be from similar documents