Foundations of Data Science

Lecture 13: Hashing

MING GAO

DaSE@ECNU
(for course related communications)
mgao@sei.ecnu.edu.cn

Dec. 23, 2016

Outline

Motivations

Bloom filter

Bloom filters find applications wherever fast set membership tests on large data sets are required.

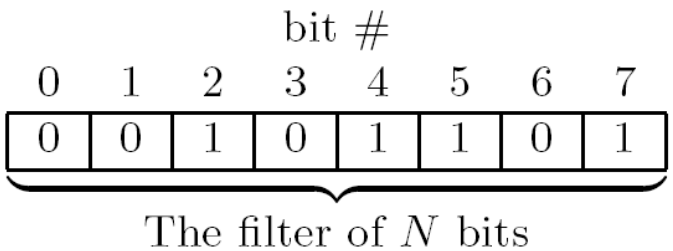
- It is a probabilistic method with a set error rate
- Errors can only occur on the side of inclusion: a true member will never be reported as not belonging to a set, but some non-members may be reported as members.
- Applications:
  - Spell checking
  - Distributed join
  - Distributed network caches
  - Differential Files

Definition

Definition

- Bloom filters use hash transforms to compute a vector (the filter) that is representative of the data set.
- Membership is tested by comparing the results of hashing on the potential members to the vector.
- In its simplest form the vector is composed of N elements, each a bit.
- An element is set if and only if some hash transform hashes to that location for some key.

$$m \text{ hash transforms} \quad \begin{cases} h_1(K) = 2 \\ h_2(K) = 5 \\ h_3(K) = 7 \\ h_4(K) = 4 \end{cases}$$



What is an error?

Error

Errors can occur when two or more transforms map to the same element.

- The membership test for a key K works by checking the elements that would have been updated if the key had been inserted into the vector.
- If all the appropriate flag bits have been set by hashes then k will be reported as a member of the set.
- If the elements have been updated by hashes on other keys, and not K, then the membership test will incorrectly report K as a member.

What is an error?

Error

Errors can occur when two or more transforms map to the same element.

- The membership test for a key K works by checking the elements that would have been updated if the key had been inserted into the vector.
- If all the appropriate flag bits have been set by hashes then k will be reported as a member of the set.
- If the elements have been updated by hashes on other keys, and not K, then the membership test will incorrectly report K as a member.

Notation

- S is a set of n elements.
- Set of k hash functions $h_1, \cdots, h_k$ with range $\{1, \cdots, m\}$.
- $m-$long array of bits initialized to 0.

Example

- We insert and query on a bloom filter of size $m = 10$ and number of hash function $k = 3$.
- Let $H(x)$ denote the result of the three hash functions which we will write as a set of three values $\{h_1(x), h_2(x), h_3(x)\}$.
- We start with an empty 10-bit long array:
- Insert x_0 and x_1, where $H(x_0) = \{1, 4, 9\}$ and $H(x_1) = \{4, 5, 8\}$.
- Query y_0 and y_1, where $H(y_0) = \{0, 4, 8\}$ (No) and $H(y_1) = \{1, 5, 8\}$ (Yes, but false positive).

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

H (X0) ={1, 4, 9}

0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	0	0	0	1

H (X1) ={4, 5, 8}

0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	1	0	0	1	1

Analysis of BF

x	$\left(1-\frac{1}{x}\right)^{-x}$
4	3.160494
16	2.808404
64	2.739827
256	2.723610
1024	2.719610
4096	2.718614
16384	2.718365
65536	2.718303
262144	2.718287
1048576	2.718283
4194304	2.718282

- After n elements inserted into bloom filter of size m, probability that a specific bit is still 0 is:

$$(1-\frac{1}{m})^{kn} \approx e^{-\frac{kn}{m}}.$$

The useful approximation comes from a well-known formula for calculating e

$$\lim_{x\to\infty}(1-\frac{1}{x})^{x} = e.$$

- Thus the probability that a specific bit has been flipped to 1 is

$$1-(1-\frac{1}{m})^{kn} \approx 1-e^{-\frac{kn}{m}}.$$

Analysis of BF cont.

- A false positive on a query of element x occurs when all of the hash function $h_1, \cdots, h_k$ applied to x return a filter position that has a 1.
- We assume hash functions to be independent. Thus the probability of a false positive is

$$f = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

Choose k to minimize false positive

- We are given m and n, so we choose a k to minimize the false positive rate.
- Let $p = e^{-\frac{kn}{m}}$. Then we have

$$f = \left(1 - e^{-\frac{kn}{m}}\right)^k = (1 - p)^k = e^{k \ln(1-p)}$$

- So we wish to minimize $g = k \ln(1 - p)$.

Analysis of BF cont.

- A false positive on a query of element x occurs when all of the hash function $h_1, \cdots, h_k$ applied to x return a filter position that has a 1.
- We assume hash functions to be independent. Thus the probability of a false positive is

$$f = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

Choose k to minimize false positive

- We are given m and n, so we choose a k to minimize the false positive rate.
- Let $p = e^{-\frac{kn}{m}}$. Then we have

$$f = \left(1 - e^{-\frac{kn}{m}}\right)^k = (1 - p)^k = e^{k \ln(1-p)}$$

- So we wish to minimize $g = k \ln(1 - p)$.

Analysis of BF cont.

Choose k to minimize false positive

- We could use calculus. Less messy, we notice that since $\ln\left(e^{-\frac{kn}{m}}\right) = -\frac{kn}{m}$.
- We have
$$g = k\ln(1-p) = -\frac{m}{n}\ln(p)\ln(1-p)$$
- And by symmetry, we see that g is minimized when $p = \frac{1}{2}$.
- Note that $p = e^{-\frac{kn}{m}}$, we have $k = \frac{m\ln 2}{n}$.
- Plugging back into $f = (1-p)^k$, we find the minimum false positive rate is $\left(\frac{1}{2}\right)^k \approx (0.6185)^{\frac{m}{n}}$.
- Recall p is the probability than any specific bit is still 0.
- So $p = 0.5$ corresponds to a half-full bloom filter array.

m, n, k Examples

False positive rates for choices of k given $\frac{m}{n}$.

m/n	k=1	k=2	k=3	k=4	k=5
2	0.393	0.400			
3	0.283	0.237	0.253		
4	0.221	0.155	0.147	0.160	
5	0.181	0.109	0.092	0.092	0.101
6	0.154	0.0804	0.0609	0.0561	0.0578
7	0.133	0.0618	0.0423	0.0359	0.0347
8	0.118	0.0489	0.0306	0.024	0.0217

Problem of BF

No deletion support

Suppose that we have a set that is changing over time, with elements being inserted and deleted. If we need to remove an item from the Bloom filter, how do we do this?

- Recall our example Bloom filter of two items $H(x_0) = \{1, 4, 9\}$ and $H(x_1) = \{4, 5, 8\}$.
- If we delete $H(x_0)$, query x_1, return *No* (False negative).
- Cannot delete one without clobbering the other since they share address 4.

0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	1	0	0	1	1

Delete x0

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	0	0	1	0

Counting bloom filter?

- In a counting Bloom filter, each entry in the Bloom filter is not a single bit but rather a small counter.
- When an item is inserted, the corresponding counters are incremented.
- When an item is deleted, the corresponding counters are decremented.
- The analysis from [Fan et al. 00] reveals that 4 bits per counter should suffice for most applications.

Motivation

Jaccard coefficient

Jaccard coefficient between two set $A = \{0, 1, 2, 5, 6\}$ and $B = \{0, 2, 3, 5, 7, 9\}$ can be computed as

$$Jac(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|\{0, 2, 5\}|}{|\{0, 1, 2, 3, 5, 6, 7, 9\}|} = \frac{3}{8}.$$

- However, it is not easy to compute if the sets are quite large.

Element	S_1	S_2	S_3	S_4
1	1	0	0	1
2	1	0	1	0
3	0	1	1	0
4	0	0	1	1
5	1	0	0	0
6	0	0	1	1

Shingling (Matrix representation)

Consider sets: $S_1 = \{1, 2, 5\}$, $S_2 = \{3\}$, $S_3 = \{2, 3, 4, 6\}$, and $S_4 = \{1, 4, 6\}$. We can represent these four sets as a single matrix. But, it wastes a lot of space.

Hash clustering

Definition

Consider the set of element is $\mathcal{E} = \{1, 2, 3, 4, 5, 6\}$ and there is also a set of possible clusters $\mathcal{C} = \{A, B, C\}$ that is smaller is size than the set of elements $|\mathcal{C}| < \mathcal{E}$. We consider a random hash function that maps each element $e \in \mathcal{E}$ to a consistent location in $\mathcal{C}$. For instance $h : \mathcal{E} \to \mathcal{C}$ is defined so

$$h : [1, 2, 3, 4, 5, 6] \to [A, B, B, C, A, A].$$

Cluster	S_1	S_2	S_3	S_4
A	1	0	1	1
B	1	1	1	0
C	0	0	1	1

Similarity estimation

$JS(S_1, S_3) = \frac{1}{3}, JS_{clu}(S_1, S_3) = \frac{2}{3}$
$JS(S_3, S_4) = \frac{2}{5}, JS_{clu}(S_3, S_4) = \frac{2}{3}$

- Similarity generally increase.
- This may appear not useful for this applications because it is hard to understand the errors induced by the clustering.

Min hashing

Element	S_1	S_2	S_3	S_4
1	1	0	0	1
2	1	0	1	0
3	0	1	1	0
4	0	0	1	1
5	1	0	0	0
6	0	0	1	1

After Permutation

Element	S_1	S_2	S_3	S_4
2	1	0	1	0
5	1	0	0	0
6	0	0	1	1
1	1	0	0	1
4	0	0	1	1
3	0	1	1	0

Definition

The next approach, called min hashing, initially seems even simpler than the clustering approach. It will need to evolve through several steps to become a useful trick.

- Randomly permute the items (by permuting the rows of the matrix).
- Record the first 1 in each column $h(S_1) = 2$, $h(S_2) = 3$, $h(S_3) = 2$, and $h(S_4) = 6$.
- Estimate the Jaccard similarity $JS(S_i, S_j)$ as

$$\widehat{JS}(S_i, S_j) = \begin{cases} 1, & \text{if } h(S_i) = h(S_j); \\ 0, & \text{otherwise.} \end{cases}$$

Analysis of min hashing

Lemma

$P[h(S_i) = h(S_j)] = E[\widehat{JS}(S_i, S_j)] = JS(S_i, S_j)$.

Proof: There are three types of rows

- (Tx) There are x rows with 1 in both column.
- (Ty) There are y rows with 1 in one column and 0 in the other.
- (Tz) There are z rows with 0 in both column.

The Jaccard coefficient is precisely $JS(S_i, S_j) = \frac{x}{x+y}$.

Let row r be the $\min\{h(S_i), h(S_j)\}$. It is either type (Tx) or (Ty), and it is (Tx) with probability exactly $\frac{x}{x+y}$ since the permutation is random. This is the only case that $h(S_i) = h(S_j)$.

Otherwise, S_i or S_j has 1, but not both.

Analysis of min hashing cont.

Analysis

To get a better estimate, we need to repeat this several k times.

- Consider k random permutations $\{h_1, \cdots, h_k\}$ and also k r.v. $\{X_1, \cdots, X_k\}$.
- Let $\widehat{JS}(S_i, S_j) = \frac{1}{k}\sum_{i=1}^{k} X_i$. Thus, we have $E(\widehat{JS}(S_i, S_j)) = JS(S_i, S_j)$ and $D(\widehat{JS}(S_i, S_j)) = \frac{JS(S_i,S_j)(1-JS(S_i,S_j))}{k}$.
- Thus the Jaccard coefficient is within ϵ error with probability at leat $1-\delta$ is we repeat this $k = \frac{2\ln(1/\delta)}{\epsilon^2}$.

Motivation

Entity resolution

The entity resolution problem is to examine a collection of records and determine which refer to the same entity. Typically, we want to merge records if their values in corresponding fields are similar.

- Company B had about 1 million records of all its customers.
- Company A had about 1 million records describing customers, some of whom it had signed up for B.
- Records had name, address, and phone, but for various reasons, they could be different for the same person.
  - Step 1: Design a measure ("score") of how similar records are.
  - Step 2: Score all pairs of records; report high scores as matches.

Analysis

- $(1million)^2$ is too many pairs of records to score.
- Solution: a simple LSH.

Locality Sensitive Hashing

Steps

- Shingling: represent sets to a matrix.
- Min-hash: compute signatures of sets.
- Locality sensitive hashing: hash sets into buckets.

Rand. Permuts.

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

S_1	S_2	S_3	S_4
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

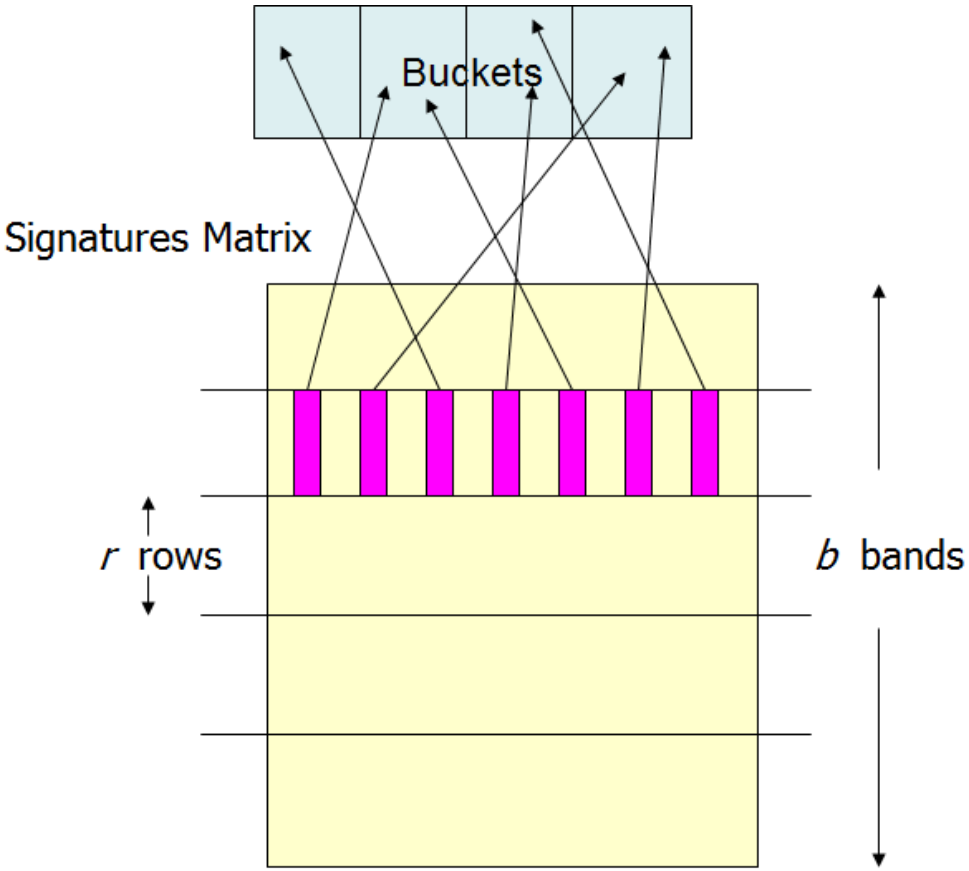
Signatures matrix

S_1	S_2	S_3	S_4
2	1	2	1
2	1	4	1
1	2	1	2

Similarities:

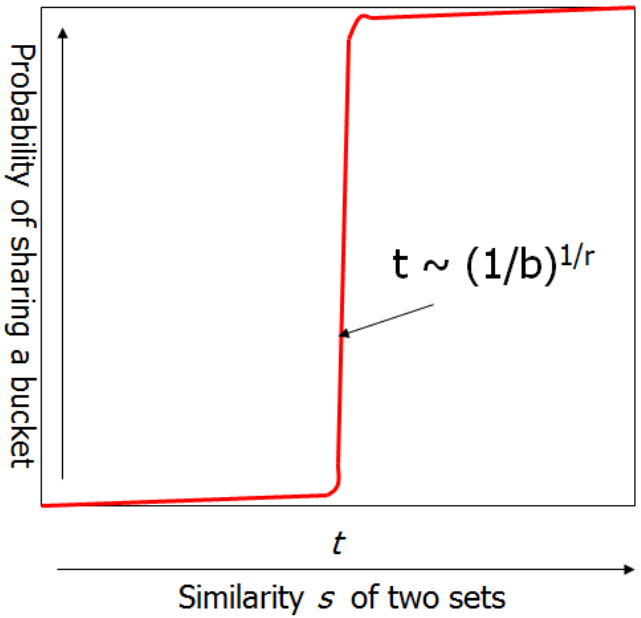
	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Locality Sensitive Hashing



- 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.
- Candidate column pairs are those that hash to the same bucket for ≥ 1 band.

Analysis of LSH



r = 3, b = 10

s	$1-(1-s^r)^b$
.2	0.0771
.3	0.2394
.4	0.4838
.5	0.7369
.6	0.9122
.7	0.9850
.8	0.9992

Analysis

- The probability of all equal rows is s^r, and the probability of some unequal rows is $1 - s^r$.
- The probability of no shared bucket is $(1 - s^r)^b$.
- Thus, the probability of at least one shared bucket is $1 - (1 - s^r)^b$.

Generalization of LSH

Definition

Given $\mathcal{C}$ and $sim(x, y)$, a locality sensitive hash function family $\mathcal{F}$ operates on $\mathcal{C}$, such that for any $x, y \in \mathcal{C}$,

$$P_{h \in \mathcal{F}}[h(x) = h(y)] = sim(x, y).$$

- Sets: $sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$.
- Vector: $sim(\overrightarrow{u}, \overrightarrow{v}) = 1 - \frac{\theta(\overrightarrow{u}, \overrightarrow{v})}{\pi}$, where $\theta(\overrightarrow{u}, \overrightarrow{v})$ is the angle between $\overrightarrow{u}$ and $\overrightarrow{v}$.

Existence of LSH

If similarity function $sim(x, y)$ admits a locality sensitive hash function family, then the distance function $1 - sim(x, y)$ satisfies the triangle inequality[a].

[a] https://www.cs.princeton.edu/courses/archive/spring02/cs493/lec16.pdf

Existence of LSH cont.

How to use the theorem

The theorem could be used to prove that LSH function families do not exist for certain set similarity measures.

- Dice coefficient: $dice(A, B) = \frac{2|A \cap B|}{|A|+|B|}$.
- Overlap coefficient: $Ovl(A, B) = \frac{|A \cap B|}{\min\{|A|,|B|\}}$.

Example

Consider the sets $A = \{a\}$, $B = \{b\}$, and $C = \{a, b\}$. For the Dice coefficient, we have $dice(A, C) = \frac{2}{3}$, $dice(C, B) = \frac{2}{3}$ and $dice(A, B) = 0$. However, we have $(1 - dice(A, C)) + (1 - dice(C, B)) < 1 - dice(A, B)$.

Drawbacks

Load balance is big problem for locality sensitive hashing.

Take-home messages

- Bloom Filter
  - Motivation
  - Example
  - Model
- Min-hashing
- Locality sensitive hashing