Tuesday, September 26, 2017 4:57 PM

Lab 5

In this Lab:

two real-life applications of probability:

- Bloom filter
- Min Hashing.

Hash function

- both the two applications reply on hash functions
- a hash function is a function

$$h: \Omega \to [m].$$

Ω is a set of objects, usually strings

$[m] = \{0, 1, 2, \cdots, m-1\}.$

- Ω is usually huge, m is usually relatively small.

- for a given string s, $h(s)$ is always the same
- but for a random s, $h(s)$ can be regarded as uniformly random in $[m]$.

Bloom filter

Motivation: efficient membership check

What is membership check?

- Is Donald Trump a student of ECE314?
- Is this website a malicious website?

"Efficient" means:

- less memory (space)
- less search time (time)

A natural approach: built a list of the members, and check the membership of any given name by searching the list.

Problem of the list approach:

- Memory requirement grows linearly with n, where n is the number of members we have
- Searching time grows as n gets large.

Bloom filter Approach

- A Bloom filter is a bit array. B
- n = total number of students in ECE314.
- m = # of bits in B.
- k = # of hash functions

Two stages: update/store and check

Tuesday, September 26, 2017 7:14 PM

eg $m = 10$

$k = 2$, h_1, h_2.

Students = {'Jack', 'Bruce', 'Jane'}.

$n = 3$

Update / store

	1	2	3	4	5	6	7	8	9	10
B	0	0	0	0	0	0	0	0	0	0

$h_1('Jack') = 5$

$h_2('Jack') = 3$

$h_1('Bruce') = 3$

$h_2('Bruce') = 9$

$h_1('Jane') = 1$

$h_2('Jane') = 4$

	1	2	3	4	5	6	7	8	9	10
B	1	0	1	1	1	0	0	0	1	0

Tuesday, September 26, 2017 7:19 PM

Check

B

1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	0	0	0	1	0

Is 'Jack' a student of ECE314?

$h_1(\text{'Jack'}) = 5 \qquad B[5] = 1$ ✓

$h_2(\text{'Jack'}) = 3 \qquad B[3] = 1$ ✓

So, possibly yes.

Is 'Donald' a student of ECE314?

$h_1(\text{'Donald'}) = 3 \qquad B[3] = 1$ ✓

$h_2(\text{'Donald'}) = 8 \qquad B[8] = 0$ ✗

So, definitely no.

Is 'Adam' a student of ECE314?

$h_1(\text{'Adam'}) = 1$ ✓

$h_2(\text{'Adam'}) = 9$ ✓

So, probably yes.

The performance of a Bloom filter is measured by the false positive probability.

$p = Pr$(item is actually not in the list / the Bloom filter says Yes)

- See the lab for details about how to calculate p. p is a function of n, m, k.
- In practice, we usually have a desired value for p. and we can make a rough estimate of n.

Then we choose the optimal m & k to construct the Bloom filter.

Min Hashing

Motivation:

Compare the similarity of two sets of items, e.g. words.

Application: Plagiarism detection

Read the two "Jack Jill" paragraphs to the students.

shingles	A	B
Jack Jill	1	1
Jill went	1	1
went up	1	0
up hill	1	1
hill fetch	1	0
fetch pail	1	0
pail water	1	1
hill Jack	0	1
went get	0	1
get pail	0	1

Jaccard measure

- quantifies the silimarity between set A & B.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{4}{10}.$$

$J = 1 \Rightarrow A \cap B = A \cup B \Rightarrow$ very similar

$J = 0 \Rightarrow A \cap B = 0 \Rightarrow$ very different.

So far so good. But what if we have 1000 documents and each document is very large?
Then the union of all possible shingles is very large, and A_i for each i is likely to be sparse (most entries are zeros).
Computationally very inefficient.

Min Hashing

Goal: efficiently estimate $J(A,B)$.

Random select a shingle s from $A \cup B$,

Let
$$X = \begin{cases} 1 & \text{if } s \in A \cap B \\ 0 & \text{if } s \notin A \cap B \end{cases}$$

So $E[X] = J(A,B)$

Given a hash function h_1, mapping a word to a number,

let
$$h_1(S) = \min \{ h_1(s) : s \in S \}.$$

signature of set S.

Idea:

- Generate samples of X without constructing $A \cup B$.

- Randomly select a shingle $s \in A \cup B$ is equivalent to return $h_1(A \cup B)$

- But $h_1(A \cup B) = \min\{h_1(A), h_1(B)\}$.

- So $s \in A \cap B$. is equivalent to $h_1(A) = h_1(B)$

So

$$X = \begin{cases} 1 & \text{if } h_1(A) = h_1(B) \\ 0 & \text{if } h_1(A) \neq h_1(B) \end{cases}$$

By LLN.

$$\frac{x_1 + \cdots + x_n}{n} \to \mathbb{E}[X] = J(A, B)$$

use n different hash functions

Alternatively,

$h_k(S)$ = the k smallest hash values from apply a hash function h to items in S.

See Lab file for details.