Scalable ML

10605-10805

The Bloom Filter

Barnabás Póczos

Probability and Computing
Randomized and Probabilistic Techniques in Algorithms and Data Analysis
Section 5.2

Michael Mitzenmacher and Eli Upfal

Motivation

Our goal is to **develop a data structure**

- that can **store elements** from a Universe (e.g. passwords, sentences, etc),
- and the IsMember(x) query is very **fast**

[For example, we have a long list of *m* banned passwords, and we want to make sure users will not choose these words when they change their passwords]

What kind of data structure should we use and how can we check memberships in a data structure?

Membership Checking

Implementing IsMember(x)

Trivial data structure:

- put each element into a memory bin [$O(m)$ storage]
- For each element s check one-by-one if s=x [$O(m)$ compute time]

Faster Query:

- Sort the elements first then put them into a memory bin [$O(m)$ storage]
- During the IsMember(x)query use binary search [$O(log(m))$ compute time]

Can we develop another data structure with faster queries?
Ideally with constant query time?

The Bloom Filter

Definition: [Bloom Filter]

The Bloom Filter data structure consists of an **array of n bits**

n memory cells:	**A[0]**	**A[1]**	**A[2]**	**…**				**A[n-1]**
Each contains a bit [0 or 1]	0	0	1	1	0	1	1	1

It has two **operations**:

- Insert([element1, element2,…])
- IsMember(element)

The Bloom Filter

The Bloom Filter also uses *k* independent **hash functions** that map elements of our Universe to the location of the memory cells

$$h_1, h_2, \ldots, h_k \qquad h_i : U \rightarrow N = \{0, 1, \ldots, n-1\}$$

Let $S = \{S_1, \ldots, S_m\} \subset U$

E.g. list of banned passwords

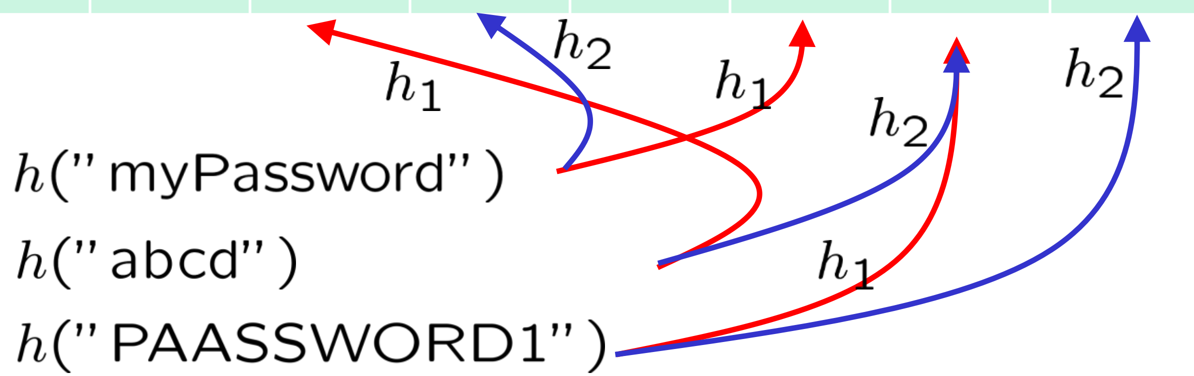
We want to represent these *m* elements with a Bloom Filter

n memory cells:

Each contains a bit [0 or 1]

0	1	2	...				n-1
0	0	1	1	0	1	1	1

Key
mypassword
abcd
PASSWORD1



The Insert and IsMember operations:

$\star$ Insert($[s_1, s_2, \ldots, s_m]$):

Set $A[0] = A[1] = \ldots = A[n-1] = 0$

Set $A[h_i(s_j)] = 1$, for all $1 \leq i \leq k$, $1 \leq j \leq m$

$\star$ IsMember(x):

Our goal is to determine if $x \in S = \{s_1, s_2, \ldots, s_m\}$

Algorithm: Check if $h_i(x) = 1$ for all $1 \leq i \leq k$.

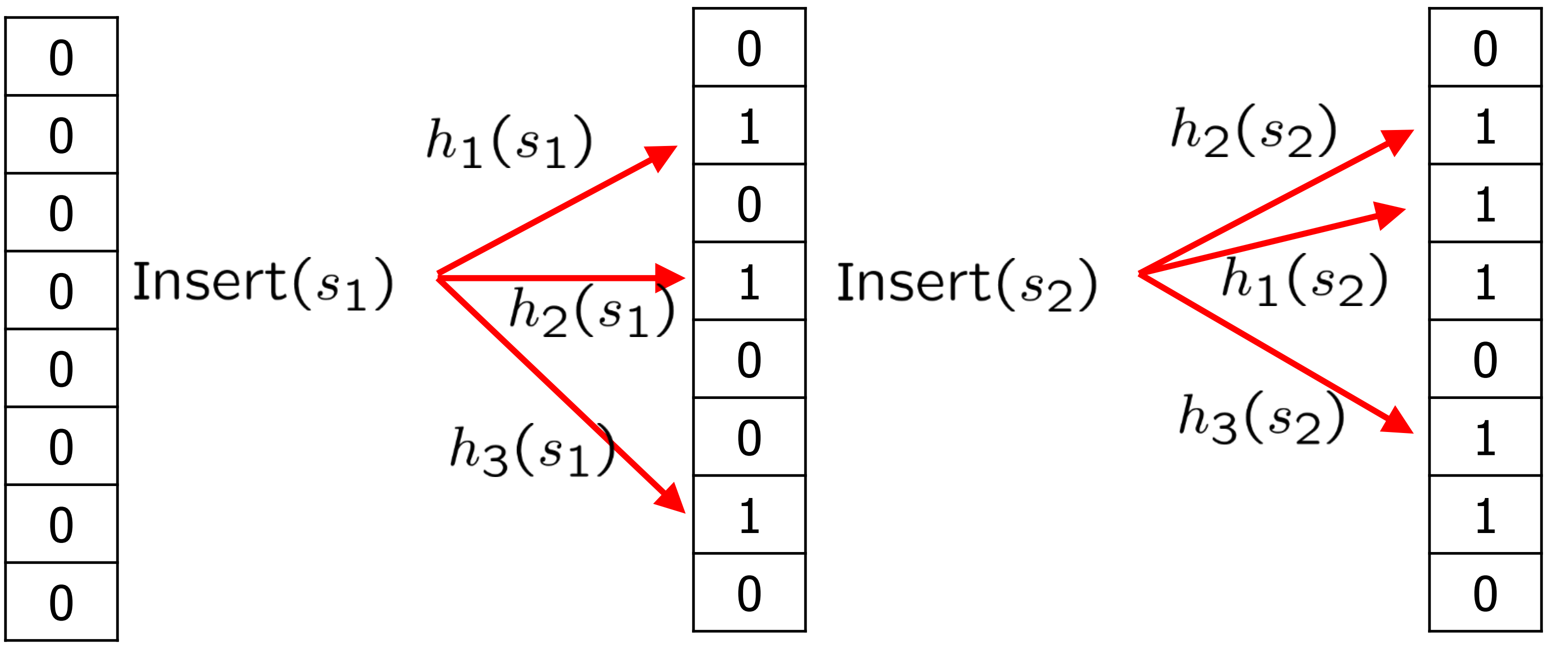
If $h_i(x) = 1$ for all $1 \leq i \leq k$, we say $x \in S$.

Although we might be wrong.

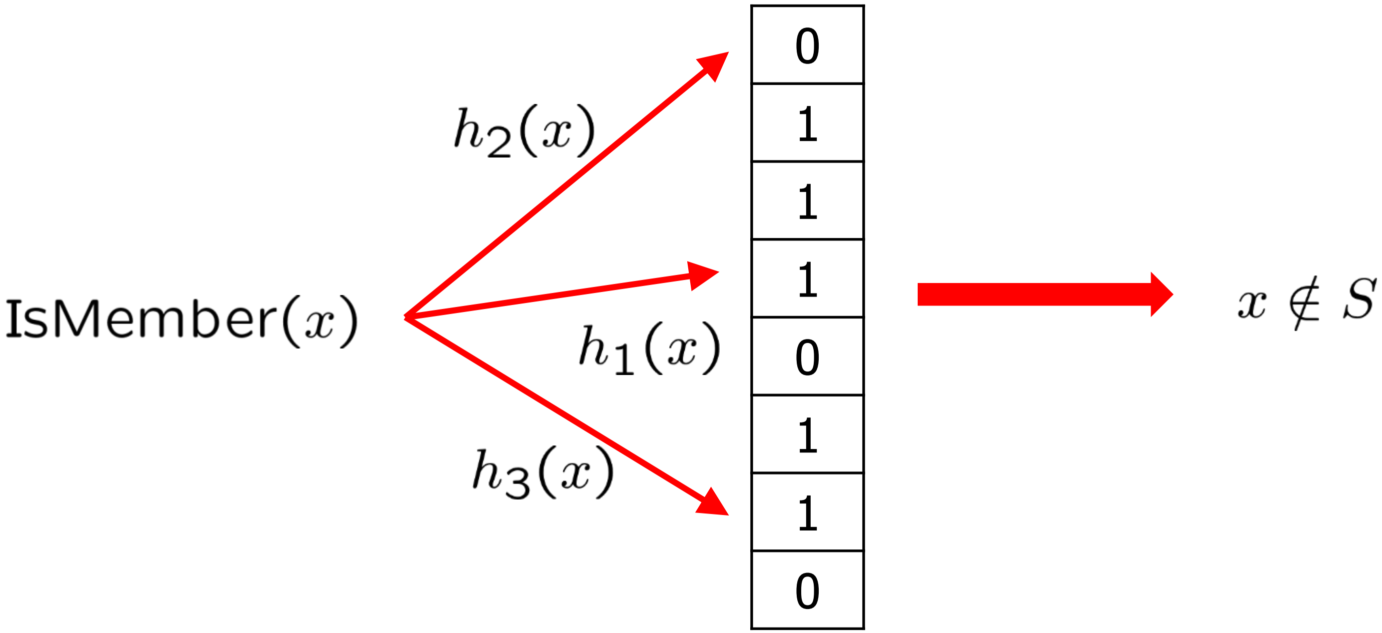
If $\exists i$ such that $h_i(x) = 0$, we say $x \notin S$ This is always correct.

The Bloom filter might give us **false positives, but no false negatives.**

Bloom Filter Example (Insert)



Bloom Filter Example (IsMember)



Probability of False Positives

If we can assume that $h_1, h_2, \ldots, h_k$ hash functions are random and independent, then we can calculate the probability of false positives.

After s_1 insterted with h_1: $\mathbb{P}[A[i] = 1] = \frac{1}{n}$

$$\mathbb{P}[A[i] = 0] = 1 - \frac{1}{n}$$

After s_1 insterted with h_1, h_2: $\mathbb{P}[A[i] = 0] = \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{n}\right) = \left(1 - \frac{1}{n}\right)^2$

$$\mathbb{P}[A[i] = 1] = 1 - \left(1 - \frac{1}{n}\right)^2$$

After s_1 insterted with $h_1, h_2, \ldots, h_k$:

$$\mathbb{P}[A[i] = 0] = \left(1 - \frac{1}{n}\right)^k$$

$$\mathbb{P}[A[i] = 1] = 1 - \left(1 - \frac{1}{n}\right)^k$$

Probability of False Positives

After $s_1, s_2, \ldots, s_m$ insterted with $h_1, h_2, \ldots, h_k$:

$$\mathbb{P}[A[i] = 0] = \left(1 - \frac{1}{n}\right)^{km} \approx e^{-\frac{km}{n}}$$

$$\mathbb{P}[A[i] = 1] = 1 - \left(1 - \frac{1}{n}\right)^{km} \approx 1 - e^{-\frac{km}{n}}$$

Therefore, for an $x \notin S = \{s_1, s_2, \ldots, s_m\}$

$$\mathbb{P}[x \text{ gives a false positive}] = \mathbb{P}[A[h_1(x)] = 1, A[h_2(x)] = 1, \ldots, A[h_k(x)] = 1]$$

$$= \prod_{i=1}^{k} \mathbb{P}[A[h_i(x)] = 1]$$

$$\approx \left(1 - e^{-\frac{km}{n}}\right)^k$$

Let m, n be fixed.

What is the optimal number of hash functions (k)?

The Optimal Number of Hash Functions

What is the optimal number of hash functions (k)?

Using more hash functions can help to find more zero bits when running IsMember(x) and therefore prove $x \notin S$

Too many hash functions might just fill in the memory cell array with ones too quickly.

We need to solve:

$$\text{argmin}_k \left(1 - e^{-\frac{km}{n}}\right)^k =?$$

$$= \text{argmin}_k \; k \log\left(1 - e^{-\frac{km}{n}}\right)$$

$$\frac{\partial}{\partial k} k \log\left(1 - e^{-\frac{km}{n}}\right) = 0 \Rightarrow k = (\log 2)\frac{n}{m}$$

$$\Rightarrow e^{-\frac{km}{n}} = e^{-\log 2} = \frac{1}{2}$$

The Optimal False Positive Rate

Using this k, the false positive rate is

$$\begin{aligned}
\left(1 - e^{-\frac{km}{n}}\right)^k &= \left(1 - \frac{1}{2}\right)^k \\
&= \left(\frac{1}{2}\right)^k \\
&= \left(\frac{1}{2}\right)^{\log 2 \frac{n}{m}} \\
&= (0.6185)^{\frac{n}{m}}
\end{aligned}$$

This goes to zero exponentially in $\frac{n}{m}$

The IsMember(x) query time is $O(k) = O(\frac{n}{m})$.

This can be much better than $O(\log m)$.

Thanks for your Attention! ☺