| CS 493: Algorithms for Massive Data Sets | Compressed Bloom Filters<br>Error Correcting Codes |
|---|---|
| 02/26/02 | Scribe: Adriana Karagiozova |

# 1 Review of Bloom Filters

A *Bloom filter* is a space-efficient randomized data structure for representing sets in order to support membership queries.

We use an $m$-bit array to represent the set $S = \{s_1, s_2, \ldots, s_n\}$. The idea is to use $k$ independent hash functions $h_1, h_2, \ldots, h_k$, such that for $1 \leq i \leq k$, $h_i : x \mapsto \{1, 2, \ldots, m\}$, for $x \in S$. The $m$-bit array is initialized to all 0's and upon the insertion of an element $x$, $h_i(x)$ is set to 1 for $1 \leq i \leq k$. To check whether $x$ is in $S$, check whether $h_i(x) = 1$ for $1 \leq i \leq k$.

A Bloom filter can yield a *false positive*, where it suggests that an element $x$ is in $S$ even if it is not. Recall that the probability of having a particular bit not set is $p = (1 - \frac{1}{m})^{kn} \approx e^{-\frac{kn}{m}}$ and that the probability of a false positive is $f = (1 - p)^k$. Recall also that

$$\arg\min_p f = \frac{1}{2}$$

which yields

$$e^{-\frac{kn}{m}} = \frac{1}{2}$$

$$\frac{kn}{m} = \ln 2$$

$$k = \frac{m}{n} \ln 2$$

Thus the minimum false positive rate is

$$f = \left(\frac{1}{2}\right)^{\frac{m}{n} \ln 2} \approx (0.6185)^{\frac{m}{n}}$$

# 2 Compressed Bloom Filters

Next we ask the question what happens if we can compress the filter. In the original setting we fixed $m$ and $n$ and found the value of $k$ that minimizes $f$. Now we consider an additional parameter $z$, which stands for the size of the compressed filter. We assume that we can achieve optimal compression and thus $z = H(p)\ m$. Now we fix $z$ and $n$ and try to find the

values of $m$ and $k$ that minimize $f$.

Consider what happens in the case $k = \frac{m}{n} \ln 2$ or equivalently $p = \frac{1}{2}$, which minimized $f$ for the standard uncompressed Bloom filter. Notice that under the assumption of good random hash functions, the bit array appears to be a random string of $m$ 0's and 1's, with each entry being 0 or 1 independently with probability $\frac{1}{2}$. Hence nothing can be gained by compressing the filter in this case.

From the discussion of uncompressed Bloom filters, we know that if we let $m = z$ and $k = \frac{m}{n} \ln 2$, we can obtain a false positive rate $f = (0.6185)^{\frac{z}{m}}$. The question is can we do better by choosing $m$ and $k$ differently. It turns out that the above choice of $k$ is the *worst* choice possible once we allow for compression.

We analyze the case of compressed Bloom filter similarly to the uncompressed case. Recall that $f = (1-p)^k$ and $z = H(p)\ m$, where $H(p) = -p \log_2 p - (1-p) \log_2(1-p)$. Expressing $k$ in terms of $m$, $n$ and $p$ we get

$$k = -\frac{m}{n} \ln p$$

Hence

$$\begin{aligned} f &= (1-p)^k \\ &= (1-p)^{-\frac{m}{n} \ln p} \\ &= (1-p)^{-\frac{\ln p}{H(p)} \left(\frac{z}{n}\right)} \\ &= \exp\left( \frac{-\ln p\ \ln(1-p)}{(-\log_2 e)(p\ \ln p + (1-p)\ln(1-p))} \left(\frac{z}{n}\right)\right) \end{aligned}$$

Since $z$ and $n$ are fixed and $z \geq n$, minimizing $f$ is equivalent to minimizing

$$\exp\left( \frac{\ln p\ \ln(1-p)}{(p\ \ln p + (1-p)\ln(1-p))} \right)$$

This, in turn, is minimized when the exponent is minimized, i.e. when the term

$$\gamma = \frac{p}{\ln(1-p)} + \frac{(1-p)}{\ln p}$$

is maximized. Note that

$$\frac{d\gamma}{dp} = \frac{1}{\ln(1-p)} - \frac{1}{\ln p} + \frac{p}{(1-p)\ln^2(1-p)} - \frac{1-p}{p\ \ln^2 p}$$

It is easy to check that

$$\frac{d\gamma}{dp} \quad \begin{cases} < 0 & \text{if } p < \frac{1}{2} \\ = 0 & \text{if } p = \frac{1}{2} \\ > 0 & \text{if } p > \frac{1}{2} \end{cases}$$

This means that the probability of a false positive with a compressed Bloom filter occurs when $p = \frac{1}{2}$ or $k = \frac{m}{n} \ln 2$, i.e the number of hash functions that minimized the false positive rate without compression maximizes the false positive rate with compression.

The maximum of $\gamma$ occurs when $p \to 0$ or $p \to 1$. We can use the approximation

$$\ln(1-p) \to p \quad \text{for } p \to 0$$

Hence

$$\gamma \to -1 \quad \text{as } p \to 0$$

This gives us a minimum false positive rate of $f = e^{-\frac{z}{n} \ln 2} = (0.5)^{\frac{z}{n}} < (0.6185)^{\frac{z}{n}}$, which is a significant improvement over the uncompressed Bloom filter case.

This result is to be taken with a grain of salt though. Notice that as $p \to 0$, the number of hash functions approaches infinity and as $p \to 1$, the number of hash functions approaches 0. In the former case, the time to access the filter can become impractically large and in the latter case the size of the bits array $m$ will have to be very large to ensure that most of the array contains 0's.

Let's look at the problem from another point of view — suppose that we wish to optimize the final compressed size $z$ while keeping the same false positive rate as in the uncompressed filter case. The false positive rate in the compressed case is $(0.5)^{\frac{m}{n} \ln 2}$. Thus the optimal compressed size that gives the same false positive rate is $z = m \ln 2$, saving roughly 30% space.

# 3 Limitations of the Compressed Bloom Filter Model

One problem with the above representation is that we allow only additions to the set but no removals. To fix this we can consider maintaining a counter of how many times each bit in the bit array has been set. Upon the addition of an element, we increment the appropriate counters. Upon deletion of an element, we decrement the appropriate counters. It can be shown that the average value of the counters tends to be pretty low, so maintaining them does not produce a significant overhead.

Throughout the above discussion we also made a number of simplifying assumptions, like the fact that we can obtain an arbitrarily large number of independent random hash functions,

or that whether a bit is set is independent of whether any other bit is set. It turns out, however, that both assumptions are reasonable in practice.

# 4 Error-correcting Codes

An important question to consider is what happens if there are errors in the representation or transmission of data. When data is stored or transmitted it is possible that the physical representation changes slightly. If we are transmitting a stream of bits and a few of them change we would like to be able to recover the original bit sequence. Another motivational example is to consider a server transmitting packets to a client. In this case some packets never make it to the final destination. Could the original message still be recovered and if so, how?

The foundations of this area were set by C. E. Shannon and R. W. Hamming in the late 50's. Shannon's paper "A Mathematical Theory of Communication"[1] set the mathematical foundations of communications systems and Hamming's paper "Error Detecting and Error Correcting Codes"[2] introduced the notion of error-correcting codes.

So far we have assumed *noiseless* communication and knowing something about the pattern of the bits transmitted we could compress the data and thus reduce the communication. Now we look at *noisy* communication where the question is how may bits we can communicate correctly.

Suppose that for any $n$ bits transmitted, we get $nR$ bits of information. Suppose we have an encoding function

$$E : \{0,1\}^{nR} \rightarrow \{0,1\}^n$$

and a decoding function (probabilistic)

$$D : \{0,1\}^n \rightarrow \{0,1\}^{nR}$$

Then we can define a channel as a model of how transmission takes place.

We will consider two types of channels: the binary symmetric channel (BSC) and the erasure channel. For the binary symmetric channel when the transmitter transmits a bit, the receiver receives it correctly with probability $(1-p)$ and receives it flipped with probability $p$. For the erasure channel when the transmitter transmits a bit, the receiver receives it correctly with probability $(1-p)$ and receives an *erasure* symbol '?' with probability $p$.

[1] C. E. Shannon. A mathematical theory of communication. Bell System Tech. J., 27:379–423; 623–656, 1948

[2] Error Detecting and Error Correcting Codes (R.W. Hamming, 1950) http://www.tcs.hut.fi/~helger/crypto/link/coding_theory/

**Definition 4.1** *The amount of useful information per bit is called the rate of the transmission scheme and is denoted by $R$.*

**Definition 4.2** *The maximum rate at which one can safely transmit is called the capacity of the channel and is denoted by $C$.*

It can be shown that the capacity of a channel is related to the entropy of a certain probability distribution. Information is transferred from the transmitter to the receiver according to some function

$$\Sigma \to \Gamma$$

where $\Sigma$ is the alphabet of the transmitter message and $\Gamma$ is the alphabet of the receiver. Let $p(x)$ be a probability distribution on $\Sigma$. Let $X$ be the random variable on $\Sigma$ that takes values according to $p(x)$ and let $Y$ be the random variable on $\Gamma$. It turns out that the capacity is the maximum of the mutual information between $X$ and $Y$ over all distributions $p(x)$, i.e.

$$C = \max_{p(x)}(H(X) - H(X|Y)) = \max_{p(x)}(H(Y) - H(Y|X))$$

For the binary symmetric channel $H(X) = 1$, $H(X|Y) = 0$ when $p(x)$ is the uniform distribution. Hence the capacity is $C = 1 - H(p) = 1 + p\log_2 p + (1-p)\log_2(1-p)$.
For the erasure channel, the capacity is $C = 1 - p$.

Now we will show a transmission scheme for the binary symmetric channel with rate equal to the capacity of the channel.

**Theorem 4.3 (Shannon)** *For a binary symmetric channel with probability of flipping a bit $p$, there exit $E$ and $D$ with rate $C - \varepsilon$ for arbitrarily small $\varepsilon$.*

The idea is to choose $2^{nR}$ strings of length $n$ at random and to use them as the codewords. Upon receiving a message, the receiver looks at all the code words and chooses whichever is the closest one as the correct one. Before proceeding with the proof, we need a few preliminary facts.

**Fact 4.4 (Stirling's approximation)**

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + O\left(\frac{1}{n^2}\right)\right)$$

**Fact 4.5 (Chernoff bounds)** *Let $X_1, X_2, \ldots, X_n$ be random variables and let $Y = \sum_{i=1}^n X_i$. For simplicity assume that the $X_i$'s are identically distributed and $X_i \in [0,1]$ for all $i$. Let $\gamma = E[X_i]$. Then*

$$Pr[|Y - n\gamma| > \varepsilon\gamma n] < e^{-\frac{\varepsilon^2}{3}n}$$

*Proof.* We will use Chernoff bounds to show that when $n$ bits are transmitted the number of errors is very close to $np$. Let

$$X_i = \begin{cases} 0 & \text{no error in bit } i \\ 1 & \text{error in bit } i \end{cases}$$

The $Y$ is the random variable corresponding to the number of errors in the transmission and we know that

$$Pr[|Y - np| > \varepsilon np] < e^{-\frac{\varepsilon^2}{3}n}$$

Let $R = C - \gamma$, where $C = 1 - H(p)$. Pick $2^{nR}$ strings of length $n$. With high probability $Y < (p + \varepsilon)n$. It is possible that the receiver associates the wrong code word with the message. We can estimate this probability:

$$Pr[\text{a fixed message} \leq (p+\varepsilon)n \text{ bits from the received message}] \leq \frac{\sum_{k=0}^{(p+\varepsilon)n} \binom{n}{k}}{2^n}$$

For $p$ sufficiently small, the largest term in the numerator sum will be $nchoosepn$. We estimate the term using Stirling's approximation:

$$\binom{n}{pn} \approx \frac{\left(\frac{n}{e}\right)^n}{\left(\frac{pn}{e}\right)^{pn}\left(\frac{(1-p)n}{e}\right)^{(1-p)n}} = \frac{1}{p^{pn}(1-p)^{(1-p)n}}$$

Therefore

$$\binom{n}{pn} \approx 2^{-n[p\log_2 p + (1-p)\log_2(1-p)]} = 2^{nH(p)}$$

Since we have $2^n R$ messages, where $R = C - \gamma$, the probability of error (the receiver decoding a different message from the original) is no greater than $2^{nR}\ 2^{-n(1-H(p))} = 2^{n(R-C)} = 2^{-\gamma n}$. So by choosing $n$ large we can get rate $R$ close to $C$ and a small probability of error. ■

Shannon's theorem showed that a random code is "good" with high probability. There are still some problems remaining:

- How do we pick the code? $2^{nR}$ is an exponential number of words.
- Decoding is very inefficient.

# 5 Next Lecture

In the next lecture we will see:

- Explicit code constructions
- Efficient encoding and decoding
- Rate approaching the capacity of the channel

# 6 References

[1] M. Mitzenmacher, “Compressed Bloom Filters”.