COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2023.

Lecture 7

Summary

Last Class:

- Finish up exponential concentration bounds. Application to max load in hashing/load balancing.
- Bloom filters for storing a set with a small false positive rate.

This Class:

- Bloom Filter Analysis.
- Start on streaming algorithms
- The distinct items problem via random hashing.

Quiz

- Average time spent on homework: 18-20 hours.
- 18 people worked alone, 103 worked in groups. Mix of approaches to splitting up work in groups.

$\mathbf{X}$ is the sum of independent random variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$, each with mean μ_i and variance σ_i. Each $\mathbf{X}_i$ takes on values in the range $[-5, 5]$.

Which of the following concentration bounds can you apply to show that $\mathbf{X}$ lies close to its expectation with good probability? Check all that apply.

Select one or more:

- ☐ a. Markov's inquality.
- ☐ b. Chebyshev's inequality
- ☐ c. Bernstein's inequality.
- ☐ d. Chernoff bound.

Check

Question 2

Not complete

Points out of 1.00

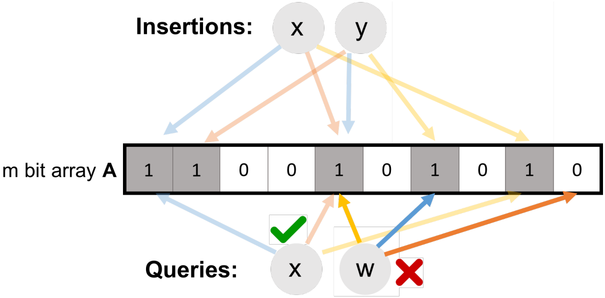
Flag question

Edit question

Bloom Filters

Chose k independent random hash functions $\mathbf{h}_1, \ldots, \mathbf{h}_k$ mapping the universe of elements $U \rightarrow [m]$.

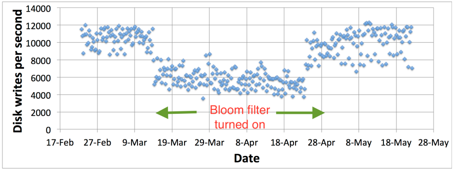
- Maintain an array A containing m bits, all initially 0.
- *insert*(x): set all bits $A[\mathbf{h}_1(x)] = \ldots = A[\mathbf{h}_k(x)] := 1$.
- *query*(x): return 1 only if $A[\mathbf{h}_1(x)] = \ldots = A[\mathbf{h}_k(x)] = 1$.



No false negatives. False positives more likely with more insertions.

Applications: Caching

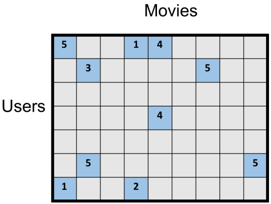
Akamai (Boston-based company serving 15 – 30% of all web traffic) applies bloom filters to prevent caching of 'one-hit-wonders' – pages only visited once fill over 75% of cache.



- When url x comes in, if $query(x) = 1$, cache the page at x. If not, run $insert(x)$ so that if it comes in again, it will be cached.
- **False positive:** A new url (possible one-hit-wonder) is cached. If the bloom filter has a false positive rate of $\delta = .05$, the number of cached one-hit-wonders will be reduced by at least 95%.

Applications: Databases

Distributed database systems, including Google Bigtable, Apache HBase, Apache Cassandra, and PostgreSQL use bloom filters to prevent expensive lookups of non-existent data.



- When a new rating is inserted for $(user_x, movie_y)$, add $(user_x, movie_y)$ to a bloom filter.
- Before reading $(user_x, movie_y)$ (possibly via an out of memory access), check the bloom filter, which is stored in memory.
- **False positive:** A read is made to a possibly empty cell. A $\delta = .05$ false positive rate gives a 95% reduction in these empty reads.

More Applications

- Database Joins: Quickly eliminate most keys in one column that don't correspond to keys in another.
- Recommendation systems: Bloom filters are used to prevent showing users the same recommendations twice.
- Spam/Fraud Detection:
  - Bit.ly and Google Chrome use bloom filters to quickly check if a url maps to a flagged site and prevent a user from following it.
  - Can be used to detect repeat clicks on the same ad from a single IP-address, which may be the result of fraud.
- Digital Currency: Some Bitcoin clients use bloom filters to quickly pare down the full transaction log to transactions involving bitcoin addresses that are relevant to them (SPV: simplified payment verification).

Bloom Filter Quiz Question

Consider a bloom filter where exactly 1/2 of the bits in the filter are set to 1, and the rest are set to 0. Consider running query(w) for some w that has not been inserted into the filter. If my implementation uses k independent, fully random hash functions, for k = 2, what is the probability at query(w) yields a false positive? Give your answer as an exact decimal number.

Answer:

Check

Analysis

For a bloom filter with m bits and k hash functions, the insertion and query time is $O(k)$. How does the false positive rate δ depend on m, k, and the number of items inserted?

Step 1: What is the probability that after inserting n elements, the i^{th} bit of the array A is still 0? $n \times k$ total hashes must not hit bit i.

$$\begin{aligned}
\Pr(A[i] = 0) &= \Pr\left(\mathbf{h}_1(x_1) \neq i \cap \ldots \cap \mathbf{h}_k(x_k) \neq i \right. \\
&\qquad\qquad \left. \cap\, \mathbf{h}_1(x_2) \neq i \ldots \cap \mathbf{h}_k(x_2) \neq i \cap \ldots \right) \\
&= \underbrace{\Pr\left(\mathbf{h}_1(x_1) \neq i\right) \times \ldots \times \Pr\left(\mathbf{h}_k(x_1) \neq i\right) \times \Pr\left(\mathbf{h}_1(x_2) \neq i\right) \ldots}_{k \cdot n \text{ events each occuring with probability } 1 - 1/m} \\
&= \left(1 - \frac{1}{m}\right)^{kn}
\end{aligned}$$

Analysis

How does the false positive rate δ depend on m, k, and the number of items inserted?

Step 1: What is the probability that after inserting n elements, the i^{th} bit of the array A is still 0?

$$\Pr(A[i] = 0) = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

Step 2: What is the probability that querying a new item w gives a false positive?

$$\begin{aligned}
\Pr\left(A[\mathbf{h}_1(w)] = \ldots = A[\mathbf{h}_k(w)] = 1\right) \\
&= \Pr(A[\mathbf{h}_1(w)] = 1) \times \ldots \times \Pr(A[\mathbf{h}_k(w)] = 1) \\
&= \left(1 - e^{-\frac{kn}{m}}\right)^k
\end{aligned}$$

Actually Incorrect! Dependent events.

n: total number items in filter, m: number of bits in filter, k: number of random hash functions, $\mathbf{h}_1, \ldots \mathbf{h}_k$: hash functions, A: bit array, δ: false positive rate.

Correct Analysis Sketch

Step 1: To avoid dependence issues, condition on the event that the A has t zeros in it after n insertions, for some $t \leq m$. For a non-inserted element w, after conditioning on this event we correctly have:

$$\begin{aligned}\Pr(A[\mathbf{h}_1(w)] = \ldots = A[\mathbf{h}_k(w)] = 1) \\ = \Pr(A[\mathbf{h}_1(w)] = 1) \times \ldots \times \Pr(A[\mathbf{h}_k(w)] = 1).\end{aligned}$$

I.e., the events $A[\mathbf{h}_1(w)] = 1$,..., $A[\mathbf{h}_k(w)] = 1$ are independent conditioned on the number of bits set in A. Why?

- Conditioned on this event, for any j, since $\mathbf{h}_j$ is a fully random hash function, $\Pr(A[\mathbf{h}_j(w)] = 1) = 1 - \frac{t}{m}$.
- Thus conditioned on this event, the false positive rate is $\left(1 - \frac{t}{m}\right)^k$.
- It remains to show that $\frac{t}{m} \approx e^{-\frac{kn}{m}}$ with high probability. We already have that $\mathbb{E}[\frac{t}{m}] = \frac{1}{m}\sum_{i=1}^{m} \Pr(A[i] = 0) \approx e^{-\frac{kn}{m}}$.

Correct Analysis Sketch

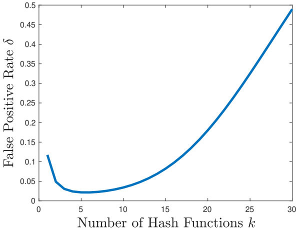
Need to show that the number of zeros t in A after n insertions is bounded by $O\left(e^{-\frac{kn}{m}}\right)$ with high probability.

Can apply Theorem 2 of:

`http://cglab.ca/~morin/publications/ds/bloom-submitted.pdf`

False Positive Rate

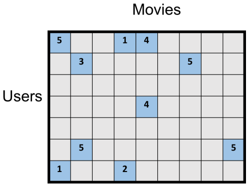
False Positive Rate: with m bits of storage, k hash functions, and n items inserted $\delta \approx \left(1 - e^{\frac{-kn}{m}}\right)^k$. How should we set k to minimize the FPR given a fixed amount of space m?



- Can differentiate to show optimal number of hashes is $k = \ln 2 \cdot \frac{m}{n}$.
- Balances filling up the array vs. having enough hashes so that even when the array is pretty full, a new item is unlikely to yield a false positive.

False Positive Rate

False Positive Rate: with m bits of storage, k hash functions, and n items inserted $\delta \approx \left(1 - e^{\frac{-kn}{m}}\right)^k$.



- Say we have 100 million users, each who have rated 10 movies.
- $n = 10^9 = n$ (user,movie) pairs with non-empty ratings.
- Allocate $m = 8n = 8 \times 10^9$ bits for a Bloom filter (1 GB).
- Set $k = \ln 2 \cdot \frac{m}{n} = 5.54 \approx 6$.
- False positive rate is $\approx \left(1 - e^{-k \cdot \frac{n}{m}}\right)^k \approx \frac{1}{2^k} \approx \frac{1}{2^{5.54}} = .021$.

Bloom Filter Note

An observation about Bloom filter space complexity:

$$\text{False Positive Rate: } \delta \approx \left(1 - e^{-\frac{kn}{m}}\right)^k.$$

For an m-bit bloom filter holding n items, optimal number of hash functions k is: $k = \ln 2 \cdot \frac{m}{n}$.

Think Pair Share: If we want a false positive rate $< \frac{1}{2}$ how big does m need to be in comparison to n?

$$m = O(\log n),\ m = O(\sqrt{n}),\ m = O(n),\ m = O(n^2)?$$

If $m = \frac{n}{\ln 2}$, optimal $k = 1$, and failure rate is:

$$\delta = \left(1 - e^{-\frac{n/\ln 2}{n}}\right)^1 = \left(1 - \frac{1}{2}\right)^1 = \frac{1}{2}.$$

I.e., storing n items in a bloom filter requires $O(n)$ space. So what's the point? Truly $O(n)$ bits, rather than $O(n \cdot \text{item size})$.

Questions on Bloom Filters?