COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.

Lecture 2

By Next Thursday 9/12:

- Sign up for Piazza.
- Pick a problem set group with 3 people and have one member email me the names of the members and a group name.
- Fill out the Gradescope consent poll on Piazza and contact me via email if you don't consent.

Last Class We Covered:

- Linearity of expectation: $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ always.
- Linearity of variance: $\text{Var}[X+Y] = \text{Var}[X] + \text{Var}[Y]$ if X and Y are independent.
- Markov's inequality: a non-negative random variable with a small expectation is unlikely to be very large:

$$\Pr(X \geq t) \leq \frac{\mathbb{E}[X]}{t}.$$

- Talked about an application to estimating the size of a CAPTCHA database efficiently.

Today: We'll see how a simple twist on Markov's inequality can give much stronger bounds.

- Enough to prove a version of the law of large numbers.

But First: Another example of how powerful linearity of expectation and Markov's inequality can be in randomized algorithm design.

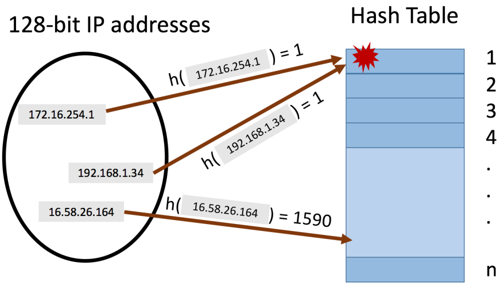
- Will learn about random hash functions, which are a key tool in randomized methods for data processing.

Want to store a set of items from some finite but massive universe of items (e.g., images of a certain size, text documents, 128-bit IP addresses).

Goal: support *query*(x) to check if x is in the set in $O(1)$ time.

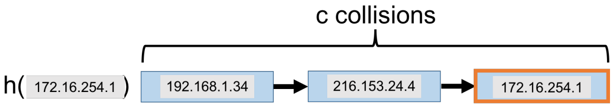
Classic Solution: Hash tables

- *Static hashing* since we won't worry about insertion and deletion today.



- **hash function** $h : U \to [n]$ maps elements from the universe to indices $1, \cdots, n$ of an array.
- Typically $|U| \gg n$. Many elements map to the same index.
- **Collisions:** when we insert m items into the hash table we may have to store multiple items in the same location (typically as a linked list).

Query runtime: $O(c)$ when the maximum number of collisions in a table entry is c (i.e., must traverse a linked list of size c).



How Can We Bound c**?**

- In the worst case could have $c = m$ (all items hash to the same location).
- Two approaches: 1) we assume the items inserted are chosen randomly from the universe U or 2) the hash function is chosen randomly.

Let $h : U \to [n]$ be a random hash function.

- I.e., for $x \in U$, $\Pr(h(x) = i) = \frac{1}{n}$ for all $i = 1, \ldots, n$ and $h(x), h(y)$ are independent for any two items $x \neq y$.
- **Caveat:** It is *very expensive* to represent and compute such a random function. We will see how a hash function computable in $O(1)$ time function can be used instead.

Assuming we insert m elements into a hash table of size n, what is the expected total number of pairwise collisions?

Let $C_{i,j} = 1$ if items i and j collide ($h(x_i) = h(x_j)$), and 0 otherwise. The number of pairwise duplicates is:

$$\mathbb{E}[C] = \sum_{i,j} \mathbb{E}[C_{i,j}]. \qquad \text{(linearity of expectation)}$$

For any pair i, j: $\mathbb{E}[C_{i,j}] = \Pr[C_{i,j} = 1] = \Pr[h(x_i) = h(x_j)] = \frac{1}{n}$.

$$\mathbb{E}[C] = \sum_{i,j} \frac{1}{n} = \frac{\binom{m}{2}}{n} = \frac{m(m-1)}{2n}.$$

Identical to the CAPTCHA analysis from last class!

> x_i, x_j: pair of stored items, m: total number of stored items, n: hash table size, **C**: total pairwise collisions in table, **h**: random hash function.

$$\mathbb{E}[C] = \frac{m(m-1)}{2n}.$$

- For $n = 4m^2$ we have: $\mathbb{E}[C] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$.
- Can you give a lower bound on the probability that we have no collisions, i.e., $\Pr[C = 0]$?

Apply Markov's Inequality: $\Pr[C \geq 1] \leq \frac{\mathbb{E}[C]}{1} = \frac{1}{8}$.

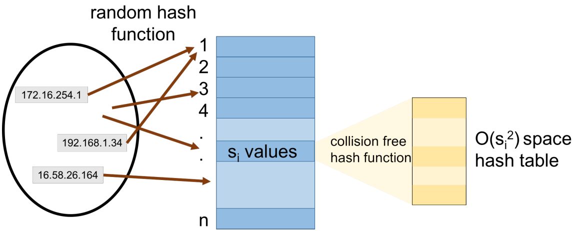
$$\Pr[C = 0] = 1 - \Pr[C \geq 1] \geq 1 - \frac{1}{8} = \frac{7}{8}.$$

Pretty good...but we are using $O(m^2)$ space to store m items.

> m: total number of stored items, n: hash table size, **C**: total pairwise collisions in table.

Want to preserve $O(1)$ query time while using $O(m)$ space.

Two-Level Hashing:



- For each bucket with s_i values, pick a collision free hash function mapping $[s_i] \to [s_i^2]$.
- **Just Showed:** A random function is collision free with probability $\geq \frac{7}{8}$ so only requires checking $O(1)$ random functions in expectation to find a collision free one.

Query time for two level hashing is $O(1)$: requires evaluating two hash functions. What is the expected space usage?

Up to constants, space used is: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

$$\mathbb{E}[s_i^2] = \mathbb{E}\left[\left(\sum_{j=1}^{m} \mathbb{I}_{h(x_j)=i}\right)^2\right]$$

$$= \mathbb{E}\left[\sum_{j,k} \mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right]$$

Collisions again!

x_j, x_k: stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored in hash table at position i.

Query time for two level hashing is $O(1)$: requires evaluating two hash functions. What is the expected space usage?

Up to constants, space used is: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

$$\mathbb{E}[s_i^2] = \mathbb{E}\left[\left(\sum_{j=1}^{m} \mathbb{I}_{h(x_j)=i}\right)^2\right]$$

$$= \mathbb{E}\left[\sum_{j,k} \mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \sum_{j,k} \mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right].$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \mathbb{E}\left[\left(\mathbb{I}_{h(x_j)=i}\right)^2\right] = \Pr[h(x_j) = i] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \Pr[h(x_j) = i \cap h(x_k) = i] = \frac{1}{n^2}$.

x_j, x_k: stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored in hash table at position i.

SPACE USAGE

$$\mathbb{E}[s_i^2] = \sum_{j,k} \mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right]$$
$$= m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2}$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n^2}$.

x_j, x_k: stored items, m: # stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored at pos i.

SPACE USAGE

$$\mathbb{E}[s_i^2] = \sum_{j,k} \mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right]$$
$$= m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2}$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n^2}$.

x_j, x_k: stored items, m: # stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored at pos i.

SPACE USAGE

$$\mathbb{E}[s_i^2] = \sum_{j,k} \mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right]$$
$$= m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2}$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n^2}$.

x_j, x_k: stored items, m: # stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored at pos i.

SPACE USAGE

$$\mathbb{E}[s_i^2] = \sum_{j,k} \mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right]$$
$$= m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2}$$
$$= \frac{m}{n} + \frac{m(m-1)}{n^2} \leq 2 \text{ (If we set } n = m.)$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i}\right] = \frac{1}{n^2}$.

Total Expected Space Usage: (if we set $n = m$)

$$\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2] \leq n + n \cdot 2 = 3n = 3m.$$

Near optimal space with $O(1)$ query time!

x_j, x_k: stored items, m: # stored items, n: hash table size, **h**: random hash function, **S**: space usage of two level hashing, $\mathbf{s}_i$: # items stored at pos i.

What if we want to store a set and answer membership queries in $O(1)$ time. But we allow a small probability of a false positive: $query(x)$ says that x is in the set when in fact it isn't.

Can we do better than $O(m)$ space?

Many Applications:

- Filter spam email addresses, phone numbers, suspect IPs, duplicate Tweets.
- Quickly check if an item has been stored in a cache or is new.
- Counting distinct elements (e.g., unique search queries.)

So Far: we have assumed a **fully random hash function** $h(x)$ with $\Pr[h(x) = i] = \frac{1}{n}$ for $i \in 1, \ldots, n$ and $h(x), h(y)$ independent for $x \neq y$.

- To store a random hash function we have to store a table of x values and their hash values. Would take at least $O(m)$ space and $O(m)$ query time if we hash m values. Making our whole quest for $O(1)$ query time pointless!

x	**h(x)**
x_1	45
x_2	1004
x_3	10
$\vdots$	$\vdots$
x_m	12

What properties did we use of the randomly chosen hash function?

2-Universal Hash Function (low collision probability). A random hash function from $h : U \rightarrow [n]$ is two universal if:

$$\Pr[h(x) = h(y)] \leq \frac{1}{n}.$$

Exercise: Rework the two level hashing proof to show that this property is really all that is needed.

When $h(x)$ and $h(y)$ are chosen independently at random from $[n]$, $\Pr[h(x) = h(y)] = \frac{1}{n}$.

Efficient Alternative: Let p be a prime with $p \geq |U|$. Choose random $a, b \in [p]$ with $a \neq 0$. Let:

$$h(x) = (ax + b \mod p) \mod n.$$

Another common requirement for a hash function:

Pairwise Independent Hash Function. A random hash function from $h : U \to [n]$ is pairwise independent if for all $i \in [n]$:

$$\Pr[h(x) = h(y) = i] = \frac{1}{n^2}.$$

Which is a more stringent requirement? 2-universal or **pairwise independent**?

$$\Pr[h(x) = h(y)] = \sum_{i=1}^{n} \Pr[h(x) = h(y) = i] = n \cdot \frac{1}{n^2} = \frac{1}{n}.$$

A closely related $(ax + b) \mod p$ construction gives pairwise independence on top of 2-universality.

Another common requirement for a hash function:

k-wise Independent Hash Function. A random hash function from $h : U \to [n]$ is k-wise independent if for all $i \in [n]$:

$$\Pr[h(x_1) = h(x_2) = \ldots = h(x_k) = i] = \frac{1}{n^k}.$$

Which is a more stringent requirement? 2-universal or **pairwise independent**?

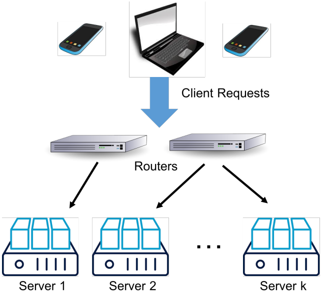
$$\Pr[h(x) = h(y)] = \sum_{i=1}^{n} \Pr[h(x) = h(y) = i] = n \cdot \frac{1}{n^2} = \frac{1}{n}.$$

A closely related $(ax + b) \mod p$ construction gives pairwise independence on top of 2-universality.

Questions on linearity of expectation/variance, Markov's, hashing?

1. We'll consider an application where our toolkit of linearity of expectation + Markov's inequality doesn't give much.
2. Then we'll show how a simple twist on Markov's can give a much stronger result.

Randomized Load Balancing:



Simple Model: n requests randomly assigned to k servers. How many requests must each server handle?

- Often assignment is done via a random hash function. Why?

Expected Number of requests assigned to server i:

$$\mathbb{E}[R_i] = \sum_{j=1}^{n} \mathbb{E}[\mathbb{I}_{\text{request } j \text{ assigned to } i}] = \sum_{j=1}^{n} \Pr[j \text{ assigned to } i] = \frac{n}{k}.$$

If we provision each server be able to handle twice the expected load, what is the probability that a server is overloaded?

Applying Markov's Inequality

$$\Pr[R_i \geq 2\mathbb{E}[R_i]] \leq \frac{\mathbb{E}[R_i]}{2\mathbb{E}[R_i]} = \frac{1}{2}.$$

Not great...half the servers may be overloaded.

n: total number of requests, k: number of servers randomly assigned requests, R_i: number of requests assigned to server i.

With a very simple twist Markov's Inequality can be made much more powerful.

For any random variable X and any value t:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

X^2 is a nonnegative random variable. So can apply Markov's inequality:

Chebyshev's inequality:

$$\Pr(|X| \geq t) \leq \frac{\mathbb{E}[X^2]}{t^2}.$$

With a very simple twist Markov's Inequality can be made much more powerful.

For any random variable X and any value t:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

X^2 is a nonnegative random variable. So can apply Markov's inequality:

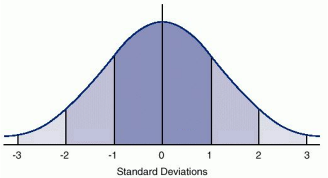
Chebyshev's inequality:

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\mathrm{Var}[X]}{t^2}.$$

(by plugging in the random variable $X - \mathbb{E}[X]$)

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\mathrm{Var}[X]}{t^2}$$

What is the probability that X falls s standard deviations from it's mean?



$$\Pr(|X - \mathbb{E}[X]| \geq s \cdot \sqrt{\mathrm{Var}[X]}) \leq \frac{\mathrm{Var}[X]}{s^2 \cdot \mathrm{Var}[X]} = \frac{1}{s^2}.$$

Why is this so powerful?

X: any random variable, t, s: any fixed numbers.

Consider drawing independent identically distributed (i.i.d.) random variables $X_1, \ldots, X_n$ with mean μ and variance σ^2.

How well does the sample average $S = \frac{1}{n}\sum_{i=1}^{n} X_i$ approximate the true mean μ?

$$\mathrm{Var}[S] = \frac{1}{n^2}\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}\left[X_i\right] = \frac{1}{n^2}\cdot n \cdot \sigma^2 = \frac{\sigma^2}{n}.$$

By Chebyshev's Inequality: for any fixed value $\epsilon > 0$,

$$\Pr(|S-\mu| \geq \epsilon) \leq \frac{\mathrm{Var}[S]}{\epsilon^2} = \frac{\sigma^2}{n\epsilon^2}.$$

Law of Large Numbers: with enough samples, the sample average will always concentrate to the mean.

- Cannot show from vanilla Markov's inequality.

Recall that R_i is the load on server i when n requests are randomly assigned to k servers.

$$R_i = \sum_{j=1}^{n} R_{i,j}$$

where $R_{i,j}$ is 1 if request j is assigned to server i and 0 o.w.

$$\begin{aligned}
\text{Var}[R_{i,j}] &= \mathbb{E}\left[\left(R_{i,j} - \mathbb{E}[R_{i,j}]\right)^2\right] \\
&= \Pr(R_{i,j} = 1) \cdot \left(1 - \mathbb{E}[R_{i,j}]\right)^2 + \Pr(R_{i,j} = 0) \cdot \left(0 - \mathbb{E}[R_{i,j}]\right)^2 \\
&= \frac{1}{k} \cdot \left(1 - \frac{1}{k}\right)^2 + \left(1 - \frac{1}{k}\right) \cdot \left(0 - \frac{1}{k}\right)^2 \\
&= \frac{1}{k} - \frac{1}{k^2} \leq \frac{1}{k} \implies \text{Var}[R_i] \leq \frac{n}{k}.
\end{aligned}$$

Applying Chebyshev's:

$$\Pr\left(R_i \geq \frac{2n}{k}\right) \leq \Pr\left(|R_i - \mathbb{E}[R_i]| \geq \frac{n}{k}\right) \leq \frac{n/k}{n^2/k^2} = \frac{k}{n}.$$

Overload probability is extremely small when $k \ll n$!

Provisioning each server with twice the expected necessary capacity ($\frac{2n}{k}$ vs. $\frac{n}{k}$) is really expensive.

If we give each server the capacity to serve $(1+\delta)\cdot\frac{n}{k}$ requests for $\delta \in (0,1)$, what is the probability that a server exceeds its capacity?

$$\mathbb{E}[R_i] = \frac{n}{k} \text{ and } \operatorname{Var}[R_i] \leq \frac{n}{k}.$$

Chebyshev's Inequality:

$$\Pr\left(|X - \mathbb{E}[X]| \geq \epsilon\right) \leq \frac{\operatorname{Var}[X]}{\epsilon^2}.$$

Bonus: What if requests are assigned to servers with a 2-universal hash function? With a pairwise independent hash function?

n: total number of requests, k: number of servers randomly assigned requests, R_i: number of requests assigned to server i. δ, ϵ any values.

If we give each server the capacity to serve $(1+\delta)\cdot\frac{n}{k}$ requests for $\delta\in(0,1)$, what is the probability that a server exceeds its capacity?

$$\mathbb{E}[R_i]=\frac{n}{k} \text{ and } \mathrm{Var}[R_i]\leq\frac{n}{k}.$$

Chebyshev's Inequality:

$$\Pr\left(|X-\mathbb{E}[X]|\geq\epsilon\right)\leq\frac{\mathrm{Var}[X]}{\epsilon^2}.$$

$$\Pr\left(R_i\geq(1+\delta)\cdot\frac{n}{k}\right)\leq\Pr\left(|R_i-\mathbb{E}[R_i]|\geq\delta\cdot\frac{n}{k}\right)\leq\frac{\mathrm{Var}[R_i]}{\delta^2\cdot n^2/k^2}=\frac{k}{\delta^2 n}.$$

Can set $\delta=O\left(\sqrt{\frac{k}{n}}\right)$ and still have a pretty good probability that a server won't be overloaded.

n: total number of requests, k: number of servers randomly assigned requests, R_i: number of requests assigned to server i.

Bonus: What if requests are assigned to servers with a 2-universal hash function? With a pairwise independent hash function?

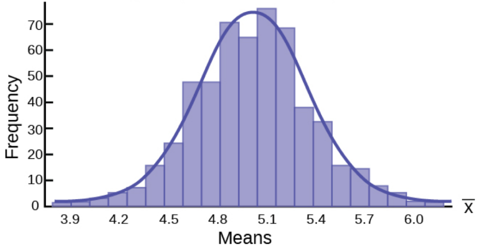
- To apply Chebyshev's need to bound

$$\text{Var}[R_i] = \mathbb{E}[R_i^2] - \mathbb{E}[R_i]^2 \leq \mathbb{E}[R_i^2].$$

- With pairwise independence can apply a similar technique as we did to bounding the expected second level table size for two level hashing, showing $\text{Var}[R_i] = O\left(\frac{n}{k}\right)$.
- Will see that 2-universal hashing is not strong enough here!

Chebyshev's Inequality: A quantitative version of the law of large numbers. The average of many independent random variables concentrates around its mean.

Chernoff Type Bounds: A quantitative version of the central limit theorem. The average of many independent random variables is distributed like a Gaussian.



Questions?