COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Spring 2020.
Lecture 2

By Next Thursday 1/30:

- Sign up for Piazza.
- Sign up for Gradescope (code on class website) and fill out the Gradescope consent poll on Piazza. Contact me via email if you don't consent to use Gradescope.

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

Today:

- Finish up the CAPTCHA example and introduce Markov's inequality a fundamental concentration bound that let us prove that a random variable lies close to its expectation with good probability.
- Learn about random hash functions, which are a key tool in randomized methods for data processing. Probabilistic analysis via linearity of expectation.
- Start on Chebyshev's inequality: a concentration bound that is enough to prove a version of the law of large numbers.

CAPTCHA REFRESH

Your CAPTCHA provider claims to have a database of $n = 1,000,000$ CAPTCHAS, with a random one selected for each security check.

- In an attempt to verify this claim, you make m random security checks. If the database size is n then expected number of pairwise duplicate CAPTCHAS you see is:

$$\mathbb{E}[\mathbf{D}] = \sum_{i,j \in [m], i \neq j} \mathbb{E}[\mathbf{D}_{i,j}] = \frac{m(m-1)}{2n}.$$

If the database size is as claimed ($n = 1,000,000$) and you take $m = 1,000$ samples:

$$\mathbb{E}[\mathbf{D}] = \frac{m(m-1)}{2n} = .4995$$

You see 10 pairwise duplicates and suspect that something is up. But how confident can you be in your test?

Concentration Inequalities: Bounds on the probability that a random variable deviates a certain distance from its mean.

- Useful in understanding how statistical tests perform, the behavior of randomized algorithms, the behavior of data drawn from different distributions, etc.

n: number of CAPTCHAS in database, m: number of random CAPTCHAS drawn to check database size, $\mathbf{D}$: number of pairwise duplicates in m random CAPTCHAS.

The most fundamental concentration bound: **Markov's inequality.**

For any non-negative random variable X and any $t > 0$:

$$\Pr[X \geq t t \cdot \mathbb{E}[X]] \leq \frac{\mathbb{E}[X]}{t} \frac{1}{t}.$$

Proof:

$$\begin{aligned} \mathbb{E}[X] = \sum_{s} \Pr(X = s) \cdot s &\geq \sum_{s \geq t} \Pr(X = s) \cdot s \\ &\geq \sum_{s \geq t} \Pr(X = s) \cdot t \\ &= t \cdot \Pr(X \geq t). \end{aligned}$$

The larger the deviation t, the smaller the probability.

Expected number of duplicate CAPTCHAS:

$\mathbb{E}[\mathbf{D}] = \frac{m(m-1)}{2n} = .4995.$

You see $\mathbf{D} = 10$ duplicates.

Applying Markov's inequality, if the real database size is $n = 1,000,000$ the probability of this happening is:

$$\Pr[\mathbf{D} \geq 10] \leq \frac{\mathbb{E}[\mathbf{D}]}{10} = \frac{.4995}{10} \approx .05$$

This is pretty small – you feel pretty sure the number of unique CAPTCHAS is much less than $1,000,000$. But how can you boost your confidence? We'll discuss later this class.

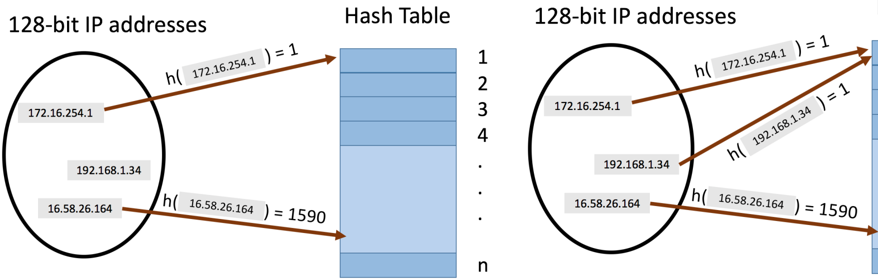
n: number of CAPTCHAS in database ($n = 1,000,000$ claimed) , m: number of random CAPTCHAS drawn to check database size ($m = 1000$ in this example), $\mathbf{D}$: number of pairwise duplicates in m random CAPTCHAS.

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.

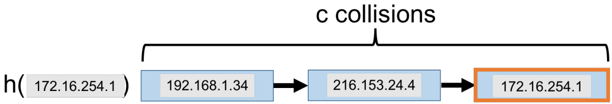
Classic Solution: Hash tables

- *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 random.

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

- I.e., for $x \in U$, $\Pr(\mathbf{h}(x) = i) = \frac{1}{n}$ for all $i = 1, \ldots, n$ and $\mathbf{h}(x), \mathbf{h}(y)$ are independent for any two items $x \neq y$.
- **Caveat 1:** 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.
- **Caveat 2:** In practice, often suffices to use hash functions like MD5, SHA-2, etc. that 'look random enough'.

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

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

$$\mathsf{C} = \sum_{i,j \in [m], i \neq j} \mathsf{C}_{i,j} . \mathbb{E}[\mathsf{C}] = \sum_{i,j \in [m], i \neq j} \mathbb{E}[\mathsf{C}_{i,j}].$$

(linearity of expectation)

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

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

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.

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

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

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

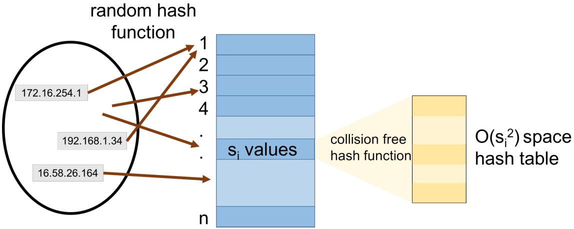
$$\Pr[\mathbf{C} = 0] = 1 - \Pr[\mathbf{C} \geq 1] \geq 1 - \frac{1}{8} = \frac{7}{8}.$$

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

m: total number of stored items, n: hash table size, $\mathbf{C}$: 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 $[s_i] \rightarrow [s_i^2]$.
- **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.

SPACE USAGE

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: $\mathbf{S} = n + \sum_{i=1}^{n} \mathbf{s}_i^2 \mathbb{E}[\mathbf{S}] = n + \sum_{i=1}^{n} \mathbb{E}[\mathbf{s}_i^2]$

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

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

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

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

SPACE USAGE

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

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

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

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

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

x_j, x_k: stored items, m: # stored items, n: hash table size, $\mathbf{h}$: random hash function, $\mathbf{S}$: 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 use even smaller 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** $\mathbf{h}(x)$ with $\Pr[\mathbf{h}(x) = i] = \frac{1}{n}$ for $i \in 1, \ldots, n$ and $\mathbf{h}(x), \mathbf{h}(y)$ independent for $x \neq y$.

- To compute 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 $\mathbf{h} : U \to [n]$ is two universal if:

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

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

When $\mathbf{h}(x)$ and $\mathbf{h}(y)$ are chosen independently at random from $[n]$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y)] = \frac{1}{n}$ (so a fully random hash function is 2-universal)

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

$$\mathbf{h}(x) = (\mathbf{a}x + \mathbf{b} \mod p) \mod n.$$

Another common requirement for a hash function:

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

$$\Pr[\mathbf{h}(x) = \mathbf{h}(y) = i] = \frac{1}{n^2}. \Pr[\mathbf{h}(x_1) = \mathbf{h}(x_2) = \ldots = \mathbf{h}(x_k) = i] = \frac{1}{n^k}.$$

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

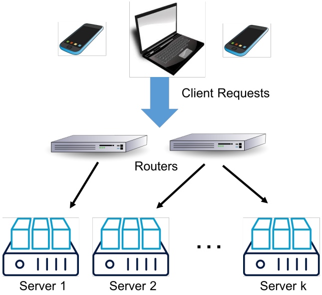
$$\Pr[\mathbf{h}(x) = \mathbf{h}(y)] = \sum_{i=1}^{n} \Pr[\mathbf{h}(x) = \mathbf{h}(y) = i] = n \cdot \frac{1}{n^2} = \frac{1}{n}.$$

A closely related $(\mathbf{a}x + \mathbf{b}) \mod p$ construction gives pairwise independence on top of 2-universality.

Questions on linearity of expectation, 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 $t > 0$:

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

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

Chebyshev's inequality:

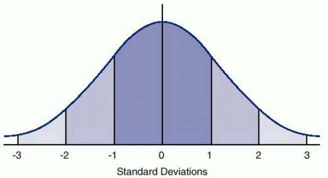
$$\Pr(|X - \mathbb{E}[X]| \geq t) \quad \Pr(|X| \geq t) = \Pr(X^2 \geq t^2) \leq \frac{\mathbb{E}[X^2]}{t^2} \quad \frac{\mathrm{Var}[X]}{t^2}.$$

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

CHEBYSHEV'S INEQUALITY

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

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



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

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 $\epsilon > 0$,

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

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

- Cannot show from vanilla Markov's inequality.

Recall that $\mathbf{R}_i$ is the load on server i when n requests are randomly assigned to k servers.

$$\mathbf{R}_i = \sum_{j=1}^{n} \mathbf{R}_{i,j} \quad \mathrm{Var}[\mathbf{R}_i] = \sum_{j=1}^{n} \mathrm{Var}[\mathbf{R}_{i,j}]$$

where $\mathbf{R}_{i,j}$ is 1 if request j is assigned to server i and 0 o.w.

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

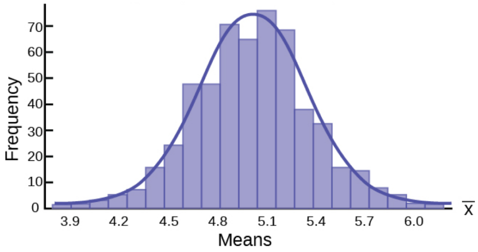
Applying Chebyshev's:

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

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

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?