COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor

Lecture 13

SUMMARY OF FIRST SECTION

WHAT WE'VE COVERED

- **Probability Tools:** Linearity of Expectation, Linear of Variance of Independent Variables, Concentration Bounds (incl. Markov, Chebyshev, Bernstein, Chernoff), Union Bound, Median Trick.
- **Hash Tables and Bloom Filters:** Analyzing collisions. Building 2-level hash tables. Bloom filters and false positive rates.
- **Locality Sensitive Hashing:** MinHash for Jaccard Similarity and SimHash for Cosine Similarity. Nearest Neighbor. All-Pairs Similarity Search.
- **Small Space Data Stream Algorithms:** a) distinct items, b) frequent elements, c) frequency moments (homework).
- **Johnson Lindenstrauss Lemma:** Reducing dimension of vectors via random projection such that pairwise distances are approximately preserved.

RANDOMIZED ALGORITHMS UNIT TAKEAWAYS

- Randomization is an important tool in working with large datasets.
- Lets us solve 'easy' problems that get really difficult on massive datasets. Fast/space efficient look up (hash tables and bloom filters), distinct items counting, frequent items counting, near neighbor search (LSH), etc.
- The analysis of randomized algorithms leads to complex output distributions, which we can't compute exactly.
- We've covered many of the key ideas used through a small number of example applications/algorithms.
- We use concentration inequalities to bound these distributions and behaviors like accuracy, space usage, and runtime.
- Concentration inequalities and probability tools used in randomized algorithms are also fundamental in statistics, machine learning theory, probabilistic modeling of complex systems, etc.

USEFUL PROBABILITY FACTS (1/2)

- Linearity of Expectation: For any random variables $X_1, \ldots, X_n$ and constants $c_1, \ldots, c_n$,

$$\mathbb{E}[c_1 X_1 + \ldots + c_n X_n] = c_1 \mathbb{E}[X_1] + \ldots + c_n \mathbb{E}[X_n]$$

USEFUL PROBABILITY FACTS (1/2)

- Linearity of Expectation: For any random variables $X_1, \ldots, X_n$ and constants $c_1, \ldots, c_n$,

$$\mathbb{E}[c_1 X_1 + \ldots + c_n X_n] = c_1 \mathbb{E}[X_1] + \ldots + c_n \mathbb{E}[X_n]$$

- Independent Random Variables: $X_1, X_2, \ldots X_n$ are independent random variables if for any set $S \subset [n]$ and values $a_1, a_2, \ldots, a_n$

$$\Pr(X_i = a_i \text{ for all } i \in S) = \prod_{i \in S} \Pr(X_i = a_i) \ .$$

They are k-wise independent if this holds for S with $|S| \leq k$.

USEFUL PROBABILITY FACTS (1/2)

- Linearity of Expectation: For any random variables $X_1, \dots, X_n$ and constants $c_1, \dots, c_n$,

$$\mathbb{E}[c_1 X_1 + \ldots + c_n X_n] = c_1 \mathbb{E}[X_1] + \ldots + c_n \mathbb{E}[X_n]$$

- Independent Random Variables: $X_1, X_2, \dots X_n$ are independent random variables if for any set $S \subset [n]$ and values $a_1, a_2, \dots, a_n$

$$\Pr(X_i = a_i \text{ for all } i \in S) = \prod_{i \in S} \Pr(X_i = a_i) \; .$$

  They are k-wise independent if this holds for S with $|S| \le k$.

- Linearity of Variance: If$X_1, \dots, X_n$ are independent (in fact 2-wise independent suffices) then for any constants $c_1, \dots, c_n$

$$\mathrm{Var}[c_1 X_1 + \ldots + c_n X_n] = c_1^2 \,\mathrm{Var}[X_1] + \ldots + c_n^2 \,\mathrm{Var}[X_n]$$

USEFUL PROBABILITY FACTS (2/2)

- Union Bound: For any events $A_1, A_2, A_3, \ldots$

$$\Pr\left[\bigcup A_i\right] \leq \sum_i \Pr[A_i] \,.$$

- An indicator random variable X just takes the values 0 or 1:

$$\mathbb{E}[X] = p \qquad \mathrm{Var}[X] = p(1-p) \qquad \text{where } p = \Pr[X=1]$$

- If $Y = X_1 + \ldots + X_n$ where each X_i are independent and $p = \Pr[X_1 = 1] = \ldots = \Pr[X_n = 1]$ then Y is a binomial random variable. Using linearity of expectation and variance,

$$\mathbb{E}[X] = np \qquad \mathrm{Var}[X] = np(1-p)$$

BALLS AND BINS (1/2)

- Most of the analysis of hash functions that we've considered can be abstracted as "balls and bins" problems: we throw n balls and each ball is equally likely to land in one of m bins.
- Let R_i be number of balls bin i. Then $R_i \sim \text{Bin}(n, \frac{1}{m})$ and $\mathbb{E}[R_i] = \frac{n}{m}$, $\text{Var}[R_i] = \frac{n}{m} \cdot (1 - \frac{1}{m})$. R_i and R_j not independent!
- Union Bound implies $\Pr[\max(R_1, \ldots, R_m) > t] \leq \sum_i \Pr[R_i > t]$
- $Pr[\text{no collisions}] = \frac{m-1}{m}\frac{m-2}{m}\ldots\frac{m-(n-1)}{m}$

$$\Pr[\text{collisions}] = \Pr[\max(R_1, \ldots, R_m) > 1] \leq 1/8 \text{ if } m > 4n^2$$

and more generally

$$\Pr[\max(R_1, \ldots, R_m) \geq 2n/m] \leq m^2/n$$

- In the exam, you'll be expected to do calculations like these.

- Let T be the number of bins where $R_i = 0$. We showed:

$$\mathbb{E}[T] = m(1 - 1/m)^n$$

- The probability the next k balls thrown all land in non-empty bins is

$$(1 - 1/T)^k$$

and this lets us analyze the false positive rate of a Bloom filter.

- Hash function $\mathbf{h} : U \rightarrow [n]$ is two universal if:

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

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

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

- Hash function $\mathbf{h} : U \to [n]$ is k-wise independent if $\{h(e)\}_{e \in U}$ are k-wise independent and each $h(e)$ is uniform in $[n]$.

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

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

- Hash function $\mathbf{h} : U \to [n]$ is k-wise independent if $\{h(e)\}_{e \in U}$ are k-wise independent and each $h(e)$ is uniform in $[n]$.
- Hash function $\mathbf{h} : U \to [n]$ is fully independent if $\{h(e)\}_{e \in U}$ are independent and each $h(e)$ is uniform in $[n]$.

- Markov. For any non-negative random variable X and $t > 0$,

$$\Pr[X \geq t] \leq \mathbb{E}[X]/t \ .$$

THREE MAIN CONCENTRATION BOUNDS

- Markov. For any non-negative random variable X and $t > 0$,

$$\Pr[X \geq t] \leq \mathbb{E}[X]/t \ .$$

- Chebyshev. For any random variable X and $t > 0$,

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

THREE MAIN CONCENTRATION BOUNDS

- Markov. For any non-negative random variable X and $t > 0$,

$$\Pr[X \geq t] \leq \mathbb{E}[X]/t \ .$$

- Chebyshev. For any random variable X and $t > 0$,

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

- Chernoff. Let $X_1, \ldots, X_n$ be independent $\{0, 1\}$ random variables with $\mu = \mathbb{E}[\sum_i X_i]$. Then for any $\delta > 0$,

$$\Pr[|(\sum_i X_i) - \mu| \geq \delta\mu] \leq 2 \exp\left(-\frac{\delta^2 \mu}{\delta + 2}\right) \ .$$

- Markov. For any non-negative random variable X and $t > 0$,

$$\Pr[X \geq t] \leq \mathbb{E}[X]/t \ .$$

- Chebyshev. For any random variable X and $t > 0$,

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

- Chernoff. Let $X_1, \ldots, X_n$ be independent $\{0, 1\}$ random variables with $\mu = \mathbb{E}[\sum_i X_i]$. Then for any $\delta > 0$,

$$\Pr[|(\sum_i X_i) - \mu| \geq \delta\mu] \leq 2 \exp\left(-\frac{\delta^2 \mu}{\delta + 2}\right) \ .$$

- Generally, Chernoff gives better results then Chebyshev and Chebyshev gives better results than Markov. So choose bound based on how much you know about X.

THREE MAIN CONCENTRATION BOUNDS

- Markov. For any non-negative random variable X and $t > 0$,

$$\Pr[X \geq t] \leq \mathbb{E}[X]/t \ .$$

- Chebyshev. For any random variable X and $t > 0$,

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

- Chernoff. Let $X_1, \ldots, X_n$ be independent $\{0, 1\}$ random variables with $\mu = \mathbb{E}[\sum_i X_i]$. Then for any $\delta > 0$,

$$\Pr[|(\sum_i X_i) - \mu| \geq \delta\mu] \leq 2\exp\left(-\frac{\delta^2\mu}{\delta + 2}\right) \ .$$

- Generally, Chernoff gives better results then Chebyshev and Chebyshev gives better results than Markov. So choose bound based on how much you know about X.
- Bernstein generalizes Chernoff to arbitrary bounded X_i variables.

- Want to learn a quantity q. Suppose you have a randomized algorithm that returns X that has expectation q and variance σ^2.

- Want to learn a quantity q. Suppose you have a randomized algorithm that returns X that has expectation q and variance σ^2.
- To get a good estimate of q, repeat algorithm t times to get $X_1, \ldots, X_t$ and let $A = (X_1 + \ldots + X_t)/t$. Then, if $t = \frac{\sigma^2}{\delta\epsilon^2 q^2}$

$$\Pr[|A - q| \geq \epsilon q] \leq \frac{\mathrm{Var}[A]}{\epsilon^2 q^2}$$

- Want to learn a quantity q. Suppose you have a randomized algorithm that returns X that has expectation q and variance σ^2.
- To get a good estimate of q, repeat algorithm t times to get $X_1, \ldots, X_t$ and let $A = (X_1 + \ldots + X_t)/t$. Then, if $t = \frac{\sigma^2}{\delta \epsilon^2 q^2}$

$$\Pr[|A - q| \geq \epsilon q] \leq \frac{\mathrm{Var}[A]}{\epsilon^2 q^2} = \frac{\sigma^2/t}{\epsilon^2 q^2}$$

AVERAGING AND THE MEDIAN TRICK

- Want to learn a quantity q. Suppose you have a randomized algorithm that returns X that has expectation q and variance σ^2.
- To get a good estimate of q, repeat algorithm t times to get $X_1, \ldots, X_t$ and let $A = (X_1 + \ldots + X_t)/t$. Then, if $t = \frac{\sigma^2}{\delta \epsilon^2 q^2}$

$$\Pr[|A - q| \geq \epsilon q] \leq \frac{\mathrm{Var}[A]}{\epsilon^2 q^2} = \frac{\sigma^2/t}{\epsilon^2 q^2} = \delta$$

AVERAGING AND THE MEDIAN TRICK

- Want to learn a quantity q. Suppose you have a randomized algorithm that returns X that has expectation q and variance σ^2.
- To get a good estimate of q, repeat algorithm t times to get $X_1, \ldots, X_t$ and let $A = (X_1 + \ldots + X_t)/t$. Then, if $t = \frac{\sigma^2}{\delta \epsilon^2 q^2}$

$$\Pr[|A - q| \geq \epsilon q] \leq \frac{\mathrm{Var}[A]}{\epsilon^2 q^2} = \frac{\sigma^2/t}{\epsilon^2 q^2} = \delta$$

- Median Trick: Let $t = t_1 t_2$ where $t_1 = \frac{4\sigma^2}{\epsilon^2 q^2}$ and $t_2 = O(\log \frac{1}{\delta})$. Let A_1 be average of first t_1 results, let A_2 be average of next t_1 results etc. Then,

$$\Pr[|A_i - q| \geq \epsilon q] \leq 1/4$$

and $\Pr[|\mathrm{median}(A_1, \ldots, A_{t_2}) - q| \geq \epsilon q] \leq \delta$.

2-level hash tables vs. Bloom filter

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.

2-level hash tables vs. bloom filter

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.
- 2-Level Hash Table:
  - Space is $O(|S|)\times$ "space required to store an element of S"

2-level hash tables vs. bloom filter

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.
- 2-Level Hash Table:
  - Space is $O(|S|)\times$ "space required to store an element of S"
- Bloom Filter:

2-LEVEL HASH TABLES VS. BLOOM FILTER

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.
- 2-Level Hash Table:
  - Space is $O(|S|)\times$"space required to store an element of S"
- Bloom Filter:
  - Does not actually store the items in S, just a binary array from which we make various deductions.

2-level hash tables vs. bloom filter

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.
- 2-Level Hash Table:
  - Space is $O(|S|) \times$ "space required to store an element of S"
- Bloom Filter:
  - Does not actually store the items in S, just a binary array from which we make various deductions.
  - Uses only $O(|S|)$ space but at the cost of sometimes answering "yes" when answer should be "no" (a false positive)

2-level hash tables vs. Bloom filter

- Input to both is a set of items S and and both support queries of the form "Is $x \in S$?" in constant time.
- 2-Level Hash Table:
  - Space is $O(|S|) \times$ "space required to store an element of S"
- Bloom Filter:
  - Does not actually store the items in S, just a binary array from which we make various deductions.
  - Uses only $O(|S|)$ space but at the cost of sometimes answering "yes" when answer should be "no" (a false positive)
  - If the Bloom Filter array is length m, false positive probability is roughly $(1 - e^{-k|S|/m})^k$ where k is the number of hash functions used. Picking $k = \ln 2 \cdot m/|S|$ gives probability $1/2^{(\ln 2)m/|S|}$

- Designed a hash function for hashing sets such that for sets A and B, $\Pr[MH(A) = MH(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

$$MH(A) = \min_{x \in A} h(x) \quad \text{where} \quad h : U \to [0, 1] \text{ is fully independent}$$

- Designed a hash function for hashing sets such that for sets A and B, $\Pr[MH(A) = MH(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

$$MH(A) = \min_{x \in A} h(x) \quad \text{where} \quad h : U \to [0, 1] \text{ is fully independent}$$

- Can form signature of set A using r independent hash functions:

$$\text{signature}(A) = (MH_1(A), \ldots, MH_r(A))$$

Note $\Pr[\text{signature}(A) = \text{signature}(B)] = J(A, B)^r$.

- Designed a hash function for hashing sets such that for sets A and B, $\Pr[MH(A) = MH(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

$$MH(A) = \min_{x \in A} h(x) \quad \text{where} \quad h : U \to [0, 1] \text{ is fully independent}$$

- Can form signature of set A using r independent hash functions:

$$\text{signature}(A) = (MH_1(A), \ldots, MH_r(A))$$

Note $\Pr[\text{signature}(A) = \text{signature}(B)] = J(A, B)^r$.

- Given rt independent hash functions, we can form t signatures $\text{signature}_1(A), \ldots \text{signature}_t(A)$. Then if $s = J(A, B)$,

$$\Pr[\text{signature}_i(A) = \text{signature}_i(B) \text{ for some } i] = 1 - (1 - s^r)^t \ .$$

- Designed a hash function for hashing sets such that for sets A and B, $\Pr[MH(A) = MH(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

  $$MH(A) = \min_{x \in A} h(x) \quad \text{where} \quad h : U \to [0, 1] \text{ is fully independent}$$

- Can form signature of set A using r independent hash functions:

  $$\text{signature}(A) = (MH_1(A), \ldots, MH_r(A))$$

  Note $\Pr[\text{signature}(A) = \text{signature}(B)] = J(A, B)^r$.

- Given rt independent hash functions, we can form t signatures $\text{signature}_1(A), \ldots \text{signature}_t(A)$. Then if $s = J(A, B)$,

  $$\Pr[\text{signature}_i(A) = \text{signature}_i(B) \text{ for some } i] = 1 - (1 - s^r)^t \ .$$

- To find all pairs of similar sets amongst $A_1, A_2, A_3, \ldots$ only compare a pair if there exists i, their ith signatures match.

- We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.
- Let f_i be the number of values in stream that equal i.
  - Distinct Items: Can estimate $D = |\{i : f_i > 0\}$ up to a factor $1+\epsilon$ with probability $1-\delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.
  - Frequently Elements Items: Can return a set S such that:

    $$f_i \geq m/k \text{ implies } i \in S \qquad \text{and} \qquad i \in S \text{ implies } f_i \geq m(1-\epsilon)/k$$

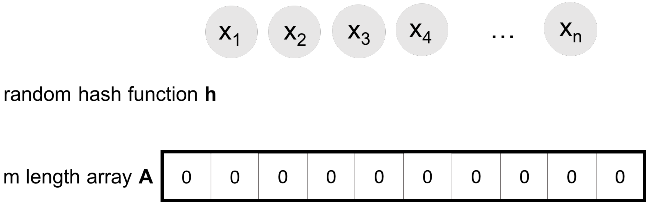
    with probability $1-\delta$ in $O(k/\epsilon \cdot \log 1/\delta)$ space.
  - Sum of Squares: Can estimate $\sum f_i^2$ up to a factor $1+\epsilon$ with probability $1-\delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.

FREQUENT ELEMENTS WITH COUNT-MIN SKETCH

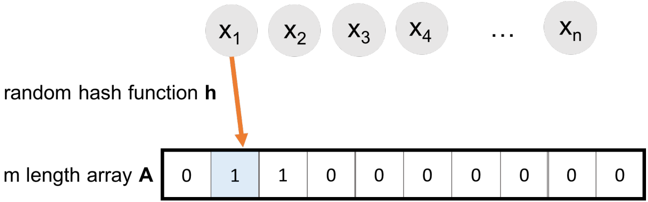
Count-Min Sketch: A random hashing based method closely related to bloom filters.

FREQUENT ELEMENTS WITH COUNT-MIN SKETCH

Count-Min Sketch: A random hashing based method closely related to bloom filters.

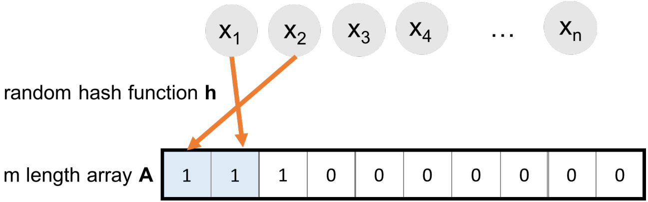


Count-Min Sketch: A random hashing based method closely related to bloom filters.

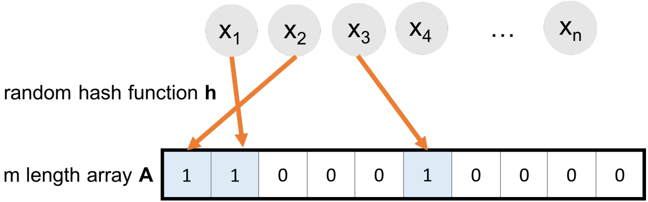


FREQUENT ELEMENTS WITH COUNT-MIN SKETCH

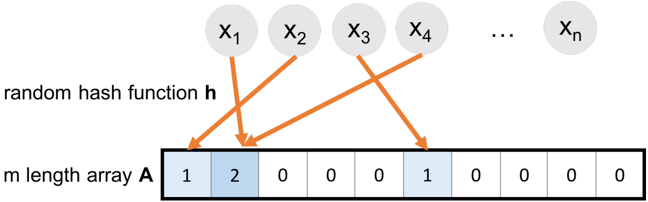
Count-Min Sketch: A random hashing based method closely related to bloom filters.



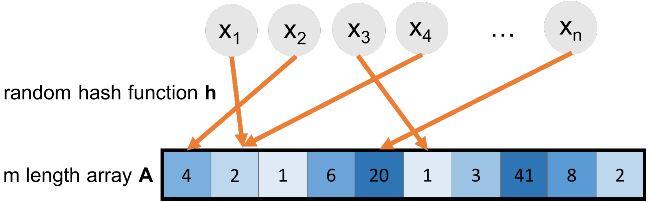
Count-Min Sketch: A random hashing based method closely related to bloom filters.



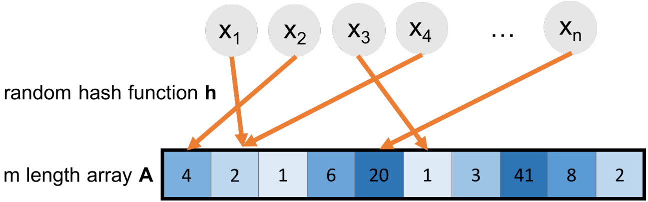
Count-Min Sketch: A random hashing based method closely related to bloom filters.



Count-Min Sketch: A random hashing based method closely related to bloom filters.



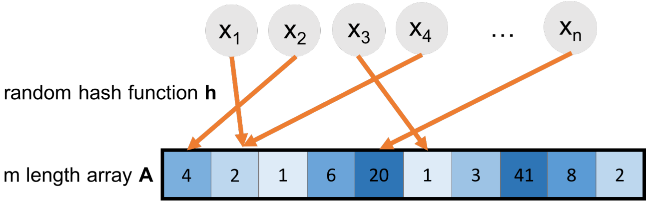
Count-Min Sketch: A random hashing based method closely related to bloom filters.



Use $A[\mathbf{h}(x)]$ to estimate $f(x)$, the frequency of x in the stream.

- **Claim:** $A[\mathbf{h}(x)] \geq f(x)$.
- **Claim:** $A[\mathbf{h}(x)] \leq f(x) + 2n/m$ with probability at least $1/2$.

Count-Min Sketch: A random hashing based method closely related to bloom filters.

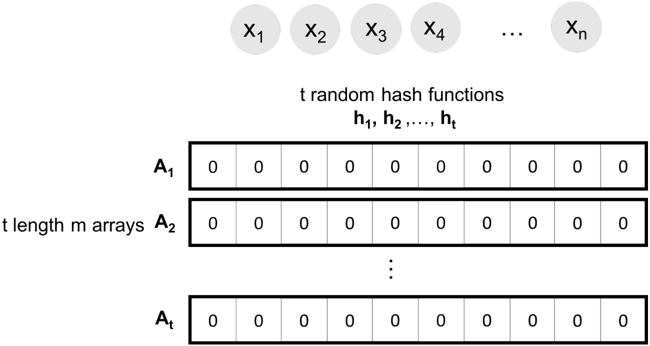


Use $A[\mathbf{h}(x)]$ to estimate $f(x)$, the frequency of x in the stream.

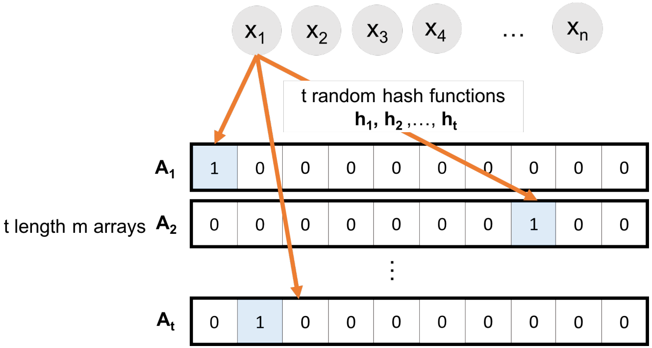
- **Claim:** $A[\mathbf{h}(x)] \geq f(x)$.
- **Claim:** $A[\mathbf{h}(x)] \leq f(x) + 2n/m$ with probability at least $1/2$.

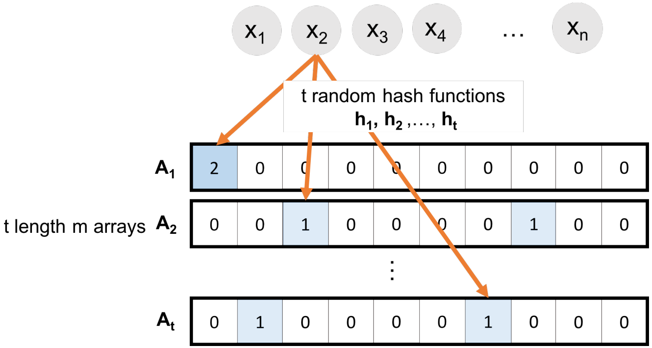
How can we increase this probability to $1 - \delta$ for arbitrary $\delta > 0$?

COUNT-MIN SKETCH ACCURACY

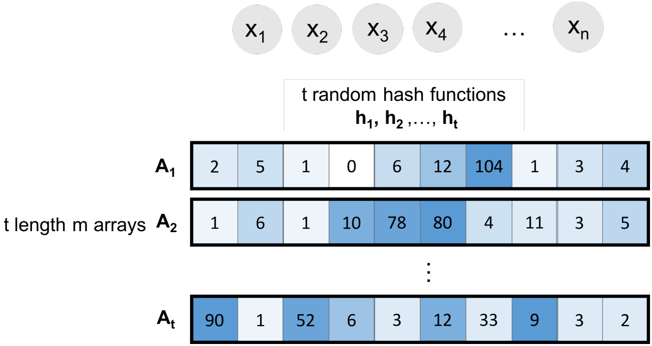


COUNT-MIN SKETCH ACCURACY

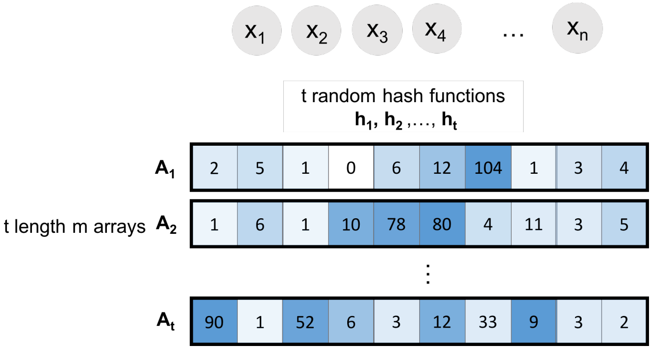




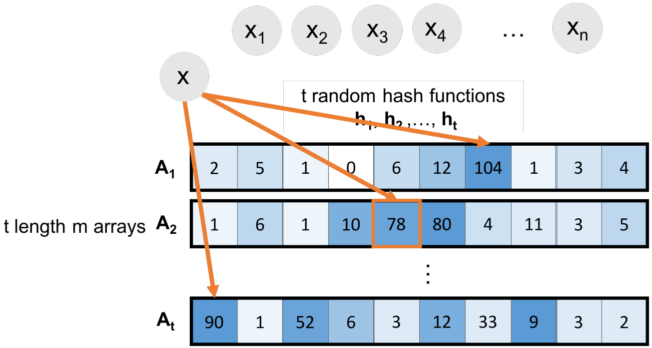
x1
x2
x3
x4
…
xn
t random hash functions
h1, h2 ,…, ht
A1
2 0 0 0 0 0 0 0 0 0
t length m arrays
A2
0 0 1 0 0 0 0 1 0 0
⋮
At
0 1 0 0 0 0 1 0 0 0

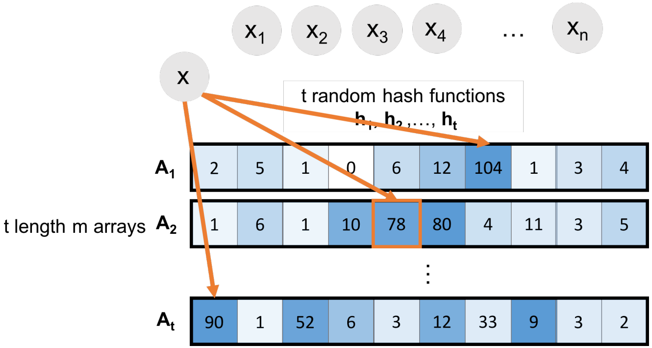


x_1
x_2
x_3
x_4
...
x_n
t random hash functions
$\mathbf{h_1, h_2, \ldots, h_t}$
t length m arrays
$\mathbf{A_1}$
2 5 1 0 6 12 104 1 3 4
$\mathbf{A_2}$
1 6 1 10 78 80 4 11 3 5
$\vdots$
$\mathbf{A_t}$
90 1 52 6 3 12 33 9 3 2


x_1 x_2 x_3 x_4 … x_n
t random hash functions
$\mathbf{h_1, h_2, \ldots, h_t}$
t length m arrays
$\mathbf{A_1}$
2 5 1 0 6 12 104 1 3 4
$\mathbf{A_2}$
1 6 1 10 78 80 4 11 3 5
⋮
$\mathbf{A_t}$
90 1 52 6 3 12 33 9 3 2

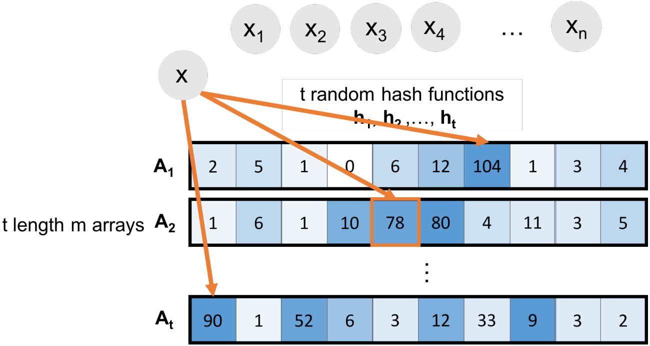
COUNT-MIN SKETCH ACCURACY





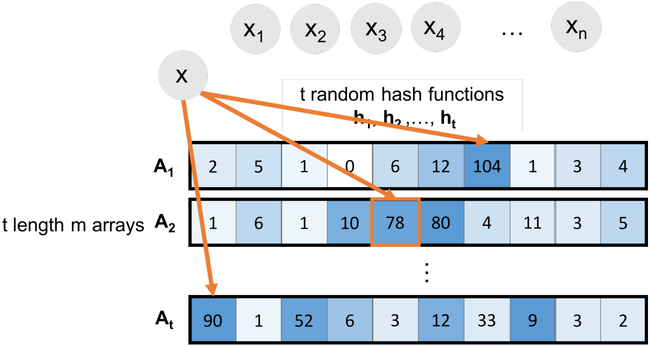
- Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$.

COUNT-MIN SKETCH ACCURACY



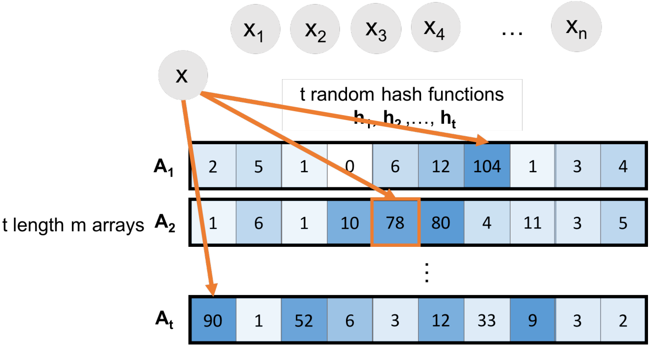
- Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$.
- Then $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + 2n/m] \geq 1 - 1/2^t$.

COUNT-MIN SKETCH ACCURACY



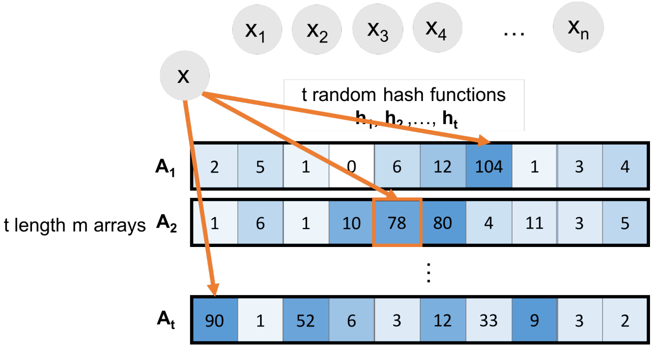
- Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$.
- Then $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + 2n/m] \geq 1 - 1/2^t$.

COUNT-MIN SKETCH ACCURACY



- Estimate $f(x)$ with $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$.
- Then $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + 2n/m] \geq 1 - 1/2^t$.
- Setting $t = \log(1/\delta)$ ensures probability is at least $1 - \delta$.

COUNT-MIN SKETCH ACCURACY



- Estimate $f(x)$ with $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$.
- Then $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + 2n/m] \geq 1 - 1/2^t$.
- Setting $t = \log(1/\delta)$ ensures probability is at least $1 - \delta$.
- Setting $m = 2k/\epsilon$ ensures $2n/m = \epsilon n/k$ and that's enough to determine whether we need to output the element.

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

Proof Idea:

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

Proof Idea:

- Follows from Distributional JL: If $\boldsymbol{M} \in \mathbb{R}^{O(\epsilon^{-2} \log(1/\delta)) \times d}$ has $\mathcal{N}(0, 1/m)$ entries then for any $\vec{y} \in \mathbb{R}^d$, $\|\boldsymbol{M}\vec{y}\|_2 \approx (1 \pm \epsilon)\|\vec{y}\|_2$ with probability at least $1 - \delta$.

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

Proof Idea:

- Follows from Distributional JL: If $\boldsymbol{M} \in \mathbb{R}^{O(\epsilon^{-2} \log(1/\delta)) \times d}$ has $\mathcal{N}(0, 1/m)$ entries then for any $\vec{y} \in \mathbb{R}^d$, $\|\boldsymbol{M}\vec{y}\|_2 \approx (1 \pm \epsilon)\|\vec{y}\|_2$ with probability at least $1 - \delta$.
- To prove Distributional JL Lemma:

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

Proof Idea:

- Follows from Distributional JL: If $\boldsymbol{M} \in \mathbb{R}^{O(\epsilon^{-2} \log(1/\delta)) \times d}$ has $\mathcal{N}(0, 1/m)$ entries then for any $\vec{y} \in \mathbb{R}^d$, $\|\boldsymbol{M}\vec{y}\|_2 \approx (1 \pm \epsilon)\|\vec{y}\|_2$ with probability at least $1 - \delta$.
- To prove Distributional JL Lemma:
  - By linearity of expectation and variance, $\mathbb{E}[\|\boldsymbol{M}\vec{y}\|_2^2] = \|\vec{y}\|_2^2$.

The Johnson Lindenstrauss lemma states that if $\boldsymbol{M} \in \mathbb{R}^{m \times d}$ is a random matrix with $m = O\left(\epsilon^{-2} \log n\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\boldsymbol{M}\vec{x}_i - \boldsymbol{M}\vec{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2$$

where $\|\vec{z}\|_2^2$ is the sum of squared entries of $\vec{z}$.

Proof Idea:

- Follows from Distributional JL: If $\boldsymbol{M} \in \mathbb{R}^{O(\epsilon^{-2} \log(1/\delta)) \times d}$ has $\mathcal{N}(0, 1/m)$ entries then for any $\vec{y} \in \mathbb{R}^d$, $\|\boldsymbol{M}\vec{y}\|_2 \approx (1 \pm \epsilon)\|\vec{y}\|_2$ with probability at least $1 - \delta$.
- To prove Distributional JL Lemma:
  - By linearity of expectation and variance, $\mathbb{E}[\|\boldsymbol{M}\vec{y}\|_2^2] = \|\vec{y}\|_2^2$.
  - $\|\boldsymbol{M}\vec{y}\|_2^2$ is the sum of m squared independent normal distributions and is tightly concentrated around the expectation.

SNEAK PEAK OF NEXT SECTION

Next Few Classes: Low-rank approximation, the SVD, and principal component analysis (PCA).

- Reduce d-dimesional data points to a smaller dimension m.
- Like JL, compression is linear – by applying a matrix.
- Choose matrix carefully based on structure of the dataset.
- Can give even better compression than random projection.

Next Few Classes: Low-rank approximation, the SVD, and principal component analysis (PCA).

- Reduce d-dimesional data points to a smaller dimension m.
- Like JL, compression is linear – by applying a matrix.
- Choose matrix carefully based on structure of the dataset.
- Can give even better compression than random projection.

Will be using a fair amount of linear algebra: orthogonal basis, column/row span, eigenvectors, etc,