Stream Computing and Applications

Benjamin Ding

November 16, 2018

1 Randomized Algorithms and Probability

1.1 Definitions

Independence

$$P[A \text{ and } B] = P[A]P[B]$$

For a random variable

$$\forall a, b \; P[Y = a \& Z = b] = P[Y = a]P[Z = b]$$
$$E[YZ] = E[Y]E[Z]$$

Mutually Exclusive

$$P[A|B] = P[A] + P[B]$$

Conditional Probability

$$P[A|B] = \frac{P[A\&B]}{P[B]}, P[B] > 0$$

Permutations

There are $n!$ ways of ordering n items There are $\frac{n!}{(n-k)!}$ ways of ordering k distinct items from a set of n There are $\binom{N}{k} = \frac{n!}{(n-k)!k!}$ ways of choosing k out of n

Binomial Distribution

$$P[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$$
$$E[X] = np$$
$$\sigma^2 = np(1-p)$$
$$P[X \geq K] \leq \binom{n}{k} p^k$$
$$P[X \leq k] = \binom{n}{k} (1-p)^{n-k}$$

1.2 Handy tricks

$$1 - x \leq e^{-x} \tag{1}$$

Markov's Inequality

If the mean of a non-negative random variable X is μ, the for every $a > 0$

$$P[X \geq \alpha\mu] \leq \frac{1}{\alpha} \tag{2}$$

Prove

Chebyshev's Inequality

$$P[|X - E[X]| \geq a] \leq \frac{1}{a^2} Var(X) \tag{3}$$

$$\tag{4}$$

Chernoff bounds

$$P[X - \mu \geq r] \leq \left(\frac{\mu e}{r}\right)^r$$

$$P[X \geq (1+\beta)\mu] \leq e^{-\beta^2\mu/3}, 0 < \beta < 1$$

$$P[X \leq (1-\beta)\mu] \leq e^{-\beta^2\mu/2}, 0 < \beta < 1$$

1.3 Birthday Paradox

- Many people in a room
- Is there a pair of people with the same birthday?

Approach 1: Probability of all birthdays distinct

Different sequences of 25 birthdays

$$= 365 * 365 * 365 * 365 = 365^{25}$$

Number of sequences where all birthdays are distinct

$$= 365 * 364 * \cdots * 341 = \binom{365}{25} 25!$$

Probability of all 25 having a different birthday

$$= \frac{\binom{365}{25} 25!}{365^{25}}$$

Hard to calculate due to large power (overflow)

Approach 2

First person has a birthday, second has birthday different with probability $1 - 1/365$, etc

With n possible birthdays and m people, probability of all unique is

$$1(1 - \frac{1}{n})(1 - \frac{2}{n})(1 - \frac{3}{n}) \dots (1 - \frac{m-1}{n})$$

By (1)

$$\begin{aligned} &\leq \exp\big(- [\frac{1}{n} + \frac{2}{n} + \dots + \frac{m-1}{n}]\big) \\ &= e^{-S} \\ S &= \frac{1}{n} + \frac{2}{n} + \dots + \frac{m-1}{n} \end{aligned}$$

Summed numerators of S constitute an arithmetic series

$$\begin{aligned} 1 + 2 + \dots + (m-1) &= \frac{m(m-1)}{2} \\ \therefore S &\geq \frac{(m-1)^2}{2n} \end{aligned}$$

When $S \geq \log_e 2$ the probability that all birthdays are different drops below 0.5

Happens when $m \geq 1 + \sqrt{2n \log_e 2}$

1.4 Coupon Collecting

- Each box contains one coupon uniformly at random from the n coupons
- Expected number of boxes to collect all n coupons?

X_i is number of boxes bought while you held $i - 1$ different kinds of coupon

- Zero coupons at first
- One coupon after first box
- Two coupons after more boxes

Total number of boxes bought is $X_1 + X_2 + \dots + X_n$ Expected value is sum of all expectations (linearity of expectation)

Calculating Expectation

With $i - 1$ different kinds of coupons, probability that next box has a new kind of coupon is $\frac{n-(i-1)}{n} = \frac{\#\text{ new kinds of coupon}}{\#\text{ kinds of coupon}}$

$$\therefore E[X_i] = \frac{n}{n - (i-1)}$$

Sum of expectations

$$\begin{aligned}\sum_i X_i &= \frac{n}{n} + \frac{n}{n-1} + \frac{n}{n-2} + \frac{n}{2} + \frac{n}{1} \\ &= n(1 + 1/2 + 1/3 + 1/4 + ... + 1/n) \\ &= n(H_n) \text{ (Harmonic series)}\end{aligned}$$

H_n is very close to $\ln_n$

Integrating $\frac{1}{x}$ from 1 to n, can show that H_n lies between $\ln n$ and $(\ln n) + 1$

$$n \ln n \leq E[\text{boxes}] \leq n \ln_n + n$$

Tighter bounds

- Each random variable X_i is an independent geometric random variable
- Calculate variance of sum $X = X_1 + X_2 + \cdots + X_n$
- Can sum variances due to independence
- Variance of geometric RV is $\frac{1-p}{p^2} \leq \frac{1}{p^2}$
- Variance is at most $\left(\frac{n}{n-(i-1)}\right)^2$
- Summed variance is $n^2\left(\frac{1}{n^2} + \frac{1}{(n-1)^2} + \frac{1}{(n-2)^2 + \cdots + \frac{1}{2^2} + 1}\right)$
- $\sum_{i=1} \frac{1}{i^2}$ converges to $\pi^2/6$
- By Chebyshev's inequality, the probability of buying more than $2nH_n$ boxes is $P[|X - nH_n| \geq nH_n] \leq \frac{n^2\pi^2/6}{(nH_n)^2} \approx O(1/\log^2 n)$

Page 31

1.5 Coin Tosses

Sequences of tosses

P46

Streaks in coin tosses

Expected length of longest streak of heads in n fair tosses

- Starting with the ith toss, probability of a streak of at least k heads
- Each flip is independent, so at most $(1/2)^k$
- For streak length $k = 2\log_2 n$, probability of length k at position i is at most $\frac{1}{2}^{2\log_2 n} = 1/n^2$
- Considering all n (actually fewer, but good upper bound) starting points, the probability of a streak of length $k = 2\log_2 n$ is at most $\frac{1}{n^2} + \frac{1}{n^2} + \frac{1}{n^2} + \cdots = \frac{1}{n}$

- Expected value of the longest streak is at most $2\log n P[\text{longest at most } 2\log n] + nP[\text{longest at least } 2\log n]$
- Streak can be at most n in length with n coin tosses
- Bound is as most $(2\log n) + n\frac{1}{n} = (2\log n) + 1$

We should expect logarithmic streaks

- Split up n coin flips into consecutive epochs of $(\log n)/2$ flips
- For one epoch, proability of all heads is $\frac{1}{2^{(\log n)/2}} = \frac{1}{(2^{\log n})^{1/2}} = \frac{1}{\sqrt{n}}$
- Probability an epoch is not a run of all heads is $1 - \frac{1}{\sqrt{n}}$
- Epochs are independent, so probability that none of the $2n/(\log n)$ epochs is a streak of all heads is $(1 - \frac{1}{\sqrt{n}})^{2n/\log n}$
- Use (1) to bound this by $\exp(-2\sqrt{n}/\log_n)$ which is at most $1/n^2$ when n is big

1.6 Balls in bins

Coupon collecting and birthday are both variants of this

Maximum Load

- For $m = n$ on average each bin has exactly one ball
- At worst, all balls in the same bin with max load n (very unlikely)
- Max load with high probability is something like $\ln n / \ln \ln n$

Focusing on a particular bin, the probability that it receives at least M balls is at most Don't understand this

- Probability that all M balls land there is $\left(\frac{1}{n}\right)^M$
- The number of sets of M balls is $\binom{n}{m}$
- A loose (union) bound is at most $\binom{n}{m}\left(\frac{1}{n}\right)^M$ which is at most $\frac{1}{M!} \leq \left(\frac{e}{M}\right)^M$
- $\frac{M^M}{M!} \leq e^M$ because $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots$
- P59 what the fuck
- Probability that some bin has at least M balls is at most $1/n$
- Very unlikely that with n balls into n bins there is some bin with more than $3\ln n / \ln \ln n$ balls

2 Frequent Items

2.1 Misra-Gries Algorithm

- Keep track of $k - 1$ items with a counter for each
- For each new item x
  - If x is a tracked item, increment counter

  - If x is not tracked and if fewer than $k-1$ items are tracked, add x to the tracked items with a count of 1
  - Else we already have k tracked items and x is not one of them
    * Decrement the count of every tracked item
    * Evict every tracked item that now has count zero
- At the end of the stream return all tracked items

Properties

- Space required is proportional to product of k and max{log(Universe size), log(Stream length)}
- Only need one pass through data
- For all items, the frequency estimate is at most the true frequency and at least the true frequency minus m/k

Charging

- Whenever we decrement, it's because the item was present earlier
- When we decrement, do k decrements together:
  - One for each of the $k-1$ tracked elements
  - One for the new item x that caused decrements (pretend gave x frequency 1 and immediately decremented)
- Charge each decrement of an item to earlier occurrence of that item
- Since m item occurrences in the stream, there are at most m/k such k-at-a-time decrement instances
- Hence estimated frequency count is at least true count minus m/k
- Items with frequency more than m/k must have a positive frequency and be tracked, but some tracked items may not be that frequent

2.2 Reservoir Sampling

- Want to take uniform random sample wihout replacement of a stream of data
- Don't know length of stream m
- Do know k, number of items want to sample

Algorithm

- Let S[1...k] be empty array
- Let m be 0
- For each item x
  - Increment m

- If $m \leq k$ put x in $S[m]$
- Else
  - Choose r uniformly in $[1 \dots m]$
  - If $r \leq k$, $S[r]$ becomes x

- When queried, output S

Proof

3 Counters

3.1 Morris counter

- Let z be 0
- When a new item arrives
  - Flip a coin with heads probability $1/2^z$
  - If heads, increment z
- Return $2^z - 1$ as estimated count

Properties

- Unbiased estimate
- Uses $\log \log n$ bits
- Variance of $Y_n = E[2^{Z_n}]$ is $n(n-1)/2$

Combining Morris counters

Morris counter is inaccurate, but several independent instances can be combined together

- Group $h = 2/\epsilon^2$ Morris counters and take mean value (rv X)
  - $P[|X - n| \geq \epsilon n] \leq \frac{Var(X)}{\epsilon^2 n^2}$ from Chebyshev's inequality
  - Variance of X is $1/h$ times the variance of an individual Y_n value, which is $n(n-1)/2$
  - Bound is $\frac{1}{\epsilon^2 n^2} \frac{n(n-1)}{4n^2} < \frac{1}{4}$
  - Probability that estimate is more than ϵn away from the true value n is bounded by less than $1/4$
- Take $c \log \frac{1}{\delta}$ groups and return median of means
  - Median of means is within ϵn of true value n with probability $1 - \delta$
  - If median of Xs is bad, it means that at least half of the independent Xs are bad
  - Using Chernoff bound with $\beta = 1$, P[at least half bad] is at most $e^{-L/12}$
  - For $L = c \ln \frac{1}{\delta}$ and $c = 12$ the probability the median is bad is at most $e^{-\ln 1/\delta}$

- With c = 12, use $24\frac{1}{\epsilon^2}\ln\frac{1\delta}{\log}\log n$ bits of space to get an accurate estimate with high probability $1-\delta$ within additive error ϵn of true count n

4 Hash functions

Desired properties

- Spreads items evenly
- Fast
- Hashes same key to same value (detemrinistic)

4.1 Bloom Filter

- Use a family of k hash functions
- Hash each arriving item with multiple hash functions
- Set hashed value in boolean array to true
- When checking for membership, check if hash of all functions is true in array

False positive probability

- m distinct elements in stream
- n cells in array (hash range)
- k different hash functions
- P[False after m elements] is $\left(1-\frac{1}{n}\right)^{km}$ bounded by $e^{-km/n} \equiv p$
- Different bits being true in the array are not independent (but assume they are for simplicity)
- P[False Positive] $= \left(1-\left(1-\frac{1}{n}\right)^{km}\right)^k \approx (1-e^{-km/n})^k = (1-p)^k$

Choosing number of hash functions

- Given the array size and number of distinct items is known
- With more hash functions, greater chance of finding a false bit when querying an item that wasn't in the stream
- With more hash functions, also too many of the bits get turned true
- Take logarithm of false positive probability $k\log(1-e^{-km/n})$
- Using calculus, minimise when $k = (\ln 2)n/m$
- P[False positive] $= 0.6185^{n/m}$
- Minimised when $p = 1/2$

5 Distinct Items

5.1 AMS

- Record the largest power of two that divides into a hash value which has been seen z
- Very efficient with binary representation, count the number of trailing zeroes
- Observe that if some item appears in the stream multiple times, it only potentially increases the values of z at its initial appearance

Algorithm

- Let z be 0
- For each item x
  - $z \leftarrow \max\{z, \text{zeros}(h(x))\}$
- Return $2^{z+1/2}$

Properties

- $P[D \geq 3d] = P[Y_a \geq 1] \leq d/2^a \leq \sqrt{2}/3$
- $P[D \leq d/3] = P[Y_{b+1} = 0] \leq 2^{b+1}/d \leq \sqrt{2}/3$
- $\text{Var}(Y_r) = \sum_j \text{Var}(X_{jr}) \leq \sum_j E[X_{jr}^2 = \sum_j E[X_{jr}] = d/2^r$
- Shows estimate is within factor of three of d with probability greater than 0.5286
- Hash function is chosen from 2-universal family with pairwise independence
  - For every pair of items, the two hash values assigned to the pair behave as if they were assigned uniformly randomly
  - Means variances can be added up

5.2 BJKST1

- Track the value of roughly $1/\epsilon^2$ items where ϵ is an accuracy parameter
- Report maximum of tracked hash values and calculate appropriately

Algorithm

- Choose a hash h function uniformly at random from a 2-universal family that maps $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, n^3\}$
- Let t be $96/\epsilon^2$
- Priority queue Q will record the t smallest hash values seen so far (initialise with t values $> n^3$)
- For each item x

  - Let m be the largest hash value in Q
  - If $h(x) < m$ AND $h(x)$ is not already in Q
    - Replace m with $h(x)$ in Q
  - Let m be the largest hash value in Q
- Return $t * n^3/m$

Analysis

- With 2/3 probability, D lies within $\pm\epsilon d$ of the true number of distinct items d
- Space needed is $O(\log n)$ for each of the t hash values stored, so $O((1/\epsilon^2)\log n)$ total
- Update time is $O(\log t)$, that is $O(\log(1/\epsilon))$ comparisons in the priority queue, with each comparison needing $O(\log n)$ time
- Probability of success can be boosted by taking the median of $O(\log(1/\delta))$ to obtain success probability $1 - \delta$
- Proof
- Lower bound

6 Universal Hash Functions

- Domain of size n and range r
- Want pairwise independence
- Will settle for something less: For a given pair of distinct items, probability of choosing a hash function that causes them to collide is at most $1/r$

6.1 The function

- Choose a prime p that is as least as big as $\max\{n, r\}$
- $h_{ab}(x) = ((ax + b) \mod p) \mod r$
- Family of hash functions has
  - a drawn from $1, 2, \ldots, p - 1$
  - b drawn from $0, 1, 2, \ldots, p - 1$

6.2 Analysis

- $g(x) = (ax + b) \mod p$
- If $g(x) = g(y)$, it means $a(x - y)$ is divisible by p
  - Primes can't be broken up into parts
  - p is bigger than each of a and x and y, can only happen if $x - y = 0$

- Therefore collision in h only occurs by the action of $\mod r$
- There is exactly one choice of a, b that leads to a particular pair $g(x) = u, g(y) = v$
- For each u there might be up to $\lceil p/r \rceil 1$ (minus 1 because of u itself) such v
- We can show that $\lceil p/r \rceil \leq \frac{p}{r} + 1 - \frac{1}{r}$
- There are p values of u so the number of colliding pairs is at most $p(p-1)/r$
- The total number of pairs (u, v) is $p(p-1)$
- The probability of a bad pair is at most $1/r$

7 Estimating "higher" functions

- Interested in functions like (Frequency of a)^2 + (Frequency of b)$^2 + \ldots$
- Estimate $F_k = \sum_x f_x^k$

7.1 AMS

Algorithm

- Choose an item y uniformly at random from the stream (with position j)
- Let r be the number of occurrences of y after and including position j
- Return the value $z = m(r^k - (r-)^k)$ where m is the length of the stream

Expected value

- Can do this with reservoir sampling
- Choose a value y to sample with probability proportional to f_y
- Choose one of those f_y instances of y uniformly at random
- For a given y
  - Equally likely to pick each of the f_y instances of y
  - $$\begin{aligned} E[Z] &= \frac{1}{f_y} \sum_{i=1}^{f_y} m[(i^k - (i-1)^k] \\ &= \frac{1}{f_y} m\big((1^k - 0^k) + (2^k - 1^k) + \cdots + (f_y^k - (f_y - 1)^k) = f_y^k\big) \\ &= m f_y^{k-1} \end{aligned}$$
  - Probability of choosing y in the stream is f_y/m
  - Sum over all possibilities of y, $E[Z] = \sum_y f_y^k$

Variance

- Mean value theorem bound
- Upper bound for variance $m \sum_y k f_y^{k-1} f_y^k = kmF_{2k-1}$
- $kn^{1-1/k}F_k^2$
- Very large

Reducing variance

- Take mean of several estimators
- Groups of size q where q is $\frac{3\text{Var}(X)}{\epsilon^2 E(X)^2}$
- Reservoir samplers are independent, variance of sum of q of these is $\text{Var}(X)/q$
- Use Chebyshev's inequality, show that the grouped mean is more than $(1+\epsilon)E[X]$ is at most $1/3$
- Take median of several of these grouped means
  - Constant times $\log(\frac{1}{\delta})$ to achieve estimate at most $(1+\epsilon)E[X]$ with probability $1-\delta$
  - $\text{Var}(X)/[E(X)]^2$ is at most $kn^{1-1/k}$
  - Dominant terms in space bound are $\frac{1}{\epsilon^2}$ and $kn^{1-1/k}$
- Can obtain bound of $O(n^{1-2/k})$ with other techniques

7.2 F_2 sketch (AMS) (Tug-of-war)

Algorithm

- Pick a random hash function h mapping $\{1, 2, \ldots, n\}$ to ± 1 from a four-universal family
- Let $z \leftarrow 0$
- As each item x arrives, with "count" c:
  - $z \leftarrow z + ch(x)$
- Return z^2

Expected value

$$Z = \sum_i f_i h(i)$$

$$E[Z^2] = E[\sum_i f_i^2 h(i)^2 + \sum i \sum_{j:j\neq i} f_i f_j h(i)h(j)]$$

Because h is four universal, the expected value of $h(i)h(j)$ is just the product of individual expected values

$$E[h(i)] = 0$$
$$E[Z^2] = \sum_i f_i^2 = F_2$$

Variance

- $E[(Z^2)^2] = E[Z^4] = \sum_i \sum_j \sum_k \sum_l f_i f_j f_k f_l E[h(i)h(j)h(k)h(l)]$
- Terms except $h(i)^4$ $h(i)^2 h(j)^2$ cancel
- There are $\binom{4}{2} = 6$ copies of $h(i)^2 h(j)^2$
- $E[(Z^2)^2] = F_4 + 6 \sum_i \sum j > i f_i^2 f_j^2$
- Can show this is at most $2F_2^2$
- Apply median of means

Geometric interpretation

- Take mean of collection of sketches, before applying median trick
- Random vector of form $(Z_1, Z_2, \ldots, Z_t)$ for $t \approx 1/\epsilon^2$
- $\frac{1}{t} \sum_j Z_j^2 \approx F_2$
- Euclidean length of the Z vector approximates Euclidean distance of the frequency vector, with a much smaller dimension

7.3 Count-min sketch

- Estimates frequencies of items
- Can help find heavy hitters
- Works in a turnstile stream
- Linear

Algorithm

- Keep a family of d pairwise-independent hash functions and w buckets
- Each hash function maps an input to a value in $\{1, 2, \ldots, w\}$ (corresponds to columns in the array)
- When an item arrives with count increment c (could be negative), add c to several counters based on hashed values of x with each hash function
- To estimate $f(x)$, take the minimum value of the cells pointed to by the hash function family
- Space usage is $d \times w$ and each cell contains a counter of $\log m$ bits plus the d hash functions of $O(\log n)$ bits each

Accuracy

- Estimate returned is never an underestimate (for intuitive reasons)
- For row j, input y, inspect $h_j(y)$
- All of y's increments contribute, as do other elements x that collide $h_j(x) = h_j(y)$
- $E[\hat{f}_j(y) = f(y) + \sum_{x \neq y} P[h_j(x) = h_j(y)]f(x)$
- Due to 2-way independence (or at least 2-universal), this is $f(y) + \frac{1}{w}\sum_{x \neq y} f(x) \leq F(y) + F_1/w$
- Let $F_k = \sum_x |f(x)|^k$ to deal with negative counts for these sketches

Choosing w

- Want estimates within multiplicative accuracy ϵ with high probability
- $w = 2/\epsilon$
- $E[\hat{f}_j(y) \leq f(y) + \frac{\epsilon F_1}{2}$, by Markov's inequality $P[Y_j - f(y) \geq \epsilon F_1] \leq \frac{1}{2}$
- This is just for one row

Choosing d

- For d rows, due to independence $P[Y_1 - f(y) \geq \epsilon F_1 \cap Y_2 - f(y) \geq \epsilon F_1 \cap Y_d - f(y) \geq \epsilon F_1] \leq \left(\frac{1}{2}\right)^d$
- $d = \log_2(\frac{1}{delta})$ so probability of a bad estimate of y for one of the values in the domain $[1, 2, \ldots, n]$ is at most δ
- $\delta' = \frac{\delta}{n}$, so the probability of some bad estimate is at most $n \times \delta' = \delta$
- The number of columns should be $\log\left(\frac{1}{\delta'}\right) = \log_n + \log\left(\frac{1}{\delta}\right)$

Summary

- d hash functions and $d \times w$ size table with counters of size $\log m$
- Total space $O\left(\frac{1}{\epsilon}(\log\frac{1}{\delta} + \log n)(\log m + \log n)\right)$
- Estimates accurate within additive term of ϵF_1

7.4 Count sketch

- Also use a table $d \times w$
- $w = \frac{3}{\epsilon^2}$, roughly square of width of count-min sketch
- Hash functions h_j, family of d hash functions g, which are also 2-universal but map to $\{-1, +1\}$ rather than w
- Uses more space, has a $\frac{1}{\epsilon^2}$ factor instead of a $\frac{1}{\epsilon}$ factor
- Has better additive error (ϵF_1 as opposed to $\epsilon\sqrt{F_2}$)

Algorithm

- When see an item x with count increment c, instead of adding c to the cells, add $g_j \times c$
- Return the median of the d values $g_j(y)T[j, h_j(y)]$ stored for a given item y where T is the table of estimated counts

Expected value

- $I_j(x)$ is the indicator that hash function j sends x and y to the same cell in row y
- $E[Y_j] = E[g_j(y)[g_j(y)f(y) + \sum_{x \neq y} g_j(x)f(x)I_j(x)]$
- $g_j(y)g_j(y) = 1$ because it's a hash function mapping to ± 1
- For each x that is not equal to y, $E[g_j(y)g_j(x)f(x)I_j(x)] = f(x)E[g_j(y)g_j(x)]E[I_j(x)]$
  - $f(x)$ is not a random variable
  - g_j and I_j are independent hash functions
- $E[g_j(y)g_j(x)] = E[g_j(y)]E[g_j(x)] = 0$ as it is 2-way independent
- $\therefore E[Y_j] = f(y)$

Variance

Expand to deal with

- Pairs of x, x (neither equal to y)
- Pairs of $x \neq z$ (neither equal to y)

$$\begin{aligned}
\text{Var}[(Y_j)] &= E\big[[Y_j - f(y)]^2\big] \\
&= E\big[\big(\sum_{x \neq y} g_j(y)g_j(x)f(x)I_j(x)\big)^2\big] \\
&= E[\sum_{x \neq y} g_j(x)^2 f(x)^2 I_j(x)^2] + E[\sum_{x \neq z, \neq y} f(x)f(z)g_j(x)g_j(z)I_j(x)I_j(z)] \\
&= \sum_{x \neq y} f(x)^2(\frac{1}{w}) + 0(\text{two way independence}) \\
&\leq \frac{1}{w}\sum_x f(x)^2 \equiv F_2/w \\
w &= \frac{3}{\epsilon^2}
\end{aligned}$$

Using Chebyshev's inequality

$$P[|Y_j - f(y)| \geq \epsilon\sqrt{F_2}] \leq \frac{\text{Var}[Y_j]}{\epsilon^2 F_2} \leq \frac{1}{\epsilon^2 w} \leq \frac{1}{3}$$

There are d rows and n items, so need $\log(\frac{1}{\delta}) + \log n$ rows to have a high probability of accurate estimates

8 Sketches

8.1 Dyadic intervals

- Includes all integers from one power of two to the next
- Organised into sub-families, each comprising intervals of th same width
  - Family 0: $[0,1), [1,2), [2,3), \ldots$
  - Family 1: $[0,2), [2,4), [4,6), \ldots$
  - Family 2: $[0,4), [4,8), [8,12), \ldots$
- General format is $[k\dot{2}^j, (k+1)\dot{2}^j)$
- An arbitrary sub-interval of $[0,n)$ can be broken into at most $2\log n$ dyadic subranges

8.2 Heavy Hitters

- Which items are especially frequent (frequency $\geq \phi F_1$)
- Naive approach to repeatedly query each item and ask for estimate of frequency (lots of false positives, slow)
- If turnstile model, only need to search for each item as it appears in the stream and record heavy hitters in a separate table (still false positive, fast)
- Count-min supports decrements

Turnstile model

- Keep a family of count-min sketches in binary tree
- Each node in binary tree represents a subset of the elements, broken into two further subsets at the next lower level
- Nodes in a single level of the tree correspond to elements that correspond to a single count-min sketch
- Each level in the tree demands searching for several super-items to determine that they are heavy hitters, but there is only a small amonut of work per node, and only $O(\frac{1}{\phi}\log n)$ nodes to explore, due to only $1/\phi$ per level
- Also helps find the total frequency of a range of items
- Can combine ranges to find other ranges

Dyadic interval sketches for range queries

- Using one count-min sketch per level, can see with $\log n$ levels (and sketches), can record enough information for every possible range query.
  - At most two interval queries per sketch
  - Since there are more queries, to obtain the same accuracy as before need each sketch to be $\log n$ wider

– Since there are $\log n$ sketches, total space needed is $O(\frac{1}{\epsilon}\log^3 n)$

9 Sparse Recovery

- Given a vector $(x_1, x_2, \ldots, x_n)$, try to recovery the k largest coordinates from an approximate version of x
- A vector with only k non-zero components is k-sparse
- Aim to minimize Euclidean error $err^k(x) = \min_{\text{k-sparse z}} ||z - x||_2^2$
- Try to estimate error using count sketch
- Difficult because allow deletions in input

9.1 Count sketch for sparse recovery

- Count sketch gives point estimates with probability 2/3 (in each row) of

$$\begin{aligned} |\hat{x}_i - x_i| &< \epsilon\sqrt{F_2} \\ &= \frac{\sqrt{3}}{\sqrt{w}}||x||_2| \\ &= \frac{\sqrt{3}}{\sqrt{w}} err^0(x) \end{aligned}$$

- To estimate $err^k(x)$, we increase the width of count-sketch from $3/\epsilon^2$ to $3k/\epsilon^2$
- Now with probability $> 1/2$

$$|\hat{x}_i - x_i| < \frac{\epsilon}{\sqrt{k}} err^k(x)$$

Heavy collisions

- Let the heavies be the k largest coordinates in the true vector x (only k heavies)
- When reporting point estimate of another coordinate j, if a heavy coordinate collides then we are likely to lose information about non-heavy coordinate
- Due to 2-universality, probability of a heavy colliding with j is only $\frac{k}{w} = \frac{\epsilon^2}{3}$
- If no heavy collision, probability of an estimate $\hat{x}_j$ of x_j within $\frac{\sqrt{3}}{\sqrt{w}} = \frac{\epsilon}{\sqrt{k}}$ factor times $||\breve{x}||_2$ where $\breve{x}$ is the vector x without the heavy components, is $> 2/3$
- $||\breve{x}|| = err^k(x)$
- The k largest components of the count sketch may not be the same as the largest components of the actual x vector, but they give a good approximation
- Let the vector obtained from the k-largest components of $\hat{x}$ (the vector hidden inside the count sketch)
- Assuming that each component of $\hat{x}$ differs from x by at most $\frac{\epsilon}{\sqrt{k}} err^k(x)$ (shown above), with high probability:

$$||x - z||_2 \leq (1 + 5\epsilon) err^k(x)$$

Error bounds

- Let k largest components of x be set S (unknown)
- Let k largest components of $\hat{x}$ be set T
- Total $||z - x||_2^2$ is sum of $||z - x||_2^2$ from each subset of coordinates with the three parts
  - T
    - There are k coordinates in T
    - Each z_i for i in T is $\hat{x}_i$
    - Error in each is by assumption at most $\frac{\epsilon}{k} err^k(x)$
    - Therefore the total $||z - x||_2^2$ on T is at most $k\left(\frac{\epsilon}{\sqrt{k}} err^k(x)\right)^2 = \epsilon^2 E^2$ where E stands for $err^k(x)$
  - $S \backslash T$
    - $z_i = 0$ so the error is $\sum_i x_i^2$
  - $[n] \backslash (S \cup T)$
    - $z_i = 0$ so the error is $\sum_i x_i^2$
- S and T are each of size k so $|S \backslash T| = |T \backslash S|$
- $x_{S \backslash T}||_2^2 \leq ||x_{T \backslash S}||_2^2 + 8\epsilon E^2$
- $E^2 = ||x_{T \backslash S}||_2^2 + ||x_{[n] \backslash (S \cup T)}||_2^2$
- Total $||z - x||_2^2$ of all parts bound by $(1 + 9\epsilon)E^2$
- Taking final square root, bound is $(1 + 5\epsilon)E$
- What the fuck

10 ℓ_0 Sampling

- Want to sample roughly in proportion to frequency in stream
- Only know how to do in additive stream
- Need to handle deletions

10.1 Sparse recovery for sampling

- If the input vector x is k-sparse, then $E = 0$ and count-sketch would return x exactly, if we knew how to identify the k components easily

Ganguly's Test

On a stream of (a_j, c_j) pairs:

$$\begin{aligned} F_1 &\leftarrow F_1 + c_i \\ U &\leftarrow U + a_j c_j \\ V &\leftarrow c_j a_j^2 \\ \therefore U &= \sum_i i f_i \text{ A scaled average across items} \\ \therefore V &= \sum_i i^2 f_i \text{ A scaled } E[I^2] \text{ across items} \end{aligned}$$

Test whether $\left(\frac{U}{F_1}\right)^2 = \frac{V}{F_1}$, that is, is the variance equal to zero.

- Only makes sense if all values of f_i are non-negative

Fingerprint Test

Deal with negative frequencies

- Choose a large prime p
- Choose a random $q \in \{0, 1, \ldots, p\}$
- Compute $\tau = \sum_i x_i q^i \mod p$
- Test whether $\tau = F_1 q^{U/F_1} \mod p$

If there is a single i with non-zero frequency x_i

- $U = i x_i$
- $F_i = x_i$
- $F_1 q^{U/F_1} = x_i q^i = \tau(\mod p)$
- Test succeeds

If there is not just one item with non-zero frequency, low probability of false positive

- $B(q) = \sum_i x_i q^i - F_1 q^{U/F_1}$ is a polynomial in q that we are testing for divisibility by p
- Since polynomial $B(q)$ has degee n, it has at most n roots modulo p
- But q was chosen independently, so probability it is a root is at most n/p where p is very large

1-sparse to k-sparse

Finding a k-sparse vector

- Make 2k buckets and hash all the coordinates to these
- In each bucket, run 1-sparse recovery
- If x is indeed k-sparse, then the probability that just one coordinate lands in a particular bucket is about 1/2

- We now make $\log\left(\frac{k}{\delta}\right)$ copies of this bucketing structure, so the probability that a given coordinate in A is never alone is at most $\left(\frac{1}{2}\right)^{\log\left(\frac{k}{\delta}\right)} = \frac{\delta}{k}$
- Since there are k non-zero coordinates, by the union bound the probability of failure is just *delta*
- The space required is $O(k \log(\frac{k}{\delta}))$

Algorithm

- Construct a nested family of random subsets of the coordinates $\{1, 2, \ldots, n\}$ so that subset S_j contains $n/2^j$ coordinates, for j from 0 up to $\log n$
- Build a k-wise independent hash functoin mapping from n to n^3
- Coordinate i is included in subset j if and only if the hash value of the coordinate i is less than $n^3/2^j$
- From each subset, attempt sparse recovery with $k = \lceil 4 \log(\frac{1}{delta}) \rceil$
- From the largest-size (smallest index) subset that succeeds in returning a k-sparse vector, return a uniformly random coordinate (the one with the smallest hash)

Analysis

- Succeeds with probability $1 - \delta$
- If there fewer than k non-zero coordinates in x, then even the all-n coordinates sample will succeed, and we ar egood
- Otherwise,
  - let A be the set of non-zero coordinates in the stream $A = \{i : f_i \neq 0\}$
  - The expected size of $|A \cap S_j|$ is $|A|/2^j$ via an averaging argument
  - So there is some j for which $E[|A \cap S_j|]$ lies between $k/3$ and $2k/3$
  - Hence with probability at least $1 - \delta$ there are at least 1 and at mots k coordinates in $|A \cap S_j|$ so a k-sparse vector, with at least one non-zero component, is returned

11 Metric-style k-center clustering

- Can build at most k warehouses/facilities/cluster centers
- Seek to minimize the worst of the service costs to each customer from its nearest facility
  - Don't add up service costs
  - Simply focus on the customer that is the furthest from all facilities
- Can't be solved exactly in a reasonable amount of time (NP hard)
- Best in a reasonable time is twice as optimal

11.1 Triangle inequality

- $d(x, y)$ represents distance between x and y
- $d(x, z) \leq d(x, y) + d(y, z)$

11.2 Standard algorithm

Run over an array, not streaming

- Pick a point arbitrarily
- While we have picked fewer than k points
  - Pick the point that is farthest from the points picked so far (that is, has maximumum minimum distance to the picked points)
  - Return the k picked points

11.3 Doubling algorithm

Uses $O(k)$ space

Lemma

If there are $k + 1$ points all at distance t apart from one another in the input, then no matter what set of k representatives I choose, the solution cost must be at lesat $t/2$

Algorithm

- Initialize by taking first $k + 1$ elements from the stream and setting (y,z) to be the closest pair in these first $k + 1$ elements
- Let τ be $d(y, z)$ and let our representatives R be the $k + 1$ elements so far, except z
- For each new item x
- If it's minimum distance to an element of R is $> 2\tau$
  - Add x to R
  - While $|R| > k$:
    - // Property 3 is true here
    - $\tau \leftarrow 2\tau$
    - Find a maximal subset R^* of R so that for every pair of distinct items in $R*$, their distance is at least τ
    - Let R be R^*

Properties

1. For all pairs of items in R, their distance is at least τ
2. The k-center cost of R with the whole stream (so far) is at most 2τ
3. After initialization, and just before each reset of R: An optimal solution has cost at least $\tau/2$

At initialisation, the properties are valid

1. Follows immediately from the fact that $\tau = d(y, z)$ for the closest pair (y, z)
2. In fact it's τ, as z is the only point not in R
3. Follows from Lemma

If x is not added, properties valid

1. R does not change, so this holds
2. x is close to R, so this holds
3. Don't need at this point

If x is added, properties valid
During add of x

1. x's minimum distance to an element of R is $> 2\tau$ and it is added to R, remains true even as R changes.
2. As x is added to R, its solution cost for the stream is at most $\leq 2\tau$
3.

During while loop

1. Choosing the new R as defined by algorithm demands this property remains true
2. Since this was already true, after τ was doubled, but before the new R^* was chosen, $\Delta(\sigma, R) \leq \tau$ Consider an arbitrary item x in stream σ
   - If it's closest point in R, x_R is also in R^*, then x is within τ of R^*
   - If x_R not in R^*, since set R^* was maximal, with each pair at distance τ from one another, every other point in R, including x_R is within τ of R^*
   - So by the Lemma, x is within 2τ of R^*
3. Before we reset R in the while loop, R is a set of $\geq k + 1$ points that have distance at least τ from one another, so by triangle inequality this holds

Effectiveness of doubling algorithm

- From property 2, at the end of the stream, k-center cost of R with the whole stream (so far) is at most 2τ
  - Final R is $\hat{R}$; final τ is called $\hat{\tau}$, σ is the stream, Δ is the k-center cost funtcion
  - $\Delta(\sigma, \hat{R}) \leq 2\hat{\tau}$
- From property 3, just before the last time we update R, an optimal solution has cost at least $\tau/2$

- When we update R for the final time, we have $\hat{\tau} = 2\tau$ (that is, the τ used to found the optimal cost) so the ratio of $\Delta(\sigma, \hat{R})$ to the optimal cost is at most $\frac{2\hat{\tau}}{-\tau/2} = \frac{2\hat{\tau}}{(-\tau/2)/2} = 8$
- Solution has at most eight times the optimal cost

11.4 Guha's algorithm

Uses $O\big(\frac{k}{\epsilon}\big)$ space

Simple Thresholding Algorithm

- Maintain a set R of representatives of the stream that are at least 2τ apart from one another
  - If the stream contains an item $\geq 2\tau$ from R then add it to R
  - If at some stage R has size at least $k+1$, then FAIL
  - Otherwise, output R
- Either R is a solution of cost 2τ because it never had more than k items
- Or if it returns fail, the Lemma states that there is no solution of cost less than τ

Guha's Algorithm

- Produces an $\alpha + O(\epsilon)$ factor approximation of optimal (in this case $\alpha = 2$)
- Initialize by determining a lower bound c on optimal cost
- Let $p = \lceil \log_{1+\epsilon} \big(\alpha/\epsilon\big) \rceil$
- Maintain at all times p instances of the simple algorithm
- Initially, the thresholds for the instances are $c(1+\epsilon), c(1+\epsilon)^2, \ldots, c(1+\epsilon)^p$
- Whenever an instance returns FAIL:
  - Raise its threshold by a factor of $(1+\epsilon)^p$
  - Although this instance cannot access directly the items it has already seen, it relies on the representatives R of the stream so far to represent the part of the stream already read

Space requirements

- Initially, $O(k)$ space to determine a lower bound
- Then p copies of a $O(k)$ space simple algorithm
- With α being constant, total space is

$$O\big(k \log_{1+\epsilon} \big(\frac{\alpha}{\epsilon}\big)\big) = O\big(\frac{k \log_{\epsilon}^{1}}{\log(1+\epsilon)}\big) = O\big(\big(\frac{k}{\epsilon} \log \frac{1}{\epsilon}\big)$$

- Relying on facts that
  - $\log(1+\epsilon) \approx \epsilon$

- In our case, $\alpha = 2$

Performance

- Focus on the instance with the smallest threshold that succeeds
- Let t be this threshold
  - But the threshold of this instance might have increased several times as the streamd flowed
  - Say it increased j times
  - Had the following thresholds $t_1, t_2, \ldots, t_{j+1} = t$
  - Break up the stream σ into $j+1$ phases, each corresponding to a different threshold $\sigma_1, \sigma_2, \ldots, \sigma_{j+1}$
- In particular, threshold $t_i = t_{j+1}/((1+\epsilon)^p)^{j+1-i}$
- Chose $p = \lceil \log_{1+\epsilon}(\alpha/\epsilon) \rceil$ so that $(1+\epsilon)^p \geq \alpha/\epsilon$
- Hence $t_i \leq \left(\frac{\epsilon}{\alpha}\right)^{j+1-1} t$
- Rest of this analysis
- Cost of an optimal solution is at least $t/(1+\epsilon)$
- Solution is within $(1+\epsilon)(\alpha + O(\epsilon)) = \alpha + O(\epsilon)$ of optimal