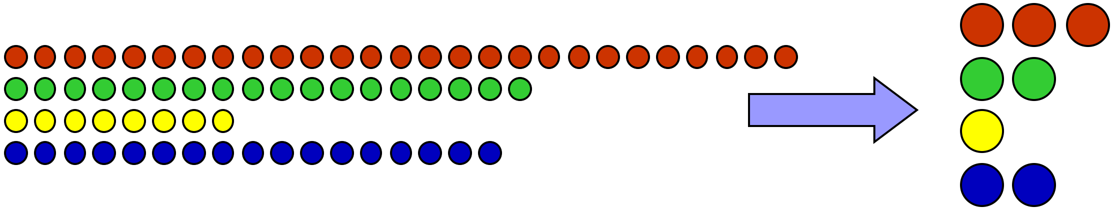
Streams, Sketching and ~~Databases~~ Big Data



Graham Cormode

University of Warwick

G.Cormode@Warwick.ac.uk

Big Data

- "Big" data arises in many forms:
  - Physical Measurements: from science (physics, astronomy)
  - Medical data: genetic sequences, detailed time series
  - Activity data: GPS location, social network activity
  - Business data: customer behavior tracking at fine detail
- Common themes:
  - Data is large, and growing
  - There are important patterns and trends in the data
  - We don't fully know how to find them

Making sense of Big Data

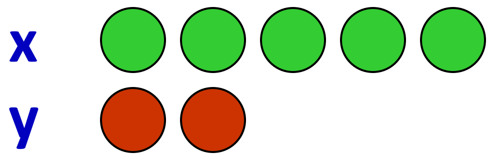
- Want to be able to interrogate data in different use-cases:
  - Routine Reporting: standard set of queries to run
  - Analysis: ad hoc querying to answer 'data science' questions
  - Monitoring: identify when current behavior differs from old
  - Mining: extract new knowledge and patterns from data
- In all cases, need to answer certain basic questions quickly:
  - Describe the distribution of particular attributes in the data
  - How many (distinct) X were seen?
  - How many X < Y were seen?
  - Give some representative examples of items in the data

Big Data and Hashing

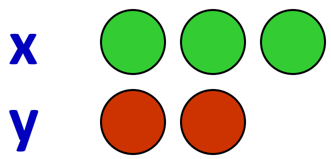
- "Traditional" hashing: compact storage of data
  - Hash tables proportional to data size
  - Fast, compact, exact storage of data
- Hashing with small probability of collisions: very compact storage
  - Bloom filters (no false negatives, bounded false positives)
  - Faster, compacter, probabilistic storage of data
- Hashing with almost certainty of collisions
  - Sketches (items collide, but the signal is preserved)
  - Fasterer, compacterer, approximate storage of data
  - Enables "small summaries for big data"

Data Models

- We model data as a collection of simple tuples
- Problems hard due to scale and dimension of input
- Arrivals only model:
  - Example: (x, 3), (y, 2), (x, 2) encodes the arrival of 3 copies of item x, 2 copies of y, then 2 copies of x.



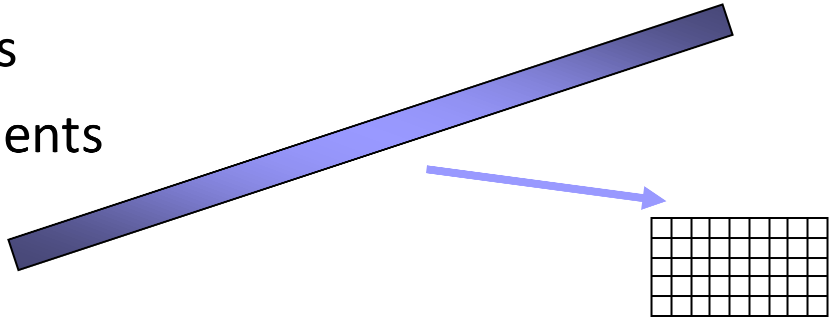
  - Could represent eg. packets on a network; power usage
- Arrivals and departures:
  - Example: (x, 3), (y,2), (x, -2) encodes final state of (x, 1), (y, 2).



  - Can represent fluctuating quantities, or measure differences between two distributions

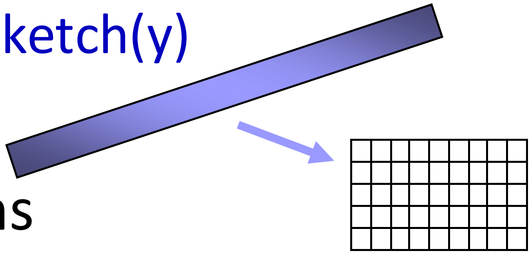
Sketches and Frequency Moments

- Sketches as hash-based linear transforms of data
- Frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



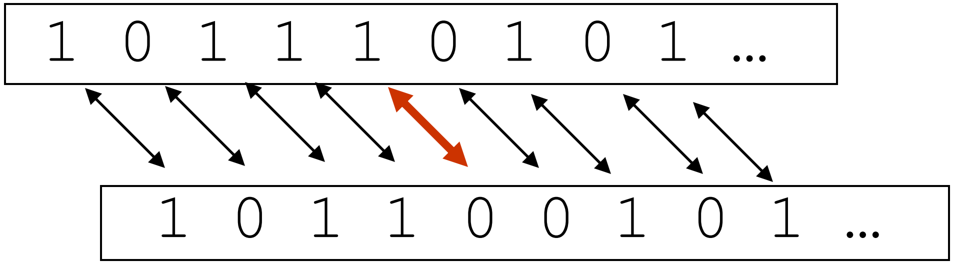
Sketch Structures

- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



- Often describe S in terms of hash functions
  - If hash functions are simple, sketch is fast
- Aim for limited independence hash functions h: [n] $\rightarrow$ [m]
  - If $\Pr_{h \in H}[\, h(i_1)=j_1 \wedge h(i_2)=j_2 \wedge \ldots h(i_k)=j_k \,] = m^{-k}$, then H is k-wise independent family ("h is k-wise independent")
  - k-wise independent hash functions take time, space O(k)

Fingerprints as sketches



- Test if two binary streams are equal

  $d_=(x,y) = 0$ iff $x=y$, 1 otherwise

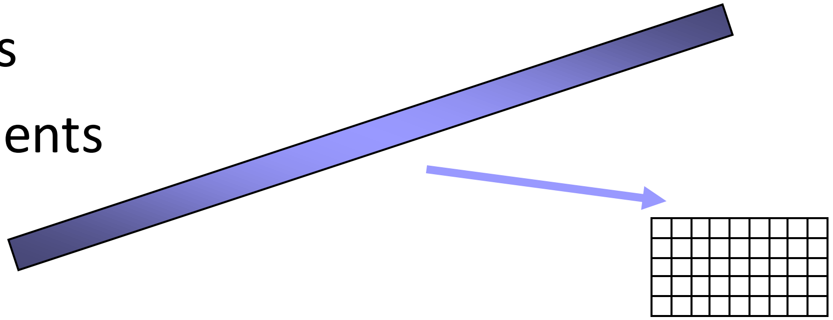
- To test in small space: pick a suitable hash function h
- Test $h(x)=h(y)$: small chance of false positive, no chance of false negative
- Compute $h(x)$, $h(y)$ incrementally as new bits arrive
  - How to choose the function $h()$?

Polynomial Fingerprints

- Pick $h(x) = \sum_{i=1}^{n} x_i\, r^i \bmod p$ for prime p, random $r \in \{1 \ldots p-1\}$
- Why?
- Flexible: $h(x)$ is linear function of x—easy to update and merge
- For accuracy, note that computation $\bmod p$ is over the field Z_p
  - Consider the polynomial in α, $\sum_{i=1}^{n} (x_i - y_i)\, \alpha^i = 0$
  - Polynomial of degree n over Z_p has at most n roots
- Probability that r happens to solve this polynomial is n/p
- So $\Pr[\, h(x) = h(y) \mid x \neq y \,] \leq n/p$
  - Pick $p = \mathrm{poly}(n)$, fingerprints are $\log p = O(\log n)$ bits
- Fingerprints applied to small subsets of data to test equality
  - Will see several examples that use fingerprints as subroutine

Sketches and Frequency Moments

- Sketches as hash-based linear transforms of data
- Frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



Frequency Distributions

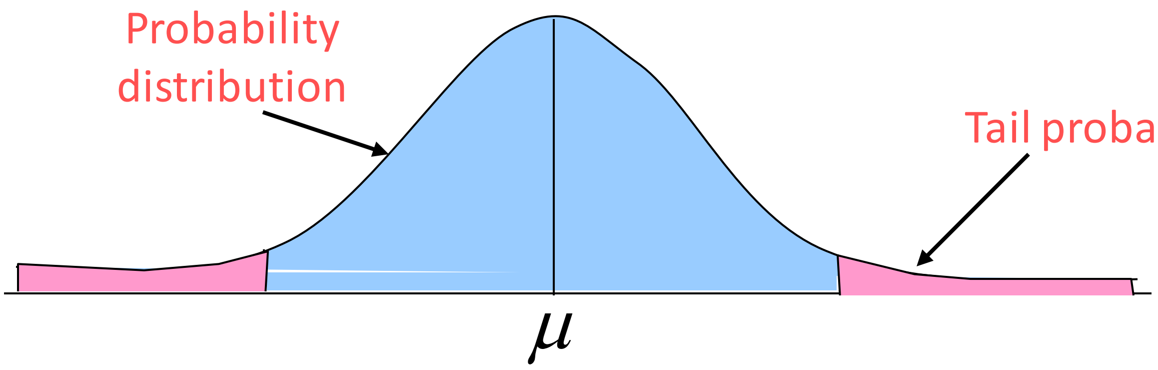
- Given set of items, let f_i be the number of occurrences of item i
- Many natural questions on f_i values:
  - Find those i's with large f_i values (heavy hitters)
  - Find the number of non-zero f_i values (count distinct)
  - Compute $F_k = \sum_i (f_i)^k$ – the k'th Frequency Moment
  - Compute $H = \sum_i (f_i/F_1) \log (F_1/f_i)$ – the (empirical) entropy
- "Space Complexity of the Frequency Moments"
  Alon, Matias, Szegedy in STOC 1996
  - Awarded Gödel prize in 2005
  - Set the pattern for many streaming algorithms to follow

Concentration Bounds

- Will provide randomized algorithms for these problems
- Each algorithm gives a (randomized) estimate of the answer
- Give confidence bounds on the final estimate X
  - Use probabilistic concentration bounds on random variables
- A concentration bound is typically of the form

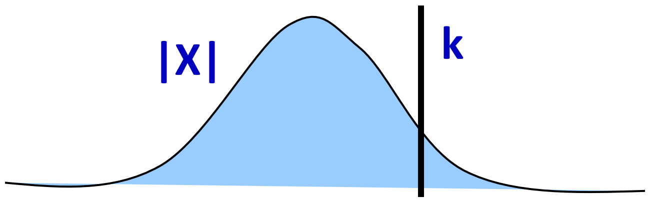
$$\Pr[\ |X - x| > \varepsilon y\ ] < \delta$$

  - At most probability δ of being more than εy away from x



Markov Inequality

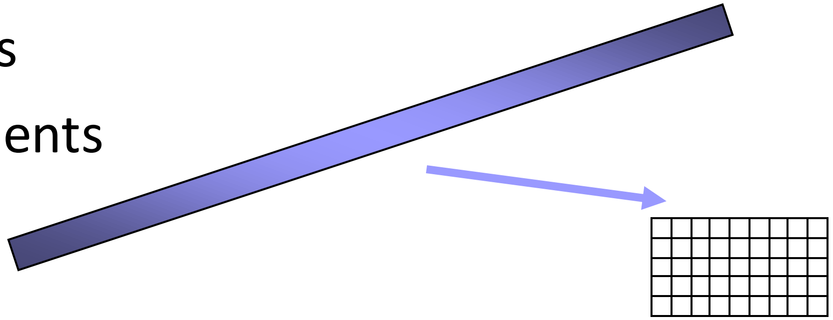
- Take *any* probability distribution X s.t. $\Pr[X < 0] = 0$
- Consider the event $X \geq k$ for some constant $k > 0$
- For any draw of X, $kI(X \geq k) \leq X$
  - Either $0 \leq X < k$, so $I(X \geq k) = 0$
  - Or $X \geq k$, lhs = k



- Take expectations of both sides: $k \Pr[X \geq k] \leq E[X]$
- Markov inequality: $\Pr[X \geq k] \leq E[X]/k$
  - Prob of random variable exceeding k times its expectation $< 1/k$
  - Relatively weak in this form, but still useful

Sketches and Frequency Moments

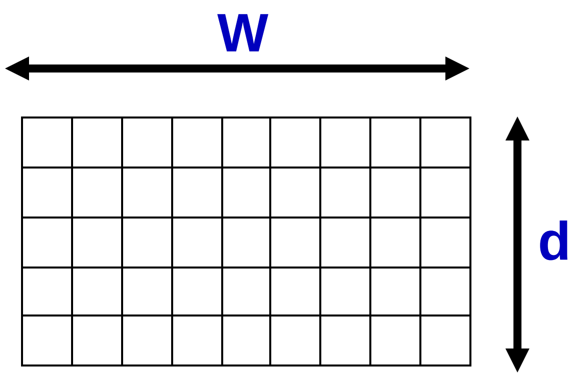
- Sketches as hash-based linear transforms of data
- Frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



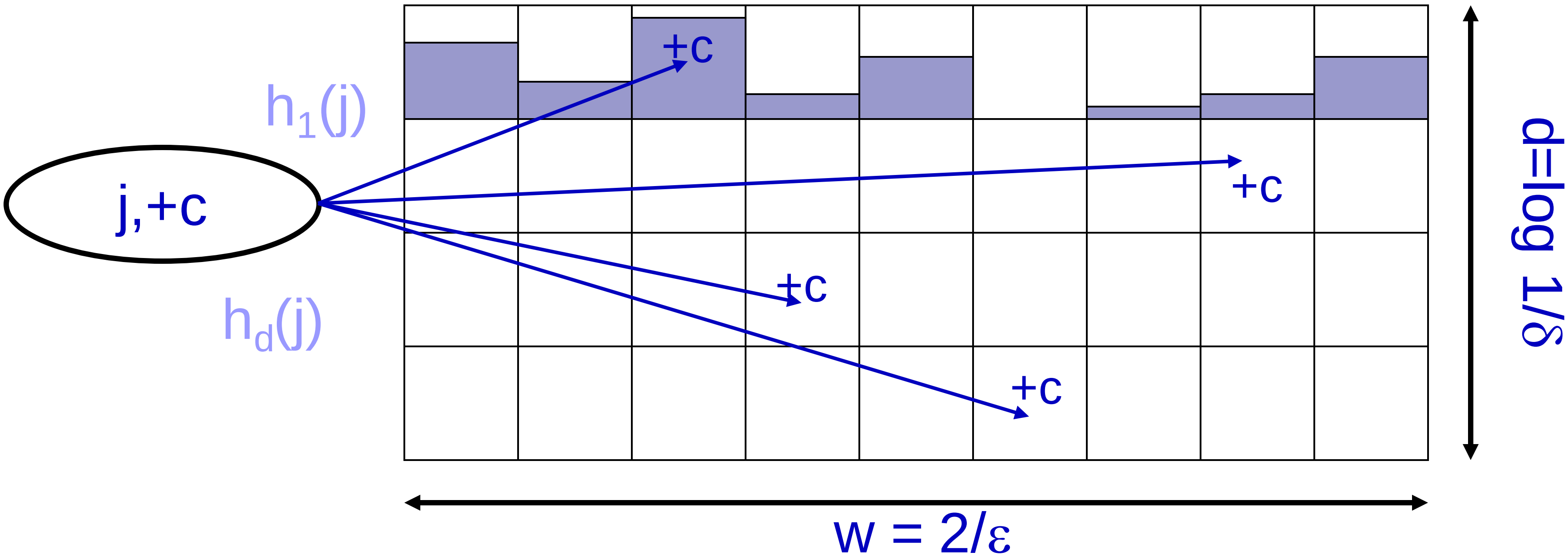
Count-Min Sketch

- Simple sketch idea relies primarily on Markov inequality
- Model input data as a vector x of dimension U
- Creates a small summary as an array of w $\times$ d in size
- Use d hash function to map vector entries to [1..w]
- Works on arrivals only and arrivals & departures streams

Array:
CM[i,j]



Count-Min Sketch Structure



- Each entry in vector x is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Estimate $x[j]$ by taking $\min_k CM[k,h_k(j)]$
  - Guarantees error less than εF_1 in size $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$

[C, Muthukrishnan '04]

Approximation of Point Queries

Approximate point query $x'[j] = \min_k CM[k,h_k(j)]$

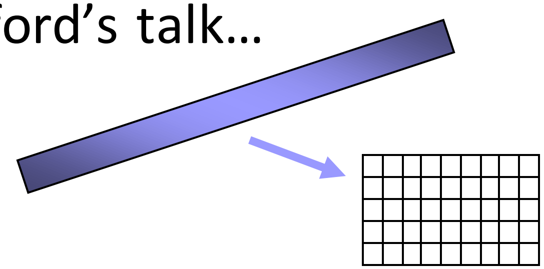
- Analysis: In k'th row, $CM[k,h_k(j)] = x[j] + X_{k,j}$
  - $X_{k,j} = \Sigma_i\, x[i]\, I(h_k(i) = h_k(j))$
  - $E[X_{k,j}] \quad = \Sigma_{i \neq j}\, x[i]*Pr[h_k(i)=h_k(j)]$
    $\leq Pr[h_k(i)=h_k(j)] * \Sigma_i\, x[i]$
    $= \varepsilon\, F_1/2$ – requires only pairwise independence of h
  - $Pr[X_{k,j} \geq \varepsilon F_1] = Pr[\, X_{k,j} \geq 2E[X_{k,j}]\,] \leq 1/2$ by Markov inequality
- So, $Pr[x'[j] \geq x[j] + \varepsilon F_1] = Pr[\forall\ k.\ X_{k,j} > \varepsilon F_1] \leq 1/2^{\log 1/\delta} = \delta$
- Final result: with certainty $x[j] \leq x'[j]$ and
  with probability at least $1-\delta$, $x'[j] < x[j] + \varepsilon F_1$

Applications of Count-Min to Heavy Hitters

- Count-Min sketch lets us estimate f_i for any i (up to εF_1)
- Heavy Hitters asks to find i such that f_i is large ($> \phi F_1$)
- Slow way: test every i after creating sketch
- Alternate way:
  - Keep binary tree over input domain: each node is a subset
  - Keep sketches of all nodes at same level
  - Descend tree to find large frequencies, discard 'light' branches
  - Same structure estimates arbitrary range sums
- A first step towards compressed sensing style results...

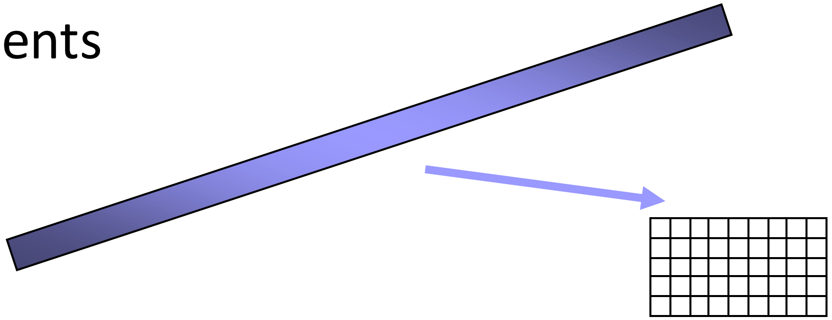
Application to Large Scale Machine Learning

- In machine learning, often have very large feature space
  - Many objects, each with huge, sparse feature vectors
  - Slow and costly to work in the full feature space
- "Hash kernels": work with a sketch of the features
  - Effective in practice! [Weinberger, Dasgupta, Langford, Smola, Attenberg '09]
- Similar analysis explains *why:*
  - Essentially, not too much noise on the important features
  - See John Langford's talk…



Sketches and Frequency Moments

- Frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments

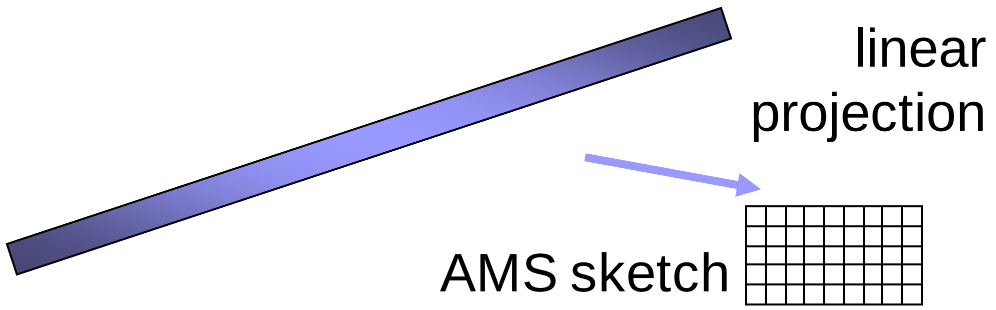


Chebyshev Inequality

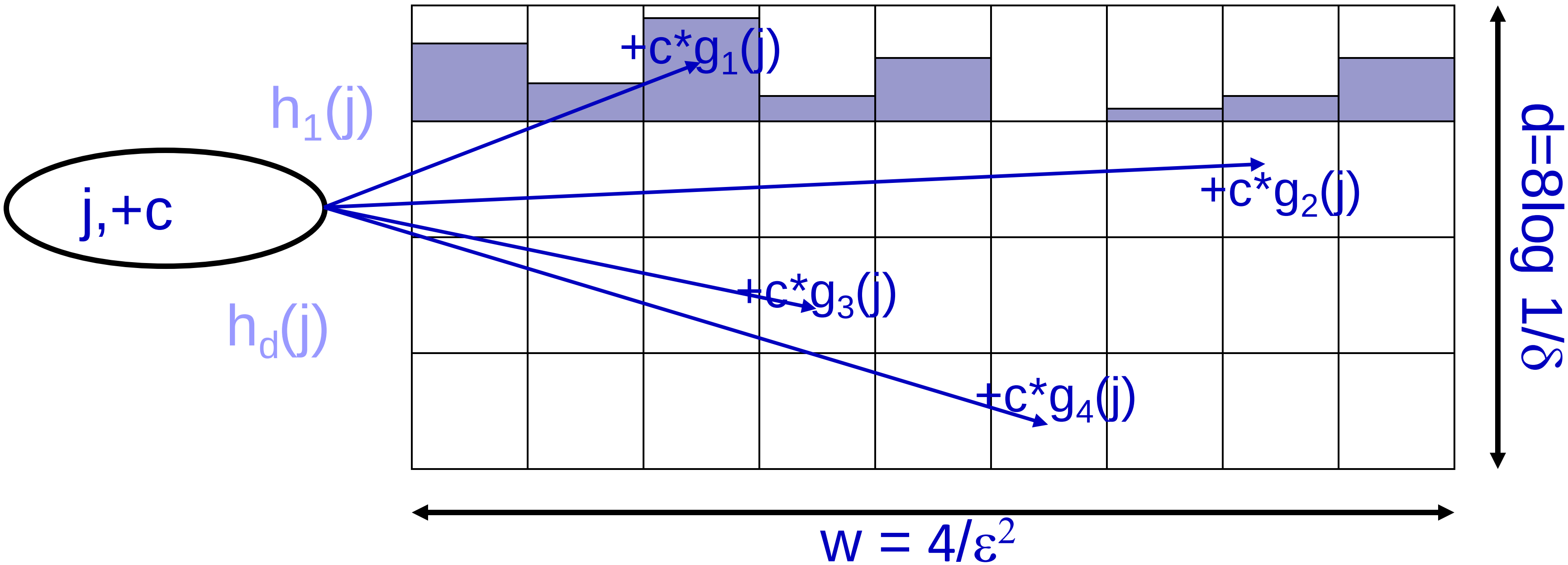
- Markov inequality applied directly is often quite weak
- But Markov inequality holds for any random variable
- Can apply to a random variable that is a function of X
- Set $Y = (X - E[X])^2$
- By Markov, $\Pr[\, Y > kE[Y] \,] < 1/k$
  - $E[Y] = E[(X-E[X])^2] = \mathrm{Var}[X]$
- Hence, $\Pr[\, |X - E[X]| > \sqrt{k\,\mathrm{Var}[X]} \,] < 1/k$
- Chebyshev inequality: $\Pr[\, |X - E[X]| > k \,] < \mathrm{Var}[X]/k^2$
  - If $\mathrm{Var}[X] \leq \varepsilon^2 E[X]^2$, then $\Pr[\,|X - E[X]| > \varepsilon E[X]\,] = O(1)$

F_2 estimation

- AMS sketch (for Alon-Matias-Szegedy) proposed in 1996
  - Allows estimation of F_2 (second frequency moment)
  - Used at the heart of many streaming and non-streaming applications: achieves dimensionality reduction
- Here, describe AMS sketch by generalizing CM sketch.
- Uses extra hash functions $g_1 \ldots g_{\log 1/\delta}$ $\{1 \ldots U\} \rightarrow \{+1,-1\}$
  - (Low independence) Rademacher variables
- Now, given update $(j,+c)$, set $CM[k,h_k(j)] \mathrel{+}= c*g_k(j)$



F_2 analysis



- Estimate $F_2 = \text{median}_k \sum_i CM[k,i]^2$
- Each row's result is $\sum_i g(i)^2 x[i]^2 + \sum_{h(i)=h(j)} 2\, g(i)\, g(j)\, x[i]\, x[j]$
- But $g(i)^2 = -1^2 = +1^2 = 1$, and $\sum_i x[i]^2 = F_2$
- $g(i)g(j)$ has 1/2 chance of +1 or –1 : expectation is 0 …

F_2 Variance

- Expectation of row estimate $R_k = \sum_i CM[k,i]^2$ is exactly F_2
- Variance of row k, $Var[R_k]$, is an expectation:
  - $Var[R_k] = E[\ (\sum_{\text{buckets } b} (CM[k,b])^2 - F_2)^2\ ]$
  - Good exercise in algebra: expand this sum and simplify
  - Many terms are zero in expectation because of terms like $g(a)g(b)g(c)g(d)$ (degree at most 4)
  - Requires that hash function g is *four-wise independent*: it behaves uniformly over subsets of size four or smaller
    - Such hash functions are easy to construct

F_2 Variance

- Terms with odd powers of $g(a)$ are zero in expectation
  - $g(a)g(b)g^2(c)$, $g(a)g(b)g(c)g(d)$, $g(a)g^3(b)$
- Leaves

$$
\begin{aligned}
Var[R_k] \leq \textstyle\sum_i g^4(i)\, x[i]^4 \\
&+ 2 \textstyle\sum_{j\neq i} g^2(i)\, g^2(j)\, x[i]^2\, x[j]^2 \\
&+ 4 \textstyle\sum_{h(i)=h(j)} g^2(i)\, g^2(j)\, x[i]^2\, x[j]^2 \\
&- (x[i]^4 + \textstyle\sum_{j\neq i} 2x[i]^2\, x[j]^2) \\
&\leq F_2^{\,2}/w
\end{aligned}
$$

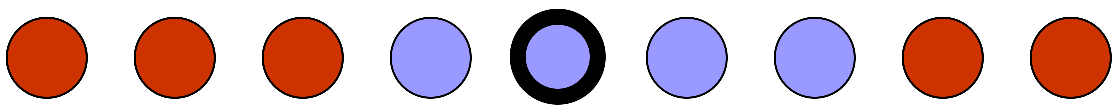
- Row variance can finally be bounded by $F_2^{\,2}/w$
  - Chebyshev for $w=4/\varepsilon^2$ gives probability ¼ of failure:
    $$\Pr[\ |R_k - F_2| > \varepsilon^2 F_2\ ] \leq ¼$$
  - How to amplify this to small δ probability of failure?
  - Rescaling w has cost linear in $1/\delta$

Tail Inequalities for Sums

- We achieve stronger bounds on tail probabilities for the sum of independent *Bernoulli trials* via the Chernoff Bound:
  - Let $X_1, \ldots, X_m$ be independent Bernoulli trials s.t. $\Pr[X_i=1] = p$ ($\Pr[X_i=0] = 1-p$).
  - Let $X = \sum_{i=1}^m X_i$,and $\mu = mp$ be the expectation of X.
  - $\boxed{\Pr[X > (1+\varepsilon)\mu]} = \Pr[\exp(tX) > \exp(t(1+\varepsilon)\mu)] \leq E[\exp(tX)]/\exp(t(1+\varepsilon)\mu)$
  - $E[\exp(tX)] = \prod_i E[\exp(tX_i)] = \prod_i (1-p + pe^t) \leq \prod_i \exp(p\,(e^t-1))$
    $= \exp(\mu(e^t - 1))$
  - $\Pr[X > (1+\varepsilon)\mu] \leq \exp(\mu(e^t - 1) - \mu t(1+\varepsilon)) = \exp(\mu(-\varepsilon t + t^2/2 + t^3/6 + \ldots))$
    $\leq \exp(\mu(t^2/2 - \varepsilon t))$
  - Balance: choose $t=\varepsilon/2$
    $\boxed{\leq \exp(-\mu\, \varepsilon^2/2)}$

Applying Chernoff Bound

- Each row gives an estimate that is within ε relative error with probability $p' > 3/4$
- Take d repetitions and find the median. Why the median?



  – Because bad estimates are either too small or too large
  – Good estimates form a contiguous group "in the middle"
  – At least $d/2$ estimates must be bad for median to be bad
- Apply Chernoff bound to d independent estimates, $p=1/4$
  – Pr[More than $d/2$ bad estimates] $< 2\exp(-d/8)$
  – So we set $d = \Theta(\ln 1/\delta)$ to give δ probability of failure
- Same outline used many times in summary construction

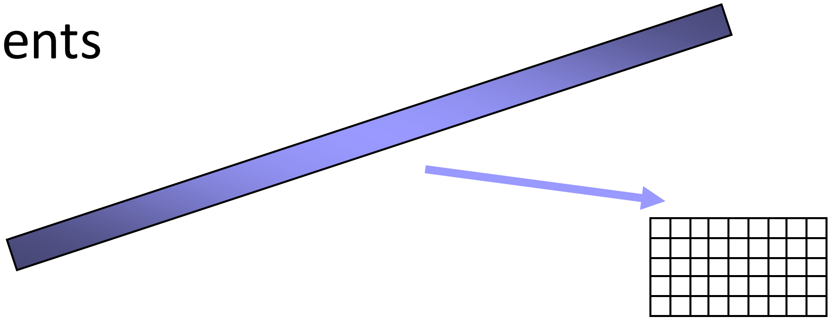
Applications and Extensions

- F_2 guarantee: estimate $\|x\|_2$ from sketch with error $\varepsilon \|x\|_2$
  - Since $\|x + y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 + 2x \cdot y$
    Can estimate $(x \cdot y)$ with error $\varepsilon\|x\|_2\|y\|_2$
  - If $y = e_j$, obtain $(x \cdot e_j) = x_j$ with error $\varepsilon \|x\|_2$:
    L_2 guarantee ("Count Sketch") vs L_1 guarantee (Count-Min)

- Can view the sketch as a low-independence realization of the Johnson-Lindendestraus lemma
  - Best current JL methods have the same structure
  - JL is stronger: embeds directly into Euclidean space
  - JL is also weaker: requires $O(1/\varepsilon)$-wise hashing, $O(\log 1/\delta)$ independence [Nelson, Nguyen 13]

Sketches and Frequency Moments

- Frequency Moments and Sketches
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments

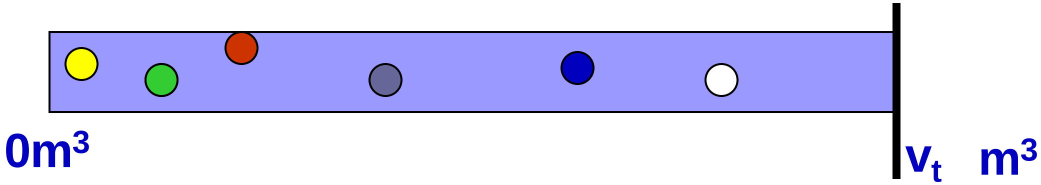


F_0 Estimation

- F_0 is the number of distinct items in the stream
  - a fundamental quantity with many applications
- Early algorithms by Flajolet and Martin [1983] gave nice hashing-based solution
  - analysis assumed fully independent hash functions
- Will describe a generalized version of the FM algorithm due to Bar-Yossef et. al with only pairwise indendence
  - Known as the "k-Minimum values (KMV)" algorithm

F_0 Algorithm

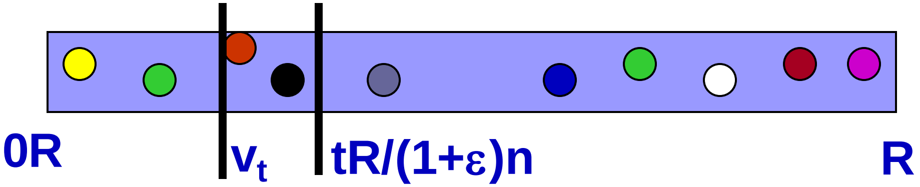
- Let m be the domain of stream elements
  - Each item in data is from $[1\ldots m]$
- Pick a random (pairwise) hash function $h: [m] \rightarrow [R]$
  - For $R = m^3$ with probability at least $1-1/m$, no collisions under h



- For each stream item i, compute $h(i)$, and track the t distinct items achieving the smallest values of $h(i)$
  - Note: if same i is seen many times, $h(i)$ is same
  - Let v_t = t'th smallest (distinct) value of $h(i)$ seen
- If $n = F_0 < t$, give exact answer, else estimate $F'_0 = tR/v_t$
  - $v_t/R \approx$ fraction of hash domain occupied by t smallest

Analysis of F_0 algorithm

- Suppose $F'_0 = tR/v_t > (1+\varepsilon)\ n$ [estimate is too high]



- So for input = set $S \in 2^{[m]}$, we have
  - $|\{ s \in S \mid h(s) < tR/(1+\varepsilon)n \}| > t$
  - Because $\varepsilon < 1$, we have $tR/(1+\varepsilon)n \leq (1-\varepsilon/2)tR/n$
  - $\Pr[\ h(s) < (1-\varepsilon/2)tR/n] \approx 1/R * (1-\varepsilon/2)tR/n = (1-\varepsilon/2)t/n$

  - (this analysis outline hides some rounding issues)

Chebyshev Analysis

- Let Y be number of items hashing to under $tR/(1+\varepsilon)n$
  - $E[Y] = n * \Pr[\ h(s) < tR/(1+\varepsilon)n] = (1-\varepsilon/2)t$
  - For each item i, variance of the event $= p(1-p) < p$
  - $\mathrm{Var}[Y] = \sum_{s \in S} \mathrm{Var}[\ h(s) < tR/(1+\varepsilon)n] < (1-\varepsilon/2)t$
    - We sum variances because of pairwise independence

- Now apply Chebyshev inequality:
  - $$\begin{aligned} \Pr[\ Y > t\ ] &\leq \Pr[|Y - E[Y]| > \varepsilon t/2] \\ &\leq 4\mathrm{Var}[Y]/\varepsilon^2 t^2 \\ &< 4t/(\varepsilon^2 t^2) \end{aligned}$$
  - Set $t=20/\varepsilon^2$ to make this $\mathrm{Prob} \leq 1/5$

Completing the analysis

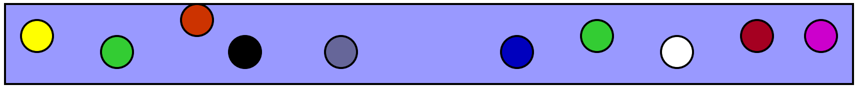
- We have shown

  $\Pr[\, F'_0 > (1+\varepsilon)\, F_0 \,] < 1/5$
- Can show $\Pr[\, F'_0 < (1-\varepsilon)\, F_0 \,] < 1/5$ similarly
  – too few items hash below a certain value
- So $\Pr[\, (1-\varepsilon)\, F_0 \le F'_0 \le (1+\varepsilon) F_0] > 3/5$ [Good estimate]

- Amplify this probability: repeat $O(\log 1/\delta)$ times in parallel with different choices of hash function h
  – Take the median of the estimates, analysis as before

F_0 Issues

- Space cost:
  – Store t hash values, so $O(1/\varepsilon^2 \log m)$ bits
  – Can improve to $O(1/\varepsilon^2 + \log m)$ with additional tricks



- Time cost:
  – Find if hash value $h(i) < v_t$
  – Update v_t and list of t smallest if $h(i)$ not already present
  – Total time $O(\log 1/\varepsilon + \log m)$ worst case

Count-Distinct

- Engineering the best constants: Hyperloglog algorithm
  - Hash each item to one of $1/\varepsilon^2$ buckets (like Count-Min)
  - In each bucket, track the function $\max \lfloor \log(h(x)) \rfloor$
    - Can view as a coarsened version of KMV
    - Space efficient: need $\log \log m \approx 6$ bits per bucket
- Can estimate intersections between sketches
  - Make use of identity $|A \cap B| = |A| + |B| - |A \cup B|$
  - Error scales with $\varepsilon \sqrt{(|A||B|)}$, so poor for small intersections
  - Higher order intersections via inclusion-exclusion principle

Subset Size Estimation from KMV

- Want to estimate the fraction $f = |A|/|S|$
  - S is the observed set of data
  - A is an arbitrary subset given later
  - E.g. fraction of customers who are female 18-24 from Denmark
- Simple algorithm:
  - Run KMV to get sample set K, estimate $f' = |A \cap K|/k$
  - Need to bound probability of getting a bad estimate
  - Analysis due to [Thorup 13]

Subset Size Estimation

- Upper bound:
  - Suppose we overestimate: $|A \cap K| > (1 + a) / (1 – b)\, fk$
  - Set threshold $t = kR/(n(1\text{-}a))$
- To have overestimate, must have one of:
  1. Fewer than k elements from B hash below t : expect $k/(1\text{-}a)$
  2. More than $(1+b)(kf)/(1\text{-}a)$ elements from A hash below t: expect $kf/(1\text{-}a)$
  - Otherwise, cannot have overestimate
- To analyze, bound the probability of 1. and 2. separately
  - Probability of overestimate is bounded by sum of these probs

Bounding error probability

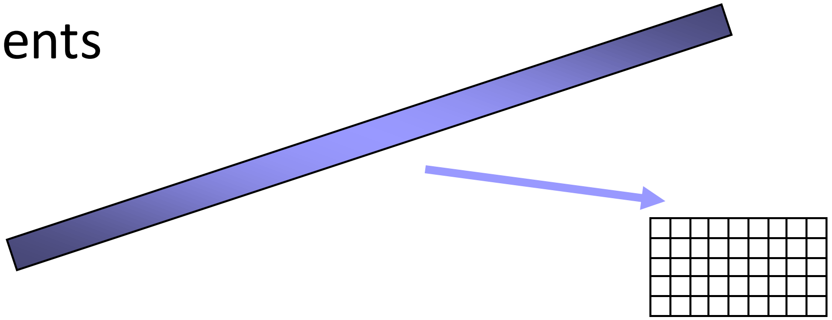
- Use Chebyshev to bound the two bad cases
  - Suppose mean number of m hash values below a threshold $\mu = mp$
  - Standard deviation $\sigma = ((1-p)pm)^{½} \leq \mu^{½}$ (via pairwise independence)
  - Set $a = 4/\sqrt{k}$, $b = 4/\sqrt{(fk)}$
  - For Event 1., we have $\mu = k/(1-a) \geq k$ so, via Chebyshev, $\Pr[\text{ Event } 1.] \leq \mu/a\sigma < 1/16$
  - Similarly, for Event 2., we have $\mu = kf/(1-a) \geq kf$ so $\Pr[\text{Event } 2.] \leq \mu/b\sigma < 1/16$
  - By union bound, at most 1/8 prob of overestimate
- Similar case analysis for the case of an underestimate

Subset count accuracy

- With probability at least ¾, the error is $O((fk)^{½})$
  - Arises from the choice of parameters b and a
  - Error scales with f
- For some lower bound on f, f', can get relative error ε:
  - Set $k \propto f'/\varepsilon^2$ for $(1 \pm \varepsilon)$ error with constant probability
- For improved error:
  - Either increase $k \propto 1/\delta$
  - Or repeat $\log 1/\delta$ times and take median estimate

Frequency Moments

- Intro to frequency distributions and Concentration bounds
- Count-Min sketch for F_∞ and frequent items
- AMS Sketch for F_2
- Estimating F_0
- Extensions:
  - Higher frequency moments
  - Combined frequency moments



Higher Frequency Moments

- F_k for $k>2$. Use a sampling trick [Alon et al 96]:
  - Uniformly pick an item from the stream length $1 \dots n$
  - Set r = how many times that item appears subsequently
  - Set estimate $F'_k = n(r^k - (r-1)^k)$

- $E[F'_k] = 1/n * n * [\, f_1^k - (f_1-1)^k + (f_1-1)^k - (f_1-2)^k + \dots + 1^k - 0^k] + \dots$
  $= f_1^k + f_2^k + \dots = F_k$
- $Var[F'_k] \leq 1/n * n^2 * [(f_1^k - (f_1-1)^k)^2 + \dots]$
  - Use various bounds to bound the variance by $k\, m^{1-1/k} F_k^2$
  - Repeat $k\, m^{1-1/k}$ times in parallel to reduce variance
- Total space needed is $O(k\, m^{1-1/k})$ machine words
  - Not a sketch: does not distribute easily. See next lecture!

Combined Frequency Moments

- Let $G[i,j] = 1$ if (i,j) appears in input.
  E.g. graph edge from i to j. Total of m distinct edges
- Let $d_i = \Sigma_{j=1}^{n} G[i,j]$ (aka degree of node i)
- Find aggregates of d_i's:
  - Estimate heavy d_i's (people who talk to many)
  - Estimate frequency moments:
    number of distinct d_i values, sum of squares
  - Range sums of d_i's (subnet traffic)
- Approach: nest one sketch inside another, e.g. HLL inside CM
  - Requires new analysis to track overall error

Range Efficiency

- Sometimes input is specified as a collection of ranges [a,b]
  - [a,b] means insert all items (a, a+1, a+2 … b)
  - Trivial solution: just insert each item in the range
- Range efficient F_0 [Pavan, Tirthapura 05]
  - Start with an alg for F_0 based on pairwise hash functions
  - Key problem: track which items hash into a certain range
  - Dives into hash fns to divide and conquer for ranges
- Range efficient F_2 [Calderbank et al. 05, Rusu,Dobra 06]
  - Start with sketches for F_2 which sum hash values
  - Design new hash functions so that range sums are fast
- Rectangle Efficient F_0 [Tirthapura, Woodruff 12]

Summary

- Sketching Techniques summarize large data sets
- Summarize vectors:
  - Test equality (fingerprints)
  - Recover approximate entries (count-min, count sketch)
  - Approximate Euclidean norm (F_2) and dot product
  - Approximate number of non-zero entries (F_0)
  - Approximate set membership (Bloom filter)

Current Directions in Streaming and Sketching

- Sparse representations of high dimensional objects
  - Compressed sensing, sparse fast fourier transform
- Numerical linear algebra for (large) matrices
  - k-rank approximation, linear regression, PCA, SVD, eigenvalues
- Computations on large graphs
  - Sparsification, clustering, matching
- Geometric (big) data
  - Coresets, facility location, optimization, machine learning
- Use of summaries in distributed computation
  - MapReduce, Continuous Distributed models