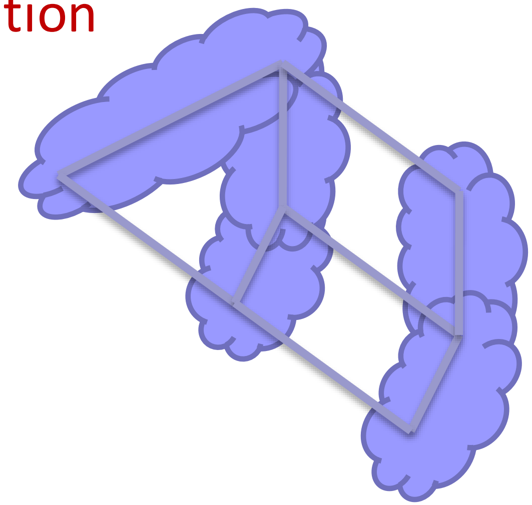
Recap

- Hashing-based sketch 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)

Advanced Topics

- L_p Sampling
  - L_0 sampling and graph sketching
  - L_2 sampling and frequency moment estimation
- Matrix computations
  - Sketches for matrix multiplication
  - Compressed matrix multiplication
- Hashing to check computation
  - Matrix product checking
  - Vector product checking
- Lower bounds for streaming and sketching
  - Basic hard problems (Index, Disjointness)
  - Hardness via reductions



Sampling from Sketches

- Given inputs with positive and negative weights
- Want to sample based on the overall frequency distribution
  - Sample from support set of n possible items
  - Sample proportional to (absolute) weights
  - Sample proportional to some function of weights
- How to do this sampling effectively?
- Recent approach: L_p sampling

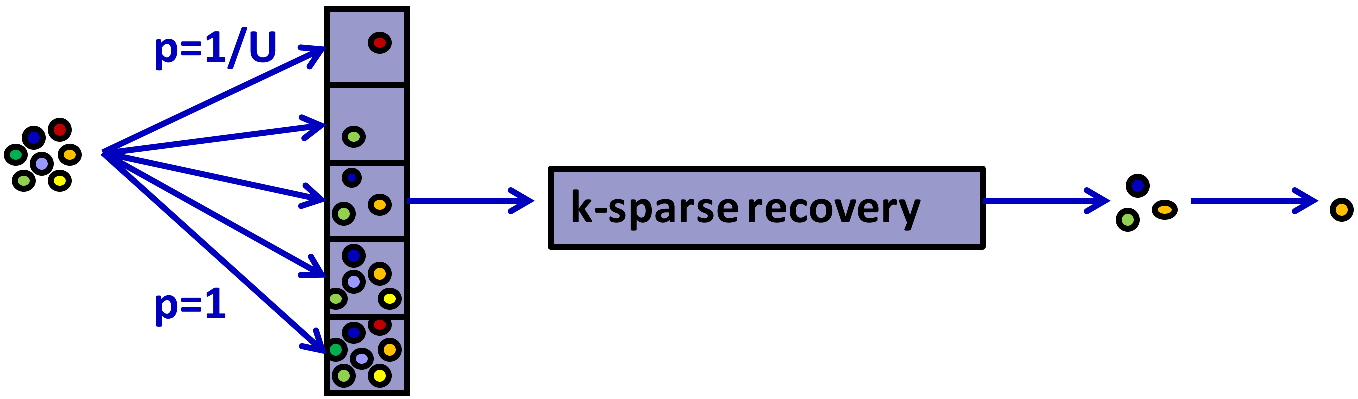
L_p Sampling

- L_p sampling: use sketches to sample i w/prob $(1\pm\varepsilon)\ f_i^p/\|f\|_p^p$
- "Efficient" solutions developed of size $O(\varepsilon^{-2} \log^2 n)$
  - [Monemizadeh, Woodruff 10] [Jowhari, Saglam, Tardos 11]
- L_0 sampling enables novel "graph sketching" techniques
  - Sketches for connectivity, sparsifiers [Ahn, Guha, McGregor 12]
- L_2 sampling allows optimal estimation of frequency moments

L_0 Sampling

- L_0 sampling: sample with prob $(1\pm\varepsilon)\ f_i^0/F_0$
  - i.e., sample (near) uniformly from items with non-zero frequency
- General approach: [Frahling, Indyk, Sohler 05, C., Muthu, Rozenbaum 05]
  - Sub-sample all items (present or not) with probability p
  - Generate a sub-sampled vector of frequencies f_p
  - Feed f_p to a *k-sparse recovery* data structure
    - Allows reconstruction of f_p if $F_0 < k$
  - If f_p is k-sparse, sample from reconstructed vector
  - Repeat in parallel for exponentially shrinking values of p

Sampling Process



- Exponential set of probabilities, $p=1, \frac{1}{2}, \frac{1}{4}, 1/8, 1/16 \ldots 1/U$
  - Let $N = F_0 = |\{ i : f_i \neq 0\}|$
  - Want there to be a level where k-sparse recovery will succeed
  - At level p, expected number of items selected S is Np
  - Pick level p so that $k/3 < Np \leq 2k/3$
- Chernoff bound: with probability exponential in k, $1 \leq S \leq k$
  - Pick $k = O(\log 1/\delta)$ to get $1-\delta$ probability

k-Sparse Recovery

- Given vector x with at most k non-zeros, recover x via sketching
  - A core problem in compressed sensing/compressive sampling
- First approach: Use Count-Min sketch of x
  - Probe all U items, find those with non-zero estimated frequency
  - Slow recovery: takes O(U) time
- Faster approach: also keep sum of item identifiers in each cell
  - Sum/count will reveal item id
  - Avoid false positives: keep fingerprint of items in each cell
- Can keep a sketch of size O(k log U) to recover up to k items

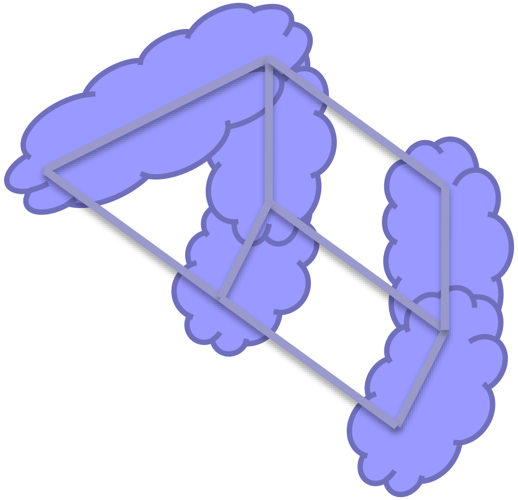
Sum, $\sum_{i : h(i)=j} i$
Count, $\sum_{i : h(i)=j} x_i$
Fingerprint, $\sum_{i : h(i)=j} x_i r^i$

Uniformity

- Also need to argue sample is uniform
  - Failure to recover could bias the process
- Pr[i would be picked if k=n] = $1/F_0$ by symmetry
- Pr[i is picked] = Pr[i would be picked if k=n $\wedge$ S$\leq$ k]
  $\geq (1-\delta)/F_0$
- So $(1-\delta)/N \leq$ Pr[i is picked] $\leq 1/N$
- Sufficiently uniform (pick $\delta = \varepsilon$)

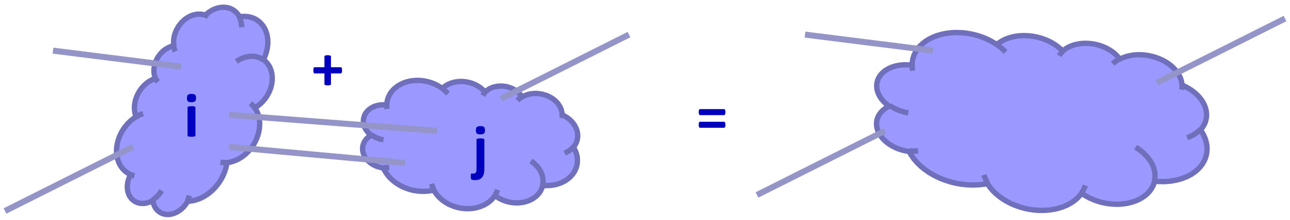
Application: Graph Sketching

- Given L_0 sampler, use to sketch (undirected) graph properties
- Connectivity: want to test if there is a path between all pairs
- Basic alg: repeatedly contract edges between components
- Use L_0 sampling to provide edges on vector of adjacencies
- Problem: as components grow, sampling most likely to produce internal links



Graph Sketching

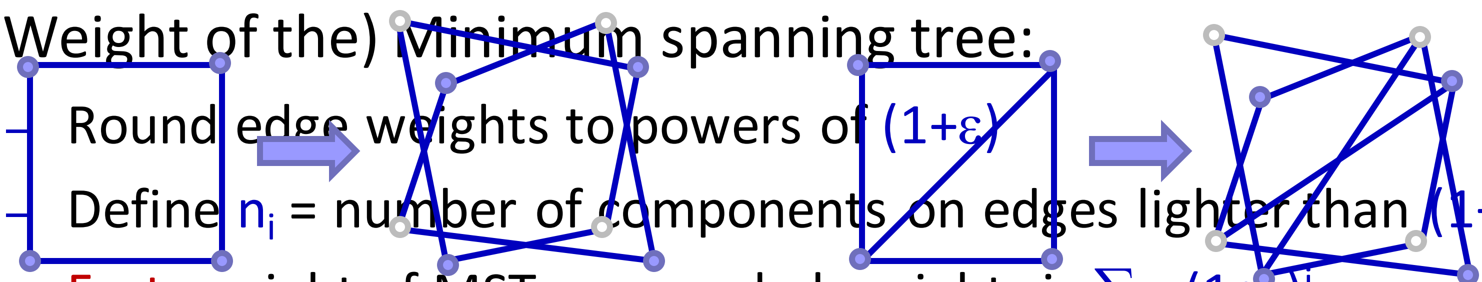
- Idea: use clever encoding of edges [Ahn, Guha, McGregor 12]
- Encode edge (i,j) as ((i,j),+1) for node i<j, as ((i,j),-1) for node j>i
- When node i and node j get merged, sum their L_0 sketches
  - Contribution of edge (i,j) exactly cancels out



- Only non-internal edges remain in the L_0 sketches
- Use independent sketches for each iteration of the algorithm
  - Only need O(log n) rounds with high probability
- Result: O(poly-log n) space per node for connectivity

Other Graph Results via sketching

- K-connectivity via connectivity
  - Use connectivity result to find and remove a spanning forest
  - Repeat k times to generate k spanning forests $F_1, F_2, \ldots F_k$
  - Theorem: G is k-connected if $\cup_{i=1}^{k} F_i$ is k-connected
- Bipartiteness via connectivity:
  - Compute c = number of connected components in G
  - Generate G' over $V \cup V'$ so $(u,v) \in E \Rightarrow (u, v') \in E', (u', v) \in E'$
  - If G is bipartite, G' has $2c$ components, else it has $<2c$ components
- (Weight of the) Minimum spanning tree:
  - Round edge weights to powers of $(1+\varepsilon)$
  - Define n_i = number of components on edges lighter than $(1+\varepsilon)^i$
  - Fact: weight of MST on rounded weights is $\sum_i \varepsilon(1+\varepsilon)^i n_i$



Application: F_k via L_2 Sampling

- Recall, $F_k = \sum_i f_i^k$
- Suppose L_2 sampling samples f_i with probability f_i^2/F_2
  – And also estimates sampled f_i with relative error ε
- Estimator: $X = F_2\, f_i^{k-2}$ (with estimates of F_2, f_i)
  – Expectation: $E[X] = F_2 \sum_i f_i^{k-2} \cdot f_i^2 / F_2 = F_k$
  – Variance: $Var[X] \leq E[X^2] = \sum_i f_i^2/F_2\, (F_2\, f_i^{k-2})^2 = F_2\, F_{2k-2}$

Rewriting the Variance

- Want to express variance $F_2\ F_{2k-2}$ in terms of F_k and domain size n
- Hölder's inequality: $\langle x, y\rangle \leq \|x\|_p\ \|y\|_q$ for $1 \leq p, q$ with $1/p+1/q=1$
  - Generalizes Cauchy-Shwarz inequality, where $p=q=2$.
- So pick $p=k/(k-2)$ and $q = k/2$ for $k > 2$. Then

$$\langle 1^n, (f_i)^2\rangle \leq \|1^n\|_{k/(k-2)}\ \|(f_i)^2\|_{k/2}$$

$$F_2 \leq n^{(k-2)/k}\ F_k^{2/k} \qquad (1)$$

- Also, since $\|x\|_{p+a} \leq \|x\|_p$ for any $p \geq 1$, $a > 0$
  - Thus $\|x\|_{2k-2} \leq \|x\|_k$ for $k \geq 2$
  - So $F_{2k-2} = \|f\|_{2k-2}^{2k-2} \leq \|f\|_k^{2k-2} = F_k^{2-2/k}$ (2)
- Multiply (1) * (2) : $F_2\ F_{2k-2} \leq n^{1-2/k}\ F_k^2$
  - So variance is bounded by $n^{1-2/k}\ F_k^2$

F_k Estimation

- For $k \geq 3$, we can estimate F_k via L_2 sampling:
  - Variance of our estimate is $O(F_k^2\ n^{1-2/k})$
  - Take mean of $n^{1-2/k}\varepsilon^{-2}$ repetitions to reduce variance
  - Apply Chebyshev inequality: constant prob of good estimate
  - Chernoff bounds: $O(\log 1/\delta)$ repetitions reduces prob to δ
- How to instantiate this?
  - Design method for approximate L_2 sampling via sketches
  - Show that this gives relative error approximation of f_i
  - Use approximate value of F_2 from sketch
  - Complicates the analysis, but bound stays similar

L_2 Sampling Outline

- For each i, draw u_i uniformly in the range $0\ldots1$
  - From vector of frequencies f, derive g so $g_i = f_i/\sqrt{u_i}$
  - Sketch g_i vector
- Sample: return (i, f_i) if there is unique i with $g_i^2 > t = F_2/\varepsilon$ threshold
  - $\Pr[\, g_i^2 > t \wedge \forall\, j \neq i : g_j^2 < t] = \Pr[g_i^2 > t] \prod_{j\neq i} \Pr[g_j^2 < t]$

    $= \Pr[u_i < \varepsilon f_i^2/F_2] \prod_{j\neq i} \Pr[u_j > \varepsilon f_j^2/F_2]$

    $= (\varepsilon f_i^2/F_2) \prod_{j\neq i} (1 - \varepsilon f_j^2/F_2)$

    $\approx \varepsilon f_i^2/F_2$
- Probability of returning anything is not so big: $\sum_i \varepsilon f_i^2/F_2 = \varepsilon$
  - Repeat $O(1/\varepsilon \log 1/\delta)$ times to improve chance of sampling

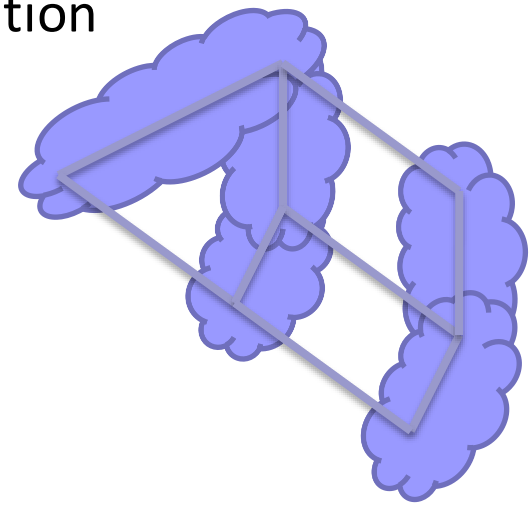
L_2 sampling continued

- Given (estimated) g_i s.t. $g_i^2 \geq F_2/\varepsilon$, estimate $f_i = u_i\, g_i$
- Sketch size $O(\varepsilon^{-1} \log n)$ means estimate of f_i^2 has error $(\varepsilon f_i^2 + u_i^2)$
  - With high prob, no $u_i < 1/\text{poly}(n)$, and so $F_2(g) = O(F_2(f) \log n)$
  - Since estimated $f_i^2/u_i^2 \geq F_2/\varepsilon$, $u_i^2 \leq \varepsilon f_i^2/F_2$
- Estimating f_i^2 with error εf_i^2 sufficient for estimating F_k

- Many details omitted
  - See Precision Sampling paper [Andoni Krauthgamer Onak 11]

Advanced Topics

- L_p Sampling
  - L_0 sampling and graph sketching
  - L_2 sampling and frequency moment estimation
- Matrix computations
  - Sketches for matrix multiplication
  - Compressed matrix multiplication
- Hashing to check computation
  - Matrix product checking
  - Vector product checking
- Lower bounds for streaming and sketching
  - Basic hard problems (Index, Disjointness)
  - Hardness via reductions



Matrix Sketching

- Given matrices A, B, want to approximate matrix product AB
- Compute normed error of approximation C: $\|AB - C\|$
- Give results for the Frobenius (entrywise) norm $\|\cdot\|_F$
  - $\|C\|_F = (\sum_{i,j} C_{i,j}^2)^{½}$
  - Results rely on sketches, so this norm is most natural

Direct Application of Sketches

- Build sketch of each row of A, each column of B
- Estimate $C_{i,j}$ by estimating inner product of A_i with B^j
- Absolute error in estimate is $\varepsilon \|A_i\|_2 \|B^j\|_2$ (whp)
- Sum over all entries in matrix, squared error is

$$\varepsilon^2 \sum_{i,j} \|A_i\|_2^2 \|B^j\|_2^2 = \varepsilon^2 (\sum_i \|A_i\|_2^2)(\sum_j \|B_j\|_2^2)$$
$$= \varepsilon^2 (\|A\|_F^2)(\|B\|_F^2)$$

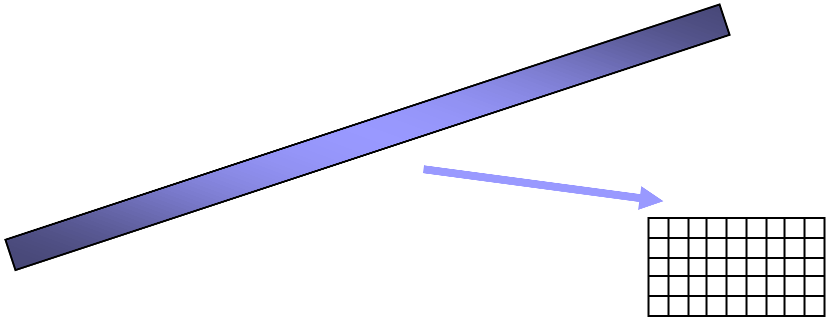
- Hence, Frobenius norm of error is $\varepsilon\|A\|_F\|B\|_F$
- Problem: need the bound to hold for all sketches simultaneously
  - Requires polynomially small failure probability
  - Increases sketch size by logarithmic factors

Improved Matrix Multiplication Analysis

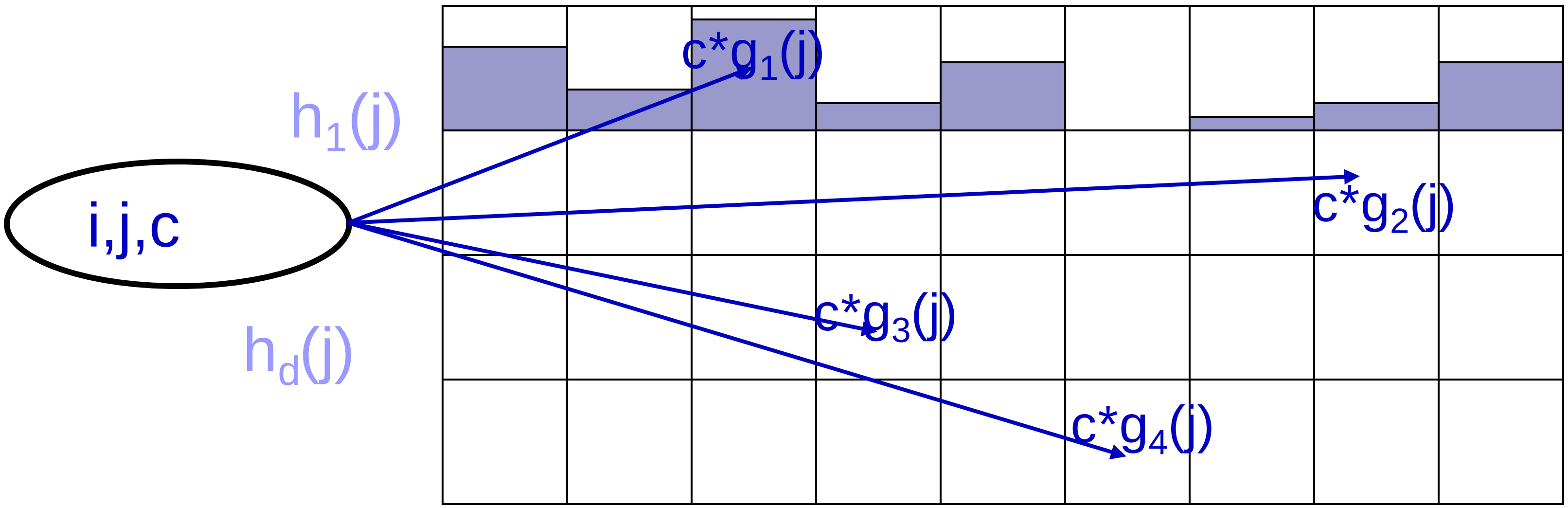
- Simple analysis is too pessimistic [Clarkson Woodruff 09]
  - It bounds probability of failure of each sketch independently
- A better approach is to directly analyze variance of error
  - Immediately, each estimate of (AB) has variance $\varepsilon^2\|A\|_F^2\|B\|_F^2$
  - Just need to apply Chebyshev inequality to this… almost
- Problem: how to amplify probability of correctness?
  - 'Median' trick doesn't work: what is median of set of matrices?
  - Find an estimate which is close to most others
    - Estimate $\|A\|_F^2\|B\|_F^2 := d$ using sketches
    - Find an estimate that's closer than $d/2$ to more than ½ the rest
    - We find an estimate with this property with probability $1-\delta$

Compressed Matrix Multiplication

- What if we are just interested in the large entries of AB?
  - Or, the ability to estimate any entry of (AB)
- If we had a sketch of (AB), could find these approximately
- Compressed Matrix Multiplication [Pagh 12]:
  - Can we compute sketch(AB) from sketch(A) and sketch(B)?
  - To do this, need to dive into structure of the Count (AMS) sketch



Compressed Matrix Multiplication



- Entry $(AB)_{ij}$ gets mapped by a pairwise hash function to a cell q
- Idea: choose a carefully structured hash function
  - $h(i,j) = h_1(i) + h_2(j) \pmod{p}$ is pairwise, if h_1 and h_2 are parwise
- Take convolution of $\text{sketch}(A_{\cdot k})$ [with h_1] and $\text{sketch}(B_{k \cdot})$ [with h_2]
  - Cell q contains $\sum A_{ik} B_{kj}\, g(i)\, g(j)$ where $h(i,j) = q$
  - Repeat for all k and sum to get $\text{sketch}(AB)$

Compressed Matrix Multiplication: Analysis

- Computing the convolution takes time $O(w \log w)$
  - Via Fast Fourier Transform
- Each sketch convolution builds sketch of k'th outer product
  - Total time cost: $O(n(n + w \log w))$
  - Compared to superquadratic cost of exact matrix product
  - Estimate of $(AB)_{ij}$ has error $\|AB\|_F^2/w$
- Several insights needed to build the method:
  - Express matrix product as summation of outer products
  - Convolution of sketches gives a sketch of outer product
  - FFT speeds up from $O(w^2)$ to $O(w \log w)$

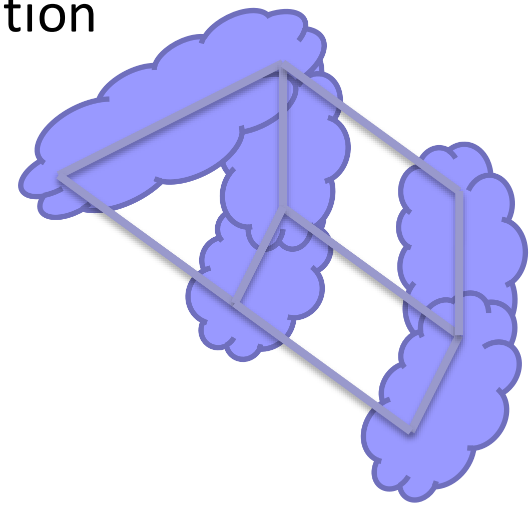
Advanced Linear Algebra

- Recent work more directly approximates matrix multiplication:
  - use more powerful hash functions in sketching
  - obtain a single accurate estimate with high probability

- Linear regression given matrix A and vector b:
  find $x \in R^d$ to (approximately) solve $\min_x \|Ax - b\|$
  - Approach: solve the minimization in "sketch space"
  - Require a summary of size $O(d^2/\varepsilon \log 1/\delta)$

Advanced Topics

- L_p Sampling
  - L_0 sampling and graph sketching
  - L_2 sampling and frequency moment estimation
- Matrix computations
  - Sketches for matrix multiplication
  - Compressed matrix multiplication
- Hashing to check computation
  - Matrix product checking
  - Vector product checking
- Lower bounds for streaming and sketching
  - Basic hard problems (Index, Disjointness)
  - Hardness via reductions



Outsourced Computation

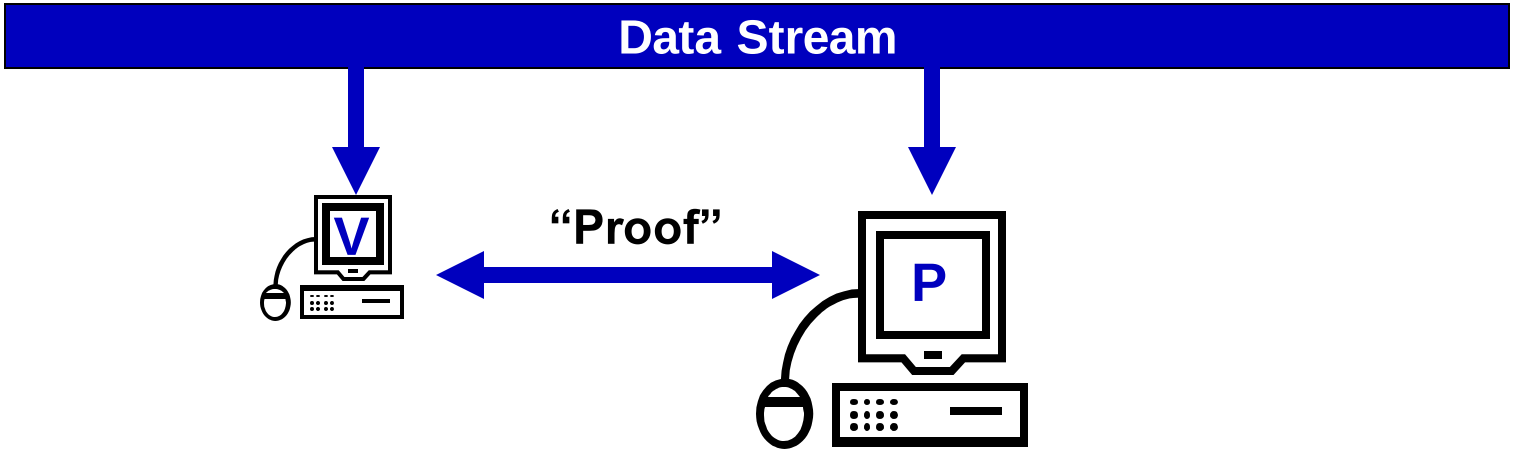
- Current trend to 'outsource' computation
  - Cloud computing: Amazon EC2, Microsoft Azure…
  - Hardware support: multicore systems, graphics cards
- We provide data to a third party, they return an answer
- How can we be sure that the computation is correct?
  - Duplicate the whole computation ourselves?
  - Find some ad hoc sanity checks on the answer?
- Hashing to the rescue: use hashing to prove the correctness
  - Previously, use hashing to test correctness of data (fingerprints)
  - Now, use hashing to test correctness of computation
  - Protocols must be very low cost for the data owner (streaming)
  - Amount of information transmitted should not be too large

Example: Freivald's Algorithm

- Goal: Check $AB = C$ for $n \times n$ matrices A, B, C
  - Naïve algorithm: compute AB, check $= C$ – $O(n^{2.37\ldots})$ time
- Freivald's: check $AB\mathbf{r}^T = C\mathbf{r}^T$ for random vector $\mathbf{r}$
  - A classic example of randomized algorithms, takes $O(n^2)$ time
- Variant: define $\mathbf{r} = [1, r, r^2 \ldots r^n]$ and $\mathbf{s} = [1, s, s^2 \ldots s^n]$ for random r, s
- Check $\mathbf{s}(AB)\mathbf{r}^T = \mathbf{s}C\mathbf{r}^T$ [mod p]
  - Define hash function $h_{r,s}(X) = \mathbf{s}X\mathbf{r}^T \bmod p = \sum_{ij} x_{ij}\, s^i\, r^j \bmod p$
  - $\Pr[h(AB) = h(C)]$ = Probability that a polynomial in r, s of total degree $2n$ evaluates to 0 for randomly chosen variables $= 2n/p$
  - p only has to be polynomial in n, so logarithmic number of bits
- Streaming friendly: compute $(\mathbf{s}A)$, $(B\mathbf{r}^T)$ and $(\mathbf{s}C\mathbf{r}^T)$ incrementally

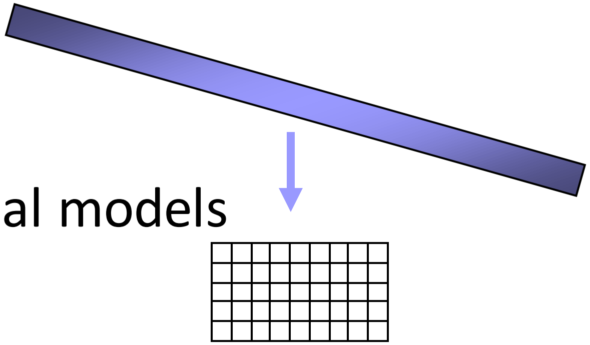
Streaming Proofs

- Objective: prove integrity of the computed solution
  - Not concerned with security: third party sees unencrypted data
- Prover provides "proof" of the correct answer
  - Ensure that "verifier" has very low probability of being fooled
  - Related to communication complexity Arthur-Merlin model, and Arithmetization, with additional streaming constraints



Inner Product Computation

- Given vectors a, b, defined in the stream, want to compute a·b
- Inner product appears in many problems
  - Core computation in data streams
  - Requires Ω(N) space to compute in traditional models



- Results: for h,v s.t. (hv) > N, there exists a protocol with proof size O(h log m), and space O(v log m) to compute inner product
  - Lower bounds: hv = Ω(N) necessary for exact computation

Inner Product Protocol

- Map $[N]$ to $h \times v$ array

3	**7**	**1**	**2**	**0**	**8**	**5**	**9**	**1**	**1**	**1**	**0**

- Interpolate entries in array as polynomials $a(x,y)$, $b(x,y)$
- Verifier picks random r, evaluates $a(r, j)$ and $b(r,j)$ for $j \in [v]$
- Prover sends $s(x) = \sum_{j\in[v]} a(x, j)b(x,j)$ (degree h)
  - Verifier checks $s(r) = \sum_{j\in[v]} a(r,j)b(r,j)$
  - Output $a\cdot b = \sum_{i \in[h]} s(i)$ if test passed

3	**7**	**1**	**2**
0	**8**	**5**	**9**
1	**1**	**1**	**0**

- Probability of failure small if evaluated over large enough field
  - A "Low Degree Extension" / arithmetization technique
  - Can view $a(x,y)$, $b(x,y)$ as (linear) hash functions of the data

Streaming Hash Functions

- Must evaluate $a(r,j)$ incrementally as $a()$ is defined by stream
- Structure of polynomial means updates to (w,z) cause

$$a(r,j) \leftarrow a(r,j) + p_{w,z}(r,j)$$

where $p_{w,z}(x,y) = \prod_{i \in [h]\setminus\{w\}} (x-i)(w-i)^{-1} \cdot \prod_{j \in [v]\setminus\{z\}} (y-j)(z-j)^{-1}$

  - p is a Lagrange polynomial corresponding to an impulse at (w,z)

- Can be computed quickly, using appropriate precomputed look-up tables
- Evaluation is linear: can be computed over distributed data

Consequences

- Verifier can keep space $O(\sqrt{n})$, process proof of size $O(\sqrt{n})$ to verify inner product of two vectors

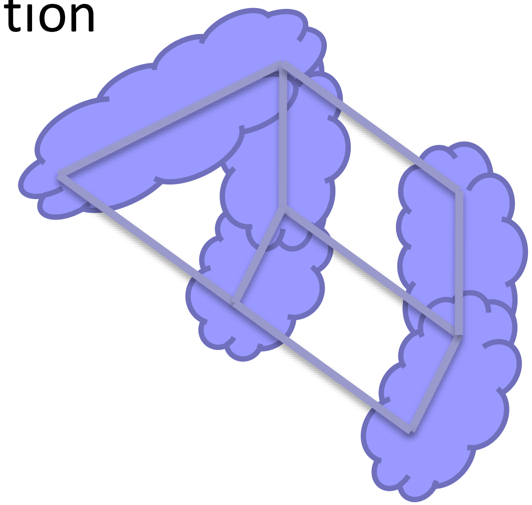
- Many consequences of inner-product verification
  - Easily check Euclidean norm of vector described in stream
  - Verify solutions to linear programs (evaluate primal and dual)
  - Graph computations, e.g. check connected components
  - Count triangles (expressed as polynomial over derived stream)
  - Flow computations (shortest paths, max flow) via IP formulation

Further Directions in Verification

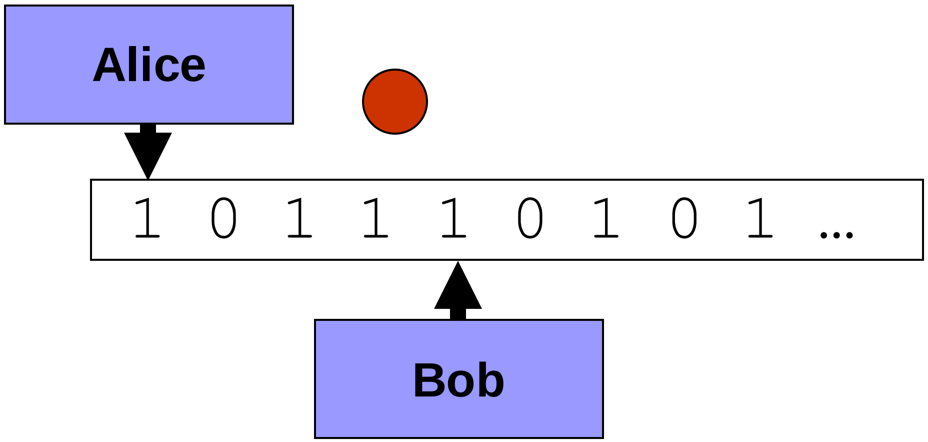
- Multi-round protocols can reduce the costs exponentially
  - Evaluate the low-degree extension of the data at one location
  - Functions as a hash function for computation
- "Interactive Proofs for Muggles" [Goldwasser et al 08]
  - A general purpose approach to verifying computation as circuits
  - Implemented and evaluated by Thaler [Thaler 13]
- Much ongoing around verification
  - Distributed/parallel versions of these protocols
  - Lower bounds for multi-round versions of the protocols
  - Engineering practical implementations

Advanced Topics

- L_p Sampling
  - L_0 sampling and graph sketching
  - L_2 sampling and frequency moment estimation
- Matrix computations
  - Sketches for matrix multiplication
  - Compressed matrix multiplication
- Hashing to check computation
  - Matrix product checking
  - Vector product checking
- Lower bounds for streaming and sketching
  - Basic hard problems (Index, Disjointness)
  - Hardness via reductions



Computation As Communication

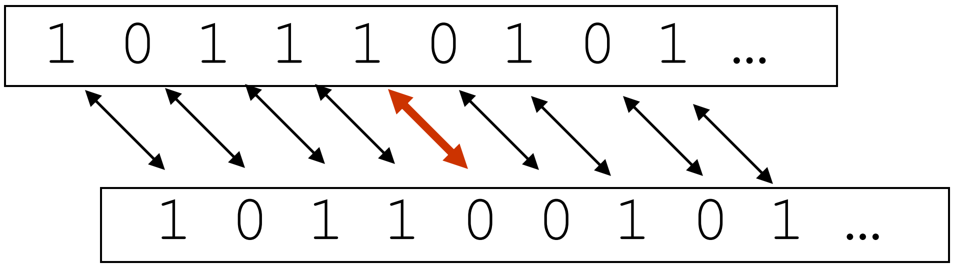


- Imagine Alice processing a prefix of the input
- Then takes the whole working memory, and sends to Bob
- Bob continues processing the remainder of the input

Computation As Communication

- Suppose Alice's part of the input corresponds to string x, and Bob's part corresponds to string y...
- ...and computing the function corresponds to computing f(x,y)...
- ...then if f(x,y) has communication complexity (CC) $\Omega(g(n))$, then the computation has a *space lower bound* of $\Omega(g(n))$
- Proof by contradiction:
  If there was an algorithm with better space usage, we could run it on x, then send the memory contents as a message, and hence solve the communication problem

Deterministic Equality Testing



- Alice has string x, Bob has string y, want to test if $x=y$
- Consider a deterministic (one-round, one-way) protocol that sends a message of length $m < n$
- There are 2^m possible messages, so some strings must generate the same message: this would cause error
- So a deterministic message (sketch) must be $\Omega(n)$ bits
  - In contrast, we saw a randomized sketch of size $O(\log n)$

Four Hard Communication Problems

- **INDEX**: Alice's x is binary string of length n, Bob's y is index in [n]
  Goal: output x[y]
  Result: one-way randomized CC of **INDEX** is $\Omega(n)$ bits
- **AUGINDEX**: as **INDEX**, but Bob also receives x[y+1]…x[n]
  Result: one-way randomized CC of **AUGINDEX** is $\Omega(n)$ bits
- **DISJ**: Alice's x and Bob's y are both length n binary strings
  Goal: Output 1 if $\exists$i: x[i]=y[i]=1, else 0
  Result: multi-round randomized CC of **DISJ** (disjointness) is $\Omega(n)$ bits
- **Gap-Hamming:** Alice's x and Bob's y are both length n binary strings
  Promise: Ham(x,y) is either $\leq N/2 - \sqrt{N}$ or $\geq N/2 + \sqrt{N}$
  Goal: determine which case holds
  Result: multi-round randomized CC of **Gap-Hamming** is $\Omega(n)$ bits

Simple Reduction to Disjointness

x: 1 0 1 1 0 1 $\longrightarrow$ **1, 3, 4, 6**

y: 0 0 0 1 1 0 $\longrightarrow$ **4, 5**

- F_∞: output the highest frequency in the input
- Input: the two strings x and y from disjointness instance
- Reduction: if x[i]=1, then put i in input; then same for y
  - A streaming reduction (compare to polynomial-time reductions)
- Analysis: if $F_\infty=2$, then intersection; if $F_\infty \leq 1$, then disjoint.
- Conclusion: Giving exact answer to F_∞ requires $\Omega(N)$ bits
  - Even approximating up to 50% relative error is hard
  - Even with randomization: `DISJ` bound allows randomness

Simple Reduction to Index

x: 1 0 1 1 0 1 $\longrightarrow$ 1, 3, 4, 6

y: 5 $\longrightarrow$ 5

- F_0: output the number of items in the stream
- Input: the strings x and index y from **INDEX**
- Reduction: if x[i]=1, put i in input; then put y in input
- Analysis: if $(1-\varepsilon)F'_0(x \cup y) > (1+\varepsilon)F'_0(x)$ then x[y]=1, else it is 0
- Conclusion: Approximating F_0 for $\varepsilon<1/N$ requires $\Omega(N)$ bits
  - Implies that space to approximate must be $\Omega(1/\varepsilon)$
  - Bound allows randomization

Reduction to AUGINDEX [Clarkson Woodruff 09]

- Matrix-Multiplication: approximate $A^T B$ with error $\varepsilon^2 \|A\|_F \|B\|_F$
  - For $r \times c$ matrices. A encodes string x, B encodes index y

$$
\begin{array}{cc}
 & \begin{array}{cccccccccccccc} & & \multicolumn{2}{c}{\overleftrightarrow{r/\log(cn)}} & & & & & & & & & & \end{array} \\
c \updownarrow &
\left[\begin{array}{cc|cc|c|cc|cccccc}
+1 & -1 & -2 & -2 & \dots & \pm 2^k & \pm 2^k & \dots & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & -2 & +2 & \dots & \pm 2^k & \pm 2^k & \dots & 0 & 0 & 0 & 0 & 0 \\
+1 & +1 & +2 & -2 & \dots & \boxed{\pm 2^k} & \pm 2^k & \dots & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & +2 & +2 & \dots & \pm 2^k & \pm 2^k & \dots & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\left[\begin{array}{ccc}
0 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots \\
1 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots \\
0 & 0 & \dots
\end{array}\right]
\end{array}
$$

$A^T B$ "reads off" j'th column of A^T

- Bob uses suffix of x in y to remove heavy entries from A

  $\|B\|_F = 1$ $\quad\quad$ $\|A\|_F = cr/\log(cn) * (1 + 4 + \dots 2^{2k}) \le 4cr2^{2k}/3\log(cn)$
- Choose $r = \log(cn)/8\varepsilon^2$ so permitted error is $c\, 2^{2k} / 6\varepsilon^2$
  - Each error in sign in estimate of $(A^T B)$ contributes 2^{2k} error
  - Can tolerate error in at most $1/6$ fraction of entries
- Matrix multiplication requires space $\Omega(rc) = \Omega(c/\varepsilon^2 \log(cn))$

Lower Bound for Entropy

Gap-Hamming instance—Alice: $x \in \{0,1\}^N$, Bob: $y \in \{0,1\}^N$

Entropy estimation algorithm **A**

- Alice runs **A** on $enc(x) = \langle(1,x_1), (2,x_2), \ldots, (N,x_N)\rangle$
- Alice sends over memory contents to Bob
- Bob continues **A** on $enc(y) = \langle(1,y_1), (2,y_2), \ldots, (N,y_N)\rangle$

Alice	0	1	0	0	1	1
	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
Bob	1	1	0	0	1	0

Lower Bound for Entropy

- Observe: there are
  - $2\mathrm{Ham}(x,y)$ tokens with frequency 1 each
  - $N-\mathrm{Ham}(x,y)$ tokens with frequency 2 each
- So (after algebra), $H(S) = \log N + \mathrm{Ham}(x,y)/N = \log N + ½ \pm 1/\sqrt{N}$
- If we separate two cases, size of Alice's memory contents = $\Omega(N)$
  Set $\varepsilon = 1/(\sqrt{(N)} \log N)$ to show bound of $\Omega(\varepsilon/\log 1/\varepsilon)^{-2})$

Alice	0	1	0	0	1	1
	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
Bob	1	1	0	0	1	0

Lower Bound for F_0

- Same encoding works for F_0 (Distinct Elements)
  - 2Ham(x,y) tokens with frequency 1 each
  - N-Ham(x,y) tokens with frequency 2 each
- $F_0(S) = N + Ham(x,y)$
- Either $Ham(x,y) > N/2 + \sqrt{N}$ or $Ham(x,y) < N/2 - \sqrt{N}$
  - If we could approximate F_0 with $\varepsilon < 1/\sqrt{N}$, could separate
  - But space bound $= \Omega(N) = \Omega(\varepsilon^{-2})$ bits
- Dependence on ε for F_0 is tight

- Similar arguments show $\Omega(\varepsilon^{-2})$ bounds for F_k
  - Proof assumes k (and hence 2^k) are constants

Summary of Tools

- Vector equality: fingerprints
- Approximate item frequencies:
  - Count-min (L_1 guarantee), Count sketch (L_2 guarantee)
- Euclidean norm, inner product: AMS sketch, JL sketches
- Count-distinct: k-Minimum values, Hyperloglog
- Compact set-representation: Bloom filters
- L_0 sampling: hashing and sparse recovery
- L_2 sampling: via count-sketch
- Graph sketching: L_0 samples of neighborhood
- Frequency moments: via L_2 sampling
- Matrix sketches: adapt AMS sketches, compressed matrix multiplication

Summary of Lower Bounds

- Can’t deterministically test equality
- Can’t retrieve arbitrary bits from a vector of n bits: **INDEX**
  - Even if some unhelpful suffix of the vector is given: **AUGINDEX**
- Can’t determine whether two n bit vectors intersect: **DISJ**
- Can’t distinguish small differences in Hamming distance: **GAP-HAMMING**
- These in turn provide lower bounds on the cost of
  - Finding the maximum frequency
  - Approximating the number of distinct items
  - Approximating matrix multiplication

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