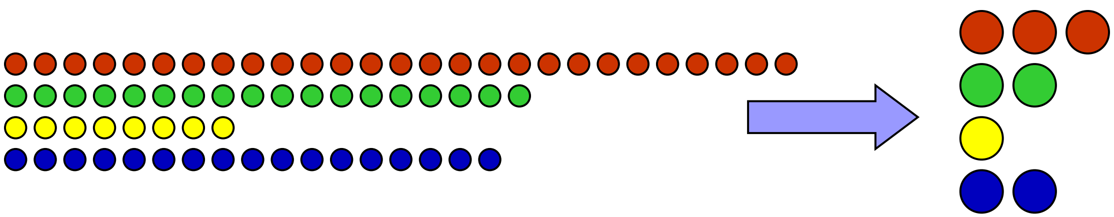
Compact Summaries for ~~Large Datasets~~ Big Data



Graham Cormode

University of Warwick

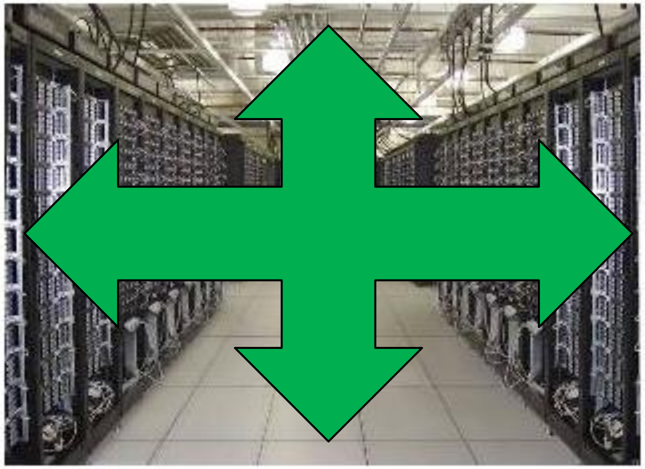
G.Cormode@Warwick.ac.uk

The case for “Big Data” in one slide

- “Big” data arises in many forms:
  - Medical data: genetic sequences, time series
  - Activity data: GPS location, social network activity
  - Business data: customer behavior tracking at fine detail
  - Physical Measurements: from science (physics, astronomy)
- 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
- “Big data” is about more than simply the volume of the data
  - But large datasets present a particular challenge for us!

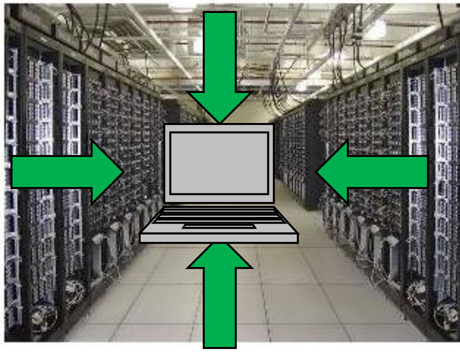
Computational scalability

- The first (prevailing) approach: **scale up the computation**
- Many great technical ideas:
  - Use many cheap commodity devices
  - Accept and tolerate failure
  - Move data to code, not vice-versa
  - MapReduce: BSP for programmers
  - Break problem into many small pieces
  - Add layers of abstraction to build massive DBMSs and warehouses
  - Decide which constraints to drop: noSQL, BASE systems



- Scaling up comes with its disadvantages:
  - Expensive (hardware, equipment, **energy**), still not always fast
- This talk is not about this approach!

Downsizing data



- A second approach to computational scalability: **scale down the data!**
  - A compact representation of a large data set
  - Capable of being analyzed on a single machine
  - What we finally want is small: human readable analysis / decisions
  - Necessarily gives up some accuracy: approximate answers
  - Often randomized (small constant probability of error)
  - Much relevant work: samples, histograms, wavelet transforms
- Complementary to the first approach: not a case of either-or
- Some drawbacks:
  - Not a general purpose approach: need to fit the problem
  - Some computations don’t allow any useful summary

Outline for the talk

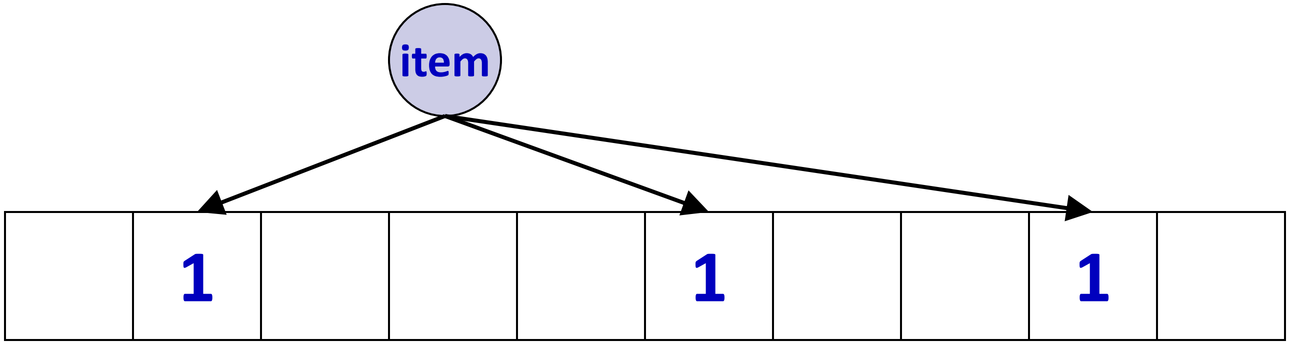
- Some examples of compact summaries (high level, no proofs)
  - Sketches: Bloom filter, Count-Min, AMS
  - Sampling: simple samples, count distinct
  - Summaries for more complex objects: graphs and matrices
- Lower bounds: limitations of when summaries can exist
  - No free lunch
- Current trends and future challenges for compact summaries
- Many abbreviations and omissions (histograms, wavelets, ...)
- A lot of work relevant to compact summaries
  - Including many papers in SIGMOD/PODS

Summary Construction

- There are several different models for summary construction
  - **Offline computation**: e.g. sort data, take percentiles
  - **Streaming**: summary merged with one new item each step
  - **Full mergeability**: allow arbitrary merges of partial summaries
    - The most general and widely applicable category
- Key methods for summaries:
  - Create an empty summary
  - Update with one new tuple: streaming processing
  - Merge summaries together: distributed processing (eg MapR)
  - Query: may tolerate some approximation (parameterized by ε)
- Several important cost metrics (as function of ε, n):
  - Size of summary, time cost of each operation

Bloom Filters

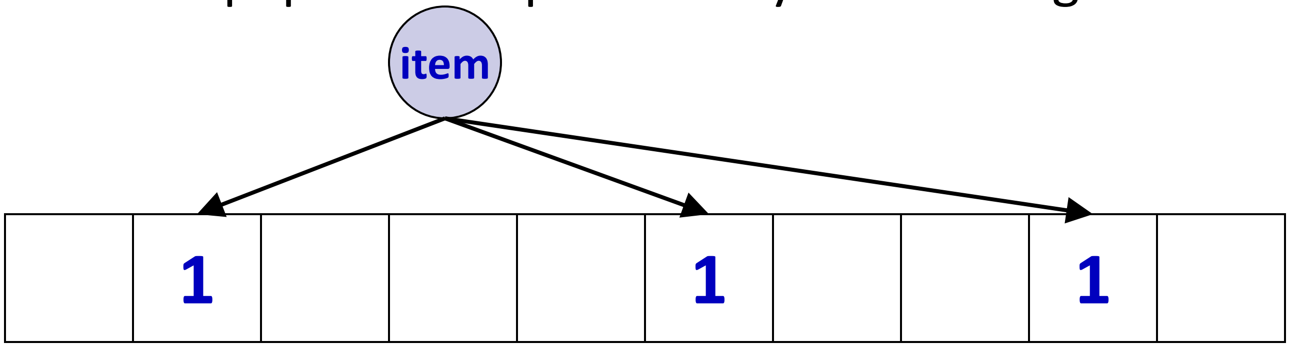
- Bloom filters [Bloom 1970] compactly encode set membership
  - E.g. store a list of many long URLs compactly
  - k hash functions map items to m-bit vector k times
  - Set all k entries to **1** to indicate item is present
  - Can lookup items, store set of size n in O(n) bits
    - Analysis: choose k and size m to obtain small false positive prob



- Duplicate insertions do not change Bloom filters
- Can be merge by OR-ing vectors (of same size)

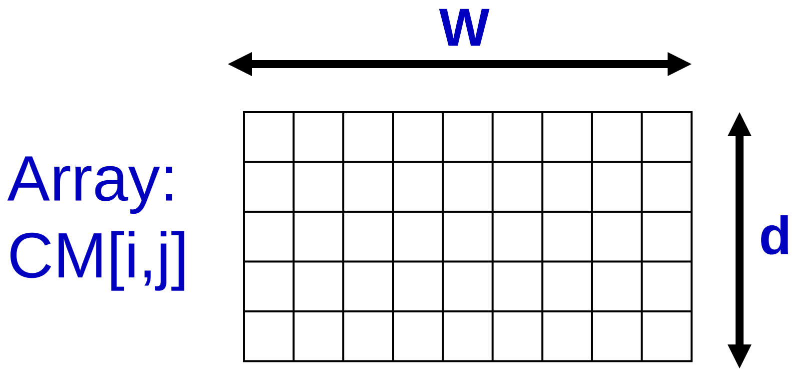
Bloom Filters Applications

- Bloom Filters widely used in “big data” applications
  - Many problems require storing a large set of items
- Can generalize to allow deletions
  - Swap bits for counters: increment on insert, decrement on delete
  - If representing sets, small counters suffice: 4 bits per counter
  - If representing multisets, obtain (counting) sketches
- Bloom Filters are an active research area
  - Several papers on topic in every networking conference…

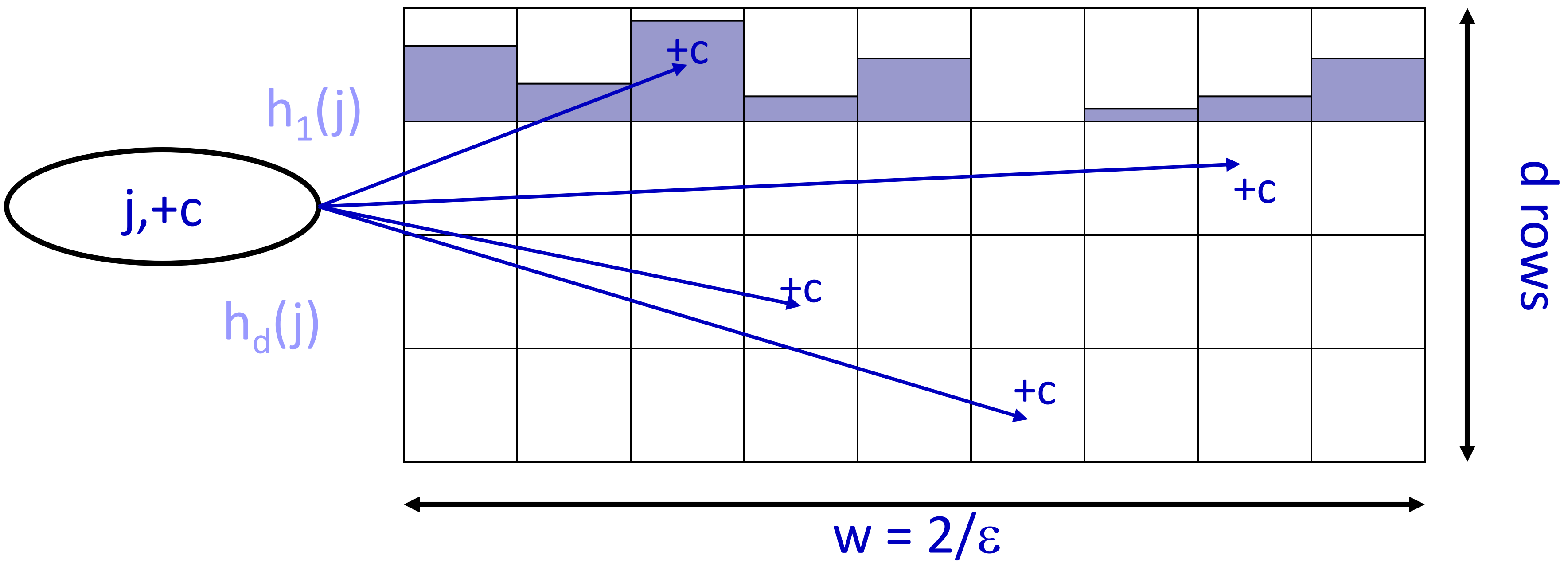


Count-Min Sketch

- Count Min sketch [C, Muthukrishnan 04] encodes item counts
  - Allows estimation of frequencies (e.g. for selectivity estimation)
  - Some similarities in appearance to Bloom filters
- Model input data as a vector x of dimension U
  - Create a small summary as an array of $w \times d$ in size
  - Use d hash function to map vector entries to [1..w]



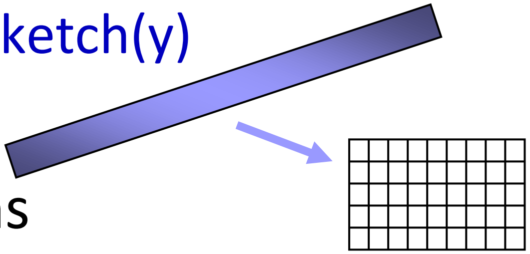
Count-Min Sketch Structure



- 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 $\varepsilon||x||_1$ in size $O(1/\varepsilon)$
  - Probability of more error reduced by adding more rows

Generalization: Sketch Structures

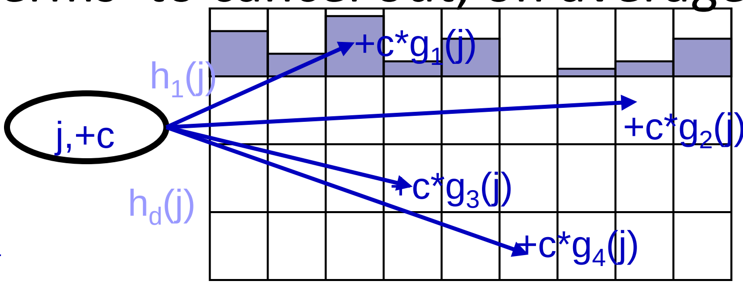
- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



- Often describe S in terms of hash functions
  - S must have compact description to be worthwhile
  - If hash functions are simple, sketch is fast
- Analysis relies on properties of the hash functions
  - Seek "limited independence" to limit space usage
  - Proofs usually study the expectation and variance of the estimates

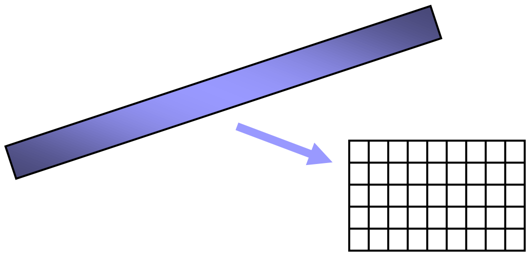
Sketching for Euclidean norm

- AMS sketch presented in [Alon Matias Szegedy 96]
  - Allows estimation of F_2 (second frequency moment)
  - Leads to estimation of (self) join sizes, inner products
  - Used at the heart of many streaming and non-streaming applications: achieves dimensionality reduction ('Johnson-Lindenstrauss lemma')
- Here, describe (fast) AMS sketch by generalizing CM sketch
  - Use extra hash functions $g_1 \ldots g_d$ $\{1 \ldots U\} \rightarrow \{+1,-1\}$
  - Now, given update (j,+c), set $CM[k,h_k(j)]$ += $c*g_k(j)$
- Estimate squared Euclidean norm (F_2) = $\text{median}_k \sum_i CM[k,i]^2$
  - Intuition: g_k hash values cause 'cross-terms' to cancel out, on average
  - The analysis formalizes this intuition
  - median reduces chance of large error



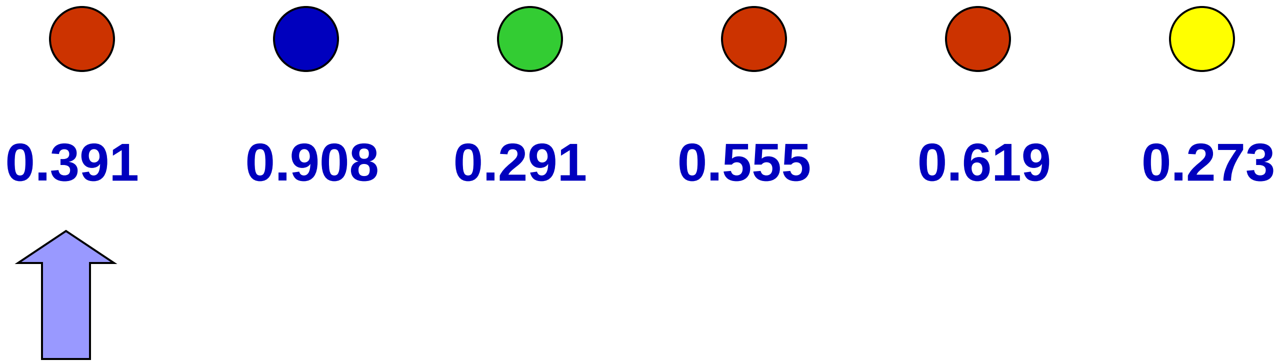
Application to Large Scale Machine Learning

- 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



Min-wise Sampling

- Fundamental problem: sample m items uniformly from data
  - Allows evaluation of query on sample for approximate answer
  - Challenge: don’t know how large total input is, so how to set rate?
- For each item, pick a random fraction between 0 and 1
- Store item(s) with the smallest random tag [Nath et al.’04]



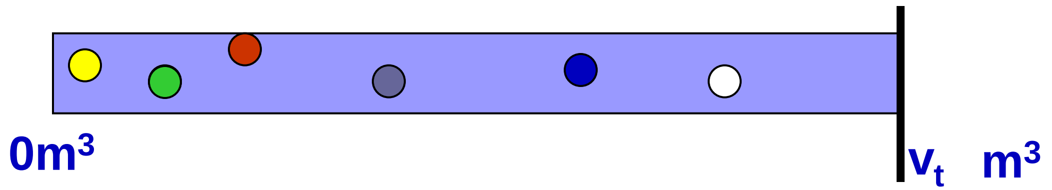
- Each item has same chance of least tag, so uniform
  - Leads to an intuitive proof of correctness
- Can run on multiple inputs separately, then merge

F_0 Estimation

- F_0 is the number of distinct items in the data
  - A fundamental quantity with many applications
  - COUNT DISTINCT estimation in DBMS
- Application: track online advertising views
  - Want to know how many distinct viewers have been reached
- Early approximate summary due to Flajolet and Martin [1983]
- Will describe a generalized version of the FM summary due to Bar-Yossef et. al with only pairwise indendence
  - Known as the "k-Minimum values (KMV)" algorithm

KMV F_0 estimation algorithm

- Let m be the domain of data elements
  - Each item in data is from [1…m]
- Pick a random (pairwise) hash function $h: [m] \to [R]$
  - For R "large enough" (polynomial), assume no collisions under h



- Keep 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 sets $t = 1/\varepsilon^2$ to give ε relative error

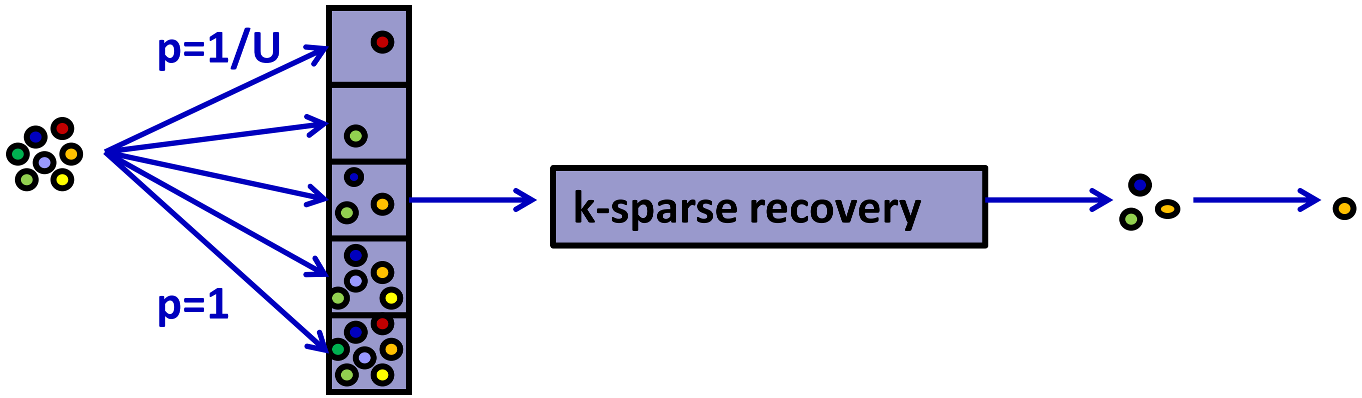
Engineering Count Distinct

- Hyperloglog algorithm [Flajolet Fusy Gandouet Meunier 07]
  - 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
  - Take harmonic mean of estimates from each bucket
    - Analysis much more involved
- 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
    - Lower bound implies should **not** estimate intersections well!
  - Higher order intersections via inclusion-exclusion principle

L_0 Sampling

- L_0 sampling: sample item i with prob $(1\pm\varepsilon)\ f_i^0/F_0$
  - i.e., sample (near) uniformly from items with non-zero frequency
  - Challenging when frequencies can increase and decrease
- 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 (summary)
    - Allows reconstruction of f_p if $F_0 < k$, uses space $O(k)$
  - If f_p is k-sparse, sample from reconstructed vector
  - Repeat in parallel for exponentially shrinking values of p

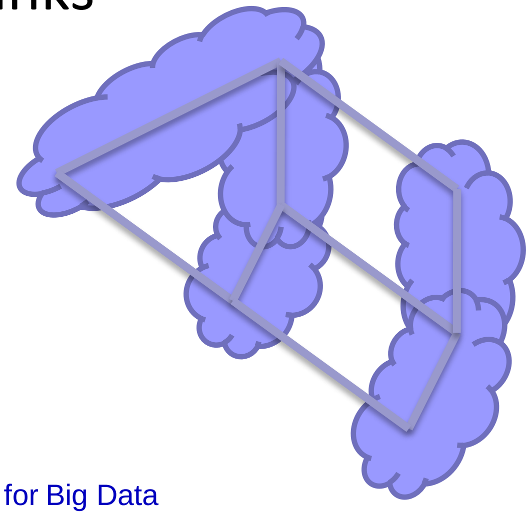
Sampling Process



- Exponential set of probabilities, p=1, ½, ¼, 1/8, 1/16… 1/U
  - Want there to be a level where k-sparse recovery will succeed
    - Sub-sketch that can decode a vector if it has few non-zeros
  - At level p, expected number of items selected S is pF_0
  - Pick level p so that $k/3 < pF_0 \leq 2k/3$
- Analysis: this is very likely to succeed and sample correctly

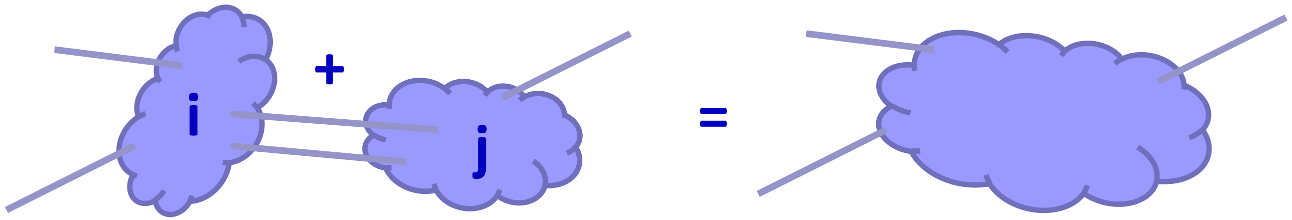
Graph Sketching

- 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
  - Implement: Use L_0 sampling to get edges from vector of adjacencies
  - One sketch for the adjacency list for each node
- Problem: as components grow, sampling edges from components most likely to produce internal links



Graph Sketching

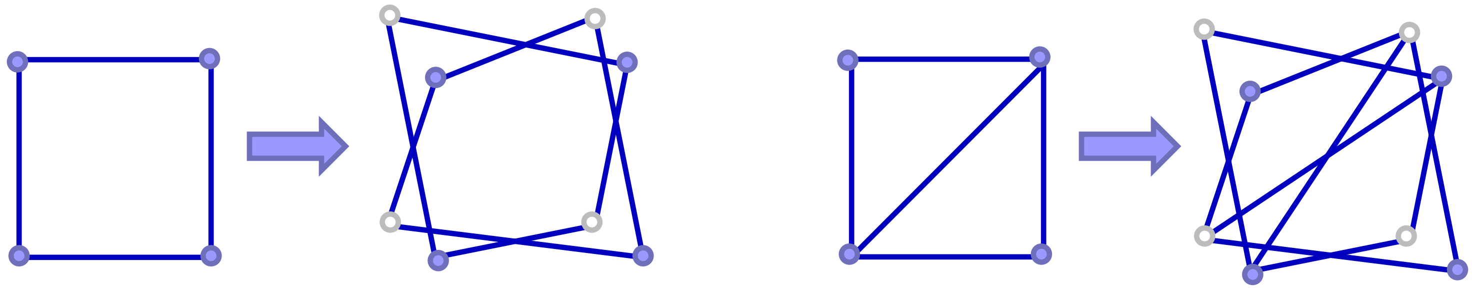
- Idea: use clever encoding of edges [Ahn, Guha, McGregor 12]
- Encode edge (i,j) as ((i,j),+1) for node i<j, as ((i,j),-1) for node j>i
- When node i and node j get merged, sum their L_0 sketches
  - Contribution of edge (i,j) exactly cancels out



  - 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

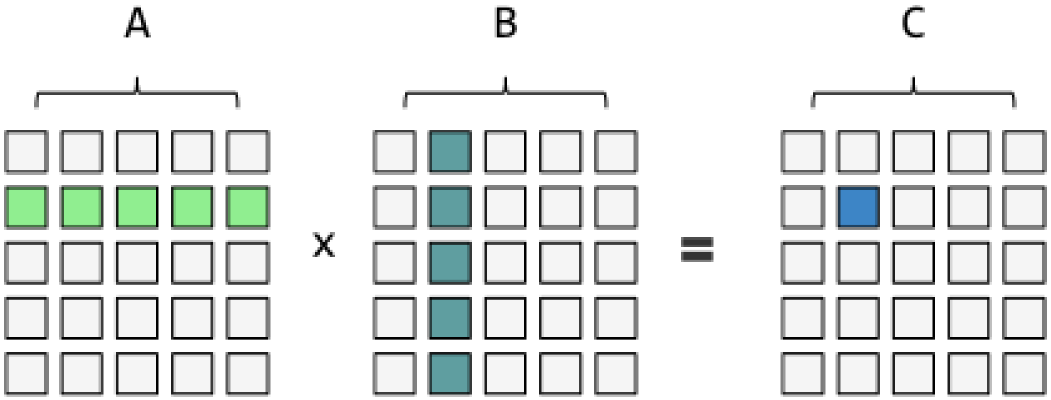
Other Graph Results via sketching

- Recent flurry of activity in summaries for graph problems
  - K-connectivity via connectivity
  - Bipartiteness via connectivity:
  - (Weight of the) Minimum spanning tree:
  - Sparsification: find G' with few edges so that cut(G,C) ≈ cut(G',C)
  - Matching: find a maximal matching (assuming it is small)
- Cost is typical O(|V|), rather than O(|E|)
  - Semi-streaming / semi-external model



Matrix Sketching

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



Direct Application of Sketches

- Build AMS sketch of each row of A (A_i), each column of B (B^j)
- 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 $\varepsilon\|A\|_F\|B\|_F$
- Outline formalized & improved by Clarkson & Woodruff [09,13]
  - Improve running time to linear in number of non-zeros in A,B

Compressed Matrix Multiplication

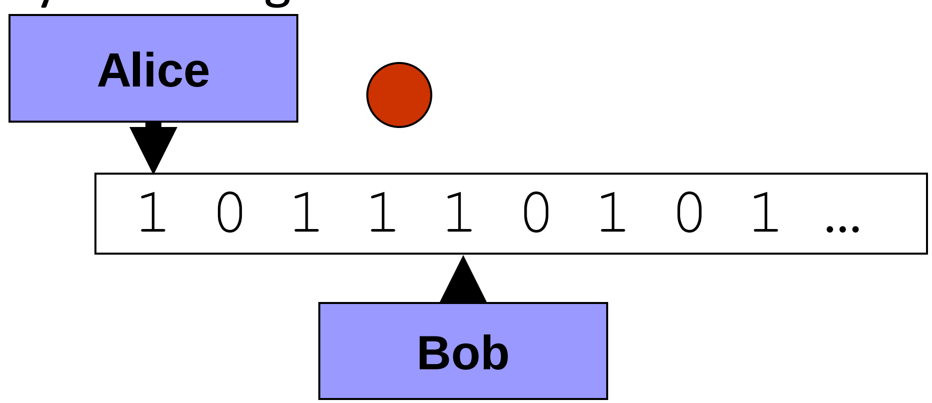
- What if we are just interested in the large entries of AB?
  - Or, the ability to estimate any entry of (AB)
  - Arises in recommender systems, other ML applications
- If we had a sketch of (AB), could find these approximately
- Compressed Matrix Multiplication [Pagh 12]:
  - Can we compute sketch(AB) from sketch(A) and sketch(B)?
  - To do this, need to dive into structure of the Count (AMS) sketch
- Several insights needed to build the method:
  - Express matrix product as summation of outer products
  - Take convolution of sketches to get a sketch of outer product
  - New hash function enables this to proceed
  - Use the FFT to speed up from $O(w^2)$ to $O(w \log w)$

More Linear Algebra

- Matrix multiplication improvement: use more powerful hash fns
  - 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”
  - From a summary of size $O(d^2/\varepsilon)$ [independent of rows of A]
- Frequent directions: approximate matrix-vector product [Ghashami, Liberty, Phillips, Woodruff 15]
  - Use the SVD to (incrementally) summarize matrices
- The relevant sketches can be built quickly: proportional to the number of nonzeros in the matrices (input sparsity)
  - Survey: Sketching as a tool for linear algebra [Woodruff 14]

Lower Bounds

- While there are many examples of things we can summarize…
  - What about things we can't do?
  - What's the best we could achieve for things we can do?
- Lower bounds for summaries from communication complexity
  - Treat the summary as a **message** that can be sent between players
- Basic principle: summaries must be proportional to the size of the information they carry
  - A summary encoding N bits of data must be at least N bits in size!



Summary of Lower Bounds

- Some fundamental hard problems:
  - Can't retrieve arbitrary bits from a vector of n bits: **INDEX**
  - 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 count (can't do this exactly in small space)
  - Approximating the number of distinct items (need $1/\varepsilon^2$, not $1/\varepsilon$)
  - Graph connectivity (can't do better than $|V|$)
  - Approximating matrix multiplication (can't get relative error)

Current Directions in Data Summarization

- Sparse representations of high dimensional objects
  - Compressed sensing, sparse fast fourier transform
- General purpose numerical linear algebra for (large) matrices
  - k-rank approximation, linear regression, PCA, SVD, eigenvalues
- Summaries to verify full calculation: a 'checksum for computation'
- Geometric (big) data: coresets, clustering, machine learning
- Use of summaries in large-scale, distributed computation
  - Build them in MapReduce, Continuous Distributed models
- Communication-efficient maintenance of summaries
  - As the (distributed) input is modified

Summary of Summaries

- Two complementary approaches in response to growing data sizes
  - Scale the computation up; scale the data down
- The theory and practice of data summarization has many guises
  - Sampling theory (since the start of statistics)
  - Streaming algorithms in computer science
  - Compressive sampling, dimensionality reduction… (maths, stats, CS)
- Continuing interest in applying and developing new theory
  - Ad: Postdoc & PhD studentships available at U of Warwick

erc European Research Council Established by the European Commission

EPSRC Engineering and Physical Sciences Research Council

YAHOO! RESEARCH

Microsoft Research

THE ROYAL SOCIETY