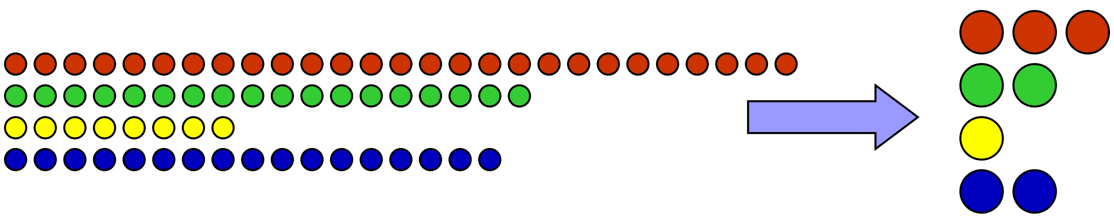
Data Summarization
for
Machine Learning



Graham Cormode

University of Warwick

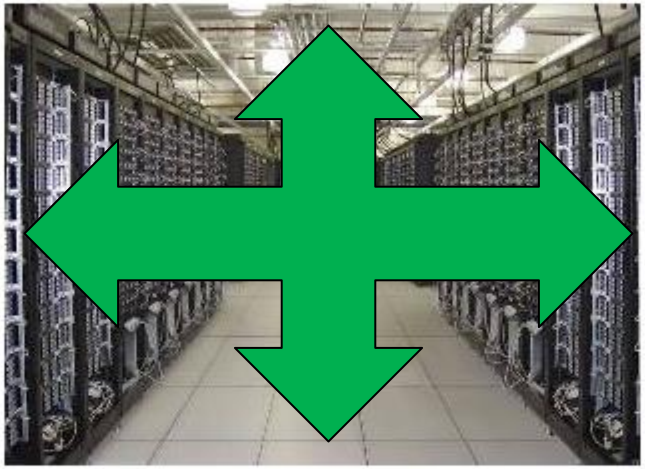
G.Cormode@Warwick.ac.uk

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 want to (efficiently) find patterns and make predictions
- “Big data” is about more than simply the volume of the data
  - But large datasets present a particular challenge for us!

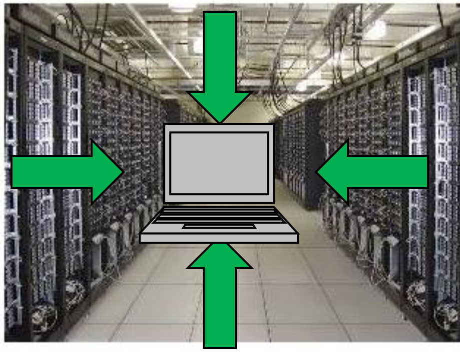
Computational scalability

- The first (prevailing) approach: **scale up the computation**
- Many great technical ideas:
  - Use many cheap commodity devices
  - Accept and tolerate failure
  - Move code to data, 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

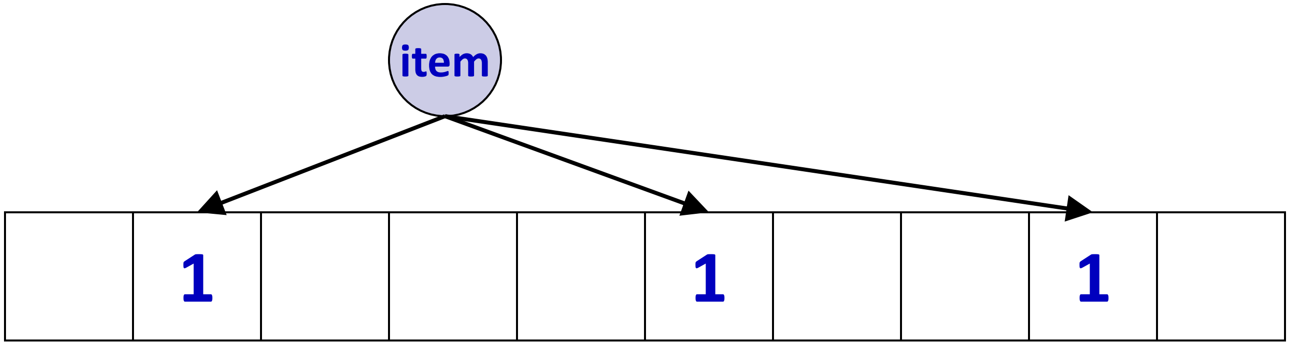
- Part 1: Few examples of compact summaries (no proofs)
  - Sketches: Bloom filter, Count-Min, AMS
  - Sampling: count distinct, distinct sampling
  - Summaries for more complex objects: graphs and matrices
- Part 2: Some recent work on summaries for ML tasks
  - Distributed construction of Bayesian models
  - Approximate constrained regression via sketching

Summary Construction

- A ‘summary’ is a small data structure, constructed incrementally
  - Usually giving approximate, randomized answers to queries
- 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

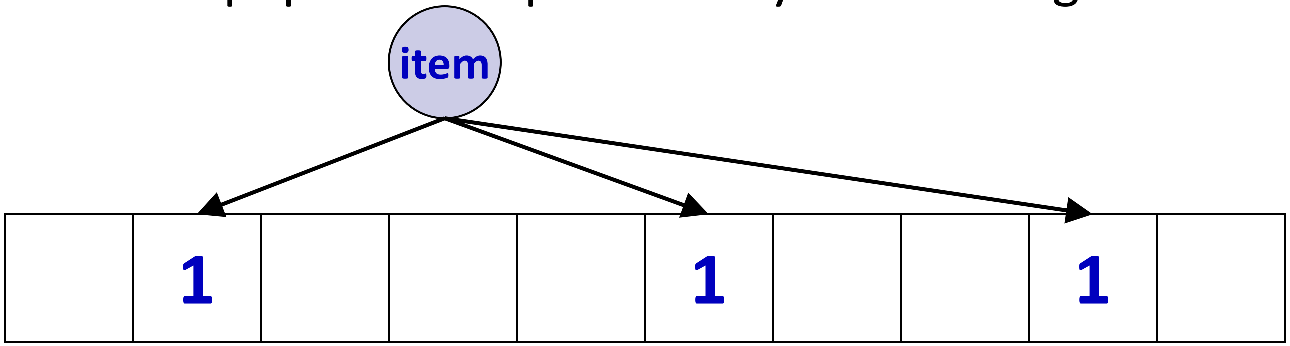
- Bloom filters [Bloom 1970] compactly encode set membership
  - E.g. store a list of many long URLs compactly
  - k hash functions map items to m-bit vector k times
  - Update: Set all k entries to **1** to indicate item is present
  - Query: Can lookup items, store set of size n in O(n) bits
    -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

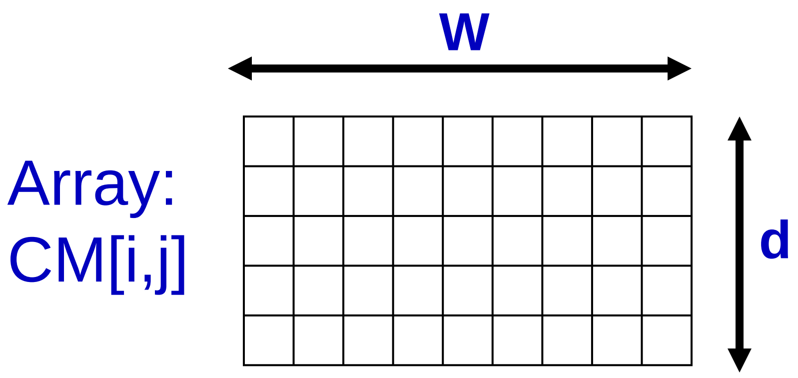
Bloom Filters Applications

- 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…

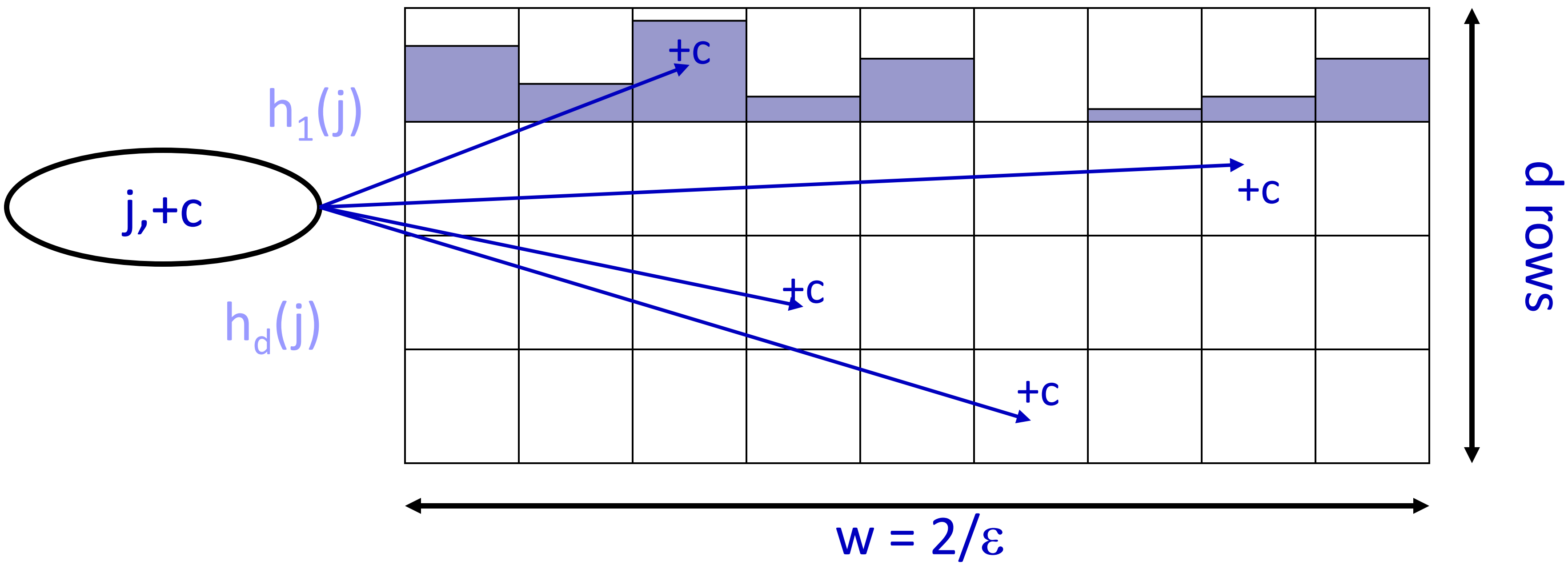


Count-Min Sketch

- 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 $[1..w]$



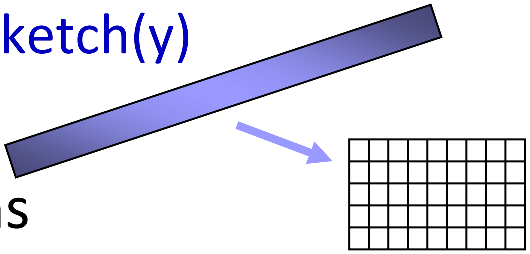
Count-Min Sketch Structure



- Update: each entry in vector x is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Query: 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

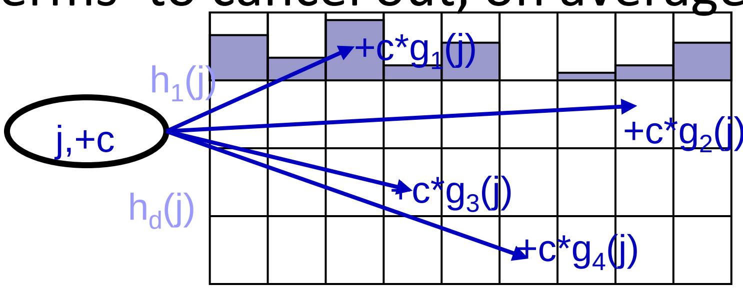
- Sketch is a class of summary that is a linear transform of input
  - Sketch(x) = Sx for some matrix S
  - Hence, Sketch(αx + βy) = α Sketch(x) + β Sketch(y)
  - Trivial to update and merge



- 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

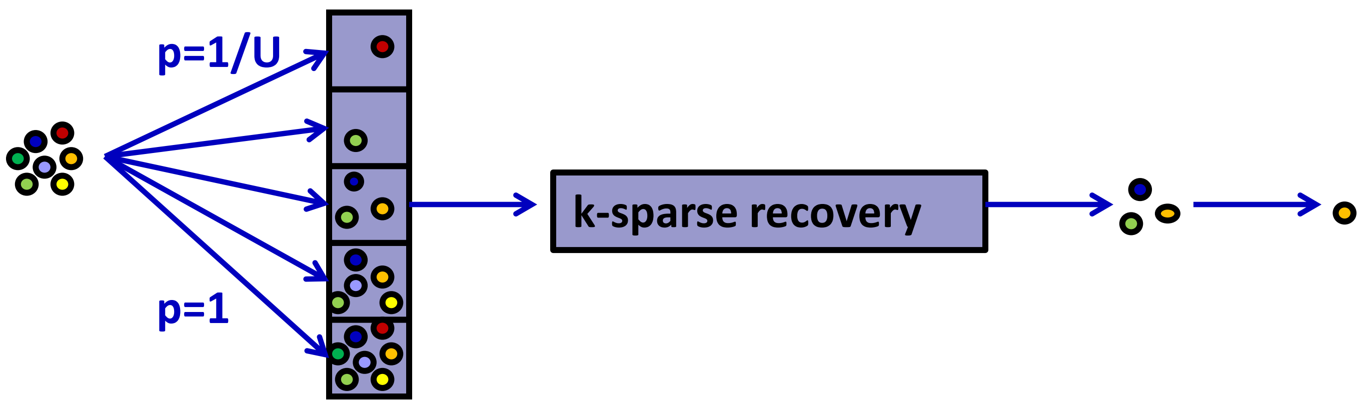
- AMS sketch presented in [Alon Matias Szegedy 96]
  - Allows estimation of F_2 (second frequency moment) aka $\|x\|_2^2$
  - Leads to estimation of (self) join sizes, inner products
  - Used at the heart of many streaming and non-streaming applications: achieves dimensionality reduction ('Johnson-Lindenstrauss lemma')
- Here, describe the related CountSketch 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 $CM[k,h_k(j)] \mathrel{+}= c * g_k(j)$
- Estimate squared Euclidean norm (F_2) = $\mathrm{median}_k \sum_i CM[k,i]^2$
  - Intuition: g_k hash values cause 'cross-terms' to cancel out, on average
  - The analysis formalizes this intuition
  - median reduces chance of large error



L_0 Sampling

- L_0 sampling: sample item i with prob $(1\pm\varepsilon)\ f_i^0/F_0$ (# distinct items)
  - 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 (sketch 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

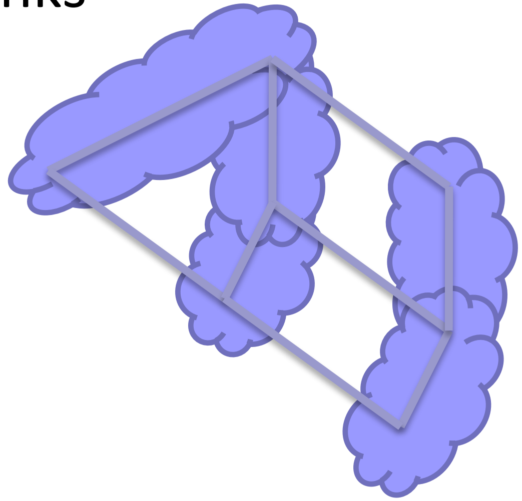
Sampling Process



- 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

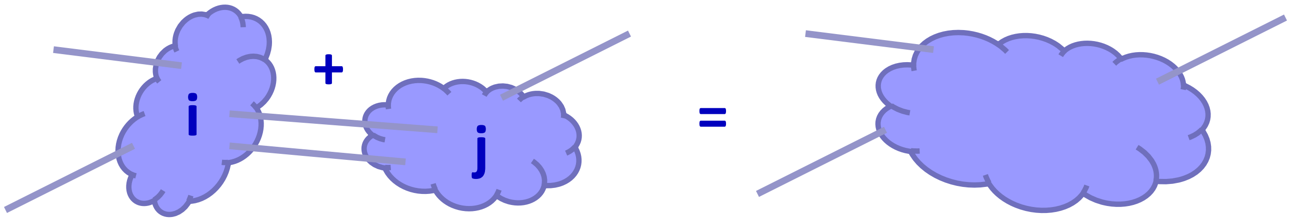
Graph Sketching

- Given L_0 sampler, use to sketch (undirected) graph properties
- Connectivity: find the connected components of the graph
- 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

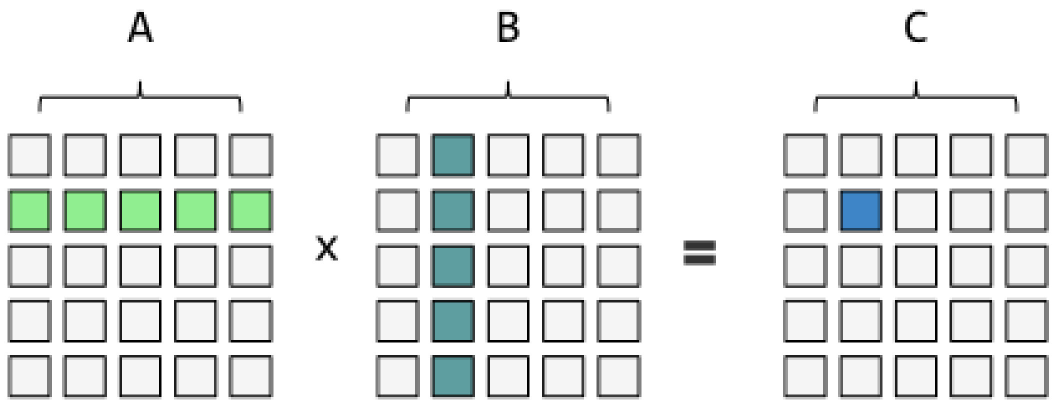
- Idea: use clever encoding of edges [Ahn, Guha, McGregor 12]
- Encode edge (i,j) as ((i,j),+1) for node i<j, as ((i,j),-1) for node j>i
- When node i and node j get merged, sum their L_0 sketches
  - Contribution of edge (i,j) exactly cancels out



  - 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 connected components

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

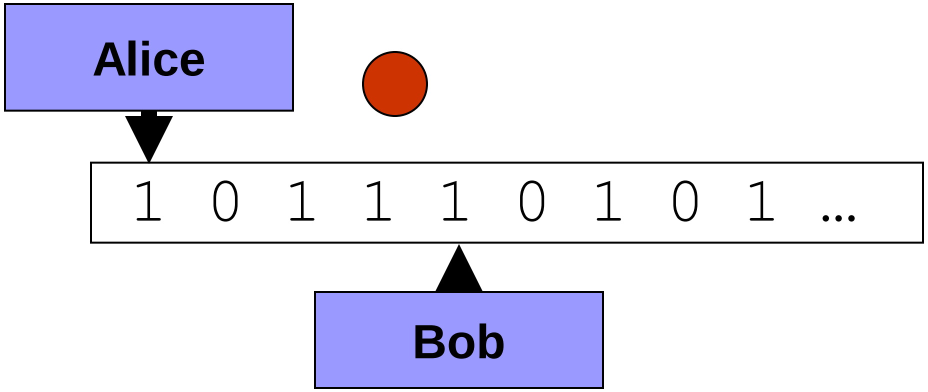
- Build AMS sketch of each row of A (A_i), each column of B (B^j)
- Estimate $C_{i,j}$ by estimating inner product of A_i with B^j
  - Absolute error in estimate is $\varepsilon \|A_i\|_2 \|B^j\|_2$ (whp)
  - Sum over all entries in matrix, Frobenius error is $\varepsilon\|A\|_F\|B\|_F$
- Outline formalized & improved by Clarkson & Woodruff [09,13]
  - Improve running time to linear in number of non-zeros in A,B

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]

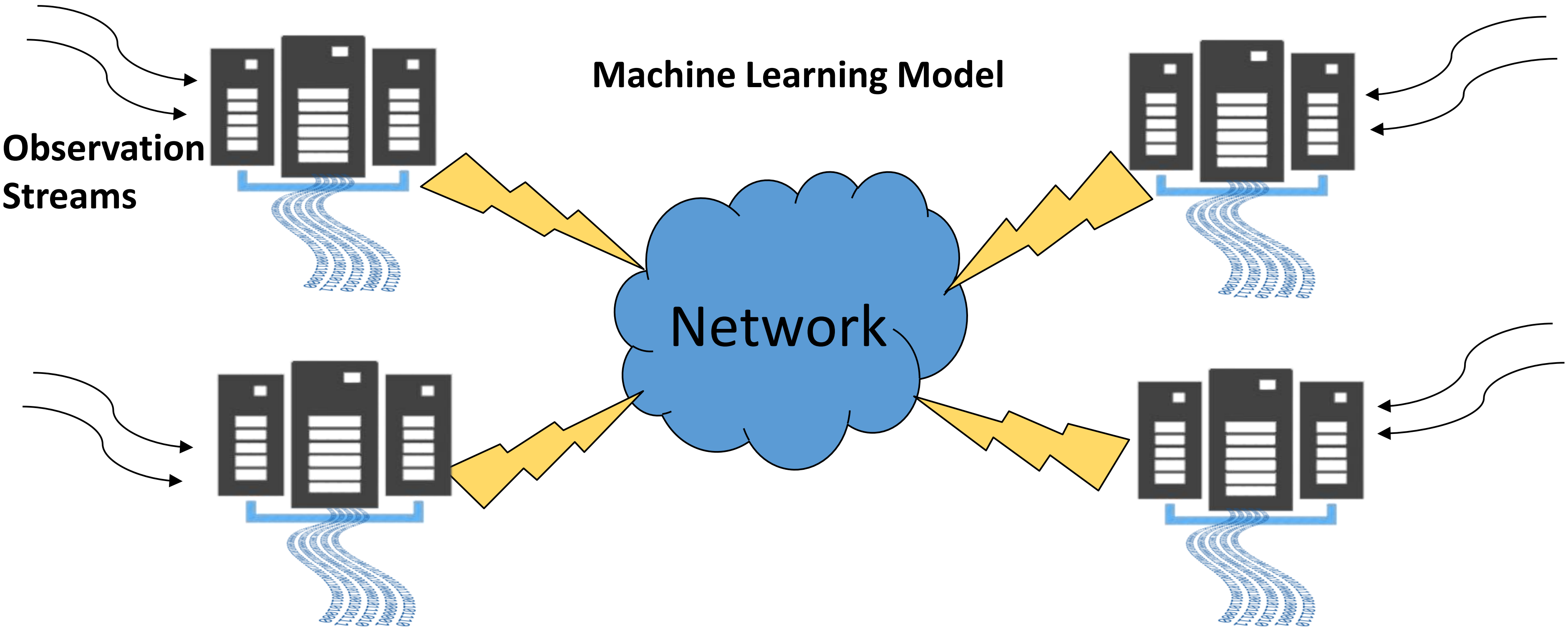
Lower Bounds

- 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!



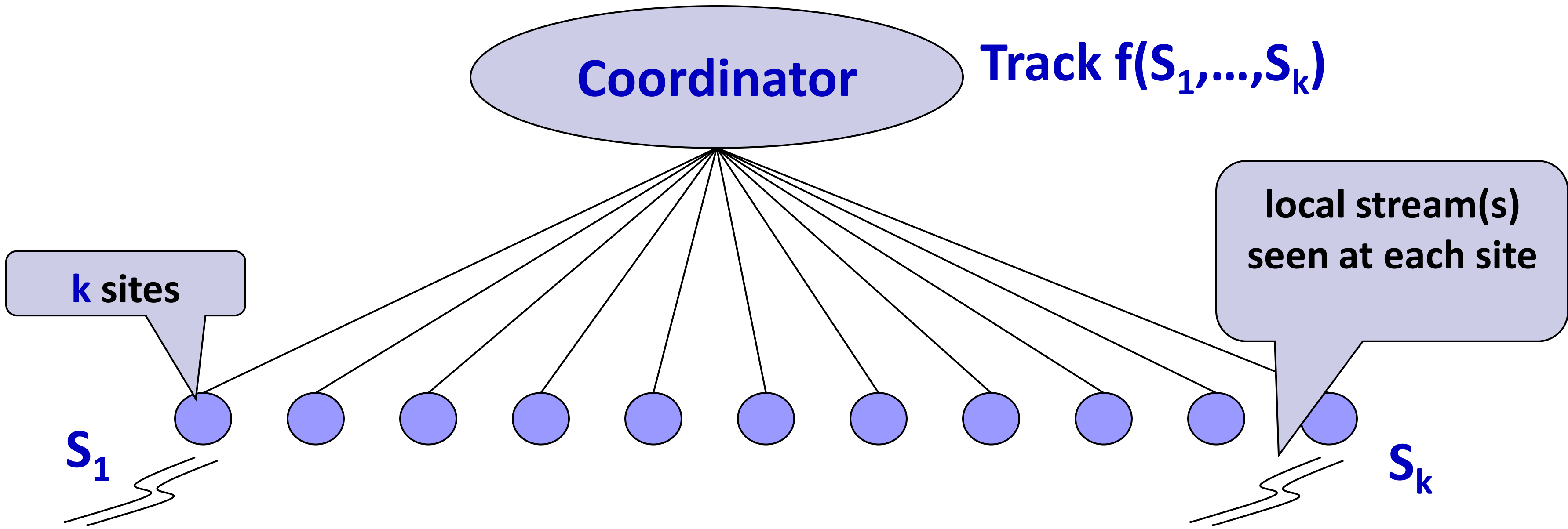
Part 2:
Applications in Machine Learning

1. Distributed Streaming Machine Learning



- Data continuously generated across distributed sites
- Maintain a model of data that enables predictions
- Communication-efficient algorithms are needed!

Continuous Distributed Model



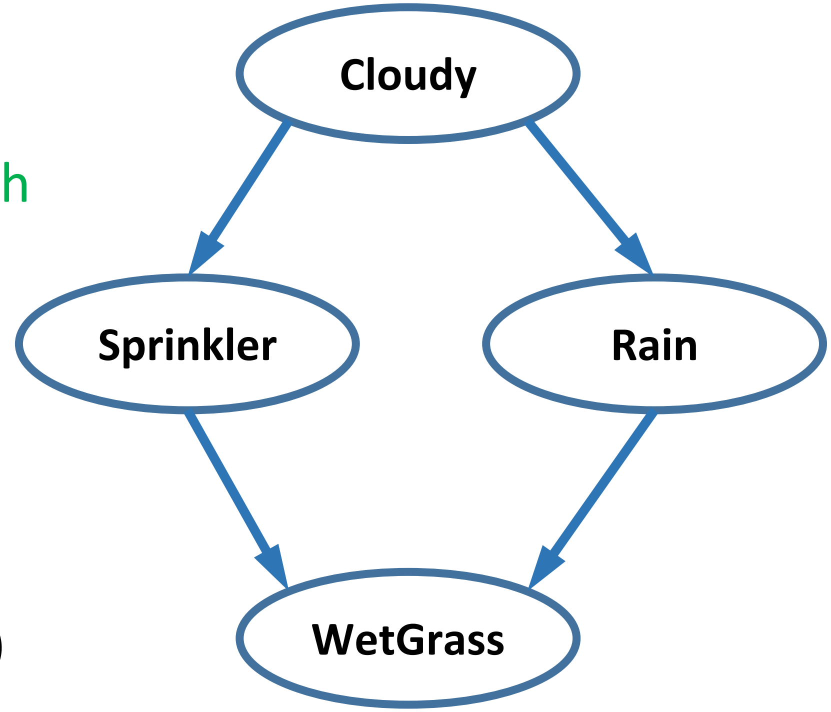
- Site-site communication only changes things by factor 2
- **Goal**: Coordinator *continuously tracks* (global) function of streams
  - Achieve communication poly(k, 1/ε, log n)
  - Also bound space used by each site, time to process each update

Challenges

- Monitoring is Continuous…
  - Real-time tracking, rather than one-shot query/response
- …Distributed…
  - Each remote site only observes part of the global stream(s)
  - Communication constraints*:* must minimize monitoring burden
- …Streaming…
  - Each site sees a high-speed local data stream and can be resource (CPU/memory) constrained
- …Holistic…
  - Challenge is to monitor the complete global data distribution
  - Simple aggregates (e.g., aggregate traffic) are easier

Graphical Model: Bayesian Network

- Succinct representation of a joint distribution of random variables
- Represented as a Directed Acyclic Graph
  - Node = a random variable
  - Directed edge = conditional dependency
- Node independent of its non-descendants given its parents
  e.g. (WetGrass ⫫ Cloudy) | (Sprinkler, Rain)
- Widely-used model in Machine Learning for Fault diagnosis, Cybersecurity

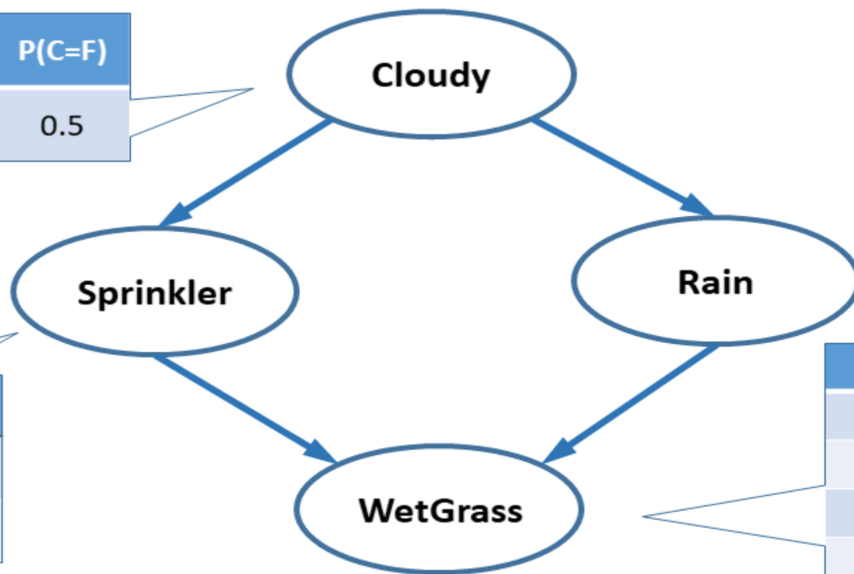


Weather Bayesian Network

https://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html

Conditional Probability Distribution (CPD)

Parameters of the Bayesian network can be viewed as a set of tables, one table per variable



P(C=T)	P(C=F)
0.5	0.5

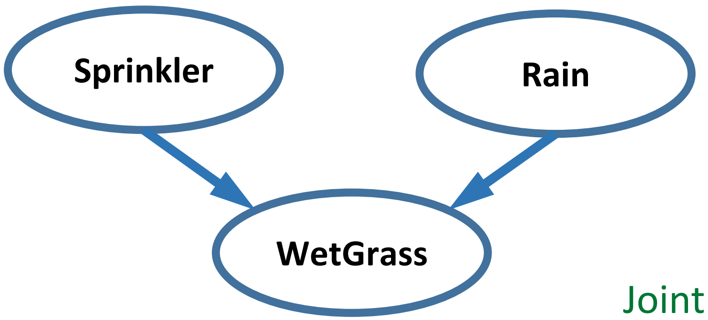
C	P(S=T)	P(S=F)
T	0.1	0.9
F	0.5	0.5

C	P(R=T)	P(R=F)
T	0.8	0.2
F	0.2	0.8

S	R	P(W=T)	P(W=F)
T	T	0.99	0.01
T	F	0.9	0.1
F	T	0.9	0.1
F	F	0.0	1.0

Goal: Learn Bayesian Network Parameters

The Maximum Likelihood Estimator (MLE) uses empirical conditional probabilities



$$Pr[W \mid S, R] = \frac{\Pr[W, S, R]}{\Pr[S, R]} = \frac{Freq(W, S, R)}{Freq(S, R)}$$

Joint Counter: W=T, W=F. Parent Counter: Total.

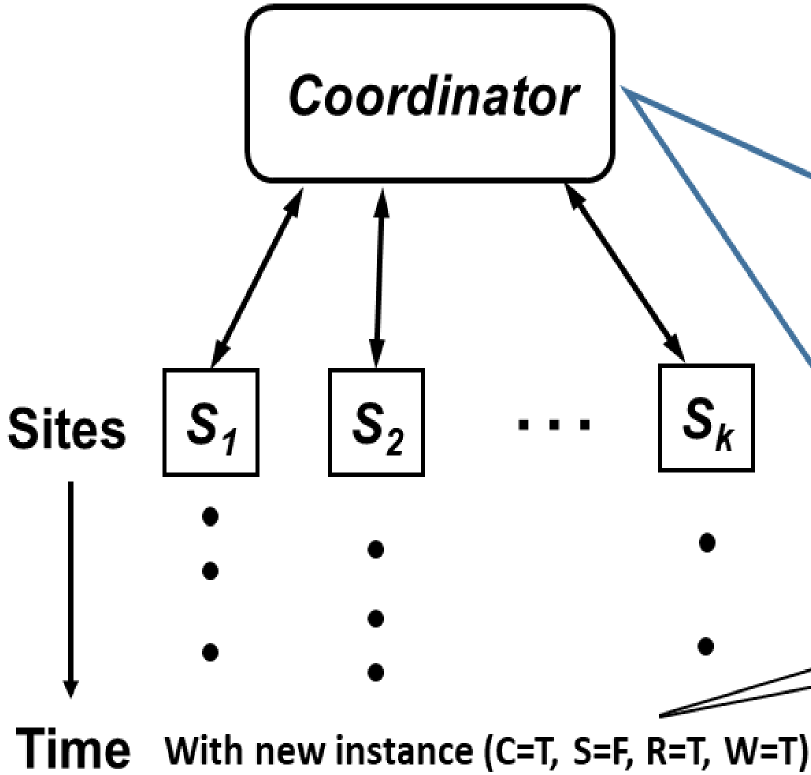
S	R	W=T	W=F	Total
T	T	99	1	100
T	F	9	1	10
F	T	45	5	50
F	F	0	10	10

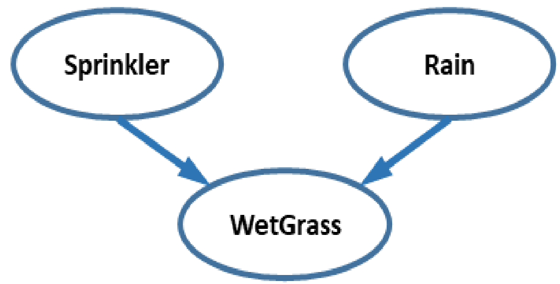
Counter Table of WetGrass

S	R	P(W=T)	P(W=F)
T	T	99/100 = 0.99	0.01
T	F	0.9	0.1
F	T	0.9	0.1
F	F	0.0	1.0

CPD of WetGrass

Distributed Bayesian Network Learning





S	R	W=T	W=F	Total
T	T	99	1	100
T	F	9	1	10
F	T	45->46	5	50->51
F	F	0	10	10

Counter Table of WetGrass

S	R	P(W=T)	P(W=F)
T	T	0.99	0.01
T	F	0.9	0.1
F	T	0.9->0.902	0.1->0.098
F	F	0.0	1.0

CPDs of WetGrass

Parameters changing with new stream instance

Naïve Solution: Exact Counting (Exact MLE)

- Each arriving event at a site sends a message to a coordinator
  - Updates counters corresponding to all the value combinations from the event
- Total communication is proportional to the number of events
  - Can we reduce this?
- Observation: we can tolerate some error in counts
  - Small changes in large enough counts won't affect probabilities
  - Some error already from variation in what order events happen
- Replace exact counters with approximate counters
  - A foundational distributed question: how to count approximately?

Distributed Approximate Counting

[Huang, Yi, Zhang PODS'12]

- We have k sites, each site runs the same algorithm:
  - For each increment of a site's counter:
    - Report the new count n'_i with probability p
  - Estimate n_i as $n'_i - 1 + 1/p$ if $n'_i > 0$, else estimate as 0
- Estimator is unbiased, and has variance less than $1/p^2$
- Global count n estimated by sum of the estimates n_i
- How to set p to give an overall guarantee of accuracy?
  - Ideally, set p to $\sqrt{(k \log 1/\delta)}/\varepsilon n$ to get εn error with probability $1-\delta$
  - Work with a coarse approximation of n up to a factor of 2
- Start with $p=1$ but decrease it when needed
  - Coordinator broadcasts to halve p when estimate of n doubles
  - Communication cost is proportional to $O(k \log(n) + \sqrt{k}/\varepsilon)$

Challenge in Using Approximate Counters

How to set the approximation parameters for learning Bayes nets?

1. Requirement: maintain an accurate model
   (i.e. give accurate estimates of probabilities)

$$e^{-\epsilon} \leq \frac{\tilde{P}(\boldsymbol{x})}{\hat{P}(\boldsymbol{x})} \leq e^{\epsilon}$$

   where:
   ϵ is the global error budget,
   $\boldsymbol{x}$ is the given any instance vector,
   $\tilde{P}(\boldsymbol{x})$ is the joint probability using approximate algorithm,
   $\hat{P}(\boldsymbol{x})$ is the joint probability using exact counting (MLE)

2. Objective: minimize the communication cost of model maintenance

We have freedom to find different schemes to meet these requirements

ϵ −Approximation to the MLE

- Expressing joint probability in terms of the counters:

$$\hat{P}(\boldsymbol{x}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{C(X_i, par(X_i))}{C(par(X_i))} \qquad \tilde{P}(\boldsymbol{x}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{A(X_i, par(X_i))}{A(par(X_i))}$$

  where:
  - A is the approximate counter
  - C is the exact counter
  - $par(X_i)$ are the parents of variable X_i
- Define local approximation factors as:
  – α_i: approximation error of counter $A(X_i,\ par(X_i))$
  – β_i: approximation error of parent counter $A(par(X_i))$
- To achieve an ϵ-approximation to the MLE we need:

$$e^{-\epsilon} \leq \prod_{i=1}^{n}((1 \pm \alpha_i) \cdot (1 \pm \beta_i)) \ \leq e^{\epsilon}$$

Algorithm choices

We proposed three algorithms [C, Tirthapura, Yu ICDE 2018]:

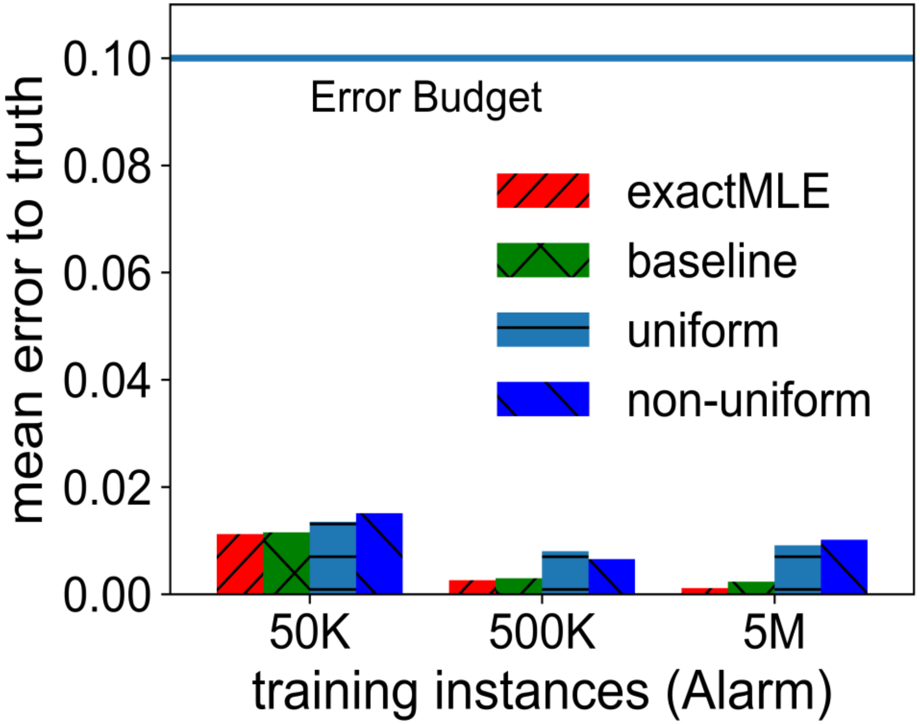
- Baseline algorithm: divide error budgets uniformly across all counters, α_i, $\beta_i \propto \varepsilon/n$
- Uniform algorithm: analyze total error of estimate via variance, rather than separately, so α_i, $\beta_i \propto \varepsilon/\sqrt{n}$
- Non-uniform algorithm: calibrate error based on cardinality of attributes (J_i) and parents (K_i), by applying optimization problem

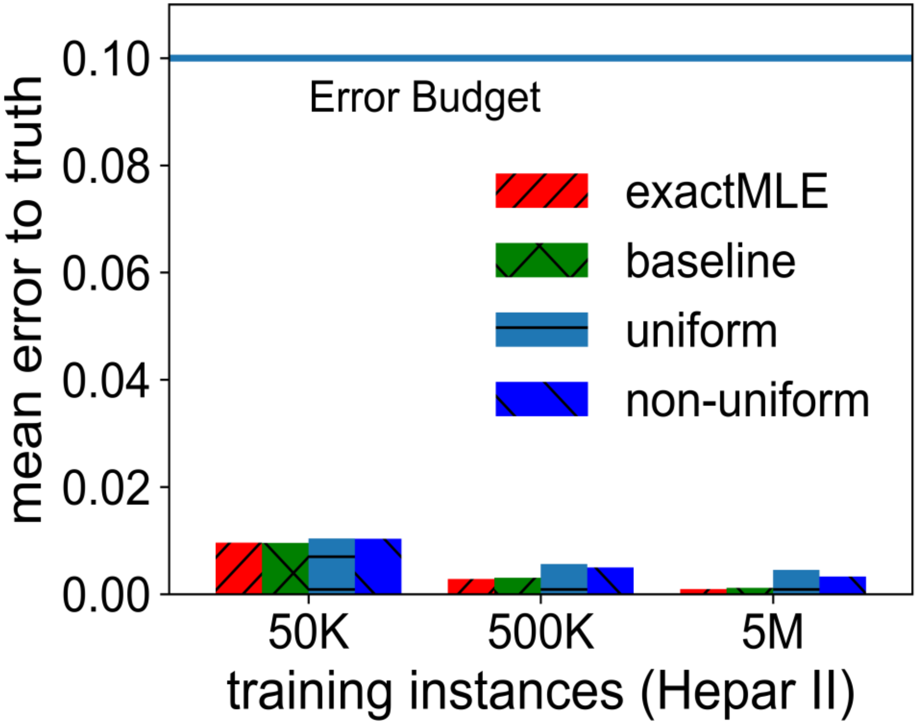
Algorithms Result Summary

Algorithm	Approx. Factor of Counters	Communication Cost (messages)
Exact MLE	None (exact counting)	$O(mn)$
Baseline	$O(\epsilon/n)$	$O(n^2 \cdot \log m \,/\, \epsilon)$
Uniform	$O(\epsilon/\sqrt{n})$	$O\ (n^{1.5} \cdot \log m \,/\, \epsilon)$
Non-uniform	$O\left(\epsilon \cdot \frac{J_i^{1/3} K_i^{1/3}}{\alpha}\right), O\left(\epsilon \cdot \frac{K_i^{1/3}}{\beta}\right)$	at most Uniform

ϵ: error budget, n: number of variables, m: total number of observations
J_i: cardinality of variable X_i, K_i: cardinality of X_i's parents
α is a polynomial function of J_i and K_i , β is a polynomial function of K_i

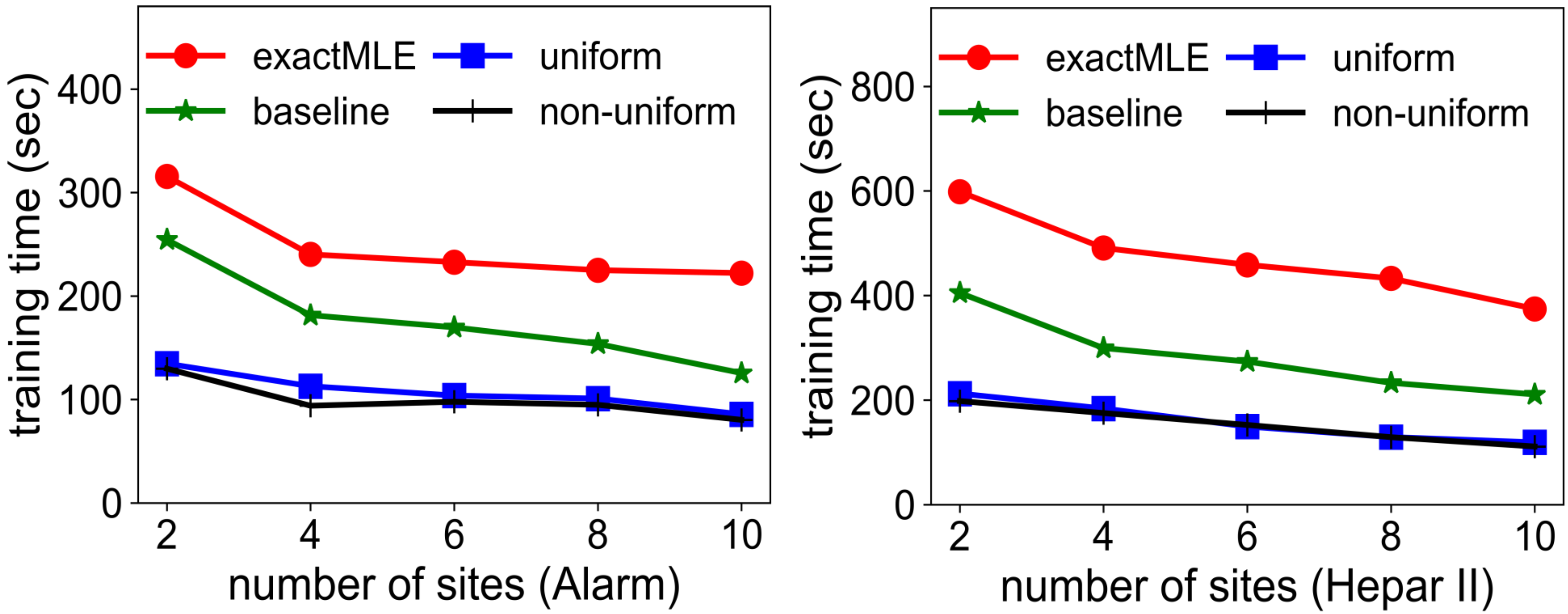
Empirical Accuracy





error to ground truth vs. training instances
(number of sites: 30, error budget: 0.1)
real world Bayesian networks Alarm (small), Hepar II (medium)

Communication Cost (training time)



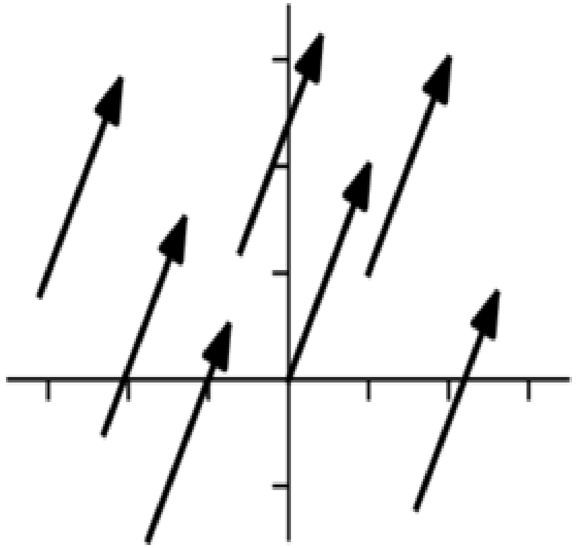
training time vs. number of sites
(500K training instances, error budget: 0.1)
time cost (communication bound) on AWS cluster

Conclusions

- Communication-Efficient Algorithms to maintaining a provably good approximation for a Bayesian Network
- Non-Uniform approach is (marginally) the best, and adapts to the structure of the Bayesian network
- Experiments show reduced communication and similar prediction errors as the exact model
- Algorithms can be extended to perform classification and other ML tasks
- Open problems: extend to richer models, learning the graph

2. Sketching for Constrained Regression

- Linear algebra computations are key to much machine learning
- We seek efficient scalable linear algebra approximate solutions making use of sketching algorithms (random projections)
  - We find efficient approximate algorithms for constrained regression
  - We show new approaches based on sketching which are fast and accurate



Constrained Least Squares Regression

- Regression: Input is $A \in \mathbb{R}^{n \times d}$ and target vector $b \in \mathbb{R}^n$
  - Least Squares formulation: find $x = \text{argmin} \, \|Ax - b\|_2$
  - Takes time $O(nd^2)$ centralized to solve via normal equations
- Can be approximated via reducing dependency on n by compressing into columns of length roughly d/ϵ^2 (JLT)
  - Can be performed distributed with some restrictions
- Constrained regression imposes additional constraints:
  - x must lie within a (convex) set C
  - Good solution methods via convex optimization, with a time cost

Regression via Sketching

- Sketch-and-solve paradigm: solve $x' = \text{argmin}_{x \in C} \|S(Ax-b)\|^2$
  - Find the x that seems to solve the problem under sketch matrix S
  - Can prove that it finds $\|Ax' - b\|^2 \leq (1+\varepsilon) \|Ax_{OPT} - b\|^2$
    i.e. a solution whose cost is near optimal
  - However, does not guarantee to approximate vector x_{OPT} itself
- Iterative Hessian Sketch [Pilanci&Wainwright 16]: iterate to solve
  - $x^{t+1} = \text{argmin}_{x \in C} \ \frac{1}{2}\|(S^{t+1}A)(x - x^t)\|^2 - \langle A^T(b - Ax^t), x - x^t \rangle$
  - Use fresh sketches ($S^1, S^2, S^3 \ldots$) to move towards the solution
  - Faster than exact solution since (SA) is much smaller than A
  - Will find an x' that is close to x_{OPT}

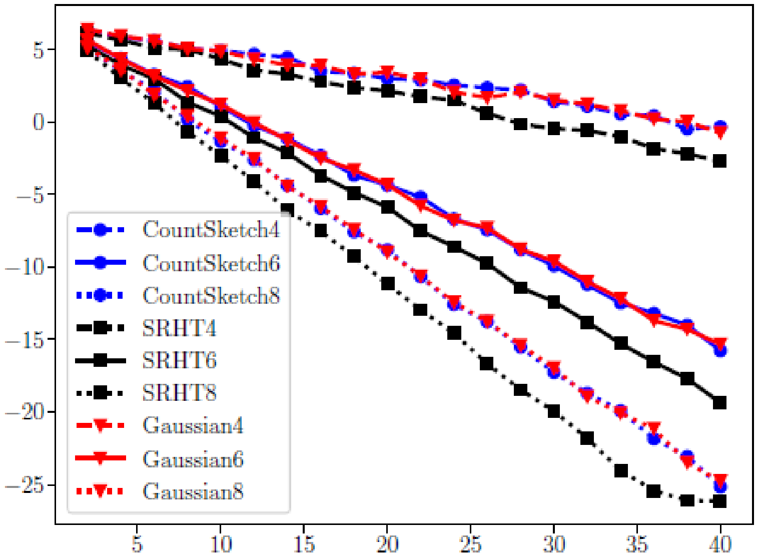
Instantiating IHS

- Iterative Hessian Sketch imposes some requirements on sketch
  - Subgaussianity: $E[SS^T]$ is a scaled identity, and rows of the sketch do not stretch arbitrary vectors with high probability
  - Spectral bound: $E[S^T(SS^T)^{-1}S]$ is bounded by a scaled identity
- Several sketches are known to meet these conditions:
  - (Dense) Gaussian sketches: entries are IID Gaussian
  - Subsampled Randomized Hadamard Transform (SRHT): composition of a sampling and sign-flipping with the Hadamard transform
- We show that CountSketch also works [Cormode, Dickens 19]
  - Not every step of IHS will preserve all directions, but with sufficient iterations, we converge
  - CountSketch is fast(er) when the input is sparse

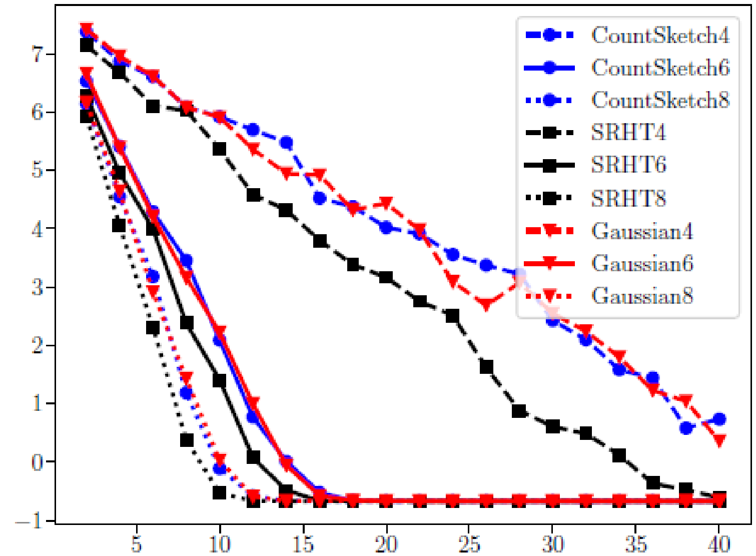
Experimental Study

- We evaluate LASSO regression with regularization parameter λ:
  $x_{OPT} = \text{argmin}_{x \text{ in } R^d} \frac{1}{2} \|Ax-b\|_2^2 + \lambda\|x\|_1$
- We evaluate on synthetic and real data:
  - YearPredictionsMSD: 515K x 91, fully dense
  - Slice: 53K x 387, 0.36 dense
  - w8a: 50K x 301, 0.042 dense
- Main parameter is how big to make the sketches?
  - We consider multiples of the input dimension, d: 4d to 10d

IHS with iterations for LASSO



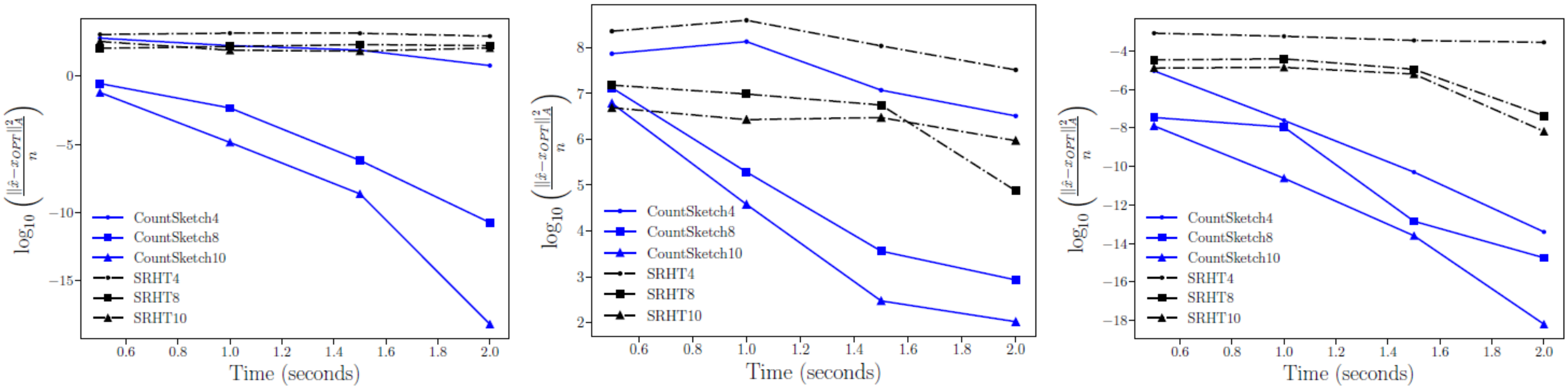
(a) Error to optimal estimator



(b) Error to truth

- All sketch methods converge to a common error level after sufficiently many iterations on synthetic data
- Number of iterations is only part of the story: not all iterations are equal(ly fast)

IHS accuracy versus time for LASSO



(a) YearPredictionsMSD (b) Slice (c) w8a

- CountSketch approach shows rapid convergence to approximate solution
- Larger sketch achieves better error in same time
- CountSketch performs well across different datasets with differing sparsity levels

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, 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
- Summaries with privacy to compactly gather accurate data: extra randomization is used to hide personal information

Final Summary

- There are two approaches in response to growing data sizes
  – Scale the computation up; scale the data down
- Summarization can be a useful tool in machine learning
  – Allows approximate solutions over distributed data
- Many open problems in this broad area
  – Machine learning/linear algebra a rich source of problems

European Research Council
Established by the European Commission

EPSRC
Engineering and Physical Sciences
Research Council

THE ROYAL SOCIETY