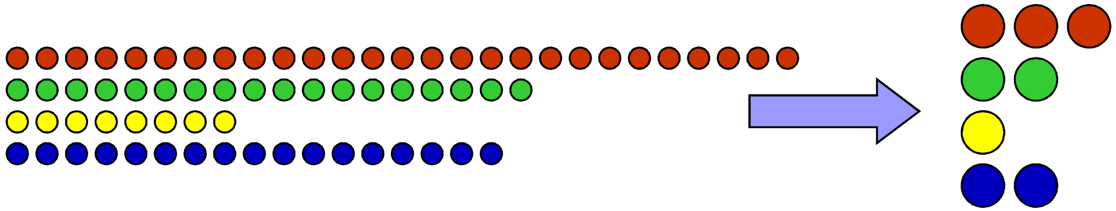
# Data Summarization
## and
# Distributed Computation



**Graham Cormode**

University of Warwick

G.Cormode@Warwick.ac.uk

# Agenda for the talk

- My (patchy) history with PODC:

**2011**

[c60] Graham Cormode, Ke Yi:
**Tracking distributed aggregates over time-based sliding windows.** PODC 2011: 213-214

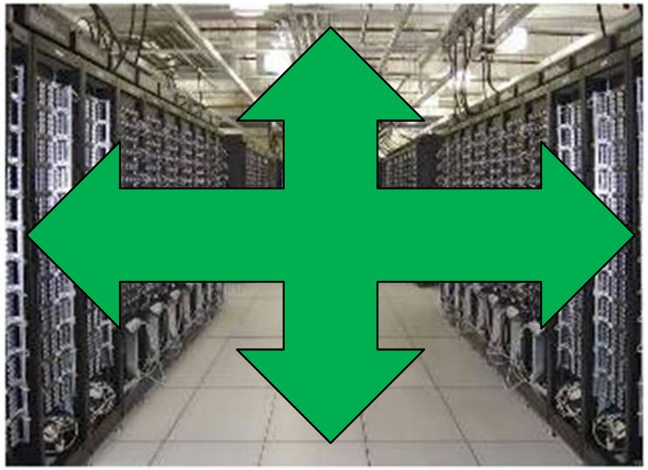
**2007**

[c31] Graham Cormode, Srikanta Tirthapura, Bojian Xu:
**Time-decaying sketches for sensor data aggregation.** PODC 2007: 215-224

- This talk: recent examples of distributed summaries
  - Learning graphical models from distributed streams
  - Deterministic distributed summaries for high-dimensional regression

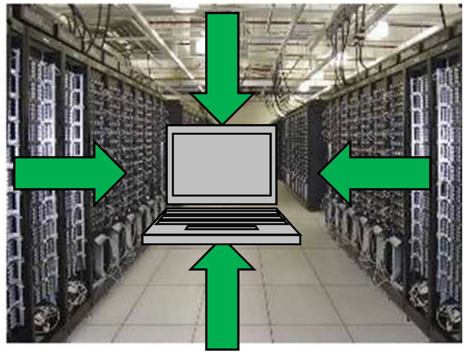
# Computational scalability and "big" data

- Industrial distributed computing means **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



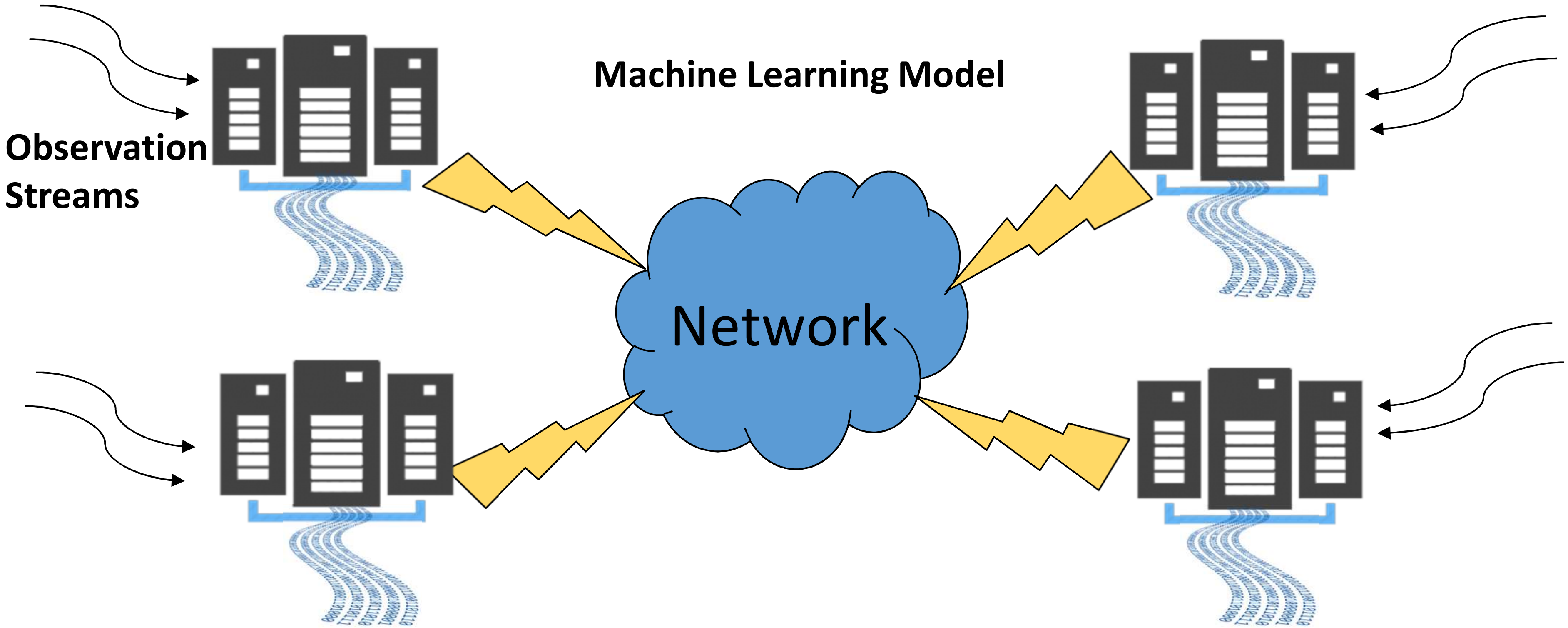
- Scaling up comes with its disadvantages:
  - Expensive (hardware, equipment, **energy**), still not always fast
- This talk is not about this approach!

# Downsizing data



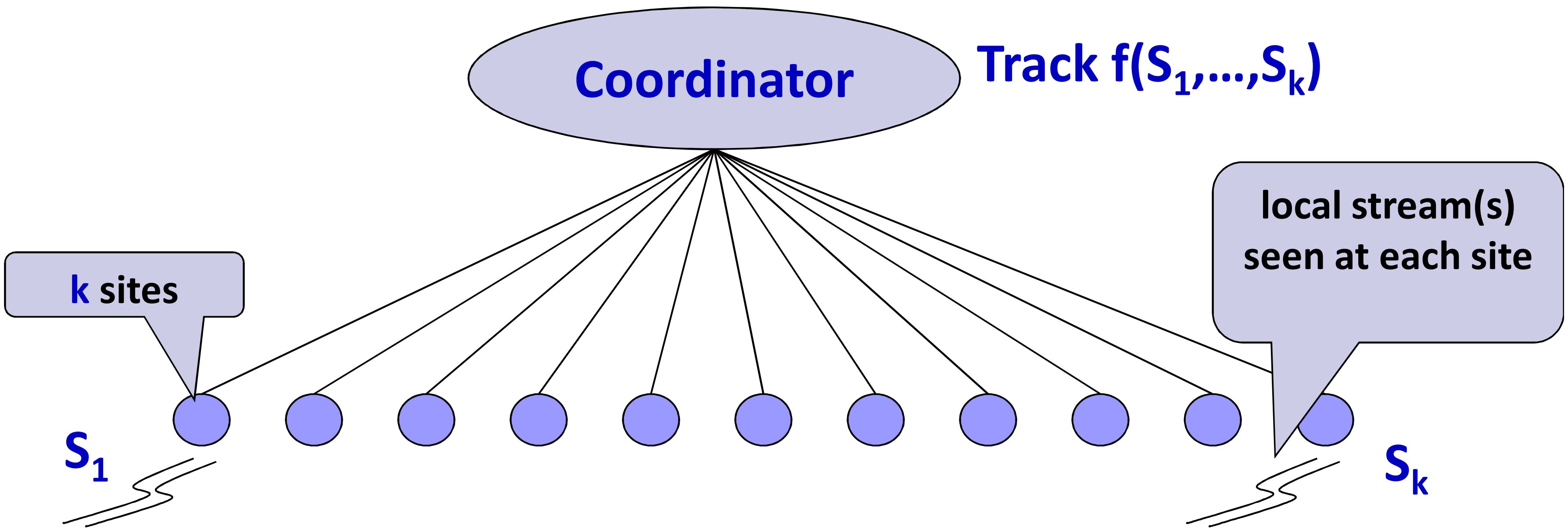
- A second approach to computational scalability: **scale down the data!**
  - A compact representation of a large data set
  - Capable of being analyzed on a single machine
  - What we finally want is small: human readable analysis / decisions
  - Necessarily gives up some accuracy: approximate answers
  - 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

# 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



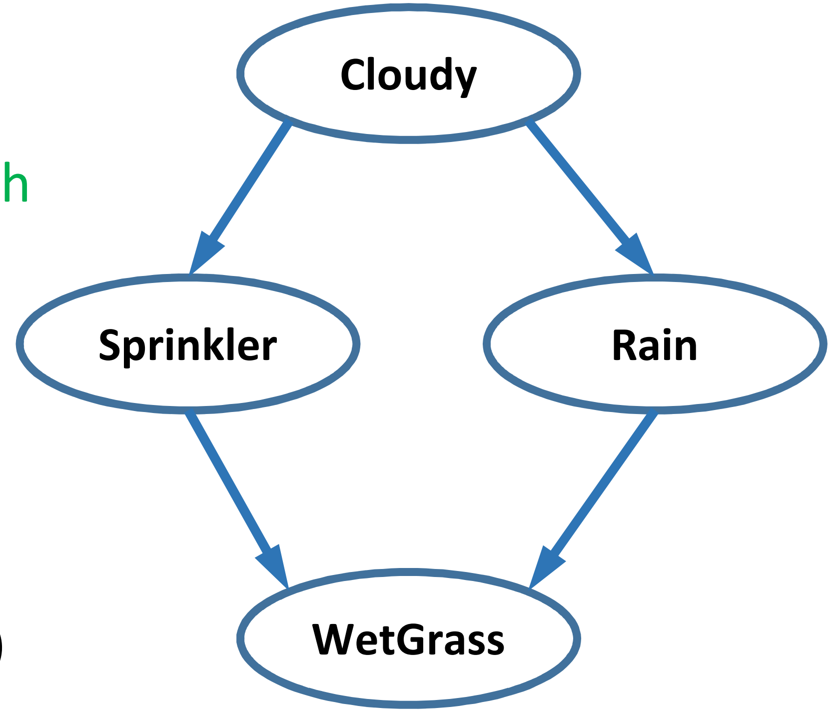

- Site-site communication only changes things by factor 2
- **Goal**: Coordinator *continuously tracks* (global) function of streams
  - Achieve communication poly(k, 1/ε, log n)
  - Also bound space used by each site, time to process each update

# 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

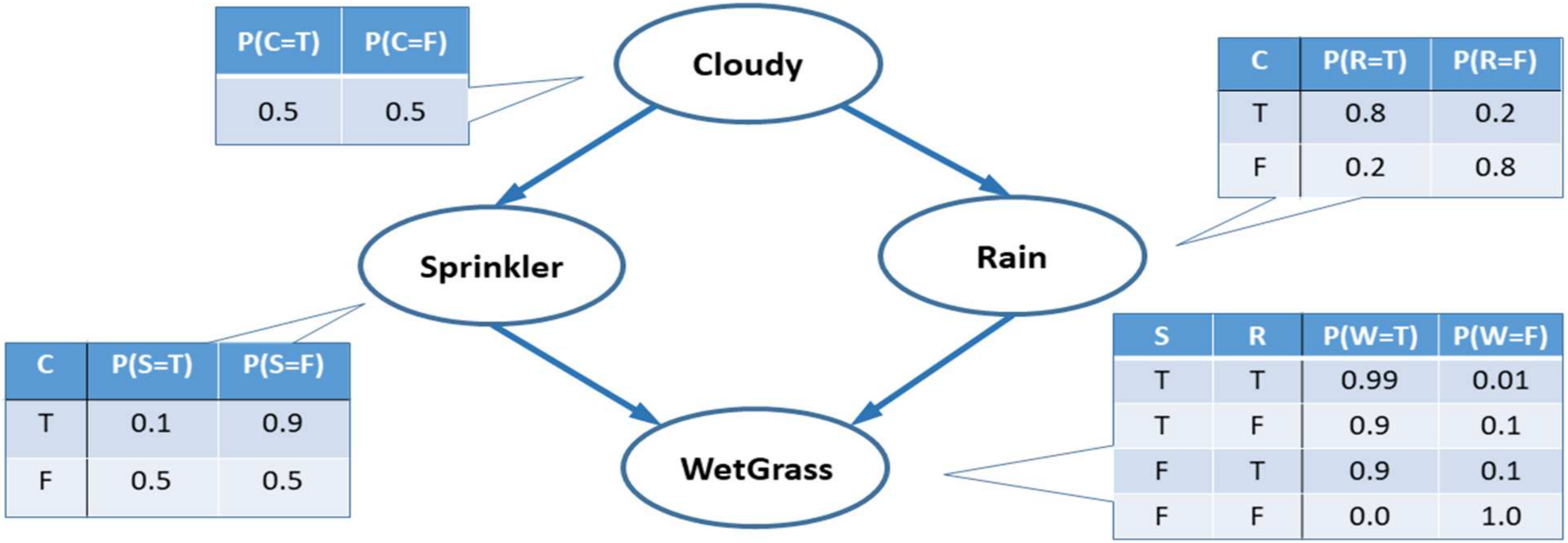



**Weather Bayesian Network**

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 |

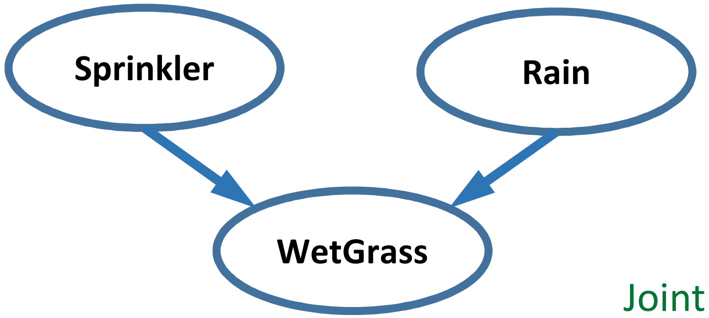
| C | P(S=T) | P(S=F) |
|---|---|---|
| T | 0.1 | 0.9 |
| F | 0.5 | 0.5 |

| 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 | Parent Counter

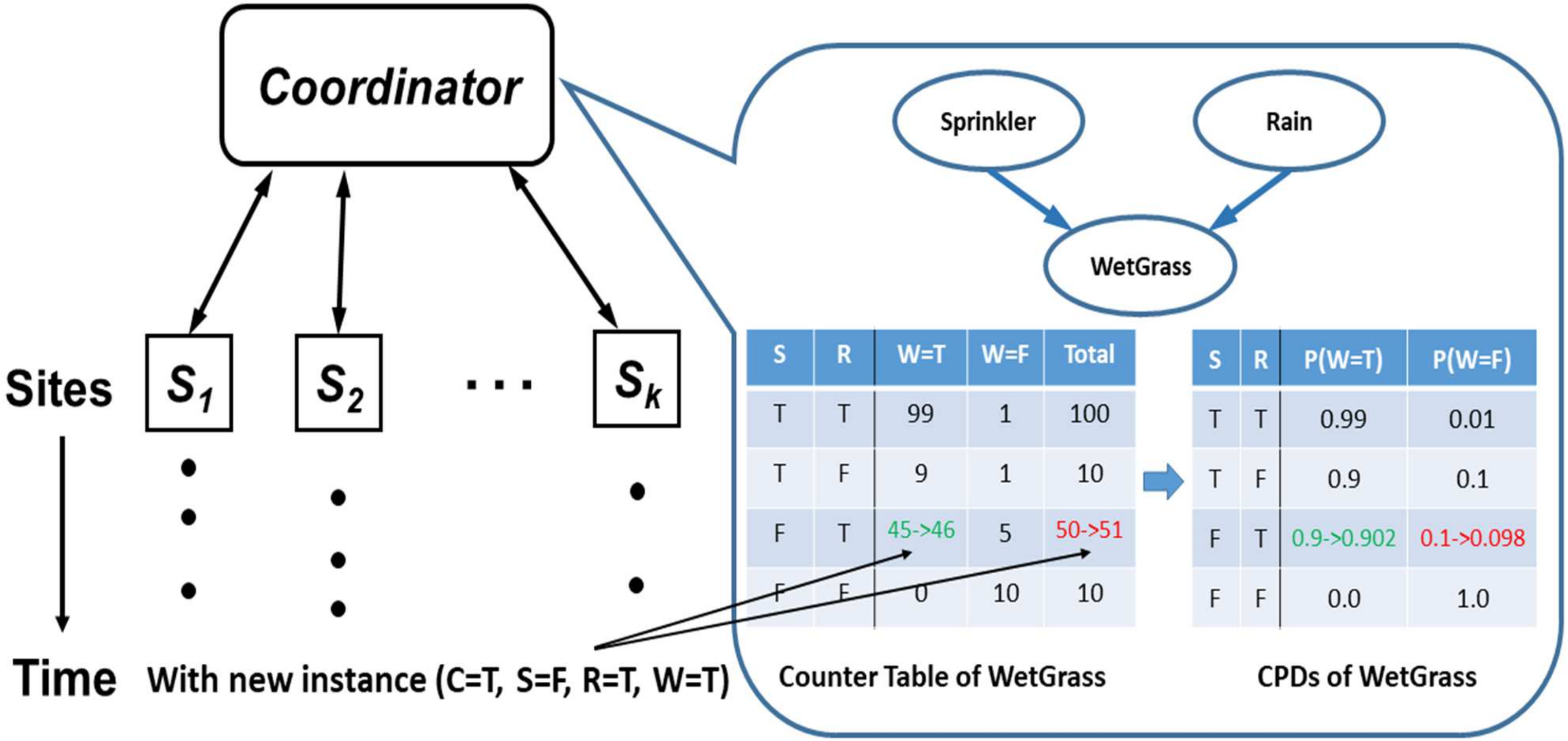
| S | R | W=T | W=F | Total |
|---|---|---|---|---|
| T | T | 99 | 1 | 100 |
| T | F | 9 | 1 | 10 |
| F | T | 45 | 5 | 50 |
| F | F | 0 | 10 | 10 |

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 |

| 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 |

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 $\varepsilon n$ 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:
   $\epsilon$ 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

# $\epsilon$ −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:
  - $\alpha_i$: approximation error of counter $A(X_i,\ par(X_i))$
  - $\beta_i$: approximation error of parent counter $A(par(X_i))$
- To achieve an $\epsilon$-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, $\alpha_i$, $\beta_i \propto \varepsilon/n$
- Uniform algorithm: analyze total error of estimate via variance, rather than separately, so $\alpha_i$, $\beta_i \propto \varepsilon/\sqrt{n}$
- Non-uniform algorithm: calibrate error based on cardinality of attributes ($J_i$) and parents ($K_i$), by applying optimization problem

# Algorithms Result Summary

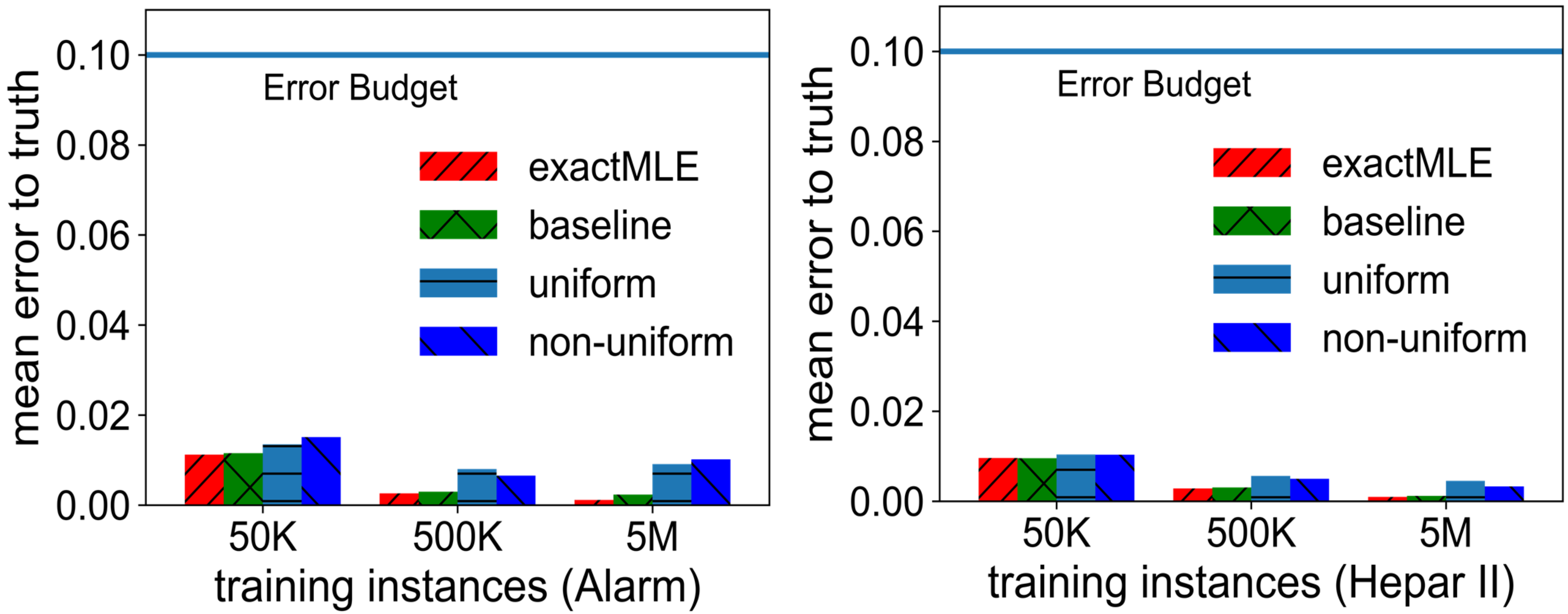
| Algorithm | Approx. Factor of Counters | Communication Cost (messages) |
|---|---|---|
| Exact MLE | None (exact counting) | $O(mn)$ |
| Baseline | $O(\epsilon/n)$ | $O(n^2 \cdot \log m \,/\, \epsilon)$ |
| Uniform | $O(\epsilon/\sqrt{n})$ | $O\ (n^{1.5} \cdot \log m \,/\, \epsilon)$ |
| Non-uniform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

$\epsilon$: 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

$\alpha$ is a polynomial function of $J_i$ and $K_i$ , $\beta$ is a polynomial function of $K_i$

# Empirical Accuracy

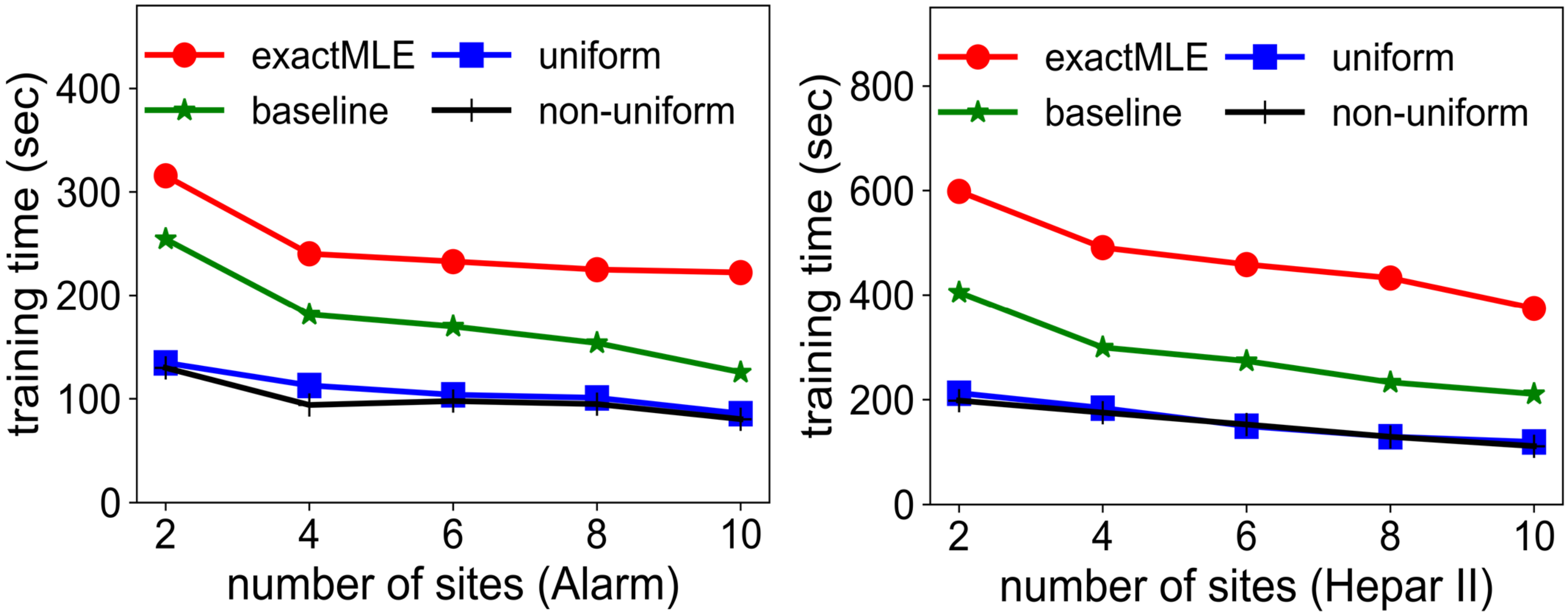



error to ground truth vs. training instances

(number of sites: 30, error budget: 0.1)

real world Bayesian networks Alarm (small), Hepar II (medium)

# Communication Cost (training time)

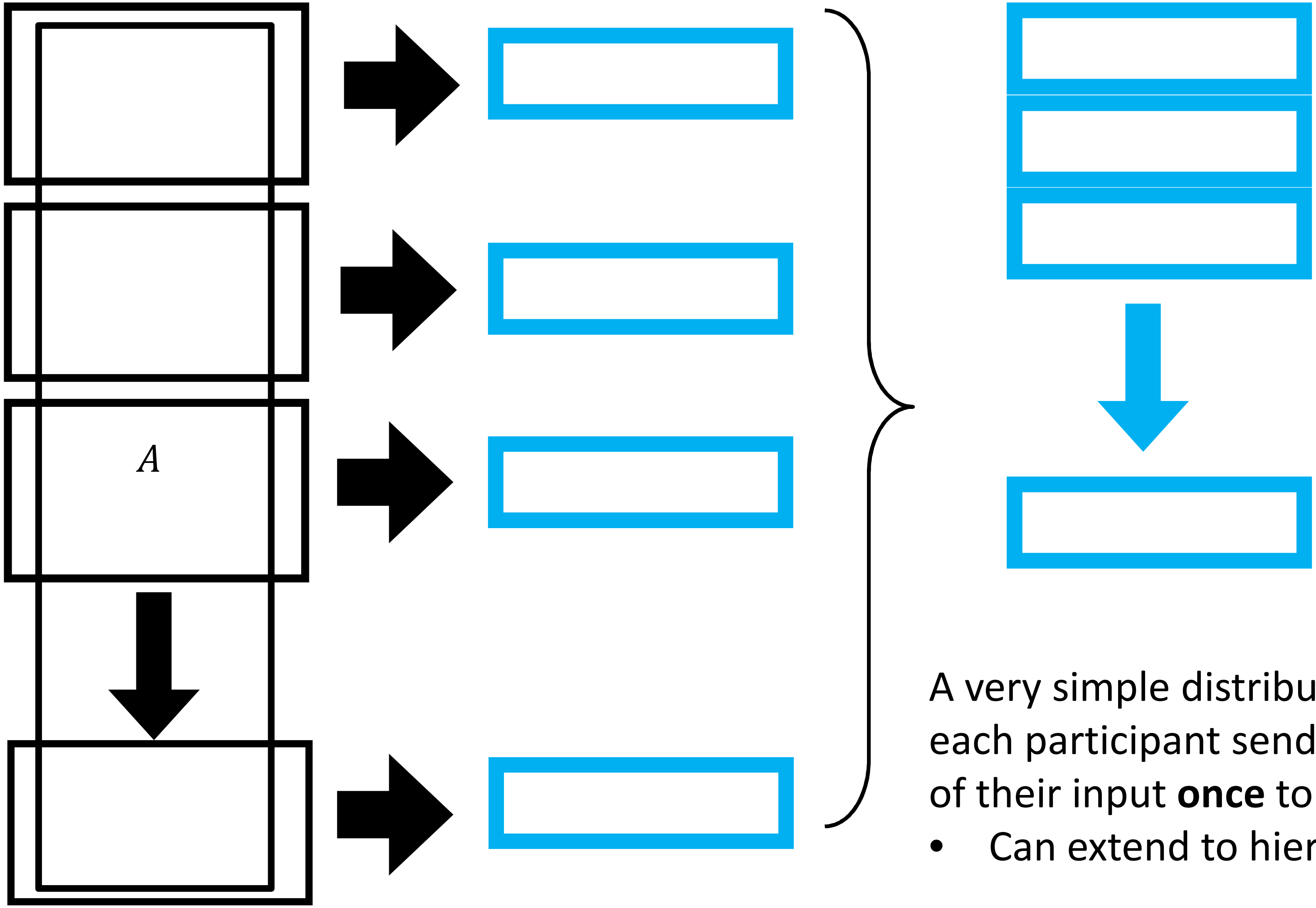



training time vs. number of sites
(500K training instances, error budget: 0.1)
time cost (communication bound) on AWS cluster

# Conclusions

- Communication-Efficient Algorithms to maintaining a provably good approximation for a Bayesian Network
- Non-Uniform approach is 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

# 2. Distributed Data Summarization



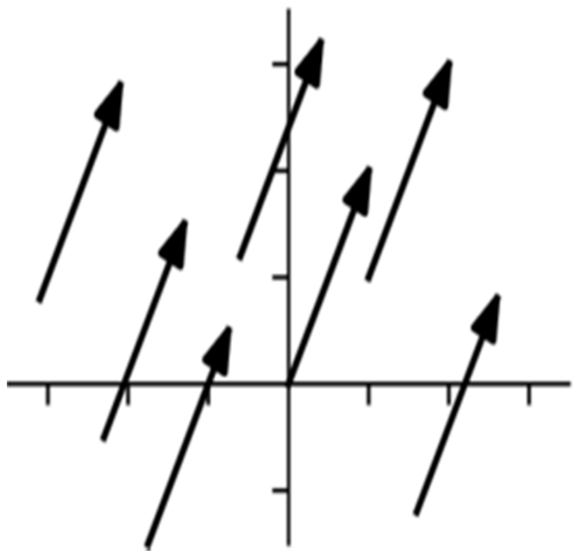

A very simple distributed model: each participant sends summary of their input **once** to aggregator

- Can extend to hierarchies

# Distributed Linear Algebra

- Linear algebra computations are key to much of machine learning
- We seek efficient scalable linear algebra approximate solutions
- We find deterministic distributed algorithms for $L_p$-regression

[C Dickens Woodruff ICML 2018]



# Ordinary Least Squares Regression

- Regression: Input is $A \in \mathbb{R}^{n \times d}$ and target vector $b \in \mathbb{R}^n$
  - OLS formulation: find $x = \mathrm{argmin} \|Ax - b\|_2$
  - 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 $d/\epsilon^2$ (JLT)
  - Can be performed distributed with some restrictions
- $L_2$ (Euclidean) space is well understood, what about other $L_p$?

# Main Tool for $L_p$: Well Conditioned Basis

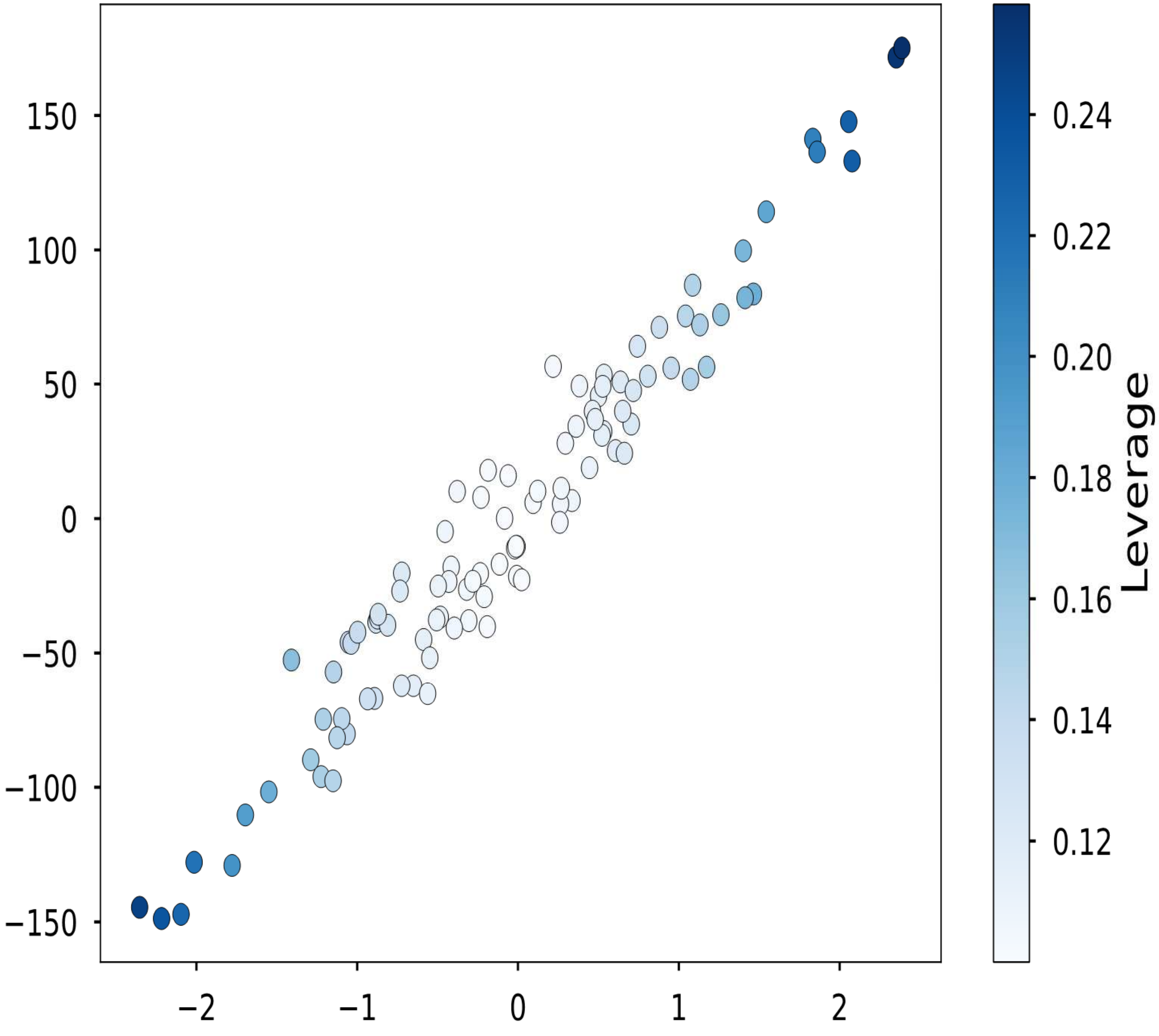
- A well-conditioned basis is akin to an '$L_p$ orthonormal basis'
- $U$ is an $(\alpha, \beta, p)$ wcb for the $col(A)$ if in *entrywise* $p$-norm:
  - $\|U\|_p \leq \alpha$
  - $\|z\|_q \leq \beta \|Uz\|_p$ when $q = 1/(1+p)$ (dual norm)
  - Can find $\alpha, \beta$ at most a small $poly(d) \approx d^{\frac{1}{p} \pm \frac{1}{2}}$
- $U$ can be found in $O(nd^2 + nd^5 \log n)$

# Leverage scores

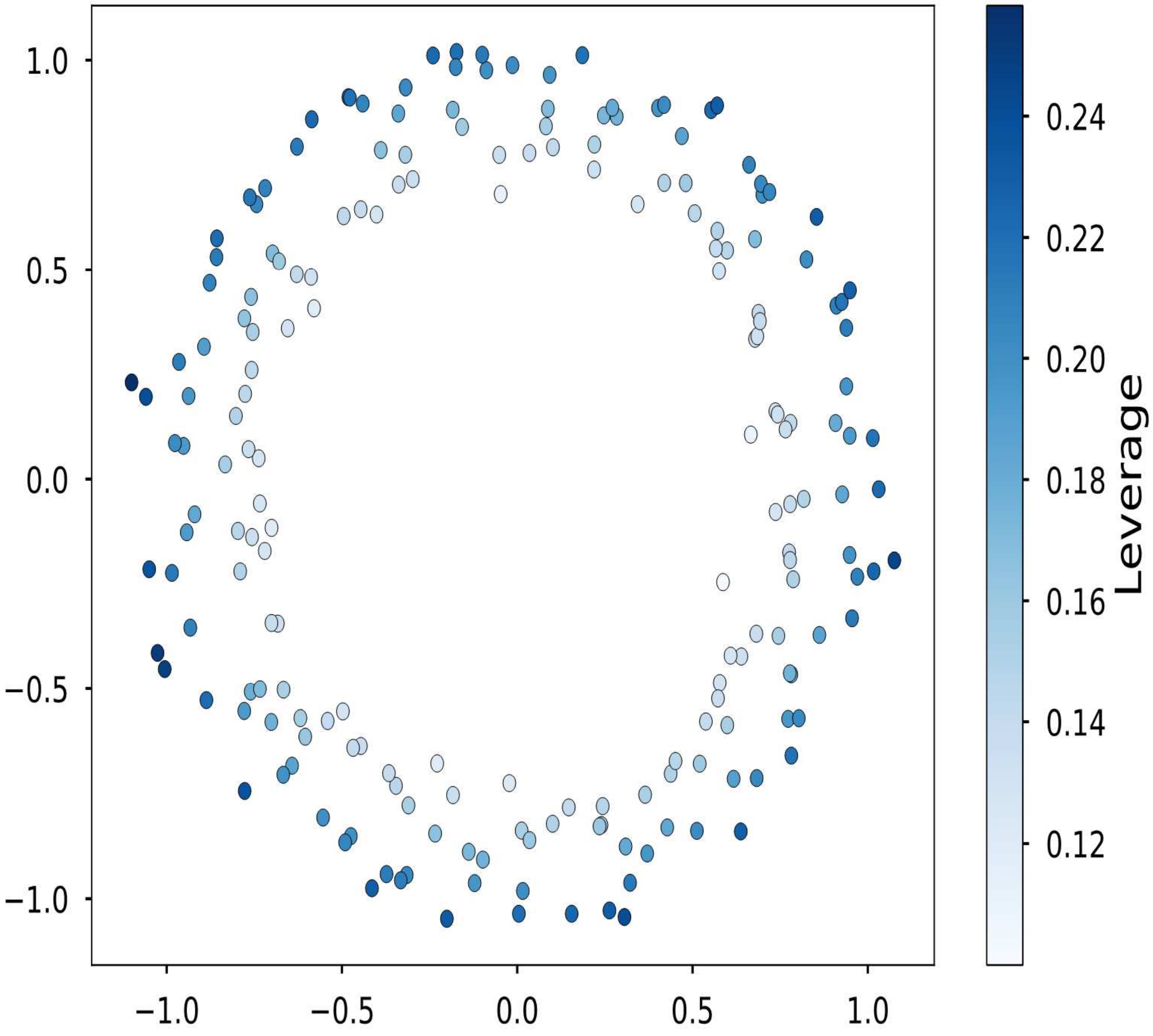
- $L_2$ leverage scores defined via row norms of orthonormal basis
  - Measure distance from the mean of the points
  - In $[0,1]$ and measure contribution to direction
  - More unique points have higher leverage
  - Approximate the shape of the data

$L_p$-leverage scores: ~~orthonormal~~

→ well-conditioned basis



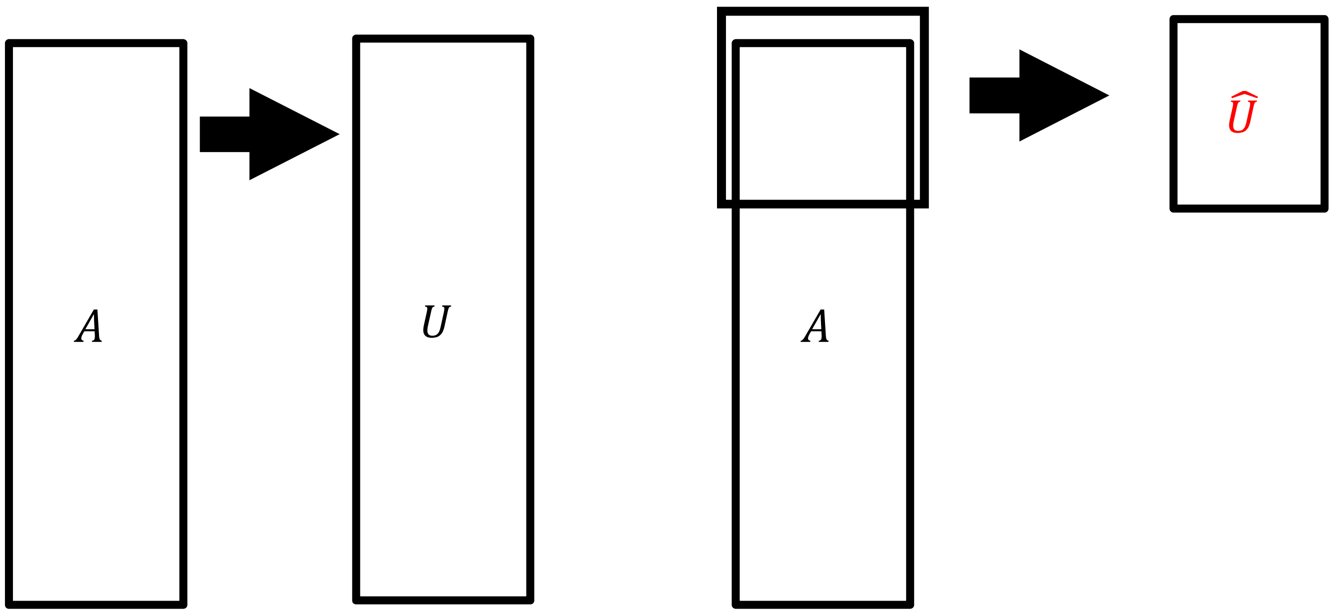
150
100
50
0
−50
−100
−150
−2
−1
0
1
2
0.24
0.22
0.20
0.18
0.16
0.14
0.12
Leverage



1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
0.24
0.22
0.20
0.18
0.16
0.14
0.12
Leverage

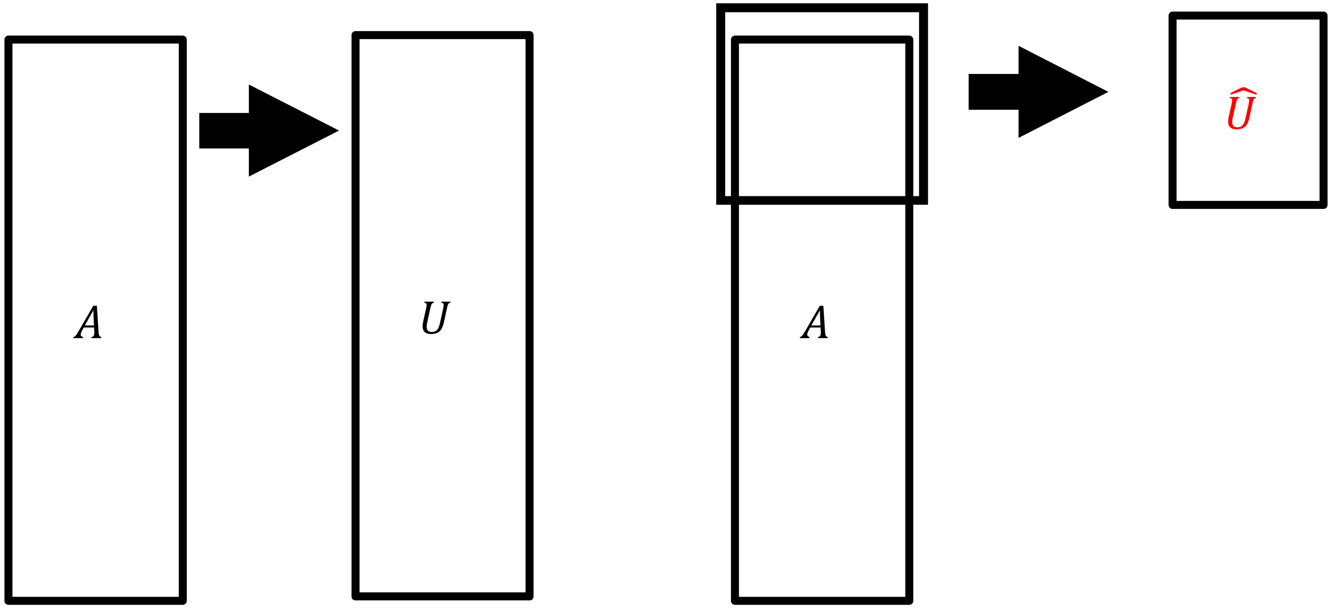
# $L_p$ leverage scores

- For $U$ a well-conditioned basis, *leverage scores* are given by row norms
- Can we find rows of high leverage without seeing the full matrix?



# $L_p$ leverage scores

- Idea: find local leverage scores in $\widehat{U}$ and communicate only the most important rows to central coordinator
- *Local scores* found by computing a well-conditioned basis on a subset of the input



# $L_p$ leverage scores - theory

- Key result shows that globally important rows remain important (up to some $poly(d)$ rescaling)

| Locally unimportant | Globally unimportant |
|---|---|
| X | X |

- Sum of the leverage rows is $\|U\|_p^p \leq \mathrm{poly}(d)$ so there can't be too many rows with high leverage score

# Application: $L_p$-regression

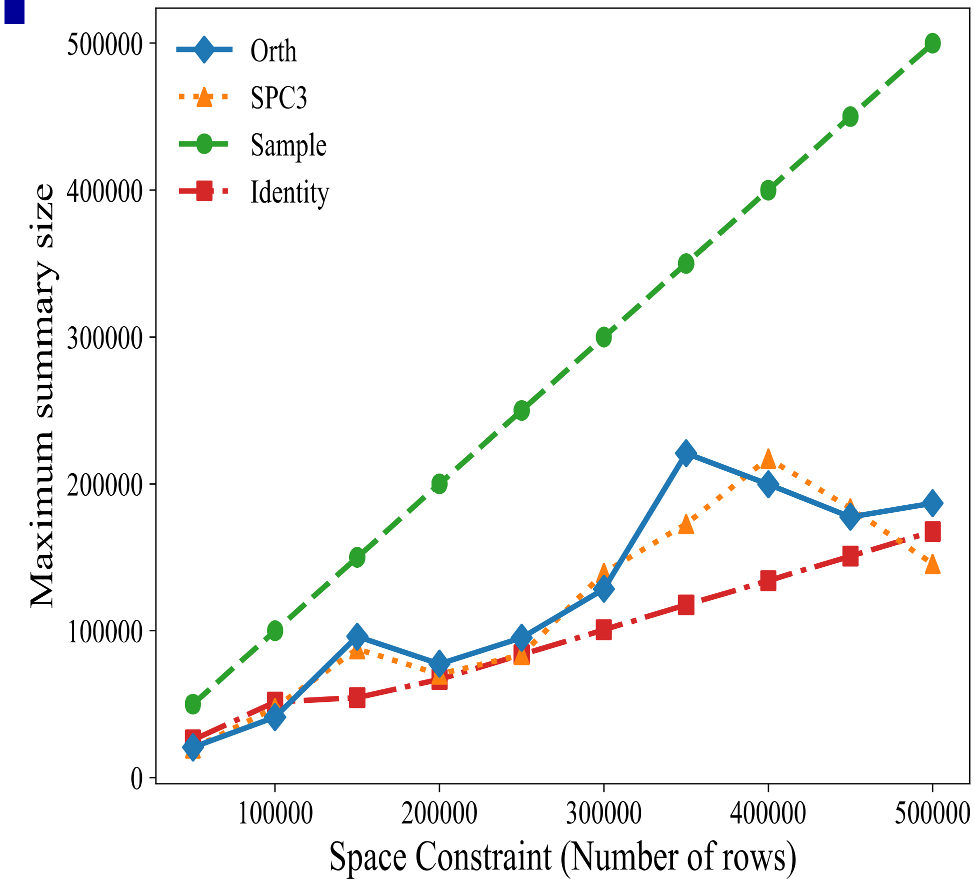
- We seek $x = \mathrm{argmin}_x \|Ax - b\|_\infty$
- Summarise $A$ to find $A'$, and restrict $b$ to these indices as $b'$
- Now find $\hat{x} = \mathrm{argmin}_x \|A'x - b'\|_\infty$ ("sketch and solve")
  - Argue correctness via well-conditioned basis
  - Obtain additive $\varepsilon\|b\|_p$ error after scaling the parameters

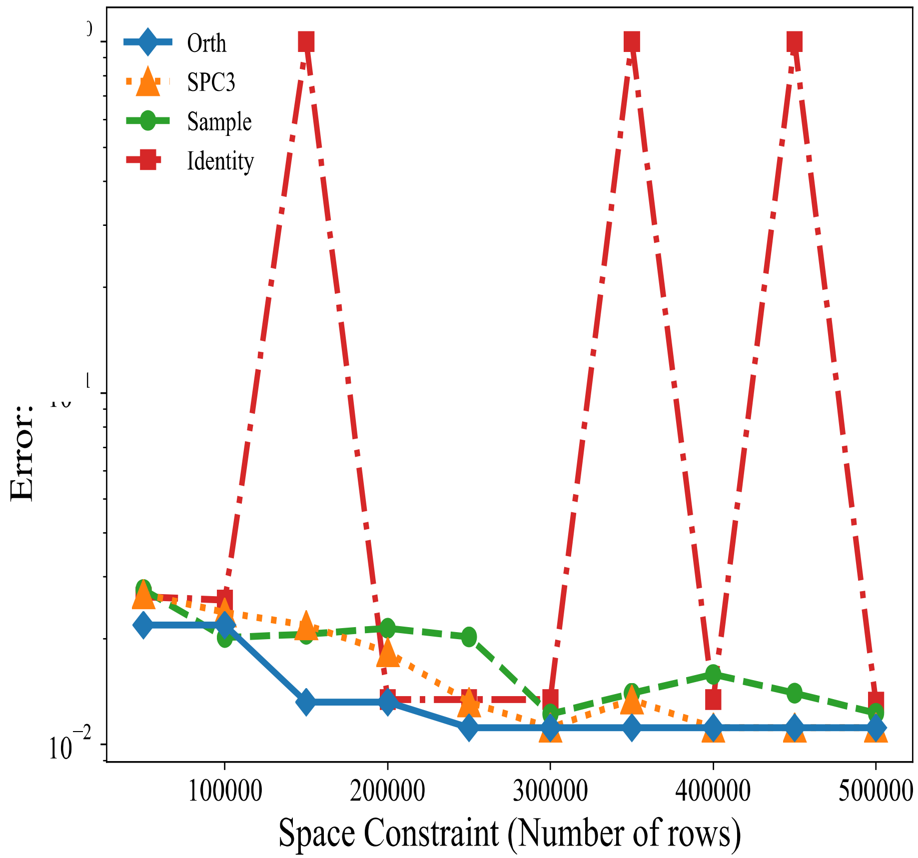
# Empirical Evaluation

| Method | WCB? | Threshold |
|---|---|---|
| Orth | $\ell_2$ | $d/m$ |
| SPC3 | $\ell_1$ | $d^{1.5}/m$ |
| Identity | No | $2/m$ |
| Uniform Sampling | No | None |

- Study two datasets: 5 million row sample of US Census Data and 50000 rows of YearPredictionMSD
- Storage parameter $b$ (number of rows sent) is varied

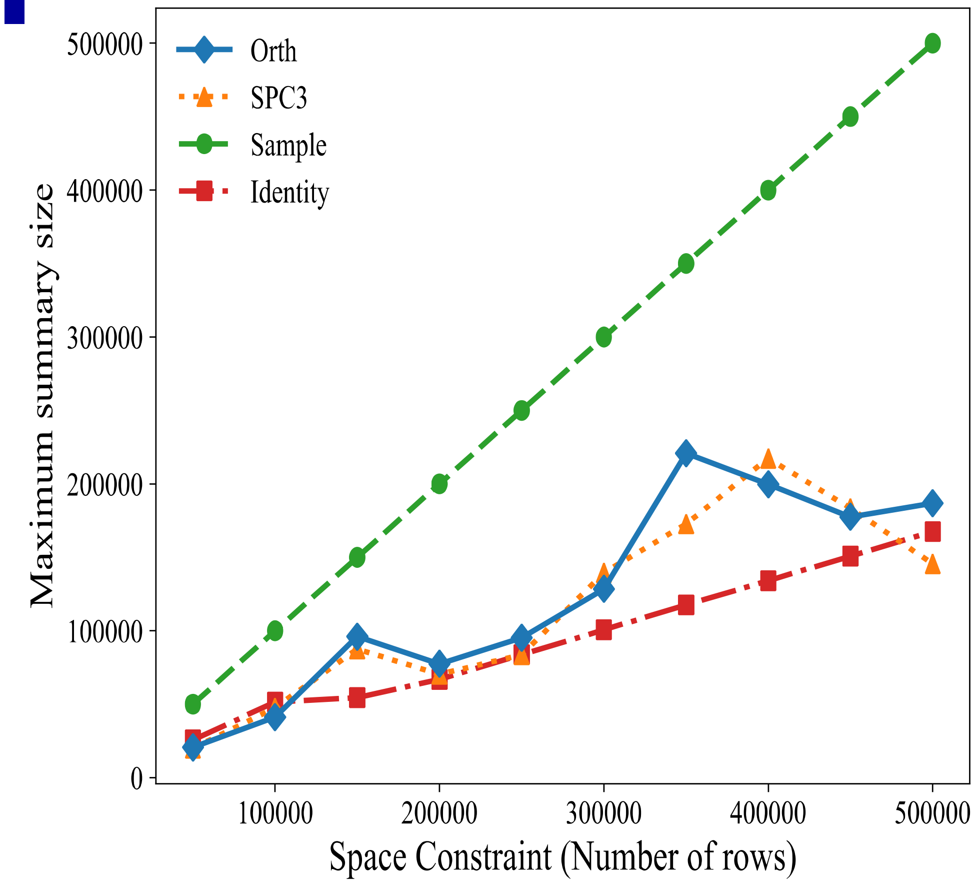
# Identity isn’t ideal



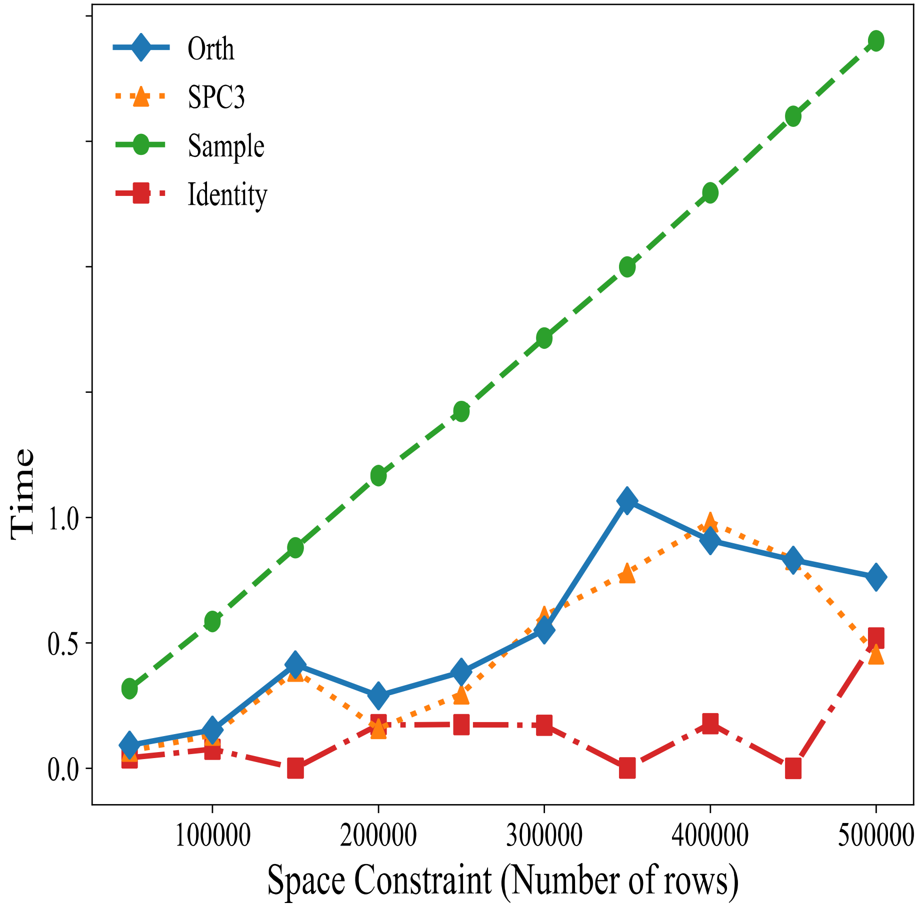




No consistent error behaviour for Identity method

# Sampling takes longer to query




Significant and growing difference in regression time



# Experimental Summary

- Constructed a summary in sublinear space
  - Census: close to 0.01 error with ~2% of the data
- The summarization step is fast, and yields a compact summary
  - Less than 1 second to summarize data of 0.5M rows
- Faster total time than to use centralized exact solver
- Conditioning is robust across different measures and datasets

# Thoughts on Distributed Data Summarization

- Data summarization leads to interesting technical questions
  - With (hopefully) interesting theory and practical implications
- Aim is often for protocols where distribution comes ‘for free’
  - i.e. Summaries have a simple algebra, can be ‘added’
  - Sometimes it’s helpful to avoid explicit synchronization
- Recent applications lean towards machine learning
  - “Everybody else is doing it, so why can’t we?”
  - ML gives challenging problems with plausible motivations

# 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 distributed protocols
  - Allow each entity to work with local data and minimize coordination
- Many open problems in this broad area
  - Machine learning/linear algebra a rich source of problems
- Continuing interest in applying and developing new theory
  - Always looking for new collaborators/students/postdocs

erc
European Research Council
Established by the European Commission

EPSRC
Engineering and Physical Sciences Research Council

YAHOO! RESEARCH

Microsoft Research

THE ROYAL SOCIETY