Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms
  3. Sophisticated data strunctures

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms
  3. Sophisticated data strunctures
  4. Optimization.

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms
  3. Sophisticated data strunctures
  4. Optimization.
- Whp, answer would have been: (2) and (3).

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms
  3. Sophisticated data strunctures
  4. Optimization.
- Whp, answer would have been: (2) and (3).
- Reality: Perhaps more (1) and (4).

Resurgence of Linear Algebra

- Circa 1990, if you asked a random Theory person, which of these methods would scale up and get applied to large problems:
  1. Singular Value Decomposition (SVD), other LA algorithms
  2. Network Flows, Shortest Paths and other graph algorithms
  3. Sophisticated data strunctures
  4. Optimization.
- Whp, answer would have been: (2) and (3).
- Reality: Perhaps more (1) and (4).
- Crucial help: from Continuous Mathematics, Developments in Randomized algorithms.

SVD in Learning

Random Selection

- Topic Modeling, Clustering.

SVD in Learning

Random Selection

- Topic Modeling, Clustering.
- Learning Mixtures of Gaussians Vempala and Wong: For a mixture of spherical Gaussians, the SVD subspace is the space of component means → Learning Algorithm.

SVD in Learning

Random Selection

- Topic Modeling, Clustering.
- Learning Mixtures of Gaussians Vempala and Wong: For a mixture of spherical Gaussians, the SVD subspace is the space of component means → Learning Algorithm.
- Non-negative Matrix Factorization (NMF)

Who tosses Coins?

1. Randomized Algorithms $\equiv$ Algorithm tosses Coins.

Who tosses Coins?

1. Randomized Algorithms $\equiv$ Algorithm tosses Coins.
   - But, data Worst-Case.

Who tosses Coins?

1. Randomized Algorithms $\equiv$ Algorithm tosses Coins.
   - But, data Worst-Case.
2. Average Case (Probabilistic) Analysis: Data tosses coins. Algorithm is deterministic.

Who tosses Coins?

1. Randomized Algorithms $\equiv$ Algorithm tosses Coins.
   - But, data Worst-Case.
2. Average Case (Probabilistic) Analysis: Data tosses coins. Algorithm is deterministic.
3. Hybrids also possible.
4. Here more (1) than (2). More useful for large matrices peculiar to a single context, like Web, FB graph, ...

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.
- Input data matrix may be distributed among servers. Randomization will help reduce communication.

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.
- Input data matrix may be distributed among servers. Randomization will help reduce communication.
- Two Scenarios, One method:

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.
- Input data matrix may be distributed among servers. Randomization will help reduce communication.
- Two Scenarios, One method:
  1. Entire matrix exists (eg. Web). Alg. draws sample.

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.
- Input data matrix may be distributed among servers. Randomization will help reduce communication.
- Two Scenarios, One method:
  1. Entire matrix exists (eg. Web). Alg. draws sample.
  2. Only sample of entries known: Netflix, Recommendation Systems. [Need handle on sampling probabilities to assert error bounds.]

Why Randomized Algorithms?

- Modern data matrices can be massive ($> 10^8$ non-zero entries).
  - $O(n^3)$ algorithms are too expensive. Time
  - RAM Cannot store matrix. So, an entry cannot be accessed in unit time. space.
- A Simple form of RA: Computes on a sample of rows/coulmns of matrix. Need:
  - Proven error guarantees and
  - Ability to Sample on the fly.
- Input data matrix may be distributed among servers. Randomization will help reduce communication.
- Two Scenarios, One method:
  1. Entire matrix exists (eg. Web). Alg. draws sample.
  2. Only sample of entries known: Netflix, Recommendation Systems. [Need handle on sampling probabilities to assert error bounds.]
  3. Here, think of (1) generally.

The Setting

- A an $m \times n$ data matrix, n, m large.

The Setting

- A an $m \times n$ data matrix, n, m large.
- Do s i.i.d. trials; in each trial:

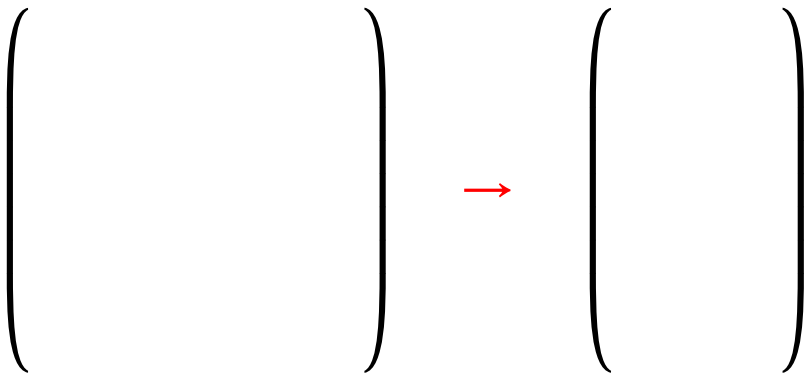
The Setting

- A an $m \times n$ data matrix, n, m large.
- Do s i.i.d. trials; in each trial:
  - Pick a random column of A and scale it.

The Setting

- A an $m \times n$ data matrix, n, m large.
- Do s i.i.d. trials; in each trial:
  - Pick a random column of A and scale it.
- Compute only with the sampled and scaled $n \times s$ matrix.

The Setting

- A an $m \times n$ data matrix, n, m large.
- Do s i.i.d. trials; in each trial:
  - Pick a random column of A and scale it.
- Compute only with the sampled and scaled $n \times s$ matrix.
- 

Problems

- Matrix Multiplication:

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification
- Linear Regression.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification
- Linear Regression.
- Tensors: Spectral Norm.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification
- Linear Regression.
- Tensors: Spectral Norm.
- Two things:

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification
- Linear Regression.
- Tensors: Spectral Norm.
- Two things:
  - No Free Lunch. Samples can only get approximate answers.

Problems

- Matrix Multiplication:
  - Compute (approximately) (all entires of) AA^T.
  - More Generally: AB.
- Singular Value Decomposition, Low Rank Approximation.
- Matrix Sketches. [Compact representations of matrix.]
- Graph Sparsification
- Linear Regression.
- Tensors: Spectral Norm.
- Two things:
  - No Free Lunch. Samples can only get approximate answers.
  - But we will prove error bounds for all input matrices.

A little Notation

- $||A||_F^2 = \sum_{i,j} A_{ij}^2$

A little Notation

- $||A||_F^2 = \sum_{i,j} A_{ij}^2$
- $||A||_2 = \mathrm{Max}_{|x|=1} |Ax|$ (Spectral norm)

A little Notation

- $||A||_F^2 = \sum_{i,j} A_{ij}^2$
- $||A||_2 = \mathrm{Max}_{|x|=1} |Ax|$ (Spectral norm)
- A is $m \times n$.

A little Notation

- $||A||_F^2 = \sum_{i,j} A_{ij}^2$
- $||A||_2 = \mathrm{Max}_{|x|=1} |Ax|$ (Spectral norm)
- A is $m \times n$.
- s number of sampled columns.

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*
- Sampling is done with probabilities proportional to squared length of columns.

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*
- Sampling is done with probabilities proportional to squared length of columns.
- Only interesting if $\varepsilon||A||_F <$ (Best Possible SVD error). Holds for PCA matrices.

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*
- Sampling is done with probabilities proportional to squared length of columns.
- Only interesting if $\varepsilon||A||_F <$ (Best Possible SVD error). Holds for PCA matrices.
  - "Length-squared sampling". Frieze, Kannan, Vempala (1998)

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*
- Sampling is done with probabilities proportional to squared length of columns.
- Only interesting if $\varepsilon||A||_F <$ (Best Possible SVD error). Holds for PCA matrices.
  - "Length-squared sampling". Frieze, Kannan, Vempala (1998)
  - Many improvements: Drineas, Mahoney, Sarlos, Deshpande, Rademacher, ...

Low Rank Approximation with Additive Error

- Seek a rank k approximation A^* to A with
  - $||A - A^*||_F \leq \underbrace{\text{Best Possible}}_{\text{SVD}} + \underbrace{\varepsilon||A||_F}_{\text{Sampling Error}}$.
- **Theorem** $s = \text{poly}(k/\varepsilon)$ suffices *provided*
- Sampling is done with probabilities proportional to squared length of columns.
- Only interesting if $\varepsilon||A||_F <$ (Best Possible SVD error). Holds for PCA matrices.
  - "Length-squared sampling". Frieze, Kannan, Vempala (1998)
  - Many improvements: Drineas, Mahoney, Sarlos, Deshpande, Rademacher, ...
- Alternative Scheme: Draw a sample of entries, set others to zero. Sparsity gain rather than reduction in dimensions. Achlioptas, McSherry; Bhojanapalli, Jain and Sanghavi.

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?
- Uniform Sampling no good. Eg. All but one column of A is all zeros.

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?
- Uniform Sampling no good. Eg. All but one column of A is all zeros.
- Unbiased Estimator: $X = \frac{1}{p_j}$ (Col. j of A) (Row j of A) (Scaling)

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?
- Uniform Sampling no good. Eg. All but one column of A is all zeros.
- Unbiased Estimator: $X = \frac{1}{p_j}$ (Col. j of A) (Row j of A) (Scaling)
- Calculus: Length squared minimizes the variance.

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?
- Uniform Sampling no good. Eg. All but one column of A is all zeros.
- Unbiased Estimator: $X = \frac{1}{p_j}$ (Col. j of A) (Row j of A) (Scaling)
- Calculus: Length squared minimizes the variance.
- With average of s samples:

  $E\left(||AA^T - \text{Estimate of } AA^T||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.

AA^T and why length squared

- $AA^T = \sum_j$(Col.j of A) (Row$j$ of A). [Outer Product]. Estimate by a sample of j.
- i.i.d. trials. p_j =probability of picking column j, $j = 1, 2, \ldots, n$. What are good p_j?
- Uniform Sampling no good. Eg. All but one column of A is all zeros.
- Unbiased Estimator: $X = \frac{1}{p_j}$ (Col. j of A) (Row j of A) (Scaling)
- Calculus: Length squared minimizes the variance.
- With average of s samples:

  $E\left(||AA^T - \text{Estimate of } AA^T||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.

  - Drineas, Kannan, Mahoney (Approx) Matrix Multiplication in $O^*(n^2)$ time.

Data Handling, Pass efficient Model

- Massive Data - too large to be stored in RAM.

Data Handling, Pass efficient Model

- Massive Data - too large to be stored in RAM.
- Simplified Model: Three resources - RAM time, space, Number of passes. Pass is a sequential read of entire matrix.

Data Handling, Pass efficient Model

- Massive Data - too large to be stored in RAM.
- Simplified Model: Three resources - RAM time, space, Number of passes. Pass is a sequential read of entire matrix.
- Sampling Algorithms use $O(1)$ passes and RAM space = $O(1)$ columns/rows of matrix. One pass computes length squared probabilities. Pass 2 draws the sample and then only RAM computation.

Data Handling, Pass efficient Model

- Massive Data - too large to be stored in RAM.
- Simplified Model: Three resources - RAM time, space, Number of passes. Pass is a sequential read of entire matrix.
- Sampling Algorithms use $O(1)$ passes and RAM space = $O(1)$ columns/rows of matrix. One pass computes length squared probabilities. Pass 2 draws the sample and then only RAM computation.
- An approximate Low rank Approximation can be carried out even in the Streaming Model - Edo Liberty (a vector version of frequent item mining).

Data Handling, Pass efficient Model

- Massive Data - too large to be stored in RAM.
- Simplified Model: Three resources - RAM time, space, Number of passes. Pass is a sequential read of entire matrix.
- Sampling Algorithms use $O(1)$ passes and RAM space = $O(1)$ columns/rows of matrix. One pass computes length squared probabilities. Pass 2 draws the sample and then only RAM computation.
- An approximate Low rank Approximation can be carried out even in the Streaming Model - Edo Liberty (a vector version of frequent item mining).
- Also, one can first do length squared sampling to pick s columns, then again do length squared sampling to pick s rows (to form a $s \times s$ matrix) and so use only $O(1)$ RAM space. Proof gets very complicated.

Sketch of a matrix

- Can a sample of rows form a sketch of the matrix? No. Sample of rows tells us nothing about the unsampled rows.

Sketch of a matrix

- Can a sample of rows form a sketch of the matrix? No. Sample of rows tells us nothing about the unsampled rows.
- How about a sample of rows and a sample of columns ? Will see the answer is Yes.

Sketch of a matrix

- Can a sample of rows form a sketch of the matrix? No. Sample of rows tells us nothing about the unsampled rows.
- How about a sample of rows and a sample of columns ? Will see the answer is Yes.
- First suppose A has rank k. Then a sample of $100k$ rows should pin down the row space of A.

Sketch of a matrix

- Can a sample of rows form a sketch of the matrix? No. Sample of rows tells us nothing about the unsampled rows.
- How about a sample of rows and a sample of columns ? Will see the answer is Yes.
- First suppose A has rank k. Then a sample of $100k$ rows should pin down the row space of A.
- But still don't know for an unsampled row what linear combination it is. Suppose we also pick a sample of $100k$ columns. Now, intuitively, (if the rows are in general position), there should be a unique linear way of expressing each row (in the $100k$-column matrix) in the row space.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.
- Given just C, R, can find a $s \times \sqrt{s}$ matrix U such that

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.
- Given just C, R, can find a $s \times \sqrt{s}$ matrix U such that
- $E\left(||A - CUR||_2^2\right) \leq \frac{c||A||_F^2}{\sqrt{s}}$.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.
- Given just C, R, can find a $s \times \sqrt{s}$ matrix U such that
- $E\left(||A - CUR||_2^2\right) \leq \frac{c||A||_F^2}{\sqrt{s}}$.
- Drineas, Kannan, Mahoney (2002) .....

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.
- Given just C, R, can find a $s \times \sqrt{s}$ matrix U such that
- $E\left(||A - CUR||_2^2\right) \leq \frac{c||A||_F^2}{\sqrt{s}}$.
- Drineas, Kannan, Mahoney (2002) .....
- Bourtsides and Woodruff (2015) Optimal time, size of U.

Length squared sample of rows and col.'s suffice

- A is $m \times n$.
- C is $m \times s$ formed by sampling (and scaling) s columns of A according to length squared.
- R is $\sqrt{s} \times n$ formed by sampling $\sqrt{s}$ rows of A according to length squared.
- Given just C, R, can find a $s \times \sqrt{s}$ matrix U such that
- $E\left(||A - CUR||_2^2\right) \leq \frac{c||A||_F^2}{\sqrt{s}}$.
- Drineas, Kannan, Mahoney (2002) .....
- Bourtsides and Woodruff (2015) Optimal time, size of U.
- $\begin{pmatrix} \\ A \\ \\ \end{pmatrix} \approx \begin{pmatrix} \\ C \\ \\ \end{pmatrix} \begin{pmatrix} \\ U \\ \\ \end{pmatrix} \begin{pmatrix} \\ R \\ \\ \end{pmatrix}$

Applications of *CUR*

- Traditional SVD, given data matrix A, finds best rank k approximation A_k to A. Two issues:

Applications of *CUR*

- Traditional SVD, given data matrix A, finds best rank k approximation A_k to A. Two issues:
  - Computation time.

Applications of *CUR*

- Traditional SVD, given data matrix A, finds best rank k approximation A_k to A. Two issues:
  - Computation time.
  - Not “interpolative”. Gene Patient Matrix. You tell the Biologist: The principal component is 17 times first patient - 12 times 31st patient + 2.5 times the 7 th ...

Applications of *CUR*

- Traditional SVD, given data matrix A, finds best rank k approximation A_k to A. Two issues:
  - Computation time.
  - Not “interpolative”. Gene Patient Matrix. You tell the Biologist: The principal component is 17 times first patient - 12 times 31st patient + 2.5 times the 7 th ...
- Interpolative approximation useful in Genetics, other areas Drineas, Mahoney.

Applications of *CUR*

- Traditional SVD, given data matrix A, finds best rank k approximation A_k to A. Two issues:
  - Computation time.
  - Not "interpolative". Gene Patient Matrix. You tell the Biologist: The principal component is 17 times first patient - 12 times 31st patient + 2.5 times the 7 th ...
- Interpolative approximation useful in Genetics, other areas Drineas, Mahoney.
- DataBase applications Falustos

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.
- A $d \times \infty$ matrix with each column a sample weighted by prob.

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.
- A $d \times \infty$ matrix with each column a sample weighted by prob.
- Varience along $\mathbf{v}$ is $\mathbf{v}^T AA^T \mathbf{v} = \left|\mathbf{v}^T A\right|^2$.

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.
- A $d \times \infty$ matrix with each column a sample weighted by prob.
- Varience along $\mathbf{v}$ is $\mathbf{v}^T AA^T \mathbf{v} = \left|\mathbf{v}^T A\right|^2$.
- Sample Complexity: How many i.i.d. samples according to P suffice to estimate varience to relative error along every direction?

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.
- A $d \times \infty$ matrix with each column a sample weighted by prob.
- Varience along $\mathbf{v}$ is $\mathbf{v}^T AA^T \mathbf{v} = \left|\mathbf{v}^T A\right|^2$.
- Sample Complexity: How many i.i.d. samples according to P suffice to estimate varience to relative error along every direction?
- Want to sample a (finite, scaled) sub-matrix B of A so that $\forall \mathbf{v} : \left|\mathbf{v}^T B\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A\right|$.

AA^T and Varience-covarience matrix

- P probability distribution (density or discrete) on $\mathbf{R}^d$. Mean $\mathbf{0}$.
- Varience-covarience matrix M: $M_{ij} = E_P(x_i x_j)$.
- A $d \times \infty$ matrix with each column a sample weighted by prob.
- Varience along $\mathbf{v}$ is $\mathbf{v}^T A A^T \mathbf{v} = \left|\mathbf{v}^T A\right|^2$.
- Sample Complexity: How many i.i.d. samples according to P suffice to estimate varience to relative error along every direction?
- Want to sample a (finite, scaled) sub-matrix B of A so that $\forall \mathbf{v} : \left|\mathbf{v}^T B\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A\right|$.
- Question in this form arose in Volume computation and log-concave sampling Kannan, Lovász, Simonovits.

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)
- Rudelson, Vershynin using beautiful technique of "Decoupling" from Probability, Functional Analysis proved:

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)
- Rudelson, Vershynin using beautiful technique of "Decoupling" from Probability, Functional Analysis proved:
- $E\left(||AA^T - \text{Estimate}||_2\right) \leq \frac{c||A||_F||A||_2}{\sqrt{s}}$.

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)
- Rudelson, Vershynin using beautiful technique of "Decoupling" from Probability, Functional Analysis proved:
- $E\left(||AA^T - \text{Estimate}||_2\right) \leq \frac{c||A||_F||A||_2}{\sqrt{s}}$.
- Much more than a technical improvement as we will see.

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)
- Rudelson, Vershynin using beautiful technique of "Decoupling" from Probability, Functional Analysis proved:
- $E\left(||AA^T - \text{Estimate}||_2\right) \leq \frac{c||A||_F||A||_2}{\sqrt{s}}$.
- Much more than a technical improvement as we will see.
- Now simpler proofs based on Matrix Höffding-Chernoff inequalities: Eg. Tropp: "User friendly tail bounds for sums of random marices"

AA^T and the Answer to the Question

- Previously: Length-squared sampling estimate for AA^T satisfies: $E\left(||AA^T - \text{Estimate}||_F\right) \leq \frac{||A||_F^2}{\sqrt{s}}$.)(Proof Elementary)
- Rudelson, Vershynin using beautiful technique of "Decoupling" from Probability, Functional Analysis proved:
- $E\left(||AA^T - \text{Estimate}||_2\right) \leq \frac{c||A||_F||A||_2}{\sqrt{s}}$.
- Much more than a technical improvement as we will see.
- Now simpler proofs based on Matrix Höffding-Chernoff inequalities: Eg. Tropp: "User friendly tail bounds for sums of random marices"
- Rudelson, Vershynin Theorem implies for log-concave probability densities on $\mathbf{R}^d$, $O^*(d)$ samples suffice (improving earlier answers of Pisier and Bourgain).

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger
- Stronger condition (than cuts):

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger
- Stronger condition (than cuts):
  - $\forall \mathbf{v}, \quad \left|\mathbf{v}^T A_G\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A_H\right|$

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger
- Stronger condition (than cuts):
  - $\forall \mathbf{v}, \quad \left|\mathbf{v}^T A_G\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A_H\right|$
  - Same question as approximating varience-covarince matrix.

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger
- Stronger condition (than cuts):
  - $\forall \mathbf{v}, \quad \left|\mathbf{v}^T A_G\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A_H\right|$
  - Same question as approximating varience-covarince matrix.
- Rudelson-Vershynin implies: $\forall \mathbf{v} : \left||\mathbf{v}^T A_G|^2 - |\mathbf{v}^T A_H|^2\right| =\leq \frac{cn||A||_2^2|\mathbf{v}|^2}{s}$.

AA^T and Spectral Sparsifiers

- Spielman and Srivatsava: G graph with n nodes and m edges; A_G = node-edge incidence matrix.
- Want sparser sub-graph H (with $O^*(n)$ edges, say,), so that all cuts are approximately right.Benzur, Karger
- Stronger condition (than cuts):
  - $\forall \mathbf{v}, \quad \left|\mathbf{v}^T A_G\right| = (1 \pm \varepsilon)\left|\mathbf{v}^T A_H\right|$
  - Same question as approximating varience-covarince matrix.
- Rudelson-Vershynin implies: $\forall \mathbf{v} : \left| |\mathbf{v}^T A_G|^2 - |\mathbf{v}^T A_H|^2 \right| =\leq \frac{cn||A||_2^2|\mathbf{v}|^2}{s}$.
- Not good enough. But if only make A_G an isometry, $||A_G||_2|\mathbf{v}|^2 = |\mathbf{v}^T A_G|^2$ and take $s \geq cn$, it all works.

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:
  - Let $p_j =$ (length squared of col j of WA) / $||WA||_F^2$.

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:
  - Let $p_j =$ (length squared of col j of WA) / $||WA||_F^2$.
  - Repeat s times: Pick col. j of A with probability p_j and scale it by $1/p_j$ to form a $n \times s$ matrix B.

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:
  - Let $p_j =$ (length squared of col j of WA) / $||WA||_F^2$.
  - Repeat s times: Pick col. j of A with probability p_j and scale it by $1/p_j$ to form a $n \times s$ matrix B.
- $\forall \mathbf{v} : \left|\mathbf{v}^T B\right|^2 = \left(1 \pm \frac{c\sqrt{n}}{\sqrt{s}}\right) \left|\mathbf{v}^T A\right|^2$. Rudelson, Vershynin

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:
  - Let $p_j =$ (length squared of col j of WA) / $||WA||_F^2$.
  - Repeat s times: Pick col. j of A with probability p_j and scale it by $1/p_j$ to form a $n \times s$ matrix B.
- $\forall \mathbf{v} : \left|\mathbf{v}^T B\right|^2 = \left(1 \pm \frac{c\sqrt{n}}{\sqrt{s}}\right) \left|\mathbf{v}^T A\right|^2$. Rudelson, Vershynin
- All of the above true for **any matrix** A.

Pre-conditioned length squared sampling

- A is $n \times m$, $m \geq n$. W is the $n \times n$ left pseudo-inverse of A: $WA =$ an isometry on the column space of A.
- Sample columns of A according to length-squared probabilities from WA:
  - Let $p_j =$ (length squared of col j of WA) / $||WA||_F^2$.
  - Repeat s times: Pick col. j of A with probability p_j and scale it by $1/p_j$ to form a $n \times s$ matrix B.
- $\forall \mathbf{v} : \left|\mathbf{v}^T B\right|^2 = \left(1 \pm \frac{c\sqrt{n}}{\sqrt{s}}\right) \left|\mathbf{v}^T A\right|^2$. Rudelson, Vershynin
- All of the above true for **any matrix** A.
- Computing p_j involves finding W. Spielman, Srivatsava For node-edge adjacency matrix of a graph, can be done in linear time.
- Open question: Are there other class of interesting matrices for which p_j can be computed fast?

Pre-conditioned length squared and leverage scores

- Rudelson-Vershynin theorem can be used to assert: $O^*(\text{rank}\,(A))$ samples suffice.

Pre-conditioned length squared and leverage scores

- Rudelson-Vershynin theorem can be used to assert: $O^*(\text{rank}\ (A))$ samples suffice.
- Can we save on rank (A)? Yes. First if we do SVD to find A_k, the best rank k approximation to A and then use pre-conditioned length squared probabilities of A_k, we can do with $O^*(k)$ samples.

Pre-conditioned length squared and leverage scores

- Rudelson-Vershynin theorem can be used to assert: $O^*(\text{rank}\ (A))$ samples suffice.
- Can we save on rank (A)? Yes. First if we do SVD to find A_k, the best rank k approximation to A and then use pre-conditioned length squared probabilities of A_k, we can do with $O^*(k)$ samples.
- Drineas, Mahoney and Muthukrishnan: **Theorem** With poly(k/ε) sample columns of A drawn according to pre-conditioned length squared probabilities on A_k, we can get an **interpolative** approximation A' to A with the following **relative error**:

Pre-conditioned length squared and leverage scores

- Rudelson-Vershynin theorem can be used to assert: $O^*(\text{rank}\ (A))$ samples suffice.
- Can we save on rank (A)? Yes. First if we do SVD to find A_k, the best rank k approximation to A and then use pre-conditioned length squared probabilities of A_k, we can do with $O^*(k)$ samples.
- Drineas, Mahoney and Muthukrishnan: **Theorem** With poly(k/ε) sample columns of A drawn according to pre-conditioned length squared probabilities on A_k, we can get an **interpolative** approximation A' to A with the following **relative error**:
- $||A - A'||_F \leq (1+\varepsilon)||A - A_k||_F$.

Pre-conditioned length squared and leverage scores

- Rudelson-Vershynin theorem can be used to assert: $O^*(\text{rank}\ (A))$ samples suffice.
- Can we save on rank (A)? Yes. First if we do SVD to find A_k, the best rank k approximation to A and then use pre-conditioned length squared probabilities of A_k, we can do with $O^*(k)$ samples.
- Drineas, Mahoney and Muthukrishnan: **Theorem** With poly(k/ε) sample columns of A drawn according to pre-conditioned length squared probabilities on A_k, we can get an **interpolative** approximation A' to A with the following **relative error**:
- $||A - A'||_F \leq (1+\varepsilon)||A - A_k||_F$.
- An alternative way to get the same theorem: Draw a sample of $r =$poly(k/ε) sample columns of A with probabilities proportional to the square of the volume of the simplex spanned by them. Deshpande, Rademacher and Vempala Volume Sampling, Determinental process

JAGUAR

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.
- Want to maximize $\sum_{i,j,k,\ldots} A_{ijk\ldots} x_i x_j x_k \ldots$ over unit length vector x.

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.
- Want to maximize $\sum_{i,j,k,\ldots} A_{ijk\ldots} x_i x_j x_k \ldots$ over unit length vector x.
- **Theorem** For any fixed $\varepsilon > 0$, can find in polynomial time a y satisfying:

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.
- Want to maximize $\sum_{i,j,k,\ldots} A_{ijk\ldots} x_i x_j x_k \ldots$ over unit length vector x.
- **Theorem** For any fixed $\varepsilon > 0$, can find in polynomial time a y satisfying:
  - $\sum_{i,j,k,\ldots} A_{ijk\ldots} y_i y_j y_k \ldots \geq$ MAX Possible $- \varepsilon ||A||_F$.

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.
- Want to maximize $\sum_{i,j,k,\ldots} A_{ijk\ldots} x_i x_j x_k \ldots$ over unit length vector x.
- **Theorem** For any fixed $\varepsilon > 0$, can find in polynomial time a y satisfying:
  - $\sum_{i,j,k,\ldots} A_{ijk\ldots} y_i y_j y_k \ldots \geq$ MAX Possible $- \varepsilon ||A||_F$.
  - Algorithm involves length squared sampling. Kannan, Vempala

Randomized Algorithm for general tensors

- A is $n_1 \times n_2 \times n_3 \ldots n_r$ symmetric.
- Want to maximize $\sum_{i,j,k,\ldots} A_{ijk\ldots} x_i x_j x_k \ldots$ over unit length vector x.
- **Theorem** For any fixed $\varepsilon > 0$, can find in polynomial time a y satisfying:
  - $\sum_{i,j,k,\ldots} A_{ijk\ldots} y_i y_j y_k \ldots \geq$ MAX Possible $- \varepsilon ||A||_F$.
  - Algorithm involves length squared sampling. Kannan, Vempala
- Better results known under assumptions on the tensor known used: (for eg. orthogonal rank 1 decomposition) - Anandkumar, Foster, Hsu, Kakade, Liu

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.
  - SA - can be computed in linear time. Then LRA just on SA suffices.

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.
  - SA - can be computed in linear time. Then LRA just on SA suffices.

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.
  - SA - can be computed in linear time. Then LRA just on SA suffices.
- Clarkson, Woodruff: Low rank approximation, Regression, Matrix Multiplication,... Optimal algorithms (time linear in number of non-zeros).

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.
  - SA - can be computed in linear time. Then LRA just on SA suffices.
- Clarkson, Woodruff: Low rank approximation, Regression, Matrix Multiplication,... Optimal algorithms (time linear in number of non-zeros).
- Clarkson, Woodruff, STOC 2009 Space optimal Streaming algorithms.

The random sign matrix

- **"Count-Sketch" Matrix**: Has a single non-zero entry in each column which is ± 1 with prob. 1/2 each, in a random row. Dasgupta, Kumar, Sarlos
- Clarkson, Woodruff, STOC(2013) Best Paper: A ANY $m \times n$ matrix. $m >> n$
  - S a $t \times m$ count-sketch matrix with t =poly(n/ε). [Indep. of m.]
  - Whp, **SIMULTANEOUSLY FOR ALL** $x \in \mathbf{R}^n$, $|SAx| = (1 \pm \varepsilon)|Ax|$.
  - SA - can be computed in linear time. Then LRA just on SA suffices.
- Clarkson, Woodruff: Low rank approximation, Regression, Matrix Multiplication,... Optimal algorithms (time linear in number of non-zeros).
- Clarkson, Woodruff, STOC 2009 Space optimal Streaming algorithms.
- Bourtsides, Woodruff, 2015 Optimal CUR.

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.
- Server i can find $PA^{(i)}$ locally and communicate this $s \times n$ matrix to a Central Processor to sum ...

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.
- Server i can find $PA^{(i)}$ locally and communicate this $s \times n$ matrix to a Central Processor to sum ...
- Communication is $O(snr)$ avoiding m, EXCEPT

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.
- Server i can find $PA^{(i)}$ locally and communicate this $s \times n$ matrix to a Central Processor to sum ...
- Communication is $O(snr)$ avoiding m, EXCEPT
  - Servers need to agree on the same P. Needs $O(smr)$ communication!
- Alon, Mataias, Szegedy Use of pseudo-random numbers.

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.
- Server i can find $PA^{(i)}$ locally and communicate this $s \times n$ matrix to a Central Processor to sum ...
- Communication is $O(snr)$ avoiding m, EXCEPT
  - Servers need to agree on the same P. Needs $O(smr)$ communication!
- Alon, Mataias, Szegedy Use of pseudo-random numbers.
- Kane, Mekha, Nelson: Let x be a fixed vector. To get $|PAx| \approx |Ax|$, $O(\log n)$-way independence suffices. Need to communicate only the $O(\log n)$ bit-seed to all servers.

Distributed Data

- Suppose r servers each has a $m \times n$ matrix with $m >> n$; server i has matrix $A^{(i)}$.
- Want to compute with $A = A^{(1)} + A^{(2)} + \cdots + A^{(r)}$. [Examples: Net-flow data, web crawl data...]
- In many contexts, it is enough to compute with a random projection of A, i.e., with PA where, P is a $s \times m$ random matrix.
- Server i can find $PA^{(i)}$ locally and communicate this $s \times n$ matrix to a Central Processor to sum ...
- Communication is $O(snr)$ avoiding m, EXCEPT
  - Servers need to agree on the same P. Needs $O(smr)$ communication!
- Alon, Mataias, Szegedy Use of pseudo-random numbers.
- Kane, Mekha, Nelson: Let x be a fixed vector. To get $|PAx| \approx |Ax|$, $O(\log n)$-way independence suffices. Need to communicate only the $O(\log n)$ bit-seed to all servers.
- To ensure $|PAx| \approx |Ax|$ for all $x \in \mathbf{R}^n$, poly(n)-way independence suffices.