

# Main Points

## Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.

## Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.

# Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.
  - ▶ Theory and algorithms for tensors are not as simple/clean as for matrices. But here: some things we can (provably) do for / with tensors (just multi-dimensional arrays). [For: Algorithms; With: applications.]

# Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.
  - ▶ Theory and algorithms for tensors are not as simple/clean as for matrices. But here: some things we can (provably) do for / with tensors (just multi-dimensional arrays). [For: Algorithms; With: applications.]
  - ▶ Many things we do not know so much about: Optimize (approximately) cubic and higher order forms over unit length vectors.

# Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.
  - ▶ Theory and algorithms for tensors are not as simple/clean as for matrices. But here: some things we can (provably) do for / with tensors (just multi-dimensional arrays). [For: Algorithms; With: applications.]
  - ▶ Many things we do not know so much about: Optimize (approximately) cubic and higher order forms over unit length vectors.
- ▶ Sampling on the fly plays a prominent role in these methods. Much mileage from a simple sampler: Pick rows/columns of a matrix with probability proportional to squared length of row/column. [Joint with [Alan Frieze and Santosh Vempala.](#)]

# Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.
  - ▶ Theory and algorithms for tensors are not as simple/clean as for matrices. But here: some things we can (provably) do for / with tensors (just multi-dimensional arrays). [For: Algorithms; With: applications.]
  - ▶ Many things we do not know so much about: Optimize (approximately) cubic and higher order forms over unit length vectors.
- ▶ Sampling on the fly plays a prominent role in these methods. Much mileage from a simple sampler: Pick rows/columns of a matrix with probability proportional to squared length of row/column. [Joint with [Alan Frieze and Santosh Vempala.](#)]
  - ▶ Subtext: Bring more of the immense talent of FOCS/STOC/SODA community to such problems.

# Main Points

- ▶ Spectral methods are useful not only for numerical, but also discrete (optimization) problems.
- ▶ Spectral methods can be extended to tensors.
  - ▶ Theory and algorithms for tensors are not as simple/clean as for matrices. But here: some things we can (provably) do for / with tensors (just multi-dimensional arrays). [For: Algorithms; With: applications.]
  - ▶ Many things we do not know so much about: Optimize (approximately) cubic and higher order forms over unit length vectors.
- ▶ Sampling on the fly plays a prominent role in these methods. Much mileage from a simple sampler: Pick rows/columns of a matrix with probability proportional to squared length of row/column. [Joint with [Alan Frieze and Santosh Vempala](#).]
  - ▶ Subtext: Bring more of the immense talent of FOCS/STOC/SODA community to such problems.
- ▶ Important topics covered elsewhere: Spectral Graph Partitioning. Algorithms (FOCS/STOC) perspective on Numerical Algorithms. [Dan Spielman's](#) tutorial.

# Blessing and Curse of Numerical Analysis

- ▶ **Blessing** Beautiful Theory. Starts with “**Greedy Works**”: For example for spectral decomposition/SVD of a matrix  $A$  (no need to recall these exactly now), can find  $x$  maximizing  $x^T Ax$  (Greedy), “peel” this  $x$  off and repeat.
- ▶ **Curse** Unless you are careful, numerical problems plague you.
- ▶ A possible Fix: Can we carry out something like greedy, but using only 0-1 vectors  $x$  ? Qualified Yes with “cut matrices” to come.
  - ▶ Yes - can do many combinatorial things with cut matrices.
  - ▶ Qualified - not as beautiful theory as SVD.

## Cut Matrices- “Combinatorial” Rank 1 matrices

**Rectangle** in an  $m \times n$  matrix is one of the  $2^{m+n}$  subsets of the form (a subset of rows)  $\times$  (subset of columns).

A **cut matrix** has all entries in a rectangle equal and all entries outside the rectangle 0.

-0.7	0	-0.7	0	-0.7
-0.7	0	-0.7	0	-0.7
0	0	0	0	0
-0.7	0	-0.7	0	-0.7
0	0	0	0	0

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

- ▶ It is very easy to show these approximations exist.

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

- ▶ It is very easy to show these approximations exist.
- ▶ They can be found (and this in non-trivial) in polynomial, in fact in constant time (implicitly) with uniform random sample of  $O(1)$  entries from the matrix.

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

- ▶ It is very easy to show these approximations exist.
- ▶ They can be found (and this in non-trivial) in polynomial, in fact in constant time (implicitly) with uniform random sample of  $O(1)$  entries from the matrix.
- ▶ Using the approximation, one can solve the maximum cut problem to additive error  $\epsilon n^2 M$  on  $n$  node graphs with maximum edge weight  $M$ . [This also extends to all MAX-2-CSP's.]

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

- ▶ It is very easy to show these approximations exist.
- ▶ They can be found (and this in non-trivial) in polynomial, in fact in constant time (implicitly) with uniform random sample of  $O(1)$  entries from the matrix.
- ▶ Using the approximation, one can solve the maximum cut problem to additive error  $\epsilon n^2 M$  on  $n$  node graphs with maximum edge weight  $M$ . [This also extends to all MAX-2-CSP's.]
- ▶ Both the existence as well as the algorithmic version (in constant time) can be extended to tensors and using this, one can approximately solve MAX- $r$ -SAT for fixed  $r$  and other problems.

# Approximation by Cut Matrices

Every matrix can be approximated by a sum of cut matrices

- ▶ It is very easy to show these approximations exist.
- ▶ They can be found (and this in non-trivial) in polynomial, in fact in constant time (implicitly) with uniform random sample of  $O(1)$  entries from the matrix.
- ▶ Using the approximation, one can solve the maximum cut problem to additive error  $\epsilon n^2 M$  on  $n$  node graphs with maximum edge weight  $M$ . [This also extends to all MAX-2-CSP's.]
- ▶ Both the existence as well as the algorithmic version (in constant time) can be extended to tensors and using this, one can approximately solve MAX- $r$ -SAT for fixed  $r$  and other problems.

Frieze, Kannan

Main Caveat: The  $\epsilon n^2 M$  error.

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .
- ▶ Otherwise, there is some rectangle with large (abs. value) sum.

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .
- ▶ Otherwise, there is some rectangle with large (abs. value) sum.
- ▶ Take rectangle with largest (abs. value) sum. **GREEDY. How to find this?**

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .
- ▶ Otherwise, there is some rectangle with large (abs. value) sum.
- ▶ Take rectangle with largest (abs. value) sum. **GREEDY. How to find this?**
- ▶ Subtract from each entry of this rectangle, the average of the rectangle. **Peel it off.**

## Approximation by sum of cut matrices - Existence

Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .
- ▶ Otherwise, there is some rectangle with large (abs. value) sum.
- ▶ Take rectangle with largest (abs. value) sum. **GREEDY. How to find this?**
- ▶ Subtract from each entry of this rectangle, the average of the rectangle. **Peel it off.**
  - ▶ This subtracts from each  $A_{ij}, i \in S, j \in T$ , their average. Subtracting the average from a set of reals decreases their sum of squares. So  $\|A\|_F^2$  goes down by a definite amount.

## Approximation by sum of cut matrices - Existence

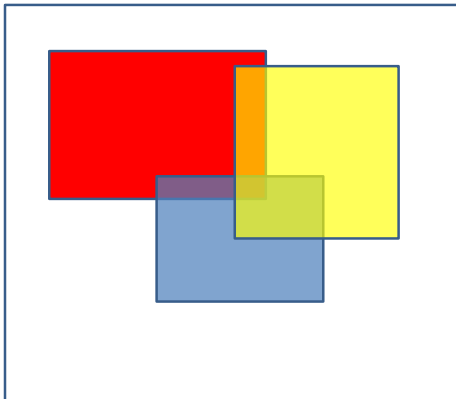
Define the “cut norm” of  $\|A\|_{\square}$  as the maximum absolute value of the sum of any rectangle.  $\|A\|_{\square} = \text{Max}_{S,T} \left| \sum_{i \in S, j \in T} A_{ij} \right|$ .

Assume  $|A_{ij}| \leq 1$ . There are  $1/\epsilon^2$  cut matrices whose sum  $B$  approximates  $A$  in the sense

$$\|A - B\|_{\square} \leq \epsilon mn.$$

### Proof

- ▶ If  $\|A\|_{\square} \leq \epsilon mn$ , we can take  $B = 0$ .
- ▶ Otherwise, there is some rectangle with large (abs. value) sum.
- ▶ Take rectangle with largest (abs. value) sum. **GREEDY. How to find this?**
- ▶ Subtract from each entry of this rectangle, the average of the rectangle. **Peel it off.**
  - ▶ This subtracts from each  $A_{ij}, i \in S, j \in T$ , their average. Subtracting the average from a set of reals decreases their sum of squares. So  $\|A\|_F^2$  goes down by a definite amount.
  - ▶ Repeat. Finite termination since  $\|A\|_F^2$  is non-negative.



## An Application - Graph Regularity Lemma

**Density** between two subsets  $S, T$  of vertices of a graph =  
(number of  $S, T$  edges) /  $|S||T|$ .

$V_1, V_2, \dots, V_k$  a partition of the vertex set of graph with  $d_{ij}$  =  
density of  $V_i, V_j$ . Partition is  $\epsilon$  regular if for “most” pairs  $V_i, V_j$ ,  
ALL sub-parts of  $V_i, V_j$  have density close to  $d_{ij}$ . Original

**Szemerédi** Regularity Lemma : There is an  $\epsilon$  regular partition of  
ANY GRAPH into TOWER OF HEIGHT  $1/\epsilon^{20}$  parts. (**Gowers**) :  
Tower is necessary.

**Lemma** simply follows from Cut Approximation: For any graph,  
there is a partition into  $k = 2^{O(1/\epsilon^2)}$  sets so that for EVERY  
 $S, T \subseteq V$ , number of  $S, T$  edges is within  $\epsilon n^2$  of what is  
“predicted by the partition densities”, namely

$$\sum_{i,j} d_{ij} |V_i \cap S| |V_j \cap T|.$$

Can partition into a “small” number of sets so that the part a  
vertex is in tells its story.

– Called the Weak Regularity Lemma.

# Regularity Pictures

Takes too long to draw pictures. DYO

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

- ▶  $X$  a finite domain. [For Cut Approx.,  $X = \{(i, j)\}$ .]

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

- ▶  $X$  a finite domain. [For Cut Approx.,  $X = \{(i, j)\}$ .]
- ▶  $\mu$  a probability distribution over  $X$ . [ $\mu(i, j) = \frac{1}{mn}$ .]

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

- ▶  $X$  a finite domain. [For Cut Approx.,  $X = \{(i, j)\}$ .]
- ▶  $\mu$  a probability distribution over  $X$ . [ $\mu(i, j) = \frac{1}{mn}$ .]
- ▶  $A : X \rightarrow [-1, 1]$ . [Matrix]

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

- ▶  $X$  a finite domain. [For Cut Approx.,  $X = \{(i, j)\}$ .]
- ▶  $\mu$  a probability distribution over  $X$ . [ $\mu(i, j) = \frac{1}{mn}$ .]
- ▶  $A : X \rightarrow [-1, 1]$ . [Matrix]
- ▶  $\mathcal{F}$  family of bounded functions  $f : X \rightarrow [-1, 1]$  [rectangles]

# A generalization of Cut approximation

Trevisan, Tulsiani, Vadhan

Thought: Main argument was: Subtracting average from a bunch of reals reduces second moment. “Random” flavor.

- ▶  $X$  a finite domain. [For Cut Approx.,  $X = \{(i, j)\}$ .]
- ▶  $\mu$  a probability distribution over  $X$ . [ $\mu(i, j) = \frac{1}{mn}$ .]
- ▶  $A : X \rightarrow [-1, 1]$ . [Matrix]
- ▶  $\mathcal{F}$  family of bounded functions  $f : X \rightarrow [-1, 1]$  [rectangles]
- ▶ There exist  $f_1, f_2, \dots, f_{1/\epsilon^2} \in \mathcal{F}$  and  $c_1, c_2, \dots, c_{1/\epsilon^2}$  with  $\sum c_i^2 \leq 1$  such that

$$\forall f \in \mathcal{F} : \left| E_{\mu}(fA) - E_{\mu}\left(f \sum c_i f_i\right) \right| \leq \epsilon.$$

Their theorem is more general.

# Approximation by Cut Matrices - Algorithm

**Theorem**  $|A_{ij}| \leq 1$ .  $\epsilon > 0$ . Can find in polynomial time a matrix  $B$  which is the sum of  $4/\epsilon^2$  cut matrices so that

$$\|A - B\|_{\square} \leq \epsilon mn.$$

In fact, given the entries of  $A$  in just a u.a.r. rectangle of size  $O^*(1/\epsilon^4) \times O^*(1/\epsilon^4)$ , we can find an implicit description of  $B$ .

**Observe (Greedy)**: From existence proof, it suffices to solve :

$$\text{Max}_{S,T} A(S, T), \text{ where, } A(S, T) = \sum_{i \in S, j \in T} A_{ij}.$$

# Solving the Maximum Rectangle Problem: $\text{MaxA}(S, T)$

# Solving the Maximum Rectangle Problem: $\text{Max}A(S, T)$

- ▶  $S$  **gives**  $T$  For given  $S$ , best  $T$  is just the columns of  $A$  whose sum in the  $S$  rows is positive.

# Solving the Maximum Rectangle Problem: $\text{Max}A(S, T)$

- ▶  **$S$  gives  $T$**  For given  $S$ , best  $T$  is just the columns of  $A$  whose sum in the  $S$  rows is positive.
- ▶ **Estimate Column sums in  $S$  rows** Pick a subset  $W$  of  $s = O(1/\epsilon^2)$  rows u.a.r.  
For each column: sum in the  $S$  rows is positive  $\approx$  sum in  $S \cap W$  rows positive. (except when sum in  $S$  rows is close to 0; then the column doesn't matter much - detail trickier).

# Solving the Maximum Rectangle Problem: $\text{Max}A(S, T)$

- ▶  **$S$  gives  $T$**  For given  $S$ , best  $T$  is just the columns of  $A$  whose sum in the  $S$  rows is positive.
- ▶ **Estimate Column sums in  $S$  rows** Pick a subset  $W$  of  $s = O(1/\epsilon^2)$  rows u.a.r.  
For each column: sum in the  $S$  rows is positive  $\approx$  sum in  $S \cap W$  rows positive. (except when sum in  $S$  rows is close to 0; then the column doesn't matter much - detail trickier).
- ▶ **Exhaustive Enumeration**<sup>1</sup> Don't know  $S$  or  $S \cap W$ . But, try each subset  $\tilde{W}$  of  $W$  as a candidate  $S \cap W$ . For each, find the set of columns-  $T$  - whose sum in the  $\tilde{W}$  rows is positive.

# Solving the Maximum Rectangle Problem: $\text{Max}A(S, T)$

- ▶  **$S$  gives  $T$**  For given  $S$ , best  $T$  is just the columns of  $A$  whose sum in the  $S$  rows is positive.
- ▶ **Estimate Column sums in  $S$  rows** Pick a subset  $W$  of  $s = O(1/\epsilon^2)$  rows u.a.r.  
For each column: sum in the  $S$  rows is positive  $\approx$  sum in  $S \cap W$  rows positive. (except when sum in  $S$  rows is close to 0; then the column doesn't matter much - detail trickier).
- ▶ **Exhaustive Enumeration**<sup>1</sup> Don't know  $S$  or  $S \cap W$ . But, try each subset  $\tilde{W}$  of  $W$  as a candidate  $S \cap W$ . For each, find the set of columns-  $T$  - whose sum in the  $\tilde{W}$  rows is positive.
- ▶ **Choose best candidate**(By turning previous argument on its head For each candidate  $T$ : Let  $S'$  be the rows with positive sum in the  $T$  columns. Take  $\max A(S', T)$  among all candidate  $T$ .

# Solving the Maximum Rectangle Problem: $\text{Max}A(S, T)$

- ▶  **$S$  gives  $T$**  For given  $S$ , best  $T$  is just the columns of  $A$  whose sum in the  $S$  rows is positive.
- ▶ **Estimate Column sums in  $S$  rows** Pick a subset  $W$  of  $s = O(1/\epsilon^2)$  rows u.a.r.  
For each column: sum in the  $S$  rows is positive  $\approx$  sum in  $S \cap W$  rows positive. (except when sum in  $S$  rows is close to 0; then the column doesn't matter much - detail trickier).
- ▶ **Exhaustive Enumeration**<sup>1</sup> Don't know  $S$  or  $S \cap W$ . But, try each subset  $\tilde{W}$  of  $W$  as a candidate  $S \cap W$ . For each, find the set of columns-  $T$  - whose sum in the  $\tilde{W}$  rows is positive.
- ▶ **Choose best candidate**(By turning previous argument on its head For each candidate  $T$ : Let  $S'$  be the rows with positive sum in the  $T$  columns. Take  $\max A(S', T)$  among all candidate  $T$ .

<sup>1</sup> Inspired by an idea of Arora, Karger, Karpinski

## Max rectangle problem

1	0.8	-0.7	-0.2	0.1
-1	0.3	-0.4	-0.6	0.7
-0.5	-0.3	0.2	0.1	0.6
0.7	-0.8	0.9	-0.5	0.2
-1	0.1	0	0.4	-0.3

**Legend** *S* Green. *T* Red. *W* yellow.

# Max rectangle problem

1      0.8      -0.7      -0.2      0.1

-1      0.3      -0.4      -0.6      0.7

-0.5      -0.3      0.2      0.1      0.6

0.7      -0.8      0.9      -0.5      0.2

-1      0.1      0      0.4      -0.3

**Legend** *S* Green. *T* Red. *W* yellow.

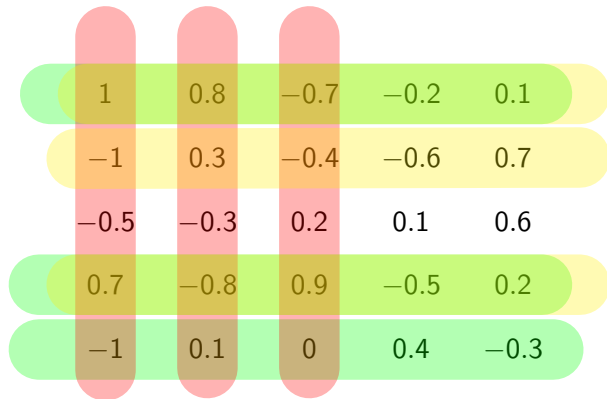
# Max rectangle problem

A 5x5 grid of numbers. Three vertical red bars highlight the first, second, and third columns. Three horizontal green bars highlight the first, fourth, and fifth rows. The intersection of these bars highlights the elements 1, 0.8, -0.7, 0.7, -0.8, 0.9, -1, 0.1, and 0.

1	0.8	-0.7	-0.2	0.1
-1	0.3	-0.4	-0.6	0.7
-0.5	-0.3	0.2	0.1	0.6
0.7	-0.8	0.9	-0.5	0.2
-1	0.1	0	0.4	-0.3

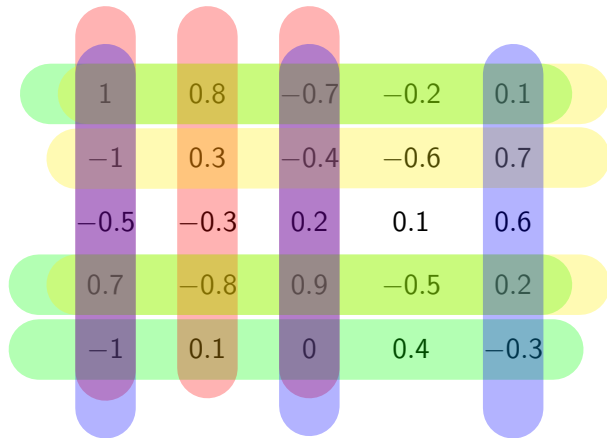
**Legend** *S* Green. *T* Red. *W* yellow.

# Max rectangle problem



**Legend** *S* Green. *T* Red. *W* yellow.

# Max rectangle problem



**Legend** *S* Green. *T* Red. *W* yellow.

# Making it Constant time

Algorithm in a nutshell

# Making it Constant time

## Algorithm in a nutshell

1. Constant number of sets -  $\tilde{W}$ , each of size  $O(1)$ .

# Making it Constant time

## Algorithm in a nutshell

1. Constant number of sets -  $\tilde{W}$ , each of size  $O(1)$ .
2. For each  $\tilde{W}$ , let  $\mathcal{T}$  be the set of columns with positive sum in  $\tilde{W}$  rows.

# Making it Constant time

## Algorithm in a nutshell

1. Constant number of sets -  $\tilde{W}$ , each of size  $O(1)$ .
2. For each  $\tilde{W}$ , let  $T$  be the set of columns with positive sum in  $\tilde{W}$  rows.
3. For each such  $T$ , let  $S'$  be the rows with positive sum in the  $T$  columns.

# Making it Constant time

## Algorithm in a nutshell

1. Constant number of sets -  $\tilde{W}$ , each of size  $O(1)$ .
2. For each  $\tilde{W}$ , let  $T$  be the set of columns with positive sum in  $\tilde{W}$  rows.
3. For each such  $T$ , let  $S'$  be the rows with positive sum in the  $T$  columns.
4. Find largest  $A(S', T)$  among these.

# Making it constant time-II

Only red stuff added

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

- ▶ For cut approximation, this is repeated  $O(1)$  times.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

- ▶ For cut approximation, this is repeated  $O(1)$  times.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

- ▶ For cut approximation, this is repeated  $O(1)$  times.
- ▶ Keep track of all  $O(1)$  matrices on the way.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

- ▶ For cut approximation, this is repeated  $O(1)$  times.
- ▶ Keep track of all  $O(1)$  matrices on the way.

# Making it constant time-II

Only red stuff added

1. Pick at outset  $O(1)$  sample of rows and columns.
2. For each  $\tilde{W}$ , let  $T$  be set of **sample** columns with pos sum in  $\tilde{W}$  rows.
3. For each  $T$ , let  $S'$  be set of **sample** rows with pos sum in  $T$  columns.
4. Find largest  $A(S', T)$

Since number of  $\tilde{W}$  is  $2^{O(1/\epsilon^2)}$ , use union bound to bound failure probability of any one of the  $A(S', T)$  being estimated wrong.  
(Details...)

- ▶ For cut approximation, this is repeated  $O(1)$  times.
- ▶ Keep track of all  $O(1)$  matrices on the way.
- ▶ To find a particular entry- say  $(i, j)$  of the final cut approximation, need only check the sign of sum of row  $i$  / column  $j$  in  $O(1)$  many  $O(1)$  size subsets of columns / rows.

## Optimization via Cut approximation

First Maximum cut problem-  $A =$  the  $n \times n$  adjacency matrix of the graph)

$$\text{Max}_{x \in \{0,1\}^n} x^T A(1 - x).$$

$x^T A(1 - x)$  is just  $A(S, T)$  for a rectangle  $S \times T$  !!

Can replace  $A$  by its cut approximation  $B$ , since:

$$\left| x^T A(1 - x) - x^T B(1 - x) \right| \leq \|A - B\|_{\square}.$$

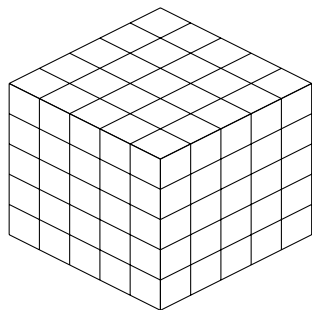
So solve instead

$$\max x^T B(1 - x).$$

Cut norm natural norm for ensuring for 0-1 vectors  $x, y$ ,  $\|A - B\|_{\square}$  small implies  $x^T A y \approx x^T B y$ .

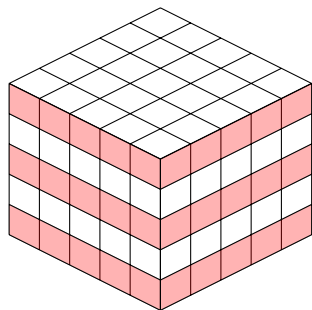
In  $x^T B(1 - x)$ , there are only  $O(1)$  “degrees of freedom”: components of  $x$  along the space spanned rows/columns of  $B$ .  
 $n$  variable problem  $\implies$  to constant number of variables.

## Algorithm/Application extends to tensors



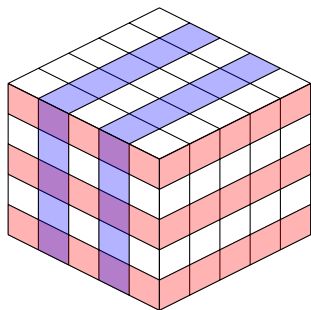
- ▶ Given  $(S, T)$ , define  $U = \{k : A(S, T, k) > 0\}$
- ▶ Pick u.a.r.  $W \subseteq \text{Rows}$  and  $X \subseteq \text{Columns}$ .
- ▶ Estimate  $\text{sign}(A(S, T, k))$  by  $\text{sign}(A(S \cap W, T \cap X, k))$ .
- ▶ **New Twist:** Gives us many candidate  $U$ 's. For each, solve a 2-dimensional array problem to estimate value. Take best.

## Algorithm/Application extends to tensors



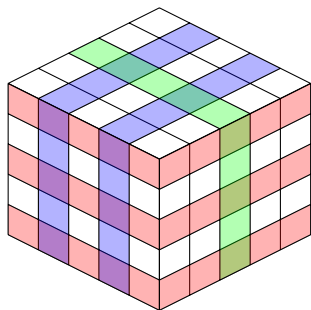
- ▶ Given  $(S, T)$ , define  $U = \{k : A(S, T, k) > 0\}$
- ▶ Pick u.a.r.  $W \subseteq \text{Rows}$  and  $X \subseteq \text{Columns}$ .
- ▶ Estimate  $\text{sign}(A(S, T, k))$  by  $\text{sign}(A(S \cap W, T \cap X, k))$ .
- ▶ **New Twist:** Gives us many candidate  $U$ 's. For each, solve a 2-dimensional array problem to estimate value. Take best.

## Algorithm/Application extends to tensors



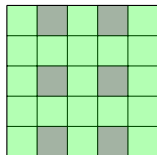
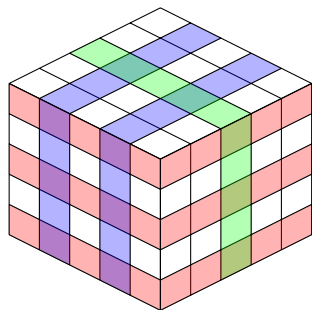
- ▶ Given  $(S, T)$ , define  $U = \{k : A(S, T, k) > 0\}$
- ▶ Pick u.a.r.  $W \subseteq \text{Rows}$  and  $X \subseteq \text{Columns}$ .
- ▶ Estimate  $\text{sign}(A(S, T, k))$  by  $\text{sign}(A(S \cap W, T \cap X, k))$ .
- ▶ **New Twist:** Gives us many candidate  $U$ 's. For each, solve a 2-dimensional array problem to estimate value. Take best.

## Algorithm/Application extends to tensors



- ▶ Given  $(S, T)$ , define  $U = \{k : A(S, T, k) > 0\}$
- ▶ Pick u.a.r.  $W \subseteq \text{Rows}$  and  $X \subseteq \text{Columns}$ .
- ▶ Estimate  $\text{sign}(A(S, T, k))$  by  $\text{sign}(A(S \cap W, T \cap X, k))$ .
- ▶ **New Twist:** Gives us many candidate  $U$ 's. For each, solve a 2-dimensional array problem to estimate value. Take best.

## Algorithm/Application extends to tensors



- ▶ Given  $(S, T)$ , define  $U = \{k : A(S, T, k) > 0\}$
- ▶ Pick u.a.r.  $W \subseteq \text{Rows}$  and  $X \subseteq \text{Columns}$ .
- ▶ Estimate  $\text{sign}(A(S, T, k))$  by  $\text{sign}(A(S \cap W, T \cap X, k))$ .
- ▶ **New Twist:** Gives us many candidate  $U$ 's. For each, solve a 2-dimensional array problem to estimate value. Take best.

# Solving Max-r-CSP's approximately

Alon, de la Vega, Karpinski and Kannan

Boolean MAX-r-CSP formula  $F(x_1, x_2, \dots, x_n)$ ,  $r$  fixed.

$Q$  u.a.r subset of variables with  $|Q| = q \in O^*(1/\epsilon^4)$  variables.

$F^Q$  "induced" MAX-r-CSP problem  $F^Q$  on the picked variables.

$$\text{Whp } \left| \frac{n^r}{q^r} \text{Max}(F^Q) - \text{Max}(F) \right| \leq \epsilon n^r.$$

Can find not only (approx.)  $\text{Max}(F)$ , but also a satisfying assignment in "O(1) time". (What could this mean ?)

Andersson and Engebretsen independently proved the above, but with  $q \in O^*(1/\epsilon^7)$  (but their result is stronger in other respects). They use methods of Property Testing. Originally, property testing (Goldwasser, Goldreich, Ron) tackled MAX-2-CSP. It uses purely combinatorial arguments, rather than any matrices and tensors.

## Random Induced sub-problems

Recap: Induced problem on a u.a.r subset of variables estimates optimal value of Max-r-CSP.

Result based on two other similar sounding result:

**easier** of the two: Similar result holds for Linear Programming:

$$Ax \leq b; 0 \leq x_i \leq 1.$$

Pick u.a.r. a subset  $Q$  of  $q$  variables. Let  $A^Q, x^Q$  be only the  $Q$  columns.

**Very easy part:** If big LP is feasible, then

$$A^Q x^Q \leq b + \Delta; 0 \leq x_i \leq 1$$

is feasible.

**Harder** (uses duality) : If big LP is infeasible, then

$$A^Q x^Q \leq b - \Delta; 0 \leq x_i \leq 1$$

is infeasible (whp).

## Random Induced sub-problems - II

**hard:** A  $n \times n \times n \dots \times n$  symmetric  $r$ -tensor, with entries in  $[-1, 1]$ .

## Random Induced sub-problems - II

**hard:**  $A$   $n \times n \times n \dots \times n$  symmetric  $r$ -tensor, with entries in  $[-1, 1]$ .

- ▶  $Q$  u.a.r.  $q$ -subset of  $[n]$  with  $q \in \Omega^*(1/\epsilon^4)$  and  $A^Q$  induced sub-matrix on  $Q$ .

## Random Induced sub-problems - II

**hard:**  $A$   $n \times n \times n \dots \times n$  symmetric  $r$ -tensor, with entries in  $[-1, 1]$ .

- ▶  $Q$  u.a.r.  $q$ -subset of  $[n]$  with  $q \in \Omega^*(1/\epsilon^4)$  and  $A^Q$  induced sub-matrix on  $Q$ .

$$\|A^Q\|_{\square} \leq \frac{q^r}{n^r} \|A\|_{\square} + \text{error} .$$

**Much easier:** If there is a rectangle  $R$  with “large” (abs. value) sum in whole matrix, there is one in sample too whp : Just the “image” of  $R$  will do.

Converse : If “most” small sub-matrices have “large” sum rectangles, the whole matrix does too. A Vapnik-Chervonenkis dimension type “double sampling” argument used.

For matrices, [Rudelson and Vershynin](#) improved this to  $1/\epsilon^2$ , but using some machinery (from Functional Analysis).

General Open Question: Can we assert something strong when we draw a sample of  $n/100$  ? The arguments so far need  $\Omega(1/\epsilon^2)$  to achieve error  $\epsilon$ , so certainly useless beyond  $\sqrt{n}$ ....

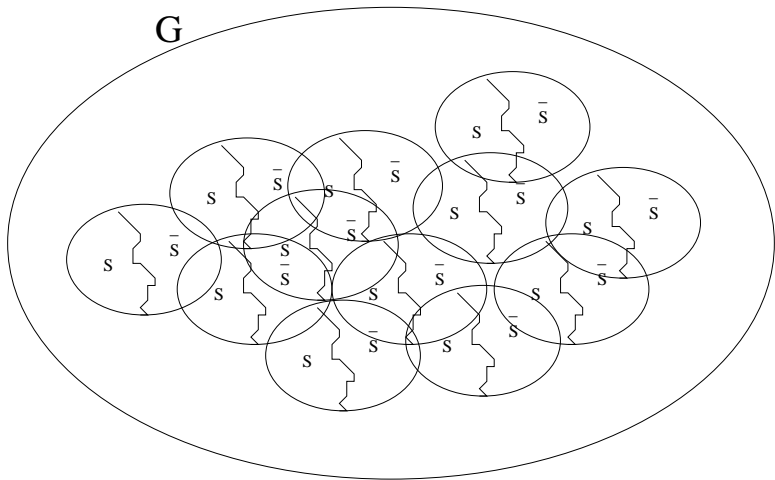


Figure: Max-Cut: Humpty-Dumpty

# Graph Limits

Borgs, Chayes, Lovász, Sós, Vestergombi

How does one define the limit of a sequence of graphs, the  $n$ th one on  $n$  vertices ?? [What happens when the web graph gets larger and larger?]

Need a notion of distance between graphs with different numbers of vertices. One intuition: Graph Regularity Lemma - partitioned the vertex set so that **the part the vertex is in told its story**.

Intuition: If we pick at random sufficiently many vertices, the sample will tell us the partition and so the whole story.

But, a new twist: should allow isomorphisms (RELABELING vertices)..

These thoughts + random induced sub-matrix having not too big a cut-norm compared to whole matrix + .....

**Theorem** Let  $G$  be a simple graph and  $\epsilon, \delta > 0$ . Then the induced sub-graph of  $G$  on a random subset of  $2^{1/\delta\epsilon^2}$  nodes is “ $\epsilon$  close” to  $G$  with probability at least  $1 - \delta$ . So a family of graphs has a limit iff random induced sub-graphs are same (close).

**Musing** Can we do sequences of matrices, tensors ? Applications ?

# Approximating the cut norm in matrices to a constant factor

Alon, Naor

Exact Problem is NP-hard.

$$\text{Reformulate as } \text{Max}_{x_i, y_j = \pm 1} \sum_{ij} x_i^T A_{ij} y_j.$$

$$\text{SDP Relaxation } \text{Max}_{|u_i|, |v_j|=1} \sum_{ij} A_{ij} (u_i \cdot v_j).$$

- Grothendik Constant: Max factor between SDP relaxation and cut norm.
- Need to find approximate  $x, y$  from vectors  $u, v$ .

Methods don't carry over to tensors. SDP relaxation ?? Other approaches for 3-tensors ?

# Non-uniform Sampling

## Non-uniform Sampling

## Non-uniform Sampling

So far - uniform sampling did everything :

- ▶ Found approximations by cut matrices.
- ▶ Solved all Max-r-CSP's to additive error  $\pm \epsilon n^r$  - all dense problems.
- ▶ Cut matrices - combinatorial rank 1 matrices. Ideally suited for uniform sampling.
- ▶ **Sampling on the Fly**: Uniform sampling can be done on Streaming matrices – can toss coins before data arrives and then just draw sample.

## Error bound for Low rank Approximations

A rank 1 matrix/tensor is an outer product of vectors.

$u \otimes v$  is a matrix with  $(i, j)$  th entry equal to  $u_i v_j$ .

$u \otimes v \otimes w$  is a 3-diml array with  $(i, j, k)$  th entry  $u_i v_j w_k$ .

For this talk, a matrix/tensor is of rank at most  $k$  if it is the sum of  $k$  rank 1 matrices/tensors.

**Lemma** (Simple: Same proof as cut approximation) For every matrix/tensor  $A$ , there is a rank  $1/\epsilon^2$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F,$$

where  $\|A - B\|_2 = \text{Max} \sum_{ijk\dots} (A - B)_{ijk\dots} x_i y_j z_k \dots$  over unit length vectors  $x, y, z, \dots$  (Operator Norm) and  $\|A\|_F$  is sq. root of sum of squares of entries.

Can't change the 2 or  $F$ . But, this approximation error suffices for PCA, MAX-CSP's ....

## Principal Component Analysis for STOCers

Usually:  $m \times n$  matrix  $A$  of data (eg. customer-product matrix),  $m, n$  large. Posit that there are a small number  $k$  of “basic factors” (income, age, consumerism, gullibility) that influence how much each customer buys of each product. Factors are unknown, but the top  $k$  singular vectors would correspond to the top  $k$  factors. But why does  $\|A - B\|_2 \leq \epsilon \|A - B\|_F$  do us any good? Another example: Document- term matrix  $A$  (rows are documents); a document is a vector telling us how many times each term occurs in it. For a new document  $v$ , its similarity to each document in our collection is the dot product; so  $Av$  is the vector of similarities. Now

$$\|A - B\|_2 \leq \Delta \implies \text{for every } v, |Av - Bv| \leq \Delta |v|.$$

Here: Similar error bounds for tensors.

But also, we saw that for DISCRETE OPTIMIZATION problems like MAX-CUT, if weight matrix  $A$  is replaced by  $B$ , where CUT NORM (an OPERATOR NORM) of  $A - B$  is small, then for EVERY CUT, its  $A$  weight  $\approx$  its  $B$  weight.

# Low-rank approximation to tensors

de la Vega, Karpinski, Kannan, Vempala

For any  $A, \epsilon > 0$ , we can find a tensor  $B$  of rank at most  $4/\epsilon^2$  in time  $(n/\epsilon)^{O(1/\epsilon^4)}$  such that with probability at least  $3/4$  we have

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

Heart of the (**greedy Algorithm**) Find  $x, y, z$  maximizing to within  $\epsilon \|A\|_F$ :

$$A(x, y, z) = \sum_{ijk} A_{ijk} x_i y_j z_k, \text{ over unit vectors } .$$

Alg. will be described later, after we see sampling in a simpler situation. In general, not much known about maximizing cubic and higher order forms. Quadratic forms doable by Linear Algebra amidst much clean theory. No such luck likely for cubic.

# What can we do with low rank approx to tensor?

We can find rank  $4/\epsilon^2$  approx  $B$  to tensor  $A$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

Saw already cut approximations solve dense Max-r-CSP's. These more general approximations are of use in PCA. In addition, the more general approximations help solve more general MAX-r-CSP's including those satisfying triangle inequality.

Common generalization of dense graphs and metrics:

Weight of each (most suffices) edge  $\leq \frac{\epsilon}{n} \times$  (total edge weight at its ends + average weight incident to a vertex)

**Musings** Are large “naturally occurring graphs” computationally easy? Without assuming a generative model? Do they fit this criterion, or, some other similar one??

# Matrix Multiplication

## Sampling in a simple example

How do we use sampling to speed up (approximate) matrix multiplication ?

# Matrix Multiplication

## Sampling in a simple example

How do we use sampling to speed up (approximate) matrix multiplication ?

$$AB = \sum_j (j \text{ th column of } A) \otimes (j \text{ th row of } B)$$

# Matrix Multiplication

## Sampling in a simple example

How do we use sampling to speed up (approximate) matrix multiplication ?

$$AB = \sum_j (j \text{ th column of } A) \otimes (j \text{ th row of } B)$$

Draw a (small) sample of  $s$   $j$ 's and estimate the sum over all  $j$ 's by the sum over the sample.

# Matrix Multiplication

## Sampling in a simple example

How do we use sampling to speed up (approximate) matrix multiplication ?

$$AB = \sum_j (j \text{ th column of } A) \otimes (j \text{ th row of } B)$$

Draw a (small) sample of  $s$   $j$ 's and estimate the sum over all  $j$ 's by the sum over the sample.

With correct scaling: estimate = sum over the  $s$  sampled  $j$ 's of

$$\frac{1}{s} \frac{1}{\text{prob. of picking } j} (j \text{ th column of } A) \otimes (j \text{ th row of } B).$$

Expectation of estimate =  $AB$  (entry-wise)

# Matrix Multiplication

## Sampling in a simple example

How do we use sampling to speed up (approximate) matrix multiplication ?

$$AB = \sum_j (j \text{ th column of } A) \otimes (j \text{ th row of } B)$$

Draw a (small) sample of  $s$   $j$ 's and estimate the sum over all  $j$ 's by the sum over the sample.

With correct scaling: estimate = sum over the  $s$  sampled  $j$ 's of

$$\frac{1}{s} \frac{1}{\text{prob. of picking } j} (j \text{ th column of } A) \otimes (j \text{ th row of } B).$$

Expectation of estimate =  $AB$  (entry-wise)

Variance = ????. Variance of each entry is different.

## Matrix Multiplication by sampling

$$AB \approx \begin{pmatrix} \text{Sam} \\ \text{pled} \\ \text{col.s} \\ \text{of } A \end{pmatrix} \cdot \begin{pmatrix} \text{Same rows of } B \text{ scaled} \end{pmatrix}$$

## Controlling variance in matrix multiplication

Consider the important case when  $B = A^T$ . Want  $AA^T = \text{sum over } j \text{ of } (j \text{ th column of } A) \otimes (j \text{ th row of } A)$ .  
What sampling probabilities minimize the **sum of the variances of all entries** ?

## Controlling variance in matrix multiplication

Consider the important case when  $B = A^T$ . Want  $AA^T = \text{sum over } j \text{ of } (j \text{ th column of } A) \otimes (j \text{ th row of } A)$ .  
What sampling probabilities minimize the **sum of the variances of all entries** ?

Answer: Probabilities proportional to the squared length of columns of  $A$ .

## Controlling variance in matrix multiplication

Consider the important case when  $B = A^T$ . Want  $AA^T = \text{sum over } j \text{ of } (j \text{ th column of } A) \otimes (j \text{ th row of } A)$ .  
What sampling probabilities minimize the **sum of the variances of all entries** ?

Answer: Probabilities proportional to the squared length of columns of  $A$ .

**Length squared sampling** introduced by Frieze, Kannan, Vempala. Used as here for matrix multiplication by Drineas, K. Many other uses. Some here.

**Sampling on the fly** (i) Can we sample in a streaming set-up ?

(ii) 2 passes clearly suffice - one to compute row lengths and one to sample.

(iii) Massive Data problems : Number of passes a resource. Sometimes more appropriate than Streaming.

A picture is worth a 1000 words

## A picture is worth a 1000 words

$$A = \begin{pmatrix} 1 & 0.8 & -0.7 & -0.2 & 0.1 \\ -1 & 0.3 & -0.4 & -0.6 & 0.7 \\ -0.5 & -0.3 & 0.2 & 0.1 & 0.6 \\ 0.7 & -0.8 & 0.9 & -0.5 & 0.2 \\ -1 & 0.1 & 0 & 0.4 & -0.3 \end{pmatrix}$$

Squared length of first col  $\approx 3.75$ ; all columns  $\approx 8.44$ .

$$AA^T \approx \frac{8.44}{3.75} \begin{pmatrix} 1 \\ -1 \\ -0.5 \\ 0.7 \\ -1 \end{pmatrix} (1 \quad -1 \quad -0.5 \quad 0.7 \quad -1).$$

Proof:  $\frac{8.44}{3.75} \approx (AA^T)_{11} = 2.18$ .

## A picture is worth a 1000 words

$$A = \begin{pmatrix} 1 & 0.8 & -0.7 & -0.2 & 0.1 \\ -1 & 0.3 & -0.4 & -0.6 & 0.7 \\ -0.5 & -0.3 & 0.2 & 0.1 & 0.6 \\ 0.7 & -0.8 & 0.9 & -0.5 & 0.2 \\ -1 & 0.1 & 0 & 0.4 & -0.3 \end{pmatrix}$$

Squared length of first col  $\approx 3.75$ ; all columns  $\approx 8.44$ .

$$AA^T \approx \frac{8.44}{3.75} \begin{pmatrix} 1 \\ -1 \\ -0.5 \\ 0.7 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -0.5 & 0.7 & -1 \end{pmatrix}.$$

Proof:  $\frac{8.44}{3.75} \approx (AA^T)_{11} = 2.18$ .

Are 1000 pictures worth a word?

# Low rank approx of matrices via sampling

Frieze, Kannan and Vempala

- ▶ Pick  $s$  columns of the given  $m \times n$  matrix  $A$  according by length-squared sampling. Scale them and form a  $m \times s$  matrix  $C$ .

# Low rank approx of matrices via sampling

Frieze, Kannan and Vempala

- ▶ Pick  $s$  columns of the given  $m \times n$  matrix  $A$  according by length-squared sampling. Scale them and form a  $m \times s$  matrix  $C$ .
- ▶ Find the top  $k$  left singular vectors -  $u_1, u_2, \dots, u_k$  of  $C$ .

# Low rank approx of matrices via sampling

Frieze, Kannan and Vempala

- ▶ Pick  $s$  columns of the given  $m \times n$  matrix  $A$  according by length-squared sampling. Scale them and form a  $m \times s$  matrix  $C$ .
- ▶ Find the top  $k$  left singular vectors -  $u_1, u_2, \dots, u_k$  of  $C$ .
- ▶ Output  $\tilde{A} = \sum_{i=1}^k u_i u_i^T A$  as the rank  $k$  approximation to  $A$ .

# Low rank approx of matrices via sampling

Frieze, Kannan and Vempala

- ▶ Pick  $s$  columns of the given  $m \times n$  matrix  $A$  according by length-squared sampling. Scale them and form a  $m \times s$  matrix  $C$ .
- ▶ Find the top  $k$  left singular vectors -  $u_1, u_2, \dots, u_k$  of  $C$ .
- ▶ Output  $\tilde{A} = \sum_{i=1}^k u_i u_i^T A$  as the rank  $k$  approximation to  $A$ .

## Theorem

$$E \left( \|A - \tilde{A}\|_F^2 \right) \leq \|A - A_k\|_F^2 + 2\sqrt{\frac{k}{s}} \|A\|_F^2$$

$$E \left( \|A - \tilde{A}\|_2^2 \right) \leq \|A - A_k\|_2^2 + \frac{2}{\sqrt{s}} \|A\|_F^2.$$

Bar-Yossef dependence of  $s$  on  $k$  optimal.

# Improvement

Rudelson, Vershynin

$r = \|A\|_F^2 / \|A\|_2^2$ , the “numerical rank” of  $A$ . [Eg. In PCA,  $r \in O(1)$ .]

## Theorem

- ▶ Sample  $s \in \Omega^*(r/\epsilon^4\delta)$  rows of  $A$  according to length-squared, scale to form  $s \times n$  matrix  $B$ .
- ▶  $u^{(1)}, \dots, u^{(k)}$  be the top  $k$  right singular vectors of  $B$ .
- ▶ For  $\tilde{A} = A \sum_{t=1}^k u^{(t)} u^{(t)T}$ , whp:

$$\|A - \tilde{A}\|_2 \leq \sigma_{k+1}(A) + \epsilon \|A\|_2.$$

- ▶ Proof uses sophisticated Probability in Banach spaces...
- ▶ Any generalization to tensors ??

Another result: [Vershynin](#) If one uses length-squared sampling in the traditional Kaczmarz iteration, get a provable rate of convergence.

# Low-Rank approx of matrices via sampling

Achlioptas and McSherry

- ▶ Now for each entry  $A_{ij}$ , there is a (low) probability  $p_{ij}$ . With prob  $p_{ij}$ , we “pick” to include  $A_{ij}$ , but we include it as  $A_{ij}/p_{ij}$ .
- ▶ With Prob  $1 - p_{ij}$ , we “zero out” entry  $A_{ij}$ .
- ▶ Resulting matrix  $B$  is sparse (many zero entries).
- ▶ But,  $A - B$  is a random matrix with independent mean-zero entries. So Random Matrix Theory implies that  $\|A - B\|_2$  is small, as required.

## Back to Low rank approximations to tensors

Central Problem: Find  $w, x, y, z$  unit vectors to maximize  $\sum_{ijkl} A_{ijkl} w_i x_j y_k z_l$ .

## Back to Low rank approximations to tensors

Central Problem: Find  $w, x, y, z$  unit vectors to maximize

$$\sum_{ijkl} A_{ijkl} w_i x_j y_k z_l.$$

1. If we knew the optimizing  $x, y, z$ , then the optimizing  $w$  is easy to find: it is just the vector  $A(\cdot, x, y, z)$  (whose  $i$  th component is  $A(e_i, x, y, z)$ ) scaled to length 1.

## Back to Low rank approximations to tensors

Central Problem: Find  $w, x, y, z$  unit vectors to maximize

$$\sum_{ijkl} A_{ijkl} w_i x_j y_k z_l.$$

1. If we knew the optimizing  $x, y, z$ , then the optimizing  $w$  is easy to find: it is just the vector  $A(\cdot, x, y, z)$  (whose  $i$ th component is  $A(e_i, x, y, z)$ ) scaled to length 1.
2. Now,  $A(e_i, x, y, z) = \sum_{j,k,l} A_{i,j,k,l} x_j y_k z_l$ . The sum can be estimated by having just a few terms. But, an important question is: how do we make sure the variance is not too high, since the entries can have disparate values ?

## Back to Low rank approximations to tensors

Central Problem: Find  $w, x, y, z$  unit vectors to maximize

$$\sum_{ijkl} A_{ijkl} w_i x_j y_k z_l.$$

1. If we knew the optimizing  $x, y, z$ , then the optimizing  $w$  is easy to find: it is just the vector  $A(\cdot, x, y, z)$  (whose  $i$  th component is  $A(e_i, x, y, z)$ ) scaled to length 1.
2. Now,  $A(e_i, x, y, z) = \sum_{j,k,l} A_{i,j,k,l} x_j y_k z_l$ . The sum can be estimated by having just a few terms. But, an important question is: how do we make sure the variance is not too high, since the entries can have disparate values ?
3. Length squared sampling works ! [Stated here without proof.]

## Back to Low rank approximations to tensors

Central Problem: Find  $w, x, y, z$  unit vectors to maximize

$$\sum_{ijkl} A_{ijkl} w_i x_j y_k z_l.$$

1. If we knew the optimizing  $x, y, z$ , then the optimizing  $w$  is easy to find: it is just the vector  $A(\cdot, x, y, z)$  (whose  $i$ th component is  $A(e_i, x, y, z)$ ) scaled to length 1.
2. Now,  $A(e_i, x, y, z) = \sum_{j,k,l} A_{i,j,k,l} x_j y_k z_l$ . The sum can be estimated by having just a few terms. But, an important question is: how do we make sure the variance is not too high, since the entries can have disparate values ?
3. Length squared sampling works ! [Stated here without proof.]
4. This gives us many candidate  $w$ 's. How do we check which one is good ? For each  $w$ , recursively solve a  $r - 1$  tensor maximization problem to determine its value !

# Low Rank Approximation to Tensors - Summary

- ▶ For ANY  $r$ -tensor  $A$  ( $r \in O(1)$ ), we can find in time exponential only in  $\epsilon$  (and polynomial in  $n$ ) a rank  $O(1/\epsilon^2)$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

# Low Rank Approximation to Tensors - Summary

- ▶ For ANY  $r$ -tensor  $A$  ( $r \in O(1)$ ), we can find in time exponential only in  $\epsilon$  (and polynomial in  $n$ ) a rank  $O(1/\epsilon^2)$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

- ▶ With this approximation, we can do PCA on tensors.

# Low Rank Approximation to Tensors - Summary

- ▶ For ANY  $r$ -tensor  $A$  ( $r \in O(1)$ ), we can find in time exponential only in  $\epsilon$  (and polynomial in  $n$ ) a rank  $O(1/\epsilon^2)$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

- ▶ With this approximation, we can do PCA on tensors.
- ▶ Also, we can solve generalizations of metrics and dense MAX- $r$ -CS problems.

# Low Rank Approximation to Tensors - Summary

- ▶ For ANY  $r$ -tensor  $A$  ( $r \in O(1)$ ), we can find in time exponential only in  $\epsilon$  (and polynomial in  $n$ ) a rank  $O(1/\epsilon^2)$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

- ▶ With this approximation, we can do PCA on tensors.
- ▶ Also, we can solve generalizations of metrics and dense MAX- $r$ -CS problems.
- ▶ There are many heuristics (meaning algorithms with no proven grantee for all tensors) in the literature.

# Low Rank Approximation to Tensors - Summary

- ▶ For ANY  $r$ -tensor  $A$  ( $r \in O(1)$ ), we can find in time exponential only in  $\epsilon$  (and polynomial in  $n$ ) a rank  $O(1/\epsilon^2)$  approximation  $B$  with

$$\|A - B\|_2 \leq \epsilon \|A\|_F.$$

- ▶ With this approximation, we can do PCA on tensors.
- ▶ Also, we can solve generalizations of metrics and dense MAX- $r$ -CS problems.
- ▶ There are many heuristics (meaning algorithms with no proven guarantee for all tensors) in the literature.
- ▶ **Challenge** Devise efficient algorithms with better provable guarantees. Prove hardness results. (Some known - [Hastad](#)).

# Interpolative Low-rank Approximations - CUR

Drineas, Kannan

Singular Vectors are linear combinations of rows/columns of matrix. May not be sparse even if matrix is. **Interpolative Approximation** is an approximation using a few actual rows/columns of matrix, not linear combinations of them.

Main Result: Sampling rows/columns of matrix as per length-squared gives us an interpolative approximation of ANY matrix with provable error guarantees.

$$\begin{pmatrix} A \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \cdot \begin{pmatrix} U \end{pmatrix} \cdot \begin{pmatrix} R \end{pmatrix}$$

## Why *CUR*?

- ▶ Sparsity of  $A$  preserved.

## Why *CUR*?

- ▶ Sparsity of  $A$  preserved.
- ▶ Interpolative: If you do Principal Component Analysis on “patient-gene” matrix, you would report back to the Biologist that these 50 linear combinations of patients/genes (negative weights and all) is important in understanding the trials !!!  
Now you can say these few individual patients/genes are important. [Paschou, Mahoney, Drineas](#) et al have a paper in Genome Research.

## Why CUR?

- ▶ Sparsity of  $A$  preserved.
- ▶ Interpolative: If you do Principal Component Analysis on “patient-gene” matrix, you would report back to the Biologist that these 50 linear combinations of patients/genes (negative weights and all) is important in understanding the trials !!!  
Now you can say these few individual patients/genes are important. [Paschou](#), [Mahoney](#), [Drineas](#) et al have a paper in Genome Research.
- ▶ Turned on its head : Recommendation System : we are only given a sample of customers (all they bought) and a sample of products (what every customer bought of these products) and want to infer the whole matrix. [Drineas](#), [Kereneidis](#), [Raghavan](#)

## Why *CUR*?

- ▶ Sparsity of  $A$  preserved.
- ▶ Interpolative: If you do Principal Component Analysis on “patient-gene” matrix, you would report back to the Biologist that these 50 linear combinations of patients/genes (negative weights and all) is important in understanding the trials !!!  
Now you can say these few individual patients/genes are important. [Paschou, Mahoney, Drineas](#) et al have a paper in Genome Research.
- ▶ Turned on its head : Recommendation System : we are only given a sample of customers (all they bought) and a sample of products (what every customer bought of these products) and want to infer the whole matrix. [Drineas, Kereneidis, Raghavan](#)
- ▶ Compact Matrix Representation in Databases. (Important) modification of *CUR* : [Sun, Xie, Zhang, Falustos](#) SIAM Datamining conference

## Why *CUR*?

- ▶ Sparsity of  $A$  preserved.
- ▶ Interpolative: If you do Principal Component Analysis on “patient-gene” matrix, you would report back to the Biologist that these 50 linear combinations of patients/genes (negative weights and all) is important in understanding the trials !!! Now you can say these few individual patients/genes are important. [Paschou, Mahoney, Drineas](#) et al have a paper in Genome Research.
- ▶ Turned on its head : Recommendation System : we are only given a sample of customers (all they bought) and a sample of products (what every customer bought of these products) and want to infer the whole matrix. [Drineas, Kereneidis, Raghavan](#)
- ▶ Compact Matrix Representation in Databases. (Important) modification of *CUR* : [Sun, Xie, Zhang, Falustos](#) SIAM Datamining conference
- ▶ **Musings** Should we sometimes include the data collection cost in measuring efficiency ? Say we have a large LP  $Ax \leq b$ , where, each  $A_{ij}$  costs us  $1/\epsilon^2$  to get within error  $\epsilon$ .

## Trouble with Length-Squared sampling

Say  $A$  consists of all but one row equal and last row orthogonal to these. Then length-squared sampling does not produce a good rank 2 approximation. Better Criterion:

**Relative Error Approximation** Want a rank  $k$  approximation  $B$  to  $A$  such that if  $A_k$  is best rank  $k$  approximation, then

$$\|A - B\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$

- ▶ **Har-Peled** linear time algorithm making  $O(\log n)$  passes over input matrix.
- ▶ **Deshpande and Vempala** :  $O(k)$  passes using **volume sampling** and **Deshpande, Rademacher, Vempala, Wang**
- ▶ **Drineas, Mahoney and Muthukrishnan** : different algorithm, (sampling probabilities based on SVD)
- ▶ **Sarlös** : linear time 2-pass algorithm - **isotropic random projection**.

# Volume Sampling

Deshpande, Vempala

Sample  $k$ -tuples of rows with probabilities proportional to volume of the simplex spanned by them. Fixes the example above. Raw volume sampling only gives a factor  $k$  approximation, but can be made to have relative error  $\epsilon$ . Btw, this also proves : any matrix  $A$  has a set of  $k$  rows whose span contains an approximation  $B$  to  $A$  with

$$\|A - B\|_F \leq (k + 1)\|A - A_k\|_F.$$

Best Possible. Many results in Numerical Analysis starting with the work of [Eisenstat and Gu](#) on the **existence** of such approximations. More recently: Deshpande, Rademacher have improved the time needed to draw a sample (according to volume sampling) to linear using some fancy sampling procedure.

# Hidden Clique Problem

- ▶  $G(n, \frac{1}{2})$  random graph, edges are independent, each with probability  $1/2$ .
- ▶ Maximum Clique Size is known to be of size  $2 \log n$ . No poly time algorithm known to find any clique of size more than  $\log n$ . [Can find maximum clique in time  $n^{O(\log n)}$ .]
- ▶ Suppose we hide a clique of size  $s$  in  $G(n, 1/2)$ . Can we find the hidden clique in poly time ?
- ▶ If  $s \in \Omega(\sqrt{n})$ , eigenvalues/vectors of the adjacency matrix give us hidden clique. [Alon, Krivalevich, Sudakov](#)
- ▶ Construct a 3-tensor  $A$  as follows:  $A_{ijk} = \pm 1$  (resp) if the number of edges in the triangle  $(i, j, k)$  is odd/even.
- ▶ Maximizing the cubic form  $\sum_{ijk} A_{ijk} x_i x_j x_k$  over unit vectors  $x$  gives us hidden clique provided  $s \in \Omega^*(n^{1/3})$ . [Frieze, Kannan](#)
- ▶ Maximizing  $r$  th order form finds clique if  $s \in \Omega(n^{1/r})$ . [Brubaker, Vempala](#)