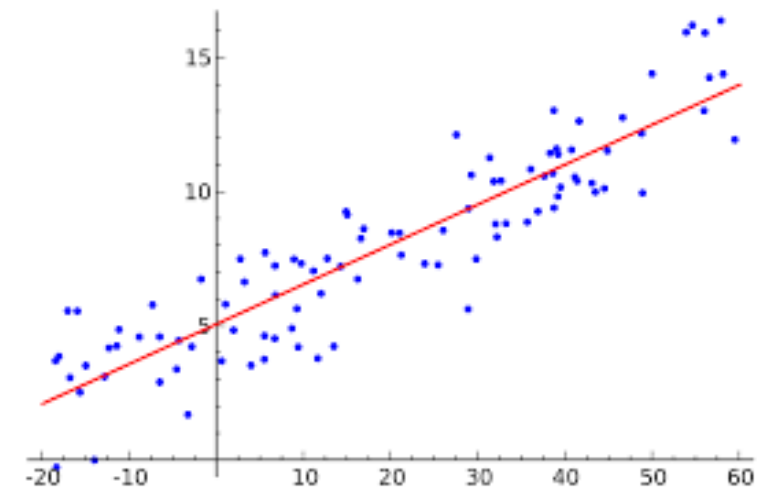
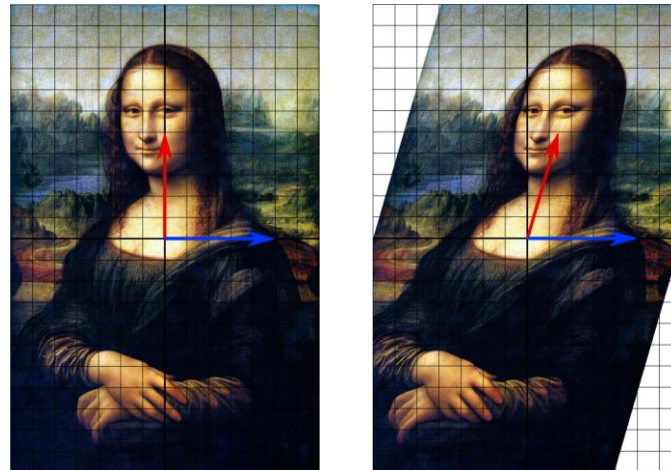
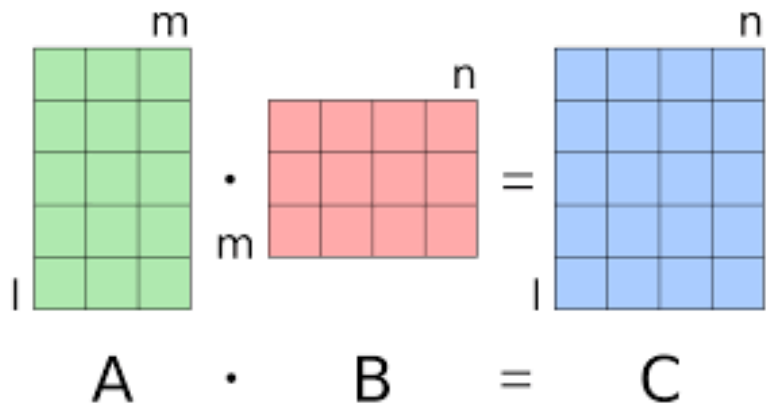


1 0 1 1 1 0 0 1 1 0 1

# Near Optimal Linear Algebra in the Online and Sliding Window Models





Vladimir Braverman



Petros Drineas



Cameron Musco





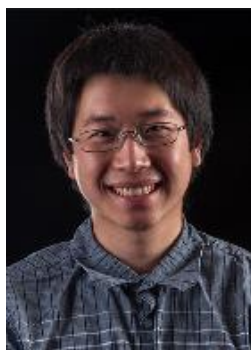
Christopher Musco



Jalaj Upadhyay



David P. Woodruff



Samson Zhou



# Model #1: Streaming / Sliding Window Model

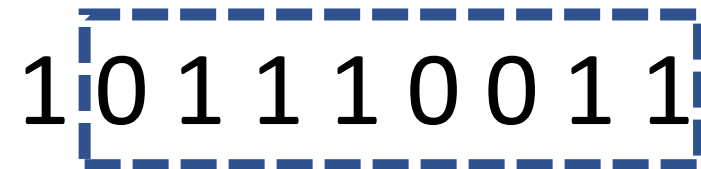
- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function
- ❖ **Goal:** Use space *sublinear* in the size of the input  $S$
  
- ❖ **Sliding Window:** “Only the  $n$  most recent updates form the underlying data set  $S$ ”

1 0 1 1 1 0 0 1

# Model #1: Streaming / Sliding Window Model

- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function
- ❖ **Goal:** Use space *sublinear* in the size of the input  $S$
  
- ❖ **Sliding Window:** “Only the  $n$  most recent updates form the underlying data set  $S$ ”

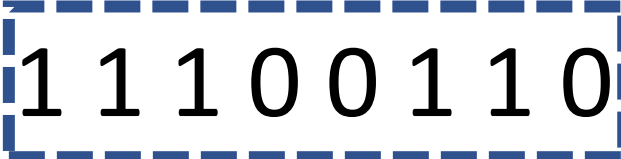
1 0 1 1 1 0 0 1 1



# Model #1: Streaming / Sliding Window Model

- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function
- ❖ **Goal:** Use space *sublinear* in the size of the input  $S$
  
- ❖ **Sliding Window:** “Only the  $n$  most recent updates form the underlying data set  $S$ ”


1 0 1 1 1 0 0 1 1 0



# Model #1: Streaming / Sliding Window Model

- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function
- ❖ **Goal:** Use space *sublinear* in the size of the input  $S$
  
- ❖ **Sliding Window:** “Only the  $n$  most recent updates form the underlying data set  $S$ ”
  - ❖ Emphasizes recent interactions, appropriate for time sensitive settings

1 0 1 1 1 0 0 1 1 0 1



# Randomized Numerical Linear Algebra (randNLA) on Sliding Windows

$n$

1	3	5	-2	7	0	11	4	-8
0	0	-1	3	13	2	8	6	2
2	5	6	1	4	0	-7	5	3
8	7	2	1	-1	-3	-2	-4	-6
-5	3	-4	-1	-2	-1	0	-3	-1
7	1	3	2	4	1	0	11	1

❖ Rows arrive one-by-one in the data stream

$d$



# Randomized Numerical Linear Algebra (randNLA) on Sliding Windows

1 3 5 -2 7 0 11 4 -8

$n$

0	0	-1	3	13	2	8	6	2
2	5	6	1	4	0	-7	5	3
8	7	2	1	-1	-3	-2	-4	-6
-5	3	-4	-1	-2	-1	0	-3	-1
7	1	3	2	4	1	0	11	1
1	2	5	-5	4	1	23	4	-3

$d$

❖ Rows arrive one-by-one in the data stream

# Randomized Numerical Linear Algebra (randNLA) on Sliding Windows

1 3 5 -2 7 0 11 4 -8

0 0 -1 3 13 2 8 6 2

2 5 6 1 4 0 -7 5 3

8 7 2 1 -1 -3 -2 -4 -6

-5 3 -4 -1 -2 -1 0 -3 -1

7 1 3 2 4 1 0 11 1

1 2 5 -5 4 1 23 4 -3

0 5 0 0 7 0 1 31 6

❖ Rows arrive one-by-one in the data stream

$n$

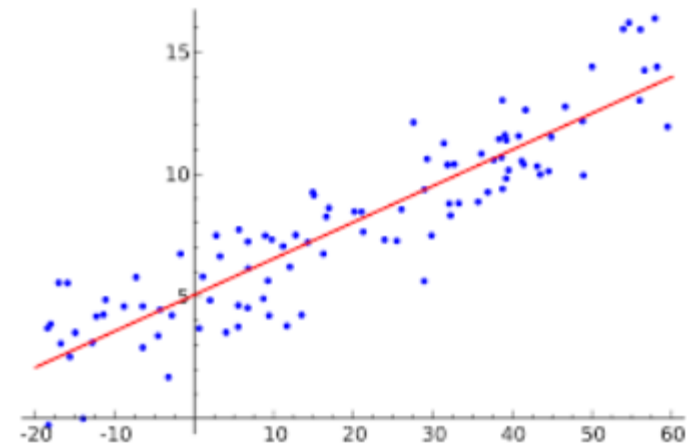
$d$

# Why randNLA on Sliding Windows?

- ❖ Sliding window emphasizes *efficiency* and *recency*, good for massive data sources and time-sensitive information
- ❖ We use linear algebra in *optimization* strategies and *prediction* of future patterns based on past data, don't want outdated information
- ❖ Principal Component Analysis (PCA), Low-Rank Approximation (LRA), Regression

		-1		
			1	
1	1	-1	1	-1
1				-1
		-1		

1	1	-1	1	-1
1	1	-1	1	-1
1	1	-1	1	-1
1	1	-1	1	-1
1	1	-1	1	-1



# Results: Sliding Window Model

- ❖  $O(d^2)$  space randomized sliding window algorithm for spectral sparsification (space optimal, up to lower order terms)
- ❖  $O(dk)$  space randomized sliding window algorithm for low-rank approximation (space optimal, up to lower order terms)
- ❖  $O(d^3)$  space randomized sliding window algorithm for  $\ell_1$  subspace embedding

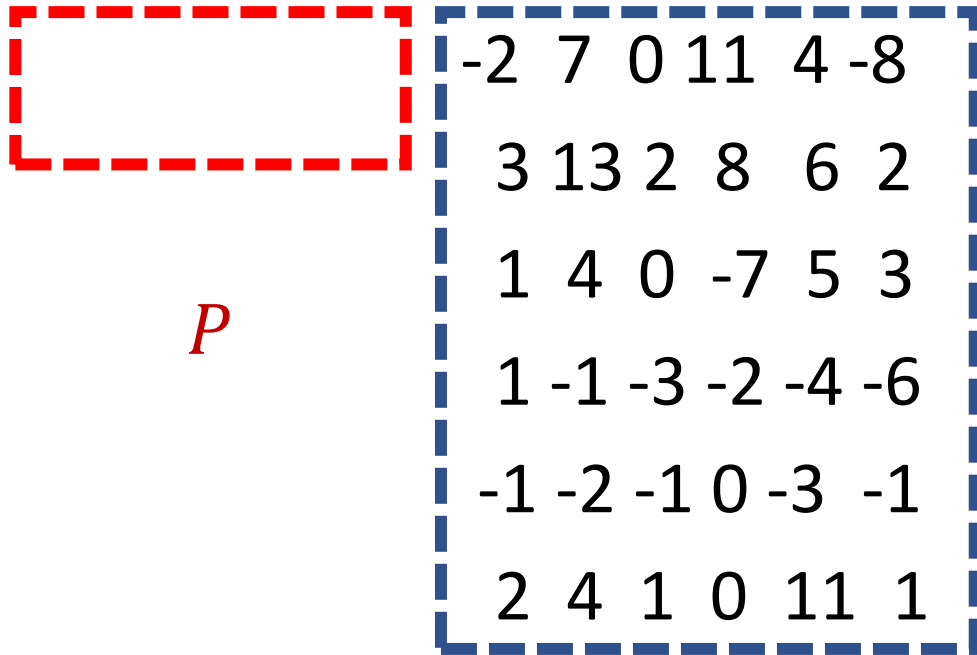
# Model #2: Online Model

- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function
- ❖ **Goal:** Use space *sublinear* in the size  $n$  of the input  $S$
  
- ❖ **Online Model:** “Each time must make irrevocable decision to the output”
  - ❖ Send decisions to some further applications downstream

# Results: Online Model

- ❖ Online algorithm for low-rank approximation that samples  $O(k)$  rows (space optimal, up to lower order terms)
- ❖ Online algorithm for row subset selection that samples  $O(k)$  rows (space optimal, up to lower order terms)
- ❖ Online algorithm for principal component analysis that embeds into a matrix with dimension  $O(k)$  (space optimal, up to lower order terms)
- ❖ Online algorithm for  $\ell_1$  subspace embedding that samples  $O(d^2)$  rows

# Challenges



The image shows a 6x6 matrix  $A$  with a 2x2 submatrix  $P$  highlighted. The matrix  $A$  is enclosed in a blue dashed border, and the submatrix  $P$  is enclosed in a red dashed border. The matrix  $A$  contains the following values:

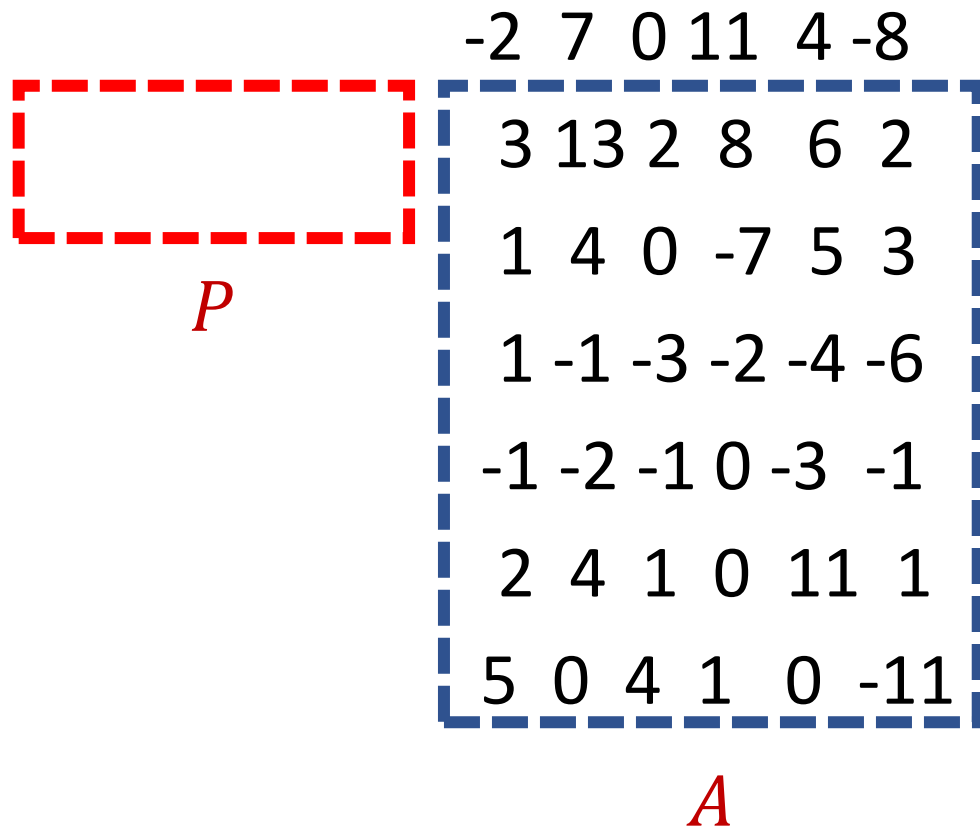
-2	7	0	11	4	-8
3	13	2	8	6	2
1	4	0	-7	5	3
1	-1	-3	-2	-4	-6
-1	-2	-1	0	-3	-1
2	4	1	0	11	1

$P$

$A$

- ❖ Approach 1: Use smooth histogram [BO07] technique for sliding windows
- ❖ ...**Functions are not smooth**
- ❖ Approach 2: Use sketching techniques for subspace embeddings
- ❖ ...**Cannot undue expirations from sliding window**

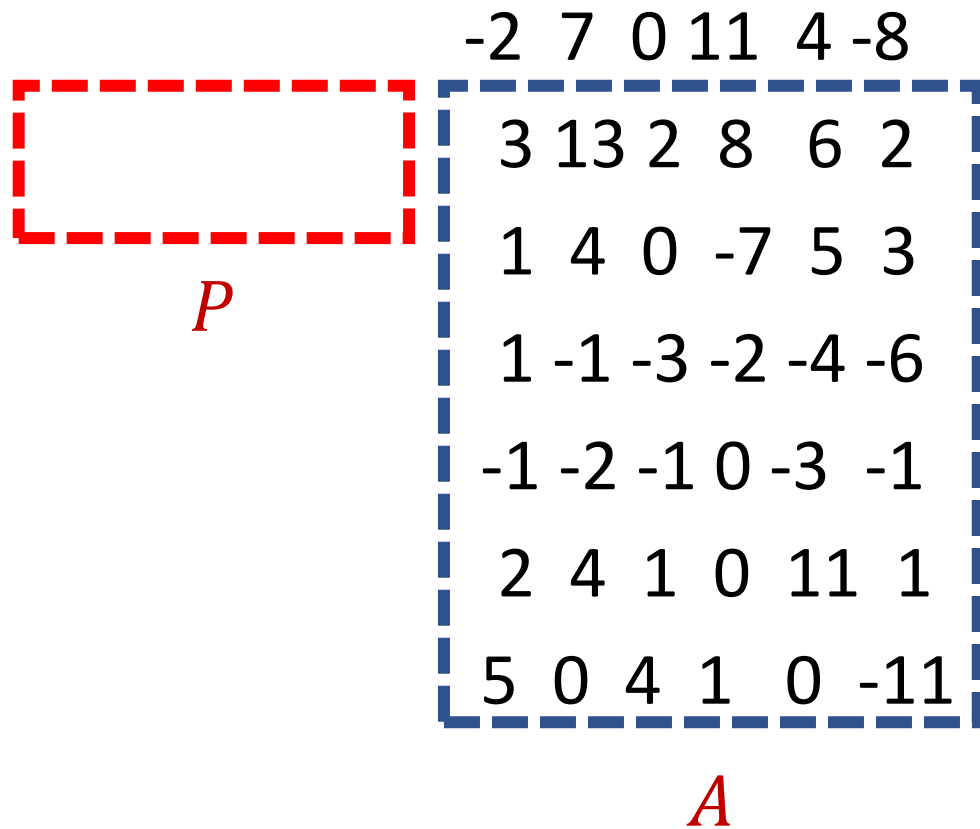
# Challenges



- ❖ Approach 1: Use smooth histogram [BO07] technique for sliding windows
- ❖ ...Functions are not smooth
- ❖ Approach 2: Use sketching techniques for subspace embeddings
- ❖ ...Cannot undue expirations from sliding window



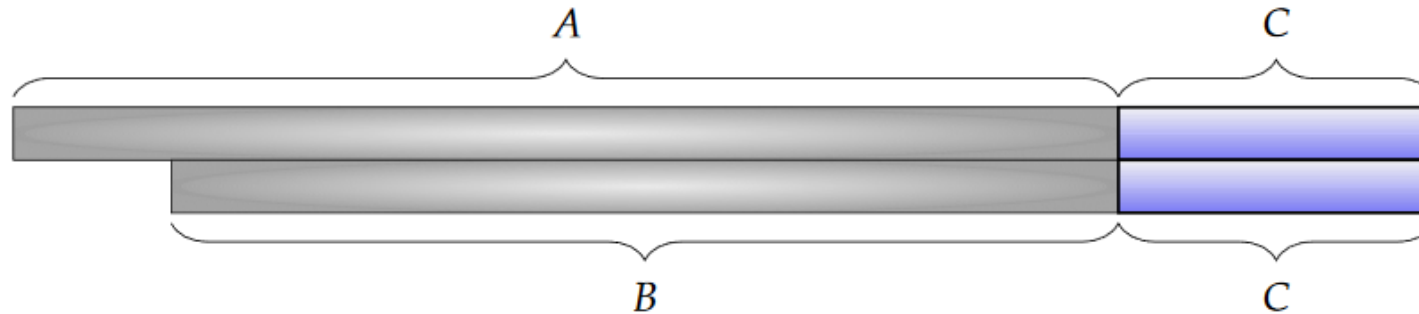
# Challenges



- ❖ Approach 1: Use smooth histogram [BO07] technique for sliding windows
- ❖ ...Functions are not smooth
- ❖ Approach 2: Use sketching techniques for subspace embeddings
- ❖ ...Cannot undue expirations from sliding window
- ❖ Approach 3: Use sampling techniques for data streams, e.g., sample rows that are unique
- ❖ ...Recent rows seem to be more important

# Sliding Window Algorithms

- ❖ Suppose we are trying to approximate some given function
  1. Suppose we have a streaming algorithm for this function
  2. Suppose this function is “smooth”: If  $f(B)$  is a “good” approximation to  $f(A)$ , then  $f(B \cup C)$  will always be a “good” approximation to  $f(A \cup C)$ .



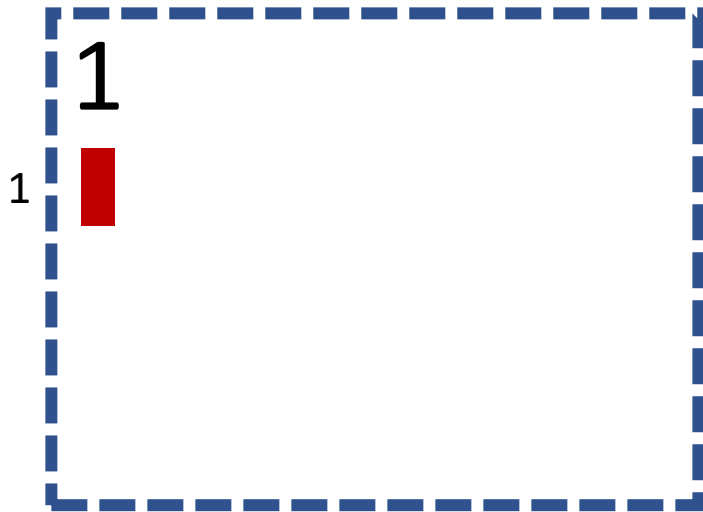
- ❖ Smooth histogram framework [BO07] gives a sliding window algorithm for this function

# Smooth Histogram

- ❖ Suppose we are trying to approximate some given function
- ❖ Smooth histogram framework [BO07] gives a sliding window algorithm for this function
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window

# Smooth Histogram

- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

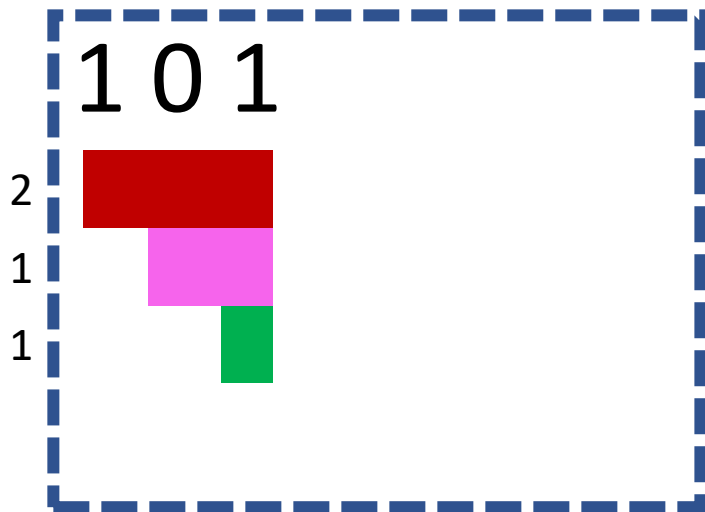
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

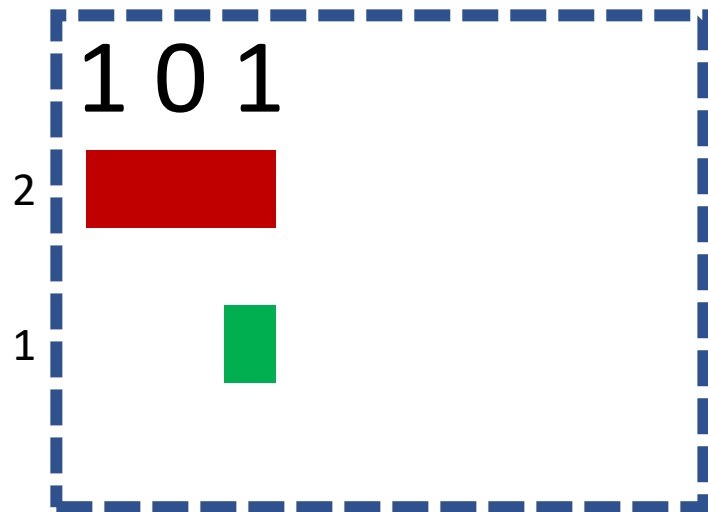
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

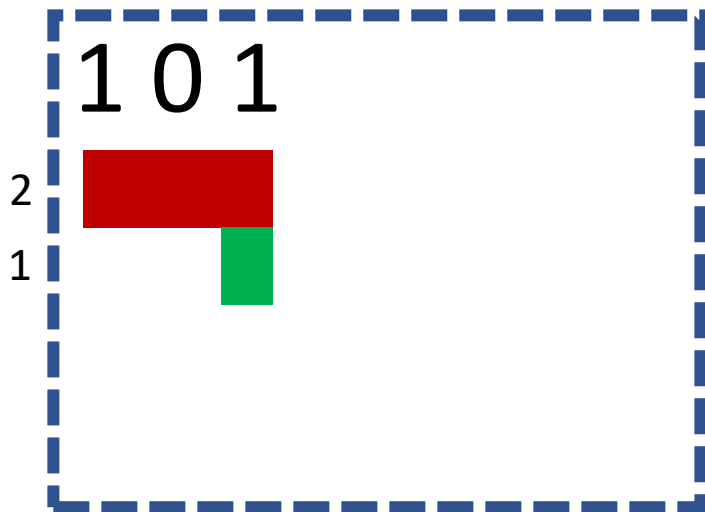
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window

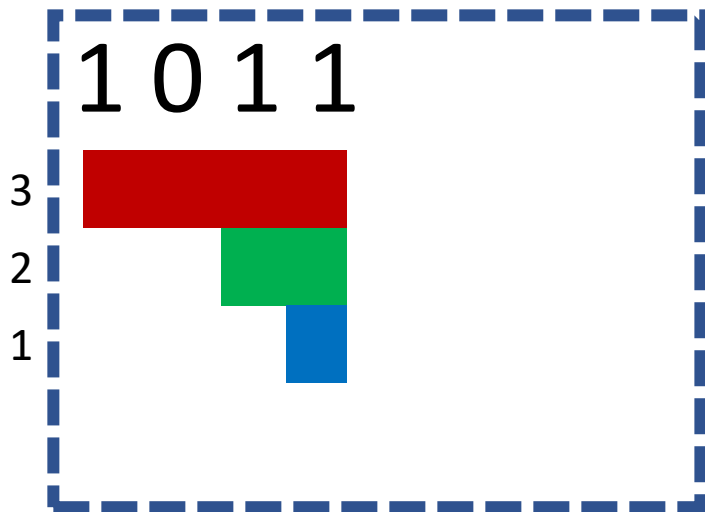


- ❖ Example: Number of ones in sliding window (2-approximation)



# Smooth Histogram

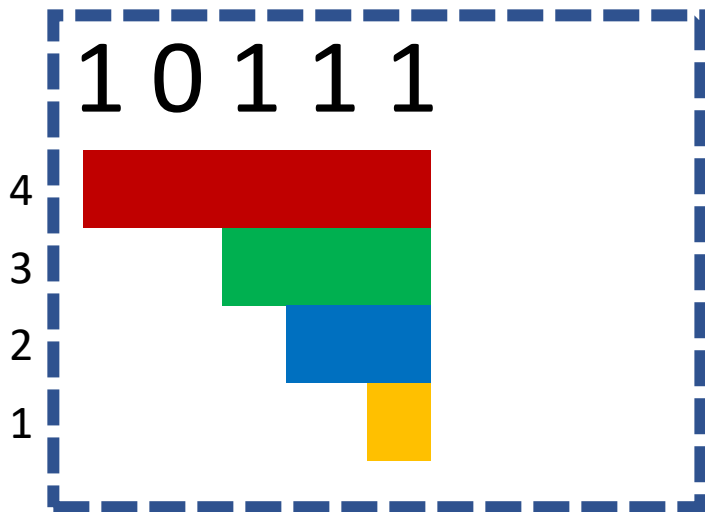
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

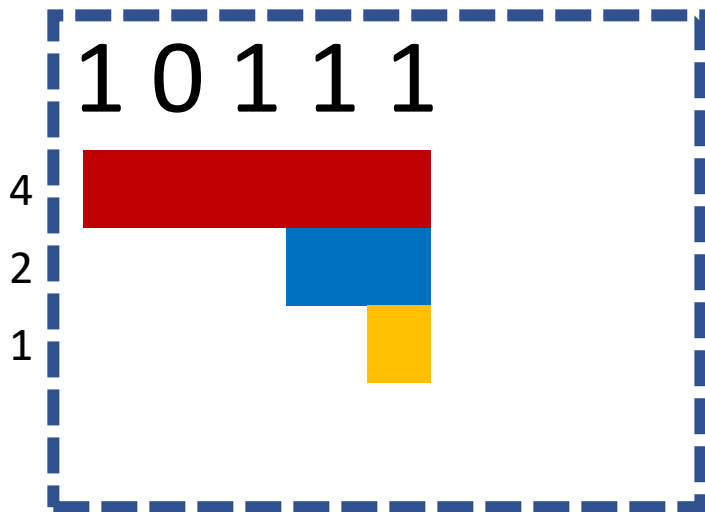
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

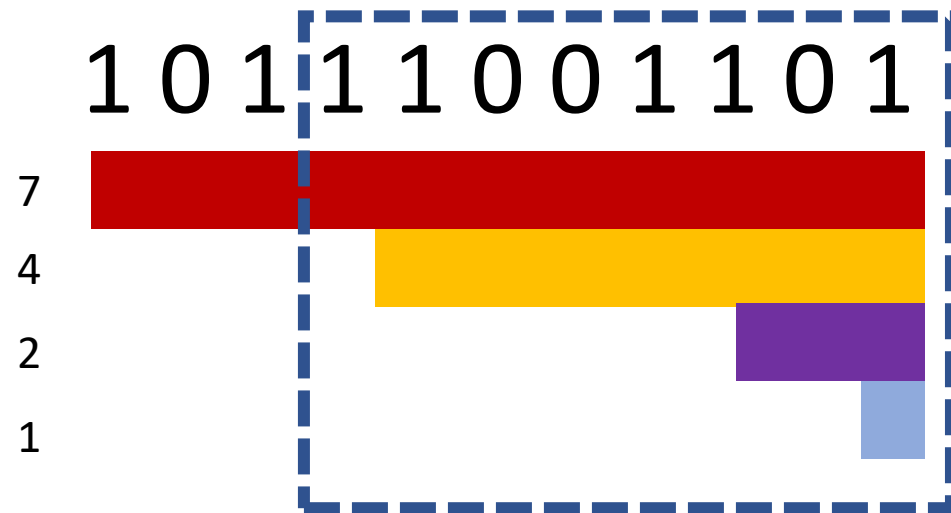
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

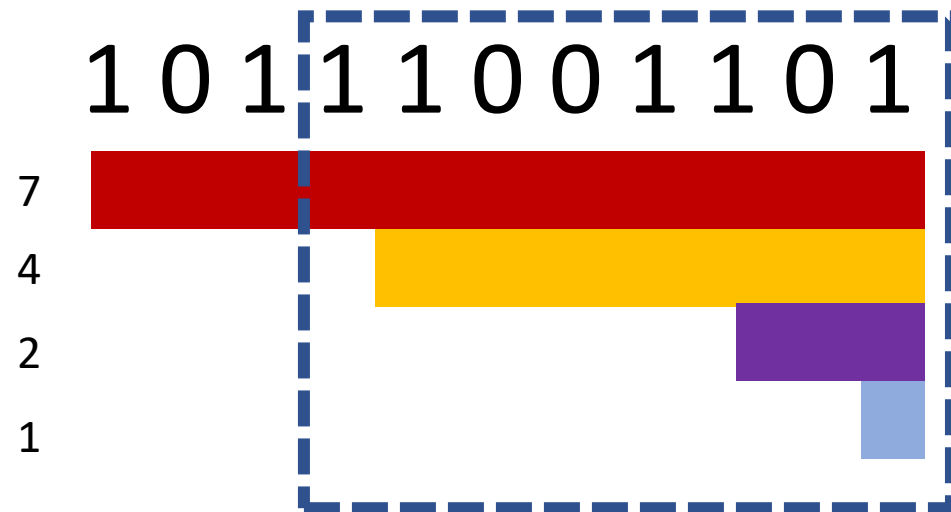
- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)

# Smooth Histogram

- ❖ Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- ❖ Each time there are three instances that report “close” values, delete the middle one
- ❖ Use different checkpoints to “sandwich” the sliding window



- ❖ Example: Number of ones in sliding window (2-approximation)
- ❖ Number of ones in sliding window is at least 4 and at most 7
- ❖ 4 is a good approximation

# Streaming Model Algorithms

- ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
- ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
- ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...
- ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...)
- ❖ Submodular optimization
- ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
- ❖ Codeword testing

# Smooth Functions

- ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
- ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
- ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...
- ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...)
- ❖ Submodular optimization
- ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
- ❖ Codeword testing

# Smooth Functions

- ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
- ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
- ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...
- ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...)
- ❖ Submodular optimization
- ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
- ❖ Codeword testing



# Smooth Functions

[BGO13, BGLWZ18]

[BOZ09]

- ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
  - ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
  - ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...
- [BLLM16]
- ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...)
  - ❖ Submodular optimization [CNZ16,ELVZ17]
  - ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
  - ❖ Codeword testing

# Smooth Functions

[BGO13, BGLWZ18]

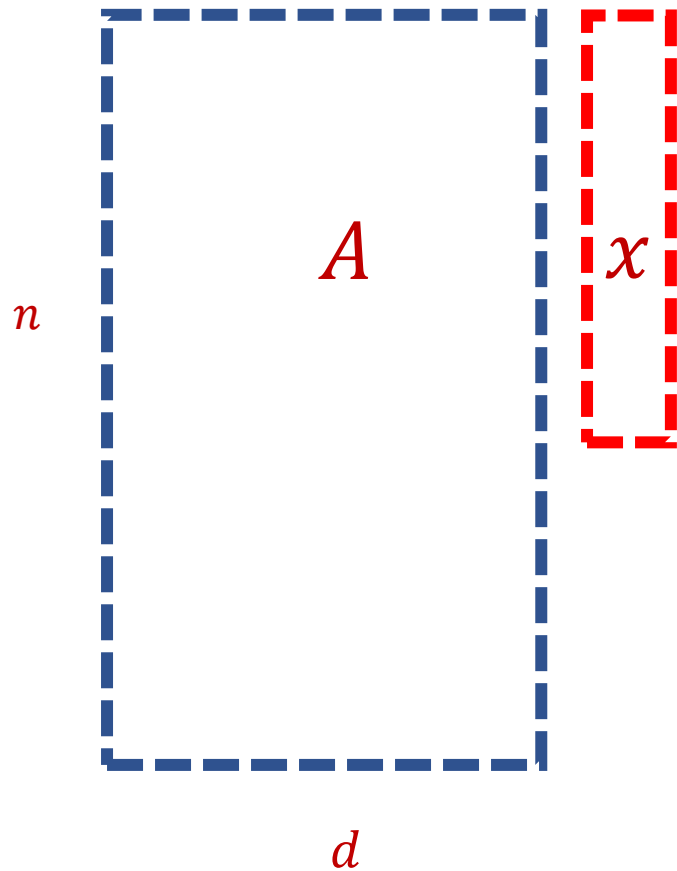
[BOZ09]

- ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
- ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
- ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...

[BLLM16]

- ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...) ✓
- ❖ Submodular optimization [CNZ16,ELVZ17]
- ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
- ❖ Codeword testing

# Spectral Approximation

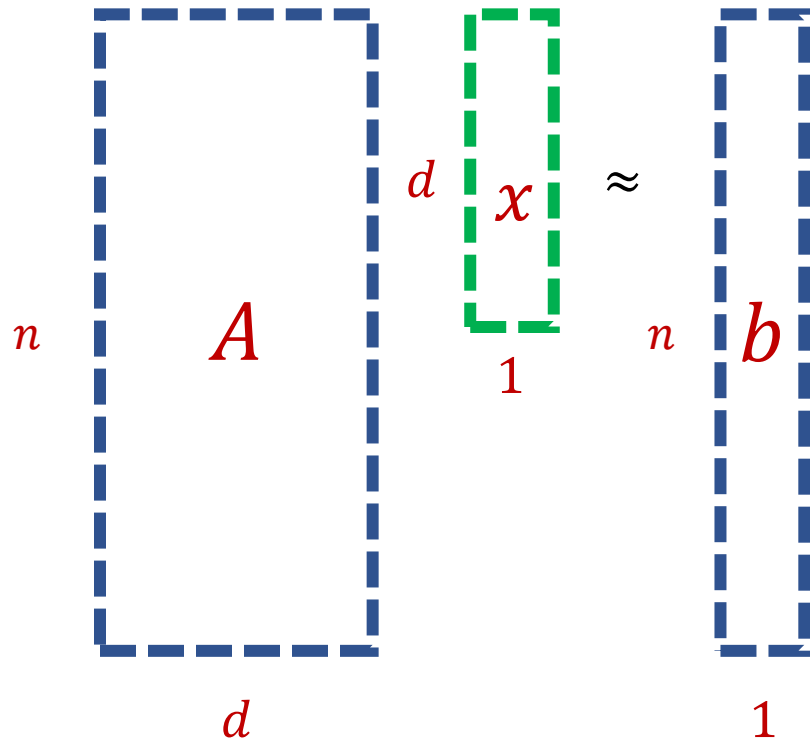


- ❖ Spectral approximation: Given  $\epsilon > 0$  and  $A \in \mathbb{R}^{n \times d}$ , find matrix  $M \in \mathbb{R}^{m \times d}$  with  $m \ll n$ , such that for *every*  $x \in \mathbb{R}^d$ ,

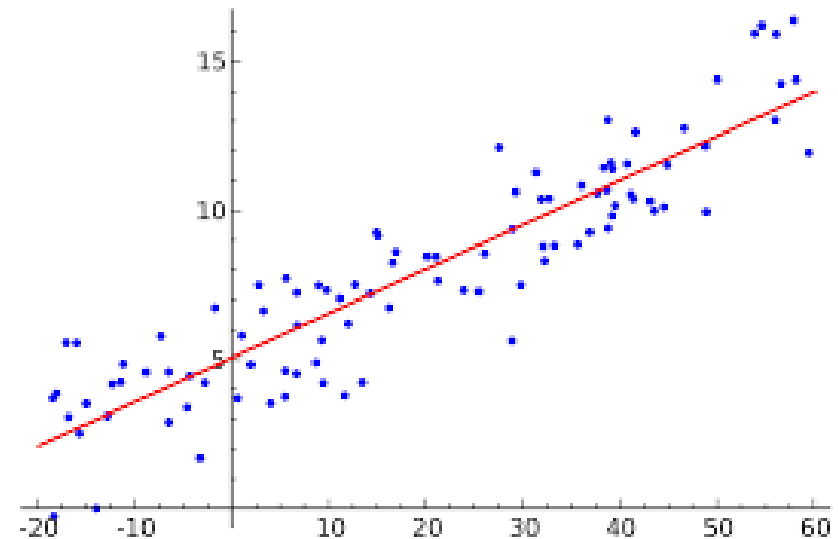
$$(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

- ❖ Equivalent to  $(1 - \epsilon)A^T A \preceq M^T M \preceq (1 + \epsilon)A^T A$

# Linear Regression



- ❖ Find the vector  $x$  that minimizes  $\|Ax - b\|_2$
- ❖ “Least squares” optimization



# Schatten Norms

- ❖ Spectral approximation: Given  $\epsilon > 0$  and  $A \in R^{n \times d}$ , find matrix  $M \in R^{m \times d}$  with  $m \ll n$ , such that for every  $x \in R^d$ ,

$$(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

- ❖ Equivalent to  $(1 - \epsilon)A^T A \preceq M^T M \preceq (1 + \epsilon)A^T A$
- ❖ Singular value: square root of eigenvalue of  $A^T A$ 
  - ❖  $\sigma_1(A) \geq \sigma_2(A) \geq \dots \geq \sigma_n(A)$
- ❖ Schatten  $p$  norm:  $\|A\|_p = (\sigma_1^p + \sigma_2^p + \dots + \sigma_n^p)^{\frac{1}{p}}$

# Linear Algebra Background

- ❖ A symmetric matrix  $M \in R^{d \times d}$  is positive semi-definite (PSD) if  $x^T M x \geq 0$  for all column vectors  $x \in R^d$
- ❖ All eigenvalues of PSD matrix  $M$  are non-negative
- ❖ If  $A - B$  is PSD, we write  $B \preceq A$
- ❖ For any row vector  $v \in R^d$ ,  $v^T v$  is a PSD matrix
- ❖ Sum of two PSD matrices is a PSD matrix

# Initial Approach (Smooth PSD Histogram)

❖ If  $(1 - \epsilon)B^T B \preceq A^T A \preceq (1 + \epsilon)B^T B$ , then for any matrix  $C$ ,

$$(1 - \epsilon)(B^T B + C^T C) \preceq A^T A + C^T C \preceq (1 + \epsilon)(B^T B + C^T C)$$

❖ The singular values of the matrices behave “smoothly”

❖ Maintain histogram based on the singular values



❖ Each substream represents a matrix  $A$

❖ Keep  $A^T A$  and merge whenever there are three matrices within  $(1 + \epsilon)$  in Loewner ordering

# Initial Approach (Smooth PSD Histogram)

- ❖ Space? Each instance stores a matrix  $A^T A$
- ❖  $A$  has at most  $n$  rows but  $A^T A \in R^{d \times d}$
- ❖ How many instances?  $d$  singular values, each of them polynomially bounded
- ❖  $O\left(\frac{1}{\epsilon} d \log n\right)$  instances
- ❖ Total space:  $O\left(\frac{1}{\epsilon} d^3 \log n\right)$  (in words)



# Summary

- ❖ **Algorithm:** Mimic smooth histogram by keeping checkpoints whenever singular values “jump” and storing  $A^T A$
- ❖ **Correctness:** smoothness of Loewner ordering
- ❖ **Space:**  $O\left(\frac{1}{\epsilon} d^3 \log d\right)$

# Questions?



# Initial Approach (Smooth PSD Histogram)

❖ Deterministic algorithm:  $O\left(\frac{1}{\epsilon} d^3 \log n\right)$  space (in words)



❖ Outputs spectral approximation of  $A^T A$  rather than  $A$  (does not generalize to other norms)

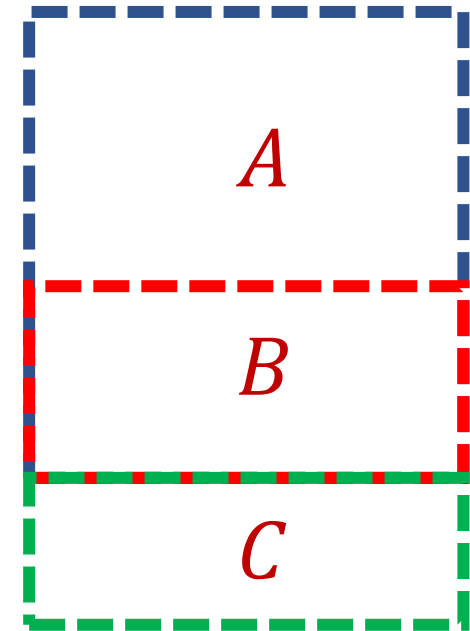
❖ Update time can be  $O\left(\frac{1}{\epsilon} d^4 \log n\right)$

❖ Can be done in  $\tilde{O}\left(\frac{1}{\epsilon^2} d^2\right)$  space in streaming



# Intuition

- ❖ To decrease the space, we first observe there is a lot of similar structure between instances  $A, B, C$ : most rows are shared!
- ❖ Try subsampling approach?
- ❖ Uniform sampling of rows is generally poor
- ❖ **Importance sampling**: Matrix multiplication uses squared row norm, but it doesn't work here...



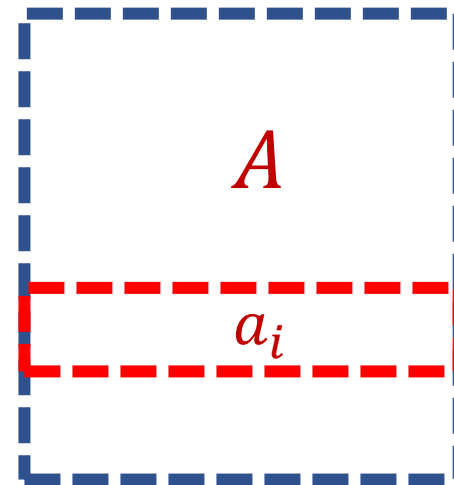
# Leverage Scores

- ❖ Intuition: how **unique** a row is (importance sampling)
- ❖  $\ell_i = \max \frac{\langle a_i, x \rangle^2}{\|Ax\|_2^2}$  are the *leverage scores* of  $A$  (in this case of row  $a_i$ )

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

- ❖ Take  $x = (1 \ -1)$  to see that  $\ell_1 = 1$
- ❖ Take  $x = (0 \ 1)$  to see that  $\ell_2 = 1$

- ❖  $\ell_i = a_i(A^T A)^{-1} a_i^T, \sum \ell_i = d$

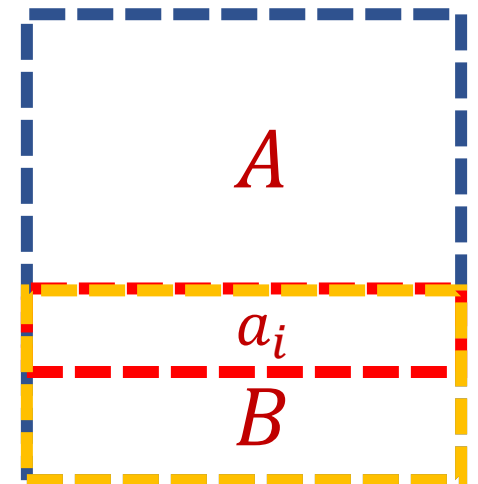
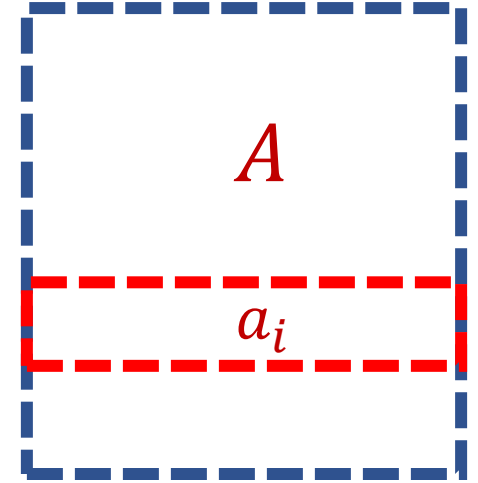






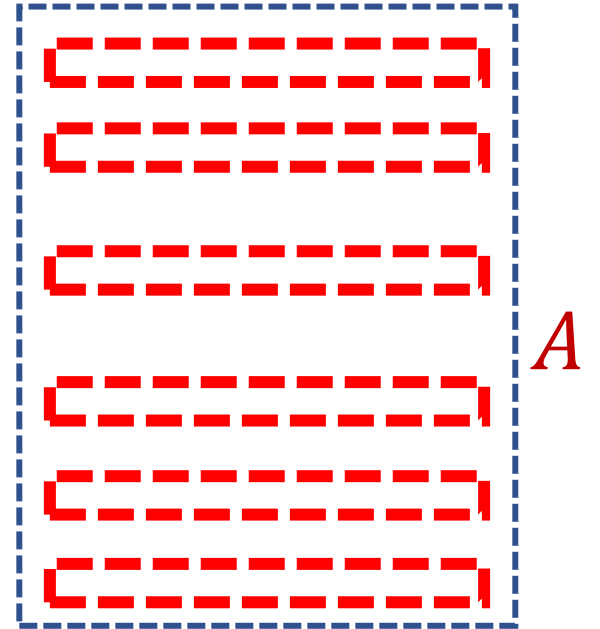
# Reverse Online Leverage Scores

- ❖ Leverage score of row  $a_i$  is  $\ell_i = a_i(A^\top A)^{-1}a_i^\top$
- ❖ Rows before  $a_i$  might be deleted so they shouldn't count towards the importance of  $a_i$
- ❖ **Reverse online leverage score** of row  $a_i$  is  $\tau_i = a_i(B^\top B)^{-1}a_i^\top$  where  $B$  are the rows after  $a_i$



# Algorithm

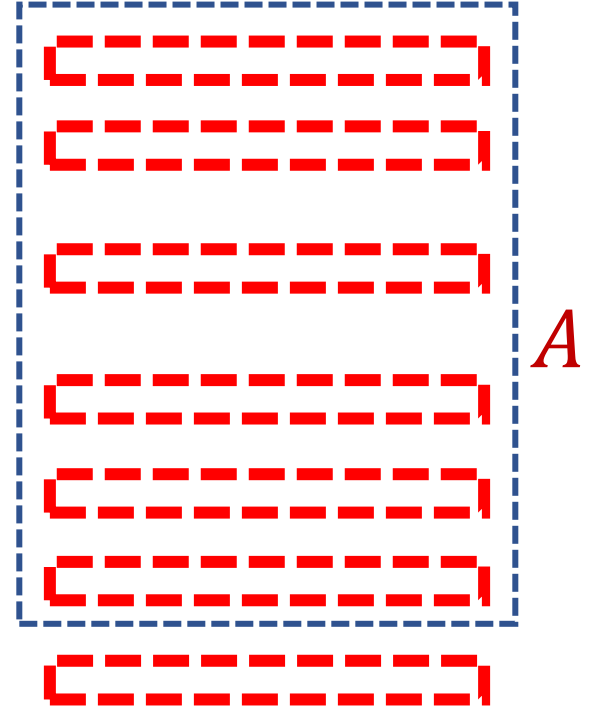
- ❖ Algorithm: sample (and rescale) a number of rows





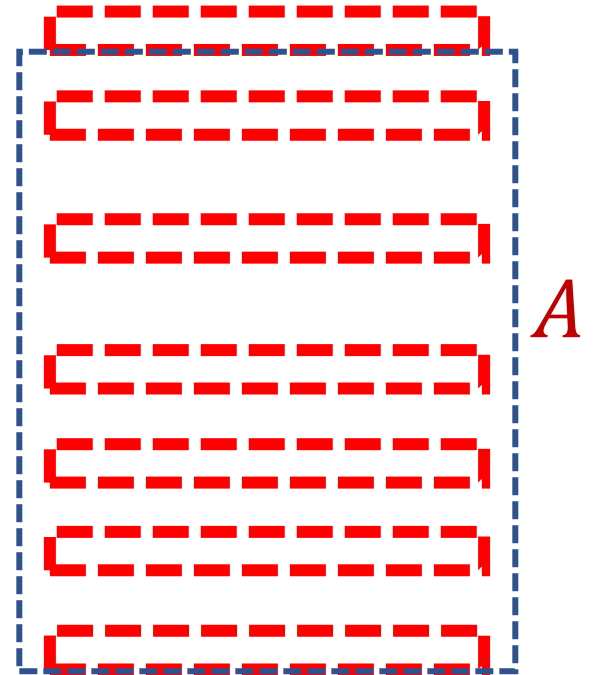
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it



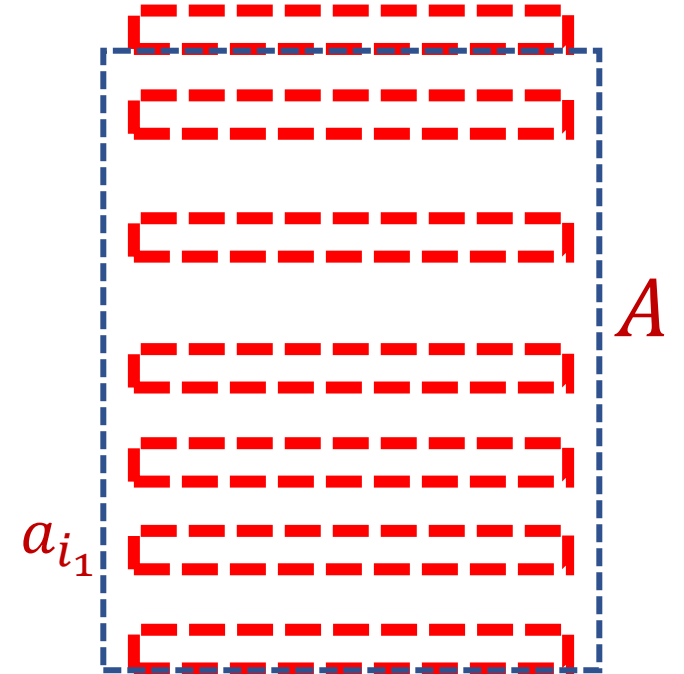
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it



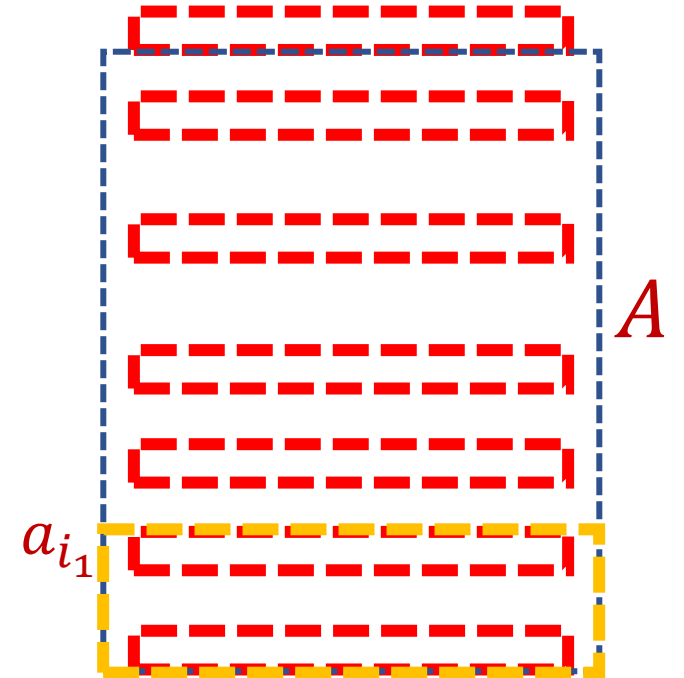
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



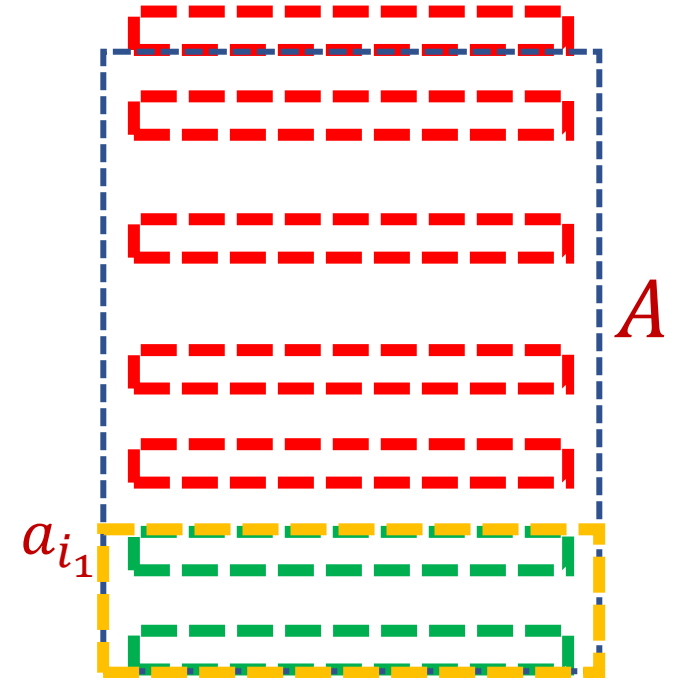
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



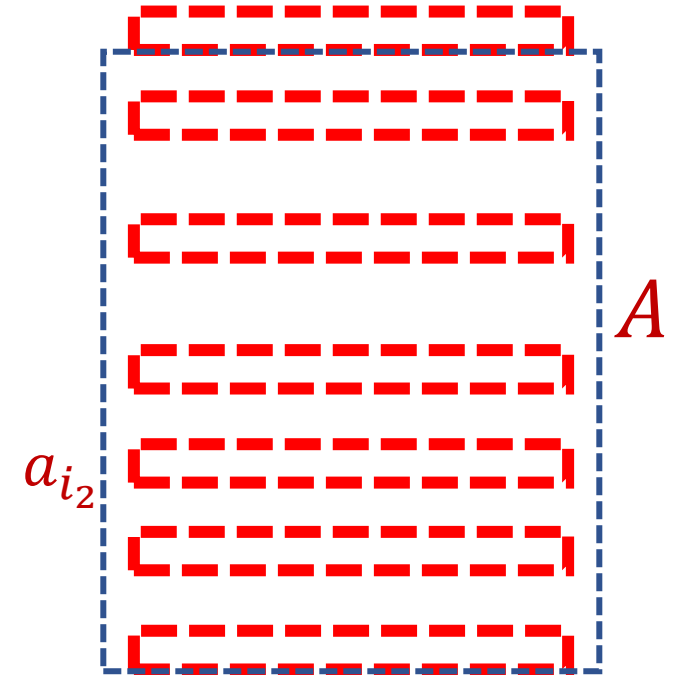
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



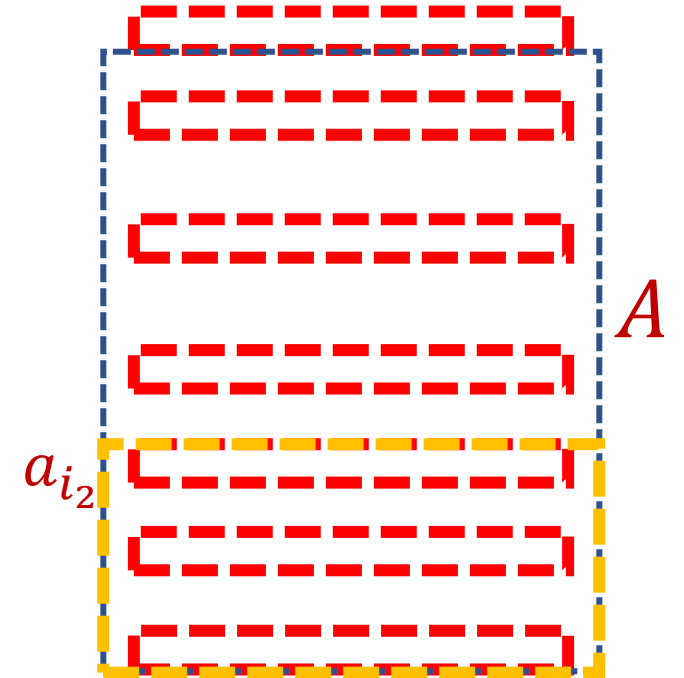
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



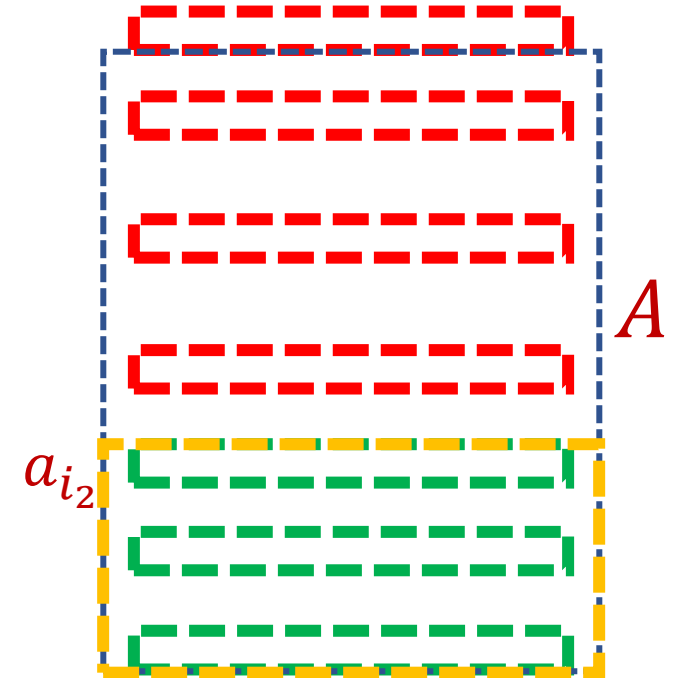
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



# Algorithm

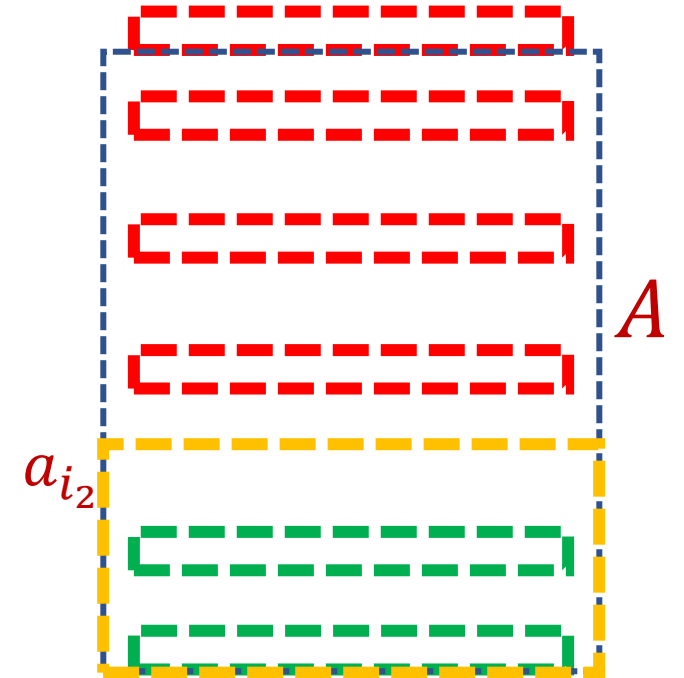
- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling





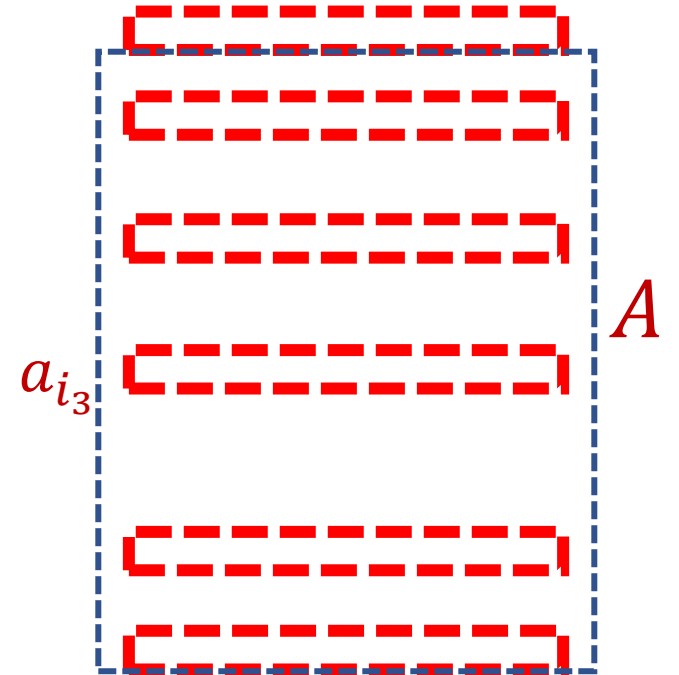
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



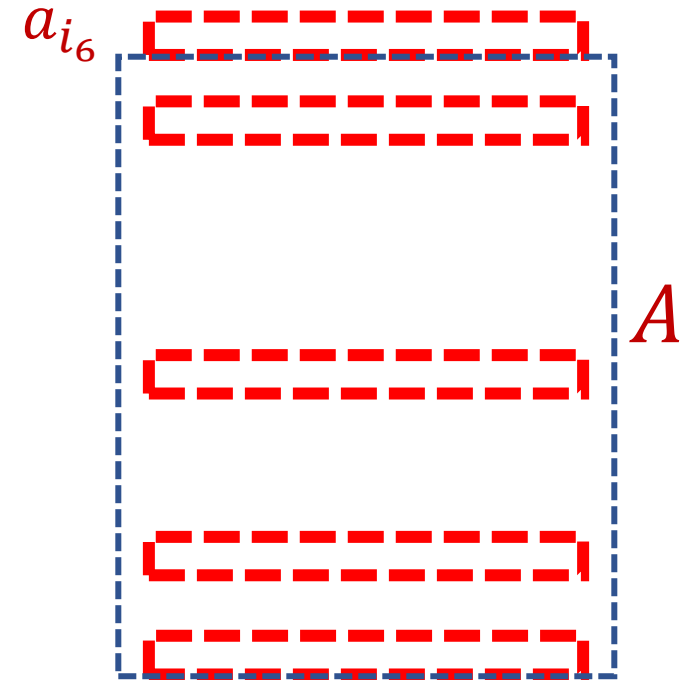
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



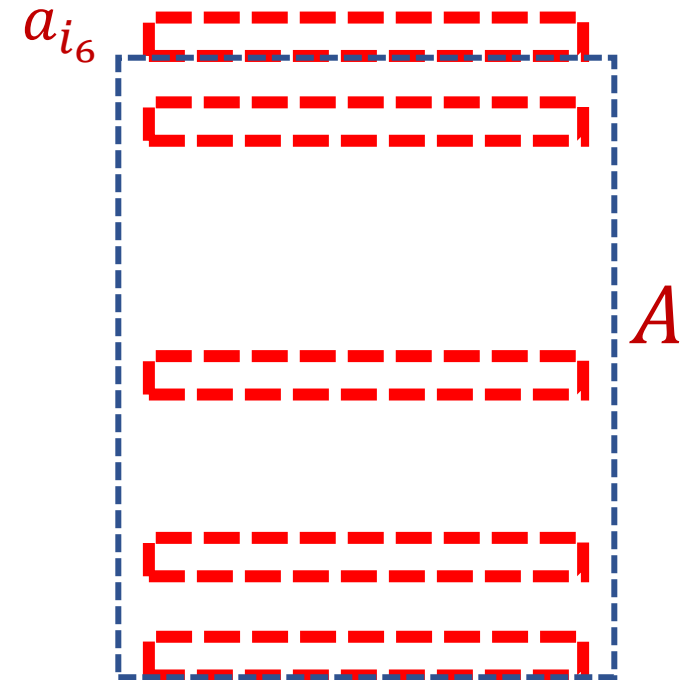
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling



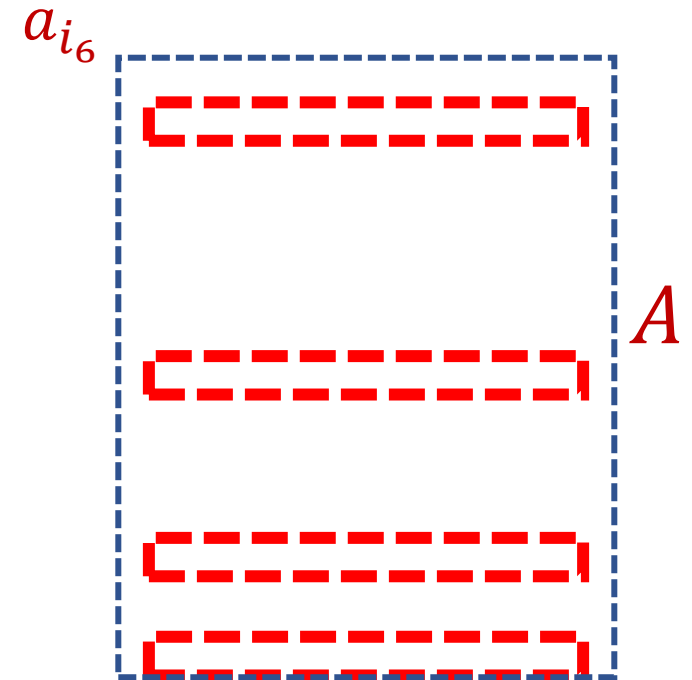
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling
- ❖ Delete expired rows



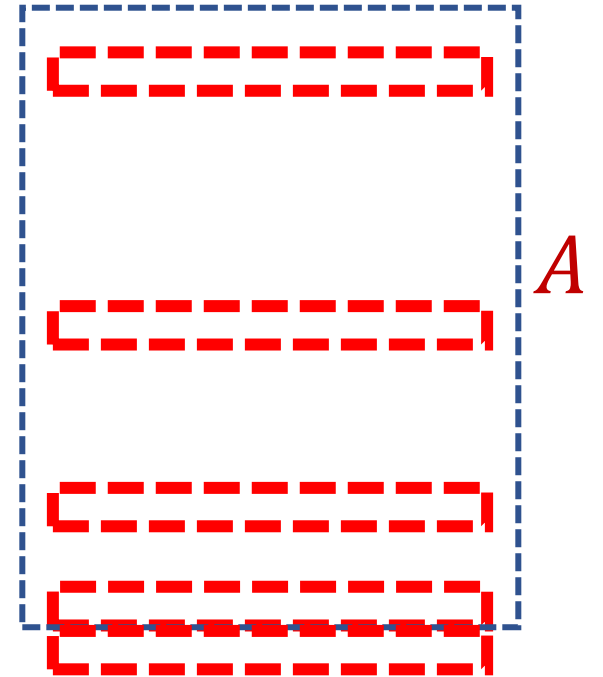
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling
- ❖ Delete expired rows



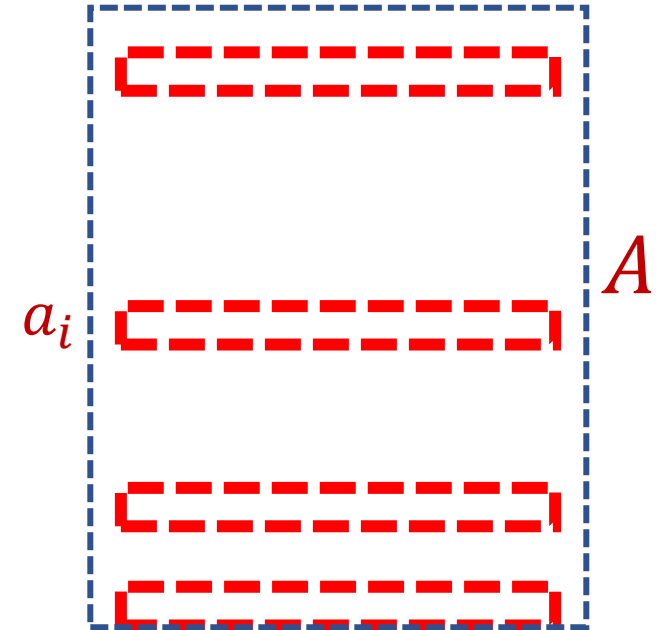
# Algorithm

- ❖ Algorithm: sample (and rescale) a number of rows
- ❖ New row arrives – store it
- ❖ For each sampled (and rescaled) row  $a_i$ , sample the row with probability  $\propto \tau_i \leftrightarrow$  downsampling
- ❖ Delete expired rows
- ❖ New row arrives – repeat



# Correctness

- ❖ **Correctness:** Show an invariant that each sampled matrix is good approximation to row  $a_i$  is sampled with probability  $\propto$  *final* reverse online leverage score
- ❖ Shown by martingale argument and monotonicity



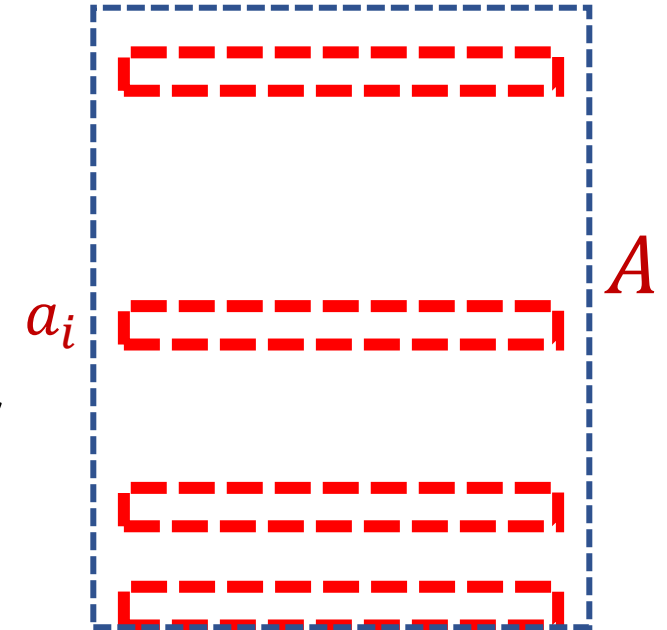
# Correctness

- ❖ **Want to show:**  $(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$  for all  $x$
- ❖ **Approach 1:** Fix  $x \in R^d$  and observe that row  $a_i$  contributes  $\frac{1}{p_i} \langle a_i, x \rangle^2$  to  $\|Mx\|_2^2$  with probability  $p_i$  and  $0$  otherwise and consider  $|\|Mx\|_2^2 - \|Ax\|_2^2|$  (Bernstein-type martingale)
- ❖ Show  $(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$  for all  $x$  in an  $\epsilon$ -net of a unit ball
- ❖ **Approach 2:** Observe row  $a_i$  contributes  $a_i^\top a_i$  to  $M^\top M$  with probability  $p_i$  and  $0$  otherwise and consider  $\|M^\top M - A^\top A\|_2$  (matrix Freedman martingale)



# Spectral Approximation (Summary)

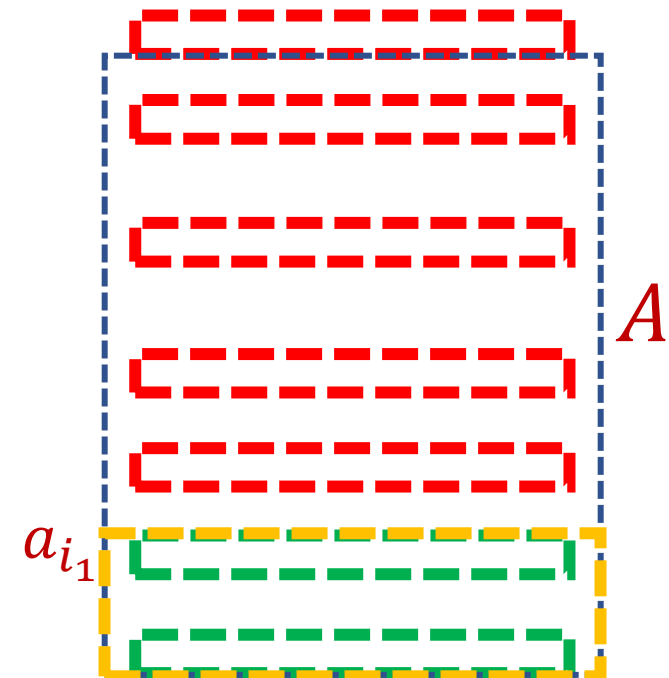
- ❖ **Correctness:** Show an invariant that each sampled matrix is good approximation to row  $a_i$  is sampled with probability  $\propto$  *final* reverse online leverage score
- ❖ Shown by martingale argument and monotonicity
- ❖ Outputs a matrix  $M \in R^{m \times d}$  such that  $(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$  for all  $x \in R^n$  [DMM06, SS08, CMP16]
- ❖ **Space?** Must bound  $\sum \tau_i = \tilde{O}\left(\frac{1}{\epsilon^2} d\right)$  [CMP16]



# Summary

- ❖ **Algorithm:** Start from most recent row and downsample through (reverse) online leverage score
- ❖ **Correctness:** expectation is correct, bound the variance and use matrix martingale
- ❖ **Space:**  $\tilde{O}\left(\frac{1}{\epsilon^2} d^2\right)$

# Questions?



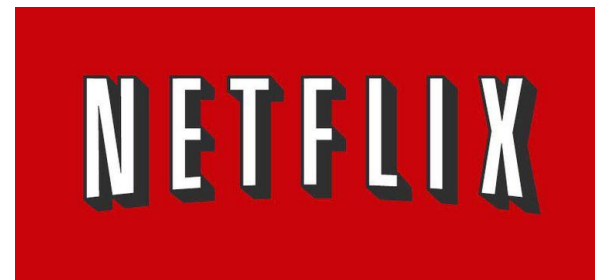
# Low-Rank Approximation

$n$

1	3	5	-2	7	0	11	4	-8
0	0	-1	3	13	2	8	6	2
2	5	6	1	4	0	-7	5	3
8	7	2	1	-1	-3	-2	-4	-6
-5	3	-4	-1	-2	-1	0	-3	-1
7	1	3	2	4	1	0	11	1

$d$

- ❖ Find rank  $k$  matrix  $A_k$  that minimizes  $\|A_k - A\|_F$
- ❖ Finding structure among noise
- ❖ Matrix completion problem



# Low-Rank Approximation

- ❖ Low-rank approximation: Given  $\epsilon > 0$  and  $A \in R^{n \times d}$ , find matrix  $M \in R^{m \times d}$  with  $m \ll n$  such that

$$(1 - \epsilon) \|A - A_k\|_F \leq \|M - M_k\|_F \leq (1 + \epsilon) \|A - A_k\|_F$$

- ❖ Spectral approximation algorithm solves low-rank approximation:  $\tilde{O}\left(\frac{1}{\epsilon^2} d^2\right)$  space



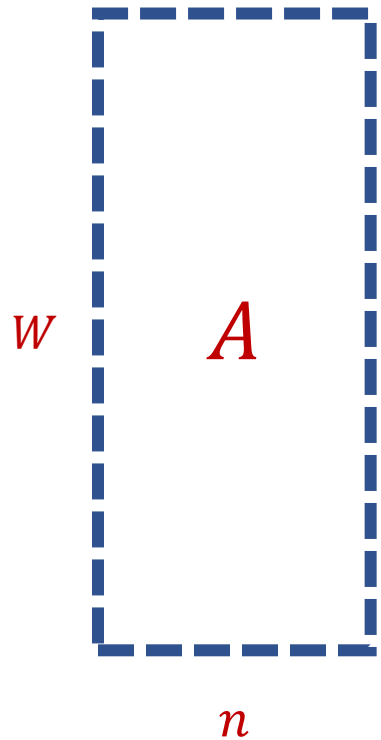
- ❖ Can be done in  $\tilde{O}\left(\frac{1}{\epsilon^2} kd\right)$  space in streaming



# Low-Rank Approximation

- ❖ Low-rank approximation: Given  $\epsilon > 0$  and  $A \in R^{n \times d}$ , find matrix  $M \in R^{m \times d}$  with  $m \ll n$  such that

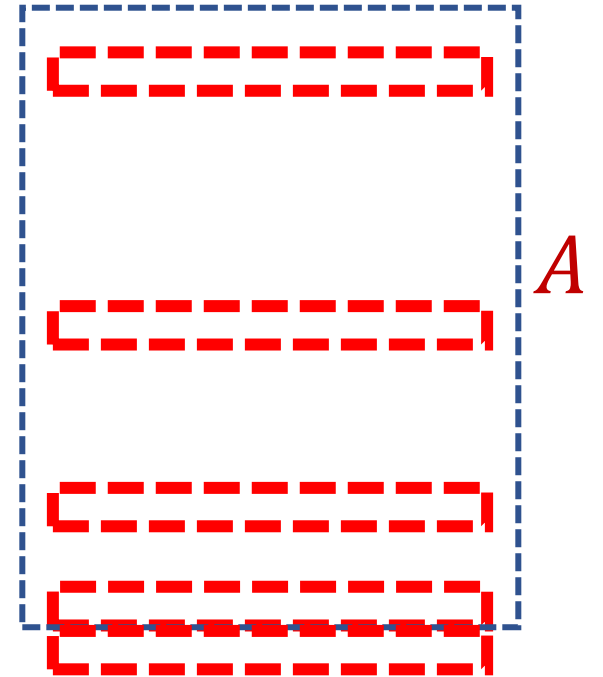
$$(1 - \epsilon) \|A - A_k\|_F^2 \leq \|M - M_k\|_F^2 \leq (1 + \epsilon) \|A - A_k\|_F^2$$



- ❖ Leverage score:  $a_i(A^T A)^{-1} a_i^T$
- ❖ Ridge leverage score:  $\ell_i = a_i(A^T A + \lambda I_n)^{-1} a_i^T$ ,  
where  $\lambda = \frac{\|A - A_k\|_F^2}{k}$

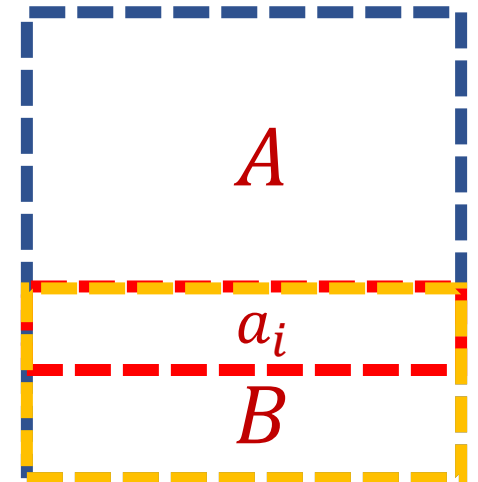
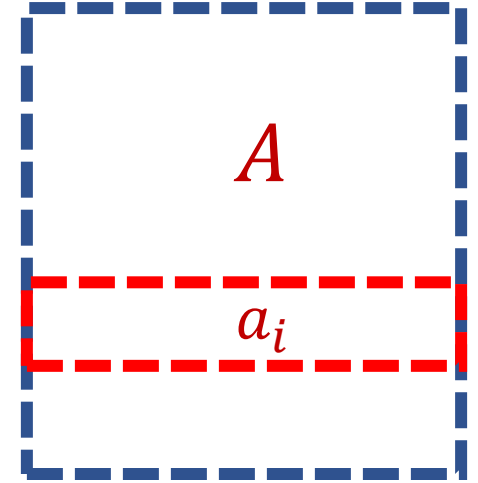
# Algorithmic Template (Sliding Windows)

- ❖ **Algorithm:** at each time, repeatedly **downsample** each stored row  $a_i$  based on a probability distribution  $\propto \tau_i$  “reverse online score” that accounts for both **importance** AND **recency** of a row with respect to the desired function
- ❖ **Correctness:** Show an invariant that each sampled matrix is a “good approximation” and row  $a_i$  is sampled with probability  $\propto$  *final* reverse online score
- ❖ **Space?** Must bound  $\sum \tau_i$



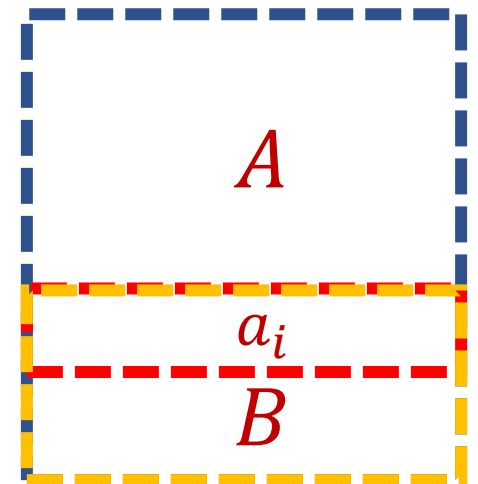
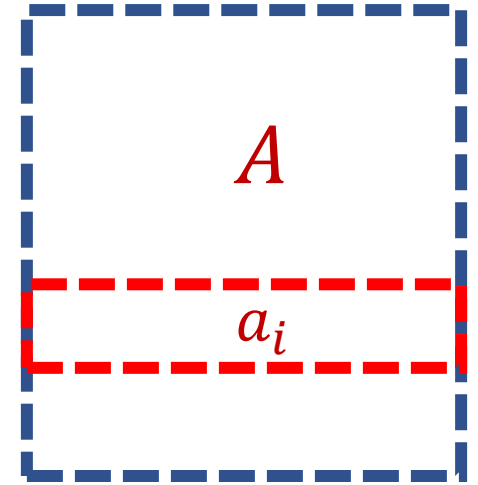
# Template

- ❖ Suppose we know  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ❖ Reverse online leverage score: Sample each row  $a_i$  with probability  $p_i \propto \tau_i = a_i(B^\top B + \lambda I_n)^{-1} a_i^\top$
- ❖ Monotonicity of ridge leverage score
- ❖ Outputs a matrix  $M \in R^{m \times d}$  such that  $(1 - \epsilon)\|A - A_k\|_F \leq \|M - M_k\|_F \leq (1 + \epsilon)\|A - A_k\|_F$  [CMM15]



# Template

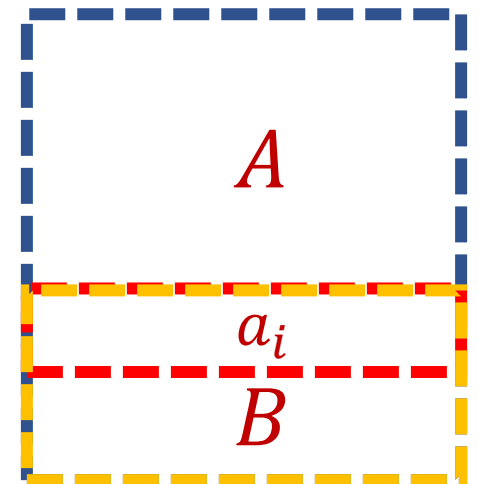
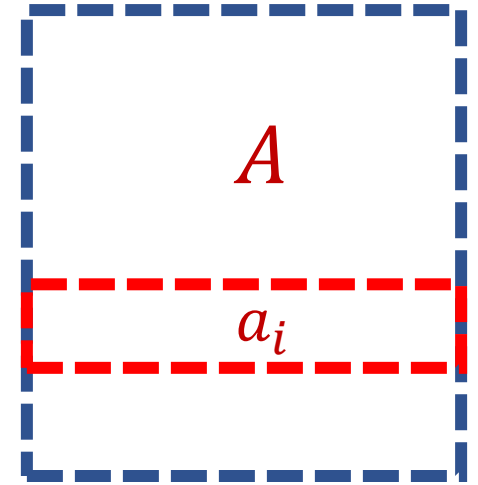
- ❖ Issue #1: Compute  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ❖ Issue #2: Bound  $\sum \tau_i$





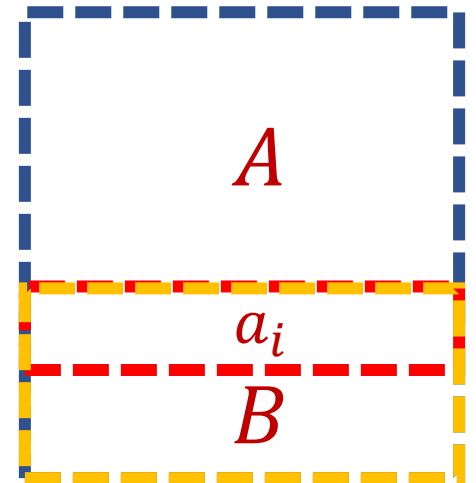
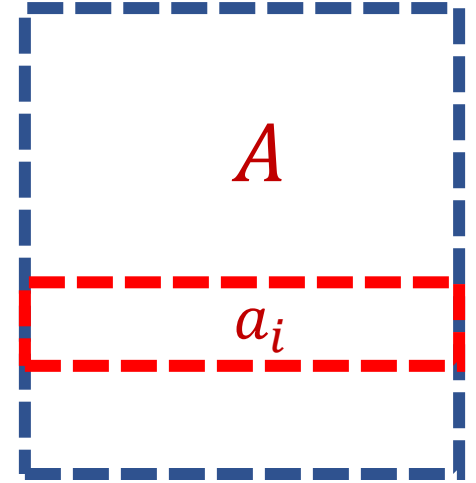
# Template

- ❖ Issue #1: Compute  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ❖ Issue #2: Bound  $\sum \tau_i$
- ❖ **Observation**: it suffices to have a constant factor approximation of  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ❖ Use projection-cost preserving sketch [CEMMP15] to reduce the dimension of each row
- ❖ Feed reduced rows into spectral approximation algorithm



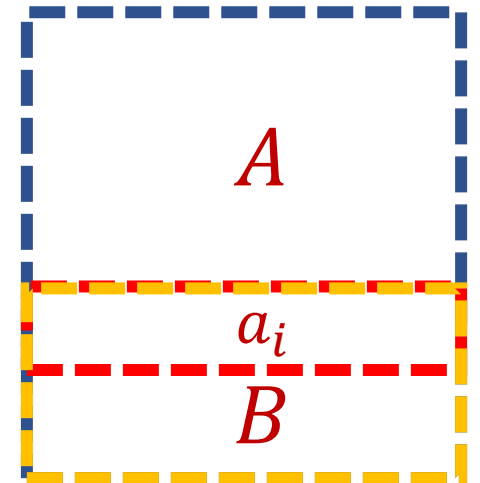
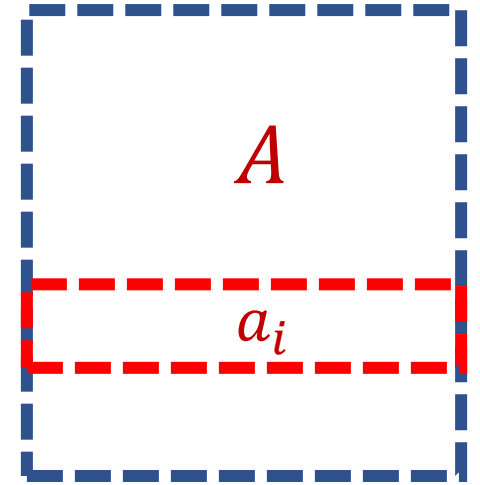
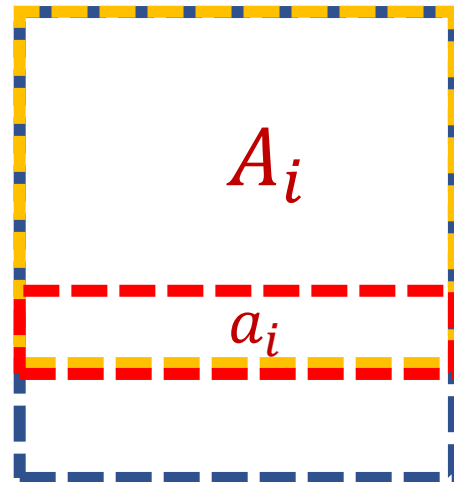
# Template

- ✓ Issue #1: Compute  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ✦ Issue #2: Bound  $\sum \tau_i$



# Template

- ✓ Issue #1: Compute  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ❖ Issue #2: Bound  $\sum \tau_i$
- ❖  $A_i$  be the first  $i$  rows of  $A$
- ❖  $\tau_i = a_i (A_i^\top A_i + \lambda I_n)^{-1} a_i^\top$
- ❖ Bound  $\sum \tau_i$



# Number of Total Rows

$$\blacklozenge \det(A^\top A + \lambda I_d) \geq \lambda^d e^{\sum \tau_i / 2} \text{ [CMP16]}$$

$$\blacklozenge \frac{\sum \tau_i}{2} + d \log \lambda \leq \log \det(A^\top A + \lambda I_d)$$

$$\blacklozenge \log \det(A^\top A + \lambda I_n) = O(k \log n) + d \log \lambda$$

$$\blacklozenge \sum \tau_i = O(k \log n)$$

# Using Matrix Determinant Lemma [CMP16]

- ❖  $\det(A + v^\top v) = \det(A)(1 + vA^{-1}v^\top)$
- ❖  $\det(A^\top A + \lambda I_d + v^\top v) = \det(A^\top A + \lambda I_d) (1 + v(A^\top A + \lambda I_d)^{-1}v^\top)$
- ❖  $\tau_i = a_i(A_i^\top A_i + \lambda I_d)^{-1} a_i^\top$

$$\begin{aligned}\det(A^\top A + \lambda I_d) &= \det(A_{n-1}^\top A_{n-1} + \lambda I_d) (1 + a_n(A_{n-1}^\top A_{n-1} + \lambda I_d)^{-1} a_n^\top) \\ &= \det(A_{n-1}^\top A_{n-1} + \lambda I_d) (1 + \tau_n) \\ &\geq \det(A_{n-1}^\top A_{n-1} + \lambda I_d) (1 + e^{\tau_n/2})\end{aligned}$$

$$\det(A^\top A + \lambda I_d) \geq \lambda^d e^{\sum \tau_i/2}$$

# Number of Total Rows

$$\blacklozenge \det(A^\top A + \lambda I_d) \geq \lambda^d e^{\sum \tau_i / 2} \text{ [CMP16]}$$

$$\blacklozenge \frac{\sum \tau_i}{2} + d \log \lambda \leq \log \det(A^\top A + \lambda I_d)$$

$$\blacklozenge \log \det(A^\top A + \lambda I_d) = O(k \log n) + d \log \lambda$$

$$\blacklozenge \sum \tau_i = O(k \log n)$$

# Number of Total Rows

- ❖  $\det(A^\top A + \lambda I_d) = \prod \sigma_i(A^\top A + \lambda I_d)$
- ❖ Small singular values:  $\sigma_{k+1} + \dots + \sigma_d = \|A - A_k\|_F^2 + \lambda(d - k)$
- ❖ By AM-GM,

$$\prod_{i=k+1}^{i=d} \sigma_i \leq \left( \frac{\|A - A_k\|_F^2 + \lambda(d - k)}{d - k} \right)^{d-k}$$

- ❖ Large singular values:  $\sigma_i \leq \|A\|_2^2 + \lambda$  for  $1 \leq i \leq k$

# Number of Total Rows

$$\prod_{i=1}^{i=d} \sigma_i \leq \left( \frac{\|A - A_k\|_F^2 + \lambda(d - k)}{d - k} \right)^{d-k} (\|A\|_2^2 + \lambda)^k$$

↑
↑  
 small singular values                      large singular values

For  $\lambda = \frac{\|A - A_k\|_F^2}{k}$ ,  $\det(A^\top A + \lambda I_d) \leq \lambda^{d-k} e^k (\|A\|_2^2 + \lambda)^k$

$$\log \det(A^\top A + \lambda I_d) = O(k \log n) + d \log \lambda$$



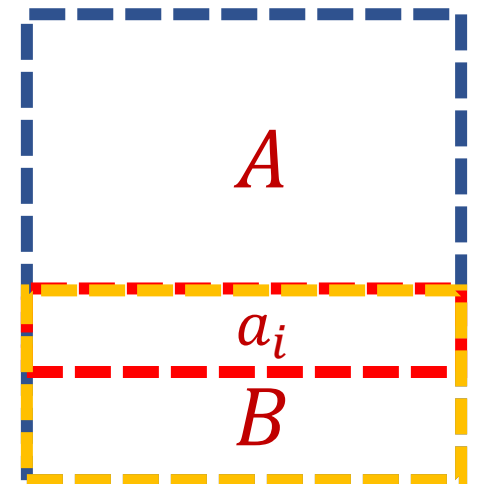
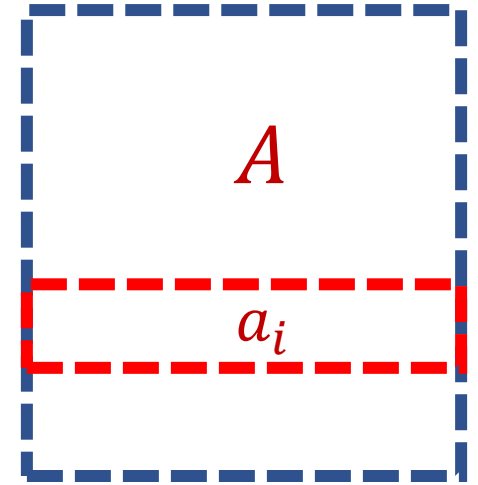
# Number of Total Rows

- ❖  $\det(A^\top A + \lambda I_d) \geq \lambda^d e^{\sum \tau_i / 2}$  [CMP16]
- ❖  $\frac{\sum \tau_i}{2} + d \log \lambda \leq \log \det(A^\top A + \lambda I_d)$

$$\text{❖ } \log \det(A^\top A + \lambda I_d) = O(k \log n) + d \log \lambda$$

# Template

- ✓ Issue #1: Compute  $\lambda = \frac{\|A - A_k\|_F^2}{k}$
- ✓ Issue #2: Bound  $\sum \tau_i$



# Low-Rank Approximation (Summary)

- ❖ Correctness from a similar argument
- ❖  $\sum \tau_i = O(k \log n)$
- ❖ Both sampling algorithms can run in *input-sparsity time*
  
- ❖ Sum of reverse online scores = sum of online scores
- ❖ Gives many space efficient *online* algorithms essentially for free!
- ❖ Improvements on online principal component analysis, column subset selection in [BLVZ19]

# Structural Results for Reverse Online Scores

- ❖ Reverse online leverage scores (for spectral approximation):  $\sum \tau_i = O(d \log \kappa)$  [CMP16]
- ❖ Reverse online ridge leverage scores (for low-rank approximation):  $\sum \tau_i = O(k \log \kappa)$
- ❖ Reverse online  $\ell_1$  sensitivities, online Lewis weights (for  $\ell_1$  subspace embedding):  $\sum \tau_i = O(d \log n \log \kappa)$
- ❖ Sum of reverse online scores = Sum of online scores!
- ❖ Our structural results essentially give our online results for free!

# Results: Sliding Window Model

- ❖  $O(d^2)$  space randomized sliding window algorithm for spectral sparsification (space optimal, up to lower order terms)
- ❖  $O(dk)$  space randomized sliding window algorithm for low-rank approximation (space optimal, up to lower order terms)
- ❖  $O(d^3)$  space randomized sliding window algorithm for  $\ell_1$  subspace embedding

# Results: Online Model

- ❖ Online algorithm for low-rank approximation that samples  $O(k)$  rows (space optimal, up to lower order terms)
- ❖ Online algorithm for row subset selection that samples  $O(k)$  rows (space optimal, up to lower order terms)
- ❖ Online algorithm for principal component analysis that embeds into a matrix with dimension  $O(k)$  (space optimal, up to lower order terms)
- ❖ Online algorithm for  $\ell_1$  subspace embedding that samples  $O(d^2)$  rows

# Results: Connections

- ❖ **Online cores**et: algorithm whose output can be used to approximate all prefixes of a stream
- ❖ Online coresets give deterministic sliding window algorithms!
- ❖  $O(d^2)$  space deterministic sliding window algorithm for spectral sparsification (space optimal, up to lower order terms)
- ❖  $O(dk)$  space deterministic sliding window algorithm for low-rank approximation (space optimal, up to lower order terms)
- ❖  $O(d^2)$  space deterministic sliding window algorithm for  $\ell_1$  subspace embedding (space optimal, up to lower order terms)





# Summary



- ❖  $O(n^3)$  space deterministic sliding window algorithm for spectral sparsification
- ❖  $O(n^2)$  space randomized sliding window algorithm for spectral sparsification
- ❖  $O(nk)$  space randomized sliding window algorithm for low rank approximation
- ❖ Results for  $\ell_1$  subspace embedding (with bit complexity assumptions), matrix multiplication on sliding windows
- ❖ Results for online principal component analysis, low rank approximation, column subset selection

# Reverse Online Leverage Scores

- ❖  $\det(A^\top A + \lambda I_n) \geq \lambda^n e^{\sum \tau_i / 2}$
- ❖  $\log \det(A^\top A + \lambda I_n) = O(k \log n)$
- ❖  $\sum \tau_i = O(k \log n)$
- ❖ Also gives a space efficient *online* algorithm for low-rank approximation!
- ❖ Can use slightly different estimator for  $\lambda = \frac{\|A - A_k\|_F^2}{k}$  [AN13]

# Reverse Online Leverage Scores

- ❖  $\det(A^\top A + \lambda I_n) \geq \lambda^n e^{\sum \tau_i / 2}$
- ❖  $\log \det(A^\top A + \lambda I_n) = O(k \log n)$
- ❖  $\sum \tau_i = O(k \log n)$
- ❖ Also gives a space efficient *online* algorithm for low-rank approximation!
- ❖ Can use slightly different estimator for  $\lambda = \frac{\|A - A_k\|_F^2}{k}$  [AN13]

# $\ell_1$ Spectral Approximation

- ❖ Given  $\epsilon > 0$  and  $A \in R^{W \times n}$ , find matrix  $M \in R^{m \times n}$  with  $m \ll W$ , such that for *every*  $x \in R^n$

$$(1 - \epsilon)\|Ax\|_1 \leq \|Mx\|_1 \leq (1 + \epsilon)\|Ax\|_1$$

- ❖ Robust to outliers, but unstable solution and possibly multiple solutions

# $\ell_1$ Leverage Scores

- ❖ Previous  $\ell_2$  leverage scores:  $\ell_i = \max \frac{\langle a_i, x \rangle^2}{\|Ax\|_2^2}$
- ❖  $\ell_1$  leverage scores:  $\ell_i = \max \frac{|\langle a_i, x \rangle|}{\|Ax\|}$
- ❖ Sample each row  $a_i$  with probability  $p_i \propto \ell_i$  gives  $\ell_1$  spectral approximation [DDHKM07]
- ❖ Bound the sum of the reverse online  $\ell_1$  leverage scores



# $\ell_1$ Leverage Scores

- ❖ Make nice assumptions: the entries of  $A$  and  $x$  are bounded integers
- ❖ Can show that if  $\|Ax\|_1$  increases by  $(1 + \epsilon)$ ,  $\|Ax\|_2^2$  must increase by  $\left(1 + \frac{\epsilon}{\text{poly}(n)}\right)$
- ❖ Can use deterministic algorithm to find these breakpoints
- ❖ Use separate instances of streaming  $\ell_1$  spectral approximation algorithm starting at each of these breakpoints [DDHKM07, CP15]

# Matrix Multiplication Lower Bounds

- ❖ Distributional INDEX:  $\{0,1\}^n \times [n]$
- ❖ Alice has string  $S \in \{0,1\}^n$  chosen uniformly at random
- ❖ Bob has index  $i \in [n]$  chosen uniformly at random and must output  $S[i]$  with probability  $\frac{2}{3}$
- ❖ Requires  $\Omega(n)$  bits of communication from Alice to Bob  
[MNSW98]



# Matrix Multiplication Lower Bounds

- ❖ Alice has  $S \in \{0,1\}^{n/c^2\epsilon^2}$
- ❖ Creates matrix  $M \in \{-c, c\}^{\frac{1}{c^2\epsilon^2} \times n}$  in the natural way (stuffs string into matrix by sign)
- ❖ Creates matrix  $A = [M \ E]$ , where  $E$  is  $c^2\epsilon^2 n$  instances of  $e_1, \dots, c^2\epsilon^2 n$  instances of  $e_{c^2/\epsilon^2}$
- ❖ If  $\|A^\top A - B^\top B\|_F \leq \epsilon \|A^\top A\|_F$ , then Bob can recover the signs of most entries in  $M$  and hence can recover most of the symbols of  $S$





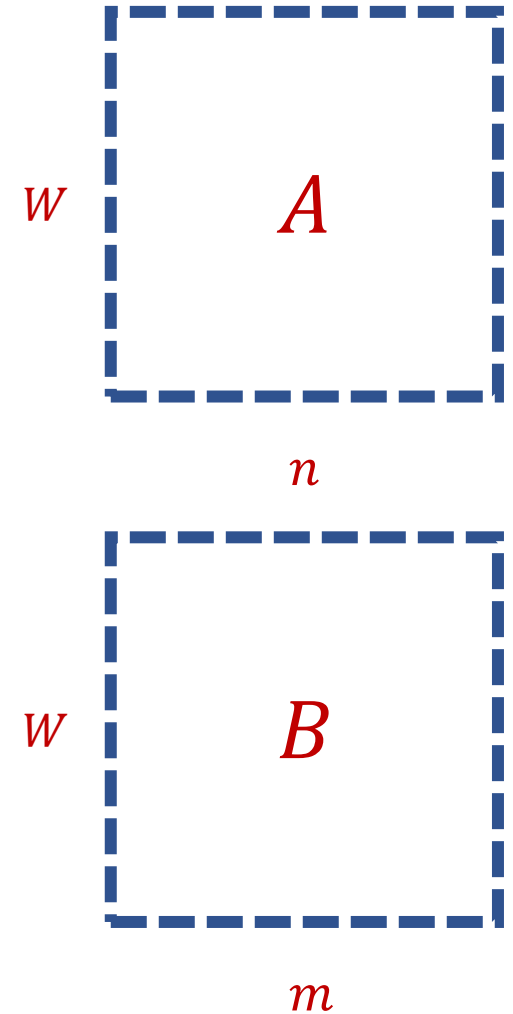
# Linear Algebra Background

- ❖ Vectors  $u, v \in R^n$
- ❖ Inner product:  $\langle u, v \rangle = \sum u_i v_i \in R$
- ❖ Outer product:  $u \otimes v = uv^T \in R^{n \times n}$

$$\begin{array}{cccc} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_n v_1 & u_n v_2 & \dots & u_n v_n \end{array}$$

# Linear Algebra Background

- ❖ Matrices:  $A \in R^{W \times n}$ ,  $B \in R^{W \times m}$
- ❖  $(A^T B)_{i,j} = \langle a_i, b_j \rangle$ , where  $a_i$  is the  $i^{\text{th}}$  column of  $A$  and  $b_j$  is the  $j^{\text{th}}$  column of  $B$ .
- ❖  $A^T B = \sum_{i=1}^W a_i \otimes b_i = \sum_{i=1}^W a_i^T b_i$



# Approximate Matrix Multiplication

- ❖ Vector norm:  $\|x\|_p = (x_1^p + x_2^p + \dots + x_n^p)^{\frac{1}{p}}$
- ❖ Frobenius norm:  $\|A\|_F = \sqrt{\sum A_{i,j}^2}$
- ❖ Matrices:  $A \in R^{W \times n}, B \in R^{W \times m}, W \gg m, n$
- ❖ Output  $A^T B$
  
- ❖ Can we find  $C \in R^{d \times n}, D \in R^{d \times m}, d \ll W$  such that  $C^T D \approx A^T B$ ?
- ❖ What does  $\approx$  mean?

# Approximate Matrix Multiplication

- ❖ Given  $\epsilon > 0$ , find  $C \in R^{d \times n}$ ,  $D \in R^{d \times m}$  such that

$$\|A^T B - C^T D\|_F \leq \epsilon \|A^T B\|_F$$

- ❖ **Information Retrieval:**  $A \in R^{W \times n}$  rows represent documents, columns represent occurrence of each word
- ❖ High entries of  $AA^T$  correspond to “similar” documents



# Approximate Matrix Multiplication (Offline)

## [DK01]

- ❖ Given  $\epsilon > 0$  and  $A \in R^{W \times n}$ ,  $W \gg n$ , find  $B \in R^{d \times n}$  such that

$$\|A^T A - B^T B\|_F \leq \epsilon \|A^T A\|_F$$

- ❖ **Intuition:** Large entries in  $A^T A$  come from large entries in  $A$
- ❖ Importance sampling: Sample row  $a_k$  of  $A$  proportional to its squared row norm
- ❖ Sample row  $a_k$  of  $A$  with probability  $p_k \propto \frac{\|a_k\|_2^2}{\|A\|_F^2}$
- ❖ Rescale each sampled row by  $\frac{1}{\sqrt{p_k}}$

# Approximate Matrix Multiplication (Offline)

- ❖ Analyze  $E[\|A^\top A - B^\top B\|_F^2]$
- ❖ Step 1: Show that  $B^\top B$  is an unbiased estimator:
- ❖  $E[B^\top B] = \sum p_k \left( \frac{1}{\sqrt{p_k}} a_k^\top \frac{1}{\sqrt{p_k}} a_k \right) = A^\top A$
- ❖ Step 2: Bound the variance of  $(B^\top B)_{i,j}$ :
- ❖  $\text{Var}[(B^\top B)_{i,j}] \leq \sum \frac{1}{p_k} (a_k^\top a_k)_{i,j}^2$

# Approximate Matrix Multiplication (Offline)

- ❖  $E[\|A^\top A - B^\top B\|_F^2] = E[\sum_{i,j} (A^\top A - B^\top B)_{i,j}^2]$
- ❖  $E[\|A^\top A - B^\top B\|_F^2] = \sum_{i,j} \text{Var}[(A^\top A - B^\top B)_{i,j}]$
- ❖  $E[\|A^\top A - B^\top B\|_F^2] \leq \sum_{i,j,k} \frac{1}{p_k} (a_k^\top a_k)_{i,j}^2 = \sum_k \frac{1}{p_k} \|a_k\|_2^4$
- ❖ For  $p_k = \frac{c \|a_k\|_2^2}{\|A\|_F^2}$ ,  $E[\|A^\top A - B^\top B\|_F^2] \leq \frac{1}{c} \|A\|_F^4$ .
- ❖  $\sum p_k = c := \frac{1}{\epsilon^2}$ , so total number of sampled rows is  $O\left(\frac{1}{\epsilon^2} \log n\right)$   
whp

# Approximate Matrix Multiplication

- ❖ **Goal:** Sample row  $a_i$  of  $A$  with probability  $p_i \propto \frac{\|a_i\|_2^2}{\|A\|_F^2}$
- ❖ See  $a_i$  in the sliding window model, can compute  $\|a_i\|_2^2$
- ❖ Cannot compute  $\|A\|_F^2$  without seeing all the rows
- ❖ How would we do matrix multiplication in the streaming model?



# Approximate Matrix Multiplication

- ❖ How would we do matrix multiplication in the streaming model?
- ❖ Track  $\|A\|_F^2$
- ❖ Suppose we have sampled row  $a_i$  of  $A$  with probability  $p_i \propto \frac{\|a_i\|_2^2}{\|A\|_F^2}$
- ❖ New row arrives  $a_t$ :  $\|A\|_F^2$  increases by  $\|a_t\|_2^2$
- ❖ What do we do with  $a_i$ ?
- ❖ **Downsample**: keep  $a_i$  with probability  $\frac{\|A\|_F^2}{\|A\|_F^2 + \|a_t\|_2^2}$
- ❖ Sampled  $a_i$  with probability  $p_i \propto \frac{\|a_i\|_2^2}{\|A\|_F^2 + \|a_t\|_2^2}$

# Approximate Matrix Multiplication

- ❖ Note it suffices to have  $\hat{A}$  a 2-approximation of  $\|A\|_F^2$
- ❖ Why? Sample row  $a_i$  of  $A$  with probability  $p_i \propto \frac{2\|a_i\|_2^2}{\hat{A}}$
- ❖ Frobenius norm is *smooth*
- ❖ Use smooth histogram to maintain  $\hat{A}$



- ❖ Smooth histogram for Frobenius norm
- ❖ Separate instance of matrix multiplication streaming algorithm for each instance tracking the Frobenius norm

# Approximate Matrix Multiplication

- ❖ Total space:  $O\left(\frac{1}{\epsilon^2} \log n\right)$  rows  $\rightarrow O\left(\frac{n}{\epsilon^2} \log^2 n\right)$  bits of space
- ❖ Can decrease to  $O\left(\frac{n}{\epsilon^2} \log n \left(\log \log n + \log \frac{1}{\epsilon}\right)\right)$  with bit representation tricks
- ❖ Also give  $\Omega\left(\frac{n}{\epsilon^2} \log n\right)$  space lower bound

Questions?



# Spectral Approximation (Offline)

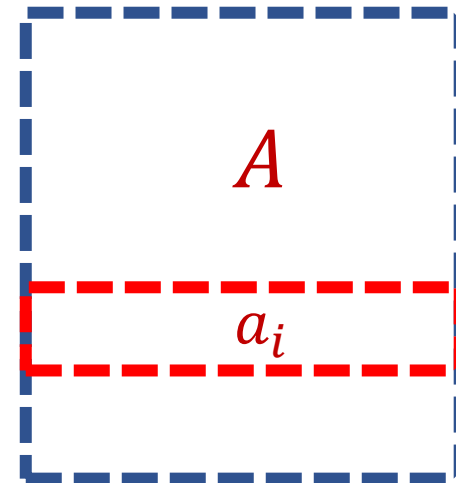
- ❖ Spectral approximation: Given  $\epsilon > 0$  and  $A \in R^{W \times n}$ , find matrix  $M \in R^{m \times n}$  with  $m \ll W$ , such that for *every*  $x \in R^n$ ,

$$(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

- ❖ How would we do this offline? Hint: fundamental tool of dimensionality reduction
- ❖ Although Johnson-Lindenstrauss reduces the number of rows and sparse JL can preserve sparsity, we want to focus on sampling

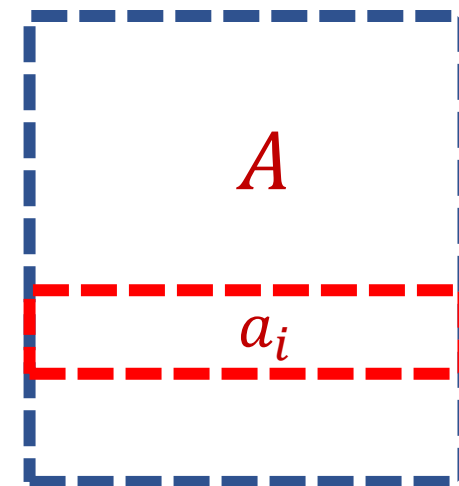
# Linear Algebra Background

- ❖ Singular Value Decomposition (SVD):  $A = U\Sigma V^T \in R^{W \times n}$ 
  - ❖  $U \in R^{W \times W}$  is an orthonormal matrix (rows, columns orthonormal)
  - ❖  $\Sigma \in R^{W \times n}$  is a rectangular diagonal matrix with non-negative entries
  - ❖  $V \in R^{n \times n}$  is an orthonormal matrix (rows, columns orthonormal)
- ❖  $\|u_i\|_2^2$  are the *leverage scores* of  $A$  (in this case of row  $a_i$ )
- ❖ Intuition: how “unique” a row is (recall importance sampling)
- ❖  $\ell_i = \max_x \frac{\langle a_i, x \rangle^2}{\|Ax\|_2^2} = \|u_i\|_2^2 = a_i(A^T A)^{-1}a_i^T$
- ❖  $\sum \ell_i = n$



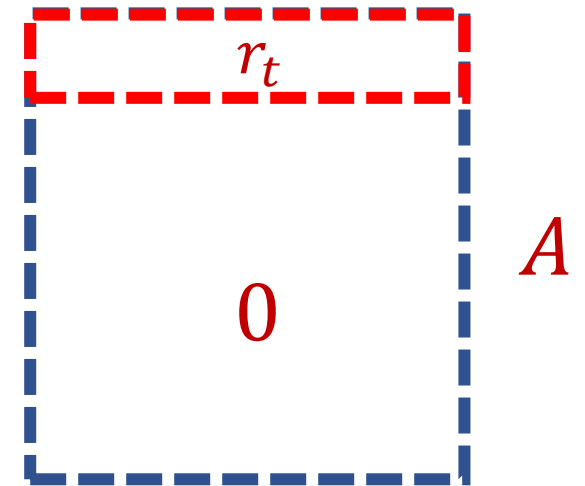
# Spectral Approximation (Offline)

- ❖ Sample each row  $a_i$  with probability  $p_i \propto \ell_i = a_i(A^\top A)^{-1}a_i^\top$
- ❖ Outputs a matrix  $M \in R^{m \times n}$  such that  $(1 - \epsilon)\|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon)\|Ax\|_2$  for all  $x \in R^n$  [DMM06, SS08]
- ❖  $\sum \ell_i = n$ , so the total number of rows sampled is  $\propto \tilde{O}(n)$
- ❖ Leverage scores are monotonic with more rows, so the offline approach can be adapted to streaming through **downsampling**



# Spectral Approximation (Sliding Window)

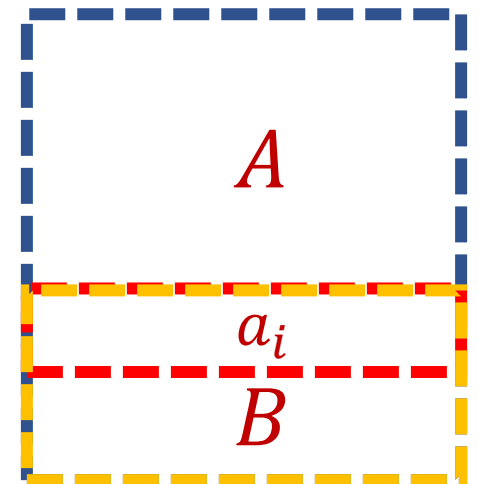
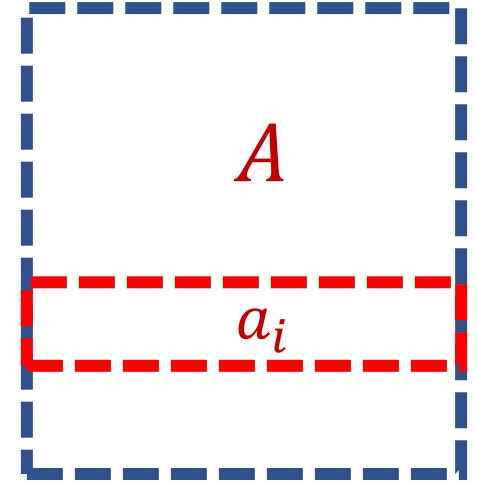
- ❖ Consider the sliding window model: we see rows  $r_1, r_2, \dots$
- ❖ Leverage score of  $r_t$  tells us  $r_t$  is not important, so we do not sample  $r_t$
- ❖ Stream proceeds: All rows before  $r_t$  expire, new rows are all zeros
- ❖ Cannot possibly get approximation of  $A$  without storing  $r_t$
- ❖ This implies we should *always* store the most recent row!
- ❖ Need a new sense of importance accounting for both **uniqueness** AND **recency** of a row





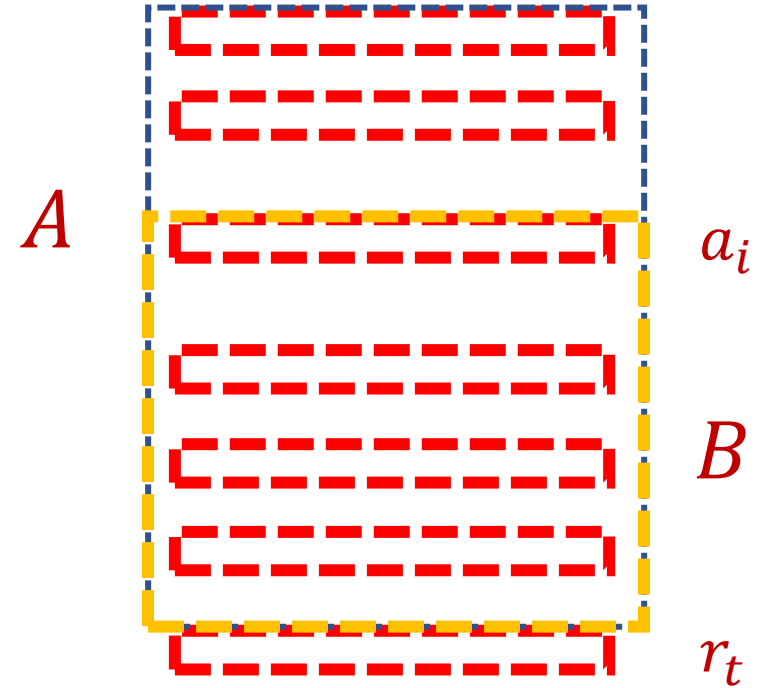
# Reverse Online Leverage Scores

- ❖ Leverage score of row  $a_i$  is  $\ell_i = a_i(A^\top A)^{-1}a_i^\top$
- ❖ Rows before  $a_i$  might be deleted so they shouldn't count towards the importance of  $a_i$
- ❖ Reverse online leverage score of row  $a_i$  is  $\tau_i = a_i(B^\top B)^{-1}a_i^\top$  where  $B$  are the rows after  $a_i$



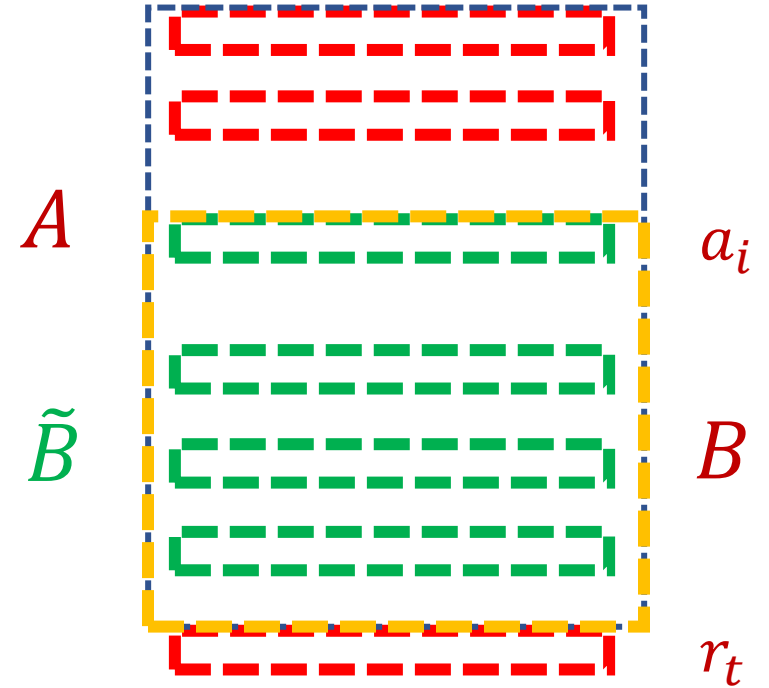
# Correctness

- ❖ Correctness: Show an invariant that each row  $a_i$  is sampled with probability  $\propto$  *final* reverse online leverage score
- ❖ Let  $B$  be the rows after  $a_i$  before row  $r_t$
- ❖ Suppose before the arrival of row  $r_t$ , row  $a_i$  has been sampled with probability  $p_i$ , where  $c_1 \tau_B(a_i) \leq p_i \leq c_2 \tau_B(a_i)$



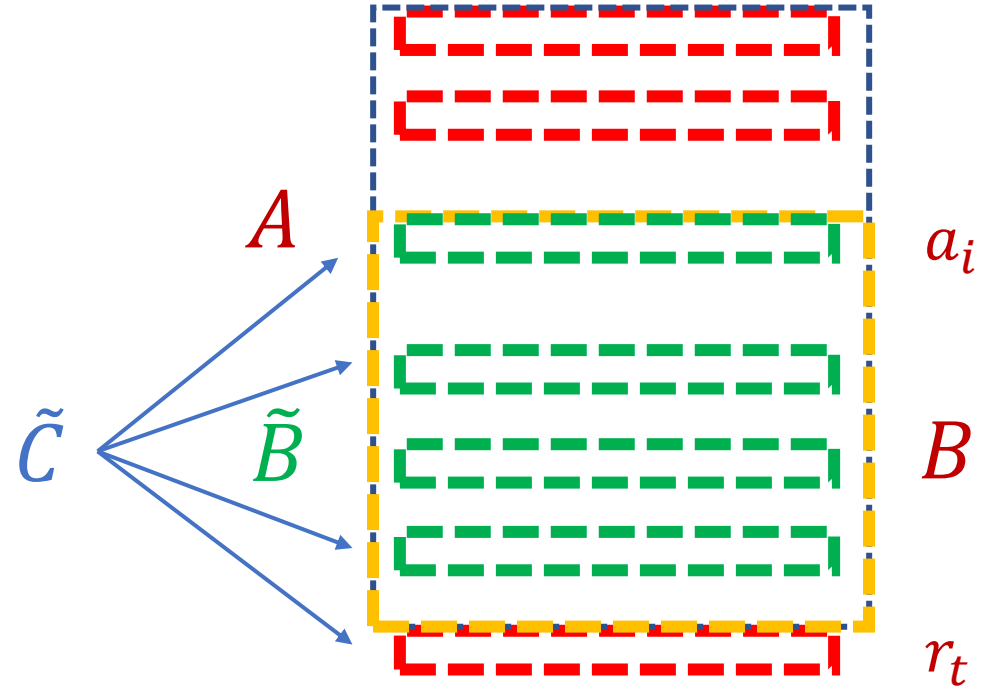
# Correctness

- ❖ Suppose before the arrival of row  $r_t$ , row  $a_i$  has been sampled with probability  $p_i$ , where  $c_1\tau_B(a_i) \leq p_i \leq c_2\tau_B(a_i)$
- ❖  $(1 - \epsilon)B^\top B \preceq \tilde{B}^\top \tilde{B} \preceq (1 + \epsilon)B^\top B$   
[DMM06, SS08]



# Correctness

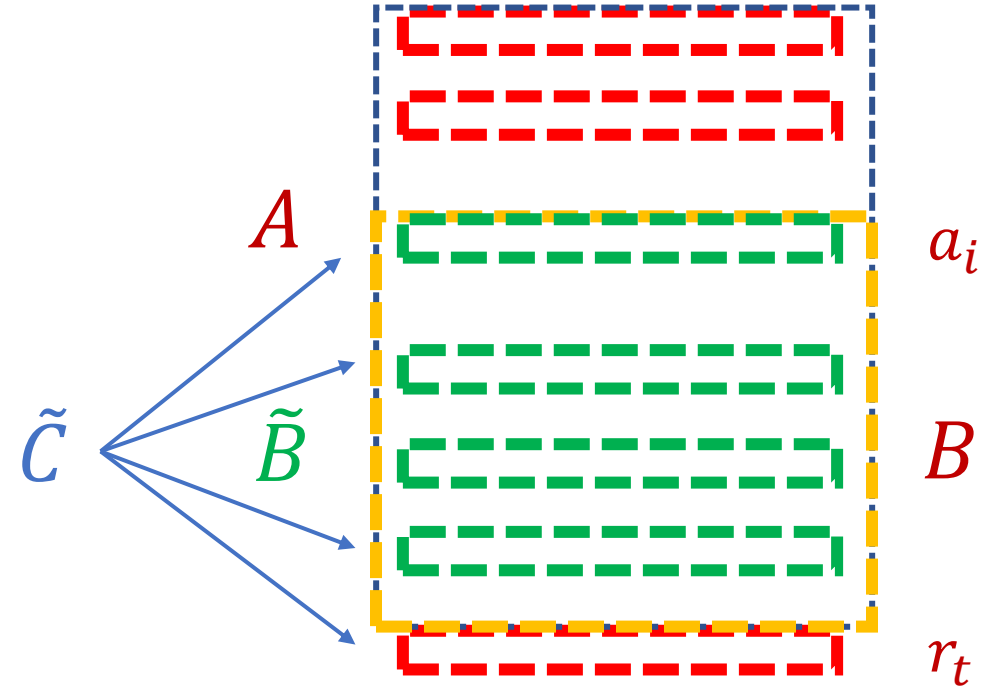
- ❖ Suppose before the arrival of row  $r_t$ , row  $a_i$  has been sampled with probability  $p_i$ , where  $c_1 \tau_B(a_i) \leq p_i \leq c_2 \tau_B(a_i)$
- ❖  $(1 - \epsilon)B^\top B \preceq \tilde{B}^\top \tilde{B} \preceq (1 + \epsilon)B^\top B$   
[DMM06, SS08]
- ❖ Let  $\tilde{C}$  be  $\tilde{B}$  appended by  $r_t$
- ❖  $a_i$  remains with probability  $\propto \tau_{\tilde{C}}\left(\frac{a_i}{\sqrt{p_i}}\right)$



# Correctness

- ❖  $a_i$  remains with probability  $\propto \tau_{\tilde{C}} \left( \frac{a_i}{\sqrt{p_i}} \right)$
- ❖ Reverse online leverage score:

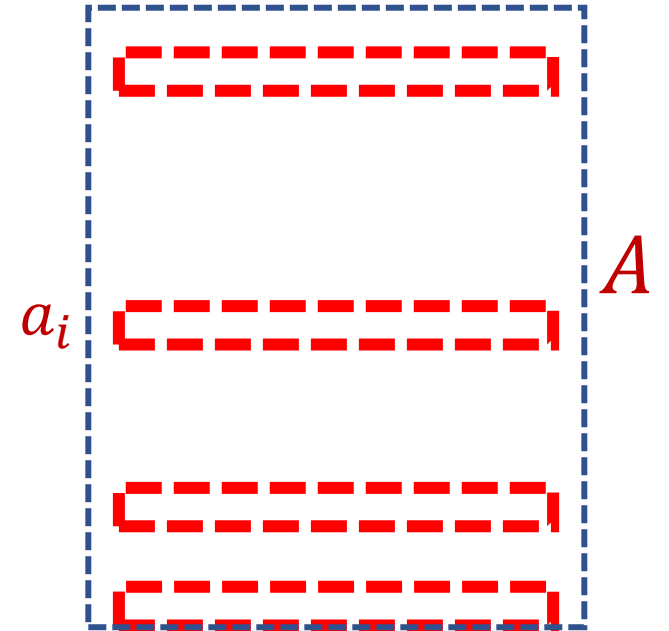
$$\left( \frac{a_i}{\sqrt{p_i}} \right) (\tilde{C}^\top \tilde{C})^{-1} \left( \frac{a_i}{\sqrt{p_i}} \right)^\top = \left( \frac{a_i}{\sqrt{p_i}} \right) (\tilde{B}^\top \tilde{B} + r_t^\top r_t)^{-1} \left( \frac{a_i}{\sqrt{p_i}} \right)^\top$$



- ❖ Recall  $(1 - \epsilon)B^\top B \preceq \tilde{B}^\top \tilde{B} \preceq (1 + \epsilon)B^\top B$ ,  
so  $(1 - \epsilon)C^\top C \preceq \tilde{C}^\top \tilde{C} \preceq (1 + \epsilon)C^\top C$
- ❖  $a_i$  survives w.p.  $c_1 \tau_C(a_i) \leq p_i \leq c_2 \tau_C(a_i)$

# Algorithm

- ❖ Correctness: Show an invariant that each row  $a_i$  is sampled with probability  $\propto$  *final* reverse online leverage score
- ❖ By monotonicity,  $a_i$  is sampled with probability  $\propto$  leverage score
- ❖ Outputs a matrix  $M \in R^{m \times n}$  such that  $(1 - \epsilon) \|Ax\|_2 \leq \|Mx\|_2 \leq (1 + \epsilon) \|Ax\|_2$  for all  $x \in R^n$  [DMM06, SS08]
- ❖ Space? Must bound  $\sum \tau_i$



# Reverse Online Leverage Scores

- ❖ Online algorithm: see rows sequentially and irrevocably store or discard row, output spectral approximation at the end
- ❖ Sum of reverse online leverage scores = sum of online leverage scores  
[CMP16]
- ❖  $\sum \tau_i = \tilde{O}\left(\frac{1}{\epsilon^2} n\right) \rightarrow \tilde{O}\left(\frac{1}{\epsilon^2} n^2\right)$  space algorithm

# Results

- ❖ Smooth histogram does not work for: vector induced  $p$  norms, generalized regression, low-rank approximation
- ❖ (Vector induced  $p$  norm:  $\|A\|_p = \max\|Ax\|_p$  for  $\|x\|_p = 1$ )

Problem	Space	Reference
Deterministic Spectral Approximation	$\tilde{O}\left(\frac{n^3}{\varepsilon}\right)$	<a href="#">Theorem 1.1</a>
Spectral Approximation	$\tilde{\Theta}\left(\frac{n^2}{\varepsilon^2}\right)$	<a href="#">Theorem 4.5</a>
Rank $k$ Approximation	$\tilde{\Theta}\left(\frac{nk}{\varepsilon^2}\right)$	<a href="#">Theorem 5.8</a>
Online Rank $k$ Approximation	$\tilde{\Theta}\left(\frac{nk}{\varepsilon^2}\right)$	<a href="#">Theorem 6.4</a>
Covariance Matrix Approximation	$\tilde{\Theta}\left(\frac{n}{\varepsilon^2}\right)$	<a href="#">Theorem 6.20</a> , <a href="#">Theorem 6.24</a>



# Results

- ❖ If the entries of  $A$  and  $x$  are bounded integers,  $O\left(\text{poly}\left(n, \frac{1}{\epsilon}\right)\right)$  space algorithm for  $\ell_1$  spectral approximation:

$$(1 - \epsilon)\|Ax\|_1 \leq \|Mx\|_1 \leq (1 + \epsilon)\|Ax\|_1$$

- ❖ Algorithms can be slightly modified to run in *input sparsity time*
  - ❖ Only require constant factor approximation to reverse online leverage score
  - ❖ Use sparse JL for subspace embedding

# Low-Rank Approximation (Offline)

- ❖ SVD:  $A = U\Sigma V^T$  with singular values  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$
- ❖ Let  $\Sigma_k$  be the matrix with diagonal entries  $\sigma_1, \dots, \sigma_k$
- ❖  $M = U\Sigma_k V^T$  is *optimal* solution

# Streaming Model

- ❖ **Input:** Elements of an underlying data set  $S$ , which arrives sequentially
- ❖ **Output:** Evaluation (or approximation) of a given function in space *sublinear* in the size of the input
  - ❖ Quantiles, heavy-hitters, norm estimation, distinct elements, sampling
  - ❖ Matchings, triangle counting, spanners, sparsifiers, densest subgraph,...
  - ❖ Minimum enclosing ball, clustering, facility location, volume maximization,...
  - ❖ Numerical linear algebra (matrix multiplication, spectral approximation,...)
  - ❖ Submodular optimization
  - ❖ Strings (pattern matching, periodicity, distance, palindromic detection,...)
  - ❖ Codeword testing

# Initial Approach (Smooth PSD Histogram)

- ❖ Recall smooth function: If  $f(A)$  is a “good” approximation to  $f(B)$ , then  $f(A \cup C)$  will always be a “good” approximation to  $f(B \cup C)$ .
- ❖ Monotonic, polynomially bounded, all properties of real values...
- ❖ Use partial (Loewner) ordering on PSD matrices
  - ❖ If matrix  $A \in R^{r \times n}$  is a submatrix of  $B \in R^{s \times n}$ , then  $A^T A \preceq B^T B$
- ❖ The singular values of  $A^T A$  are respectively at most those of  $B^T B$
- ❖ Have “monotonicity”, what about smoothness?