Testing Matrix Rank, Optimally

Hongyang Zhang, CMU

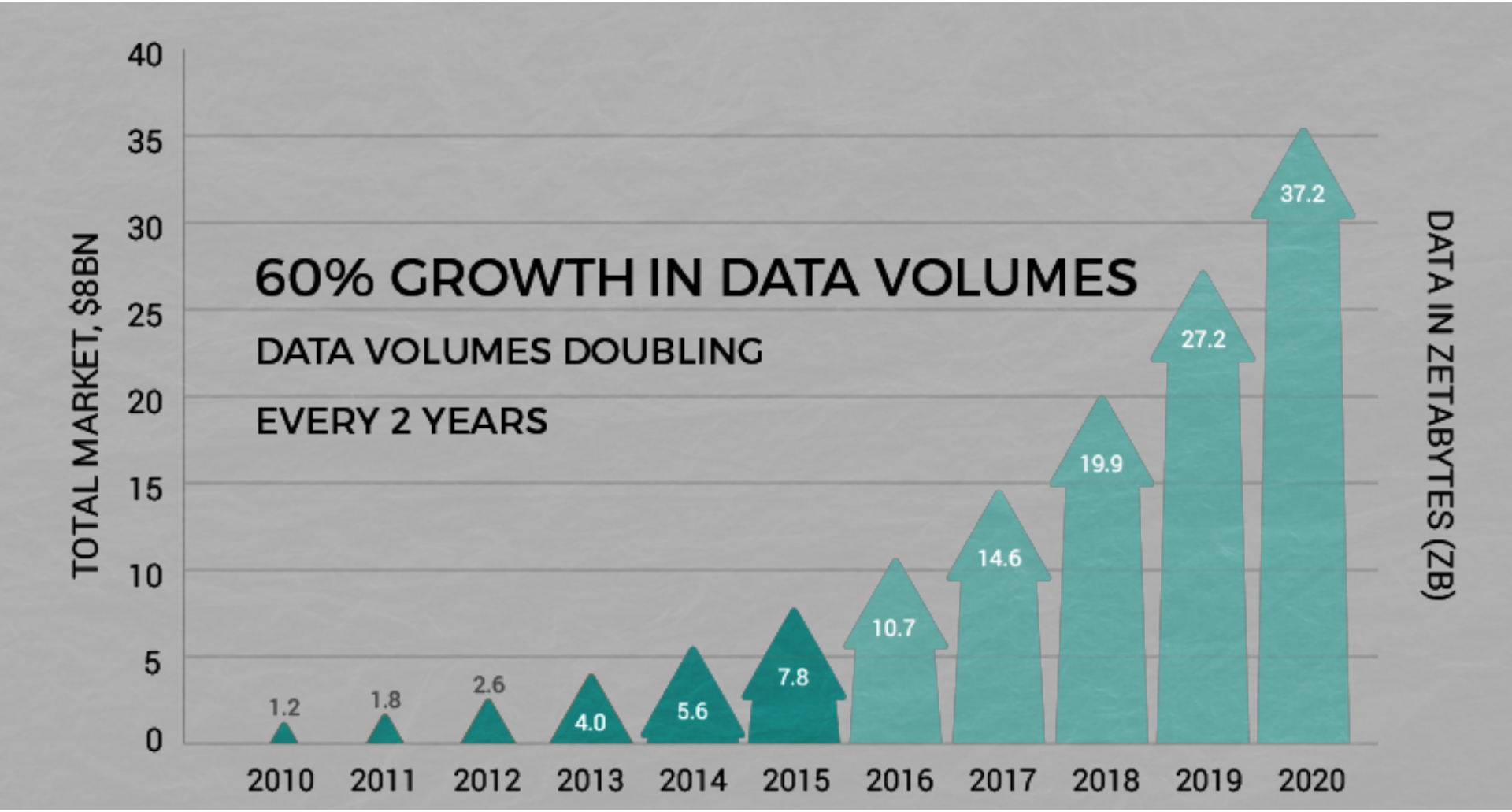
Nina Balcan (CMU)

Yi Li (NTU)

David P. Woodruff (CMU)

Era of Big Data

Era of Big Data



*Zettabyte = 35, 000,000,000,000,000,000,000 bytes

Netflix Challenge (low-rank data)

	8	8	1	1
	8	8	1	1
	4	4	2	2
?	4	4	2	2
	1	1	4	4
	1	1	4	4
	0	0	8	8
	0	0	8	8

The user-movie matrix is of low rank *r*.

Need to know *r* in many applications, e.g., in matrix completion.

Question:

- What is the true rank given a few observations? (Testing of Rank)

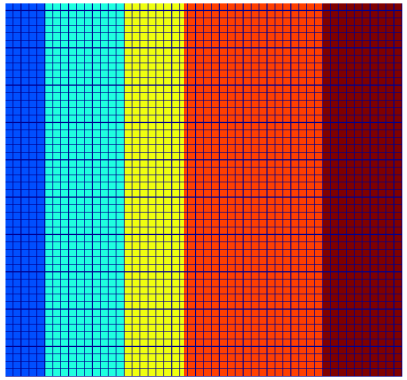
Outline

- Motivation and examples
- **Our settings**
- Our algorithms
- Our hardness results
- Conclusions

Property Testing of Rank

Goal: Design non-adaptive sampling scheme to distinguish H_0 from H_1
with sample complexity independent of size of matrix *n*, or even optimally

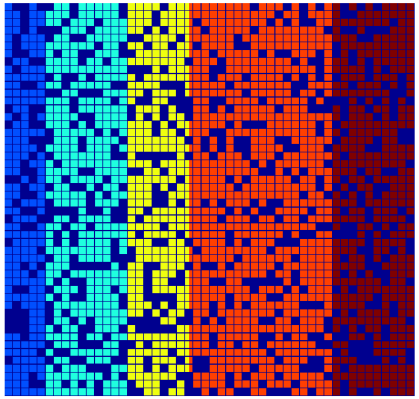
$\boldsymbol{H_0}$:



$\boldsymbol{rank(A) \leq d}$

v.s.

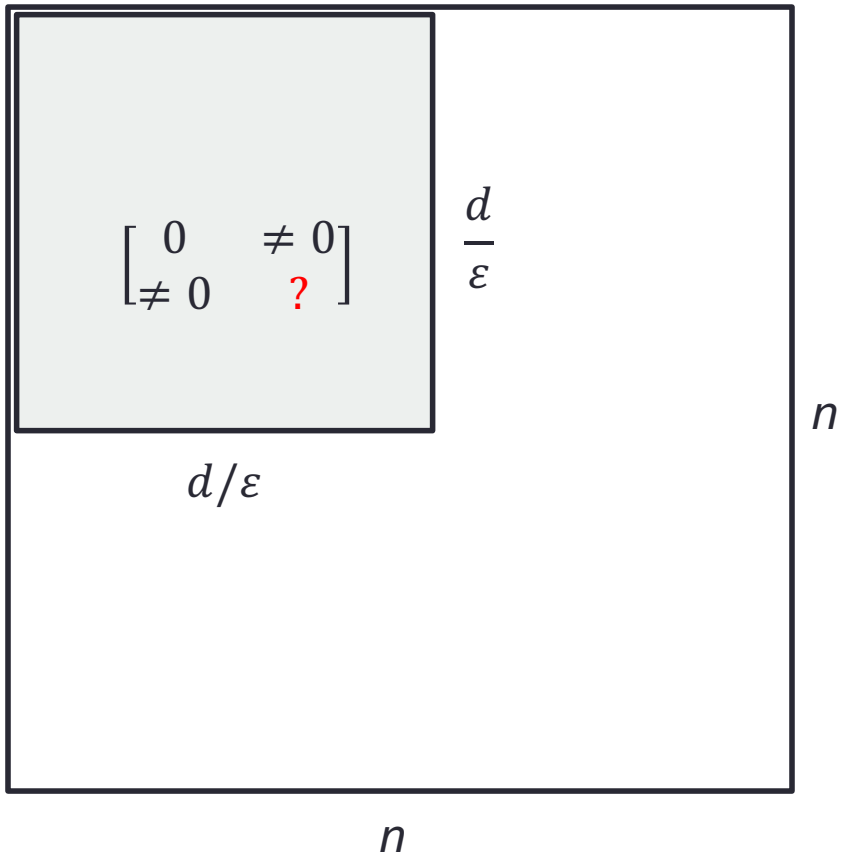
$\boldsymbol{H_1}$:



ε-far from $rank(A) \leq d$:

One needs to change at least an ε-fraction of entries so that the matrix becomes rank-*d*

Previous Work



Algorithm [KS'03]:

- Randomly permute the row and column indices of the matrix
- Sample top $O(\frac{d}{\varepsilon})\times O(\frac{d}{\varepsilon})$ submatrix X
- If $rank(X) \leq d$, output "H_0"; otherwise, output "H_1"

Sample complexity: $O(\frac{d^2}{\varepsilon^2})$

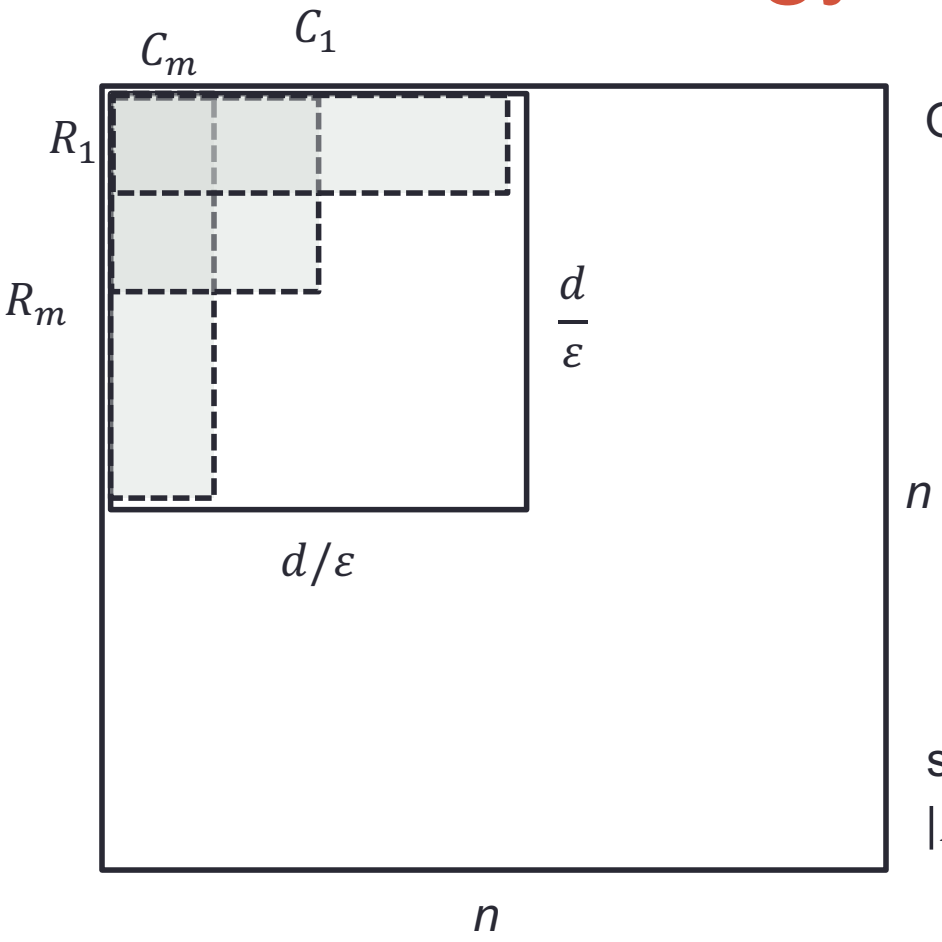
Problem of the algorithm (d=1, $A \in H_1$)

[KS'03] Property testing of data dimensionality, SODA '03

Outline

- Motivation and examples
- Our settings
- **Our algorithms**
- Our hardness results
- Conclusions

Our Methodology



Our algorithm:

- Randomly permute the row and column indices of the matrix
- Sample the block region $\mathcal{Q}$
- $r := \min_{A_{[d/\varepsilon]\times[d/\varepsilon]\setminus Q}} rank(A_{[d/\varepsilon]\times[d/\varepsilon]})$
- If $r \le d$, output "H_0"; otherwise, output "H_1"

$R_1 \subseteq \cdots \subseteq R_m,\ C_1 \supseteq \cdots \supseteq C_m$

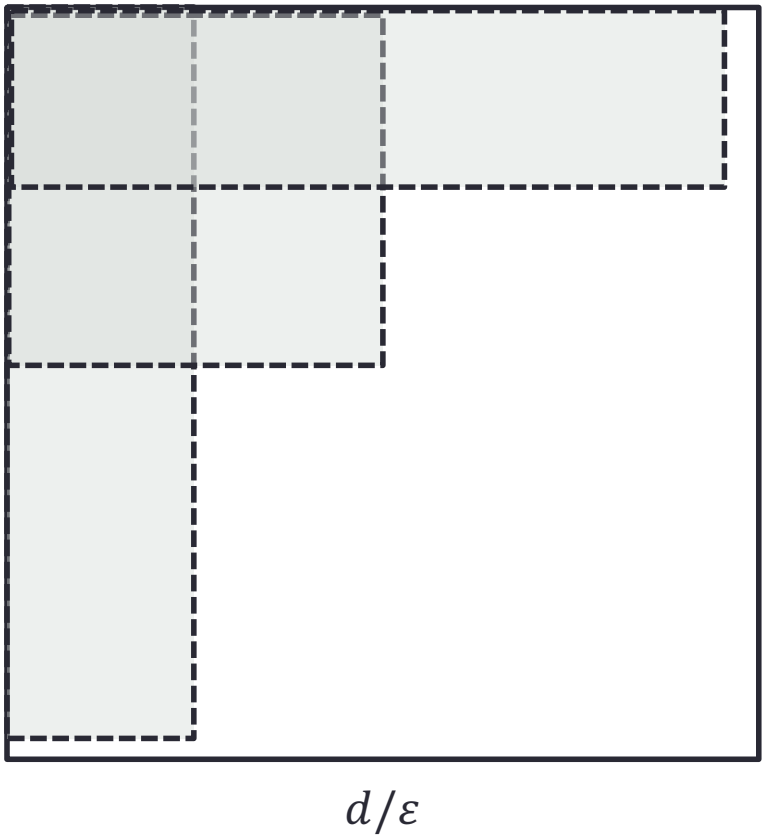
such that $m = \lceil \log(1/\varepsilon) \rceil$

$|R_i| = \tilde{O}(d2^i),\ |C_i| = \tilde{O}(d/(\varepsilon 2^i))$

Sample complexity: $\tilde{O}(\frac{d^2}{\varepsilon})$

Why does the Algorithm Work?

Proof Idea: Starting from an empty-by-empty matrix, augment the matrix until finding a $(d+1)\times(d+1)$ full-rank matrix



- $r := \min_{A_{[d/\varepsilon]\times[d/\varepsilon]\setminus Q}} rank(A_{[d/\varepsilon]\times[d/\varepsilon]})$
- If $r \le d$, output "H_0"; otherwise, output "H_1"

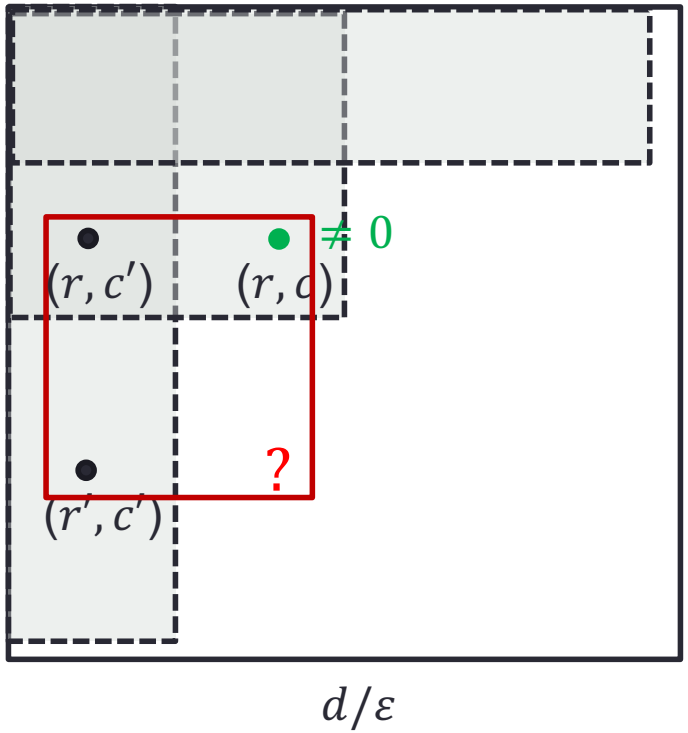
When $A \in H_0$, $r \le d$

Only need to show when $A \in H_1$, $r > d$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations. *(Let's say d=1)*

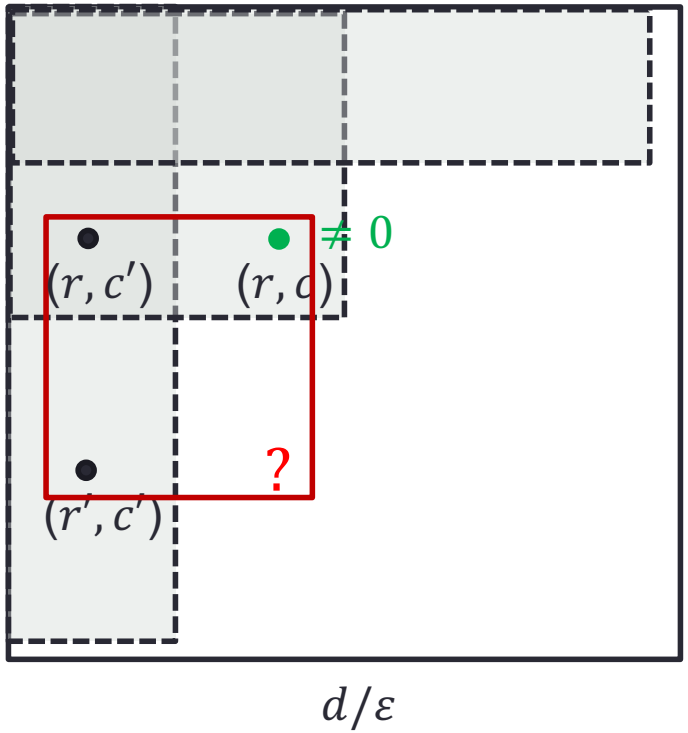
$$\begin{bmatrix} A_{r,c'} & A_{r,c} \\ A_{r',c'} & ? \end{bmatrix} = \begin{bmatrix} 0 & \neq 0 \\ A_{r',c'} & ? \end{bmatrix}$$

Case I: $A_{r,c'} = 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

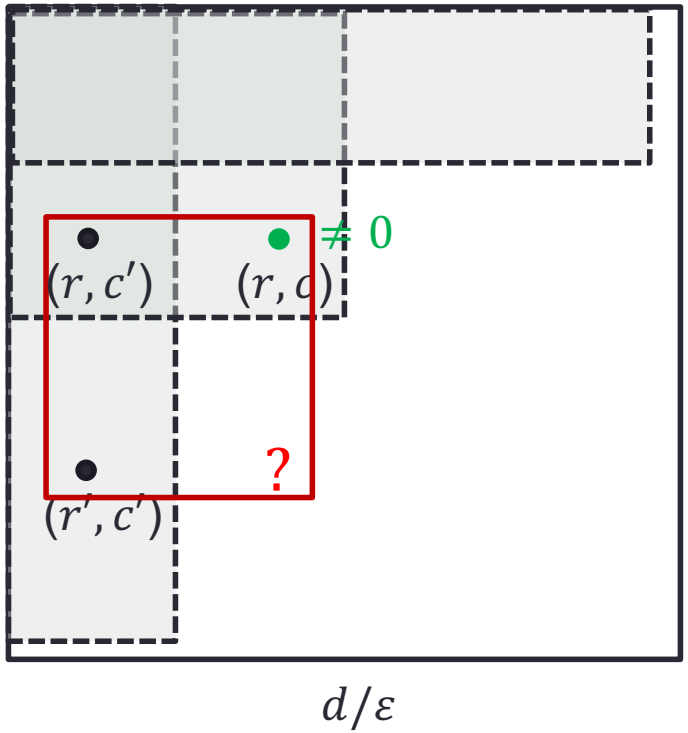
$$\begin{bmatrix} A_{r,c'} & A_{r,c} \\ A_{r',c'} & ? \end{bmatrix} = \begin{bmatrix} 0 & \neq 0 \\ \neq 0 & ? \end{bmatrix}$$

Case I: $A_{r,c'} = 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

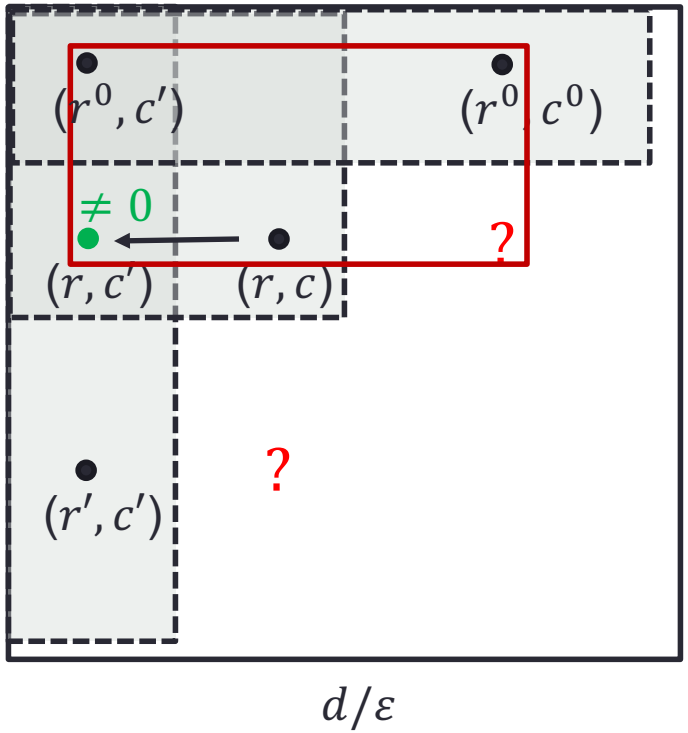
$$\begin{bmatrix} A_{r,c'} & A_{r,c} \\ A_{r',c'} & ? \end{bmatrix} = \begin{bmatrix} \neq 0 & A_{r,c} \\ A_{r',c'} & ? \end{bmatrix}$$

Case II: $A_{r,c'} \neq 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

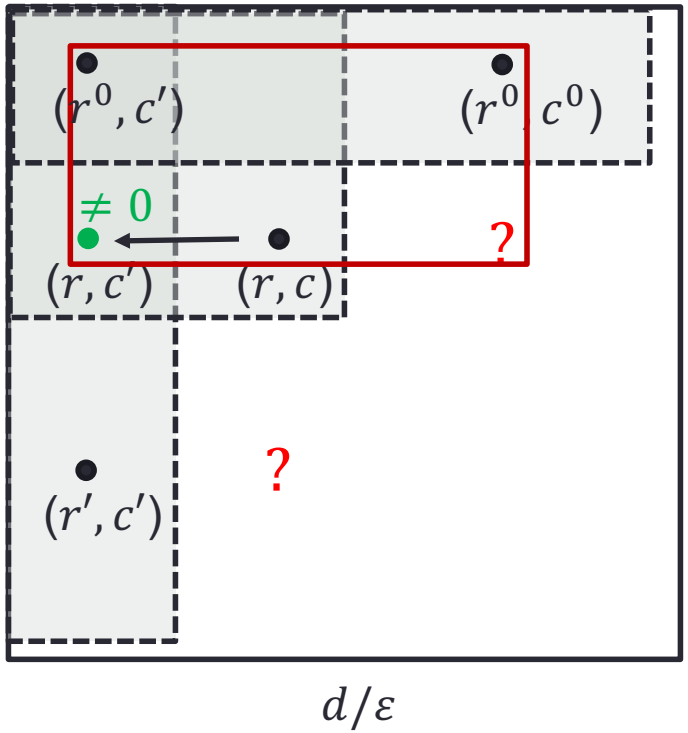
$$\begin{bmatrix} A_{r^0,c'} & A_{r^0,c^0} \\ A_{r,c'} & ? \end{bmatrix} = \begin{bmatrix} 0 & A_{r^0,c^0} \\ \neq 0 & ? \end{bmatrix}$$

Case I: $A_{r^0,c'} = 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

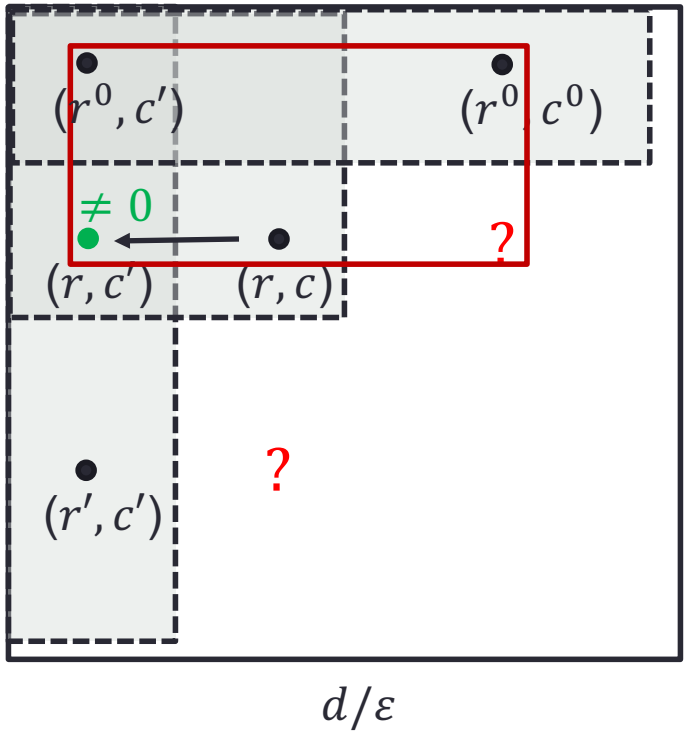
$$\begin{bmatrix} A_{r^0,c'} & A_{r^0,c^0} \\ A_{r,c'} & ? \end{bmatrix} = \begin{bmatrix} 0 & \neq 0 \\ \neq 0 & ? \end{bmatrix}$$

Case I: $A_{r^0,c'} = 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.



Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

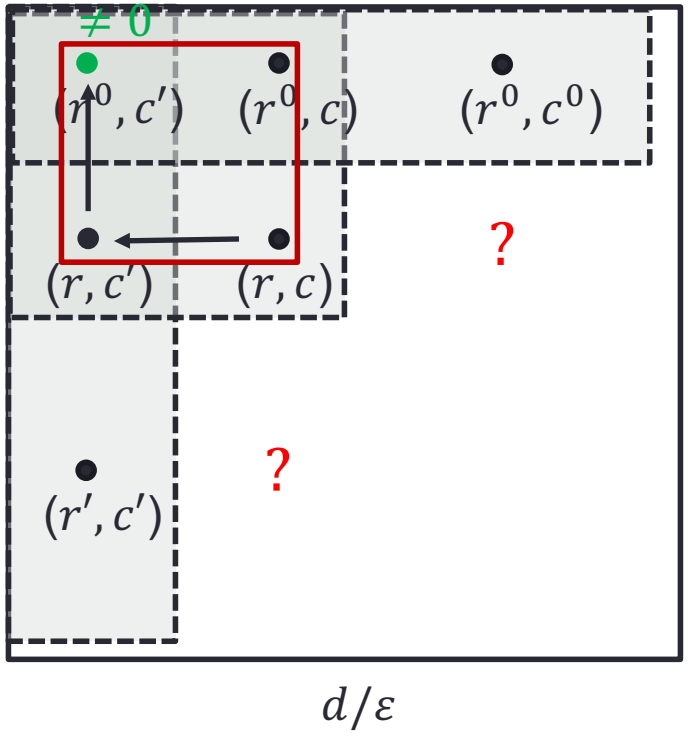
$$\begin{bmatrix} A_{r^0,c'} & A_{r^0,c^0} \\ A_{r,c'} & ? \end{bmatrix} = \begin{bmatrix} \neq 0 & A_{r^0,c^0} \\ A_{r,c'} & ? \end{bmatrix}$$

Case II: $A_{r^0,c'} \neq 0$

Proof Sketch

Oracle Lemma:

For any t, we can augment any $t \times t$ full-rank matrix to a $(t+1)\times(t+1)$ full-rank matrix by an augmentation entry in the sampled region, as long as $A \in H_1$ and $t \leq d$.

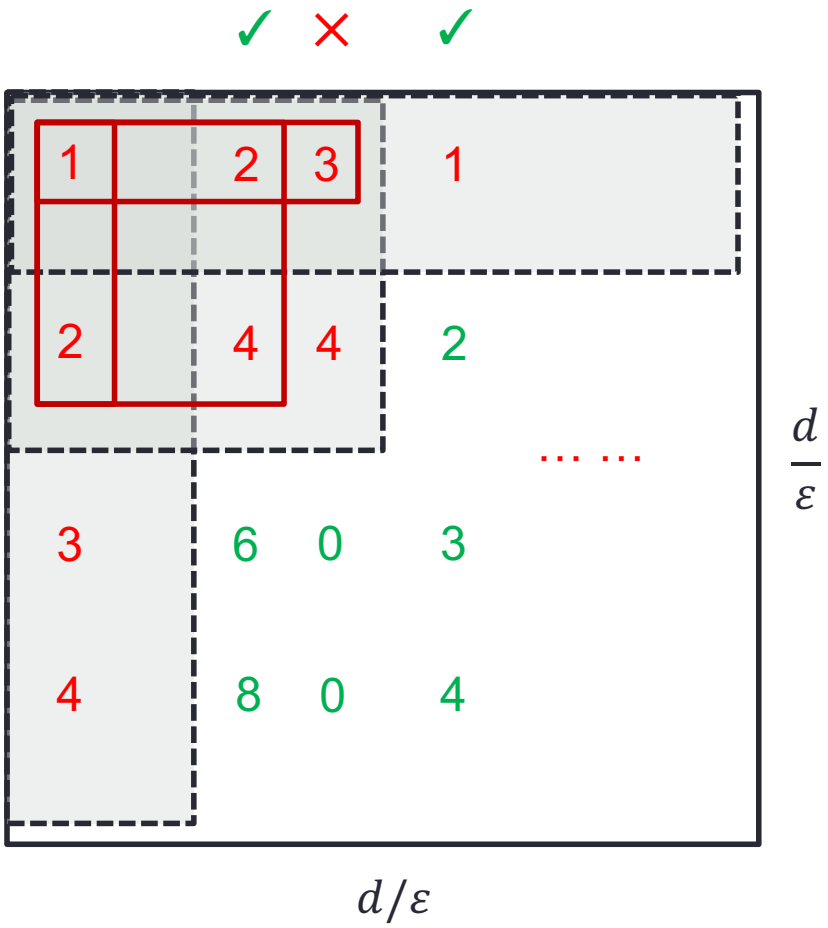


Challenge: Though the oracle lemma guarantees we find a $(t+1)\times(t+1)$ full-rank matrix from hindsight, the algorithm may not know it is full-rank due to unobservations.

$$\begin{bmatrix} A_{r^0,c'} & A_{r^0,c} \\ A_{r,c'} & A_{r,c} \end{bmatrix}$$

fully observed

A Polynomial-Time Algorithm



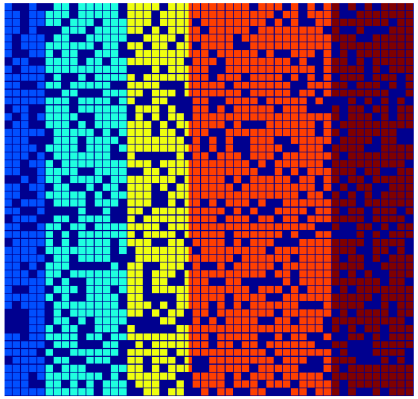
Our algorithm:

- Randomly permute the row and column indices of the matrix
- Sample the block region $\mathcal{Q}$
- $r := \min_{A_{[\mathrm{d}/\varepsilon]\times[\mathrm{d}/\varepsilon]\setminus\mathrm{Q}}} rank(A_{[\mathrm{d}/\varepsilon]\times[\mathrm{d}/\varepsilon]})$
- If $r \leq d$, output "H_0"; otherwise, output "H_1"

Other Extensions

Theorem (Stable Rank Upper Bounds):

There is an algorithm which tests the stable rank by a constant success probability with $\tilde{O}(d^3/\varepsilon^4)$ samples.



$$\mathrm{srank}(A) = \frac{\|A\|_F^2}{\sigma_1^2(A)}$$

stable version of matrix rank

Outline

- Motivation and examples
- Our settings
- Our algorithms
- **Our hardness results**
- Conclusions

Hardness --- Testing Matrix Rank

Our positive result: $\tilde{O}\left(\frac{d^2}{\varepsilon}\right)$ samples

Theorem (Lower Bounds):

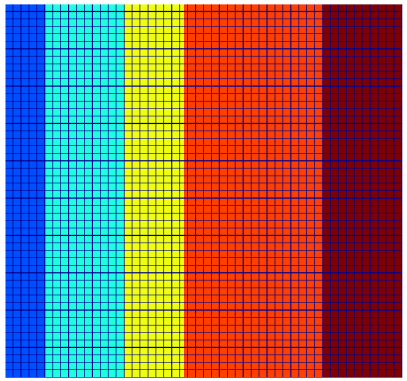
Any non-adaptive algorithm with constant success probability requires at least $\widetilde{\Omega}(d^2/\varepsilon)$ samples over reals and finite fields.

$\boldsymbol{U, V \sim \mathcal{G}(n, d)}$ $\quad$ $\boldsymbol{G \sim \mathcal{G}(n, n)}$

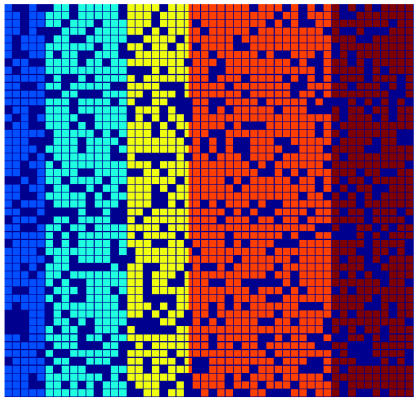
d_{TV} distance is a constant with small samples

$\boldsymbol{UV^T}$ $\quad$ $\boldsymbol{UV^T + n^{-14}G}$

$\boldsymbol{H_0}$:



$\boldsymbol{H_1}$:

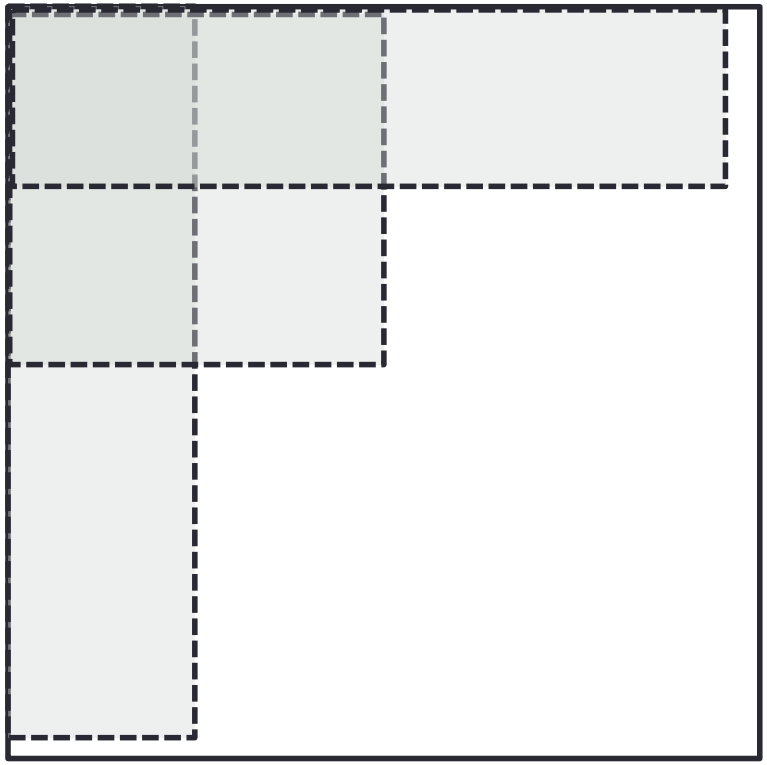


Outline

- Motivation and examples
- Our settings
- Our algorithms
- Our hardness results
- **Conclusions**

Conclusions

- Property testing of rank
  - Polynomial-time algorithm
  - Rebasing argument
  - Sample complexity $\tilde{O}(d^2/\varepsilon)$
  - Other extensions
- Hardness results
  - Sample complexity $\widetilde{\Omega}(d^2/\varepsilon)$



Thank You

Structure

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & \cdots & 0 & \neq 0 \\ 0 & 0 & \cdots & 0 & \cdots & \neq 0 & ? \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & \neq 0 & \cdots & ? & ? \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ 0 & \neq 0 & \cdots & ? & \cdots & ? & ? \\ \neq 0 & ? & \cdots & ? & \cdots & ? & ? \end{bmatrix}$$