Improved Testing of Low Rank Matrices

Yi Li
Max-Planck Institute for Informatics
yli@mpi-inf.mpg.de

Zhengyu Wang
Institute for Interdisciplinary Informatics Science (IIIS)
Tsinghua University
wangsincos@163.com

David P. Woodruff
IBM Research, Almaden
dpwoodru@us.ibm.com

ABSTRACT

We study the problem of determining if an input matrix $A \in \mathbb{R}^{m \times n}$ can be well-approximated by a low rank matrix. Specifically, we study the problem of quickly estimating the rank or stable rank of A, the latter often providing a more robust measure of the rank. Since we seek significantly sublinear time algorithms, we cast these problems in the property testing framework. In this framework, A either has low rank or stable rank, or is far from having this property. The algorithm should read only a small number of entries or rows of A and decide which case A is in with high probability. If neither case occurs, the output is allowed to be arbitrary. We consider two notions of being far: (1) A requires changing at least an ϵ-fraction of its entries, or (2) A requires changing at least an ϵ-fraction of its rows. We call the former the "entry model" and the latter the "row model". We show:

- For testing if a matrix has rank at most d in the entry model, we improve the previous number of entries of A that need to be read from $O(d^2/\epsilon^2)$ (Krauthgamer and Sasson, SODA 2003) to $O(d^2/\epsilon)$. Our algorithm is the first to *adaptively* query the entries of A, which for constant d we show is necessary to achieve $O(1/\epsilon)$ queries. For the important case of $d = 1$ we also give a new non-adaptive algorithm, improving the previous $O(1/\epsilon^2)$ queries to $O(\log^2(1/\epsilon)/\epsilon)$.
- For testing if a matrix has rank at most d in the row model, we prove an $\Omega(d/\epsilon)$ lower bound on the number of rows that need to be read, even for adaptive algorithms. Our lower bound matches a non-adaptive upper bound of Krauthgamer and Sasson.
- For testing if a matrix has stable rank at most d in the row model or requires changing an ϵ/d-fraction of its rows in order to have stable rank at most d, we prove that reading $\tilde{\Theta}(d/\epsilon^2)$ rows is necessary and sufficient.

We also give an empirical evaluation of our rank and stable rank algorithms on real and synthetic datasets.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623736.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computation on matrices*; G.2.3 [**Discrete Mathematics**]: Applications

General Terms

Algorithms, Theory

Keywords

dimensionality reduction, principal component analysis, property testing, robustness, stable rank

1. INTRODUCTION

Low rank approximation is a popular tool in computer science with applications to computer vision, information retrieval, and machine learning. In many of these applications, such as image, video, multimedia processing, web data, and bioinformatics the dimensionality of the data is very large. This makes designing algorithms for processing such data more challenging, requiring very low memory and extremely fast processing time.

A saving grace of large-scale data is that it is often of low intrinsic dimension. For example, in Principal Component Analysis (PCA) [6, 7, 13] the data points are column vectors of a matrix A with the assumption that A can be expressed as $L+N$ for L a matrix of low rank and N a matrix of small Frobenius norm, which could typically model noise that has been added to A. Replacing A with the matrix L provides a good low rank approximation to A. PCA has a wide range of applications, including non-negative matrix factorization [9], latent dirichlet allocation [1], clustering [3], and geometric shape fitting problems [4]. There is a large body of work on randomized algorithms for low rank approximation; we refer the reader to Section 5 of the survey by Mahoney [11].

Recently, a new form of PCA called robust PCA was introduced [2]. In this problem, the data points are again column vectors of a matrix $A = L+N$, where L is a low rank matrix, but now N is only guaranteed to be a sparse matrix. Unlike classical PCA, the entries of N can be arbitrarily large provided there are a small number of non-zero entries (the locations of the non-zero entries of N are unknown). This makes robust PCA less sensitive to outlier contamination. We refer the reader to [2] in which applications of robust PCA to video surveillance, face recognition, latent semantic indexing, ranking and collaborative filtering are given. In typical applications, such as recommender systems [15], L is

a matrix of a small constant rank. Surprisingly, under certain assumptions there are efficient algorithms for recovering L and N. One assumption is that the number of non-zero entries of N is at most a sufficiently small constant fraction of the total number of entries.

Independently of the work above, the property testing community has also studied whether a matrix can be expressed as a small perturbation of a low rank matrix [8, 12]. In the property testing model there is an unknown, typically very large object, such as a graph, a matrix, or a vector. This object is queried in certain positions in order to determine if it satisfies a property $\mathcal{P}$ or is far from satisfying $\mathcal{P}$. For an introduction to property testing, we refer the reader to a survey by Goldreich [5]. The relevant results in the property testing literature for robust PCA are those for what we refer to as the Rank property. In this problem, the input matrix A is either of rank at most d, or requires changing an ϵ-fraction of its entries in order to become a matrix of rank at most d. Note that this is a decision version of the robust PCA problem: either $A = L$ in the notation above, or if $A = L + N$ for a matrix L of rank at most d, then necessarily more than an ϵ-fraction of entries of N are non-zero. Distinguishing these two cases allows one to decide whether the assumptions required for a robust PCA algorithm to succeed hold. If the input A is in neither case, then it is allowed for the algorithm to output an arbitrary answer, which is acceptable for the robust PCA application since robust PCA is guaranteed to work if N has at most an ϵ-fraction of non-zero entries.

The Rank problem was studied by Krauthgamer and Sasson [8], who showed there exists a randomized algorithm succeeding with 99% probability on every input matrix A and reading only $O(d^2/\epsilon^2)$ entries of A. This bound is independent of the dimensions of the matrix A. This provides a quick, provably correct method for determining whether robust PCA procedures will work on A, without having to run them in case A is not well-approximated by a low rank matrix. Other methods such as clustering and recommendation systems can also benefit by first running an algorithm for Rank to determine if A is close to a low rank matrix.

Despite this progress, there are several natural questions that remain:

1. In machine learning problems a quadratic dependence on ϵ is often prohibitive. Can one improve the $O(d^2/\epsilon^2)$ algorithm of [8] to have a linear dependence on $1/\epsilon$?

2. In differential equation applications, one often has a sparse matrix stored in Compressed Sparse Row (CSR) or Compressed Sparse Columns (CSC) representation, which allows the retrieval of an entire row or column almost as quickly as a single entry. What is the complexity of the Rank problem in this model? To distinguish this model from the previous model, we refer to this as the "row model", while the model in which individual entries are changed is the "entry model".

3. It is often more common for a matrix to have low *stable rank* than low rank, where the stable rank is defined as $\|A\|_F^2/\|A\|^2$. Here $\|A\|_F$ is the Frobenius norm and $\|A\|$ the operator norm. The stable rank is a continuous, robust relaxation of the rank, with applications to finding well-conditioned submatrices [16]. Can we design algorithms for the StableRank problem, of determining if A has stable rank at most d, or requires changing an ϵ/d-fraction of rows to have stable rank at most d? For this question to make sense, we assume as is often done when working with the stable rank [16], that the rows of A have Euclidean norm at most 1, as otherwise one can increase the norm of a single row of A until its stable rank is arbitrarily close to 1. It also makes sense to parameterize the problem in terms of changing an ϵ/d-fraction of rows rather than an ϵ-fraction of rows, since by replacing a $1/d$ fraction of rows with the vector v for an arbitrary unit vector v, one can always reduce the stable rank to at most d.

Our Contributions: In this paper we thoroughly study both the Rank and StableRank problems. We answer the questions above, providing new theoretical and empirical guarantees for these problems.

Results for the Rank Problem:

1. In the entry model, by allowing queries (i, j) to be adaptively chosen based on the values $A_{i',j'}$ of previously queried entries (i', j'), we are able to improve the algorithm of [8] to give an algorithm which makes only $O(d^2/\epsilon)$ rather than $O(d^2/\epsilon^2)$ queries. Our algorithm, like that of [8] has one-sided error, meaning that if A is of rank at most d the algorithm will be correct with probability 1, while if A is ϵ-far from this property, the algorithm succeeds with probability .99.

2. We show that, for constant d, adaptivity is necessary for achieving this improved algorithm. That is, we show that any algorithm which makes only non-adaptive queries, meaning it chooses its query set before reading any of the entries of A, requires reading $\Omega((\log 1/\epsilon)/\epsilon)$ entries of A. As our upper bound for constant d is $O(1/\epsilon)$ queries, this demonstrates a separation in the power of adaptivity.

3. We further study the problem when $d = 1$, which has important applications to parsing images of building facades [17]. In this case we design a non-adaptive algorithm which achieves $O((1/\epsilon)\log^2(1/\epsilon))$ queries in the entry model, improving the $O(1/\epsilon^2)$ non-adaptive algorithm of [8].

4. In the row model, we show that any, possibly adaptive, algorithm requires reading $\Omega(d/\epsilon)$ rows of A. This matches a non-adaptive $O(d/\epsilon)$ algorithm of [8].

Results for the StableRank Problem:

1. We show in the row model that reading a total of $O(d \log n \log(d \log n)/\epsilon^2)$ non-adaptively chosen rows suffices to solve the problem.

2. We also show an $\Omega(d/\epsilon^2)$ lower bound in the row model. Our lower bound holds even for adaptive algorithms, and is optimal up to an $O(\log n \log(d \log n))$ factor.

We experimentally validate our algorithms for Rank and StableRank on several natural input distributions on A and sparsity patterns N.

For the StableRank problem, we use real datasets from the University of Florida Sparse Matrix Collection. We show that for a large fraction of the matrices in this dataset, our algorithms only need to sample a very small fraction of rows

to solve the StableRank problem. We parameterize the number of rows that need to be read as a function of the stable rank parameter d for these datasets.

For the Rank problem, we use synthetic datasets. Our experiments show particularly noticeable improvements for adaptive query algorithms over non-adaptive query algorithms for small ϵ. For example, for $\epsilon = 0.01$ and $d = 1$, for one of our input distributions the number of adaptive queries is 7% of the number of non-adaptive queries required.

Paper Outline: We give our adaptive algorithm for the Rank problem in the entry model in Section 2, and show that adaptivity is essential by proving a lower bound for non-adaptive algorithms in Section 3. In Section 4, we give a new non-adaptive algorithm for the important case of $d = 1$, which comes close to the lower bound we prove for non-adaptive algorithms in Section 3. In Section 5 we consider the row model, and prove a lower bound on the number of rows read for the RANK problem. In Section 6 we give an algorithm for the StableRank problem and show a nearly matching lower bound, both in the row model. Finally, we present our experimental results in Section 7.

2. ALGORITHM FOR RANK PROBLEM

In this section we study the Rank problem with adaptive queries. We assume that $\min(m, n) = \omega(d/\epsilon)$, that is, that $\min(m, n)$ is larger than cd/ϵ for any fixed constant $c > 0$. This is consistent with our goal of testing if A has small rank.

We first review the algorithm for Rank in [8]. Suppose that the input matrix A has rank greater than d. That algorithm tries to find a submatrix with rank greater than d. The algorithm starts with an empty submatrix and iteratively grows the submatrix by appending one random row and one random column. Let B_t be the submatrix maintained at step t and $X_t = \text{rank}(B_t)$. It was shown in [8] that $\Pr\{X_{t+1} > X_t | X_t \le d\} \ge \epsilon/3$ and thus by a Chernoff bound, $t = O(d/\epsilon)$ suffices to reach $X_t > d$ with constant probability.

Algorithm 1 Our Algorithm for the Rank problem

```
I ← ∅, J ← ∅
for t = 1 to O(d²/ε) do
    Pick (i, j) uniformly random from I^c × J^c
    Query A_{I,j}, A_{i,J} and A_{i,j}
    if rank(A_{I∪{i},J∪{j}}) > rank(A_{I,J}) then
        I ← I ∪ {i}, J ← J ∪ {j}
    end if
    B_t ← A_{I,J}
    if rank(B_t) > d then
        return "A is ε-far from rank d"
    end if
end for
return "A is of rank d"
```

In our adaptive algorithm, we also augment B_t in each step until $\text{rank}(B_t) > d$. We formally write our algorithm in Algorithm 1. Suppose at step t, $\text{rank}(B_t) < d$ and I and J are the index sets of the rows and columns of B_t, respectively. Consider the index pairs $I^c \times J^c$, where $I^c = [m] \setminus I$ and $J^c = [n] \setminus J$, where for an integer ℓ, $[\ell] = \{1, 2, \ldots, \ell\}$. We claim that at least an $\Omega(\epsilon)$ fraction of the index pairs in $I^c \times J^c$ would increase $\text{rank}(B_t)$. Assume that this is true for the moment. Then in expectation, $O(1/\epsilon)$ random samples in $I^c \times J^c$ suffice for there to exist a sample index pair that would increase the rank B_t after augmenting with respect to that index pair. We can find one such pair by checking each chosen possible augmentation of B_t. Call the pair found B_{t+1}. By linearity of expectation and a Chernoff bound, $t = O(d)$ steps suffice to give $\text{rank}(B_t) > d$. The number of entries read is, in expectation, bounded by

$$\sum_{t=0}^{O(d)} O\left(\frac{2t+1}{\epsilon}\right) = O\left(\frac{d^2}{\epsilon}\right).$$

Now we prove our claim above to complete the proof. We can assume, without loss of generality, that B_t consists of an upper left submatrix of A. Since we assume that $\min(m, n) = \omega(d/\epsilon)$, and B_t has at most d rows and columns, we can change all the entries of A in the first t columns and first t rows so that the rows restricted to the first t columns are in the row span of B_t, and the columns restricted to the first t rows are in the column span of B_t. This only changes at most an $\epsilon/2$-fraction of the total number of entries of A. Next, for each entry (i, j) not among the first t columns or rows, we can change the value of $A_{i,j}$ so that augmenting B_t by the pair (i, j) does not increase the rank of B_t. Since we must change at least an ϵ-fraction of overall entries of A to reduce the rank to at most d, and B_t has rank at most d, the number of index pairs in $I^c \times J^c$ that would increase $\text{rank}(B_t)$ must be at least $\epsilon mn/2$.

Our algorithm is optimal for constant d, because it requires $\Omega(1/\epsilon)$ queries just to distinguish a zero matrix from a matrix with ϵmn randomly placed non-zero entries.

3. LOWER BOUND FOR RANK PROBLEM FOR NON-ADAPTIVE ALGORITHMS

In this section, we start with a simple example to demonstrate that it is generally hard to improve the non-adaptive upper bound of $O(1/\epsilon^2)$ for Rank even for $d = 1$, for a class of natural non-adaptive algorithms which query submatrices and make their decision based on the maximum rank of them. Next, we give a proof that any randomized non-adaptive algorithm requires $\Omega((\log 1/\epsilon)/\epsilon)$ queries for $d \ge 1$.

3.1 A Hard Input for a Class of Natural Non-adaptive Algorithms

To design non-adaptive algorithms, a natural way is to select some submatrices of A to query, namely $A_1, \cdots, A_t$, and then make a decision based on whether $\max_{i \in [t]} \text{rank}(A_i) > d$. However, there is an example of A such that the number of queries required is at least $\Omega(1/\epsilon^2)$ for such algorithms, even when $d = 1$. In the following we fix $d = 1$. One can easily extend the result to any d.

Denote

$$M = \begin{pmatrix} 0_{\epsilon n \times \epsilon n} & 1_{\epsilon n \times (1-\epsilon)n} \\ 1_{(1-\epsilon)n \times \epsilon n} & 0_{(1-\epsilon)n \times (1-\epsilon)n} \end{pmatrix},$$

where $1_{r,c}$ is an r-by-c matrix whose entries are all 1s. Let A be the matrix obtained from uniformly randomly permuting the rows and columns of M.

In order to find a fully queried submatrix whose rank is more than 1, one must query an entry in A corresponding to an entry of the top-left submatrix in M (we call such an entry critical), whose size is just $\epsilon n \times \epsilon n$. Therefore, if the

total query size is $o(1/\epsilon^2)$, the probability is $o(1)$ that one has queried a critical entry in order to find that $\text{rank}(A) = 2$ instead of 1. Hence, a lower bound of $\Omega(1/\epsilon^2)$ holds for non-adaptive algorithms which query a set of submatrices and decide on whether the maximum rank of those submatrices is more than d.

In fact, for more complicated algorithms, it is possible to reduce the non-adaptive query size when $d = 1$. We shall study it in Section 4.

The example here also illustrates the superiority of adaptive queries over non-adaptive ones. An adaptive algorithm needs $O(1/\epsilon)$ queries (in expectation) to find an entry of value 1, and based on the position of that entry, the algorithm can then extend it to a matrix of rank 2 with $O(1/\epsilon)$ more queries; while a non-adaptive algorithm does not know which rank-1 matrix to extend.

3.2 An $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ Non-adaptive Lower Bound for Constant d

In this subsection, we prove the following theorem, which can be automatically extended to arbitrary d.

THEOREM 1. *Any randomized non-adaptive algorithm for the* Rank *problem with $d = 1$ requires $\Omega((1/\epsilon)\log(1/\epsilon))$ queries.*

To give a lower bound for non-adaptive queries for any randomized algorithm, we apply Yao's Lemma, and (1) define two distributions D_0, D_1, such that D_0 is a distribution of matrices of rank at most d (or $\Pr_{M\sim D_0}\{\text{rank}(M) \le d\} = 1$, the same below), while D_1 is a distribution of matrices which are ϵ-far from rank d; (2) prove that with high probability, any deterministic non-adaptive set of $(c/\epsilon)\log(1/\epsilon)$ entries cannot distinguish D_0 from D_1, where $c > 0$ is a constant.

Algorithm 2 Hard Distribution

1: Let i be uniformly sampled in $[k]$.
2: Let $r = n/2^{i-1}$, $c = \epsilon n \cdot 2^i$, and x_1, x_2, y_1, y_2 be i.i.d. $N(0, I_n)$ vectors.
3: Let $M_0, M_1 \in \mathbb{R}^{n\times n}$ be

$$M_0 = \begin{pmatrix} x_1 y^T & 0_{r,n-c} \\ 0_{n-r,c} & 0_{n-r,n-c} \end{pmatrix}$$

and

$$M_1 = \begin{pmatrix} x_1 y_1^T + x_2 y_2^T & 0_{r,n-c} \\ 0_{n-r,c} & 0_{n-r,n-c} \end{pmatrix},$$

where $y_i = \sqrt{(y_1)_i^2 + (y_2)_i^2}$.
4: Let $P_r, P_c \in \mathbb{R}^{n\times n}$ be two uniformly random permutation matrices.
5: Let $\mathcal{D}_0$ be the distribution of $P_r M_0 P_c$ and D_1 the distribution of $P_r M_1 P_c$.

We define the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ on $\mathbb{R}^{n\times n}$ in Algorithm 2. Notice that $\mathcal{D}_0$ is a distribution of matrices of rank 1 with probability 1 while $\mathcal{D}_1$ is a distribution such that a random sample is ϵ-far from a rank-1 matrix with probability 1.

Now consider a deterministic algorithm for testing the matrix A sampled from either of the two distributions with equal probability. The queries of the algorithm can be written as a deterministic subset $S \subseteq [n] \times [n]$. The following lemma is straightforward by the construction of the distributions, together with the property of normal distributions that $N(\mu_1, \sigma_1^2) + N(\mu_2, \sigma_2^2) \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.

LEMMA 1. *If for each row and column of A the number of observed non-zero entries is at most 1, then the algorithm cannot determine whether A is "of rank 1" or "ϵ-far from any rank-1 matrix" better than a random guess. Formally,*

$$\begin{aligned} &(\forall j \in [n], |\{(i,j) \in S \,|A_{i,j} \neq 0\}| \le 1) \\ &\wedge (\forall i \in [n], |\{(i,j) \in S \,|A_{i,j} \neq 0\}| \le 1) \ . \\ &\Rightarrow \Pr\{\text{rank}(A) \le 1 \mid A_S\} = \tfrac{1}{2} \end{aligned}$$

To upper-bound the probability that two or more non-zero observations are in a query row or column, we need the following lemma. It follows from a union bound argument and simple inequalities.

LEMMA 2. *Suppose that there are n bins, m of which contain a ball each. Then choosing b bins uniformly at random collects at least 2 balls with probability at most $(bm/n)^2$.*

PROOF. We pick b bins one by one. The probability that two particular bins both contain balls is at most $(m/n)^2$. Also notice that if at least 2 balls are picked, it must be the case that there exist two attempts both of which have balls. Applying a union bound, we obtain the probability that we collect at least 2 balls is at most $\binom{b}{2} \cdot \left(\frac{m}{n}\right)^2 \le \left(\frac{bm}{n}\right)^2$. □

The next is the most important lemma, which is a bit technical. It says that if the number of non-adaptive queries is small, then the probability will be small that there exists one column such that the number of non-zero observations on that column is larger than 1.

LEMMA 3. *If $|S| \le \frac{1}{192\epsilon}\log\frac{1}{\epsilon}$, then*

$$\Pr\{\exists j \in [n], |\{(i,j) \in S \,|A_{i,j} \neq 0\}| \ge 2\} \le 1/8.$$

PROOF. We start with some definitions. For every $i \in [k-1]$, let x_i be the number of columns in $[n]$ such that the number of entries observed on that column is larger than 2^{i-1} but no more than 2^i. Let x_k be the number of columns in $[n]$ such that the number of entries observed on that column is larger than 2^{k-1}. More formally, for $i \in [k-1]$, let

$$x_i = \left|\left\{ j \,\middle|\, 2^{i-1} < |(\cdot, j) \cap S| \le 2^i \right\}\right|,$$

and for $i = k$,

$$x_i = \left|\left\{ j \,\middle|\, 2^{i-1} < |(\cdot, j) \cap S| \right\}\right|.$$

We know that

$$2\,|S| \ge \sum\nolimits_{i\in[k]} 2^i \cdot x_i.$$

For $i \in [k]$, let P_i bet the probability that there exists one column containing 2 or more observed non-zero entries, conditioned on the event that A has an $(n/2^{i-1}) \times (\epsilon n 2^i)$ submatrix of non-zero entries (i.e., i is chosen when it is generated in Algorithm 2). By Lemma 2, we obtain that for all $j \in [k]$,

$$P_j \le \epsilon \cdot 2^j \cdot \left(\sum_{i=1}^{j-1} x_i \cdot 4^{1+i-j} + \sum_{i=j}^{k} x_i \right).$$

Notice that the factor $\epsilon 2^j$ comes from the fact that there are only $\epsilon 2^j n$ columns that are non-zero in A. If we visit

2^i entries on a column of $n/2^{j-1}$ non-zero entries, the probability that we hit at least 2 non-zero entries is at most $\left(\frac{2^i \cdot n/2^{j-1}}{n}\right)^2 = 4^{1+i-j}$. If it is more than 1, we bound it by 1 since it is a probability. Therefore,

$$P_j \le \epsilon \cdot \left(\sum_{i=1}^{j-1} x_i \cdot 2^{2+2i-j} + \sum_{i=j}^{k} 2^j \cdot x_i \right).$$

Summing over all $j \in [k]$ yields that

$$\sum_{j\in[k]} P_j \le 12\epsilon \cdot \sum_{j\in[k]} 2^j \cdot x_j \le 24\epsilon \left|S\right|.$$

Therefore, if $|S| \le \frac{1}{192\epsilon} \log \frac{1}{\epsilon}$, then

$$\frac{1}{k} \sum_{j\in[k]} P_j \le 1/8,$$

i.e.,

$$\Pr\left\{\exists j \in [n], \left|\{(i,j) \in S \left| A_{i,j} \ne 0 \right.\}\right| \ge 2\right\} \le 1/8.$$

□

Extending Lemma 3 to rows and combining with Lemma 1, we can prove Theorem 1, i.e., any non-adaptive algorithm that solves our problem takes $\Omega\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ queries.

4. NON-ADAPTIVE RANK ONE ALGORITHM

In this section, we give a non-adaptive algorithm for the Rank problem with $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ queries when $d = 1$ and $\epsilon \le 1/e$. Let η be such that $\eta\log(1/\eta) = \epsilon$ and $\eta < 1/2$. Also let $k = \log 1/\eta$. The proposed algorithm is as follows. We describe it for an $n \times n$ matrix A, though it immediately extends to rectangular matrices as well.

Choose $R_1, \cdots, R_k$ and $C_1, \cdots, C_k$ from $[n]$ uniformly at random such that

$$R_1 \subseteq \cdots \subseteq R_k, \quad C_1 \supseteq \cdots \supseteq C_k,$$

and

$$|R_i| = c_0 2^i d, \quad |C_i| = c_0 d/2^i\eta,$$

where c_0 is a sufficiently large constant to be determined later. Denote $Q = \bigcup_{i=1}^{k}(R_i \times C_i)$, the overall set of entries the algorithm will query. Then, the algorithm computes

$$\min_{A_{(R_k,C_1)\setminus Q}} \operatorname{rank}(A_{R_k,C_1}),$$

the minimum possible rank of the matrix, and decides that "A is ϵ-far from being rank-d" iff the minimum possible rank is more than d.

Notice that the total number of entries the algorithm queries is $O(d^2\log(1/\eta)/\eta) = O((d^2/\epsilon)\log^2(1/\epsilon))$. Now we justify the correctness of the proposed algorithm for $d = 1$.

For fixed $A \in \mathbb{R}^{n\times n}$ which is ϵ-far from being rank -d, call (r,c) an augment for $R \times C \subseteq [n] \times [n]$ if $r \in [n] \setminus R$, $c \in [n] \setminus C$ and $\operatorname{rank}(A_{R\cup\{r\},C\cup\{c\}}) > \operatorname{rank}(A_{R,C})$. Let $aug(R,C)$ be the set of all the augments, that is,

$$aug(R,C) = \{(r,c) \in ([n]\setminus R) \times ([n]\setminus C) : \operatorname{rank}(M_{R\cup\{r\},C\cup\{c\}}) > \operatorname{rank}(M_{R,C})\}.$$

For fixed R, C and A, define $count_r$ ($r \in [n]\setminus R$) to be the number of c's such that $(r,c) \in aug(R,C)$. Let $count^*_{i\in[n-|R|]}$ be the non-increasing reordering of the sequence $(count_r)_{r\in[n]\setminus R}$. For simplicity of notation, let $count^*_i = 0$ if $i > n - |R|$. The following lemma follows from the fact that the number of augments is at least ϵn^2 if A is ϵ-far from being rank-d and $\operatorname{rank}(A_{R,C}) \le d$, as argued in Section 2.

LEMMA 4. *If A is ϵ-far from being rank-d and $rank(A_{R,C}) \le d$, then*

$$|aug\,(R,C)\,| = \sum_{r\in[n]\setminus R} count_r = \sum_i count^*_i \ge \epsilon n^2.$$

We define the concept of an augment pattern below.

Definition 1. For M, R, C and $i \in [\log(1/\eta)]$, we say that (R,C) has augment pattern i on A iff $count^*_{n/2^i} \ge 2^{i-1}\eta n$.

Following the definition, we show the existence of at least one augment pattern for (R,C) when A is ϵ-far from being rank-d and $\operatorname{rank}(M_{R,C}) \le d$.

LEMMA 5. *If A is ϵ-far from being rank-d and $\operatorname{rank}(A_{R,C}) \le d$, then there exists i such that (R,C) has augment pattern i.*

PROOF. We prove the lemma by contradiction. Suppose that (R,C) does not have augment pattern i for all $i \in [\log(1/\eta)]$, i.e.,

$$count^*_{n/2^i} < 2^{i-1}\eta n, \quad i = 1, 2, \ldots, \log(1/\eta).$$

It follows that

$$\begin{aligned}
\sum_i count^*_i &= \sum_{i=\frac{n}{2}+1}^{n} count^*_i + \sum_{i=\frac{n}{4}+1}^{\frac{n}{2}} count^*_i + \cdots \\
&\quad + \sum_{i=\frac{n}{2^{\log(1/\eta)}}+1}^{\frac{n}{2^{\log(1/\eta)-1}}} count^*_i + \sum_{i=1}^{\eta n} count^*_i \\
&< \frac{\eta n^2}{2} \cdot (\log(1/\eta) + 1) \\
&< \eta \log(1/\eta)\, n^2 \quad (\text{since } \eta < 1/2) \\
&= \epsilon n^2,
\end{aligned}$$

which contradicts Lemma 4. □

Note that if (R,C) has augment pattern i on A, a uniformly random rectangle sample of dimension $c2^i \times c/2^i\eta$ will hit at least one augment with high probability, which is at least

$$\left(1 - \left(1 - 2^{-i}\right)^{c2^i}\right)\left(1 - \left(1 - 2^{i-1}\eta\right)^{c/2^i\eta}\right) \ge 1 - \frac{2}{e^{c/2}}.$$

We conclude this fact as

LEMMA 6. *Suppose that (R,C) has augment pattern i on A and $j \in \{i-1, i\}$. Let $R', C' \subseteq [n]$ be uniformly random such that $|R'| = c2^j, |C'| = c/2^j\eta$. Then the probability that (R',C') contains at least one augment of (R,C) on A is at least $1 - 2e^{-c/2}$.*

Now we are ready to show the correctness for the proposed algorithm.

THEOREM 2. *Suppose that $\epsilon \le 1/e$. For any matrix A (either of rank at most $d = 1$, or at least ϵ-far from it), the probability that the proposed algorithm is erroneous is at most $1/3$, provided that $c_0 \ge 12$.*

PROOF. If A is of rank at most 1, the algorithm will never be wrong. Now we analyze the case that A is ϵ-far from being rank-1. We discuss the two cases based on the number of augment patterns for $(\emptyset,\emptyset)$ on A.

Case (i) $(\emptyset,\emptyset)$ has only one single augment pattern.

Let i denote the only augment pattern that $(\emptyset,\emptyset)$ has. We divide R_i uniformly at random into two even parts, $R_i^{(1)}$ and $R_i^{(2)}$. Do the same with C_i, obtaining $C_i^{(1)}$ and $C_i^{(2)}$. By Lemma 6, the probability that $A_{R_i^{(1)},C_i^{(1)}}$ contains at least one non-zero entry is at least $1-2e^{-c_0/4}$. Let us condition on this event.

Let $(r,c) \in (R_i^{(1)}, C_i^{(1)})$ be such that $A_{r,c} \neq 0$. Then $(\{r\},\{c\})$ has augment pattern i by Lemma 5, while on the other hand it is impossible that $(\{r\},\{c\})$ has augment pattern other than i, since $(\emptyset,\emptyset)$ does not have the augment pattern. Now consider the probability that $(R\backslash\{r\}, C\backslash\{c\})$ contains an augment for $(\{r\},\{c\})$. We claim that this probability is also at least $1-2e^{-c_0/4}$. Since $R_i^{(2)}$ and $C_i^{(2)}$ are uniformly random given $R_i^{(1)}$ and $C_i^{(1)}$, we can use a coupling argument to show that the probability that $(R_i\backslash\{r\}, C_i\backslash\{c\})$ contains at least one augment for $(\{r\},\{c\})$ is greater than a uniformly random sample of dimension $c_0 2^i/2 \times c_0/2^{i+1}\eta$ in $([n]\setminus\{r\})\times([n]\setminus\{c\})$ does.

Therefore, the probability to augment one empty matrix to a 2×2 full rank matrix is at least $1-4e^{-c_0/4} > 2/3$, and the algorithm answers correctly in this case.

Case (ii) $(\emptyset,\emptyset)$ has multiple augment patterns.

In this case, suppose that $(\emptyset,\emptyset)$ has augment patterns i and j $(i<j)$. Divide R_i uniformly at random into two even parts $R_i^{(1)}$ and $R_i^{(2)}$, and C_j into $C_j^{(1)}$ and $C_j^{(2)}$. Also divide $R_j\backslash R_i$ evenly into $R^{(1)}$ and $R^{(2)}$, $C_i\backslash C_j$ into $C^{(1)}$ and $C^{(2)}$. According to Lemma 6, the probability that both $(R_i^{(1)}, C_j^{(1)}\cup C^{(1)})$ and $(R_i^{(1)}\cup R^{(1)}, C_j^{(1)})$ intersect with $aug(\emptyset,\emptyset)$ is at least $1-4^{-c_0/4}$. Conditioned on this, we discuss two cases based on whether $(R_i^{(1)}, C_j^{(1)})$ intersects with $aug(\emptyset,\emptyset)$.

Case (ii.1): $(R_i^{(1)}, C_j^{(1)}) \cap aug(\emptyset,\emptyset) = \emptyset$. Let $(r_i,c_i) \in (R_i^{(1)}, C^{(1)})$ be such that $A_{r_i,c_i}\neq 0$ and $(r_j,c_j)\in(R^{(1)}, C_j^{(1)})$ be such that $A_{r_j,c_j}\neq 0$. Since $A_{r_i,c_j}=0$, we know that $\mathrm{rank}(A_{(\{r_i,r_j\},\{c_i,c_j\})}) = 2$ so the algorithm answers correctly.

Case (ii.2): $(R_i^{(1)}, C_j^{(1)}) \cap aug(\emptyset,\emptyset) \neq \emptyset$. Let $(r,c) \in (R_i^{(1)}, C_j^{(1)}) \cap aug(\emptyset,\emptyset)$. Following a similar argument of case (ii.1), we can prove that the probability is at least $1-2e^{-c_0/4}$ that (r,c) could be augmented with augment pattern i by $(R_i\backslash\{r\}, C_i\backslash\{c\})$ (or with augment pattern j by $(R_j\backslash\{r\}, C_j\backslash\{c\})$. So the overall probability is at least $1-6e^{-c_0/4} > 2/3$ that the algorithm answers correctly in this case by finding a submatrix of rank 2. □

5. LOWER BOUND FOR RANK IN THE ROW MODEL

In this section, we discuss the Rank problem in the row model. Recall that we say A is ϵ-far from having property P if at least ϵn rows of A have to be changed for A to have property P. The Rank problem in this model is to test whether the matrix has rank at most d or is ϵ-far from having rank at most d.

In this model, the algorithm of [8] gives an upper bound of $O(d/\epsilon)$ rows. Next we show a matching lower bound when the entries of A come from any field $\mathbb{F}$, e.g., the real numbers. Assume that $n \geq 2d/\epsilon$ throughout this section.

First assume $\mathbb{F}$ is a finite field. Let $\mathcal{D}_1$ be a distribution over $n\times n$ matrices defined as follows. Choose a random d-dimensional subspace W in $\mathbb{F}^n$ and then choose $2\epsilon n$ uniformly random vectors from W. Place these $2\epsilon n$ vectors on $2\epsilon n$ uniformly random rows of an $n\times n$ matrix. The resulting distribution is $\mathcal{D}_1$. We define $\mathcal{D}_2$ similarly, except that W is a uniformly random $(d+\epsilon n)$-dimensional subspace in $\mathbb{F}^n$. Clearly $\mathrm{rank}(A)\leq d$ when $A\sim\mathcal{D}_1$. When $B\sim\mathcal{D}_2$, with probability $1-o(1)$, one needs to change at least ϵn rows of B to reduce its rank to d.

By construction, adaptively choosing rows does not help in distinguishing $\mathcal{D}_1$ from $\mathcal{D}_2$, and so we may assume the query algorithm is non-adaptive. Fix $Q\subseteq\{1,\ldots,n\}$ with $|Q|=q$. Let $A_Q=(A_{ij})_{i\in Q,1\leq j\leq n}$ and define B_Q similarly. Each defines a distribution on $q\times n$ matrices, denoted by $\mathcal{L}(A_Q)$ and $\mathcal{L}(B_Q)$, respectively.

LEMMA 7. *Suppose that $\mathbb{F}$ is a finite field. When $q\leq \alpha d/(8\epsilon)$, it holds that $d_{TV}(\mathcal{L}(A_Q),\mathcal{L}(B_Q))\leq \alpha + |\mathbb{F}|^{-d/4} + o(1)$, where d_{TV} denotes total variation distance.*

PROOF. When $q\leq\alpha d/(8\epsilon)$, by a Markov bound, with probability $\geq 1-\alpha$ at most $d/4$ vectors of the chosen $2\epsilon n$ ones are read. For distribution $\mathcal{D}_1$, with probability $\geq 1-|\mathbb{F}|^{-d/4}$, the vectors are linearly independent. For distribution $\mathcal{D}_2$, with probability $\geq 1-o(1)$, the vectors are linearly independent. The conclusion follows immediately from the observation that conditioned on the vectors being linearly independent, they are distributed as a set of uniformly chosen $d/4$ linearly independent vectors in $\mathbb{F}^n$. □

For $\mathbb{F}=\mathbb{R}$, we define $\mathcal{D}_1$ and $\mathcal{D}_2$ similarly, except that the $2\epsilon n$ random vectors are chosen subject to the multidimensional Gaussian measure on W. Similarly to the lemma above, we have,

LEMMA 8. *Suppose that $\mathbb{F}=\mathbb{R}$ and $\alpha>0$. When $q\leq\alpha d/(8\epsilon)$, it holds that $d_{TV}(\mathcal{L}(A_Q),\mathcal{L}(B_Q))\leq\alpha+o(1)$.*

PROOF. When $q\leq\alpha d/(8\epsilon)$, by a Markov bound, with probability $\geq 1-\alpha$ at most $d/4$ vectors of the chosen $2\epsilon n$ ones are read. For both distributions, the randomly chosen vectors are linearly independent almost surely. The conclusion follows immediately from the observation that conditioned on the vectors being linearly independent, they are distributed as a set of uniformly chosen $d/4$ linearly independent vectors in $\mathbb{R}^n$. □

The lower bound follows immediately as a corollary.

COROLLARY 1. *In the row model, any algorithm for the Rank problem needs to sample $\Omega(d/\epsilon)$ rows.*

6. STABLE RANK IN THE ROW MODEL

6.1 Upper Bound

Definition 2. (stable rank) Let $A\in\mathbb{R}^{n\times n}$ be a non-zero matrix. The *stable rank* of A is $\mathrm{srank}(A)=\|A\|_F^2/\|A\|^2$.

We will design an algorithm for the StableRank problem for $n\times n$ matrices. We denote the i-th row of A by $A_{i,\cdot}$.

Algorithm 3 Algorithm for the StableRank problem

// $c = \frac{1}{8}(1-\frac{1}{d})^2$
1: $q \leftarrow \Theta\left(\frac{d}{(1-\frac{1}{d})^6\epsilon^2}\log n + d\log n\log(d\log n)\right)$
2: Sample q rows of A, forming $\tilde{A}$
3: $X \leftarrow \frac{n}{q}\|\tilde{A}\|_F^2$
4: **if** $X \le \frac{9}{10}(1-\frac{1}{d})\epsilon n$ **then**
5: output 'stable rank $\le d$'
6: **else**
7: **if** $\frac{n}{q}\|\tilde{A}\|^2 \ge \frac{X}{(1+\frac{cd}{d-1})d}$ **then**
8: output 'stable rank $\le d$'
9: **else**
10: output '$\epsilon n/d$-far from having stable rank $\le d$'
11: **end if**
12: **end if**

LEMMA 9. *Suppose that $d/\epsilon \ge 2$. If A is (ϵ/d)-far from having stable rank $\le d$, then*

$$\|A\|_F^2 \ge (\frac{\epsilon n}{d} - 1)(d-1) \tag{1}$$

$$\|A\|^2 \le (1 + \frac{\epsilon}{d}(1-\frac{1}{d}))\frac{\|A\|_F^2}{d} - (1-\frac{1}{d})(\frac{\epsilon n}{d} - 1). \tag{2}$$

PROOF. Suppose that $x \in S^{n-1}$ satisfies $\|A\| = \|Ax\|_2$. Without loss of generality, assume that $\langle A_{1,\cdot}, x\rangle^2 \le \langle A_{2,\cdot}, x\rangle^2 \le \cdots \le \langle A_{n,\cdot}, x\rangle^2$. Let $m = \lceil \epsilon n/d\rceil - 1$, so that $n > 2m$. Changing each $A_{i,\cdot}$ $(1 \le i \le m)$ to x forms a new matrix B, and it must hold that $\mathrm{srank}(B) > d$.

It is clear that $\|B\|^2 \ge m$ and $\|B\|_F^2 \le \|A\|_F^2 + m$, so

$$d < \mathrm{srank}(B) \le \frac{\|A\|_F^2 + m}{m},$$

whence (1) follows.

Next we prove the second conclusion. It is clear that

$$S_m := \sum_{i=1}^m \langle A_{i,\cdot}, x\rangle^2 \le \frac{m}{n}\sum_{i=1}^n \langle A_{i,\cdot}, x\rangle^2 = \frac{m}{n}\|A\|^2 \le \frac{m}{n}\cdot\frac{\|A\|_F^2}{d}.$$

Observe that

$$\|B\|_F^2 \le \|A\|_F^2 - \sum_{i=1}^m \|A_{i,\cdot}\|_2^2 + m \le \|A\|_F^2 - S_m + m$$

and

$$\|B\|^2 \ge \|A\|^2 - S_m + m.$$

It follows that

$$d < \mathrm{srank}(B) \le \frac{\|A\|_F^2 - S_m + m}{\|A\|^2 - S_m + m}$$

whence (2) follows. □

LEMMA 10. *In Algorithm 3, it holds that $|X - \|A\|_F^2| \le \frac{1}{8}(1-\frac{1}{d})^2\epsilon n$ with probability $\ge 9/10$.*

PROOF. Let $\tau = \frac{1}{8}(1-\frac{1}{d})^2$. By a Chernoff bound, sampling q rows uniformly gives failure probability $2e^{-2q(\tau\epsilon)^2} < 0.1$, that is, $q = \Omega(1/(\tau\epsilon)^2)$. □

LEMMA 11 ([10]). *Let $\tilde{A}$ be a matrix formed by r independent row samples of A according to probability $p_t \ge \beta\|A_{t,\cdot}\|_2^2/\|A\|_F^2$. If $r \ge \frac{4\,\mathrm{srank}(A)}{\beta\eta^2}\ln\frac{2n}{\delta}$ then with probability at least $1-\delta$, it holds that $(1-\eta)\|A\|^2 \le \frac{n}{r}\|\tilde{A}\|^2 \le (1+\eta)\|A\|^2$.*

LEMMA 12. *Let $X \sim \mathrm{Unif}(S^{n-1})$ then $\|x\|_\infty \le \sqrt{\frac{2\log n}{n}}$ with probability $\ge 1 - n^{-2}$.*

THEOREM 3. *Suppose that $\|A\|_{\mathrm{row}} = 1$, then Algorithm 3 is a correct algorithm for the StableRank problem with success probability ≥ 0.6 in the row model. It reads $O(\frac{d\log n}{(1-\frac{1}{d})^6\epsilon^2} + d\log n\log(d\log n))$ rows.*

PROOF. By Lemma 1, if A is far from having stable rank at most d, it must hold that $\|A\|_F^2 \ge (1-1/d)\epsilon n$. Conditioned on the event that X is a good estimator to $\|A\|_F^2$, that is, X satisfies the conclusion of Lemma 10, it holds that $X \ge \frac{9}{10}(1-\frac{1}{d})\epsilon n$. Hence the algorithm is correct on Line 5. Now we assume that $\|A\|_F^2 \ge (1-1/d)\epsilon n$. Let $\eta = \frac{c\epsilon n}{\|A\|_F^2} \le \frac{cd}{d-1} =: \eta'$ then $(1-\frac{\tau\eta'}{c})\|A\|_F^2 \le X \le (1+\frac{\tau\eta'}{c})\|A\|_F^2$.

Now suppose that $\mathrm{srank}(A) > c_1 d$. Let U be a uniformly random $n\times n$ orthogonal matrix. Since we only care about norms of $\tilde{A}$ we can replace $\tilde{A}$ with $\tilde{A}U$, which is a random sample of q rows of AU. Observe that $(AU)_{i,\cdot}$ is a random vector uniform on $\|A_{i,\cdot}\|_2 S^{n-1}$, it follows from Lemma 12 and a union bound that $\|A_{i,\cdot}\|_\infty \le 2\|A_{i,\cdot}\|_2^2(\log n)/n$ for all i with probability $\ge 1 - 1/n$. Conditioned on this event, $\|A\|_{\mathrm{col}}^2 \le 2\|A\|_F^2(\log n)/n \le 2\log n$. Invoking [14, Theorem 1.8],

$$\begin{aligned}\mathbb{E}\|\tilde{A}'\| &\le C_1\sqrt{\frac{q}{n}}\|A\| + C_2\sqrt{\log q}\cdot\sqrt{\frac{2\log n}{n}}\|A\|_F \\ &\le C_1\sqrt{\frac{q}{c_1 d}}\cdot\frac{\|A\|_F}{\sqrt{n}} + 2C_2\sqrt{\log q\log n}\cdot\frac{\|A\|_F}{\sqrt{n}}\end{aligned}$$

and thus with probability ≥ 0.9,

$$\begin{aligned}\|\tilde{A}'\| &\le 10C_1\sqrt{\frac{q}{c_1 d}}\cdot\frac{\|A\|_F}{\sqrt{n}} + 20C_2\sqrt{\log q\log n}\cdot\frac{\|A\|_F}{\sqrt{n}} \\ &\le \frac{1}{\sqrt{1-\frac{\tau\eta'}{c}}}\left(10C_1\sqrt{\frac{q}{c_1 d}} + 20C_2\sqrt{\log q\log n}\right)\sqrt{\frac{X}{n}}\end{aligned}$$

On the other hand, when $\mathrm{srank}(A) \le d$, it holds with probability ≥ 0.9 that

$$\|\tilde{A}'\| \ge \frac{1}{2}\sqrt{\frac{q}{n}}\|A\| \ge \frac{1}{2}\sqrt{\frac{q}{n}}\frac{\|A\|_F}{\sqrt{d}} \ge \frac{1}{2}\sqrt{\frac{q}{n}}\sqrt{1-\frac{\tau\eta'}{c}}\sqrt{\frac{X}{d}}$$

By our choice of parameters,

$$\begin{aligned}&\frac{1}{2}\sqrt{\frac{q}{n}}\sqrt{1-\frac{\tau\eta'}{c}}\sqrt{\frac{X}{d}} \\ &\quad\ge \frac{1}{\sqrt{1-\frac{\tau\eta'}{c}}}\left(10C_1\sqrt{\frac{q}{c_1 d}} + 20C_2\sqrt{\log q\log n}\right)\sqrt{\frac{X}{n}},\end{aligned}$$

provided that c_1 is less than a constant times $1/(1-\frac{1}{d})^2$ and $c = \tau$. Hence we can distinguish the two cases.

Now we assume that $\mathrm{srank}(A) \le c_1 d$. Let $\beta = \|A\|_F^2/n$, so $\beta\|A_{i,\cdot}\|_2^2/\|A\|_F^2 \le 1/n$ for all i, that is, uniform sampling satisfies the assumption of Lemma 11. It then follows from Lemma 11 that with probability at least 0.9, it holds that

$$(1-\eta)\|A\|^2 \le \frac{n}{q}\|\tilde{A}\|^2 \le (1+\eta)\|A\|^2.$$

Conditioned on this event; in the first case, $\|A\|^2 \ge \|A\|_F^2/d$; in the second case, by Lemma 9, $\|A\|^2 \le (1+\frac{\epsilon}{d}(1-\frac{1}{d}))\frac{\|A\|_F^2}{d} -$

$(1-\frac{1}{d})(\frac{\epsilon n}{d}-1)$ and thus

$$\frac{n}{q}\|\tilde{A}\|^2 \le (1+\eta)\left(\frac{1+\frac{\epsilon}{d}(1-\frac{1}{d})}{1-\tau\eta/c}\frac{X}{d}-(1-\frac{1}{d})(\frac{\epsilon n}{d}-1)\right).$$

It is not difficult to establish that

$$(1+\eta)\left(\frac{1+\frac{\epsilon}{d}(1-\frac{1}{d})}{1-\tau\eta/c}\frac{X}{d}-(1-\frac{1}{d})(\frac{\epsilon n}{d}-1)\right) < \frac{1-\eta}{1+\tau\eta/c}\cdot\frac{X}{d}$$

when $c=\tau=\frac{1}{8}(1-\frac{1}{d})^2$. Therefore we can distinguish the two cases. Combining with the discussion above for the case where $\mathrm{srank}(A) > c_1 d$, we see that Line 8 and Line 10 are correct. □

6.2 Lower bound

Let $\mathcal{D}_1$ be a distribution over $n \times n$ matrices defined as follows. Choose a random $x_0 \in S^{n-1}$ and place x_0 in n/d randomly chosen rows of an $n \times n$ matrix A. Place the first $n-n/d$ rows of a random orthogonal matrix in the remaining $n-n/d$ rows of A. Let $\mathcal{D}_1$ be the distribution of A.

We define $\mathcal{D}_2$ similarly as follows. Choose random $x_0 \in S^{n-1}$ and place x_0 in $(1-2\epsilon)n/d$ randomly chosen rows of an $n \times n$ matrix B. Place the first $n-(1-2\epsilon)n/d$ rows of a random orthogonal matrix in the remaining $n-(1-\epsilon)n/d$ rows of B. Let $\mathcal{D}_2$ be the distribution of B.

Suppose that $A \sim \mathcal{D}_1$ and $B \sim \mathcal{D}_2$. It is clear $\|A\|_F^2 = n$ and $\|A\|^2 \ge n/d$, and so $\mathrm{srank}(A) \le d$. Now we upper bound $\|B\|^2$. With probability 1, we know that x_0 does not lie in the span of the orthogonal rows, and so $\|Bx\|^2 < 1+(1-3\epsilon)n/d$, that is, $\|B\|^2 < 1+(1-3\epsilon)n/d \le (1-2\epsilon)n/d$. Changing $\delta n/d$ rows of B forms a new matrix B' with $\mathrm{srank}(B') \le d$. We know that $\|B'\|_F^2 \ge \|B\|_F^2 - \delta n/d = (1-\delta/d)n$ and $\|B'\|^2 \le \|B\|^2 + \delta n/d$. It follows from $\|B'\|_F^2/d \le \|B'\|^2$ that $1-\delta/d \le 1-2\epsilon+\delta$, thus $\delta \ge 2\epsilon(1+1/d) > \epsilon$, and we conclude that with probability 1, the matrix B is (ϵ/d)-far from having stable rank $\le d$.

Fix $Q \subseteq \{1,\dots,n\}$ with $|Q| = q$. Let $A_Q = (A_{ij})_{i\in Q, 1\le j\le n}$ and define B_Q similarly. Each defines a distribution on $q \times n$ matrices, denoted by $\mathcal{L}(A_Q)$ and $\mathcal{L}(B_Q)$, respectively. Also denote by $B(n,p)$ the binomial distribution of n trials and success probability p.

LEMMA 13. *The Hellinger distance between two binomial distributions is given by*

$$d_H(B(n,p),B(n,q)) = \sqrt{1-(\sqrt{pq}+\sqrt{(1-p)(1-q)})^n}.$$

LEMMA 14. *Suppose that $d \ge 2$. When $q \le \alpha^2 d/(18\epsilon^2)$, it holds that $d_{TV}(\mathcal{L}(A_Q),\mathcal{L}(B_Q)) \le \alpha + o(1)$.*

PROOF. Observe that A_Q and B_Q contain the same number of rows of x_0 then the conditional distributions are the same. Note that the distance between $\mathcal{L}(A_Q)$ and $B(q,1/d)$ is $o(1)$ and a similar result holds for $\mathcal{L}(B_Q)$ and $B(q,\frac{1-3\epsilon}{d})$. Therefore using that $\sqrt{2}$ times the Hellinger distance is at least as large as the variation distance, we have,

$$\begin{aligned}
&d_{TV}(\mathcal{L}(A_Q),\mathcal{L}(B_Q))\\
&= d_{TV}(B(q,\frac{1}{d}),B(q,\frac{1-3\epsilon}{d})) + o(1)\\
&\le \sqrt{2}d_H(B(q,\frac{1}{d}),B(q,\frac{1-3\epsilon}{d})) + o(1)\\
&\le 2\left(1-\left(\sqrt{\frac{1-3\epsilon}{d^2}}+\sqrt{\left(1-\frac{1}{d}\right)\left(1-\frac{1}{d}+\frac{3\epsilon}{d}\right)}\right)^q\right) + o(1)
\end{aligned}$$

It is not difficult to verify that

$$\sqrt{\frac{1-3\epsilon}{d^2}}+\sqrt{\left(1-\frac{1}{d}\right)\left(1-\frac{1}{d}+\frac{3\epsilon}{d}\right)} \ge 1-\frac{9\epsilon^2}{d}$$

whenever $d \ge 2$ and $0 < \epsilon < \frac{1}{3}$. Therefore, it holds that

$$d_{TV}(\mathcal{L}(A_Q),\mathcal{L}(B_Q)) \le \sqrt{2\cdot\frac{9q\epsilon^2}{d}} + o(1) \le \alpha + o(1).$$

whenever $q \le \frac{\alpha^2}{18}\cdot\frac{d}{\epsilon^2}$. □

The lower bound follows immediately as a corollary.

COROLLARY 2. *Suppose that $d \ge 2$. Under the row sampling model, any algorithm that is correct on the* StableRank *problem needs to sample $\Omega(d/\epsilon^2)$ rows.*

7. EMPIRICAL RESULTS

All programs are written in MATLAB and the source code can be found at `http://www.mpi-inf.mpg.de/~yli/codes.pdf`.

7.1 Stable Rank Testing

Algorithm 3 takes $\tilde{O}(\frac{d}{\epsilon^2}\log n)$ row samples with a theoretical guarantee, however, a literal interpretation of the bound makes it less useful in practice, since for $d=2$, $\epsilon=0.1$, it holds that $d\log n/\epsilon^2 > n$ for $n \le 1500$. Indeed, the theoretical upper bound is too pessimistic, i.e., very often we do not need so many samples for real data sets. We justify our thoughts in the following experiment.

We test our algorithm with the University of Florida Sparse Matrix Collection[1]. There are 628 square real matrices with dimension between 100 and 1000 (inclusive). Among them, there are 220 matrices at least 0.1-far from having stable rank 2. There are also 35 square matrices with stable rank ≤ 2. For each matrix A of the 255 matrices, we determine the minimum q such that our algorithm, when sampling q rows at random, succeeds with probability ≥ 0.9 in distinguishing whether its stable rank is at most 2, or it is at least 0.1-far from having stable rank 2. The probability is determined by 100 independent trials. The cumulative distribution of q/n (where n is the dimension of A) is plotted in Figure 2. We can see that our algorithm needs to sample only at most 15% of the rows for 90% of the matrices. The remaining 10% have relatively small stable rank and it is natural to expect that more rows are needed.

We conducted similar experiments for $d=3$ and $d=5$, too. The results are also plotted in Figure 2. Regarding $d=3$, there are 174 matrices at least 0.1-far from having stable rank 3 and 67 matrices with stable rank ≤ 3. We ran our algorithm on each of the 81 matrices and plotted the cumulative distribution of q/n. We can see that our algorithm needs to sample only at most 15% of the rows for 90% of the 241 matrices. Regarding $d=5$, there are 105 matrices at least 0.1-far from having stable rank 5 and 161 matrices with stable rank ≤ 5. We can see that our algorithm needs to sample only at most 10% of the rows for 90%, and 15% of the rows for 95%, of the 266 matrices.

7.2 Rank Testing

We have seen there is a gap of a $1/\epsilon$ factor in the theoretical results between $O(d^2/\epsilon^2)$ samples for the non-adaptive

[1] `http://www.cise.ufl.edu/research/sparse/matrices/`

algorithm and $O(d^2/\epsilon)$ samples for the adaptive one. As above, both bounds could be too pessimistic as well. Thus we design the following experiments to show that the adaptive tester has a real advantage over the non-adaptive algorithm even when both algorithms read much fewer samples than the respective theoretical upper bound.

We conducted three sets of experiments on different matrix distributions as follows.

- strip distribution:

$$A = P \begin{pmatrix} \sum_{j=1}^{d+1} x_j y_j^T & 0 \\ 0 & 0 \end{pmatrix} Q,$$

  where $x_1, \ldots, x_{d+1}$ are i.i.d. $N(0, I_{\epsilon n})$ vectors, $y_1, \ldots, y_{d+1}$ are i.i.d. $N(0, I_n)$ vectors, P and Q are independent random $n \times n$ permutation matrices.

- rectangular distribution:

$$A = P \begin{pmatrix} \sum_{j=1}^{d+1} x_j y_j^T & 0 \\ 0 & 0 \end{pmatrix} Q,$$

  where $x_1, \ldots, x_{d+1}$ are i.i.d. $N(0, I_{n/2^{i-1}})$ vectors and $y_1, \ldots, y_{d+1}$ are i.i.d. $N(0, I_{\epsilon n 2^i})$ vectors, i is chosen uniformly at random from $\{1, \ldots, \lfloor \log(1/\epsilon) \rfloor\}$, P and Q are independent random $n \times n$ permutation matrices.

- square distribution:

$$A = P \begin{pmatrix} \sum_{j=1}^{d+1} x_j x_j^T & 0 \\ 0 & 0 \end{pmatrix} Q,$$

  where $x_1, \ldots, x_{d+1}$ are i.i.d. $N(0, I_{\lfloor \sqrt{\epsilon} n \rfloor})$ vectors, P and Q are independent random $n \times n$ permutation matrices.

In each case it holds that $\text{rank}(A) = d + 1$ with probability 1. We consider three cases of d: $d = 1, 2, 5$. For both the strip and the square distribution, we set $n = 1000$ and $\epsilon = 0.01$, 0.03, 0.05, 0.07, 0.09, 0.11, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.5; for the rectangular distribution we set $n = 1024$ and $\epsilon = 1/128$, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2. For each configuration of d and ϵ and each matrix distribution, we ran both the non-adaptive query algorithm [8] and the adaptive query algorithm (Algorithm 1) for 1000 times independently to obtain the number of queries needed to conclude $\text{rank}(A) > d$ with a success probability of at least 0.9. The results are shown in Figure 1 in logarithmic scale.

In all settings above, adaptive queries outperform non-adaptive ones, and particularly heavily for small ϵ. It is also notable that the strip distribution is especially adversarial for the non-adaptive tester, which needs to makes at least $1/\epsilon^2$ queries. When $\epsilon = 0.01$, the number of adaptive queries is only 7.1%, 9.4%, 12.4% of that of non-adaptive queries for $d = 1$, 2, 5, respectively. Even when $\epsilon = 0.5$, the number of adaptive queries is less than 1/3 of that of non-adaptive queries. The difference between non-adaptive and adaptive queries is less pronounced under the other two distributions, still the number of adaptive queries is at most a half of that of non-adaptive ones.

8. ACKNOWLEDGEMENTS

David Woodruff would like to acknowledge the support from the XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C0323.

9. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 2003.

[2] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):11, 2011.

[3] C. H. Q. Ding and X. He. *K*-means clustering via principal component analysis. In *ICML*, 2004.

[4] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[5] O. Goldreich. A brief introduction to property testing. In *Studies in Complexity and Cryptography*, pages 465–469. 2011.

[6] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[7] I. T. Jolliffe. *Graphical Representation of Data Using Principal Components*. Springer, 2002.

[8] R. Krauthgamer and O. Sasson. Property testing of data dimensionality. In *SODA*, pages 18–27, 2003.

[9] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 2001.

[10] M. Magdon-Ismail. Row sampling for matrix algorithms via a non-commutative bernstein bound. arXiv:1008.0587, 2010.

[11] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

[12] M. Parnas and D. Ron. Testing metric properties. In *STOC*, pages 276–285, 2001.

[13] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[14] M. Rudelson and R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4), July 2007.

[15] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems ï£¡ a case study. In *Proceedings of the ACM WebKDD Workshop*, 2000.

[16] J. A. Tropp. Column subset selection, matrix factorization, and eigenvalue optimization. In *SODA*, pages 978–986, 2009.

[17] C. Yang, T. Han, L. Quan, and C.-L. Tai. Parsing façade with rank-one approximation. In *CVPR*, pages 1720–1727, 2012.

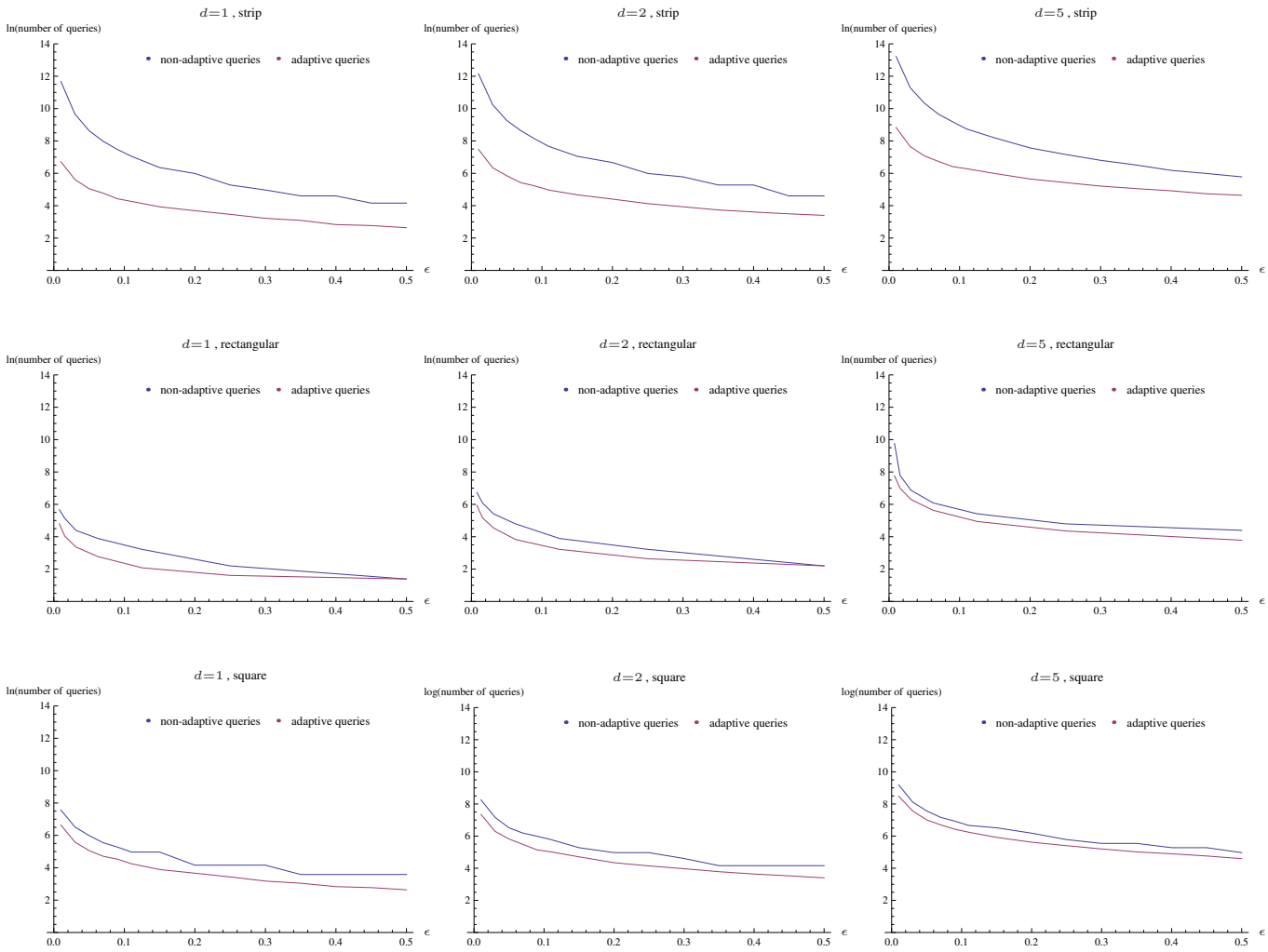

Figure 1: Experiment results for rank. The first row corresponds to the case where A is subject to the strip distribution, the second row the rectangular distribution and the third row the square distribution.

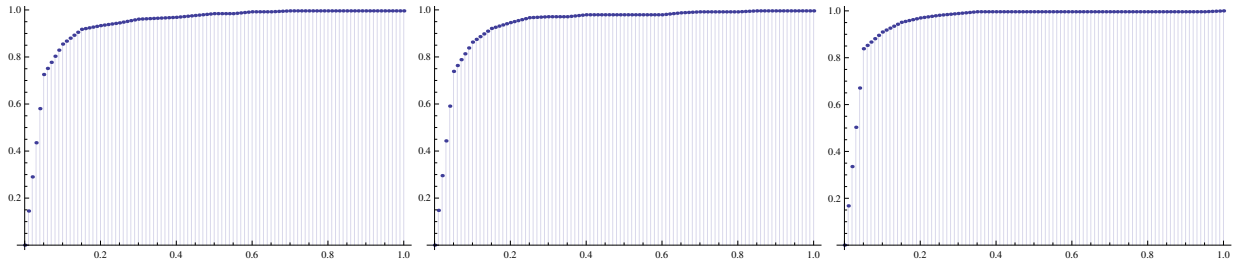

Figure 2: Experiment results for stable rank under row access model. The horizontal axis represents q/n, the percentage of rows sampled. The vertical axis is percentage of the tested matrices for which the algorithm succeeds with probability ≥ 0.9 at the corresponding sampling rate. The three plots correspond to $d = 2, 3, 5$, respectively, from left to right.