Online Row Sampling

Michael B. Cohen*1, Cameron Musco†2, and Jakub Pachocki‡3

1 Massachusetts Institute of Technology, Cambridge, MA, USA
micohen@mit.edu
2 Massachusetts Institute of Technology, Cambridge, MA, USA
cnmusco@mit.edu
3 Carnegie Mellon University, Pittsburgh, PA, USA
pachocki@cs.cmu.edu

Abstract

Finding a small spectral approximation for a tall $n \times d$ matrix $\mathbf{A}$ is a fundamental numerical primitive. For a number of reasons, one often seeks an approximation whose rows are sampled from those of $\mathbf{A}$. Row sampling improves interpretability, saves space when $\mathbf{A}$ is sparse, and preserves row structure, which is especially important, for example, when $\mathbf{A}$ represents a graph.

However, correctly sampling rows from $\mathbf{A}$ can be costly when the matrix is large and cannot be stored and processed in memory. Hence, a number of recent publications focus on row sampling in the streaming setting, using little more space than what is required to store the outputted approximation [12, 11].

Inspired by a growing body of work on online algorithms for machine learning and data analysis, we extend this work to a more restrictive *online* setting: we read rows of $\mathbf{A}$ one by one and immediately decide whether each row should be kept in the spectral approximation or discarded, without ever retracting these decisions. We present an extremely simple algorithm that approximates $\mathbf{A}$ up to multiplicative error ϵ and additive error δ using $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ online samples, with memory overhead proportional to the cost of storing the spectral approximation. We also present an algorithm that uses $\mathcal{O}(d^2)$ memory but only requires $\mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ samples, which we show is optimal.

Our methods are clean and intuitive, allow for lower memory usage than prior work, and expose new theoretical properties of leverage score based matrix approximation.

1998 ACM Subject Classification F.2.1 [Numerical Algorithms and Problems] Computations on Matrices, F.1.2 [Modes of Computation] Online computation

Keywords and phrases spectral sparsification, leverage score sampling, online sparsification

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.7

1 Introduction

1.1 Background

A spectral approximation to a tall $n \times d$ matrix $\mathbf{A}$ is a smaller, typically $\tilde{\mathcal{O}}(d) \times d$ matrix $\tilde{\mathbf{A}}$ such that $\|\tilde{\mathbf{A}}\mathbf{x}\|_2 \approx \|\mathbf{A}\mathbf{x}\|_2$ for all $\mathbf{x}$. Typically one asks for a multiplicative approximation, which guarantees that $(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2$. In other notation,

$$(1-\epsilon)\mathbf{A} \preceq \tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}.$$

[*] Michael B. Cohen is partially supported by NSF grant CCF-1111109.
[†] Cameron Musco is partially supported by NSF Graduate Research Fellowship No. 1122374, AFOSR grant FA9550-13-1-0042 and the NSF Center for Science of Information.
[‡] Jakub Pachocki is partially supported by NSF grant CCF-1065106.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).
Editors: Klaus Jansen, Claire Matthieu, José D. P. Rolim, and Chris Umans; Article No. 7; pp. 7:1–7:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Such approximations have many applications, most notably for solving least squares regression over $\mathbf{A}$ [6, 8]. If $\mathbf{A}$ is the vertex edge incidence matrix of a graph, $\tilde{\mathbf{A}}$ is a *spectral sparsifier* [20]. It can be used to approximate effective resistances, spectral clustering, mixing time and random walk properties, and many other computations.

A number of recent papers focus on fast algorithms for spectral approximation. Using sparse random subspace embeddings [6, 18, 17], it is possible to find $\tilde{\mathbf{A}}$ in input sparsity time, essentially by randomly recombining the rows of $\mathbf{A}$ into a smaller number of rows. In some cases these embeddings are not enough, as it is desirable for the rows of $\tilde{\mathbf{A}}$ to be a subset of rows sampled from $\mathbf{A}$. If $\mathbf{A}$ is sparse, this ensures that $\tilde{\mathbf{A}}$ is also sparse. If $\mathbf{A}$ represents a graph, it ensures that $\tilde{\mathbf{A}}$ is also a graph, specifically a weighted subgraph of the original.

It is well known that sampling $\mathcal{O}(d \log d/\epsilon^2)$ rows of $\mathbf{A}$ with probabilities proportional to their *leverage scores* yields a $(1+\epsilon)$ multiplicative factor spectral approximation to $\mathbf{A}$. Further, this sampling can be done in input sparsity time, either using subspace embeddings to approximate leverage scores, or using iterative sampling techniques [15], some that only work with subsampled versions of the original matrix [8].

1.2 Streaming and Online Row Sampling

When $\mathbf{A}$ is very large, input sparsity runtimes are not enough – memory restrictions also become important. Hence, recent work has tackled row sampling in a streaming model of computation. [12] presents a simple algorithm for sampling rows from an insertion only stream, using space approximately proportional to the size of the final approximation. [11] gives a sparse-recovery based algorithm that works in dynamic streams with row insertions and deletions, also using nearly optimal space. Unfortunately, to handle dynamic streams, the algorithm in [11] is complex, requires additional restrictions on the input matrix, and uses significantly suboptimal runtime to recover a spectral approximation from its low memory representation of the input stream.

While the algorithm in [12] is simple and efficient, we believe that its proof is incomplete, and do not see an obvious way to fix it. The main idea behind the algorithm is to sample rows by their leverage scores with respect to the stream seen so far. These leverage scores may be coarse overestimates of the true scores. However as more rows are streamed in, better estimates can be obtained and the sampled rows pruned to a smaller set. Unfortunately, the probability of sampling a row becomes dependent on which other rows are sampled. This seems to break the argument in that paper, which essentially claims that their process has the same distribution as would a single round of leverage score sampling.

In this paper we initiate the study of row sampling in an *online setting*. As in an insertion stream, we read rows of $\mathbf{A}$ one by one. However, upon seeing a row, we immediately decide whether it should be kept in the spectral approximation or discarded, without ever retracting these decisions. We present a similar algorithm to [12], however, since we never prune previously sampled rows, the probability of sampling a row only depends on whether previous rows in the stream were sampled. This limited dependency structure allows us to rigorously argue that a spectral approximation is obtained.

In addition to addressing gaps in the literature on streaming spectral approximation, our restricted model extends work on online algorithms for a variety of other machine learning and data analysis problems, including principal component analysis [4], clustering [16], classification [3, 10], and regression [10]. In practice, online algorithms are beneficial since they can be highly computationally and memory efficient. Further, they can be applied in scenarios in which data is produced in a continuous stream and intermediate results must be output as the stream is processed. Spectral approximation is a widely applicable primitive for approximate learning and computation, so studying its implementation in an online setting is a natural direction.

1.3 Our Results

Our primary contribution is a very simple algorithm for leverage score sampling in an online manner. The main difficultly with row sampling using leverage scores is that leverage scores themselves are not easy to compute. They are given by $l_i = \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{a}_i$, and so require solving systems in $\mathbf{A}^T\mathbf{A}$ if computed naively. This is not only expensive, but also impossible in an online setting, where we do not have access to all of $\mathbf{A}$.

A critical observation is that it always suffices to sample rows by overestimates of their true leverage scores. The number of rows that must be sampled is proportional to the sum of these overestimates. Since the leverage score of a row can only go up when we remove rows from the matrix, a simple way to obtain an overestimate is to compute leverage score using just a subset of the other rows of $\mathbf{A}$. That is, letting $\mathbf{A}_j$ contain just j of $\mathbf{A}$'s n rows, we can overestimate l_i by $\tilde{l}_i = \mathbf{a}_i^T(\mathbf{A}_j^T\mathbf{A}_j)^{-1}\mathbf{a}_i$

[8] shows that if $\mathbf{A}_j$ is a subset of rows sampled uniformly at random, then the expected leverage score of $\mathbf{a}_i$ is d/j. This simple fact immediately gives a result for online sampling from a *randomly ordered stream.* If we compute the leverage score of the current row $\mathbf{a}_i$ against all previously seen rows (or some approximation to these rows), then the expected sum of our overestimates is bounded by $d + d/2 + ... + ... + d/n = \mathcal{O}(d \log n)$. So, sampling $\mathcal{O}(d \log d \log n/\epsilon^2)$ rows is enough obtain a $(1+\epsilon)$ multiplicative factor spectral approximation.

What if we cannot guarantee a randomly ordered input stream? Is there any hope of being able to compute good leverage score estimates in an online manner? Surprisingly the answer to this is yes - we can in fact run nearly the exact same algorithm and be guaranteed that the sum of estimated leverage scores is low, *regardless of stream order.* Roughly, each time we receive a row which has high leverage score with respect to the previous rows, it must compose a significant part of $\mathbf{A}$'s spectrum. If $\mathbf{A}$ does not continue to grow unboundedly, there simply cannot be too many of these significant rows.

Specifically, we show that if we sample by the *ridge leverage scores* [1] over all previously seen rows, which are the leverage scores computed over $\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I}$ for some small regularizing factor λ, then with just $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ samples we obtain a $(1+\epsilon)$ multiplicative, δ additive error spectral approximation. That is, with high probability we sample a matrix $\tilde{\mathbf{A}}$ with $(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$.

To gain intuition behind this bound, note that we can convert it into a multiplicative one by setting $\delta = \epsilon\sigma_{min}(\mathbf{A})^2$ (as long as we have some estimate of $\sigma_{min}(\mathbf{A})$). This setting of δ will require taking $\mathcal{O}(d \log d \log(\kappa(\mathbf{A}))/\epsilon^2)$ samples. If we have a polynomial bound on the condition number of $\mathbf{A}$, as we do, for instance, for graphs with polynomially bounded edges weights, this becomes $\mathcal{O}(d \log^2 d/\epsilon^2)$ – nearly matching the $\mathcal{O}(d \log d/\epsilon^2)$ achievable if sampling by true leverage scores.

Our online sampling algorithm is extremely simple. When each row comes in, we compute the online ridge leverage score, or an estimate of it, and then irrevocably either add the row to our approximation or remove it. As mentioned, it is similar in form to the streaming algorithm of [12], except that it does not require pruning previously sampled rows. This allows us to avoid difficult dependency issues. Additionally, without pruning, we do not even need to store all previously sampled rows. As long as we store a constant factor spectral approximation our previous samples, we can compute good approximations to the online ridge leverage scores. In this way, we can store just $\mathcal{O}(d \log d \log(\epsilon\|\mathbf{A}\|_2^2/\delta))$ rows in working memory ($\mathcal{O}(d \log^2 d)$ if we want a spectral graph sparsifier), filtering our input stream into an $\mathcal{O}(d \log d \log(\kappa(\mathbf{A}))/\epsilon^2)$ sized output stream. Note that this memory bound in fact *improves* as ϵ decreases, and regardless, can be significantly smaller than the output size of the algorithm.

In addition to our main sampling result, we use our bounds on online ridge leverage score approximations to show that an algorithm in the style of [2] allows us to remove a $\log d$ factor and sample just $\mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows (Theorem 10). This algorithm is more complex and can require $\mathcal{O}(d^2)$ working memory. However, in Theorem 12 we show that it is asymptotically optimal. The $\log(\epsilon\|\mathbf{A}\|_2^2/\delta)$ factor is not an artifact of our analysis, but is truly the cost of the restricting ourselves to online sampling. No algorithm can obtain a multiplicative $(1+\epsilon)$ additive δ spectral approximation taking fewer than $\Omega(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows in an online manner.

2 Overview

Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. A natural approach to row sampling from $\mathbf{A}$ is picking an *a priori* probability with which each row is kept, and then deciding whether to keep each row independently. A common choice is for the sampling probabilities to be proportional to the *leverage scores* of the rows. The leverage score of the i-th row of $\mathbf{A}$ is defined to be

$$\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A})^{\dagger}\mathbf{a}_i,$$

where the dagger symbol denotes the pseudoinverse. In this work, we will be interested in approximating $\mathbf{A}^T\mathbf{A}$ with some (very) small multiple of the identity added. Hence, we will be interested in the λ-*ridge leverage scores* [1]:

$$\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i,$$

for a parameter $\lambda > 0$.

In many applications, obtaining the (nearly) exact values of $\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$ for sampling is difficult or outright impossible. A key idea is that as long as we have a sequence $l_1, \ldots, l_n$ of *overestimates* of the λ-ridge leverage scores, that is for $i = 1, \ldots, n$

$$l_i \geq \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i,$$

we can sample by these overestimates and obtain rigorous guarantees on the quality of the obtained spectral approximation. This notion is formalized in Theorem 1.

▶ **Theorem 1.** *Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Let $\epsilon \in (0,1), \delta > 0, \lambda := \delta/\epsilon, c := 8 \log d/\epsilon^2$. Assume we are given $l_1, \ldots, l_n$ such that for all $i = 1, \ldots, n$,*

$$l_i \geq \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i.$$

For $i = 1, \ldots, n$, let $p_i := \min(cl_i, 1)$. Construct $\tilde{\mathbf{A}}$ by independently sampling each row $\mathbf{a}_i$ of $\mathbf{A}$ with probability p_i, and rescaling it by $1/\sqrt{p_i}$ if it is included in the sample. Then, with high probability,

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}\left((\sum_{i=1}^n l_i) \log d/\epsilon^2\right)$.

Proof. This sort of guarantee for leverage score sampling is well known. See for example Lemma 4 of [8]. If we sampled both the rows of $\mathbf{A}$ and the rows of $\sqrt{\lambda}\mathbf{I}$ with the leverage scores over $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$, we would have $(1-\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$. However, we do not sample the rows of the identity. Since we could have sampled them each with probability 1, we can simply subtract $\lambda\mathbf{I} = (\delta/\epsilon)\mathbf{I}$ from the multiplicative bound and have: $(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$. ◀

The idea of using overestimates of leverage scores to perform row sampling has been applied successfully to various problems (see e.g. [13, 8]). However, in these applications, access to the entire matrix is required beforehand. In the streaming and online settings, we have to rely on partial data to approximate the true leverage scores. The most natural idea is to just use the portion of the matrix seen thus far as an approximation to $\mathbf{A}$. This leads us to introduce the *online λ-ridge leverage scores*:

$$l_i := \min(\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1),$$

where $\mathbf{A}_i$ $(i = 0, \ldots, n)$ is defined as the matrix consisting of the first i rows of $\mathbf{A}$[1].

Since clearly $\mathbf{A}_i^T\mathbf{A}_i \preceq \mathbf{A}^T\mathbf{A}$ for all i, it is not hard to see that l_i does overestimate the true λ-ridge leverage score for row $\mathbf{a}_i$. A more complex question, however, is establishing an upper bound on $\sum_{i=1}^n l_i$ so that we can bound the number of samples needed by Theorem 1.

A core result of this work, stated in Theorem 2, is establishing such an upper bound; in fact, this bound is shown to be tight up to constants (Theorem 12) and is nearly-linear in most cases.

▶ **Theorem 2.** *Let $\mathbf{A}$ be an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Let $\mathbf{A}_i$ for $i \in \{0, \ldots, n\}$ be the matrix consisting of the first i rows of $\mathbf{A}$. For $\lambda > 0$, let*

$$l_i := \min(\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1).$$

be the online λ-ridge leverage score of the i^{th} row of $\mathbf{A}$. Then

$$\sum_{i=1}^n l_i = \mathcal{O}(d\log(\|\mathbf{A}\|_2^2/\lambda)).$$

Theorems 2 and 1 suggest a simple algorithm for online row sampling: simply use the online λ-ridge leverage scores, for $\lambda := \delta/\epsilon$. This produces a spectral approximation with only $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows. Unfortunately, computing l_i exactly requires us to store *all* the rows we have seen in memory (or alternatively to store the sum of their outer products, $\mathbf{A}_i^T\mathbf{A}_i$). In many cases, such a requirement would defeat the purpose of streaming row sampling.

A natural idea is to use the sample we have kept thus far as an approximation to $\mathbf{A}_i$ when computing l_i. It turns out that the approximate online ridge leverage scores $\tilde{l}_i$ computed in this way will not always be good approximations to l_i; however, we can still prove that they satisfy the requisite bounds and yield the same row sample size! We formalize these results in the algorithm Online-Sample (Figure 1) and Theorem 3.

▶ **Theorem 3.** *Let $\tilde{\mathbf{A}}$ be the matrix returned by* Online-Sample$(\mathbf{A}, \epsilon, \delta)$. *With high probability,*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.

[1] We use the proposed scores l_i for simplicity, however note that the following, perhaps more natural, definition of online leverage scores would also be effective: $l_i' := \mathbf{a}_i^T(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I})^{-1}\mathbf{a}_i$.

```
$\tilde{\mathbf{A}}$ = ONLINE-SAMPLE($\mathbf{A}, \epsilon, \delta$), where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$,
$\epsilon \in (0,1)$, $\delta > 0$.
1. Set $\lambda := \delta/\epsilon$, $c := 8\log d/\epsilon^2$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix.
3. For $i = 1, \ldots, n$:
   a. Let $\tilde{l}_i := \min((1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1}+\lambda\mathbf{I})^{-1}\mathbf{a}_i, 1)$.
   b. Let $p_i := \min(c\tilde{l}_i, 1)$.
   c. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix}\tilde{\mathbf{A}}_{i-1}\\ \mathbf{a}_i/\sqrt{p_i}\end{bmatrix} & \text{with probability } p_i,\\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.}\end{cases}$
4. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 1 The basic online sampling algorithm.

To save computation, we note that, with a small modification to our analysis, we can run ONLINE-SAMPLE with batch processing of rows. Specifically, say we start from the i^{th} position in the stream. we can store the next $b = \mathcal{O}(d)$ rows. We can then compute sampling probabilities for these rows all at once using a system solver for $(\tilde{\mathbf{A}}_{i+b}^T\tilde{\mathbf{A}}_{i+b} + \lambda\mathbf{I})$. Using a trick introduced in [19], by applying a Johnson-Lindenstrauss random projection to the rows whose scores we are computing, we need just $\mathcal{O}(\log(1/\delta))$ system solves to compute constant factor approximations to the ridge scores with probability $1-\delta$. If we set $\delta = 1/\text{poly}(n)$ then we can union bound over our whole stream, using this trick with each batch of $\mathcal{O}(d)$ input rows. The batch probabilities will only be closer to the true ridge leverage scores than the non-batch probabilities and we will enjoy the same guarantees as ONLINE-SAMPLE.

Additionally, it turns out that with a simple trick, it is possible to reduce the memory usage of the algorithm by a factor of ϵ^{-2}, bringing it down to $\mathcal{O}(d\log d\log(\epsilon\|A\|_2^2/\delta))$ (assuming the row sample is output to an output stream). Note that this expression gets *smaller* with ϵ; hence we obtain a row sampling algorithm with memory complexity independent of desired multiplicative precision. The basic idea is that, instead of keeping all previously sampled rows in memory, we store a smaller set of rows that give a constant factor spectral approximation, still enough to give good estimates of the online ridge leverage scores.

This result is presented in the algorithm SLIM-SAMPLE (Figure 2) and Lemma 9. A particularly interesting consequence for graphs with polynomially bounded edge weights is:

▶ **Corollary 4.** *Let G be a simple graph on d vertices, and $\epsilon \in (0,1)$. We can construct a $(1+\epsilon)$-sparsifier of G of size $\mathcal{O}(d\log^2 d/\epsilon^2)$, using only $\mathcal{O}(d\log^2 d)$ working memory in the online model.*

Proof. This follows simply from applying Theorem 3 with $\delta = \epsilon/\sigma_{min}^2(\mathbf{A})$. For an unweighted graph on d vertices, $\|\mathbf{A}\|_2^2 \leq d$, since d is the largest squared singular value of the complete graph. Combining with Lemma 6.1 of [21], we have that the condition number of a graph on d vertices whose edge weights are within a multiplicative $\text{poly}(d)$ of each other is polynomial in d. So $\log(\epsilon\|\mathbf{A}\|_2^2/\delta) = \log(\kappa^2(\mathbf{A})) = \mathcal{O}(\log d)$. ◀

We remark that the algorithm of Corollary 4 can be made to run in nearly linear time in the stream size. We combine SLIM-SAMPLE with the batch processing idea described above. Because $\mathbf{A}$ is a graph, our matrix approximation is always a symmetric diagonally dominant matrix, with $\mathcal{O}(d)$ nonzero entries. We can solve systems in it in time $\tilde{\mathcal{O}}(d)$. Using the

Johnson-Lindenstrauss random projection trick of [19], we can compute approximate ridge leverage scores for a batch of $\mathcal{O}(d)$ rows with failure probability polynomially small in n in $\tilde{\mathcal{O}}(d \log n)$ time. Union bounding over the whole stream, we obtain nearly linear runtime.

To complement the row sampling results discussed above, we explore the limits of the proposed online setting. In Section 4 we present the algorithm Online-BSS, which obtains spectral approximations with $\mathcal{O}(d \log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$ rows in the online setting (with larger memory requirements than the simpler sampling algorithms). Its analysis is given in Theorem 10. In Section 5, we show that this number of samples is in fact the best achievable, up to constant factors (Theorem 12). The $\log(\epsilon\|\mathbf{A}\|_2^2/\delta)$ factor is truly the cost of requiring rows to be selected in an online manner.

3 Analysis of Sampling Schemes

We begin by bounding the sum of online λ-ridge leverage scores. The intuition behind the proof of Theorem 2 is that whenever we add a row with a large online leverage score to a matrix, we increase its determinant significantly, as follows from the matrix determinant lemma (Lemma 5). Thus we can reduce upper bounding the online leverage scores to bounding the matrix determinant.

▶ **Lemma 5** (Matrix determinant lemma). *Assume* $\mathbf{S}$ *is an invertible square matrix and* $\mathbf{u}$ *is a vector. Then*

$$\det(\mathbf{S} + \mathbf{u}\mathbf{u}^T) = (\det \mathbf{S})(1 + \mathbf{u}^T\mathbf{S}^{-1}\mathbf{u}).$$

Proof of Theorem 2. By Lemma 5, we have

$$\begin{aligned}\det(\mathbf{A}_{i+1}^T\mathbf{A}_{i+1} + \lambda\mathbf{I}) &= \det(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I}) \cdot \left(1 + \mathbf{a}_{i+1}^T(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I})^{-1}\mathbf{a}_{i+1}\right) \\ &\geq \det(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I}) \cdot (1 + l_{i+1}) \\ &\geq \det(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{I}) \cdot e^{l_{i+1}/2}.\end{aligned}$$

Hence,

$$\begin{aligned}\det(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) &= \det(\mathbf{A}_n^T\mathbf{A}_n + \lambda\mathbf{I}) \\ &\geq \det(\lambda\mathbf{I}) \cdot e^{\sum l_i/2} \\ &= \lambda^d e^{\sum l_i/2}.\end{aligned}$$

We have $\det(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) \leq (\|\mathbf{A}\|_2^2 + \lambda)^d$. Therefore

$$(\|\mathbf{A}\|_2^2 + \lambda)^d \geq \lambda^d e^{\sum l_i/2}.$$

Taking logarithms of both sides, we obtain

$$\begin{aligned}d \log(\|\mathbf{A}\|_2^2 + \lambda) &\geq d \log \lambda + \sum l_i/2 \\ \sum l_i &\leq 2d \log(1 + \|\mathbf{A}\|_2^2/\lambda).\end{aligned}$$ ◀

We now turn to analyzing the algorithm Online-Sample. Because the samples taken by the algorithm are *not* independent, we are not able to use a standard matrix Chernoff bound like the one in Theorem 1. However, we do know that whether we take row i does not depend on later rows; thus, we are able to analyze the process as a martingale. We will use a matrix version of the Freedman inequality given by Tropp.

▶ **Theorem 6** (Matrix Freedman inequality [22]). *Let $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n$ be a matrix martingale whose values are self-adjoint matrices with dimension d, and let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be the difference sequence. Assume that the difference sequence is uniformly bounded in the sense that*

$$\|\mathbf{X}_k\|_2 \leq R \text{ almost surely, for } k = 1, \ldots, n.$$

Define the predictable quadratic variation process of the martingale:

$$\mathbf{W}_k := \sum_{j=1}^{k} \mathbf{E}_{j-1}\left[\mathbf{X}_j^2\right], \text{ for } k = 1, \ldots, n.$$

Then, for all $\epsilon > 0$ and $\sigma^2 > 0$,

$$\mathbf{P}\left[\|\mathbf{Y}_n\|_2 \geq \epsilon \text{ and } \|\mathbf{W}_n\|_2 \leq \sigma^2\right] \leq d \cdot \exp\left(-\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}\right)$$

We begin by showing that the output of Online-Sample is in fact an approximation of $\mathbf{A}$, and that the approximate online leverage scores are lower bounded by the actual online leverage scores.

▶ **Lemma 7.** *After running* Online-Sample*, it holds with high probability that*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and also

$$\tilde{l}_i \geq \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$$

for $i = 1, \ldots, n$.

Proof. Let

$$\mathbf{u}_i := (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}\mathbf{a}_i.$$

We construct a matrix martingale $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n \in \mathbb{R}^{d\times d}$ with the difference sequence $\mathbf{X}_1, \ldots, \mathbf{X}_n$. Set $\mathbf{Y}_0 = \mathbf{0}$. If $\|\mathbf{Y}_{i-1}\|_2 \geq \epsilon$, we set $\mathbf{X}_i := \mathbf{0}$. Otherwise, let

$$\mathbf{X}_i := \begin{cases} (1/p_i - 1)\mathbf{u}_i\mathbf{u}_i^T & \text{if } \mathbf{a}_i \text{ is sampled in } \tilde{\mathbf{A}}, \\ -\mathbf{u}_i\mathbf{u}_i^T & \text{otherwise.} \end{cases}$$

In the case that $\|\mathbf{Y}_{i-1}\|_2 < \epsilon$, by construction, $\|\mathbf{Y}_j\|_2 < \epsilon$ for all $j < i-1$. So we have:

$$\mathbf{Y}_{i-1} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} - \mathbf{A}_{i-1}^T\mathbf{A}_{i-1})(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1/2}.$$

Hence, we have

$$\begin{aligned} \tilde{l}_i &= \min((1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i, 1) \\ &\geq \min((1+\epsilon)\mathbf{a}_i^T(\mathbf{A}_{i-1}^T\mathbf{A}_{i-1} + \lambda\mathbf{I} + \epsilon(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}))^{-1}\mathbf{a}_i, 1) \\ &\geq \min((1+\epsilon)\mathbf{a}_i^T((1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}))^{-1}\mathbf{a}_i, 1) \\ &= \mathbf{a}_i^T(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{a}_i \qquad (1) \\ &= \mathbf{u}_i^T\mathbf{u}_i, \end{aligned}$$

and so $p_i \geq \min(c\mathbf{u}_i^T\mathbf{u}_i, 1)$. If $p_i = 1$, then $\mathbf{X}_i = 0$. Otherwise, we have $p_i \geq c\mathbf{u}_i^T\mathbf{u}_i$ and:

$$\|\mathbf{X}_i\|_2 \leq \max\{1, 1/p_i - 1\} \cdot \|\mathbf{u}_i\mathbf{u}_i^T\|_2 \leq \frac{1}{p_i}\mathbf{u}_i^T\mathbf{u}_i \leq 1/c. \tag{2}$$

Further

$$\begin{aligned}\mathbf{E}_{i-1}\left[\mathbf{X}_i^2\right] &\preceq p_i \cdot (1/p_i - 1)^2(\mathbf{u}_i\mathbf{u}_i^T)^2 + (1-p_i)\cdot(\mathbf{u}_i\mathbf{u}_i^T)^2\\ &= (\mathbf{u}_i\mathbf{u}_i^T)^2 \cdot (1-p_i)/p_i\\ &\preceq \mathbf{u}_i\mathbf{u}_i^T \cdot \left(\mathbf{u}_i^T\mathbf{u}_i/p_i\right)\\ &\preceq \mathbf{u}_i\mathbf{u}_i^T/c. && \text{(by equation (2))}\end{aligned}$$

And so, for the predictable quadratic variation process of the martingale $\{\mathbf{Y}_i\}$:

$$\mathbf{W}_i := \sum_{k=1}^{i}\mathbf{E}_{k-1}\left[\mathbf{X}_k^2\right],$$

we have

$$\|\mathbf{W}_i\|_2 \leq \left\|\sum_{k=1}^{i}\mathbf{u}_i\mathbf{u}_i^T/c\right\|_2 \leq 1/c.$$

Therefore by, Theorem 6, we have

$$\begin{aligned}\mathbf{P}\left[\|\mathbf{Y}_n\|_2 \geq \epsilon\right] &\leq d \cdot \exp\left(\frac{-\epsilon^2/2}{1/c + \epsilon/(3c)}\right)\\ &\leq d\cdot\exp(-c\epsilon^2/4)\\ &= 1/d.\end{aligned}$$

This implies that with high probability

$$\|(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1/2}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}+\lambda\mathbf{I})(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1/2} - \mathbf{I}\|_2 \leq \epsilon$$

and so

$$(1-\epsilon)(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I}) \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}}+\lambda\mathbf{I} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I}).$$

Subtracting $\lambda\mathbf{I} = (\delta/\epsilon)\mathbf{I}$ from all sides, we get

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A}+\delta\mathbf{I}.$$

Finally, note that, since we set $\mathbf{X}_i = \mathbf{0}$ if $\|\mathbf{Y}_{i-1}\|_2 \geq \epsilon$, $\|\mathbf{Y}_n\|_2 < \epsilon$ implies $\|\mathbf{Y}_i\|_2 < \epsilon$ for all $i < n$. We thus have the desired bound on $\tilde{l}_i$ by equation (1). ◀

If we set c in ONLINE-SAMPLE to be proportional to $\log n$ rather than $\log d$, we would be able to take a union bound over all the rows and guarantee that with high probability all the approximate online leverage scores $\tilde{l}_i$ are close to true online leverage scores l_i. Thus Theorem 2 would imply that ONLINE-SAMPLE only selects $\mathcal{O}(d\log n\log(\|\mathbf{A}\|_2^2/\lambda)/\epsilon^2)$ rows with high probability.

In order to remove the dependency on n, we have to sacrifice achieving close approximations to l_i at every step. Instead, we show that the *sum* of the computed approximate online leverage scores is still small with high probability, using a custom Chernoff bound.

▶ **Lemma 8.** *After running* Online-Sample*, it holds with high probability that*

$$\sum_{i=1}^{n} \tilde{l}_i = \mathcal{O}(d \log(\|\mathbf{A}\|_2^2/\lambda)).$$

Proof. Define

$$\delta_i := \log\det(\tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i + \lambda\mathbf{I}) - \log\det(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I}).$$

The proof closely follows the idea from the proof of Theorem 2. We will aim to show that large values of $\tilde{l}_i$ correlate with large values of δ_i. However, the sum of δ_i can be bounded by the logarithm of the ratio of the determinants of $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}$ and $\lambda\mathbf{I}$. First, we will show that $\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right]$ is always at most 1. We begin by an application of Lemma 5.

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &= p_i \cdot e^{\tilde{l}_i/8}(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i/p_i)^{-1} + (1-p_i)e^{\tilde{l}_i/8}\\ &\le p_i \cdot (1 + \tilde{l}_i/4)(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i/p_i)^{-1} + (1-p_i)(1+\tilde{l}_i/4).\end{aligned}$$

If $c\tilde{l}_i < 1$, we have $p_i = c\tilde{l}_i$ and $\tilde{l}_i = (1+\epsilon)\mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i$, and so:

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &\le c\tilde{l}_i \cdot (1+\tilde{l}_i/4)(1 + 1/((1+\epsilon)c))^{-1} + (1 - c\tilde{l}_i)(1+\tilde{l}_i/4)\\ &= (1+\tilde{l}_i/4)(c\tilde{l}_i(1+1/((1+\epsilon)c))^{-1} + 1 - c\tilde{l}_i)\\ &\le (1+\tilde{l}_i/4)(1 + c\tilde{l}_i(1 - 1/(4c) - 1))\\ &= (1+\tilde{l}_i/4)(1-\tilde{l}_i/4)\\ &\le 1.\end{aligned}$$

Otherwise, we have $p_i = 1$ and so:

$$\begin{aligned}\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] &\le (1+\tilde{l}_i/4)(1 + \mathbf{a}_i^T(\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} + \lambda\mathbf{I})^{-1}\mathbf{a}_i)^{-1}\\ &\le (1+\tilde{l}_i/4)(1+\tilde{l}_i)^{-1}\\ &\le 1.\end{aligned}$$

We will now analyze the expected product of $\exp(\tilde{l}_i/8 - \delta_i)$ over the first k steps. We group the expectation over the first k steps into one over the first $k-1$ steps, aggregating the expectation for the last step by using one-way independence. For $k \ge 1$ we have

$$\begin{aligned}\mathbf{E}\left[\exp\left(\sum_{i=1}^{k}\tilde{l}_i/8 - \delta_i\right)\right] &= \mathop{\mathbf{E}}_{\text{first } k-1 \text{ steps}}\left[\exp\left(\sum_{i=1}^{k-1}\tilde{l}_i/8 - \delta_i\right)\mathbf{E}_{k-1}\left[\exp(\tilde{l}_k/8 - \delta_k)\right]\right]\\ &\le \mathbf{E}\left[\exp\left(\sum_{i=1}^{k-1}\tilde{l}_i/8 - \delta_i\right)\right],\end{aligned}$$

and so by induction on k

$$\mathbf{E}\left[\exp\left(\sum_{i=1}^{n}\tilde{l}_i/8 - \delta_i\right)\right] \le 1.$$

Hence by Markov's inequality

$$\mathbf{P}\left[\sum_{i=1}^{n}\tilde{l}_i > 8d + 8\sum_{i=1}^{n}\delta_i\right] \le e^{-d}.$$

```
$\tilde{\mathbf{A}} = $ SLIM-SAMPLE$(\mathbf{A}, \epsilon, \delta)$, where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$,
$\epsilon \in (0,1)$, $\delta > 0$.
1. Set $\lambda := \delta/\epsilon$, $c := 8 \log d/\epsilon^2$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix.
3. Let $\tilde{l}_1, \ldots, \tilde{l}_n$ be the approximate online leverage scores computed by an independent instance of ONLINE-SAMPLE$(\mathbf{A}, 1/2, \delta/(2\epsilon))$.
4. For $i = 1, \ldots, n$:
    a. Let $p_i := \min(c\tilde{l}_i, 1)$.
    b. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix} \tilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i/\sqrt{p_i} \end{bmatrix} & \text{with probability } p_i, \\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
5. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 2 The low-memory online sampling algorithm.

By Lemma 7, with high probability we have $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I} \preceq (1+\epsilon)(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})$. We also have with high probability:

$$\begin{aligned} \det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) &\leq (1+\epsilon)^d(\|\mathbf{A}\|_2^2 + \lambda)^d, \\ \log\det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) &\leq d(1 + \log(\|\mathbf{A}\|_2^2 + \lambda)). \end{aligned}$$

Hence, with high probability it holds that

$$\begin{aligned} \sum_{i=1}^n \delta_i &= \log\det(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}) - d\log(\lambda) \\ &\leq d(1 + \log(\|\mathbf{A}\|_2^2 + \lambda) - \log(\lambda)) \\ &= d(1 + \log(1 + \|\mathbf{A}\|_2^2/\lambda)). \end{aligned}$$

And so, with high probability,

$$\begin{aligned} \sum_{i=1}^n \tilde{l}_i &\leq 8d + 8\sum_{i=1}^n \delta_i \\ &\leq 9d + 8d\log(1 + \|\mathbf{A}\|_2^2/\lambda) \\ &= \mathcal{O}(d\log(\|\mathbf{A}\|_2^2/\lambda)). \end{aligned}$$ ◀

Proof of Theorem 3. The thesis follows immediately from Lemmas 7 and 8. ◀

We now consider a simple modification of ONLINE-SAMPLE that removes dependency on ϵ from the working memory usage with no additional cost.

▶ **Lemma 9.** *Let $\tilde{\mathbf{A}}$ be the matrix returned by* SLIM-SAMPLE$(\mathbf{A}, \epsilon, \delta)$. *Then, with high probability,*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \preceq \tilde{\mathbf{A}}^T\tilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I},$$

and the number of rows in $\tilde{\mathbf{A}}$ is $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.

Moreover, with high probability the algorithm SLIM-SAMPLE*'s memory requirement is dominated by storing $\mathcal{O}(d\log d\log(\epsilon\|\mathbf{A}\|_2^2/\delta))$ rows of $\mathbf{A}$.*

Proof. As the samples are independent, the thesis follows from Theorem 1 and Lemmas 7 and 8. ◀

4 Asymptotically Optimal Algorithm

In addition to sampling by online leverage scores, there is also a variant of the “BSS” method [2] that applies in our setting. Like the original [2], this approach removes the $\log d$ factor from the row count of the output spectral approximation, matching the lower bound for online sampling given in Theorem 12.

Unlike [2] itself, our algorithm is randomized – it is similar to, and inspired by, the randomized version of BSS from [14], especially the simpler “Algorithm 1” from that paper (the main difference from that is considering each row separately). In fact, this algorithm is of the same form as the basic sampling algorithm, in that when each row comes in, a probability p_i is assigned to it, and it is kept (and rescaled) with probability p_i and rejected otherwise. The key difference is the definition of the p_i.

There are also some differences in the nature of the algorithm and its guarantees. Notably, the p_i cannot be computed solely based on the row sample output so far–it is necessary to “remember” the entire matrix given so far. This means that the BSS method is not memory efficient, using $O(d^2)$ space. Additionally, online leverage score sampling gives bounds on both the size of the output spectral approximation and its accuracy with high probability. In contrast, this method gives an *expected* bound on the output size, while it *never* fails to output a correct spectral approximation. Note that these guarantees are essentially the same as those in the appendix of [14].

One may, however, improve the memory dependence in some cases simply by running it on the output stream of the online leverage score sampling method. This reduces the storage cost to the size of that spectral approximation. The BSS method still does not produce an actual space *savings* (in particular, there is a still a $\log d$ factor in space), but it does reduce the number of rows in the output stream while only blowing up the space usage by $O(1/\epsilon^2)$ (due to requiring the storage of an ϵ-quality approximation rather than only $O(1)$).

The BSS method maintains two matrices, $\mathbf{B}_i^U$ and $\mathbf{B}_i^L$, acting as upper and lower “barriers”. The current spectral approximation will always fall between them:

$$\mathbf{B}_i^L \prec \tilde{\mathbf{A}}_i^T \tilde{\mathbf{A}}_i^T \prec \mathbf{B}_i^U .$$

This guarantee, at the end of the algorithm, will ensure that $\tilde{\mathbf{A}}$ is a valid approximation.

Below, we give the actual BSS algorithm and its performance guarantees.

▶ **Theorem 10.**

1. *The online BSS algorithm always outputs $\tilde{A}$ such that*

$$(1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I} \prec \tilde{\mathbf{A}}^T\tilde{\mathbf{A}}^T \prec (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I} .$$

2. *The probability that a row $\mathbf{a}_i$ is included in $\tilde{\mathbf{A}}$ is at most $\frac{8}{\epsilon^2}l_i$, where l_i is the online $\frac{2\delta}{\epsilon}$-ridge leverage score of $\mathbf{a}_i$. That is $l_i = \min(\mathbf{a}_i^T\left(\mathbf{A}_i^T\mathbf{A}_i + \frac{2\delta}{\epsilon}I\right)^{-1}\mathbf{a}_i, 1)$. The expected number of rows in $\tilde{\mathbf{A}}$ is thus at most $\frac{8}{\epsilon^2}\sum_{i=1}^n l_i = \mathcal{O}(d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)/\epsilon^2)$.*

Proof of Theorem 10 part 1. We first note the basic invariant that $\mathbf{X}_i^U$ and $\mathbf{X}_i^L$ always remain positive definite–or equivalently,

$$\mathbf{B}_i^L \prec \tilde{\mathbf{A}}_i^T \tilde{\mathbf{A}}_i^T \prec \mathbf{B}_i^U .$$

We may prove this by induction on i. The base case follows from the initialization of $\tilde{\mathbf{A}}_0$, $\mathbf{B}_0^U$ and $\mathbf{B}_0^L$. For each successive step, we consider two possibilities.

```
$\tilde{\mathbf{A}} = \textsc{Online-BSS}(\mathbf{A}, \epsilon, \delta)$, where $\mathbf{A}$ is an $n \times d$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$,
$\epsilon \in (0, 1)$, $\delta > 0$.
1. Set $c_U = \frac{2}{\epsilon} + 1$ and $c_L = \frac{2}{\epsilon} - 1$.
2. Let $\tilde{\mathbf{A}}_0$ be a $0 \times d$ matrix, $\mathbf{B}_0^U = \delta\mathbf{I}$, $\mathbf{B}_0^L = -\delta\mathbf{I}$.
3. For $i = 1, \ldots, n$:
   a. Let $\mathbf{X}_{i-1}^U = (\mathbf{B}_{i-1}^U - \tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1})$, $\mathbf{X}_{i-1}^L = (\tilde{\mathbf{A}}_{i-1}^T\tilde{\mathbf{A}}_{i-1} - \mathbf{B}_{i-1}^L)$.
   b. Let $p_i := \min(c_U\mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i + c_L\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i, 1)$.
   c. Set $\tilde{\mathbf{A}}_i := \begin{cases} \begin{bmatrix} \tilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i/\sqrt{p_i} \end{bmatrix} & \text{with probability } p_i, \\ \tilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
   d. Set $\mathbf{B}_i^U = \mathbf{B}_{i-1}^U + (1+\epsilon)\mathbf{a}_i\mathbf{a}_i^T$, $\mathbf{B}_i^L = \mathbf{B}_{i-1}^L + (1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T$.
4. Return $\tilde{\mathbf{A}} := \tilde{\mathbf{A}}_n$.
```

Figure 3 The Online BSS Algorithm.

The first is that $p_i = 1$. In that case, $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ always increases by exactly $\mathbf{a}_i\mathbf{a}_i^T$, $\mathbf{B}^U$ by $(1+\epsilon)\mathbf{a}_i\mathbf{a}_i^T$ and $\mathbf{B}^L$ by $(1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T$. Thus $\mathbf{X}^U$ and $\mathbf{X}^L$ increase by exactly $\epsilon\mathbf{a}_i\mathbf{a}_i^T$, which is positive semidefinite, and so remain positive definite.

In the other case, $p_i < 1$. Now, $\mathbf{X}^U$ decreases by at most the increase in $\tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i^T$, or

$$\mathbf{M}_i = \frac{\mathbf{a}_i\mathbf{a}_i^T}{p}.$$

Since $c_U > 1$, $p > \mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i$, so $\mathbf{a}_i\mathbf{a}_i^T \prec p\mathbf{X}_{i-1}^U$ and $\mathbf{M}_i \prec \mathbf{X}_{i-1}^U$. Subtracting this then must leave $\mathbf{X}^U$ positive definite. Similarly, $\mathbf{X}^L$ decreases by at most the increase in $\mathbf{B}^L$, which is $(1-\epsilon)\mathbf{a}_i\mathbf{a}_i^T \prec \mathbf{a}_i\mathbf{a}_i^T$. Since $c_L > 1$ and $p < 1$, $\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i < 1$, and $\mathbf{a}_i\mathbf{a}_i^T \prec \mathbf{X}_{i-1}^L$. Subtracting this similarly leaves $\mathbf{X}^L$ positive definite. Finally, we note that

$$\mathbf{B}_n^U = (1+\epsilon)\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}$$
$$\mathbf{B}_n^L = (1-\epsilon)\mathbf{A}^T\mathbf{A} - \delta\mathbf{I}.$$

This gives the desired result. ◀

To prove part 2, we will use quantities of the form $\mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}$. We will need a lemma describing how this behaves under a random rank-1 update:

▶ **Lemma 11.** *Given a positive definite matrix $\mathbf{X}$, two vectors $\mathbf{u}$ and $\mathbf{v}$, two multipliers a and b and a probability p, define the random variable $\mathbf{X}'$ to be $X - a\mathbf{u}\mathbf{u}^T$ with probability p and $X - b\mathbf{u}\mathbf{u}^T$ otherwise. Then if $\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u} = 1$,*

$$\mathbf{E}\left[\mathbf{v}^T\mathbf{X}'^{-1}\mathbf{v} - \mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}] = (\mathbf{v}\mathbf{T}\mathbf{X}^{-1}\mathbf{u})^2\frac{pa + (1-p)b - ab}{(1-a)(1-b)}\right].$$

Proof. We apply the Sherman-Morrison formula to each of the two possibilities (subtracting $a\mathbf{u}\mathbf{u}^T$ and $b\mathbf{u}\mathbf{u}^T$ respectively). These give $\mathbf{X}'$ values of respectively

$$\mathbf{X}^{-1} + a\frac{\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}}{1 - a\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u}} = \mathbf{X}^{-1} + \frac{a}{1-a}\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}$$

and

$$\mathbf{X}^{-1} + b\frac{\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}}{1 - b\mathbf{u}^T\mathbf{X}^{-1}\mathbf{u}} = \mathbf{X}^{-1} + \frac{b}{1-b}\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}.$$

The values of $\mathbf{v}^T\mathbf{X}'^{-1}\mathbf{v} - \mathbf{v}^T\mathbf{X}^{-1}\mathbf{v}$ are then respectively

$$\frac{a}{1-a}\mathbf{v}^T\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}\mathbf{v} = (\mathbf{v}^{\mathbf{T}}\mathbf{X}^{-1}\mathbf{u})^2\frac{a}{1-a}$$

and

$$\frac{b}{1-b}\mathbf{v}^T\mathbf{X}^{-1}\mathbf{u}\mathbf{u}^T\mathbf{X}^{-1}\mathbf{v} = (\mathbf{v}^{\mathbf{T}}\mathbf{X}^{-1}\mathbf{u})^2\frac{b}{1-b}.$$

Combining these gives the stated result. ◀

Proof of Theorem 10 part 2. First, we introduce some new matrices to help in the analysis:

$$\begin{aligned}
\mathbf{C}^U_{i,j} &= \delta\mathbf{I} + \frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \left(1 + \frac{\epsilon}{2}\right)\mathbf{A}_j^T\mathbf{A}_j \\
\mathbf{C}^L_{i,j} &= -\delta\mathbf{I} - \frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \left(1 - \frac{\epsilon}{2}\right)\mathbf{A}_j^T\mathbf{A}_j.
\end{aligned}$$

Note that $\mathbf{C}^U_{i,i} = \mathbf{B}^U_i$, $\mathbf{C}^L_{i,i} = \mathbf{B}^L_i$, and for $j \le i$, $\mathbf{C}^U_{i,j} \succeq \mathbf{B}^U_j$ and $\mathbf{C}^L_{i,j} \preceq \mathbf{B}^L_j$. We can then define:

$$\begin{aligned}
\mathbf{Y}^U_{i,j} &= \mathbf{C}^U_{i,j} - \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j \\
\mathbf{Y}^L_{i,j} &= \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \mathbf{C}^L_{i,j}.
\end{aligned}$$

We then have, similarly, $\mathbf{Y}^U_{i,i} = \mathbf{X}^U_i$, $\mathbf{Y}^L_{i,i} = \mathbf{X}^L_i$, and for $j \le i$, $\mathbf{Y}^U_{i,j} \succeq \mathbf{X}^U_j$ and $\mathbf{Y}^L_{i,j} \succeq \mathbf{X}^L_j$.

We will assume that $l_i < 1$, since otherwise the claim is immediate (as probabilities cannot exceed 1). Now, note that

$$\begin{aligned}
\mathbf{a}_i^T(\mathbf{Y}^U_{i,0})^{-1}\mathbf{a}_i &= \mathbf{a}_i^T(\mathbf{Y}^L_{i,0})^{-1}\mathbf{a}_i \\
&= \mathbf{a}_i^T\left(\frac{\epsilon}{2}\mathbf{A}_i^T\mathbf{A}_i + \delta I\right)^{-1}\mathbf{a}_i \\
&= \frac{2}{\epsilon}\left(\mathbf{A}_i^T\mathbf{A}_i + \frac{2\delta}{\epsilon}I\right)^{-1}\mathbf{a}_i \\
&= \frac{2}{\epsilon}l_i.
\end{aligned}$$

Next, we will aim to show that for $j < i - 1$,

$$\begin{aligned}
&\mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j+1}\mathbf{a}_i\right] \le \mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j}\mathbf{a}_i\right] \\
&\mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^L_{i-1,j+1}\mathbf{a}_i\right] \le \mathbf{E}\left[\mathbf{a}_i^T\mathbf{Y}^L_{i-1,j}\mathbf{a}_i\right]
\end{aligned}$$

In particular, we will simply show that conditioned on any choices for the first j rows, the expected value of $\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j+1}\mathbf{a}_i$ is no larger than $\mathbf{a}_i^T\mathbf{Y}^U_{i-1,j}\mathbf{a}_i$, and analogously for $\mathbf{Y}^L$.

Similar to the proof of part 1, we separately consider the case where $p_{j+1} = 1$. In that case, the positive semidefinite matrix $\frac{\epsilon}{2}\mathbf{a}_j\mathbf{a}_j^T$ is simply added to $\mathbf{Y}^U$ and $\mathbf{Y}^L$. Adding this can only decrease the values of $\mathbf{a}_i^T\mathbf{Y}^U\mathbf{a}_i$ and $\mathbf{a}_i^T\mathbf{Y}^L\mathbf{a}_i$.

The $p_{j+1} < 1$ case is more tricky. Here, we define the vector $\mathbf{w}_{j+1} = \frac{\mathbf{a}_{j+1}}{\sqrt{p_{j+1}}}$. Importantly

$$\begin{aligned}
&p_{j+1} \ge c_U\mathbf{a}_{j+1}^T(\mathbf{X}^U_j)^{-1}\mathbf{a}_{j+1} \ge c_U\mathbf{a}_{j+1}^T(\mathbf{Y}^U_{i-1,j})^{-1}\mathbf{a}_{j+1} \\
&p_{j+1} \ge c_L\mathbf{a}_{j+1}^T(\mathbf{X}^L_j)^{-1}\mathbf{a}_{j+1} \ge c_L\mathbf{a}_{j+1}^T(\mathbf{Y}^L_{i-1,j})^{-1}\mathbf{a}_{j+1}.
\end{aligned}$$

This means that

$$\begin{aligned}\mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{w}_{j+1}^T &\le \frac{1}{c_U}\\ \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{w}_{j+1}^T &\le \frac{1}{c_L}.\end{aligned}$$

Now, we additionally define

$$\begin{aligned}s_{j+1}^U &= \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{w}_{j+1}^T\\ s_{j+1}^L &= \mathbf{w}_{j+1}^T(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{w}_{j+1}^T\\ \mathbf{u}_{j+1}^U &= \frac{\mathbf{w}_{j+1}}{\sqrt{s_{j+1}^U}}\\ \mathbf{u}_{j+1}^L &= \frac{\mathbf{w}_{j+1}}{\sqrt{s_{j+1}^L}}.\end{aligned}$$

We then deploy Lemma 11 to compute the expectations. For the contribution from the upper barrier, we use $\mathbf{X} = \mathbf{Y}_{i-1,j}^U$, $\mathbf{u} = \mathbf{u}_{j+1}^U$, $\mathbf{v} = \mathbf{a}_i^T$, $a = -s_{j+1}^U(1 - p_{j+1}(1+\epsilon/2))$, $b = s_{j+1}^U p_{j+1}(1+\epsilon/2)$, $p = p_{j+1}$. For the lower barrier, we use $\mathbf{X} = \mathbf{Y}_{i-1,j}^L$, $\mathbf{u} = \mathbf{u}_{j+1}^L$, $\mathbf{v} = \mathbf{a}_i^T$, $a = s_{j+1}^L(1 - p_{j+1}(1-\epsilon/2))$, $b = -s_{j+1}^L p_{j+1}(1-\epsilon/2)$, $p = p_{j+1}$. In both cases we can see that the numerator of the expected change is nonpositive. Finally, this implies that the probability that row i is sampled is

$$\begin{aligned}\mathbf{E}\left[p_i\right] &= c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i\right]\\ &= c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,i-1}^L)^{-1}\mathbf{a}_i\right]\\ &\le c_U\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i\right] + c_L\,\mathbf{E}\left[\mathbf{a}_i^T(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i\right]\\ &= \frac{2}{\epsilon}(c_U + c_L)l_i\\ &= \frac{8}{\epsilon^2}l_i\end{aligned}$$

as desired. ◀

5 Matching Lower Bound

Here we show that the row count obtained by Theorem 10 is in fact optimal. While it is possible to obtain a spectral approximation with $\mathcal{O}(d/\epsilon^2)$ rows in the offline setting, online sampling always incurs a loss of $\Omega\left(\log(\epsilon\|\mathbf{A}\|_2^2/\delta)\right)$ and must sample $\Omega\left(\frac{d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)}{\epsilon^2}\right)$ rows.

▶ **Theorem 12.** *Assume that $\epsilon\|\mathbf{A}\|_2^2 \ge c_1\delta$ and $\epsilon \ge c_2/\sqrt{d}$, for fixed constants c_1 and c_2. Then any algorithm that selects rows in an online manner and outputs a spectral approximation to $\mathbf{A}^T\mathbf{A}$ with $(1+\epsilon)$ multiplicative error and δ additive error with probability at least 1/2 must sample $\Omega\left(\frac{d\log(\epsilon\|\mathbf{A}\|_2^2/\delta)}{\epsilon^2}\right)$ rows of $\mathbf{A}$ in expectation.*

Note that the lower bounds we assume on $\epsilon\|\mathbf{A}\|_2^2$ and ϵ are very minor. They just ensure that $\log(\epsilon\|\mathbf{A}\|_2^2/\delta) \ge 1$ and that ϵ is not so small that we can essentially sample all rows.

Proof. We apply Yao's minimax principle, constructing, for any large enough M, a distribution on inputs $\mathbf{A}$ with $\|\mathbf{A}\|_2^2 \le M$ for which any deterministic online row selection algorithm that succeeds with probability at least $1/2$ must output $\Omega\left(\frac{d\log(\epsilon M/\delta)}{\epsilon^2}\right)$ rows in expectation. The best randomized algorithm that works with probability $1/2$ on any input matrix with $\|\mathbf{A}\|_2^2 \leq M$ therefore must select at least $\Omega\left(\frac{d\log(\epsilon M/\delta)}{\epsilon^2}\right)$ rows in expectation on the worst case input, giving us the theorem.

Our distribution is as follows. We select an integer N uniformly at random from $[1, \log(M\epsilon/\delta)]$. We then stream in the vertex edge incidence matrices of N complete graphs on d vertices. We double the weight of each successive graph. Intuitively, spectrally approximating a complete graph requires selecting $\Omega(d/\epsilon^2)$ edges [2] (as long as $\epsilon \geq c_2/\sqrt{d}$ for some fixed constant c_2). Each time we stream in a new graph with double the weight, we force the algorithm to add $\Omega(d/\epsilon^2)$ more edges to its output, eventually forcing it to output $\Omega(d/\epsilon^2 \cdot N)$ edges – $\Omega(d\log(M\epsilon/\delta)/\epsilon^2)$ in expectation.

Specifically, let $\mathbf{K}_d$ be the $\binom{d}{2} \times d$ vertex edge incidence matrix of the complete graph on d vertices. $\mathbf{K}_d^T\mathbf{K}_d$ is the Laplacian matrix of the complete graph on d vertices. We weight the first graph so that its Laplacian has all its nonzero eigenvalues equal to δ/ϵ. (That is, each edge has weight $\frac{\delta}{d\epsilon}$). In this way, even if we select $N = \lfloor\log(M\epsilon/\delta)\rfloor$ we will have overall $\|\mathbf{A}\|_2^2 \leq \delta/\epsilon + 2\delta/\epsilon + ...2^{\lfloor\log(M\epsilon/\delta)\rfloor-1}\delta/\epsilon \leq M$.

Even if $N = 1$, all nonzero eigenvalues of $\mathbf{A}^T\mathbf{A}$ are at least δ/ϵ, so achieving $(1+\epsilon)$ multiplicative error and $\delta\mathbf{I}$ additive error is equivalent to achieving $(1+2\epsilon)$ multiplicative error. $\mathbf{A}^T\mathbf{A}$ is a graph Laplacian so has a null space. However, as all rows are orthogonal to the null space, achieving additive error $\delta\mathbf{I}$ is equivalent to achieving additive error $\delta\mathbf{I}_r$ where $\mathbf{I}_r$ is the identity projected to the span of $\mathbf{A}^T\mathbf{A}$. $\delta\mathbf{I}_r \preceq \epsilon\mathbf{A}^T\mathbf{A}$ which is why we must achieve $(1+2\epsilon)$ multiplicative error.

In order for a deterministic algorithm to be correct with probability $1/2$ on our distribution, it must be correct for at least $1/2$ of our $\lfloor\log(M\epsilon/\delta)\rfloor$ possible choices of N.

Let i be the lowest choice of N for which the algorithm is correct. By the lower bound of [2], the algorithm must output $\Omega(d/\epsilon^2)$ rows of $\mathbf{A}_i$ to achieve a $(1+2\epsilon)$ multiplicative factor spectral approximation. Here $\mathbf{A}_i$ is the input consisting of the vertex edge incidence matrices of i increasingly weighted complete graphs. Call the output on this input $\tilde{\mathbf{A}}_i$. Now let j be the second lowest choice of N on which the algorithm is correct. Since the algorithm was correct on $\mathbf{A}_i$ to within a multiplicative $(1+2\epsilon)$, to be correct on $\mathbf{A}_j$, it must output a set of edges $\tilde{\mathbf{A}}_j$ such that

$$(\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i) - 4\epsilon\mathbf{A}_j^T\mathbf{A}_j \preceq \tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i \preceq (\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i) + 4\epsilon\mathbf{A}_j^T\mathbf{A}_j.$$

Since we double each successive copy of the complete graph, $\mathbf{A}_j^T\mathbf{A}_j \preceq 2(\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i)$. So, $\tilde{\mathbf{A}}_j^T\tilde{\mathbf{A}}_j - \tilde{\mathbf{A}}_i^T\tilde{\mathbf{A}}_i$ must be a $1+8\epsilon$ spectral approximation to the true difference $\mathbf{A}_j^T\mathbf{A}_j - \mathbf{A}_i^T\mathbf{A}_i$. Noting that this difference is itself just a weighting of the complete graph, by the lower bound in [2] the algorithm must select $\Omega(d/\epsilon^2)$ additional edges between the i^{th} and j^{th} input graphs. Iterating this argument over all $\lfloor\log(M\epsilon/\delta)\rfloor/2$ inputs on which the algorithm must be correct, it must select a total of $\Omega(d\log(M\epsilon/\delta)/\epsilon^2)$ edges in expectation over all inputs. ◀

6 Future Work

An obvious open question arising from our work is if one can prove that the algorithm of [12] works despite dependencies arising due to the row pruning step. By operating in the online setting, our algorithm avoids row pruning, and hence is able to skirt these dependencies, as the probability that a row is sampled only depends on earlier rows in the stream. However, because the streaming setting offers the potential for sampling fewer rows than in the online case, obtaining a rigorous proof of [12] would be very interesting.

While our work focuses on spectral approximation, variants on (ridge) leverage score sampling and the BSS algorithm are also used to solve low-rank approximation problems, including column subset selection [5, 9] and projection-cost-preserving sketching [7, 9]. Compared with spectral approximation, there is less work on streaming sampling for low-rank approximation, and understanding how online algorithms may be used in this setting would an interesting extension of our work.

Acknowledgements. The authors would like to thank Kenneth Clarkson, Jonathan Kelner, Gary Miller, Christopher Musco and Richard Peng for helpful discussions and comments.

Cameron Musco and Jakub Pachocki both acknowledge the Gene Golub SIAM Summer School program on Randomization in Numerical Linear Algebra, where work on this project was initiated.

References

1 Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 775–783, 2015.

2 Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

3 Antoine Bordes and Léon Bottou. The huller: a simple and efficient online SVM. In *Machine Learning: ECML 2005*, pages 505–512. Springer, 2005.

4 Christos Boutsidis, Dan Garber, Zohar Karnin, and Edo Liberty. Online principal components analysis. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 887–901, 2015.

5 Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362, 2014.

6 Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

7 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–172, 2015.

8 Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

9 Michael B Cohen, Cameron Musco, and Christopher Musco. Ridge leverage scores for low-rank approximation. *arXiv:1511.07263*, 2015.

10 Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

11 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 561–570, 2014.

12 Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013.

13 Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 235–244, 2010.

14 Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 250–269, 2015.

15 Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 127–136, 2013.

16 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 81–89, 2016.

17 Michael W. Mahoney and Xiangrui Meng. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 2013.

18 Jelani Nelson and Huy L. Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 117–126, 2013.

19 Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

20 Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2004.

21 Daniel A Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

22 Joel Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.