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

Vladimir Braverman† Petros Drineas‡ Cameron Musco§ Christopher Musco¶
Jalaj Upadhyay‖ David P. Woodruff▽ Samson Zhou⋈

April 21, 2020

Abstract

We initiate the study of numerical linear algebra in the sliding window model, where only the most recent W updates in a stream form the underlying data set. Although many existing algorithms in the sliding window model use or borrow elements from the smooth histogram framework (Braverman and Ostrovsky, FOCS 2007), we show that many interesting linear-algebraic problems, including spectral and vector induced matrix norms, generalized regression, and low-rank approximation, are not amenable to this approach in the row-arrival model. To overcome this challenge, we first introduce a unified row-sampling based framework that gives *randomized* algorithms for spectral approximation, low-rank approximation/projection-cost preservation, and ℓ_1-subspace embeddings in the sliding window model, which often use nearly optimal space and achieve nearly input sparsity runtime. Our algorithms are based on "reverse online" versions of offline sampling distributions such as (ridge) leverage scores, ℓ_1 sensitivities, and Lewis weights to quantify both the importance and the recency of a row; our structural results on these distributions may be of independent interest for future algorithmic design.

Although our techniques initially address numerical linear algebra in the sliding window model, our row-sampling framework rather surprisingly implies connections to the well-studied online model; our structural results also give the first sample optimal (up to lower order terms) online algorithm for low-rank approximation/projection-cost preservation. Using this powerful primitive, we give online algorithms for column/row subset selection and principal component analysis that resolves the main open question of Bhaskara *et al.* (FOCS 2019). We also give the first online algorithm for ℓ_1-subspace embeddings. We further formalize the connection between the online model and the sliding window model by introducing an *additional* unified framework for *deterministic* algorithms using a merge and reduce paradigm and the concept of online coresets, which we define as a weighted subset of rows of the input matrix that can be used to compute a good approximation to some given function on all of its prefixes. Our sampling based algorithms in the row-arrival online model yield online coresets, giving deterministic algorithms for spectral approximation, low-rank approximation/projection-cost preservation, and ℓ_1-subspace embeddings in the sliding window model that use nearly optimal space.

*Considerably strengthens and subsumes earlier versions appearing on arXiv.

†Johns Hopkins University. E-mail: vova@cs.jhu.edu.
‡Purdue University. E-mail: pdrineas@purdue.edu
§University of Massachusetts Amherst. E-mail: cmusco@cs.umass.edu
¶New York University. E-mail: cmusco@nyu.edu
‖Apple, Inc. E-mail: jalaj.kumar.upadhyay@gmail.com
▽Carnegie Mellon University. E-mail: dwoodruf@cs.cmu.edu
⋈Carnegie Mellon University. E-mail: samsonzhou@gmail.com

1 Introduction

The advent of big data has reinforced efforts to design and analyze algorithms in the *streaming model*, where data arrives sequentially, can be observed in a small number of passes (ideally once), and the proposed algorithms are allowed to use space that is sublinear in the size of the input. For example in a typical e-commerce setup, the entries of a row represent the number of each item purchased by a customer in a transaction. As the transaction is completed, the advertiser receives an entire row of information as an update, which corresponds to the *row-arrival model*. Then the underlying covariance matrix summarizes information about which items tend to be purchased together, while low-rank approximation identifies a representative subset of transactions.

However, the streaming model does not fully address settings where the data is time-sensitive; the advertiser is not interested in the outdated behavior of customers. Thus one scenario that is not well-represented by the streaming model is when recent data is considered more accurate and important than data that arrived prior to a certain time window, as in applications such as network monitoring [CM05, CG08, Cor13], event detection in social media [OMM+14], and data summarization [CNZ16, ELVZ17]. To model such settings, Datar *et al.* [DGIM02] introduced the *sliding window model*, which is parametrized by the size W of the window that represents the size of the *active data* that we want to analyze, in contrast to the so-called "expired data". The objective is to compute or approximate statistics only on the active data using memory that is sublinear in the window size W.

The sliding window model is more appropriate than the unbounded streaming model in a number of applications [BBD+02, MM12, PGD15, WLL+16]. For example in large scale social media analysis, each row in a matrix can correspond to some online document, such as the content of a Twitter post, and given some corresponding time information. Although a streaming algorithm can analyze the data starting from a certain time, analysis with a recent time frame, e.g., the most recent week or month, could provide much more attractive information to advertisers or content providers. Similarly in the task of data summarization, the underlying data set is a matrix whose rows correspond to a number of subjects, while the columns correspond to a number of features. Information on each subject arrives sequentially and the task is to select a small number of representative subjects, which is usually done through some kind of PCA [PY06, QAWZ15]. However, if the behavior of the subjects has recently and indefinitely changed, we would like the summary to only be representative of the updated behavior, rather than the outdated information.

Another time-sensitive scenario that is not well-represented by the streaming model is when irreversible decisions must be made upon the arrival of each update in the stream, which enables further actions downstream, such as in scheduling, facility location, and data structures. The goal of the *online model* is to address such settings by requiring immediate and permanent actions on each element of the stream as it arrives, while still remaining competitive with an optimal offline solution that has full knowledge of the entire input. We specifically study the case where the online model must also use space sublinear in the size of the input, though this restriction is not always enforced across algorithms in the online model for other problems. In the context of online PCA, an algorithm receives a stream of input vectors and must immediately project each input vector into a lower dimension space of its choice. The projected vector can then be used as input to some downstream rotationally invariant algorithm, such as classification, clustering, or regression, which would run more efficiently due to the lower dimensional input. Moreover, PCA serves as a popular preprocessing step because it often actually *improves* the quality of the solution by removing isotropic noise [BGKL15] from the data, so that in applications such as clustering, the

denoised projection can perform better than the original input. The online model has also been extensively used in a number of other applications, such as learning [BKRW03, HK16, BDV18], (prophet) secretary problems [Rub16, EHLM17, EHKS18], ad allocation [NW18], and a variety of graph applications [CW18, GKM+19, CPW19].

Generally, the sliding window model and the online model do not seem related, resulting in a different set of techniques being developed for each problem and each setting. Surprisingly, our results exhibit a seemingly unexplored and interesting connection between the sliding window model and the online model. Our observation is that an online algorithm should be correct on all prefixes of the input, in case the stream terminates at that prefix; on the other hand, a sliding window algorithm should be correct on all suffixes of the input, in case the previous elements expire leaving only the suffix (and perhaps a bunch of "dummy" elements). Then can we gain something by viewing each update to a sliding window algorithm as an update to an online algorithm *in reverse*? At first glance, the answer might seem to be no; we cannot simulate an online algorithm with the stream in reverse order because it would have access to the entire stream whereas a sliding window algorithm only maintains a sketch of the previous elements upon each update. However, it turns out that in the row-arrival model, a sketch of the previous elements often suffices to approximately simulate the entire stream input to the online algorithm. Indeed, we show that any row-sampling based online algorithm for the problems of spectral approximation, low-rank approximation/projection-cost preservation, and ℓ_1-subspace embedding automatically implies a corresponding deterministic sliding window algorithm for the problem!

Formally, we study the following numerical linear algebraic problems in the row-arrival online and sliding window models:

Spectral Approximation. Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and an approximation parameter $\varepsilon > 0$, we wish to find a matrix $\mathbf{M} \in \mathbb{R}^{m\times d}$ with $m \ll n$ that is a spectral approximation (or interchangeably, an ℓ_2-subspace embedding) of $\mathbf{A}$. That is, with high probability, our output matrix $\mathbf{M}$ should satisfy $(1-\varepsilon)\|\mathbf{Ax}\|_2 \le \|\mathbf{Mx}\|_2 \le (1+\varepsilon)\|\mathbf{Ax}\|_2$ for all $\mathbf{x} \in \mathbb{R}^d$. Equivalently, we require $(1-\varepsilon)\mathbf{A}^\top\mathbf{A} \preceq \mathbf{M}^\top\mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top\mathbf{A}$.

Low-Rank Approximation/Projection-Cost Preservation. In the low-rank approximation problem, we are given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, a rank parameter $k > 0$, and an approximation parameter $\varepsilon > 0$, and we wish to find a matrix $\mathbf{M} \in \mathbb{R}^{m\times d}$ with $m \ll n$ such that $(1-\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2 \le \left\|\mathbf{M}-\mathbf{M}_{(k)}\right\|_F^2 \le (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$, where $\mathbf{A}_{(k)}$ for a matrix $\mathbf{A}$ represents the best rank k approximation to $\mathbf{A}$. A stronger notion is a projection-cost preservation[1]:

Definition 1.1 (Rank k Projection-Cost Preservation [CMM17])**.** *For $m < n$, a matrix $\mathbf{M} \in \mathbb{R}^{m\times d}$ of rescaled rows of $\mathbf{A} \in \mathbb{R}^{n\times d}$ is a $(1+\varepsilon)$ projection-cost preservation if, for all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

$$(1-\varepsilon)\|\mathbf{A}-\mathbf{AP}\|_F^2 \le \|\mathbf{M}-\mathbf{MP}\|_F^2 \le (1+\varepsilon)\|\mathbf{A}-\mathbf{AP}\|_F^2.$$

Note if $\mathbf{M}$ is a projection-cost preservation of $\mathbf{A}$, then its best low-rank approximation can be used to find a projection matrix that gives an approximation of the best low-rank approximation to $\mathbf{A}$.

[1]Rank k projection-cost preservation was originally formulated as an additive-multiplicative guarantee by [CEM+15] but subsequent literature uses the purely additive form as in Definition 1.1.

ℓ_1-Subspace Embedding. The ℓ_1-subspace embedding problem has similar demands to the spectral approximation problem, but the structural differences between the ℓ_1 and ℓ_2 norms require vastly different techniques. Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and an approximation parameter $\varepsilon > 0$, we wish to find a matrix $\mathbf{M} \in \mathbb{R}^{m\times d}$ with $m \ll n$ so that, with high probability, we have $(1-\varepsilon)\|\mathbf{Ax}\|_1 \leq \|\mathbf{Mx}\|_1 \leq (1+\varepsilon)\|\mathbf{Ax}\|_1$ for all $\mathbf{x} \in \mathbb{R}^d$.

Row/Column Subset Selection. The row/column subset selection problems are symmetric, depending on whether the arrival model is rows or columns; we address the row subset selection problem in the row-arrival model. Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, a rank parameter $k > 0$, and an approximation parameter $\varepsilon > 0$, the goal is to select k rows of $\mathbf{A}$ to form a matrix $\mathbf{M}$ to minimize $\left\|\mathbf{A} - \mathbf{AM}^\dagger\mathbf{M}\right\|_F$. Since the matrix $\mathbf{M}$ has rank at most k, then $\left\|\mathbf{A} - \mathbf{AM}^\dagger\mathbf{M}\right\|_F \geq \left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F$, but we would ideally like to obtain some guarantee for $\left\|\mathbf{A} - \mathbf{AM}^\dagger\mathbf{M}\right\|_F$ relative to $\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F$, where $\mathbf{A}_{(k)}$ is the best rank k approximation to $\mathbf{A}$.

Principal Component Analysis. In the (online) PCA problem, rows of the matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ arrive sequentially in a data stream and after each row $\mathbf{a}_i$ arrives, the goal is to immediately output a row $\mathbf{m}_i \in \mathbb{R}^m$ with $m \ll d$ such that at the end of the stream, there exists a low-rank matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ with

$$\|\mathbf{A} - \mathbf{MX}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2,$$

where m should be minimal and $\mathbf{A}_{(k)}$ is the best rank k approximation to $\mathbf{A}$. Here the matrix $\mathbf{M} = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_n$ is formed by the concatenation of the rows that we output at each time.

1.1 Our Contributions

We initiate and perform a comprehensive study for both randomized and deterministic algorithms in the sliding window model. We first present a randomized row sampling framework for spectral approximation, low-rank approximation/projection-cost preservation, and ℓ_1-subspace embeddings in the sliding window model. Most of our results are space or time optimal, up to lower order terms. Our sliding window structural results imply structural results for the online setting, which we use to give algorithms for row/column subset selection, PCA, projection-cost preservation, and subspace embeddings in the online model. Our online algorithms are simple and intuitive, yet they either are novel for the particular problem or improve upon the state-of-the-art, e.g., Bhaskara *et al.* (FOCS 2019) [BLVZ19]. Finally, we formalize a surprising connection between online algorithms and sliding window algorithms by describing a unified framework for deterministic algorithms in the sliding window model based on the merge-and-reduce paradigm and the concept of online coresets, which are provably generated by online algorithms. We now describe our results in greater detail.

Row Sampling Framework for the Sliding Window Model. One may ask whether existing algorithms in the sliding window model can be generalized to problems for numerical linear algebra. For example, it is known that elementary linear-algebraic problems, such as estimating the Frobenius norm, can be addressed in the sliding window model using the smooth histogram framework [BO07] (we refer to Appendix A.1 for general background on the smooth histogram framework). We show that it is not the case in general. In Appendix A.2, we give counterexamples showing that various linear-algebraic functions, including the spectral norm, vector induced matrix

norms, generalized regression, and low-rank approximation, are not smooth according to the definitions of [BO07] and therefore cannot be used in the smooth histogram framework. This motivates the need for new frameworks for problems of linear algebra in the sliding window model. We first give a row sampling based framework for space and runtime efficient randomized algorithms for numerical linear algebra in the sliding window model.

Framework 1.2 (Row Sampling Framework for the Sliding Window Model)**.** *There exists a row sampling based framework in the sliding window model that upon the arrival of each new row of the stream with condition number κ chooses whether to keep or discard each previously stored row, according to some predefined probability distribution for each problem. Using the appropriate probability distribution, we obtain for any approximation parameter $\varepsilon > 0$:*

(1) A randomized algorithm for spectral approximation in the sliding window model that with high probability, outputs a matrix $\mathbf{M}$ that is a subset of (rescaled) rows of an input matrix $\mathbf{A} \in \mathbb{R}^{W \times d}$ such that $(1-\varepsilon)\mathbf{A}^\top\mathbf{A} \preceq \mathbf{M}^\top\mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top\mathbf{A}$, while storing $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n \log \kappa\right)$ rows at any time and using nearly input sparsity time. (See Theorem 2.6.)

(2) A randomized algorithm for low-rank approximation/projection-cost preservation in the sliding window model that with high probability, outputs a matrix $\mathbf{M}$ that is a subset of (rescaled) rows of an input matrix $\mathbf{A} \in \mathbb{R}^{W \times d}$ such that for all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d \times d}$,

$$(1-\varepsilon)\left\|\mathbf{A}-\mathbf{AP}\right\|_F^2 \leq \left\|\mathbf{M}-\mathbf{MP}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{AP}\right\|_F^2,$$

while storing $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ rows at any time and using nearly input sparsity time. (See Theorem 2.12.)

(3) A randomized algorithm for ℓ_1-subspace embeddings in the sliding window model that with high probability, outputs a matrix $\mathbf{M}$ that is a subset of (rescaled) rows of an input matrix $\mathbf{A} \in \mathbb{R}^{W \times d}$ such that $(1-\varepsilon)\left\|\mathbf{Ax}\right\|_1 \leq \left\|\mathbf{Mx}\right\|_1 \leq (1+\varepsilon)\left\|\mathbf{Ax}\right\|_1$ for all $\mathbf{x} \in \mathbb{R}^d$, while storing $\mathcal{O}\left(\frac{d^2}{\varepsilon^2}\log^2 n \log \kappa\right)$ rows at any time. (See Theorem 4.9.)

Here we say the stream has condition number κ if any matrix formed by consecutive rows of the stream has condition number at most κ.

We further show that for low-rank approximation/projection-cost preservation, we can further improve the polylogarithmic factors from $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ rows stored at any time to $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log^2 n\right)$, under the assumption that the entries of the underlying matrix are integers with magnitude at most $\text{poly}(n)$ even though $\log \kappa$ can be as large as $\mathcal{O}(d \log n)$ with these assumptions. To the best of our knowledge, not only are our contributions in Framework 1.2 the first such algorithms for these problems in the sliding window model, but also Theorem 2.6 and Theorem 2.12 are both space and runtime optimal up to lower order terms, even compared to row sampling algorithms in the offline setting for most reasonable regime of parameters [ACK+16, DV06].

Numerical Linear Algebra in the Online Model. An important step in the analysis of our row sampling framework for numerical linear algebra in the sliding window model is bounding the sum of the sampling probabilities for each row. In particular, we provide a tight bound on the sum of the online ridge leverage scores that was previously unexplored. We show that our bounds along with the paradigm of row sampling with respect to online ridge leverage scores offer simple online algorithms that improve upon the state-of-the-art across broad applications.

Theorem 1.3 (Online Rank k Projection-Cost Preservation)**.** *Given parameters $\varepsilon > 0$, $k > 0$, and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that with high probability, outputs a matrix $\mathbf{M}$ that has $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ (rescaled) rows of $\mathbf{A}$ and for all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

$$(1-\varepsilon)\left\|\mathbf{A}-\mathbf{AP}\right\|_F^2 \leq \left\|\mathbf{M}-\mathbf{MP}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{AP}\right\|_F^2.$$

(See Theorem 3.1.)

Theorem 3.1 immediately yields improvements on the two online algorithms recently developed by Bhaskara *et al.* (FOCS 2019) for online row subset selection and online PCA [BLVZ19].

Theorem 1.4 (Online Row Subset Selection)**.** *Given parameters $\varepsilon > 0$, $k > 0$, and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that with high probability, outputs a matrix $\mathbf{M}$ with $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log^2 \kappa\right)$ rows that contains a matrix $\mathbf{T}$ of k rows such that*

$$\left\|\mathbf{A}-\mathbf{AT}^{-1}\mathbf{T}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2.$$

(See Theorem 3.5.)

By comparison, the online row subset selection algorithm of [BLVZ19] stores $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ rows to succeed with high probability. Moreover, our algorithm provides the guarantee of the existence of a subset $\mathbf{T}$ of k rows that provides a $(1+\varepsilon)$-approximation to the best rank k solution, whereas [BLVZ19] promises the bicriteria result that their matrix with rank $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ is a $(1+\varepsilon)$-approximation to the best rank k solution.

The online PCA algorithm of [BLVZ19] also offers this bicriteria guarantee; for an input matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, they give an algorithm that outputs a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$, where $m = \mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log\kappa)^4\right)$ and a matrix $\mathbf{X}$ of rank m, so that $\|\mathbf{A}-\mathbf{MX}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. Our online row subset selection can also be adjoined with the online PCA algorithm of [BLVZ19] to offer the promise of the existence of a submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ within $\mathbf{X}$ such that $\|\mathbf{A}-\mathbf{BY}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$.

Theorem 1.5 (Online Principal Component Analysis)**.** *Given parameters $n, d, k, \varepsilon > 0$ and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ whose rows arrive sequentially in a stream with condition number κ, let $m = \mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log\kappa)^4\right)$. There exists an algorithm for online PCA that immediately outputs a row $\mathbf{m}_i \in \mathbb{R}^m$ after seeing row $\mathbf{a}_i \in \mathbb{R}^d$ and with high probability, outputs a matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ at the end of the stream such that*

$$\|\mathbf{A}-\mathbf{MX}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2,$$

where $\mathbf{A}_{(k)}$ is the best rank k approximation to $\mathbf{A}$. Moreover, $\mathbf{X}$ contains a submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ such that there exists a matrix $\mathbf{B}$ such that

$$\|\mathbf{A}-\mathbf{BY}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2.$$

(See Theorem 3.6.)

Our sliding window algorithm for ℓ_1-subspace embeddings also uses an online ℓ_1-subspace embedding algorithm that we develop.

Theorem 1.6 (Online ℓ_1-Subspace Emebedding). *Given $\varepsilon > \frac{1}{n}$ and a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that outputs a matrix $\mathbf{M}$ with $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2 n \log \kappa\right)$ (rescaled) rows of $\mathbf{A}$ such that*

$$(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_1 \leq \|\mathbf{M}\mathbf{x}\|_1 \leq (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_1,$$

for all $\mathbf{x} \in \mathbb{R}^d$ with high probability. (See Theorem 4.8.)

We summarize these results in Figure 1.

Online Algorithms	Rows Sampled/Output Dimension	Notes
Rank k Approximation	$\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ (Theorem 3.1)	
Row Subset Selection	$\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ [BLVZ19]	Bicriteria
	$\mathcal{O}\left(\frac{k}{\varepsilon} \log n \log^2 \kappa\right)$ (Theorem 3.5)	
Principal Component Analysis	$\mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log \kappa)^4\right)$ [BLVZ19]	Bicriteria
	$\mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log \kappa)^4\right)$ (Theorem 3.6)	
ℓ_1-Subspace Embedding	$\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2 n \log \kappa\right)$ (Theorem 4.9)	

Fig. 1: Online algorithms for an input matrix of dimension $n \times d$. κ is the condition number of the stream. Bicriteria denotes that the rank of the matrix within $(1+\varepsilon)$ approximation of the best rank k solution need not have rank at most k.

A Coreset Framework for Deterministic Sliding Window Algorithms. To formalize a connection between online algorithms and sliding window algorithms, we give a framework for deterministic sliding window algorithms based on the merge-and-reduce paradigm and the concept of an online coreset, which we define as a weighted subset of rows of $\mathbf{A}$ that can be used to compute a good approximation to some given function on all prefixes of $\mathbf{A}$. On the other hand, observe that a row-sampling based online algorithm does not know when the input might terminate, so it must output a good approximation to any prefix of the input, which is exactly the requirement of an online coreset! Moreover, an online cannot revoke any of its decisions, so the history of its decisions are fully observable. Indeed, each of our online algorithms imply the existence of an online coreset for the corresponding problem. Intuition for our framework is presented in Figure 2.

Framework 1.7 (Coreset Framework for Deterministic Sliding Window Algorithms). *There exists a merge-and-reduce framework for numerical linear algebra in the sliding window model using online coresets. If the input stream has condition number κ, then for approximation parameter $\varepsilon > \frac{1}{n}$, the framework gives:*

(1) A deterministic algorithm for spectral approximation in the sliding window model that outputs a matrix $\mathbf{M}$ that is a subset of (rescaled) rows of an input matrix $\mathbf{A} \in \mathbb{R}^{W \times d}$ such that $(1-\varepsilon)\mathbf{A}^\top\mathbf{A} \preceq \mathbf{M}^\top\mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top\mathbf{A}$, while storing $\mathcal{O}\left(\frac{d}{\varepsilon^2} \log^4 n \log \kappa\right)$ rows at any time. (See Theorem 5.5).

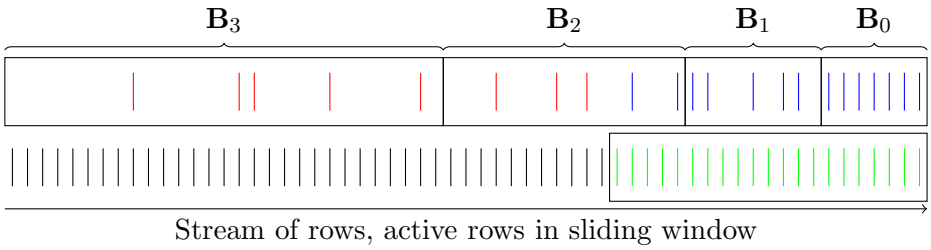

Fig. 2: Merge and reduce framework for deterministic sliding window algorithms via coresets that accurately approximate *any* suffix of the input. The stream of rows proceeds from left to right, with the active rows of the sliding window in green. The rows sampled by the coresets $\mathbf{B}_0, \mathbf{B}_1, \ldots$ are in color above, with the blue rows used to approximate the active rows and red rows approximating expired portions of the stream, even if the size of the sliding window is given after the stream.

(2) A deterministic algorithm for low-rank approximation/projection-cost preservation in the sliding window model that outputs a matrix $\mathbf{M}$ *that is a subset of (rescaled) rows of an input matrix* $\mathbf{A} \in \mathbb{R}^{W \times d}$ *such that for all rank* k *orthogonal projection matrices* $\mathbf{P} \in \mathbb{R}^{d \times d}$*,*

$$(1-\varepsilon)\left\|\mathbf{A}-\mathbf{A}\mathbf{P}\right\|_F^2 \leq \left\|\mathbf{M}-\mathbf{M}\mathbf{P}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}\mathbf{P}\right\|_F^2,$$

while storing $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log^4 n \log^2 \kappa\right)$ *rows at any time. (See Theorem 5.7.)*

(3) A deterministic algorithm for ℓ_1*-subspace embeddings in the sliding window model that outputs a matrix* $\mathbf{M}$ *that is a subset of (rescaled) rows of an input matrix* $\mathbf{A} \in \mathbb{R}^{W \times d}$ *such that* $(1-\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1 \leq \left\|\mathbf{M}\mathbf{x}\right\|_1 \leq (1+\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1$ *for all* $\mathbf{x} \in \mathbb{R}^d$*, while storing* $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log^4 n \log \kappa\right)$ *rows at any time. (See Theorem 5.16).*

All of the results presented using Framework 1.7 are space optimal, up to lower order terms [ACK+16, DV06, CP15]. Again we have the property for low-rank approximation/projection-cost preservation that the number of sampled rows can be improved from $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2 \kappa\right)$ to $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log^2 n\right)$, under the assumption that the entries of the underlying matrix are integers at most $\text{poly}(n)$ in magnitude.

We remark that neither our randomized framework Framework 1.2 nor our deterministic framework Framework 1.7 requires the sliding window parameter W as input during the processing of the stream. Instead, they create *oblivious* data structures from which approximations for any window can be computed after processing the stream. Intuitively, this can be visualized by Figure 2. We summarize our sliding window algorithms in Figure 3.

1.2 Overview of Our Techniques

The design and analysis of many existing algorithms in the sliding window model use either the exponential histogram framework [DGIM02] or the smooth histogram framework [BO07]. Unfortunately, we show in Appendix A.2 that these frameworks cannot be applied to many interesting linear-algebraic functions, such as approximating the spectral norm or vector induced matrix norms, generalized regression, and low-rank approximation. This motivates the need for new frameworks for problems of linear algebra in the sliding window model.

Our first observation is that as additional rows arrive in the stream, the singular values of the underlying matrix cannot decrease. Then we must have the Loewner ordering $\mathbf{M}_1^\top \mathbf{M}_1 \succeq \ldots \succeq$

Sliding Window Algorithms	Rows Sampled (Randomized)	Rows Sampled (Deterministic)
Spectral Approximation	$\widetilde{\Theta}\left(\frac{d}{\varepsilon^2}\right)$ (Theorem 2.6)	$\widetilde{\Theta}\left(\frac{d}{\varepsilon^2}\right)$ (Theorem 5.5)
Rank k Approximation	$\widetilde{\Theta}\left(\frac{k}{\varepsilon^2}\right)$ (Theorem 2.12)	$\widetilde{\Theta}\left(\frac{k}{\varepsilon^2}\right)$ (Theorem 5.7)
ℓ_1-Subspace Embedding	$\widetilde{\mathcal{O}}\left(\frac{d^2}{\varepsilon^2}\right)$ (Theorem 4.9)	$\widetilde{\Theta}\left(\frac{d}{\varepsilon^2}\right)$ (Theorem 5.16)

Fig. 3: Sliding window algorithms using Framework 1.2 for randomized algorithms and Framework 1.7 for deterministic algorithms for a stream of length n of dimension d rows. We omit dependencies on $\log n$ and $\log \kappa$, where κ is the condition number of the stream. Furthermore, Theorem 2.6 and Theorem 2.12 run in nearly input sparsity time.

$\mathbf{M}_n^\top \mathbf{M}_n$, where $\mathbf{M}_i$ is the matrix formed by the rows that have arrived since time i. For spectral approximation, we can reduce the number of outer products stored, since we only care when some singular value of a matrix has increased by $(1+\varepsilon)$. Thus we can repeatedly delete an index i if $\mathbf{M}_{i-1}^\top \mathbf{M}_{i-1} \preceq (1+\varepsilon)\mathbf{M}_{i+1}^\top \mathbf{M}_{i+1}$ and relabel the indices, since there are no significant differences between the singular values of $\mathbf{M}_{i-1}$ and $\mathbf{M}_{i+1}$. We can repeatedly delete matrices until there are about $\mathcal{O}\left(\frac{d}{\varepsilon}\log\kappa\right)$ matrices remaining, where d is the dimension of each row and κ is the condition number of the stream. Now if A and B are substreams, where B is a suffix of A, corresponding to matrices $\mathbf{A}$ and $\mathbf{B}$ and $(1-\varepsilon)\mathbf{A}^\top\mathbf{A} \preceq \mathbf{B}^\top\mathbf{B} \preceq \mathbf{A}^\top\mathbf{A}$, then

$$(1-\varepsilon)(\mathbf{A}^\top\mathbf{A}+\mathbf{C}^\top\mathbf{C}) \preceq \mathbf{B}^\top\mathbf{B}+\mathbf{C}^\top\mathbf{C} \preceq \mathbf{A}^\top\mathbf{A}+\mathbf{C}^\top\mathbf{C}$$

for any matrix $\mathbf{C}$ that represents a substream C that arrives right after the substream A. Hence, if we have a $(1+\varepsilon)$ spectral approximation to some suffix of the stream, it will remain a $(1+\varepsilon)$ spectral approximation upon the arrival of new rows. In particular, the matrix represented by the active rows in the sliding window will be sandwiched between two matrices maintained by the algorithm and thus be well-approximated. In fact, this approach can easily be seen as a generalization of the smooth histogram framework to matrix functions and is formalized in Appendix A.3.

Alas, not only is this approach not space optimal, but it does not seem to generalize to other linear algebraic problems in the sliding window model. Nevertheless, this warm-up algorithm crucially gives insight into a more space efficient spectral approximation that *does* generalize to other algorithms. Observe that the rows of $\mathbf{M}_i$ are a subset of the rows of $\mathbf{M}_j$ for any $i < j$, so $\mathbf{M}_i^\top \mathbf{M}_i$ and $\mathbf{M}_j^\top \mathbf{M}_j$ be storing a lot of redundant information, which suggests a row sampling approach for more space efficient algorithms.

Spectral Approximation via Row Sampling in the Sliding Window Model. The challenge for row sampling approaches in the sliding window model results from two conflicting forces. Suppose we have a good approximation $\mathbf{M}$ to the matrix $\mathbf{A}$ consisting of the rows that have already arrived in the stream. When a new row $\mathbf{r}$ arrives, we would like to sample $\mathbf{r}$ with high probability if $\mathbf{r}$ is "important" based on the rows in $\mathbf{M}$. Namely, if $\mathbf{r}$ has high norm or a different direction than the rows of $\mathbf{M}$, then we would like to capture that information by sampling $\mathbf{r}$. On the other hand, if $\mathbf{r}$ has low importance based on the existing rows of $\mathbf{M}$, then it seems like we should not sample $\mathbf{r}$.

However, the sliding window model also emphasizes the more recent rows. For example, it may be possible that the rows that follow $\mathbf{r}$ all contain only zeroes and that all rows of before $\mathbf{r}$ are

expired at the time of query. Since all rows of $\mathbf{M}$ have expired at the time of query, then we would be left with no information about the underlying matrix if we did not sample $\mathbf{r}$. This implies that we must *always* store the most recent row and similarly place greater emphasis on more recent rows. Although the (ridge) leverage score of a row quantifies the "uniqueness" or "importance" of a row with respect to all other rows in the matrix, there is no measure that combines both uniqueness and recency.

We first consider "online" versions of sampling distributions, which quantifies the importance of a row with respect to the *previous* rows in the matrix. For example, [CMP16] introduces online (ridge) leverage scores for a row sampling approach to online spectral approximation. By contrast, the sliding window model seems to value the importance of a row with respect to the *following* rows in the matrix. Thus we introduce the concept of *reverse online leverage scores* for spectral approximation in the sliding window model. Given rows $\mathbf{r}_1, \ldots, \mathbf{r}_t$, we say the reverse online leverage score of row $\mathbf{r}_i$ is the leverage score of $\mathbf{r}_i$ with respect to the matrix formed by the concatenation of the following rows $\mathbf{r}_{i+1} \circ \ldots \circ \mathbf{r}_t$[2]. Note specifically that the reverse online leverage score of the most recent row is always 1, which matches the previous observation that we should always sample the most recent row in the sliding window model.

On the other hand, we cannot compute the reverse online leverage scores of each row without storing the entire stream. We can compute the reverse online leverage score of a row with respect to the rows that we have sampled as a $(1+\varepsilon)$ approximation to the true score, but a possible concern is that the error at each step compounds and the error at the end could be as large as $(1+\varepsilon)^n$. To resolve this issue, we use an idea of [CLM+15] that shows if we have a $(1+\varepsilon)$ approximation for each score but then oversample each row by a factor $C > (1 + \mathcal{O}(\varepsilon))$, then the error *will not* compound and the resulting matrix will still be a $(1+\varepsilon)$ approximation. We also use a matrix martingale argument similar to [CMP16] to avoid known row sampling dependencies [KL13].

To bound the rows sampled by our algorithm, we note that each row is sampled with probability proportional to the reverse online leverage score, and [CMP16] bounds the sum of the online leverage scores by $\mathcal{O}(d \log \kappa)$ assuming bounded entries in the underlying matrix, which must also bound the sum of the reverse online leverage scores. We also note that by batching until a certain number of rows have arrived before choosing to discard rows, we can amortize the runtime needed to approximate each of the reverse online leverage scores and obtain nearly input sparsity time by using standard projection techniques to embed each of the sampled rows into an $\mathcal{O}\left(\log \frac{n}{\varepsilon}\right)$ dimensional subspace.

From Spectral Approximation to a Row Sampling Framework. The spectral approximation algorithm suggests a simple row sampling framework for sliding window algorithms. Suppose at each time, we have stored a matrix $\mathbf{M}$ that serves as a good approximation for some function, e.g., a low-rank approximation or an ℓ_1-subspace embedding, on input matrix $\mathbf{A}$. When a new row $\mathbf{r}$ arrives, we add $\mathbf{r}$ to $\mathbf{M}$ and then we start from the most recent row in $\mathbf{M}$ and iteratively choose whether to keep and rescale each row of $\mathbf{M}$ based on some "reverse online" sampling distribution that is monotonic – if the sampling probability of a row increases as additional rows arrive, we can no longer guarantee that we sampled the row with sufficient probability. Then generally our analysis must first show correctness of the sampling probabilities and then bound the number of sampled rows.

Low-Rank Approximation/Projection-Cost Preservation. For low-rank approximation, the same

[2]We define the reverse online leverage score to be 1 when a row is not in the span of the following rows.

matrix martingale argument shows that a reverse online version of ridge leverage scores [AM15, CEM+15, CMM17] with the proper regularization provides a rank k projection-cost preservation of the underlying matrix, and thus a low-rank approximation. We then provide a tighter bound on the sum of the (reverse) online ridge leverage scores with regularization parameter $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$, which shows that the number of sampled rows will be proportional to k rather than d. However, this *still* does not suffice for our purpose; we do not know $\frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$ in advance and thus we cannot regularize. Fortunately, the fact that our algorithm is a rank k projection-cost preservation means that we have an $(1+\varepsilon)$ approximation to the regularization parameter at all times. Thus we can again oversample by a factor of say 2 to compensate and avoid compounding errors. Similar to our spectral approximation algorithm, our low-rank approximation algorithm has input sparsity runtime, up to lower order factors.

ℓ_1-Subspace Embedding. For ℓ_1-sampling, we define the reverse online ℓ_1-sensitivity of a row $\mathbf{a}_i \in \mathbb{R}^d$ as $\max_{\mathbf{x}\in\mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{Z}_i\mathbf{x}\|_1}$, where $\mathbf{Z}_i$ is the matrix that consists of the rows following $\mathbf{a}_i$[3]. The analysis showing correctness of this probability distribution is straightforward; it follows from a scalar martingale concentration inequality to show that $\|\mathbf{M}\mathbf{x}\|_1 \approx \|\mathbf{A}\mathbf{x}\|_1$ for all points $\mathbf{x}$ in an ε-net, where $\mathbf{A}$ is the input matrix and $\mathbf{M}$ is the sampled matrix. By a simple chaining argument, it then follows that $\|\mathbf{M}\mathbf{x}\|_1 \approx \|\mathbf{A}\mathbf{x}\|_1$ for all $\mathbf{x} \in \mathbb{R}^d$. It follows that given the correctness of $\mathbf{M}$ at all times, it is straightforward for our algorithm to approximate the reverse online ℓ_1 sensitivity of each row. We could use a reverse online version of the ℓ_1 leverage scores [DDH+08], but these quantities would result in a higher number of sampled rows. We could also try a reverse online version of the Lewis weights [CP15], but it does seem apparent how to approximate these quantities.

The challenge is bounding the sum of the (reverse) online ℓ_1 sensitivities. A natural approach would be adapting the approach [CMP16], who relate the sum of the online ℓ_2 leverage scores to the evolution of the determinant of $\mathbf{A}^\top\mathbf{A}$ as additional rows of $\mathbf{A}$ arrive in the stream. A similar geometric argument of relating the sum of the online ℓ_1 sensitivities to the change in volume of some polytope induced by $\mathbf{A}$ does not seem obvious. A primary reason for this is that, unlike for ℓ_2, the unit ball $\{x \mid \|\mathbf{A}\mathbf{x}\|_1 \leq 1\}$ is not an ellipsoid and it is not clear how the John ellipsoid or the ℓ_1 sensitivities of this polytope change when a new row is added.

Instead, we first show that if the online ℓ_2 leverage scores are uniformly bounded by roughly $\frac{Cd\log\kappa}{n}$ given some regularization of the matrix for any constant $C > 1$, then the online ℓ_1 sensitivities must also be uniformly bounded by $\frac{Cd\log\kappa}{n}$. Now even under our regularization, the input matrix $\mathbf{A}$ does not have uniformly bounded online ℓ_2 leverage scores. We thus use a reweighting idea of [CLM+15] to argue that we can remove half of the rows of $\mathbf{A}$, while increasing the online ℓ_2 leverage scores of the other half of the rows of $\mathbf{A}$ to at most $\frac{Cd\log\kappa}{n}$, given our regularization. Since sensitivities only increase with the removal of rows, it follows that the sum of the (regularized) online ℓ_1 sensitivities of the half of the rows that remain with respect to $\mathbf{A}$ must be $\mathcal{O}(d\log\kappa)$. We then induct on the matrix formed by the removed rows to argue that the total sum of the (regularized) online ℓ_1 sensitivities of $\mathbf{A}$ must be $\mathcal{O}(d\log n\log\kappa)$, which implies a bound on the total number of sampled rows. This also implies the first online algorithm for an ℓ_1-subspace embedding. In fact, our analysis can be used to improve a number of other online algorithms.

[3]We again define this quantity to be 1 when a row is not in the span of the following rows.

Rank Constrained Online Algorithms. Our analysis for the sum of the online ridge leverage scores with regularization $\lambda = \frac{\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}$ immediately gives a nearly space optimal algorithm for low-rank approximation/projection-cost preservation in the online model. As in the sliding window setting, we do not know the value of λ in advance, but since our sketch is a rank k projection-cost preservation, we can track the evolution of $\frac{\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}$ as additional rows of $\mathbf{A}$ arrive in the stream. Correctness follows from the same matrix martingale argument as [CMP16] and the improved space bounds follow from our analysis bounding the sum of the online ridge leverage scores. We then show that an online algorithm providing a rank-k projection-cost preservation is a powerful primitive that can be used in conjunction with existing techniques to improve previous work.

Online Row Subset Selection. For row subset selection in the online model, our starting point is an offline algorithm by [CMM17], who observe that given a matrix $\mathbf{Z}$ that is a constant factor low-rank approximation to the underlying matrix $\mathbf{A}$, a theorem by [DRVW06] shows that adaptive sampling $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ additional rows $\mathbf{S}$ of $\mathbf{A}$ against the rows of $\mathbf{Z}$ suffices for $\mathbf{Z} \cup \mathbf{S}$ to contain a $(1+\varepsilon)$ factor approximation to the online row subset selection problem. Moreover, $\mathbf{Z} \cup \mathbf{S}$ contains a submatrix $\mathbf{T}$ of k rows that is a good low-rank approximation to $\mathbf{A}$. Since our online projection-cost preservation algorithm can output such a matrix $\mathbf{Z}$, we adapt the theorem of [DRVW06] to the streaming model, showing that if $\mathbf{Z}$ is given, adaptive sampling can also be performed on data streams to obtain a different but valid $\mathbf{S}$ by oversampling each row of $\mathbf{A}$ by an $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ factor.

We do not know $\mathbf{Z}$ until the end of the stream, so we cannot immediately perform adaptive sampling in the online model. Fortunately, if we adaptively sample against the current output of the online projection-cost preservation algorithm, then we will only oversample with respect to the true sampling probability. [CMM17] shows that the adaptive sampling probabilities can be upper bounded by the λ-ridge leverage scores, where $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$. Since the λ-ridge leverage scores are at most the online λ-ridge leverage scores, we can again sample rows proportional to their online λ-ridge leverage scores. It then suffices to again use our bound on the sum of the online ridge leverage scores to bound the number of total rows sampled by this algorithm.

Online Principal Component Analysis. Our starting point is a recent algorithm by [BLVZ19] that maintains and updates a matrix $\mathbf{X}$ throughout the data stream. After the arrival of row $\mathbf{a}_i$, the matrix $\mathbf{X}$ is updated using a combination of residual based sampling and a black-box theorem of Boutsidis et al. [BGKL15]. Row $\mathbf{m}_i$ is then output as the embedding of $\mathbf{a}_i$ into $\mathbf{X}$ by $\mathbf{m}_i = \mathbf{a}_i\mathbf{X}^{(i)}$, where $\mathbf{X}^{(i)}$ is the matrix $\mathbf{X}$ after row i has been processed by $\mathbf{X}$. $\mathbf{X}$ has the property that no rows from $\mathbf{X}$ are ever removed across the duration of the algorithm, so then $\mathbf{m}_i$ is only an upper bound on the best embedding of $\mathbf{a}_i$. However, this matrix $\mathbf{X}$ does not contain a good rank k approximation to $\mathbf{A}$. On the other hand, the output to our online row subset selection algorithm *does* contain a submatrix $\mathbf{Y}$ of k rows that is a good approximation to $\mathbf{A}$. Thus, our online row subset selection (RSS) algorithm can be combined with the algorithm of [BLVZ19] to output matrices $\mathbf{M}$ and $\mathbf{W}$ that is a good approximation to the online PCA problem but also so that $\mathbf{W}$ contains a submatrix $\mathbf{Y}$ of k rows that is a good rank k approximation of $\mathbf{A}$. Namely, the algorithm of [BLVZ19] can be run to produce a matrix $\mathbf{X}^{(i)}$ after the arrival of each row $\mathbf{a}_i$. Simultaneously, our online RSS algorithm produces a matrix $\mathbf{Z}^{(i)}$ after $\mathbf{a}_i$ arrives. We then immediately output the embedding $\mathbf{m}_i = \mathbf{a}_i\mathbf{W}^{(i)}$, where $\mathbf{W}^{(i)}$ appends the new rows of $\mathbf{X}^{(i)}$ and $\mathbf{Z}^{(i)}$ to $\mathbf{W}^{(i-1)}$. Since the dimension of $\mathbf{W}$ is the sum of the dimensions of $\mathbf{X}$ and $\mathbf{Z}$ but $\mathbf{X}$ has smaller dimension than $\mathbf{Z}$, we do not suffer additional asymptotic space over the algorithm of [BLVZ19].

Online Coresets for Deterministic Sliding Window Algorithms. Our algorithms have repeatedly hinted at a connection between row sampling algorithms for the online model and the sliding window model; if there exists an "online" probability distribution for a linear algebraic problem, then there seems to be a corresponding "reverse online" probability distribution. Online algorithms demand correctness on prefixes; sliding window algorithms demand correctness on suffixes. We formalize this intuition by showing a framework for *deterministic* sliding window algorithms using a merge-and-reduce framework based on online coresets, which we define to be a weighted subset of rows of an input matrix $\mathbf{A}$ that can be used to compute a good approximation to some given function on all prefixes of $\mathbf{A}$. Observe that any online algorithm in a row-arrival stream that succeeds with high probability *must* yield an online coreset. By a simple union bound, it must output the correct answer at all times. Thus, it must be correct for all prefixes of the stream. Since an online algorithm cannot revoke any of its sampled rows, it follows that we can just consider the rows output at the end of the algorithm and consider the subset of rows that were sampled prior to a certain time to obtain a good approximation to the corresponding prefix of $\mathbf{A}$. Hence, online algorithms imply the existence of an online coreset for the corresponding problem.

Given an online coreset for a particular problem, we obtain a corresponding deterministic sliding window algorithm, as in Figure 2. The idea is to store up to the most recent m rows in a block $\mathbf{B}_0$, for some parameter m related to the coreset size. When $\mathbf{B}_0$ becomes full, we create a $\left(1+\frac{\varepsilon}{\log n}\right)$ online coreset $\mathbf{B}_1$ for $\mathbf{B}_0$ starting with the most recent row. We then reset $\mathbf{B}_0$ to empty and start adding rows to $\mathbf{B}_0$ again. At some point $\mathbf{B}_0$ will contain m rows again. We then merge all of the rows $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_i$ where $\mathbf{B}_{i+1}$ is the first empty block. It can be shown that the merged rows form a $\left(1+\frac{\varepsilon}{\log n}\right)^i$ coreset for the most recent $2^i \cdot m$ rows, which we can reduce back down to m rows to form block $\mathbf{B}_{i+1}$. From a simple induction, it follows that using $\mathcal{O}(\log n)$ blocks will give a $\left(1+\frac{\varepsilon}{\log n}\right)^{\log n}$ coreset, starting with the most recent row. Rescaling ε, this gives a merge-and-reduce based framework for $(1+\varepsilon)$ deterministic sliding window algorithms based on online coresets.

ℓ_1-Subspace Embedding. For this framework, we can actually use a probability distribution for ℓ_1-subspace embeddings that is more space efficient than ℓ_1 sensitivities. The Lewis weights [Lew78, CP15] have been shown to be space optimal in the offline setting, up to lower order terms. However, their properties are less understood; rather than impossibility results, the challenge in using online Lewis weights in the row sampling framework is a gap in analysis. It did not seem evident how to approximate the online Lewis weight of $\mathbf{A}$ for a row $\mathbf{r}$, given a matrix $\mathbf{M}$ such that $\|\mathbf{M}\mathbf{x}\|_1 \approx \|\mathbf{A}\mathbf{x}\|_1$ for all $\mathbf{x} \in \mathbb{R}^d$. For the merge-and-reduce framework, we have access to each of the rows stored by an online coreset, so we can compute their online Lewis weights through an iterative process.

It then remains to bound the sum of the online Lewis weights to bound the size of the online coreset. We first prove elementary properties of the (online) Lewis weights, such as monotonicity to reweighting and a splitting invariance. We also show that if the online ℓ_2 leverage scores are uniformly bounded by roughly $\frac{Cd\log\kappa}{n}$ given some regularization of the matrix for any constant $C > 1$, then the online Lewis weights must also be uniformly bounded by $\frac{Cd\log\kappa}{n}$. We then use these properties to bound the online Lewis weights using the same uniformity of reweighting technique used to bound the online ℓ_1 sensitivities. Namely, we again argue that we can remove half of the rows of $\mathbf{A}$, while increasing the online Lewis weights of the other half of the rows of $\mathbf{A}$ to at most $\frac{Cd\log\kappa}{n}$, given our regularization. Since Lewis weights only increase with the removal of rows, the sum of the (regularized) online Lewis weights of the half of the rows that remain

with respect to $\mathbf{A}$ must be $\mathcal{O}(d \log \kappa)$. An inductive argument then shows that the total sum of the (regularized) online Lewis weights of $\mathbf{A}$ must be $\mathcal{O}(d \log n \log \kappa)$, which implies a bound on the online Lewis weights and thus the coreset size. This gives a more space efficient algorithm for ℓ_1-subspace embedding in the sliding window model compared to the previous row sampling framework.

1.3 Organization and Open Questions

In Section 2, we introduce a general framework for space and time efficient randomized matrix algorithms in the sliding window model. We use the framework to give algorithms for spectral approximation and low-rank approximation that are nearly space optimal and input sparsity runtime, up to lower order terms. We also provide a structural result that bounds the number of rows sampled by a probability distribution induced by online ridge leverage scores, which were recently introduced, but not fully explored. Our framework implicitly demonstrates connections between the sliding window and online models through the probability distribution used to sample each row and the corresponding analysis and we make this connection explicit in later sections.

In Section 3, we show that the paradigm of row sampling using online ridge leverage scores along with our structural result achieves simple and intuitive online algorithms for low-rank approximation, row subset selection, and principal component analysis that nevertheless improve on the state-of-the-art. In Section 4, we characterize and analyze an intuitive probability distribution for the ℓ_1-subspace embedding problem and show that it can be used to obtain both online and sliding window algorithms.

Finally, in Section 5, we give a general framework for deterministic matrix algorithms in the sliding window model using a merge-and-reduce paradigm for the concept of online coresets, which are generated by all of our online algorithms. We define and analyze a space optimal distribution for the ℓ_1-subspace embedding problem, so that in all, we obtain nearly space optimal deterministic algorithms for spectral approximation, low-rank approximation, and ℓ_1-subspace embeddings.

In Appendix A, we give background on the popular smooth histogram framework for sliding window algorithms and then give counterexamples showing the approach is not amenable to many interesting linear algebraic functions.

Our work leads to a number of interesting open questions. First, note that although we give the first online algorithm for ℓ_1-subspace embedding, the number of sampled rows is $\mathcal{O}\left(\frac{d^2}{\varepsilon^2}\log^2 n \log \kappa\right)$ from using online ℓ_1 sensitivities to determine the sampling probability for each row. This is because even though we show that the sum of the online ℓ_1 sensitivities is $\mathcal{O}(d \log n \log \kappa)$, the corresponding analysis requires an exponentially small probability of failure due to the ε-net argument. We show that the sum of the online Lewis weights is $\mathcal{O}(d \log n \log \kappa)$ and the analysis of [CP15] that uses offline Lewis weights does not require the ε-net, but the challenge is approximating the online Lewis weight of a row $\mathbf{a}_i$ given a sketch $\mathbf{M}$ for $\mathbf{A}$. Thus, a natural question is whether the online Lewis weights can be used to improve the sample complexity for online ℓ_1-subspace embedding.

Another interesting question is whether assumptions on the bit complexity of the underlying matrix $\mathbf{A}$ can remove the dependency on $\log \kappa$ for spectral approximation. We showed this is possible for low-rank approximation/projection-cost preservation by separately considering the case when $\mathbf{A}$ has rank at most $2k$, since we can efficiently upper bound $\frac{\|\mathbf{A}\|_F}{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F}$ when $\mathbf{A}$ has rank at least $2k$.

Finally, we describe how to construct online coresets in polynomial time for spectral approxi-

mation by generalizing an online version of [BSS12] by [CMP16]. Do there exist corresponding fast deterministic constructions of online coresets for low-rank approximation/projection-cost preservation and ℓ_1-subspace embeddings?

1.4 Preliminaries

For a positive integer n, we use $[n]$ to represent the set $\{1, \ldots, n\}$. We use $\frac{1}{\text{poly}(n)}$ to denote some arbitrary degree polynomial in n, generally some failure event that can be avoided by fixing sufficiently large constants. When an event has probability $1 - \frac{1}{\text{poly}(n)}$ of occurring, we say the event occurs with high probability. We use polylog(n) to omit terms that are polynomial in $\log n$ and write $\exp(n)$ to denote e^n.

In the row-arrival model, the stream has length n and the i^{th} update in the stream is precisely a row $\mathbf{r}_i$. In the sliding window model, the input matrix $\mathbf{A}$ is implicitly defined through the stream and a parameter $W > 0$ that represents the window size, so that $\mathbf{A}$ is the matrix consisting of the last W rows of the stream, $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$, where $\mathbf{a} \circ \mathbf{b}$ denotes the vertical concatenation of rows: $\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$. In the online model, the input matrix $\mathbf{A}$ is the matrix that consists of all n rows, so that $\mathbf{A} = \mathbf{r}_1 \circ \ldots \circ \mathbf{r}_n$.

We use $\mathbb{I}_n$ to denote the $n \times n$ identity matrix, but drop the subscript when the dimensions are clear from context. We use the notation $\mathbf{A}^\top$ to denote the transpose of $\mathbf{A}$ and $\mathbf{A}^{-1}$ to denote the Moore-Penrose pseudoinverse of $\mathbf{A}$ so that $\mathbf{A}\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}$, $\mathbf{A}^{-1}\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}$, $(\mathbf{A}\mathbf{A}^{-1})^\top = \mathbf{A}\mathbf{A}^{-1}$, and $(\mathbf{A}^{-1}\mathbf{A})^\top = \mathbf{A}^{-1}\mathbf{A}$. A symmetric matrix $\mathbf{A} \in \mathbf{R}^{n \times n}$ is positive semidefinite if $\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, in which case we say $0 \preceq \mathbf{A}$. Then the Loewner partial ordering of matrices has $\mathbf{A} \preceq \mathbf{B}$ if and only if $0 \preceq \mathbf{B} - \mathbf{A}$. If $\mathbf{A}$ has rank r, then we write its nonzero singular values as $\sigma_{\max}(\mathbf{A}) = \sigma_1(\mathbf{A}) \geq \ldots \geq \sigma_r(\mathbf{A}) = \sigma_{\min}(\mathbf{A})$. We define the condition number of $\mathbf{A}$ by $\frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}$ and the operator norm of $\mathbf{A}$ by $\|\mathbf{A}\|_2 = \sigma\text{max}(\mathbf{A})$. We use κ to represent the condition number of the stream; for the online model, the condition number of the stream is the maximum condition number across any matrix formed by prefixes of the stream, but for the sliding window model, the condition number of the stream is the maximum condition number across any matrix formed by up to W consecutive rows in the stream.

For a vector $\mathbf{x} \in \mathbb{R}^n$, we have the Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n v_i^2}$ and more generally, $\|\mathbf{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we denote its Frobenius norm by $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^d A_{i,j}^2}$. We denote $\mathbf{A}_{(k)} = \text{argmin}_{\text{rank}(\mathbf{X}) \leq k} \|\mathbf{A} - \mathbf{X}\|_F^2$ to be the best rank k approximation to $\mathbf{A}$.

We introduce and use multiple formulations, variants, and generalizations of (ridge) leverage scores, which we will define in their various sections, but we use the following definition of (ridge) leverage scores throughout.

Definition 1.8 ((Ridge) Leverage Scores)**.** *For a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ and a regularization parameter $\lambda \geq 0$, the ridge leverage score of row $\mathbf{a}_i$ for each $i \in [n]$ is the quantity $\mathbf{a}_i(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I})^{-1}\mathbf{a}_i^\top$. When $\lambda = 0$, we refer to the quantity as the leverage score of row $\mathbf{a}_i$.*

Informally, the (ridge) leverage score of row $\mathbf{a}_i$ quantifies how "important" or "unique" a row is. When the regularization parameter is $\frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2}{k}$, the intuition is that the importance of a row is only considered with respect to the top directions. Sampling rows with probability proportional to their leverage scores has been for spectral approximations [MD09, DMMW12] while sampling rows

Algorithm 1 Row sampling framework for matrix algorithms in the sliding window model

Input: A stream of rows $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$, window size W, and an accuracy parameter $\varepsilon > 0$
Output: A $(1+\varepsilon)$ approximation for various matrix functions in the sliding window model.

$\mathbf{M}_0 \leftarrow \emptyset$.
$\alpha \leftarrow \frac{C}{\varepsilon^2} \log n$ with sufficiently large constant $C > 0$
for each row $\mathbf{r}_t$ **do** ▷*Process stream*
  $\mathbf{M}_t = \mathbf{r}_t$ ▷*Keep timestamps of all rows*
  Let $\mathbf{M}_{t-1} = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_{m_{t-1}}$.
  for $i = m_{t-1}$ down to $i = 1$ **do**
    $\tau_i \leftarrow \textsc{Score}(\mathbf{m}_i, \mathbf{M}_t)$ ▷*Importance of row i based on matrix function*
    $p_i \leftarrow \min(1, \alpha\tau_i)$
    With probability p_i, $\mathbf{M}_t \leftarrow \frac{\mathbf{m}_i}{\sqrt{p_i}} \circ \mathbf{M}_t$ ▷*Downsample and rescale row*
  Delete $\mathbf{M}_{t-1}$.
$\mathbf{M} \leftarrow \emptyset$ ▷*Return rows relevant to sliding window*
Let $\mathbf{M}_n = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_{m_n}$.
for $i = 1$ to $i = m_n$ **do**
  if timestamp of $\mathbf{m}_i$ is at least $n - W + 1$ **then**
    $\mathbf{M} \leftarrow \mathbf{M} \circ \mathbf{m}_i$
return $\mathbf{M}$

with probability proportional to their ridge leverage scores has been used for low-rank approximations [AM15, CEM+15, CMM17].

2 Row Sampling Framework for the Sliding Window Model

In this section, we give space and time efficient algorithms for matrix functions in the sliding window model. Our general approach will be to use the following framework. As the stream $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ arrives, we shall maintain a weighted subset of these rows at each time. Suppose at some time t, we have a matrix $\mathbf{M}_t = \mathbf{r}_{t,1} \circ \ldots \circ \mathbf{r}_{t,m_t}$ of *weighted* rows of the stream that can be used to give a good approximation to the function applied to any *suffix* of the stream. Upon the arrival of row $t+1$, we first set $\mathbf{M}_{t+1} = \mathbf{r}_{t+1}$. Then starting with $i = m_t$ and moving backwards toward $i = 1$, we repeatedly prepend a weighted version of $\mathbf{r}_{t,i}$ to $\mathbf{M}_{t+1}$ with some probability that depends on $\mathbf{r}_{t,i}$, $\mathbf{M}_{t+1}$, and the matrix function to be approximated. Once the rows of $\mathbf{M}_t$ have each been either added to $\mathbf{M}_{t+1}$ or discarded, we proceed to row $t+2$.

Note that the matrices $\mathbf{M}_t$ serve no real purpose other than for presentation; the framework is just storing a subset of weighted rows at each time and repeatedly performing online row sampling, *starting with the most recent row*. Since an online algorithm must be correct on all prefixes of the input, then our framework must be correct on all suffixes of the input and in particular, on the sliding window. This observation demonstrates a connection between online algorithms and sliding window algorithms that we explore in greater detail in future sections. We give our framework in Algorithm 1.

Algorithm 2 SCORE$(\mathbf{r}, \mathbf{A})$ function for spectral approximation

Input: A row $\mathbf{r} \in \mathbb{R}^d$ and a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$.
Output: Scaled leverage score of $\mathbf{r}$ with respect to $\mathbf{A}$.

1: **if** $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A} \circ \mathbf{r})$ **then**
2: **return** $2\mathbf{r}(\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{r}^\top$ ▷*In particular,* SCORE$(\mathbf{m}_i, \mathbf{M}_t)$ *in Algorithm 1 is* $2\mathbf{m}_i(\mathbf{M}_t^\top \mathbf{M}_t)\mathbf{m}_i^\top$.
3: **else**
4: **return** 1

2.1 ℓ_2-Subspace Embedding

We first give a randomized algorithm for spectral approximation in the sliding window model that is both space and time efficient. [CMP16] define the concept of online (ridge) leverage scores and show that by sampling each row of a matrix $\mathbf{A}$ with probability proportional to its online leverage score, the weighted sample at the end of the stream provides a $(1+\varepsilon)$ spectral approximation to $\mathbf{A}$. We recall the definition of online ridge leverage scores of a matrix from [CMP16], as well as introduce reverse online ridge leverage scores.

Definition 2.1 (Online/Reverse Online (Ridge) Leverage Scores)**.** *For a matrix* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$*, let* $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ *and* $\mathbf{Z}_i = \mathbf{a}_n \circ \ldots \circ \mathbf{a}_i$*. Let* $\lambda \geq 0$*. The* online λ-ridge leverage score *of row* $\mathbf{a}_i$ *is defined to be* $\min(1, \mathbf{a}_i(\mathbf{A}_{i-1}^\top \mathbf{A}_{i-1} + \lambda \mathbb{I})^{-1}\mathbf{a}_i^\top)$*, while the* reverse online λ-ridge leverage score *of row* $\mathbf{a}_i$ *is defined to be* $\min(1, \mathbf{a}_i(\mathbf{Z}_{i+1}^\top \mathbf{Z}_{i+1} + \lambda \mathbb{I})^{-1}\mathbf{a}_i^\top)$*. The (reverse) online leverage scores are defined respectively by setting* $\lambda = 0$*, though we use the convention that if the (reverse) online leverage score of* $\mathbf{a}_i$ *is* 1 *if* $\text{rank}(\mathbf{A}_i) > \text{rank}(\mathbf{A}_{i-1})$ *(respectively if* $\text{rank}(\mathbf{Z}_i) > \text{rank}(\mathbf{Z}_{i+1})$*).*

Intuitively, the online leverage score quantifies how important row $\mathbf{a}_i$ is, with respect to the previous rows, while the reverse online leverage score quantifies how important row $\mathbf{a}_i$ is, with respect to the following rows, and the ridge leverage scores are regularized versions of these quantities. As the name suggests, online (ridge) leverage scores seem appropriate for online algorithms while reverse online (ridge) leverage scores seem appropriate for sliding window algorithms, where more recency is an emphasis. Hence, we use reverse online leverage scores in computing the sampling probability of each particular row in Algorithm 2 that serves as our customized SCORE function in Algorithm 1 for spectral approximation.

However, these quantities are related; [CMP16] provide an asymptotic bound on the sum of the online ridge leverage scores of any matrix, which also implies a bound on the sum of the reverse online ridge leverage scores, by reversing the order of the rows in a matrix.

Lemma 2.2 (Bound on Sum of Online Ridge Leverage Scores)**.** *[CMP16] Let the rows of* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ *arrive in a stream with condition number* κ *and let* ℓ_i *be the online (ridge) leverage score of* $\mathbf{a}_i$ *with regularization* λ*. Then* $\sum_{i=1}^n \ell_i = \mathcal{O}\left(d \log \frac{\|\mathbf{A}\|_2}{\lambda}\right)$ *for* $\lambda > \sigma_{\min}(\mathbf{A})$ *and* $\sum_{i=1}^n \ell_i = \mathcal{O}(d \log \kappa)$ *for* $\lambda \leq \sigma_{\min}(\mathbf{A})$*. It follows that if* τ_i *is the reverse online ridge leverage score of* $\mathbf{a}_i$*, then* $\sum_{i=1}^n \tau_i = \mathcal{O}\left(d \log \frac{\|\mathbf{A}\|_2}{\lambda}\right)$ *for* $\lambda > \sigma_{\min}(\mathbf{A})$ *and* $\sum_{i=1}^n \tau_i = \mathcal{O}(d \log \kappa)$ *for* $\lambda \leq \sigma_{\min}(\mathbf{A})$*.*

From the definition, it is evident that the reverse online (ridge) leverage scores are monotonic; whenever a new row is added to $\mathbf{A}$, the scores of existing rows cannot increase.

Lemma 2.3 (Monotonicity of Reverse Online (Ridge) Leverage Scores)**.** *For a matrix* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$*,* $\lambda \geq 0$*, and* $i \in [n]$*, let* $\tau_i(\mathbf{A})$ *denote the reverse online* λ*-ridge leverage score of*

row $\mathbf{a}_i$ *with respect to* $\mathbf{A}$ *and let* $\tau_i(\mathbf{B})$ *denote the reverse online* λ*-ridge leverage score of row* $\mathbf{a}_i$ *with respect to* $\mathbf{B} := \mathbf{A} \circ \mathbf{r}$ *for any row* $\mathbf{r} \in \mathbb{R}^d$*. Then* $\tau_i(\mathbf{A}) \geq \tau_i(\mathbf{B})$*.*

The proof follows immediately from Definition 2.1 and the fact that $\mathbf{A}_i^\top \mathbf{A}_i + \lambda \mathbb{I} \preceq (\mathbf{A}_i^\top \mathbf{A}_i + \mathbf{r}^\top \mathbf{r} + \lambda \mathbb{I})$ for any $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ and vector $\mathbf{r} \in \mathbb{R}^d$.

We also require the following version of the matrix Freedman concentration inequality:

Theorem 2.4 (Matrix Freedman Inequality)**.** *[Tro11] Let* $\mathbf{U}_0, \ldots, \mathbf{U}_i \in \mathbb{R}^{d \times d}$ *be a matrix martingale of symmetric matrices with difference sequence* $\mathbf{W}_1, \ldots, \mathbf{W}_i$*, where* $\mathbf{W}_j = \mathbf{U}_j - \mathbf{U}_{j-1}$*. If* $\|\mathbf{W}_j\|_2 \leq R$ *for each* $j \in [i]$ *with high probability and* $\left\|\sum_{j=1}^i \mathbb{E}\left[\mathbf{W}_j^2\right]\right\|_2 \leq \sigma^2$*, then for all* $\varepsilon > 0$*,*

$$\mathbf{Pr}\left[\|\mathbf{U}_i\|_2 \geq \varepsilon\right] \leq d \cdot \exp\left(-\frac{\varepsilon^2/2}{\sigma^2 + R\varepsilon/3}\right).$$

We now show that at any time t, Algorithm 1 using the SCORE function of Algorithm 2 stores a matrix $\mathbf{M}_t$ whose rows with timestamp after a time $i \in [t]$ provides a $(1+\varepsilon)$ spectral approximation to any matrix $\mathbf{Z}_i = \mathbf{r}_t \circ \ldots \circ \mathbf{r}_i$. This statement shows a good approximation to *any* suffix of the stream at all times and in particular for $t = n$ and $i = n - W + 1$, shows that Algorithm 1 using the SCORE function of Algorithm 2 outputs a spectral approximation for the matrix induced by the sliding window model.

Lemma 2.5 (Spectral Approximation Guarantee, Bounds on Sampling Probabilities)**.** *Let* $t \in [n]$*,* $\lambda \geq 0$ *and* $\varepsilon > 0$*. For* $i \in [t]$*, let* $q_i = \min(1, \alpha \cdot \mathbf{r}_i(\mathbf{Z}_{i+1}{}^\top \mathbf{Z}_{i+1})^{-1}\mathbf{r}_i^\top)$*, where* $\mathbf{Z}_{i+1} = \mathbf{r}_t \circ \ldots \circ \mathbf{r}_{i+1}$*. Then with high probability after the arrival of row* $\mathbf{r}_t$*, Algorithm 1 using the* SCORE *function of Algorithm 2 will have sampled each row* $\mathbf{r}_i$ *with probability at least* q_i *and probability at most* $4q_i$*. Moreover, if* $\mathbf{Y}$ *is the suffix of* $\mathbf{M}_t$ *consisting of the (scaled) rows whose timestamps are at least* i*, then*

$$(1-\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}) \preceq \mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I} \preceq (1+\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}).$$

Proof. We assume $\varepsilon \in \left(0, \frac{1}{2}\right)$ and give the proof by induction on t. For $t = 1$ and ignoring the trivial case where $\mathbf{r}_1$ is the all zeros row, then the input consists of a single nonzero row $\mathbf{r}_1$ whose reverse online leverage score is 1, so that Algorithm 1 using the SCORE function of Algorithm 2 will store $\mathbf{r}_1$, which completes the base case.

Now we suppose that the conditions hold for $t-1$ and prove they must also for t with high probability. We show the conditions hold for each $i \in [t]$ by following the sampling process from $\mathbf{r}_t$ down to $\mathbf{r}_i$. We also assume that Algorithm 1 has sampled each row $\mathbf{r}_i$ with probability $\widehat{p}_i$ at least $\frac{1}{2}q_i$ and probability at most $2q_i$. Let $k = t - i + 1$ and $\mathbf{U}_0, \ldots, \mathbf{U}_k \in \mathbb{R}^{d \times d}$ be a matrix martingale so that $\mathbf{U}_0$ is the all zeros matrix. We define $\mathbf{W}_j = \mathbf{U}_j - \mathbf{U}_{j-1}$ for each $j \in [k]$. For $j \geq 1$, we set $\mathbf{W}_j$ to be the all zeros matrix if $\|\mathbf{U}_{j-1}\|_2 \geq \varepsilon$ and otherwise if $\|\mathbf{U}_{j-1}\|_2 < \varepsilon$, we set the random matrix variable

$$\mathbf{W}_j = \begin{cases} \left(\frac{1}{\widehat{p}_j} - 1\right)\mathbf{s}_j^\top \mathbf{s}_j & \text{if } \mathbf{r}_{t-j+1} \text{ is sampled in } \mathbf{Y} \\ -\mathbf{s}_j^\top \mathbf{s}_j & \text{otherwise,} \end{cases}$$

where $\mathbf{s}_j := \mathbf{r}_{t-j+1}(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1/2}$ for each $j \in [k]$.

Thus, the difference sequence $\mathbf{W}_1, \ldots, \mathbf{W}_k$ defines

$$\mathbf{U}_{j-1} = (\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1/2}(\mathbf{Y}_{j-1}^\top \mathbf{Y}_{j-1} - \mathbf{Z}_{j-1}^\top \mathbf{Z}_{j-1})(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1/2},$$

where $\mathbf{Y}_{j-1}$ are the rows of $\mathbf{Y}$ with timestamp at least $j-1$. The predictable quadratic variation process of the martingale $\{\mathbf{U}_j\}$ is hence defined by $\sum_{q=1}^{j} \mathbb{E}\left[\mathbf{W}_q^2\right]$ for $1 \leq j \leq k$.

The remainder of the argument proceeds in the same manner as Lemma 3.3 of [CMP16]. Specifically, we first note that since $\|\mathbf{U}_{j-1}\|_2 < \varepsilon$ then either $\mathbf{W}_j$ is the all zeros matrix if $\widehat{p_j} = 1$ or if $\widehat{p_j} < 1$, then $\|\mathbf{W}_j\|_2 \leq \frac{2}{\alpha}$ by the assumption that $\widehat{p_j} \geq \frac{1}{2} q_j$ and the definition of q_j. Thus $\mathbb{E}\left[\mathbf{W}_j^2\right] \preceq \frac{2}{\alpha}\mathbf{s}_j^\top \mathbf{s}_j$ and so we can use the $\mathbf{W}_j$ matrices to bound the spectral norm of the predictable quadratic variation process $\left\|\sum_{x=1}^{j} \mathbb{E}\left[\mathbf{W}_x^2\right]\right\|_2$ of the martingale $\{\mathbf{U}_j\}$ by

$$\left\|\sum_{x=1}^{j} \mathbb{E}\left[\mathbf{W}_x^2\right]\right\|_2 \leq \left\|\sum_{x=1}^{j} \frac{2}{\alpha}\mathbf{s}_j^\top \mathbf{s}_j\right\|_2 \leq \frac{2}{\alpha}\left\|\sum_{x=1}^{j} \mathbf{r}_{t-j+1}^\top (\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1} \mathbf{r}_{t-j+1}\right\|_2 \leq \frac{2}{\alpha},$$

where the last inequality follows from the fact that $\sum_{x=1}^{j} \mathbf{r}_{t-j+1}^\top \mathbf{r}_{t-j+1} = \mathbf{Z}_i^\top \mathbf{Z}_i$.

By applying the Matrix Freedman inequality (Theorem 2.4), then it follows that $\|\mathbf{U}_k\|_2 < \varepsilon$ with probability at least $1 - d \cdot \exp\left(\frac{-\varepsilon^2/2}{2/\alpha + 2\varepsilon/(3\alpha)}\right)$. Since $\alpha = \frac{C}{\varepsilon^2}\log n$, then for sufficiently large C, $\|\mathbf{U}_k\|_2 < \varepsilon$ with high probability so that

$$\left\|(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1/2}(\mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I})(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1/2} - \mathbb{I}\right\|_2 \leq \varepsilon,$$

which implies that

$$(1-\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}) \preceq \mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I} \preceq (1+\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}),$$

which completes a single step of the second part of the claim.

Finally, consider the sampling probability of row $\mathbf{r}_{i-1}$. Conditioned on $\mathbf{Y}$ being a $(1+\varepsilon)$-spectral approximation to $\mathbf{Z}_i$, then $\mathbf{r}_{i-1}$ is in $\mathbf{M}_{t-1}$ with some probability $\gamma \leq 1$ but has been rescaled to $\frac{1}{\sqrt{\gamma}}\mathbf{r}_{i-1}$. Crucially, the online ridge leverage scores are monotonic by Lemma 2.3 so that $\gamma \geq \min(1, \alpha(\mathbf{r}_{i-1}(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1}\mathbf{r}_{i-1}^\top))$. Although we only have a $(1+\varepsilon)$-spectral approximation to $\mathbf{Z}_i$, the SCORE function of Algorithm 2 increases the sampling probability by an extra factor of 2 to compensate. Thus conditioned on $\mathbf{r}_{i-1}$ being in $\mathbf{M}_{t-1}$, then $\mathbf{r}_{i-1}$ remains in $\mathbf{M}_t$ with probability

$$\min\left(1, \frac{2}{\gamma}\alpha(\mathbf{r}_{i-1}(\mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I})^{-1}\mathbf{r}_{i-1}^\top)\right).$$

Hence, the overall probability that $\mathbf{r}_{i-1}$ is retained is at most $\min(1, 4\alpha(\mathbf{r}_{i-1}(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1}\mathbf{r}_{i-1}^\top))$ and at least $\min(1, \alpha(\mathbf{r}_{i-1}(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I})^{-1}\mathbf{r}_{i-1}^\top))$ for $\varepsilon < \frac{1}{2}$, which completes a single step of the first part of the claim. Thus, both parts of the claim hold by induction. □

Theorem 2.6 (Randomized Spectral Approximation Sliding Window Algorithm)**.** *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ be a stream of rows and κ be the condition number of the stream. Let $W > 0$ be a window size parameter and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. Given a parameter $\varepsilon > 0$, there exists an algorithm that outputs a matrix $\mathbf{M}$ with a subset of (rescaled) rows of $\mathbf{A}$ such that $(1-\varepsilon)\mathbf{A}^\top \mathbf{A} \preceq \mathbf{M}^\top \mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top \mathbf{A}$ and stores $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n \log \kappa\right)$ rows at any time, with high probability.*

Proof. The fact that $(1-\varepsilon)\mathbf{A}^\top\mathbf{A} \preceq \mathbf{M}^\top\mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top\mathbf{A}$ holds immediately from Lemma 2.5 using $t = n$ and $i = n - W + 1$. Moreover, Lemma 2.5 implies that each row $\mathbf{r}_i$ is sampled with probability at most $4\alpha\tau_i$, where τ_i is the reverse online leverage score of row i. By Lemma 2.2, we have $\sum_{i=1}^{n}\tau_i = \mathcal{O}(d\log\kappa)$ and since $\alpha = \mathcal{O}\left(\frac{1}{\varepsilon^2}\log n\right)$, then the space complexity of the algorithm follows from a coupling argument and standard Chernoff bounds. ☐

Nearly Input Sparsity Runtime. We remark that the amortized running time per arriving row can be improved by batching, i.e. processing $\frac{d}{\varepsilon^2}\log n\log\kappa$ rows at a time. Since Algorithm 1 using the SCORE function of Algorithm 2 stores $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n\log\kappa\right)$ rows, the asymptotic space used by the algorithm will remain the same. Observe that it suffices to obtain some constant factor of the reverse online leverage score, since α can just be scaled accordingly.

To compute a constant factor approximation of the reverse online leverage score of any sampled row $\mathbf{a}_i$, we use standard projection tricks [SS11, CMP16, CMM17] embedding each of the $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n\log\kappa\right)$ rows into a $\mathcal{O}\left(\log\frac{n}{\varepsilon}\right)$ dimension subspace by applying a Johnson-Lindenstrauss transform. Applying this subspace embedding to each of the rows takes $\mathcal{O}\left(\log\frac{n}{\varepsilon}\cdot\overline{\text{nnz}}\right)$ time, where $\overline{\text{nnz}}$ is the input sparsity of the batch. Subsequently, all operations are in $\mathcal{O}\left(\log\frac{n}{\varepsilon}\right)$ dimension subspace, so approximating each reverse online leverage score requires $\text{polylog}\,\frac{n}{\varepsilon}$ time for each of the $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n\log\kappa\right)$ scores. Hence, the amortized time across each batch of $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log n\log\kappa\right)$ rows is $\text{polylog}\,\frac{n}{\varepsilon}\cdot\overline{\text{nnz}}$ so the total runtime is $\text{polylog}\,\frac{n}{\varepsilon}\cdot\text{nnz}$, where nnz is the input sparsity of the stream. One might observe that the algorithm does not actually know κ in advance, but for algorithmic purposes it suffices to downsample when the number of newly arrived rows in the batch equals the number of stored rows after the previous downsampling procedure.

2.2 Low-Rank Approximation

In this section, we give a randomized algorithm for low-rank approximation in the sliding window model that is both space and time optimal, up to lower order terms. Throughout this section, we use $\mathbf{A}_{(k)}$ to denote the best rank k approximation to a matrix $\mathbf{A}\in\mathbb{R}^{n\times d}$ so that $\mathbf{A}_{(k)} := \text{argmin}_{\text{rank}(\mathbf{X})\le k}\|\mathbf{A}-\mathbf{X}\|_F^2$. Recall the definition of a projection-cost preservation in Definition 1.1, from which it follows that obtaining a projection-cost preservation of $\mathbf{A}$ suffices to produce a low-rank approximation of $\mathbf{A}$. [CMM17] show that an additive-multiplicative spectral approximation of a matrix $\mathbf{A}$ along with an additional moderate condition that holds for ridge leverage score sampling gives a projection-cost preservation of $\mathbf{A}$.

Lemma 2.7. *[CMM17] Let $\mathbf{A} = \mathbf{a}_1\circ\ldots\circ\mathbf{a}_n\in\mathbb{R}^{n\times d}$ and $\lambda \le \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$. Let p be the largest integer such that $\sigma_p(\mathbf{A})^2\ge\lambda$ and let $\mathbf{X} = \mathbf{A}-\mathbf{A}_{(p)}$. Let $\mathbf{S}\in\mathbb{R}^{m\times n}$ be a sampling matrix so that $\mathbf{M} = \mathbf{AS}$ is a subset of scaled rows of $\mathbf{A}$. If $(1-\varepsilon)(\mathbf{A}^\top\mathbf{A}+\lambda\mathbb{I}) \preceq \mathbf{M}^\top\mathbf{M}+\lambda\mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}^\top\mathbf{A}+\lambda\mathbb{I})$ and $\left|\|\mathbf{SX}\|_F^2 - \|\mathbf{X}\|_F^2\right| \le \varepsilon\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$, then $\mathbf{M}$ is a rank k projection-cost preservation of $\mathbf{A}$ with approximation parameter 24ε.*

We focus our discussion on the additive-multiplicatve spectral approximation since the same argument of [CMM17] with Freedman's inequality rather than Chernoff bounds shows sampling matrices generated from ridge leverage scores satisfy the condition $\left|\|\mathbf{SX}\|_F^2 - \|\mathbf{X}\|_F^2\right| \le \varepsilon\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$ with high probability, even when the entries of $\mathbf{S}$ are not independent.

Algorithm 3 SCORE$(\mathbf{r}, \mathbf{A})$ function for rank k projection-cost preservation

Input: A row $\mathbf{r} \in \mathbb{R}^d$ and a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$.
Output: Scaled ridge leverage score of $\mathbf{r}$ with respect to $\mathbf{A}$.

$\lambda \leftarrow \frac{1}{k}\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$
if $\lambda \neq 0$ or $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}(\mathbf{A} \circ \mathbf{r})$ **then**
    return $2\mathbf{r}(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I})^{-1}\mathbf{r}^\top$
else
    return 1

Lemma 2.8. *[CMM17] Let $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$, $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$, and τ_i be the ridge leverage score of $\mathbf{a}_i$ with regularization λ. Let p be the largest integer such that $\sigma_p(\mathbf{A})^2 \geq \lambda$ and let $\mathbf{X} = \mathbf{A} - \mathbf{A}_{(p)}$. Let $\mathbf{S} \in \mathbb{R}^{m \times n}$ be a sampling matrix so that row $\mathbf{a}_i$ is sampled by $\mathbf{S}$, not necessarily independently, with probability at least $\min\left(1, \frac{C\tau_i}{\varepsilon^2}\log n\right)$ for sufficiently large constant C. Then $\left|\|\mathbf{SX}\|_F^2 - \|\mathbf{X}\|_F^2\right| \leq \varepsilon\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$ with high probability.*

On the other hand, our space analysis in Section 2.1 relied on bounding the sum of the online leverage scores by $\mathcal{O}(d \log \kappa)$ through Lemma 2.2; a better bound is not known if we set $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$. This gap provides a barrier for algorithmic design not only in the sliding window model but also in the online model. We show a tighter analysis showing that the sum of the online ridge leverage scores for $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$ is $\mathcal{O}(k \log \kappa)$.

Now if we knew the value of $\frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$ a priori, we could set $\lambda \leq \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$ and immediately apply Lemma 2.5 to show that the output of Algorithm 1 with a SCORE function that uses the λ regularization outputs a matrix $\mathbf{M}$ that is a rank k projection-cost preservation of $\mathbf{A}$.

Initially, even a constant factor approximation to $\frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$ seems challenging because the quantity is not smooth. This issue can be circumvented using additional procedures, such as spectral approximation on rows with reduced dimension [CEM+15]. Even simpler, observe that sampling with any regularization factor $\lambda < \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$ would still provide the guarantees of Lemma 2.7.

We could set $\lambda = 0$ and still obtain a rank k projection-cost preservation of $\mathbf{A}$, but larger values of λ correspond to smaller number of sampled rows and the total number of sampled rows for $\lambda = 0$ would be proportional to d, as opposed to our goal of k. Instead, observe that if $\mathbf{B}$ is any prefix or suffix of rows of $\mathbf{A}$, then $\left\|\mathbf{B} - \mathbf{B}_{(k)}\right\|_F^2 \leq \left\|\mathbf{B} - \mathbf{B}_{(k)}\right\|_F^2$. In other words, we can use the rows that have already been sampled to give a constant factor approximation to $\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$ as it evolves, i.e., as more rows of $\mathbf{A}$ arrive. We *again* pay for the underestimate to $\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$ by sampling an additional number of rows, but we show that we cannot sample too many rows before our approximation to $\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$ doubles, which only incurs an additional $\mathcal{O}(\log \kappa)$ factor in the number of sampled rows. We give the SCORE function for low-rank approximation in Algorithm 3.

We first bound the probability that each row is sampled, analogous to Lemma 2.5.

Lemma 2.9 (Projection-Cost Preservation Guarantee, Bounds on Sampling Probabilities)**.** *Let $t \in [n]$ be fixed and for each $i \in [t]$, let $\mathbf{Z}_i = \mathbf{r}_t \circ \ldots \circ \mathbf{r}_i$. Let $\lambda = \frac{\left\|\mathbf{Z}_{i+1} - (\mathbf{Z}_{i+1})_{(k)}\right\|_F^2}{k}$ and $\varepsilon > 0$. Let $q_i = \min(1, \alpha \cdot \mathbf{r}_i(\mathbf{Z}_{i+1}{}^\top \mathbf{Z}_{i+1} + \lambda \mathbb{I})^{-1}\mathbf{r}_i^\top)$. Then with high probability after the arrival of row $\mathbf{r}_t$,*

Algorithm 1 using the SCORE *function of Algorithm 3 will have sampled row* $\mathbf{r}_i$ *with probability at least* q_i *and probability at most* $2q_i$*. Moreover, if* $\mathbf{Y}$ *is the suffix of* $\mathbf{M}_t$ *consisting of the (scaled) rows whose timestamps are at least* i*, then*

$$(1-\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}) \preceq \mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I} \preceq (1+\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}).$$

Proof. Consider Algorithm 1 using the SCORE function of Algorithm 3. We again assume $\varepsilon \in \left(0, \frac{1}{2}\right)$ and prove the statement by induction. Recall the convention that if $\lambda = 0$ and $\mathbf{r}_1$ is nonzero, then the online leverage score of $\mathbf{r}_1$ is 1 rather than 0. Thus, the base case holds for $t = 1$ because any nonzero will be sampled.

Now suppose the claim holds for $t-1$ and all indices $i \in [t-1]$ after the arrival of row $\mathbf{r}_{t-1}$. We show it holds for $i = t$ down to $i = 1$ after the arrival of row $\mathbf{r}_t$. Consider a fixed $i \in [t]$. By Lemma 2.7 and Lemma 2.8, the additive-multiplicative spectral guarantee of the claim along with the arrival of row $\mathbf{r}_t$ gives a $(1+24\varepsilon)$ approximation to $\left\|\mathbf{Z}_{i+1} - (\mathbf{Z}_{i+1})_{(k)}\right\|_F^2$. Note that the error does not compound since the SCORE function scales the sampling probability by 2 and thus the SCORE function of Algorithm 3 effectively computes the reverse online ridge leverage score of each row $\mathbf{r}_j$ with $j \geq t-i+1$ using an underestimate of $\lambda = \frac{\left\|\mathbf{Z}_{i+1} - (\mathbf{Z}_{i+1})_{(k)}\right\|_F^2}{k}$. The matrix martingale argument in Lemma 2.5 then shows that

$$(1-\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}) \preceq \mathbf{Y}^\top \mathbf{Y} + \lambda \mathbb{I} \preceq (1+\varepsilon)(\mathbf{Z}_i^\top \mathbf{Z}_i + \lambda \mathbb{I}),$$

which completes a single step of the second part of the claim. By Lemma 2.7 and Lemma 2.8, it follows that $\left\|\mathbf{Y} - \mathbf{Y}_{(k)}\right\|_F^2$ is a $(1+\mathcal{O}(\varepsilon))$-approximation to $\left\|\mathbf{Z}_{i+1} - (\mathbf{Z}_{i+1})_{(k)}\right\|_F^2$. The bounds on the sampling probability of row $\mathbf{r}_i$ then follows from the fact that the reverse online ridge leverage scores are monotonic by Lemma 2.3. Though we only have a $(1+\varepsilon)$ additive-multiplicative spectral approximation to $\mathbf{Z}_i$, which translates to a $(1+24\varepsilon)$ approximation of the regularization, the SCORE function of Algorithm 3 has again compensated by increasing the sampling probability by a factor of 2. The entire claim then follows by induction. ∎

We now give a tighter analysis of the sum of the online λ-ridge leverage scores l_i for $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_2^2}{k}$. We require an upper bound from [CMP16] on the sum of the online λ-ridge leverage scores, proven using the matrix determinant lemma.

Lemma 2.10. *[CMP16] For a matrix* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$*, let* l_i *denote the online* λ*-ridge leverage score of* $\mathbf{a}_i$*, for each* $i \in [n]$*. Then* $\det(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}) \geq \lambda^d e^{\sum l_i / 2}$.

Lemma 2.11 (Bound on Sum of Online Ridge Leverage Scores)**.** *Let* $\mathbf{A} \in \mathbb{R}^{n \times d}$ *have condition number* κ*. Let* $\beta \geq 1$*,* $k \geq 1$ *be constants and* $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_2^2}{\beta k}$*. Then* $\sum_{i=1}^n l_i = \mathcal{O}(k \log \kappa)$.

Proof. Since $\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I} \succeq \lambda \mathbb{I}$, let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq \lambda$ be the singular values of $\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}$. Then $\det(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}) = \prod_{i=1}^d \sigma_i$. Observe that $\sigma_{k+1} + \ldots + \sigma_d = \left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2 + (d-k)\lambda$ by the Eckart-Young-Mirsky theorem. By the AM-GM inequality, we have

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

Combining with the fact that $\sigma_i \leq \|\mathbf{A}\|_2^2 + \lambda$ for $1 \leq i \leq k$, we have for $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{\beta k}$,

$$\det(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}) = \prod_{i=1}^{d} \sigma_i \leq (\|\mathbf{A}\|_2^2 + \lambda)^k \left(\frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2}{d-k} + \lambda \right)^{d-k}$$
$$\leq (\|\mathbf{A}\|_2^2 + \lambda)^k \lambda^{d-k} \left(\frac{\beta k}{d-k} + 1 \right)^{d-k} \leq (2\|\mathbf{A}\|_F^2)^k \lambda^{d-k} e^{\beta k}.$$

Combining with Lemma 2.10 and taking logarithms, it follows that $d \log \lambda + \sum \frac{l_i}{2} \leq k + 2k \log 2 \|\mathbf{A}\|_F + (d-k)\log\lambda + \beta k$. Since $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{\beta k}$, then $\sum_i l_i \leq 2(1+\beta)k + 4k \log 2\kappa$. □

The sum of the reverse online λ-ridge leverage scores is bounded by the same quantity, since the rows of the input matrix can simply be considered in reverse order. We now show that Algorithm 1 using the SCORE function of Algorithm 3 gives a relative error low-rank approximation with efficient space usage.

Theorem 2.12 (Randomized Low-Rank Approximation Sliding Window Algorithm)**.** *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ be a stream of rows and κ be the condition number of the matrix $\mathbf{r}_1 \circ \ldots \circ \mathbf{r}_n$. Let $W > 0$ be a window size parameter and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. Given a parameter $\varepsilon > 0$, there exists an algorithm that outputs a matrix $\mathbf{M}$ that is a $(1+\varepsilon)$ rank k projection-cost preservation of $\mathbf{A}$ and stores $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ rows at any time, with high probability.*

Proof. Let $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$. Then $(1-\varepsilon)(\mathbf{A}^\top\mathbf{A} + \lambda\mathbb{I}) \preceq \mathbf{M}^\top\mathbf{M} + \lambda\mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}^\top\mathbf{A} + \lambda\mathbb{I})$ holds immediately from Lemma 2.9 using $t = n$ and $i = n - W + 1$. Thus by Lemma 2.7, Lemma 2.8, and rescaling ε, it follows that $\mathbf{M}$ is a $(1+\varepsilon)$ rank k projection-cost preservation of $\mathbf{A}$.

To bound the number of sampled rows, observe that we can sample at most k linearly independent rows before the regularization becomes nonzero. Then by Lemma 2.9 each row $\mathbf{r}_i$ is sampled with probability at most $\min(1, 2c \cdot \mathbf{r}_i(\mathbf{Z}_{i+1}^\top \mathbf{Z}_{i+1} + \lambda_i \mathbb{I})^{-1} \mathbf{r}_i^\top)$, where $\mathbf{Z}_i = \mathbf{r}_i \circ \ldots \circ \mathbf{r}_n$ and $\lambda_i = \frac{\|\mathbf{Z}_{i+1} - (\mathbf{Z}_{i+1})_{(k)}\|_F^2}{k}$ for each $i \in [n]$. For the purposes of analysis, set $u_0 = n$ and for each $i \geq 1$, let u_i be the largest index j such that $\lambda_j > 2\lambda_{u_{i-1}}$. This partitions the stream into breakpoints u_i starting from the most recent row, so that for each $u_{i+1} < t \leq u_i$, we have $\lambda_{u_i} \leq \lambda_t < 2\lambda_{u_i}$. Thus by Lemma 2.11, the sum of the reverse online ridge leverage scores for the rows $\mathbf{r}_t$ with $u_{i+1} < t \leq u_i$ is $\mathcal{O}(k \log \kappa)$. Algorithm 1 scales the reverse online ridge leverage score by a factor of $\alpha = \mathcal{O}\left(\frac{1}{\varepsilon^2} \log n\right)$ to determine the sampling probability, so by a coupling argument and standard Chernoff bounds, the number of sampled rows $\mathbf{r}_t$ with $u_{i+1} < t \leq u_i$ is $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log \kappa\right)$. Since there are $\mathcal{O}(\log \kappa)$ such breakpoints u_i, the total number of sampled rows is $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$. □

Nearly Input Sparsity Runtime. Nearly input sparsity runtime can be achieved through similar batching and dimensionality reduction techniques as Section 2.1. Since Algorithm 1 using the SCORE function of Algorithm 3 stores $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ rows, the asymptotic space used by the algorithm will remain the same if it processes $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ rows at a time. To compute a constant factor approximation of the reverse online ridge leverage score of any sampled row $\mathbf{a}_i$, each of the $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ rows into a $\mathcal{O}\left(\log \frac{n}{\varepsilon}\right)$ dimension subspace by applying a Johnson-Lindenstrauss transform [SS11, CMP16, CMM17]. Applying this subspace embedding to each of

the rows takes $\mathcal{O}\left(\log \frac{n}{\varepsilon} \cdot \overline{\text{nnz}}\right)$ time, where $\overline{\text{nnz}}$ is the input sparsity of the batch. Subsequently, all operations are in $\mathcal{O}\left(\log \frac{n}{\varepsilon}\right)$ dimension subspace, so approximating the each regularization parameter λ and each reverse online leverage score requires $\text{polylog}\, \frac{n}{\varepsilon}$ time for each of the $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log n \log^2 \kappa\right)$ scores. Hence, the total runtime is $\text{polylog}\, \frac{n}{\varepsilon} \cdot \text{nnz}$, where nnz is the input sparsity of the stream. Again the algorithm need not know κ in advance since it suffices to process a batch when the number of newly arrived rows in the batch equals the number of stored rows after the process.

Bounded Precision and Condition Number. In typical applications, the entries of the underlying matrix will have some bounded precision, say $\mathcal{O}(\log n)$ bits. From invariance through scaling the entries, we assume that the entries of the matrix are integers whose magnitudes are bounded by $\text{poly}(n)$. Nevertheless, the condition number κ of the underlying matrix can be as large as $\text{poly}(n)^d$, which can cause the $\log \kappa$ term to incur an additional factor of $\mathcal{O}(\delta \log n)$. We show this is not necessary by reducing the $\log \kappa$ factor down to $\log n$ in some cases and removing the $\log \kappa$ factor altogether in other cases. We first relate matrices $\mathbf{A}$ with bounded entries and sufficiently large rank with a bound on $\frac{\|\mathbf{A}\|_F}{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F}$.

Lemma 2.13. *[CW09] If $\mathbf{A} \in \mathbb{R}^{n \times d}$ has integer entries bounded in magnitude by $\text{poly}(n)$ and rank at least $2k$, then $\frac{\|\mathbf{A}\|_F}{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F} \leq \text{poly}(n, d)$ as $nd \to \infty$.*

By Lemma 2.13, we have $\mathcal{O}(\log \kappa) = \mathcal{O}(\log n)$ if $\text{rank}(\mathbf{A}) \geq 2k$. Moreover, we do not incur the extra $\mathcal{O}(\log n)$ factor from Chernoff bounds to achieve high probability over all times in the stream, so the total number of sampled rows is $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log^2 n\right)$. Thus it suffices to maintain a rank k projection-cost preservation in the case that $\text{rank}(\mathbf{A}) \leq 2k$.

Observe that we can decompose $\mathbf{A}^\top \mathbf{A} = \mathbf{R}^\top \mathbf{U} \mathbf{R}$, where $\mathbf{R} \in \mathbb{R}^{2k \times d}$ is the row span of $\mathbf{A}$. Now the row span of any suffix of $\mathbf{A}$ will be contained within $\mathbf{R}$. Thus it suffices to maintain a sequence of matrices $\mathbf{U}_1, \mathbf{U}_2, \ldots \in \mathbb{R}^{2k \times 2k}$ so that $\mathbf{R}^\top \mathbf{U}_1 \mathbf{R}, \mathbf{R}^\top \mathbf{U}_2 \mathbf{R}, \ldots$ provides a $(1+\varepsilon)$ spectral approximation to all suffixes of $\mathbf{A}$. We give the algorithm in Algorithm 4 and the guarantees in Theorem 2.14, but defer the full proof to Appendix A.3, as it is a special case of a more general data structure that we introduce for deterministic algorithms for numerical linear algebra in the sliding window model. The key observation is that a new matrix $\mathbf{U}_i$ is needed only when some singular value of $\mathbf{A}$ increase by $(1+\varepsilon)$. Thus the number of matrices $\mathbf{U}_i$ that are needed is $\mathcal{O}\left(\frac{1}{\varepsilon} \log \kappa\right)$, where κ is the condition number of $\mathbf{A}$. Since the entries of $\mathbf{A}$ are bounded integers, then the characteristic polynomial of $\mathbf{A}^\top \mathbf{A}$ has integer coefficients bounded by $(\text{poly}(n))^{\mathcal{O}(k)}$. Similarly, the largest eigenvalue of $\mathbf{A}^\top \mathbf{A}$ is bounded by its Frobenius norm, which is bounded by $\text{poly}(n)$, so then $\log \kappa = \mathcal{O}(k \log n)$. As each matrix $\mathbf{U}_i$ has dimension $2k \times 2k$, the space bound follows.

Theorem 2.14. *There exists a deterministic algorithm in the sliding window model that outputs a rank k projection-cost preservation of an input matrix $\mathbf{A}$ whose rank is at most $2k$. If the entries of $\mathbf{A}$ are integers that are bounded in magnitude by $\text{poly}(\mathbf{A})$, then this algorithm stores $\mathcal{O}(k)$ rows of $\mathbf{A}$ and uses $\mathcal{O}\left(\frac{k^4}{\varepsilon} \log n\right)$ additional words of space.*

Given Theorem 2.14 to handle $\text{rank}(\mathbf{A}) \leq 2k$ and Lemma 2.13 to handle $\text{rank}(\mathbf{A}) \geq 2k$ along with Theorem 2.12 and the previous observation that we no longer need to incur an extra $\mathcal{O}(\log n)$ factor for Chernoff bounds to achieve high probability over all times in the stream, then we have the following guarantees for low-rank approximation/projection-cost preservation when the underlying matrix has integer entries that are bounded in magnitude by $\text{poly}(n)$.

Algorithm 4 Projection-cost preservation for low-rank matrices in the sliding window model

Input: A stream of rows $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$, window size W, and an accuracy parameter $\varepsilon > 0$
Output: Rank k projection-cost preservation in the sliding window model.

$\mathbf{M}_0 \leftarrow 0^{d \times d}$
for each row $\mathbf{r}_t$ **do**
  Suppose $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_s$ are defined
  $\mathbf{R}_{s+1} \leftarrow \mathbf{r}_t$, $\mathbf{M}_{s+1} \leftarrow \mathbf{R}_{s+1}^\top \mathbf{R}_{s+1}$ ▷*Keep timestamp and decomposition for* $\mathbf{M}_{s+1}$
  for $i = s$ to $i = 1$ **do** ▷*Update sketches with* $\mathbf{r}_t$
    Let $\mathbf{M}_i = \mathbf{R}_i^\top \mathbf{U}_i \mathbf{R}_i$ and $\mathbf{B}_i$ be a basis for $\mathbf{R}_i \setminus \mathbf{R}_{i+1}$.
    $\mathbf{R}_i \leftarrow \mathbf{R}_{i+1} \circ \mathbf{B}_i$ ▷*Ensures* $\mathbf{R}_{i+1} \subseteq \mathbf{R}_i$
    Set $\mathbf{U}_i$ so that $\mathbf{R}_i^\top \mathbf{U}_i \mathbf{R}_i = \mathbf{M}_i + \mathbf{r}_t^\top \mathbf{r}_t$.
  for $i = s$ to $i = 2$ **do**
    if $\mathbf{M}_{i-1}^\top \mathbf{M}_{i-1} \preceq (1+\varepsilon)\mathbf{M}_{i+1}^\top \mathbf{M}_{i+1}$ **then**
      Delete $\mathbf{M}_i$ and relabel indices
  if $\mathbf{M}_2$ has expired **then** ▷*Timestamp is at most* $t - W + 1$
    Delete $\mathbf{M}_1$ and relabel indices
return $\mathbf{M}_1$

Corollary 2.15. *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ be a stream of rows of integers whose magnitudes are at most* $\mathrm{poly}(n)$*. Let $W > 0$ be a window size parameter and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. Given a parameter $\varepsilon > 0$, there exists an algorithm that with high probability, outputs a matrix $\mathbf{M}$ that is a $(1+\varepsilon)$ rank k projection-cost preservation of $\mathbf{A}$, stores $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log^2 n\right)$ rows at any time, and uses $\mathcal{O}\left(\frac{k^4}{\varepsilon}\log n\right)$ additional words of space.*

3 Simple Rank Constrained Algorithms in the Online Model

In this section, we show that the paradigm of row sampling with respect to online ridge leverage scores offers simple analysis for a number of online algorithms that improve upon the state-of-the-art.

3.1 Online Projection-Cost Preservation

As a warm-up, we first demonstrate how our previous analysis bounding the sum of the online ridge leverage scores can be applied to analyze a natural online algorithm for producing a projection-cost preservation of a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$. Our algorithm samples each row with probability equal to the online ridge leverage scores, where the regularization parameter λ_i is computed at each step. Note that if $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$, then $\left\|\mathbf{A}_i - (\mathbf{A}_i)_{(k)}\right\|_F^2 \leq \left\|\mathbf{A}_j - (\mathbf{A}_j)_{(k)}\right\|_F^2$ for any $i < j$. Thus if $\lambda_i = \frac{\left\|\mathbf{A}_{i-1} - (\mathbf{A}_{i-1})_{(k)}\right\|_F^2}{k}$, then sampling row $\mathbf{a}_i$ with online ridge leverage score regularized by λ_i has a *higher* probability than with ridge leverage score regularized by $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$. Although [CMP16] is only interested in spectral approximation and therefore set $\lambda = \varepsilon\sigma_{\min}(\mathbf{A})$, they nevertheless shows that online row sampling with any regularization λ gives an additive-multiplicative spectral approximation to $\mathbf{A}$. Thus by setting $\lambda = \frac{\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2}{k}$, our algorithm outputs a rank k projection-

cost preservation of $\mathbf{A}$ by Lemma 2.7 and Lemma 2.8. Moreover, our bounds for the sum of the online ridge leverage scores in Lemma 2.11 show that our algorithm only samples a small number of rows, optimal up to lower order factors.

Algorithm 5 OnlinePCP: Online algorithm for projection-cost preservation

Input: Stream of rows $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{1\times d}$, accuracy $\varepsilon > 0$, and parameter k.
Output: Rank k projection-cost preservation of $\mathbf{A} := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$.

$\mathbf{M} \leftarrow \emptyset$.
$\alpha \leftarrow \frac{C}{\varepsilon^2}\log n$ with sufficiently large constant $C > 0$
for each row $\mathbf{a}_t$ **do**
  $\widetilde{\lambda}_t \leftarrow \frac{\|\mathbf{M}-\mathbf{M}_{(k)}\|_F^2}{2k}$
  $\tau_t \leftarrow 2\mathbf{a}_t(\mathbf{M}^\top\mathbf{M} + \widetilde{\lambda}_t\mathbb{I})^{-1}\mathbf{a}_t^\top$
  $p_t \leftarrow \min(1, \alpha\tau_t)$
  With probability p_t, $\mathbf{M} \leftarrow \mathbf{M} \circ \frac{\mathbf{a}_t}{\sqrt{p_t}}$
return $\mathbf{M}$.

Theorem 3.1 (Online Rank k Projection-Cost Preservation)**.** *Given parameters $\varepsilon > 0$, $k > 0$, and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that outputs a matrix $\mathbf{M}$ with $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n \log^2\kappa\right)$ (rescaled) rows of $\mathbf{A}$ such that*

$$(1-\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2 \le \left\|\mathbf{M}-\mathbf{M}_{(k)}\right\|_F^2 \le (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2,$$

and thus $\mathbf{M}$ is a rank k projection-cost preservation of $\mathbf{A}$, with high probability.

Proof. Consider Algorithm 5. For all $i \in [n]$, let $\mathbf{M}_i$ denote the matrix $\mathbf{M}$ at time i, which is used to compute λ_{i+1}. Let $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$. [CMP16] show that sampling row $\mathbf{a}_i$ with probability at least $\min(1, \alpha\mathbf{a}_i(\mathbf{M}_{i-1}^\top\mathbf{M}_{i-1} + \lambda\mathbb{I})^{-1}\mathbf{a}_i^\top)$ suffices to give a matrix $\mathbf{M}$ such that $(1-\varepsilon)(\mathbf{A}^\top\mathbf{A} + \lambda\mathbb{I}) \preceq \mathbf{M}^\top\mathbf{M} + \lambda\mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}^\top\mathbf{A} + \lambda\mathbb{I})$ with high probability. This argument also follows from the matrix martingale analysis in Lemma 2.5, which is based on their argument. Since we sample each row $\mathbf{a}_i$ with probability $\min(1, 2\alpha\mathbf{a}_i(\mathbf{M}_{i-1}^\top\mathbf{M}_{i-1} + \widetilde{\lambda}_i\mathbb{I})^{-1}\mathbf{a}_i^\top)$ instead, it suffices to show that $\widetilde{\lambda}_i \le 2\lambda$. But this must hold since by a union bound, $(1-\varepsilon)(\mathbf{A}_{i-1}^\top\mathbf{A}_{i-1} + \lambda_i\mathbb{I}) \preceq \mathbf{M}_{i-1}^\top\mathbf{M}_{i-1} + \lambda_i\mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}_{i-1}^\top\mathbf{A}_{i-1} + \lambda_i\mathbb{I})$ for all $i \in [n]$. In particular, this implies by Lemma 2.7 and Lemma 2.8 that $\mathbf{M}_i$ is a rank k projection-cost preservation for $\mathbf{A}_i$ and thus,

$$\widetilde{\lambda}_i = \frac{\left\|\mathbf{M}_{i-1} - (\mathbf{M}_{i-1})_{(k)}\right\|_F^2}{k} \le \frac{2\left\|\mathbf{A}_{i-1} - (\mathbf{A}_{i-1})_{(k)}\right\|_F^2}{k} \le \frac{2\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k} = 2\lambda.$$

To bound the number of rows sampled by $\mathbf{M}$, we use a similar approach to that in Theorem 2.6. Let $u_0 = 1$ and for each $i \ge 1$, let u_i be the smallest index j such that $\lambda_j > 2\lambda_{u_{i-1}}$, which partitions the stream into breakpoints u_i starting from the beginning of the stream. For each $u_{i+1} < t \le u_i$, we have $\lambda_{u_i} \le \lambda_t < 2\lambda_{u_i}$. Thus by Lemma 2.11, the sum of the online ridge leverage scores for the rows $\mathbf{a}_t$ with $u_i \le t < u_{i+1}$ is $\mathcal{O}(k\log\kappa)$. Algorithm 5 scales the online ridge leverage score by a factor of $\alpha = \mathcal{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ to determine the sampling probability, so by a coupling argument and standard Chernoff bounds, the number of sampled rows $\mathbf{a}_t$ in $\mathbf{M}$ with $u_i \le t < u_{i+1}$ is $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n\log\kappa\right)$. Since there are $\mathcal{O}(\log\kappa)$ such breakpoints u_i, the total number of rows in $\mathbf{M}$ is $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n\log^2\kappa\right)$.

Note that here we partition the stream from the beginning since we use online ridge leverage scores, whereas the argument in Theorem 2.6 forms the breakpoints beginning from the end of the stream since it considers *reverse* online ridge leverage scores. □

Our algorithm uses space optimal up to lower order terms for low-rank approximation, but we note that more efficient algorithms exist if the objective is only to approximate $\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$, rather than determining the entire projection matrix. For example, the Theorem 1.3 in [AN13] uses space roughly $\mathcal{O}\left(k^2\right)$ whereas Algorithm 5 uses space roughly $\mathcal{O}\left(kd\right)$ to store each of the $\widetilde{\mathcal{O}}\left(k\right)$ sampled rows.

3.2 Online Row Subset Selection

We next describe how to perform row subset selection in the online model. The algorithm and analysis are both relatively simple, but the algorithm only stores $\mathcal{O}\left(\frac{k}{\varepsilon}\log n\log^2\kappa\right)$ rows. By comparison, the recent online row subset selection algorithm of [BLVZ19] stores $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log n\log^2\kappa\right)$ rows to succeed with high probability. Moreover, our algorithm provides the stronger guarantee of the existence of a subset of k rows that provides a $(1+\varepsilon)$ factor approximation to the best rank k solution.

Our starting point is an offline algorithm by [CMM17] and the paradigm of adaptive sampling [DV06, DRVW06, MRWZ20], which is the procedure of repeatedly sampling rows of $\mathbf{A}$ with probability proportional to their squared distances to the subspace spanned by $\mathbf{Z}$. [CMM17] first obtains a matrix $\mathbf{Z}$ that is a constant factor low-rank approximation to the underlying matrix $\mathbf{A}$ through ridge-leverage score sampling. They observe that a theorem by [DRVW06] shows that adaptive sampling $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ additional rows $\mathbf{S}$ of $\mathbf{A}$ against the rows of $\mathbf{Z}$ suffices for $\mathbf{Z}\cup\mathbf{S}$ to contain a $(1+\varepsilon)$ factor approximation to the online row subset selection problem.

[CMM17] then adapts this approach to the streaming model by maintaining a reservoir of $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ rows, and replacing rows appropriately as new rows arrive and more information about $\mathbf{Z}$ is obtained. Alternatively, we modify the proof of [DRVW06] if $\mathbf{Z}$ is given, adaptive sampling can also be performed on data streams to obtain a different but valid $\mathbf{S}$ by oversampling each row of $\mathbf{A}$ by a $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ factor. Moreover by running a low-rank approximation algorithm in parallel, downsampling can be performed as rows of $\mathbf{Z}$ arrive, so that the above approach can be performed in one stream.

In the online model, we cannot downsample rows of $\mathbf{S}$ once they are selected, since $\mathbf{Z}$ evolves as the stream arrives. Fortunately, [CMM17] shows that the adaptive sampling probabilities can be upper bounded by the λ-ridge leverage scores, where $\lambda=\frac{1}{k}\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. Since the λ-ridge leverage scores are at most the online λ-ridge leverage scores, we can again sample rows proportional to their online λ-ridge leverage scores. It then suffices to again use Lemma 2.11 to bound the number of rows sampled in this manner.

We first show an analog for Theorem 2.1 in [DRVW06] for adaptive sampling through oversampling each row by a $\mathcal{O}\left(\frac{k}{\varepsilon}\right)$ factor. The proof is almost verbatim, but uses a different estimator since we do not perform offline sampling with replacement. As before, we use $\mathbf{A}_{(k)}$ to denote the best rank k approximation to $\mathbf{A}$ in the Frobenius norm.

Theorem 3.2. *Let $\mathbf{A}\in\mathbb{R}^{n\times d}$, $\mathbf{Z}$ be a set of vectors in $\mathbb{R}^d$, k be a non-negative integer, and $\varepsilon>0$ be a given constant. Let $\mathbf{S}$ be a matrix formed by selecting each row i of $\mathbf{A}$ independently with*

probability at least

$$p_i = \min\left(1, \frac{k}{\varepsilon}\frac{\left\|(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\right\|_2^2}{\sum_{j=1}^n \left\|(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_j\right\|_2^2}\right).$$

Then there exists a matrix $\mathbf{T}$ *that contains* k *rows of* $\mathbf{Z}\cup\mathbf{S}$ *such that*

$$\mathbb{E}\left[\left\|\mathbf{A}-\mathbf{A}\mathbf{T}^{-1}\mathbf{T}\right\|_F^2\right] \le \left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2 + \varepsilon\left\|\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z}\right\|_F^2.$$

Proof. Define matrices $\mathbf{U}$ and $\mathbf{V}$ to be the left and right singular vectors of $\mathbf{A}$ so that $\mathbf{A}=\mathbf{U}\Sigma\mathbf{V}$. Let $\mathbf{u}^{(i)}\in\mathbb{R}^{1\times n}$ denote the transpose of the i^{th} column of $\mathbf{U}$ and $\mathbf{v}^{(i)}\in\mathbb{R}^{1\times d}$ denote the i^{th} row of $\mathbf{V}$. Denote the singular values of $\mathbf{A}$ by $\sigma_1\ge\sigma_2\ge\ldots$. We show there exist $\mathbf{t}_1,\ldots,\mathbf{t}_k$ in the span of $\mathbf{Z}\cup\mathbf{T}$ that are a good approximation of the span of top right singular vectors $\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(k)}$.

For $i\in[n]$ and $j\in[k]$, let $\mathbf{x}_i^{(j)}=\frac{\mathbf{u}_i^{(j)}}{p_i}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i$ with probability p_i and the all zeros row vector otherwise so that $\mathbf{x}_i^{(j)}\in\mathbb{R}^{1\times d}$. Let $\mathbf{x}_j=\sum_{i=1}^n\mathbf{x}_i^{(j)}$ so that $\mathbb{E}[\mathbf{x}_j]=\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})$. For each $j\in[k]$, we define each $\mathbf{t}_j\in\mathbb{R}^{1\times d}$ by $\mathbf{t}_j:=\mathbf{u}^{(j)}(\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})+\mathbf{x}_j$ so that $\mathbb{E}[\mathbf{t}_j]=\sigma_j\mathbf{v}^{(j)}$.

We first bound the variance of $\mathbf{t}_j$:

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{t}_j-\mathbb{E}[\mathbf{t}_j]\|_2^2\right] &= \mathbb{E}\left[\left\|\mathbf{t}_j-\sigma_j\mathbf{v}^{(j)}\right\|_2^2\right]\\
&= \mathbb{E}\left[\left\|\mathbf{x}_j-\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})\right\|_2^2\right]\\
&= \mathbb{E}\left[\|\mathbf{x}_j\|_2^2\right]-2\mathbb{E}[\mathbf{x}_j]\cdot\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})+\left\|\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})\right\|_2^2\\
&= \mathbb{E}\left[\|\mathbf{x}_j\|_2^2\right]-\left\|\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})\right\|_2^2.
\end{aligned}$$

To bound the expected squared norm of $\mathbf{x}_j$,

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{x}_j\|_2^2\right] = \mathbb{E}\left[\left\|\sum_{i=1}^n\mathbf{x}_i^{(j)}\right\|_2^2\right] &= \sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_i^{(j)}\right\|_2^2\right]+\sum_{1\le i_1\ne i_2\le n}\mathbb{E}\left[x_{i_1}^{(j)}\cdot x_{i_2}^{(j)}\right]\\
&\le \sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_i^{(j)}\right\|_2^2\right]+\left\|\mathbf{u}^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})\right\|_2^2,
\end{aligned}$$

where the last step holds since $\mathbf{x}_{i_1}^{(j)}$ and $\mathbf{x}_{i_2}^{(j)}$ are independent and $\mathbb{E}[\mathbf{x}_j]=\mathbf{u}_{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})$.

Thus by our choice of p_i,

$$\begin{aligned}
\mathbb{E}\left[\left\|\mathbf{t}_j-\sigma_j\mathbf{v}^{(j)}\right\|_2^2\right] &\le \sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_i^{(j)}\right\|_2^2\right] = \sum_{i=1}^n\frac{\left\|\mathbf{u}_i^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\right\|_2^2}{p_i}\\
&\le \sum_{i=1}^n\frac{\varepsilon}{k}\frac{\left\|\mathbf{u}_i^{(j)}(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\right\|_2^2\left\|\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z}\right\|_F^2}{\|(\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\|_2^2}\\
&= \frac{\varepsilon}{k}\left\|\mathbf{A}-\mathbf{A}\mathbf{Z}^{-1}\mathbf{Z}\right\|_F^2
\end{aligned}$$

Define the matrix $\mathbf{M} = \mathbf{A}\sum_{j=1}^{k} \frac{1}{\sigma_j}(\mathbf{v}^{(j)})^\top \mathbf{t}_j$, so that the row space of $\mathbf{M}$ is spanned by the vectors $\mathbf{t}_1, \ldots, \mathbf{t}_k$. Let $\mathbf{T}$ be the matrix whose rows are $\mathbf{t}_1, \ldots, \mathbf{t}_k$, so that $\left\|\mathbf{A} - \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\right\|_F^2 \leq \|\mathbf{A} - \mathbf{M}\|_F^2$. We now bound the expected value of the Frobenius norm of the difference. Since the Frobenius norm is invariant under rotations,

$$\mathbb{E}\left[\left\|\mathbf{A} - \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\right\|_F^2\right] \leq \mathbb{E}\left[\|\mathbf{A} - \mathbf{M}\|_F^2\right] = \sum_{j=1}^{\operatorname{rank}(\mathbf{A})} \mathbb{E}\left[\|\mathbf{u}_j(\mathbf{A} - \mathbf{M})\|_F^2\right]$$

$$= \sum_{j=1}^{k} \mathbb{E}\left[\left\|\sigma_j \mathbf{v}^{(j)} - \mathbf{t}_j\right\|_F^2\right] + \sum_{j=k+1}^{\operatorname{rank}(\mathbf{A})} \sigma_j^2 \leq \varepsilon \left\|\mathbf{A} - \mathbf{A}\mathbf{Z}^{-1}\mathbf{Z}\right\|_F^2 + \left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2.$$

□

By considering a matrix $\mathbf{Z}$ that is a constant c factor low-rank approximation to $\mathbf{A}$, then we may choose the accuracy parameter in the statement of Theorem 3.2 to be $\frac{\varepsilon}{c}$ so that $\frac{\varepsilon}{c}\left\|\mathbf{A} - \mathbf{A}\mathbf{Z}^{-1}\mathbf{Z}\right\|_F^2 = \varepsilon\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2$. Moreover as [CMM17] notes, the expectation can be converted to a high probability bound through $\log\frac{1}{\delta}$ repetitions and Markov's inequality.

Corollary 3.3. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$, k be a non-negative integer, and $\varepsilon, \delta > 0$ be given constants. Let $\mathbf{Z}$ be a constant factor low-rank approximation to $\mathbf{A}$. Then there exists a constant C such that sampling each row i of $\mathbf{A}$ independently with probability at least*

$$p_i = \frac{Ck\log(1/\delta)}{\varepsilon} \frac{\left\|(\mathbf{A} - \mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\right\|_2^2}{\sum_{j=1}^{n}\left\|(\mathbf{A} - \mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_j\right\|_2^2}$$

forms a matrix $\mathbf{S}$ such that $\mathbf{Z} \cup \mathbf{S}$ contains a matrix $\mathbf{T}$ of k rows such that

$$\mathbf{Pr}\left[\left\|\mathbf{A} - \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2\right] \geq 1 - \delta.$$

Before describing our algorithm, we need the following relationship between ridge leverage scores and adaptive sampling, shown by [CMM17].

Lemma 3.4. *[CMM17] Let $\mathbf{Z}$ be a constant factor approximation to the best rank k approximation to $\mathbf{A}$. There exists a universal constant γ such that if $\tau_i(\mathbf{A})$ is the i^{th} ridge leverage score of a matrix $\mathbf{A}$, with regularization $\frac{\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}$, then $\frac{\gamma\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}\tau_i(\mathbf{A}) \geq \left\|(\mathbf{A} - \mathbf{A}\mathbf{Z}^{-1}\mathbf{Z})_i\right\|_2^2$.*

We now show the correctness and space bounds for OnlineRSS.

Theorem 3.5 (Online Row Subset Selection)**.** *Given parameters $0 < \varepsilon < \frac{1}{2}$, $k > 0$, and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that outputs a matrix $\mathbf{M}$ with $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log^2\kappa\right)$ rows that contains a matrix $\mathbf{T}$ of k rows such that*

$$\left\|\mathbf{A} - \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\right\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2,$$

with high probability.

Algorithm 6 OnlineRSS: Online algorithm for row subset selection

Input: Stream of rows $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{1\times d}$, accuracy $\varepsilon \leq \frac{1}{2}$, and parameter k.
Output: $(1+\varepsilon)$-approximate row subset selection of $\mathbf{A} := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$.

Run a 2-approximation OnlinePCP for online low-rank approximation of $\mathbf{A}$ in parallel. ▷*Algorithm 5*
$\alpha \leftarrow \frac{C}{\varepsilon} \log n$ for sufficiently large $C > 0$
$\mathbf{S} \leftarrow \emptyset$
for each row $\mathbf{a}_t$ **do**
    Update OnlinePCP with $\mathbf{a}_t$.
    Let $\mathbf{Z}_t$ be the rows stored by OnlinePCP.
    $\widetilde{\lambda}_t \leftarrow \frac{\|\mathbf{Z}_t - (\mathbf{Z}_t)_{(k)}\|_F^2}{2k}$
    $p_t \leftarrow \min(1, 2\alpha \cdot \mathbf{a}_t(\mathbf{Z}_t^\top \mathbf{Z}_t + \widetilde{\lambda}_t \mathbb{I})^{-1}\mathbf{a}_t^\top)$
    With probability p_t, $\mathbf{S} \leftarrow \mathbf{S} \circ \mathbf{a}_t$ ▷*Independently sampled*
Let $\mathbf{Z}$ be the output of OnlinePCP.
return $\mathbf{Z} \cup \mathbf{S}$.

Proof. Consider OnlineRSS (Algorithm 6). We set $\mathbf{M}$ as the output $\mathbf{Z} \cup \mathbf{S}$ from OnlineRSS. Let τ_i denote the ridge leverage score of $\mathbf{a}_i$, with regularization $\frac{\|\mathbf{A}-\mathbf{A}_{(k)}\|_F^2}{k}$. By choosing OnlinePCP to be a 2-approximation to the best low-rank approximation to $\mathbf{A}$, we have that $\mathbf{Z}$ contains a rank k matrix $\mathbf{W} \in \mathbb{R}^{k\times d}$ such that $\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}\mathbf{W}^{-1}\mathbf{W}\|_F^2 \leq 2\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2$. We condition on OnlinePCP to return such a matrix $\mathbf{Z}$ with high probability.

By Lemma 3.4, it follows that

$$\frac{\alpha k \log n}{\varepsilon} \frac{\|(\mathbf{A} - \mathbf{A}\mathbf{W}^{-1}\mathbf{W})_i\|_2^2}{\|\mathbf{A} - \mathbf{A}\mathbf{W}^{-1}\mathbf{W}\|_F^2} \leq \frac{\alpha\gamma \log n}{\varepsilon} \tau_i(\mathbf{A}).$$

Observe that each row $\mathbf{a}_i$ of $\mathbf{A}$ is sampled with probability p_i, which is a 2-approximation to the online λ-ridge leverage score. Online ridge leverage scores are at least as large as ridge leverage scores, so $p_i \geq \frac{\alpha}{\varepsilon}\tau_i$. Thus for sufficiently large α, $p_i \geq \frac{\alpha k \log n}{\varepsilon} \frac{\|(\mathbf{A}-\mathbf{A}\mathbf{W}^{-1}\mathbf{W})_i\|_2^2}{\|\mathbf{A}-\mathbf{A}\mathbf{W}^{-1}\mathbf{W}\|_F^2}$. Moreover, the sampling probabilities p_i to determine whether a row should be added to $\mathbf{S}$ depend only on OnlinePCP and not on whether previous rows were added to $\mathbf{S}$. Hence, the sampling probabilities are independent and by Corollary 3.3, it follows that $\mathbf{W} \cup \mathbf{S}$ contains a matrix $\mathbf{T}$ of k rows such that $\|\mathbf{A} - \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\|_F^2 \leq (1+\varepsilon)\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2$, with high probability. Since $\mathbf{W}$ is a subset of $\mathbf{Z}$, then $\mathbf{T}$ is also contained in $\mathbf{Z} \cup \mathbf{S}$.

It remains to bound the number of rows in $\mathbf{Z} \cup \mathbf{S}$. Since $\mathbf{Z}$ uses OnlinePCP with a constant factor approximation, it stores $\mathcal{O}\left(k \log n \log^2 \kappa\right)$ rows by Theorem 3.1. To bound the number of rows sampled by $\mathbf{S}$, we use the same idea from Theorem 3.1. Define $\lambda_i = \frac{\|\mathbf{A}_i - (\mathbf{A}_i)_{(k)}\|_F^2}{k}$, where $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$, for each $i \in [n]$. We set $u_0 = 1$ and for each $i \geq 1$, let u_i be the smallest index j such that $\lambda_j > 2\lambda_{u_{i-1}}$, which partitions the stream into breakpoints u_i starting from the beginning of the stream. For each $u_{i+1} < t \leq u_i$, we have $\lambda_{u_i} \leq \lambda_t < 2\lambda_{u_i}$ so the sum of the online ridge leverage scores for the rows $\mathbf{a}_t$ with $u_i \leq t < u_{i+1}$ is $\mathcal{O}(k \log \kappa)$ by Lemma 2.11. OnlineRSS scales the online ridge leverage score by a factor of $\alpha = \mathcal{O}\left(\frac{1}{\varepsilon}\log n\right)$ to determine the sampling probability into $\mathbf{S}$, so by a coupling argument and standard Chernoff bounds, the number of sampled rows

$\mathbf{a}_t$ in $\mathbf{S}$ with $u_i \leq t < u_{i+1}$ is $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log \kappa\right)$. Since there are $\mathcal{O}(\log \kappa)$ such breakpoints u_i, the total number of rows in $\mathbf{S}$ is $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log^2 \kappa\right)$. Hence, the total number of rows in $\mathbf{Z} \cup \mathbf{S}$ is $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log^2 \kappa\right)$ with high probability. □

3.3 Online Principal Component Analysis

Recall that in the online PCA problem, rows of the matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$ arrive sequentially in a data stream and after each row $\mathbf{a}_i$ arrives, the goal is to immediately output a row $\mathbf{m}_i \in \mathbb{R}^{1\times m}$ such that at the end of the stream, there exists a low-rank matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ such that

$$\|\mathbf{A} - \mathbf{M}\mathbf{X}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2,$$

where $\mathbf{M} = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_n$ and $\mathbf{A}_{(k)}$ is again the best rank k approximation to $\mathbf{A}$.

[BLVZ19] gives an algorithm for the online PCA problem that embeds into a matrix $\mathbf{X}$ that has rank $m = \mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log \kappa)^4\right)$ with high probability, where κ is the condition number of $\mathbf{A}$. Their algorithm maintains and updates the matrix $\mathbf{X}$ throughout the data stream. After the arrival of row $\mathbf{a}_i$, the matrix $\mathbf{X}$ is updated using a combination of residual based sampling and a black-box theorem of [BGKL15]. Row $\mathbf{m}_i$ is then output as the embedding of $\mathbf{a}_i$ into $\mathbf{X}$ by $\mathbf{m}_i = \mathbf{a}_i\mathbf{X}^{(i)}$, where $\mathbf{X}^{(i)}$ is the matrix $\mathbf{X}$ after row i has been processed by $\mathbf{X}$. $\mathbf{X}$ has the property that no rows from $\mathbf{X}$ are ever removed across the duration of the algorithm, so then $\mathbf{m}_i$ is only an upper bound on the best embedding of $\mathbf{a}_i$. However, this matrix $\mathbf{X}$ does that contain a good rank k approximation to $\mathbf{A}$. That is, $\mathbf{X}$ does not contain a rank k submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ such that there exists a matrix $\mathbf{B}$ such that

$$\|\mathbf{A} - \mathbf{B}\mathbf{Y}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2.$$

Recall that our online row subset selection algorithm OnlineRSS (Algorithm 6) returns a matrix $\mathbf{Z} \in \mathbb{R}$ with $\mathcal{O}\left(\frac{k}{\varepsilon}\log n \log^2 \kappa\right)$ rows of $\mathbf{A}$ that contains a submatrix $\mathbf{Y}$ of k rows that is a good rank k approximation of $\mathbf{A}$. Thus, our algorithm for online row subset selection algorithm can be combined with the algorithm of [BLVZ19] to output matrices $\mathbf{M}$ and $\mathbf{W}$ that is a good approximation to the online PCA problem but also so that $\mathbf{W}$ contains a submatrix $\mathbf{Y}$ of k rows that is a good rank k approximation of $\mathbf{A}$. Namely, the algorithm of [BLVZ19] can be run to produce a matrix $\mathbf{X}^{(i)}$ after the arrival of each row $\mathbf{a}_i$. Moreover, our online row subset selection algorithm Algorithm 6 can be run to produce a matrix $\mathbf{Z}^{(i)}$ after the arrival of each row $\mathbf{a}_i$. Let $\mathbf{W}^{(0)} = \mathbb{R}^{0\times d}$ and let $\mathbf{W}^{(i)}$ append the newly added rows of $\mathbf{X}^{(i)}$ and $\mathbf{Z}^{(i)}$ to $\mathbf{W}^{(i-1)}$. We then immediately output the embedding $\mathbf{m}_i = \mathbf{a}_i\mathbf{W}^{(i)}$.

Theorem 3.6 (Online Principal Component Analysis)**.** *Given parameters $n, d, k, \varepsilon > 0$ and a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ whose rows arrive sequentially in a stream with condition number κ, let $m = \mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n + \log \kappa)^4\right)$. There exists an algorithm for online PCA that immediately outputs a row $\mathbf{m}_i \in \mathbb{R}^m$ after seeing row $\mathbf{a}_i \in \mathbb{R}^d$ and outputs a matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ at the end of the stream such that*

$$\|\mathbf{A} - \mathbf{M}\mathbf{X}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2,$$

where $\mathbf{A}_{(k)}$ is the best rank k approximation to $\mathbf{A}$. Moreover, $\mathbf{X}$ contains a submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ such that there exists a matrix $\mathbf{B}$ such that

$$\|\mathbf{A} - \mathbf{B}\mathbf{Y}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A} - \mathbf{A}_{(k)}\right\|_F^2.$$

Proof. Since $\mathbf{W}$ contains the matrix $\mathbf{X}$ from the algorithm of [BLVZ19], it immediately follows that $\|\mathbf{A}-\mathbf{MW}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. Moreover, since $\mathbf{X}$ contains the matrix $\mathbf{Z}$ from our online row subset selection algorithm, which also never removes any rows of $\mathbf{Z}$ once they are added, then $\mathbf{X}$ contains a rank k submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ such that there exists a matrix $\mathbf{B}$ such that $\|\mathbf{A}-\mathbf{BY}\|_F^2 \leq (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. Finally, we note that since $\mathbf{X}$ has $\mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n+\log\kappa)^4\right)$ rows and $\mathbf{Z}$ has $\mathcal{O}\left(\frac{k}{\varepsilon}\log n\log^2\kappa\right)$ rows, then $\mathbf{W}$ has $\mathcal{O}\left(\frac{k}{\varepsilon^2}(\log n+\log\kappa)^4\right)$ rows. □

4 ℓ_1-Subspace Embeddings

In this section, we consider ℓ_1-subspace embeddings in both the online model and the sliding window model. For ℓ_2-subspace embeddings in Section 2, a central concept used to sample a row $\mathbf{a}_i \in \mathbb{R}^d$ in a matrix $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ is the online leverage score, $\mathbf{a}_i(\mathbf{A}_{i-1}^\top \mathbf{A}_{i-1})^{-1}\mathbf{a}_i^\top$. An equivalent formulation for the (online) leverage score of $\mathbf{a}_i$ is the following:

Definition 4.1 (Maximization Characterization of (Online) Leverage Scores)**.** *For a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$, the leverage score of $\mathbf{a}_i$ can also be written as $\max_{\mathbf{x}\in\mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|^2}{\|\mathbf{Ax}\|_2^2}$. Similarly, the online leverage score of $\mathbf{a}_i$ can also be written as $\max_{\mathbf{x}\in\mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|^2}{\|\mathbf{A}_{i-1}\mathbf{x}\|_2^2}$, where $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$. As by convention in Definition 2.1, the online leverage score is set to 1 when the fraction is undefined.*

Thus we define the analogous quantities for ℓ_1-subspace embeddings that we shall use to govern sampling proabilities for each row:

Definition 4.2 ((Online) ℓ_1 Sensitivity)**.** *For a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$, we define the ℓ_1 sensitivity of $\mathbf{a}_i$ by $\max_{\mathbf{x}\in\mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{Ax}\|_1}$ and the* online ℓ_1 sensitivity *of $\mathbf{a}_i$ by $\max_{\mathbf{x}\in\mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{A}_i\mathbf{x}\|_1}$, where $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$. We again that use the convention that the online ℓ_1 sensitivity of $\mathbf{a}_i$ is 1 if $\operatorname{rank}(\mathbf{A}_i) > \operatorname{rank}(\mathbf{A}_{i-1})$.*

Note that from the definition, the online ℓ_1 sensitivity of a row is at least as large as the ℓ_1 sensitivity of the row. Similarly, the (online) ℓ_1 sensitivity of each of the previous rows in $\mathbf{A}$ cannot increase when a new row $\mathbf{r}$ is added to $\mathbf{A}$. Thus we can use the online ℓ_1 sensitivities to define an online algorithm for ℓ_1-subspace embedding similar to Algorithm 5.

Algorithm 7 OnlineL1: Online algorithm for ℓ_1-subspace embedding

Input: Stream of rows $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{1\times d}$, accuracy $\varepsilon > \frac{1}{n}$, and parameter k.
Output: ℓ_1-subspace embedding of $\mathbf{A} := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$.

```
M ← ∅
α ← (Cd/ε²) log n with sufficiently large constant C > 0
for each row a_t do
    τ_t ← 4 · max_{x∈R^d} |a_t^T x| / ||Mx||_1
    p_t ← min(1, ατ_t)
    With probability p_t, M ← M ∘ a_t/p_t
return M.
```

We require the following concentration inequality on scalar martingales.

Theorem 4.3 (Freedman's inequality)**.** *[Fre75] Let $Y_0, Y_1, \ldots, Y_n$ be a scalar martingale with difference sequence $X_1, \ldots, X_n$. Suppose $|X_t| \leq R$ for all $t \in [n]$ with high probability and let the predictable quadratic variation process of the martingale be defined by $w_k := \sum_{t=1}^{k} \mathop{\mathbb{E}}_{t-1}\left[X_t^2\right]$, for $k \in [n]$. Then for all $\varepsilon \geq 0$ and $\sigma^2 > 0$, and every $k \in [n]$,*

$$\mathbf{Pr}\left[\max_{t\in[k]} |Y_t| > \varepsilon \text{ and } w_k \leq \sigma^2\right] \leq 2\exp\left(-\frac{\varepsilon^2/2}{\sigma^2 + R\varepsilon/3}\right).$$

We first show ℓ_1 sensitivity sampling gives an ℓ_1-subspace embedding.

Lemma 4.4. *For $\varepsilon > \frac{1}{n}$, Algorithm 7 returns a matrix $\mathbf{M}$ such that for all $\mathbf{x} \in \mathbb{R}^d$,*

$$\left|\, \|\mathbf{M}\mathbf{x}\|_1 - \|\mathbf{A}\mathbf{x}\|_1 \,\right| \leq \varepsilon \|\mathbf{A}\mathbf{x}\|_1,$$

with high probability.

Proof. Let $\mathbf{x} \in \mathbb{R}^d$ and suppose without loss of generality that $\|\mathbf{A}\mathbf{x}\|_1 = 1$. Let $Y_0, Y_1, \ldots, Y_n$ be a martingale with difference sequence $X_1, \ldots, X_n$. For $j \geq 1$, we set $X_j = 0$ if $Y_{j-1} \geq \varepsilon$ and otherwise if $Y_{j-1} < \varepsilon$, we define

$$X_j = \begin{cases} \left(\frac{1}{p_j} - 1\right)|\mathbf{a}_j^\top \mathbf{x}| & \text{if } \mathbf{a}_j \text{ is sampled in } \mathbf{M} \\ -|\mathbf{a}_j^\top \mathbf{x}| & \text{otherwise.} \end{cases}$$

Observe that $\mathbb{E}\left[Y_j | Y_1, \ldots, Y_{j-1}\right] = Y_{j-1}$ so the sequence is a valid martingale. Moreover, the difference sequence defines $Y_j = \left|\, \|\mathbf{M}_j\mathbf{x}\|_1 - \|\mathbf{A}_j\mathbf{x}\|_1 \,\right|$, where $\mathbf{M}_j$ is the rows of $\mathbf{M}$ sampled at time j and $\mathbf{A}_j = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_j$.

Thus if $p_j = 1$, then $X_j = 0$ and otherwise, $\mathbb{E}\left[X_j^2\right] \leq \frac{1}{p_j}|\mathbf{a}_j^\top \mathbf{x}|^2 = \frac{1}{\alpha\tau_j}|\mathbf{a}_j^\top \mathbf{x}|^2$. Since $\tau_j \geq |\mathbf{a}_j^\top \mathbf{x}|$ and we scaled $\|\mathbf{A}\mathbf{x}\|_1 = 1$, then $|\mathbf{a}_j^\top \mathbf{x}| \leq 1$ so that $\sum_{j=1}^{n} \mathbb{E}\left[X_j\right]^2 \leq \frac{1}{\alpha}$. By Freedman's inequality (Theorem 4.3) and $\alpha = \mathcal{O}\left(\frac{d}{\varepsilon^2}\log n\right)$, it follows that

$$\mathbf{Pr}\left[|Y_n| > \varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2/2}{\alpha + R\alpha/3}\right) \leq \frac{1}{2^d \operatorname{poly}(n)},$$

for sufficiently large α. Since we scaled $\|\mathbf{A}\mathbf{x}\|_1 = 1$, then $\left|\, \|\mathbf{M}\mathbf{x}\|_1 - \|\mathbf{A}\mathbf{x}\|_1 \,\right| \leq \varepsilon \|\mathbf{A}\mathbf{x}\|_1$ with probability at least $1 - \frac{1}{2^d \operatorname{poly}(n)}$.

Now consider the unit ball $B = \{\mathbf{A}\mathbf{y} \in \mathbb{R}^d \mid \|\mathbf{A}\mathbf{y}\|_1 = 1\}$ and let $\mathcal{N}$ be an ε-net of B, constructed greedily. Observe that $\mathcal{N}$ has at most $\left(\frac{3}{\varepsilon}\right)^d$ points, since balls of radius $\frac{\varepsilon}{2}$ around each point cannot overlap, but must all fit into a ball of radius $1 + \frac{\varepsilon}{2}$. Thus by a union bound for $\frac{1}{\varepsilon} < n$, we have $\left|\, \|\mathbf{M}\mathbf{y}\|_1 - \|\mathbf{A}\mathbf{y}\|_1 \,\right| \leq \varepsilon \|\mathbf{A}\mathbf{y}\|_1$ for all $\mathbf{A}\mathbf{y} \in \mathcal{N}$, with probability at least $1 - \frac{1}{\operatorname{poly}(n)}$.

For any vector $\mathbf{z} \in \mathbb{R}^d$ such that $\|\mathbf{A}\mathbf{z}\|_1 = 1$, we construct a sequence $\mathbf{A}\mathbf{y}_1, \mathbf{A}\mathbf{y}_2, \ldots$ such that $\left\|\mathbf{A}\mathbf{z} - \sum_{j=1}^{i} \mathbf{A}\mathbf{y}_j\right\|_1 \leq \varepsilon^i$ and there exists a constant $\gamma_i \leq \varepsilon^{i-1}$ such that $\frac{1}{\gamma_i}\mathbf{A}\mathbf{y}_i \in \mathcal{N}$ for all i. Let $\mathbf{A}\mathbf{y}_1$ be the point in $\mathcal{N}$ closest to $\mathbf{A}\mathbf{z}$ so that $\|\mathbf{A}\mathbf{z} - \mathbf{A}\mathbf{y}_1\|_1 \leq \varepsilon$. Given a sequence $\mathbf{A}\mathbf{y}_1, \ldots, \mathbf{A}\mathbf{y}_{i-1}$ such that $\gamma_i := \left\|\mathbf{A}\mathbf{z} - \sum_{j=1}^{i-1} \mathbf{A}\mathbf{y}_j\right\|_1 \leq \varepsilon^{i-1}$, note that $\frac{1}{\gamma_i}\left\|\mathbf{A}\mathbf{z} - \sum_{j=1}^{i-1} \mathbf{A}\mathbf{y}_j\right\|_1 = 1$, so there exists a

point $\mathbf{A}\mathbf{y}_i \in \mathcal{N}$ that is within distance ε of $\mathbf{A}\mathbf{z} - \sum_{j=1}^{i-1} \mathbf{A}\mathbf{y}_j$, which completes the induction. Hence we have

$$|\,\|\mathbf{M}\mathbf{z}\|_1 - \|\mathbf{A}\mathbf{z}\|_1\,| \leq \sum_{i=1}^{\infty} |\,\|\mathbf{M}\mathbf{y}_i\|_1 - \|\mathbf{A}\mathbf{y}_i\|_1\,| \leq \sum_{i=1}^{\infty} \varepsilon^i \|\mathbf{A}\mathbf{y}_i\|_1 = \mathcal{O}(\varepsilon)\|\mathbf{A}\mathbf{z}\|_1 .$$

The claim follows from a rescaling of ε. ☐

To analyze the space complexity, it remains to bound the sum of the online ℓ_1 sensitivities. Much like the proof of Theorem 2.6, we first bound the sum of regularized online ℓ_1 sensitivities and then argue relate these quantities. We then require a few structural results relating online leverage scores and their relationships to regularized online ℓ_1 sensitivities.

Lemma 4.5 (Online whack-a-mole)**.** *[CLM+15] Let $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$. Let $\lambda > 0$, $\mathbf{B}_0 = \lambda \mathbb{I}_d$, $\mathbf{B} = \underbrace{\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_0}_{n \text{ times}}$ and T be the sum of the online leverage scores of $\mathbf{B} \circ \mathbf{A}$. For any $\gamma \geq 1$, there exists a diagonal matrix $\mathbf{W}$ with entries in $[0,1]$ that contains at most $\frac{n}{\gamma}$ entries strictly less than 1 so that the online leverage score of any row of $\mathbf{WA}$ with respect to $\mathbf{C} := \mathbf{B} \circ (\mathbf{WA})$ is at most $\frac{\gamma T}{n}$. That is, if $\tau_i^{\mathsf{OL}}(\mathbf{C})$ denotes the leverage score of the i^{th} row of $\mathbf{WA}$ with respect to $\mathbf{C}$, then $\tau_i^{\mathsf{OL}}(\mathbf{C}) \leq \frac{\gamma T}{n}$ for all $i \in [n]$.*

It should be noted that [CLM+15] prove a slightly different version of Lemma 4.5 so that the reweighted input matrix $\mathbf{WA}$ has all of its leverage scores uniformly bounded. However, given the regularization matrix $\mathbf{B}$, the same proof demonstrates that the reweighted input matrix $\mathbf{WA}$ has all of its online leverage scores uniformly bounded. For completeness, we provide the modified proof of [CLM+15] in Appendix B.1.

We now show that if the rows of a matrix has uniformly bounded leverage scores, then the ℓ_1 sensitivities must also be uniformly bounded.

Lemma 4.6 (Uniform leverage scores imply uniform ℓ_1 sensitivities)**.** *Given a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$, let $\tau_i(\mathbf{A})$ and $\zeta_i(\mathbf{A})$ denote the leverage score and ℓ_1 sensitivity of $\mathbf{a}_i$, respectively, for each $i \in [n]$. Let $\gamma > 1$ and suppose that $\zeta_i(\mathbf{A}) \leq \frac{\gamma d}{n}$ for all $i \in [n]$. Then $\tau_i(\mathbf{A}) \leq \frac{\gamma d}{n}$ for all $i \in [n]$.*

Proof. For any given i, let $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{A}\mathbf{x}\|_1}$. If $\frac{\max_j |[\mathbf{A}\mathbf{x}^*]_j|}{\|\mathbf{A}\mathbf{x}^*\|_2} > \sqrt{\frac{\gamma d}{n}}$, where $[\mathbf{A}\mathbf{x}^*]_j$ denotes coordinate j of vector $\mathbf{A}\mathbf{x}^*$, then row $\mathbf{a}_j$ would have leverage score $\tau_j(\mathbf{A}) > \frac{\gamma d}{n}$, which violates the assumption of the claim. Thus, $\frac{\max_j |\mathbf{A}\mathbf{x}^*|_j}{\|\mathbf{A}\mathbf{x}^*\|_2} \leq \sqrt{\frac{\gamma d}{n}}$.

Now for any vector $\mathbf{z} \in \mathbb{R}^n$, we have $\frac{\|\mathbf{z}\|_2}{\max_j |\mathbf{z}|_j} \cdot \|\mathbf{z}\|_2 \leq \|\mathbf{z}\|_1$. Therefore,

$$\zeta_i(\mathbf{A}) = \frac{|\mathbf{a}_i^\top \mathbf{x}^*|}{\|\mathbf{A}\mathbf{x}^*\|_1} \leq |\mathbf{a}_i^\top \mathbf{x}^*| \frac{\max_j |[\mathbf{A}\mathbf{x}^*]_j|}{\|\mathbf{A}\mathbf{x}^*\|_2^2} = \max \frac{\max_j |\mathbf{A}\mathbf{x}^*|_j}{\|\mathbf{A}\mathbf{x}^*\|_2} \cdot \sqrt{\frac{\gamma d}{n}} \leq \frac{\gamma d}{n}.$$

☐

We now bound the sum of the online ℓ_1 sensitivities of a matrix.

Lemma 4.7 (Bound on Sum of Online ℓ_1 Sensitivities)**.** *Let $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$. Let $\zeta_i^{\mathsf{OL}}(\mathbf{A})$ be the online ℓ_1 sensitivity of $\mathbf{a}_i$ with respect to $\mathbf{A}$, for each $i \in [n]$. Then $\sum_{i=1}^n \zeta_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d \log n \log \kappa)$.*

Proof. Let $\lambda > 0$, $\mathbf{B}_0 = \lambda \mathbb{I}_d$, $\mathbf{B} = \underbrace{\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_0}_{n \text{ times}}$, and T be an upper bound on the sum of the online leverage scores of any submatrix of $\mathbf{X} := \mathbf{B} \circ \mathbf{A}$. For each $i \in [n]$, let $\zeta_i(\mathbf{X})$ and $\tau_i(\mathbf{X})$ be the ℓ_1 sensitivity and leverage score of row $\mathbf{a}_i$ with respect to $\mathbf{X}$, respectively. Similarly, let $\zeta_i^{\mathsf{OL}}(\mathbf{X})$ denote the online ℓ_1 sensitivity of row $\mathbf{a}_i$ with respect to $\mathbf{X}$. By Lemma 2.2 and Lemma 4.5, there exist a set $S \subseteq [n]$ of size at most d and a diagonal matrix $\mathbf{W}$ with $\frac{n}{2}$ non-unit entries such that $\tau_i^{\mathsf{OL}}(\mathbf{B} \circ (\mathbf{WA})) \leq \frac{2T}{n}$ for all $i \in [n]$, where T is the sum of the online ridge leverage scores of $\mathbf{A}$ with regularization $\lambda > 0$. Let $\mathbf{C} = \mathbf{B} \circ (\mathbf{WA})$ and S be the set of all indices i such that $\mathbf{W}_{i,i} = 1$. For each $i \in [n]$, let $\mathbf{C}_{i+nd}$ be the matrix formed by the first $i + nd$ rows of $\mathbf{C}$. Since the online leverage score of each row $\mathbf{a}_j$ with $j \leq i$ in $\mathbf{C}$ is at most $\frac{2T}{n}$ and the leverage scores of the first nd rows of $\mathbf{C}_{i+nd}$ is at most $\frac{1}{n}$, then all rows of $\mathbf{C}_{i+nd}$ have leverage score at most $\frac{2T}{n}$. By Lemma 4.6, all rows of $\mathbf{C}_{i+nd}$ have ℓ_1 sensitivity at most $\frac{2T}{n}$.

Now for any $i \in S$, we have from the definition of online ℓ_1 sensitivity that

$$\zeta_i^{\mathsf{OL}}(\mathbf{X}) \leq \zeta_i^{\mathsf{OL}}(\mathbf{C}) = \zeta_i^{\mathsf{OL}}(\mathbf{C}_{i+nd}) = \zeta_i(\mathbf{C}_{i+nd}) \leq \frac{2T}{n}$$

so that $\sum_{i \in S} \zeta_i^{\mathsf{OL}}(\mathbf{X}) \leq 2T$. Let $U = \{1, \ldots, n\} \setminus S$ and let $\mathbf{X}_U$ be the matrix $\mathbf{X}$ restricted to the rows whose indices are in U. By monotonicity of sensitivities, $\zeta_i^{\mathsf{OL}}(\mathbf{X}) \leq \zeta_i^{\mathsf{OL}}(\mathbf{X}_U)$. By induction,

$$\sum_{i \in U} \zeta_i^{\mathsf{OL}}(\mathbf{X}) \leq \sum_{i \in U} \zeta_i^{\mathsf{OL}}(\mathbf{X}_U) \leq 2T \log n$$

Thus we have $\sum_i \zeta_i^{\mathsf{OL}}(\mathbf{X}) = \mathcal{O}\left(T \log n\right)$, which is $\mathcal{O}\left(d \log n \log \kappa\right)$ by Lemma 2.2. Now suppose we set $\lambda = \frac{\gamma}{\sqrt{d}n}$, where γ is the minimum of the nonzero singular values across all submatrices of $\mathbf{A}$. We also have for any $\mathbf{x} \in \mathbb{R}^d$ that

$$\|\mathbf{x}\|_1 \leq \sqrt{d} \|\mathbf{x}\|_2 \leq \frac{\sqrt{d}}{\sigma_{\min}(\mathbf{A}_i)} \|\mathbf{A}_i \mathbf{x}\|_2 \leq \frac{\sqrt{d}}{\sigma_{\min}(\mathbf{A}_i)} \|\mathbf{A}_i \mathbf{x}\|_1 .$$

Thus for our choice of λ, we have

$$\frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|(\mathbf{B} \circ \mathbf{A}_i)\mathbf{x}\|_1} \geq \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{A}_i \mathbf{x}\|_1 + \frac{\lambda \sqrt{d} n}{\sigma_{\min}(\mathbf{A}_i)} \|\mathbf{A}_i \mathbf{x}\|_1} \geq \frac{|\mathbf{a}_i^\top \mathbf{x}|}{2 \|\mathbf{A}_i \mathbf{x}\|_1},$$

so that the online ℓ_1 sensitivity of $\mathbf{a}_i$ with respect to $\mathbf{A}$ is within a constant factor of the online ℓ_1 sensitivity of $\mathbf{a}_i$ with respect to $\mathbf{X} = \mathbf{B} \circ \mathbf{A}$. That is, $\sum_{i=1}^n \zeta_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}\left(d \log n \log \kappa\right)$. ☐

We now give the full guarantees of Algorithm 7.

Theorem 4.8 (Online ℓ_1-Subspace Emebedding)**.** *Given $\varepsilon > 0$ and a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ whose rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ arrive sequentially in a stream with condition number κ, there exists an online algorithm that outputs a matrix $\mathbf{M}$ with $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2 n \log \kappa\right)$ (rescaled) rows of $\mathbf{A}$ such that*

$$(1 - \varepsilon) \|\mathbf{Ax}\|_1 \leq \|\mathbf{Mx}\|_1 \leq (1 + \varepsilon) \|\mathbf{Ax}\|_1 ,$$

for all $\mathbf{x} \in \mathbb{R}^d$ with high probability.

Algorithm 8 SCORE$(\mathbf{r}, \mathbf{A})$ function for ℓ_1-subspace embedding

Input: A row $\mathbf{r} \in \mathbb{R}^d$ and a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$.
Output: Scaled ℓ_1 sensitivity of $\mathbf{r}$ with respect to $\mathbf{A}$.

```
if rank(A) = rank(A ∘ r) then
    return 4d · max_{x∈R^d} |r^T x| / ||Ax||_1
else
    return 1
```

Proof. Note that for $\varepsilon < \frac{1}{n}$, a trivial algorithm is just to sample every row. For $\varepsilon > \frac{1}{n}$, consider Algorithm 7 and recall that the property that $\mathbf{M}$ gives an ℓ_1-subspace embedding follows from Lemma 4.4 with high probability. Moreover by a union bound, $\mathbf{M}_t$ is a ℓ_1-subspace embedding of $\mathbf{A}_t$ with approximation $(1+\varepsilon)$, where $\mathbf{M}_t$ are the rows of $\mathbf{M}$ stored at time t and $\mathbf{A}_t = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_t$. It suffices to consider $\varepsilon < \frac{1}{2}$, for which each row $\mathbf{a}_i$ is thus sampled with probability at most $4\alpha\zeta_i^{\mathsf{OL}}$, where $\zeta_i^{\mathsf{OL}}(\mathbf{A})$ is the online ℓ_1 sensitivity of $\mathbf{a}_i$ with respect to $\mathbf{A}$. By Lemma 4.7, we have $\sum_{i=1}^n \zeta_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d \log n \log \kappa)$. Since we oversample by a factor of $\alpha = \mathcal{O}\left(\frac{d}{\varepsilon^2} \log n\right)$, then the space complexity of the algorithm follows from a coupling argument and standard Chernoff bounds. □

Finally, note that we can use the reverse online ℓ_1 sensitivities in the framework of Algorithm 1 to obtain an ℓ_1-subspace embedding in the sliding window model.

Since we are considering a sliding window algorithm, we consider the reverse online ℓ_1 sensitivities rather than using the online ℓ_1 sensitivities as for the online ℓ_1-subspace embedding algorithm in Algorithm 7. For a matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$, the reverse online ℓ_1 sensitivity of row $\mathbf{a}_i$ is defined in the natural way, by $\max_{\mathbf{x} \in \mathbb{R}^d} \frac{\mathbf{a}_i^\top \mathbf{x}}{\|\mathbf{Z}_{i-1}\mathbf{x}\|_1}$ if $\operatorname{rank}(\mathbf{Z}_{i-1}) = \operatorname{rank}(\mathbf{Z}_i)$ and by 1 otherwise, where $\mathbf{Z}_i = \mathbf{a}_n \circ \ldots \circ \mathbf{a}_i$. Note that Algorithm 1 with the SCORE function in Algorithm 8 evaluates the importance of each row compared to the following rows, so we are approximately sampling by reverse online ℓ_1 sensitivities, as desired. The proof follows along the same lines as Theorem 2.6 and Theorem 2.12, using a martingale argument to show that the approximations for the reverse online ℓ_1 sensitivities induce sufficiently high sampling probabilities, while still using the sum of the online ℓ_1 sensitivities to bound the total number of sampled rows. We sketch the proof below, as Section 5 presents an improved algorithm for ℓ_1-subspace embedding in the sliding window model that is nearly space optimal, up to lower order factors.

Theorem 4.9 (Randomized ℓ_1-Subspace Embedding Sliding Window Algorithm)**.** *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ be a stream of rows and κ be the condition number of the matrix $\mathbf{r}_1 \circ \ldots \circ \mathbf{r}_n$. Let $W > 0$ be a window size parameter and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. Given a parameter $\varepsilon > 0$, there exists an algorithm that outputs a matrix $\mathbf{M}$ with a subset of (rescaled) rows of $\mathbf{A}$ such that $(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_1 \leq \|\mathbf{M}\mathbf{x}\|_1 \leq (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_1$ for all $\mathbf{x} \in \mathbb{R}^d$ and stores $\mathcal{O}\left(\frac{d^2}{\varepsilon^2} \log^2 n \log \kappa\right)$ rows at any time, with high probability.*

Proof. Consider Algorithm 1 using the SCORE function of Algorithm 8. Since we have already established the martingale argument of the online ℓ_1 sensitivities in Lemma 4.4, the same argument follows for the reverse online ℓ_1 sensitivities. It follows that each row $\mathbf{r}_i$ that increases the rank of $\mathbf{Z}_i$ for $i \geq n - W + 1$ is sampled with probability within a constant factor of $\alpha d \cdot \max_{\mathbf{x} \in \mathbb{R}^d} \frac{|\mathbf{r}_i^\top \mathbf{x}|}{\|\mathbf{Z}_{i-1}\mathbf{x}\|_1}$,

where $\mathbf{Z}_i = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_i$ for each $i \in [n]$. There are at most d rows that increase the rank of $\mathbf{Z}_i$ from $\mathbf{Z}_{i+1}$. Since $\alpha = \mathcal{O}\left(\frac{d}{\varepsilon^2} \log n\right)$ and the sum of the reverse online ℓ_1 sensitivities is bounded by $\mathcal{O}(d \log n \log \kappa)$ by Lemma 4.7, then the space complexity of the algorithm follows from a coupling argument and standard Chernoff bounds. □

Approximating the Sensitivities. We observe that explicitly computing the (online) ℓ_1 sensitivity $\max_{\mathbf{x} \in \mathbb{R}^d} \frac{|\mathbf{a}_i^\top \mathbf{x}|}{\|\mathbf{A}\mathbf{x}\|_1}$ for a row $\mathbf{a}_i$ in $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ may be infeasible. However, it suffices to approximate the ℓ_1 sensitivity to within an additive $\frac{1}{n^2}$ in polynomial time using linear programming. The oversampling parameter α can then be scaled by a factor of two to handle any row with online ℓ_1 sensitivity at least $\frac{2}{n^2}$ and those rows with online ℓ_1 sensitivity less than $\frac{2}{n^2}$ will only incur $\mathcal{O}(\log n)$ additional samples in total, with high probability.

5 A Coreset Framework for Deterministic Sliding Window Algorithms

In this section, we give a framework for deterministic sliding window algorithms based on the merge and reduce paradigm and the concept of online coresets. We define an online coreset for a matrix $\mathbf{A}$ as a weighted subset of rows of $\mathbf{A}$ that also provides a good approximation to prefixes of $\mathbf{A}$:

Definition 5.1 (Online Coreset)**.** *An* online coreset *for a function f, an approximation parameter $\varepsilon > 0$, and a matrix $\mathbf{A} \in \mathbb{R}^{n \times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$ is a subset of weighted rows of $\mathbf{A}$ such that for any $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ with $i \in [n]$, we have $f(\mathbf{M}_i)$ is a $(1+\varepsilon)$-approximation of $f(\mathbf{A}_i)$, where $\mathbf{M}_i$ is the matrix that consists of the weighted rows of $\mathbf{A}$ in the coreset that appear at time i or later.*

We can use deterministic online coresets for deterministic sliding window algorithms using a merge and reduce framework. The idea is to store the $\mathsf{m}_{\mathsf{space}}$ most recent rows in a block $\mathbf{B}_0$, for some parameter $\mathsf{m}_{\mathsf{space}}$ related to the coreset size. Once $\mathbf{B}_0$ is full, we reduce $\mathbf{B}_0$ to a smaller number of rows by setting $\mathbf{B}_1$ to be a $\left(1 + \frac{\varepsilon}{\log n}\right)$ coreset of $\mathbf{B}_0$, *starting with the most recent row*, and then empty $\mathbf{B}_0$ so that it can again store the most recent rows. Subsequently, whenever $\mathbf{B}_0$ is full, we merge successive non-empty blocks $\mathbf{B}_0, \ldots, \mathbf{B}_i$ and reduce them to a $\left(1 + \frac{\varepsilon}{\log n}\right)^i$ coreset, indexed as $\mathbf{B}_{i+1}$. Since the entire stream has length n, then by using $\mathcal{O}(\log n)$ blocks $\mathbf{B}_i$, we will have a $\left(1 + \frac{\varepsilon}{\log n}\right)^{\log n}$ coreset, starting with the most recent row. Rescaling ε, this gives a merge and reduce based framework for $(1+\varepsilon)$ deterministic sliding window algorithms based on online coresets. We give the framework in full in Algorithm 9.

Lemma 5.2. *$\mathbf{B}_i$ in Algorithm 9 is a $\left(1 + \frac{\varepsilon}{\log n}\right)^i$ online coreset for $2^{i-1}\mathsf{m}_{\mathsf{space}}$ rows.*

Proof. Note that $\mathbf{B}_i$ can only be non-empty if at some point $\mathbf{B}_0$ contains $\mathsf{m}_{\mathsf{space}}$ rows and $\mathbf{B}_1, \ldots, \mathbf{B}_{i-1}$ are all non-empty. By induction, $\mathbf{B}_j$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)^j$ online coreset for $2^{j-1}\mathsf{m}_{\mathsf{space}}$ rows for each $1 \leq j < i$. $\mathbf{B}_i$ is then a $\left(1 + \frac{\varepsilon}{\log n}\right)$ online coreset for the rows in $\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_{i-1}$. Thus for $i = 1$, then $\mathbf{B}_i$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)$ online coreset for $\mathsf{m}_{\mathsf{space}}$ rows and for $i > 1$, $\mathbf{B}_i$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)\left(1 + \frac{\varepsilon}{\log n}\right)^{i-1} = \left(1 + \frac{\varepsilon}{\log n}\right)^i$ online coreset for $\mathsf{m}_{\mathsf{space}} + \sum_{j=0}^{i-2} 2^j \mathsf{m}_{\mathsf{space}} = 2^{i-1}\mathsf{m}_{\mathsf{space}}$ rows. □

Algorithm 9 Merge and reduce framework for deterministic sliding window matrix algorithms using online coresets.

Input: A matrix function f that admits an online coreset, a stream of rows $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^{1\times d}$, approximation parameter $\varepsilon > 0$, and a parameter $W > 0$ for the window

Output: An approximation to $f(\mathbf{A})$, where $\mathbf{A} \in \mathbb{R}^{W\times d} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$

Initialize blocks $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_{\log n} \leftarrow \emptyset$
for each row $\mathbf{r}_t$ **do**
  if $\mathbf{B}_0$ does not contain $\mathsf{m}_{\mathsf{space}}$ rows **then** ▷ $\mathsf{m}_{\mathsf{space}}$ *is the coreset size*
    $\mathbf{B}_0 \leftarrow \mathbf{r}_t \circ \mathbf{B}_0$ ▷ *Keep timestamps for all stored rows*
  else
    Let $i > 0$ be the minimal index such that $\mathbf{B}_i = \emptyset$. ▷ *Prepare to merge and reduce*
    $\mathbf{B}_i \leftarrow \textsc{Coreset}\left(\mathbf{M}, \frac{\varepsilon}{\log n}\right)$, where $\mathbf{M} = \mathbf{B}_0 \circ \ldots \circ \mathbf{B}_{i-1}$ ▷ *Online coreset for function f*
    for $j = 0$ to $j = i - 1$ **do**
      $\mathbf{B}_j \leftarrow \emptyset$
    $\mathbf{B}_0 \leftarrow \mathbf{r}_t$
  if there exists a row $\mathbf{r}$ in a block $\mathbf{B}_i$ with timestamp before $t - W + 1$ **then**
    Delete $\mathbf{r}$ from $\mathbf{B}_i$
return $\mathbf{B}_{\log n} \circ \ldots \circ \mathbf{B}_1 \circ \mathbf{B}_0$

Theorem 5.3. *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^{1\times d}$ be a stream of rows, $\varepsilon > 0$, and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. If there exists a deterministic online coreset algorithm for a matrix function f that stores $S(n, d, \varepsilon)$ rows, then there exists a deterministic sliding window algorithm that stores $\mathcal{O}\left(S\left(n, d, \frac{\varepsilon}{\log n}\right) \log n\right)$ rows and outputs a matrix $\mathbf{M}$ such that $f(\mathbf{M})$ is a $(1+\varepsilon)$-approximation of $f(\mathbf{A})$.*

Proof. Let Coreset be a deterministic online coreset algorithm for a function f that stores $S(n, d, \varepsilon)$ rows. Consider Algorithm 9, setting $\mathsf{m}_{\mathsf{space}} = S\left(n, d, \frac{\varepsilon}{\log n}\right)$. Let $t \le \mathsf{m}_{\mathsf{space}}$ and suppose $\mathbf{B}_0$ contains the t rows $\mathbf{r}_{m-t+1}, \ldots, \mathbf{r}_m$ at any time m. We show the stronger property that at any time m, the rows of $\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_i$ can be simultaneously used to provide a $\left(1 + \frac{\varepsilon}{\log n}\right)^i$ approximation to $f(\mathbf{A}_i)$ for all integers $1 \le i \le t + (2^{i-1})\mathsf{m}_{\mathsf{space}}$, where $\mathbf{A}_i = \mathbf{r}_{m-i+1} \circ \ldots \circ \mathbf{r}_m$.

Let $\mathbf{B}_{i_1}, \ldots, \mathbf{B}_{i_x}$ be the nonempty blocks after B_0, with $0 < i_1 < \ldots < i_x$. By Lemma 5.2, $\mathbf{B}_{i_1}$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)^{i_1}$ online coreset for the $2^{i_1-1}\mathsf{m}_{\mathsf{space}}$ rows before time $m - t + 1$. Similarly, $\mathbf{B}_{i_y}$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)^{i_y}$ coreset for the $2^{i_y-1}\mathsf{m}_{\mathsf{space}}$ rows before the rows represented by $\mathbf{B}_{i_{y-1}}$. If $t < \mathsf{m}_{\mathsf{space}}$, then no coresets are merged and thus the hypothesis holds by the definition of online coreset.

Let j be the maximal index such that $i_j = j$. If $t = \mathsf{m}_{\mathsf{space}}$, then $\mathbf{B}_{i_j+1}$ is formed by using Coreset to reduce from the merging of $\mathbf{B}_0 \circ \mathbf{B}_{i_j}$. Thus, $\mathbf{B}_{i_j+1}$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)^{j+1}$ online coreset for the $2^{i_j}\mathsf{m}_{\mathsf{space}}$ rows that appear before the rows represented by $\mathbf{B}_{i_{j+1}}$, so the hypothesis again holds by the definition of online coreset. Since $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$, then Algorithm 9 outputs a matrix $\mathbf{M}$ such that $f(\mathbf{M})$ is a $\left(1 + \frac{\varepsilon}{\log n}\right)^{\log n}$ approximation of $f(\mathbf{A})$. The correctness then follows by rescaling ε.

$\mathbf{B}_0$ stores at most $\mathsf{m}_{\mathsf{space}} = S\left(n, d, \frac{\varepsilon}{\log n}\right)$ rows. The algorithm stores at most $\log n$ blocks and since the error parameter is $\frac{\varepsilon}{\log n}$ to each call of CORESET in each block $\mathbf{B}_i$ with $i > 1$ stores at most $S\left(n, d, \frac{\varepsilon}{\log n}\right)$ rows by the definition of $\mathsf{m}_{\mathsf{space}}$. Hence, the algorithm stores $\mathcal{O}\left(S\left(n, d, \frac{\varepsilon}{\log n}\right) \log n\right)$ rows at any given time. □

The online row sampling algorithm of [CMP16] shows the existence of an online coreset for spectral approximation. Note that if runtime and space are not issues, this coreset can be explicitly computed by computing the online leverage scores, enumeration over sufficiently small subsets of scaled rows, and checking whether a subset is an online coreset for spectral approximation.

Theorem 5.4 (Online Coreset for Spectral Approximation)**.** *[CMP16] For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$, there exists a constant $C > 0$ and a deterministic algorithm $\textsc{Coreset}(\mathbf{A}, \varepsilon)$ that outputs an online coreset of $\frac{Cd}{\varepsilon^2} \log n \log \kappa$ weighted rows of $\mathbf{A}$. For any $i \in [n]$, let $\mathbf{M}_i$ be the weighted rows of $\mathbf{A}$ in the coreset that appear at time i or later. Then $(1-\varepsilon)\|\mathbf{A}_i\mathbf{x}\|_2 \leq \|\mathbf{M}_i\mathbf{x}\|_2 \leq (1+\varepsilon)\|\mathbf{A}_i\mathbf{x}\|_2$ for all $\mathbf{x} \in \mathbb{R}^d$, where $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$.*

Then Theorem 5.3 and Theorem 5.4 imply:

Theorem 5.5 (Deterministic Sliding Window Algorithm for Spectral Approximation)**.** *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^{1\times d}$ be a stream of rows and κ be the condition number of the stream. Let $\varepsilon > 0$ and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. There exists a deterministic algorithm that stores $\mathcal{O}\left(\frac{d}{\varepsilon^2} \log^4 n \log \kappa\right)$ rows and outputs a matrix $\mathbf{M}$ such that $(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{M}\mathbf{x}\|_2 \leq (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_2$ for all $\mathbf{x} \in \mathbb{R}^d$.*

Theorem 3.1 shows that the existence of an online coreset for computing a rank k projection-cost preservation.

Theorem 5.6 (Online Coreset for Rank k Projection-Cost Preservation)**.** *For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$, there exists a constant $C > 0$ and a deterministic algorithm $\textsc{Coreset}(\mathbf{A}, \varepsilon)$ that outputs an online coreset of $\frac{Ck}{\varepsilon^2} \log n \log \kappa$ weighted rows of $\mathbf{A}$. For any $i \in [n]$, let $\mathbf{M}_i$ be the weighted rows of $\mathbf{A}$ in the coreset that appear at time i or later. Then $\mathbf{M}_i$ is a $(1+\varepsilon)$ projection-cost preservation for $\mathbf{A}_i := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$.*

Thus Theorem 5.3 and Theorem 5.6 give:

Theorem 5.7 (Deterministic Sliding Window Algorithm for Rank k Projection-Cost Preservation)**.** *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^{1\times d}$ be a stream of rows and κ be the condition number of the stream. Let $\varepsilon > 0$ and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. There exists a deterministic algorithm that stores $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log^4 n \log \kappa\right)$ rows and outputs a matrix $\mathbf{M}$ such that $(1-\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F \leq \|\mathbf{M} - \mathbf{M}\mathbf{P}\|_F \leq (1+\varepsilon)\|\mathbf{A} - \mathbf{A}\mathbf{P}\|_F$ for all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{d\times d}$.*

For ℓ_1-subspace embeddings, we can use our online coreset from Theorem 4.8, but in fact [CP15] showed the existence of an offline coreset for ℓ_1-subspace embeddings that stores a smaller number of rows. The offline coreset of [CP15] is based on sampling rows proportional to their Lewis weights. We define a corresponding online version of Lewis weights:

Definition 5.8 ((Online) Lewis Weights)**.** *For a matrix* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$*, let* $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$*. Let* $w_i(\mathbf{A})$ *denote the Lewis weight of row* $\mathbf{a}_i$*. Then the Lewis weights of* $\mathbf{A}$ *are the unique weights such that* $w_i(\mathbf{A}) = \left(\mathbf{a}_i(\mathbf{A}^\top \mathbf{W}^{-1}\mathbf{A})^{-1}\mathbf{a}_i^\top\right)^{1/2}$*, where* $\mathbf{W}$ *is a diagonal matrix with* $\mathbf{W}_{i,i} = w_i(\mathbf{A})$*. Equivalently,* $w_i(\mathbf{A}) = \tau_i(\mathbf{W}^{-1/2}\mathbf{A})$*, where* $\tau_i(\mathbf{W}^{-1/2}\mathbf{A})$ *denotes the leverage score of row* i *of* $\mathbf{W}^{-1/2}\mathbf{A}$*. We define the* online Lewis weight *of* $\mathbf{a}_i$ *to be the Lewis weight of row* $\mathbf{a}_i$ *with respect to the matrix* $\mathbf{A}_{i-1}$ *and use the convention that the Lewis weight of* $\mathbf{a}_i$ *is* 1 *if* $\text{rank}(\mathbf{A}_i) > \text{rank}(\mathbf{A}_{i-1})$*.*

Their results hold with high probability and they also show that Lewis weights cannot increase with the addition of new rows, so perhaps it is not surprising that their construction can be easily modified to form an online coreset based on sampling rows proporitional to their *online* Lewis weights.

Lemma 5.9 (Monotonicity of Lewis Weights)**.** *[CP15] For a matrix* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ *and* $i \in [n]$*, let* $w_i(\mathbf{A})$ *denote the Lewis weight of row* $\mathbf{a}_i$ *with respect to* $\mathbf{A}$ *and let* $\tau_i(\mathbf{B})$ *denote the Lewis weight of row* $\mathbf{a}_i$ *with respect to* $\mathbf{B} := \mathbf{A} \circ \mathbf{r}$ *for any row* $\mathbf{r} \in \mathbb{R}^d$*. Then* $\tau_i(\mathbf{A}) \geq \tau_i(\mathbf{B})$*.*

Theorem 5.10 (Online Coreset for ℓ_1-Subspace Embedding)**.** *[CP15] Let* $\mathbf{A} \in \mathbb{R}^{n \times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$*, If there exists an upper bound* C *on the sum of the online Lewis weights of* $\mathbf{A}$*, then there exists a constant* $C > 0$ *and a deterministic algorithm* $\textsc{Coreset}(\mathbf{A}, \varepsilon)$ *that outputs an online coreset of* $\frac{C}{\varepsilon^2}\log n$ *weighted rows of* $\mathbf{A}$*. For any* $i \in [n]$*, let* $\mathbf{M}_i$ *be the weighted rows of* $\mathbf{A}$ *in the coreset that appear at time* i *or later. Then* $\mathbf{M}_i$ *is an* ℓ_1*-subspace embedding for* $\mathbf{A}_i := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ *with approximation* $(1+\varepsilon)$*.*

It thus suffices to analyze the sum of the online Lewis weights. We first require a few structural results on Lewis weights. [CP15] gives an iterative algorithm (Figure 4) that converges toward the Lewis weights.

(1) Given input matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, initialize $\mathbf{W}^{(0)} \leftarrow \mathbb{I}_n$

(2) Repeat: $\mathbf{W}^{(j)}_{i,i} \leftarrow \left(\mathbf{a}_i(\mathbf{A}^\top(\mathbf{W}^{(j-1)})^{-1}\mathbf{A})^{-1}\mathbf{a}_i^\top\right)^{1/2}$ for all $i \in [n]$

Fig. 4: Iterative algorithm for computing Lewis weights.

The algorithm in Figure 4 has the following property, which shows that it is a contraction mapping and thus converges to the Lewis weights due to the Banach fixed point theorem.

Lemma 5.11. *[CP15] If* $\mathbf{W}$ *is diagonal matrix representing the Lewis weights and* $\frac{1}{\gamma}\mathbf{W}^{(j)} \preceq \mathbf{W} \preceq \gamma\mathbf{W}^{(j)}$ *for some* $j > 0$ *and* $\gamma > 1$ *in the algorithm in Figure 4, then* $\frac{1}{\sqrt{\gamma}}\mathbf{W}^{(j+1)} \preceq \mathbf{W} \preceq \sqrt{\gamma}\mathbf{W}^{(j+1)}$*. Thus,* $\mathbf{W}_{(j)}$ *converges to the Lewis weights.*

We now show that if the rows of a matrix has uniformly bounded leverage scores, then the Lewis weights must also be uniformly bounded. Although the statement is similar to Lemma 4.6, the structure of the Lewis weights requires a different strategy of iteratively computing values that converge to the Lewis weights.

Lemma 5.12 (Uniform leverage scores imply uniform Lewis weights)**.** *If the leverage scores of* $\mathbf{A}$ *are at most* $\frac{\gamma d}{n}$ *for some* $\gamma \geq 1$*, then the Lewis weights of* $\mathbf{A}$ *are at most* $\frac{\gamma d}{n}$*.*

Proof. Let $C > 0$ and suppose $\mathbf{a}_i(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{a}_i^\top \leq C$ for all $i \in [n]$. Then for iteration j in the above procedure, we have $\mathbf{W}^{(j)} \preceq C^{1-2^{-j}}\mathbb{I}_n$.

For base case $j = 1$, we have $\mathbf{W}_{i,i}^{(1)} = \left(\mathbf{a}_i(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{a}_i^\top\right)^{1/2} \leq C^{1/2}$ so that $\mathbf{W}^{(1)} \preceq C^{1/2}\mathbb{I}_n$. For iteration j, we have $\mathbf{W}_{i,i}^{(j)} = \left(\mathbf{a}_i(\mathbf{A}^\top(\mathbf{W}^{(j-1)})^{-1}\mathbf{A})^{-1}\mathbf{a}_i^\top\right)^{1/2}$, where $\mathbf{W}^{(j-1)} \preceq C^{1-2^{-(j-1)}}\mathbb{I}_n$ by the inductive hypothesis. Thus $(\mathbf{W}^{(j-1)})^{-1} \succeq \frac{1}{C^{1-2^{-(j-1)}}}\mathbb{I}_n$, so

$$\mathbf{A}^\top(\mathbf{W}^{(j-1)})^{-1}\mathbf{A} \succeq \frac{1}{C^{1-2^{-(j-1)}}}\mathbf{A}^\top\mathbf{A}$$
$$(\mathbf{A}^\top(\mathbf{W}^{(j-1)})^{-1}\mathbf{A})^{-1} \preceq C^{1-2^{-(j-1)}}(\mathbf{A}^\top\mathbf{A})^{-1}$$
$$(\mathbf{W}_{i,i}^{(j)})^2 = \mathbf{a}_i(\mathbf{A}^\top(\mathbf{W}^{(j-1)})^{-1}\mathbf{A})^{-1}\mathbf{a}_i^\top \leq C^{1-2^{-(j-1)}}\mathbf{a}_i(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{a}_i^\top \leq C^{2-2^{-(j-1)}}$$

Thus $\mathbf{W}_{i,i}^{(j)} \leq C^{1-2^{-j}}$ so that $\mathbf{W}^{(j)} \preceq C^{1-2^{-j}}\mathbb{I}_n$. The claim follows from setting $C = \frac{\gamma d}{n}$ and the convergence of the algorithm to the Lewis weights by Lemma 5.11. ☐

We now show that splitting a row of a matrix into two rows that sum up to the original row alters the Lewis weights in the natural way.

Lemma 5.13 (Splitting Invariance of Lewis Weights)**.** *Given* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$*, let*

$$\mathbf{B} \in \mathbb{R}^{(n+1)\times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_{j-1} \circ \mathbf{b}_j \circ \mathbf{a}_{j+1} \circ \ldots \circ \mathbf{a}_n \circ \mathbf{b}_{n+1}$$

have the same rows but with row j *have* $\mathbf{b}_j = (1-\gamma)\cdot\mathbf{a}_j$ *and row* $n+1$ *have* $\mathbf{b}_{n+1} = \gamma\mathbf{a}_j$ *for some* $\gamma \in [0,1]$ *and* $j \in [n]$*. , Then we have* $w_i(\mathbf{A}) = w_i(\mathbf{B})$ *for any* $i \notin \{j, n+1\}$*,* $w_j(\mathbf{B}) = (1-\gamma)w_j(\mathbf{A})$ *and* $w_{n+1}(\mathbf{B}) = \gamma w_j(\mathbf{A})$*.*

Proof. Suppose without loss of generality that $j = n$. Let $\mathbf{W} \in \mathbb{R}^{n\times n}$ be the diagonal Lewis weight scaling matrix with $\mathbf{W}_{i,i} = w_i(\mathbf{A})$. Let $\bar{\mathbf{W}} \in \mathbb{R}^{(n+1)\times(n+1)}$ match $\mathbf{W}$ on the first $n-1$ rows. Let $\bar{\mathbf{W}}_{n,n} = (1-\gamma)w_n(\mathbf{A})$ and $\bar{\mathbf{W}}_{n+1,n+1} = \gamma w_n(\mathbf{A})$. To prove the claim, it suffices by the uniqueness of Lewis weights to show that $\tau_i(\bar{\mathbf{W}}^{-1/2}\mathbf{B}) = \bar{\mathbf{W}}_{i,i}$ for all $i \in [n+1]$:

Note that the first $n-1$ rows of $\bar{\mathbf{W}}^{-1/2}\mathbf{B}$ are the same as those of $\mathbf{W}^{-1/2}\mathbf{A}$. The last two rows are $w_n(\mathbf{A})^{-1/2}(1-\gamma)^{-1/2}\cdot(1-\gamma)\mathbf{a}_n = w_n(\mathbf{A})^{-1/2}(1-\gamma)^{1/2}\mathbf{a}_n$ and $w_n(\mathbf{A})^{-1/2}\gamma^{1/2}\mathbf{a}_n$, respectively. That is, the last two rows are scaled by $(1-\gamma)^{1/2}$ and $\gamma^{1/2}$, respectively, compared to $\mathbf{W}^{-1/2}\mathbf{A}$. Thus we can see that for any vector $\mathbf{y}$, $\left\|\mathbf{W}^{-1/2}\mathbf{A}\mathbf{y}\right\|_2^2 = \left\|\bar{\mathbf{W}}^{-1/2}\mathbf{B}\mathbf{y}\right\|_2^2$. By the maximization characterization of leverage scores, the leverage scores of the first $n-1$ rows of $\mathbf{W}^{-1/2}\mathbf{A}$ are thus identical to those of $\bar{\mathbf{W}}^{-1/2}\mathbf{B}$, so that $\mathbf{W}_{i,i} = \bar{\mathbf{W}}_{i,i}$ for $1 \leq n-1$.

For the last two rows, we have $\tau_n(\bar{\mathbf{W}}^{-1/2}\mathbf{B}) = (1-\gamma)\cdot\tau_n(\mathbf{W}^{-1/2}\mathbf{A}) = (1-\gamma)w_n(\mathbf{A}) = \bar{\mathbf{W}}_{n,n}$. Similarly, $\tau_{n+1}(\bar{\mathbf{W}}^{-1/2}\mathbf{B}) = \gamma\cdot\tau_n(\mathbf{W}^{-1/2}\mathbf{A}) = \gamma w_n(\mathbf{A}) = \bar{\mathbf{W}}_{n+1,n+1}$. Hence, $\tau_i(\bar{\mathbf{W}}^{-1/2}\mathbf{B}) = \bar{\mathbf{W}}_{i,i}$ for all $i \in [n+1]$, which implies the claim by the uniqueness of Lewis weights. ☐

Corollary 5.14 (Monotonicity of Lewis Weights II)**.** *For any* $\mathbf{A} \in \mathbb{R}^{n\times d}$*, let* $\mathbf{B} \in \mathbb{R}^{n\times d}$ *have the same rows but with row* j *reweighted by a factor* $(1-\gamma)$ *for some* $\gamma \in [0,1]$*. Then for all* $i \neq j$*,* $w_i(\mathbf{B}) \geq w_i(\mathbf{A})$*.*

Proof. Let $\bar{\mathbf{B}} \in \mathbf{R}^{(n+1)\times d}$ have row j set to $(1-\gamma)\cdot\mathbf{a}_j$ and row $n+1$ row set to $\gamma\cdot\mathbf{a}_j$. Then by Lemma 5.13, for all $i \neq j$, $w_i(\bar{\mathbf{B}}) = w_i(\mathbf{A})$. By the monotonicity of Lewis weights through the addition of a new row from Lemma 5.9, we thus have $w_i(\mathbf{B}) \geq w_i(\bar{\mathbf{B}}) = w_i(\mathbf{A})$. ☐

We now bound the sum of the online Lewis weights by first considering a regularization of the input matrix, which we show only slightly alters each score.

Lemma 5.15 (Bound on Sum of Online Lewis Weights)**.** *Let* $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$. *Let* $w_i^{\mathsf{OL}}(\mathbf{A})$ *be the online Lewis weight of* $\mathbf{a}_i$ *with respect to* $\mathbf{A}$*, for each* $i \in [n]$. *Then* $\sum_{i=1}^n w_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d \log n \log \kappa)$.

Proof. For the first part of the proof, we use the same argument as Lemma 4.7. Let $\lambda > 0$, $\mathbf{B}_0 = \lambda \mathbb{I}_d$, $\mathbf{B} = \underbrace{\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_0}_{n \text{ times}}$, and T be an upper bound on the sum of the online leverage scores of any submatrix of $\mathbf{X} := \mathbf{B} \circ \mathbf{A}$. For each $i \in [n]$, let $w_i(\mathbf{X})$ and $\tau_i(\mathbf{X})$ be the Lewis weight and leverage score of $\mathbf{a}_i$ with respect to $\mathbf{X}$, respectively. By Lemma 2.2 and Lemma 4.5, there exist a set $S \subseteq [n]$ of size at most d and a diagonal matrix $\mathbf{W}$ with $\frac{n}{2}$ non-unit entries such that $\tau_i^{\mathsf{OL}}(\mathbf{B} \circ (\mathbf{WA})) \leq \frac{2T}{n}$ for all $i \in [n]$, where T is the sum of the online ridge leverage scores of $\mathbf{A}$ with regularization $\lambda > 0$. Let $\mathbf{C} = \mathbf{B} \circ (\mathbf{WA})$ and S be the set of all indices i such that $\mathbf{W}_{i,i} = 1$. For each $i \in [n]$, let $\mathbf{C}_{i+nd}$ be the matrix formed by the first $i + nd$ rows of $\mathbf{C}$. Since the online leverage score of each row $\mathbf{a}_j$ with $j \leq i$ in $\mathbf{C}$ is at most $\frac{2T}{n}$ and the leverage scores of the first nd rows of $\mathbf{C}_{i+nd}$ is at most $\frac{1}{n}$, then all rows of $\mathbf{C}_{i+nd}$ have leverage score at most $\frac{2T}{n}$. By Lemma 5.12, all rows of $\mathbf{C}_{i+nd}$ have Lewis weight at most $\frac{2T}{n}$.

Now for any $i \in S$, we have from the monotonicity of Lewis weights from decreasing the weight of other rows in Corollary 5.14 that

$$w_i^{\mathsf{OL}}(\mathbf{X}) \leq w_i^{\mathsf{OL}}(\mathbf{C}) = w_i^{\mathsf{OL}}(\mathbf{C}_{i+nd}) = w_i(\mathbf{C}_{i+nd}) \leq \frac{2T}{n}$$

so that $\sum_{i \in S} w_i^{\mathsf{OL}}(\mathbf{X}) \leq 2T$. Let $U = \{1, \ldots, n\} \setminus S$ and let $\mathbf{X}_U$ be the matrix $\mathbf{X}$ restricted to the rows whose indices are in U. By monotonicity of Lewis weights from the addition of new rows in Lemma 5.9, $w_i^{\mathsf{OL}}(\mathbf{X}) \leq w_i^{\mathsf{OL}}(\mathbf{X}_U)$. By induction,

$$\sum_{i \in U} w_i^{\mathsf{OL}}(\mathbf{X}) \leq \sum_{i \in U} w_i^{\mathsf{OL}}(\mathbf{X}_U) \leq 2T \log n$$

Thus we have $\sum_i w_i^{\mathsf{OL}}(\mathbf{X}) = \mathcal{O}(T \log n)$, which is $\mathcal{O}(d \log n \log \kappa)$ by Lemma 2.2.

We now assume without loss of generality that all rows of $\mathbf{A}$ have Lewis weight at least $\frac{1}{n}$, since if $\mathbf{a}_n$ has Lewis weight less than $\frac{1}{n}$, then its online Lewis weight is similarly bounded. Thus, these rows contribute at most $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ samples with high probability and removing these rows would only increase the Lewis weights of the other rows by monotonicity Lemma 5.9. Let $\lambda = \frac{\gamma}{n}$, where γ is the minimum of the nonzero singular values across all submatrices of $\mathbf{A}$. Consider approximating the Lewis weights for $\mathbf{A}$ through Figure 4. Since all Lewis weights of $\mathbf{A}$ are at least $\frac{1}{n}$, then by Lemma 5.11, at most $\mathcal{O}(\log n)$ iterations of the algorithm in Figure 4 are necessary to obtain a constant factor approximation to the true Lewis weights. The intuition is that the approximation for the Lewis weight of each row $\mathbf{a}_i$ with respect to $\mathbf{A}$ in iteration j of the algorithm is within $\left(1 + \frac{j}{n}\right)$ of the approximation for the Lewis weight of row $\mathbf{a}_i$ with respect to $\mathbf{X}$. Hence in $\mathcal{O}(\log n)$ iterations, the respective approximations will be within a constant factor approximation.

Formally, let $\mathbf{W}_{\mathbf{A}}^{(j)}$ be the weight matrix on iteration j of the algorithm in Figure 4 on input $\mathbf{A}$ and let $\mathbf{U}_{\mathbf{X}}^{(j)} \circ \mathbf{W}_{\mathbf{X}}^{(j)}$ be the weight matrix on iteration j of the algorithm on input $\mathbf{X}$, so that $\mathbf{U}_{\mathbf{X}}^{(j)}$ has nd rows and $\mathbf{W}_{\mathbf{X}}^{(j)}$ has n rows. Thus, we have $\mathbf{W}_{\mathbf{A}}^{(0)} = \mathbf{W}_{\mathbf{X}}^{(0)} = \mathbb{I}_n$ from the initialization of the

algorithm in Figure 4. The weight for row $\mathbf{a}_i$ is then updated by $\left(\mathbf{a}_i(\mathbf{A}^\top(\mathbf{W}_\mathbf{A}^{(j-1)})^{-1}\mathbf{A})^{-1}\mathbf{a}_i^\top\right)^{1/2}$ for $\mathbf{W}_\mathbf{A}^{(j)}$ and by $\left(\mathbf{a}_i(\mathbf{A}^\top(\mathbf{W}_\mathbf{X}^{(j-1)})^{-1}\mathbf{A}+\mathbf{B}^\top(\mathbf{U}_\mathbf{X}^{(j-1)})^{-1}\mathbf{B})^{-1}\mathbf{a}_i^\top\right)^{1/2}$. Since each Lewis weight is at most 1 and the algorithm is a contraction mapping, then we certainly have $\mathbf{U}_\mathbf{X}^{(j-1)} \preceq \mathbb{I}_{nd}$ so that

$$\mathbf{B}^\top(\mathbf{U}_\mathbf{X}^{(j-1)})^{-1}\mathbf{B} \preceq n\lambda^2\mathbb{I}_{nd} \preceq \frac{\sigma_{\min}(\mathbf{A}^\top\mathbf{A})}{n}\mathbb{I}_{nd} \preceq \frac{\sigma_{\min}(\mathbf{A}^\top(\mathbf{W}_\mathbf{A}^{(j-1)})^{-1}\mathbf{A})}{n}\mathbb{I}_{nd}.$$

Thus $\mathbf{W}_\mathbf{A}^{(1)} \preceq \left(1+\frac{1}{n}\right)\mathbf{W}_\mathbf{X}^{(1)}$ and by induction, $\mathbf{W}_\mathbf{A}^{(j)} \preceq \left(1+\frac{j}{n}\right)\mathbf{W}_\mathbf{X}^{(j)}$.

Hence the Lewis weight of $\mathbf{a}_n$ with respect to $\mathbf{A}$ is within a constant factor of the Lewis weight of $\mathbf{a}_n$ with respect to $\mathbf{X}$, and thus the *online* Lewis weight of $\mathbf{a}_n$ with respect to $\mathbf{A}$ is within a constant factor of the *online* Lewis weight of $\mathbf{a}_n$ with respect to $\mathbf{X}$. Since we have $\sum_i w_i^{\mathsf{OL}}(\mathbf{X}) = \mathcal{O}(d\log n\log\kappa)$, then it follows that $\sum_i w_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d\log n\log\kappa)$. □

Then from Theorem 5.3, Theorem 5.10, and Lemma 5.15, we have the following:

Theorem 5.16 (Deterministic Sliding Window Algorithm for ℓ_1-Subspace Embedding)**.** *Let $\mathbf{r}_1,\ldots,\mathbf{r}_n \in \mathbb{R}^{1\times d}$ be a stream of rows and κ be the condition number of the stream. Let $\varepsilon > 0$ and $\mathbf{A} = \mathbf{r}_{n-W+1}\circ\ldots\circ\mathbf{r}_n$ be the matrix consisting of the W most recent rows. There exists a deterministic algorithm that stores $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log^4 n\log\kappa\right)$ rows and outputs a matrix $\mathbf{M}$ such that $(1-\varepsilon)\|\mathbf{A}\mathbf{x}\|_1 \le \|\mathbf{M}\mathbf{x}\|_1 \le (1+\varepsilon)\|\mathbf{A}\mathbf{x}\|_1$ for all $\mathbf{x}\in\mathbb{R}^d$.*

Note that Theorem 5.3 also provides an approach for a *randomized* ℓ_1-subspace embedding sliding window algorithm that improves upon the space requirements of Theorem 4.9, by using online coresets *randomly* generated sampling rows with respect to their online Lewis weights. Moreover, recall that in some settings, the online model does not require algorithms to use space sublinear in the size of the input. In these settings, Lemma 5.15 could also potentially be useful in a row-sampling based algorithm for online ℓ_1-subspace embedding that improves upon the sample complexity of Theorem 4.8.

Fast Online Coreset Construction. To quickly obtain an online coreset for spectral approximation, we derandomize the OnlineBSS algorithm of [CMP16] with a $\mathcal{O}(\log n)$ overhead. Their algorithm requires an input $\lambda \ge 0$ and the rows of $\mathbf{A} = \mathbf{a}_1\circ\ldots\circ\mathbf{a}_n \in \mathbb{R}^{n\times d}$ in a stream and uses upper barrier and lower barrier method of [BSS12] to design a probability distribution for each row that depends on the previously sampled rows but will *always* output a matrix $\mathbf{M}$ such that $(1-\varepsilon)(\mathbf{A}^\top\mathbf{A}+\lambda\mathbb{I}) \preceq \mathbf{M}^\top\mathbf{M}+\lambda\mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}^\top\mathbf{A}+\lambda\mathbb{I})$. Instead, [CMP16] bounds the expected number of rows sampled by OnlineBSS based on the sampling probabilities of each row.

First, we observe that the sampling probability for each row in OnlineBSS can be rounded up to a power of 2 between $\frac{1}{n}$ and 1. As before, the rows with sampling probability less than $\frac{1}{n}$ incur only $\mathcal{O}(1)$ additional rows sampled in expectation. Although the sampling probability p_i of row $\mathbf{a}_i$ is a random variable that depends on the previous rows sampled, [CMP16] bounds the expectation of p_i by the online (ridge) leverage score τ_i^{OL}, so that the expected number of sampled rows is at most $\sum_{i=1}^n \tau_i$. Now consider a process in which we pick a random threshold $t \in [0,1]$ and then sample row $\mathbf{a}_i$ if $p_i > t$. It can be shown that the expectation of p_i is still at most the online (ridge) leverage score τ_i^{OL}. Since there are only $\log n$ distinct sampling probabilities, we can derandomize this process by trying all $\log n$ values of t. For completeness, we describe OnlineBSS and our derandomization in full detail in Appendix B.2.

References

[ACK+16] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319, 2016. 4, 7

[AM15] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 775–783, 2015. 10, 15

[AN13] Alexandr Andoni and Huy L. Nguyen. Eigenvalues of a matrix in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1729–1737, 2013. 26

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002. 1

[BDV18] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 603–614, 2018. 2

[BGKL15] Christos Boutsidis, Dan Garber, Zohar Shay Karnin, and Edo Liberty. Online principal components analysis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 887–901, 2015. 1, 11, 30

[BGL+18] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. Nearly optimal distinct elements and heavy hitters on sliding windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:22, 2018. 49

[BGO14] Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch ℓ_2-heavy-hitters on sliding windows. *Theoretical Computer Science*, 554:82–94, 2014. 49

[BKRW03] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202–204, 2003. 2

[BLLM15] Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 350–364, 2015. 49

[BLLM16] Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1374–1390, 2016. 49

[BLUZ19] Vladimir Braverman, Harry Lang, Enayat Ullah, and Samson Zhou. Improved algorithms for time decay streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 145, pages 27:1–27:17, 2019. 49

[BLVZ19] Aditya Bhaskara, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Residual based sampling for online low rank approximation. In *60th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 117–126, 2019. (to appear). 3, 5, 6, 11, 26, 30, 31

[BO07] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS) Proceedings*, pages 283–293, 2007. 3, 4, 7, 48, 49, 52

[BSS12] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012. 14, 42, 55

[CEM+15] 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 Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 163–172, 2015. 2, 10, 15, 20

[CG08] Graham Cormode and Minos N. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *EDBT 2008, 11th International Conference on Extending Database Technology, Proceedings*, page 745, 2008. 1

[CLM+15] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the Conference on Innovations in Theoretical Computer Science, ITCS*, pages 181–190, 2015. 9, 10, 33, 54

[CM05] Graham Cormode and S. Muthukrishnan. What's new: finding significant differences in network data streams. *IEEE/ACM Transactions on Networking*, 13(6):1219–1232, 2005. 1

[CMM17] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1758–1777, 2017. 2, 10, 11, 15, 19, 20, 22, 26, 28

[CMP16] Michael B. Cohen, Cameron Musco, and Jakub W. Pachocki. Online row sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:18, 2016. 9, 10, 11, 14, 16, 18, 19, 21, 22, 24, 25, 38, 42, 55, 56

[CNZ16] Jiecao Chen, Huy L. Nguyen, and Qin Zhang. Submodular maximization over sliding windows. *CoRR*, abs/1611.00129, 2016. 1, 49

[Cor13] Graham Cormode. The continuous distributed monitoring model. *SIGMOD Record*, 42(1):5–14, 2013. 1

[CP15] Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 183–192, 2015. 7, 10, 12, 13, 38, 39

[CPW19] Ilan Reuven Cohen, Binghui Peng, and David Wajc. Tight bounds for online edge coloring. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1–25. IEEE Computer Society, 2019. 2

[CW09] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 205–214, 2009. 23

[CW18] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 960–979, 2018. 2

[DDH+08] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 932–941, 2008. 10

[DGIM02] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002. A preliminary version appeared in the Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2002. 1, 7, 47

[DM07] Mayur Datar and Rajeev Motwani. The sliding-window computation model and results. In *Data Streams - Models and Algorithms*, pages 149–167. Springer, 2007. 48

[DMMW12] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012. A preliminary version appeared in the Proceedings of the 29th International Conference on Machine Learning, ICML 2012. 14

[DRVW06] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006. A preliminary version appeared in the Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006. 11, 26

[DV06] Amit Deshpande and Santosh S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings*, pages 292–303, 2006. 4, 7, 26

[EHKS18] Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. In *Proceedings of the*

Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 700–714. SIAM, 2018. 2

[EHLM17] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary. *SIAM J. Discrete Math.*, 31(3):1685–1701, 2017. 2

[ELVZ17] Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. Submodular optimization over sliding windows. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 421–430, 2017. 1, 49

[Fre75] David A. Freedman. On tail probabilities for martingales. *the Annals of Probability*, 3(1):100–118, 1975. 32

[GKM+19] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 26–37. IEEE Computer Society, 2019. 2

[HK16] Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 128–141, 2016. 2

[KL13] Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013. 9

[Lew78] D Lewis. Finite dimensional subspaces of ℓ_p. *Studia Mathematica*, 63(2):207–212, 1978. 12

[LT06] Lap-Kei Lee and H. F. Ting. A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 290–297, 2006. 49

[MD09] Michael W. Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci. U.S.A.*, 106(3):697–702, 2009. 14

[MM12] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. *PVLDB*, 5(12):1699, 2012. 1

[MRWZ20] Sepideh Mahabadi, Ilya Razenshteyn, David P. Woodruff, and Samson Zhou. Non-adaptive adaptive sampling on turnstile streams. In *Symposium on Theory of Computing Conference, STOC*, 2020. 26

[NW18] Joseph (Seffi) Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3-4):16:1–16:20, 2018. 2

[OMM+14] Miles Osborne, Sean Moran, Richard McCreadie, Alexander Von Lunen, Martin Sykora, Elizabeth Cano, Neil Ireson, Craig MacDonald, Iadh Ounis, Yulan He, Tom Jackson, Fabio Ciravegna, and Ann O'Brien. Real-time detection, tracking and monitoring of automatically discovered events in social media. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014. 1

[PGD15] Odysseas Papapetrou, Minos N. Garofalakis, and Antonios Deligiannakis. Sketching distributed sliding-window data streams. *VLDB J.*, 24(3):345–368, 2015. 1

[PY06] Spiros Papadimitriou and Philip S. Yu. Optimal multi-scale patterns in time series streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 647–658, 2006. 1

[QAWZ15] Abdulhakim Ali Qahtan, Basma Alharbi, Suojin Wang, and Xiangliang Zhang. A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944, 2015. 1

[Rub16] Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 324–332, 2016. 2

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011. 19, 22

[Tro11] Joel Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011. 17

[Upa19] Jalaj Upadhyay. Sublinear space private algorithms under the sliding window model. In *International Conference on Machine Learning*, pages 6363–6372, 2019. 49

[WLL+16] Zhewei Wei, Xuancheng Liu, Feifei Li, Shuo Shang, Xiaoyong Du, and Ji-Rong Wen. Matrix sketching over sliding windows. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference*, pages 1465–1480, 2016. 1

A Smooth Histograms and RandNLA Functions

In this section, we define the popular smooth histogram framework for sliding window algorithms and then give counterexamples showing the approach is not amenable to many interesting linear algebraic functions.

A.1 Background on Smooth Histograms

An initial framework for approaching problems in the sliding window model is the *exponential histogram* data structure, introduced by Datar *et al.* [DGIM02]. Given a function f to approximate in the sliding window model, the exponential histogram partitions the data stream into "buckets", time intervals for which the evaluation of f on the data in each partition is exponentially increasing. For example, suppose we are given a data stream of integers and we want to approximate the number of ones in the sliding window within a factor of 2. In the exponential histogram data structure, the smallest bucket consists of all elements in the data stream from the most recent element to the most recent element whose value is one. The next bucket would consist of all previous elements until two elements whose values are one are seen. Similarly, the i^{th} bucket consists of the previous elements until 2^i instances of ones are seen. The key observation is that because the buckets are

exponentially increasing by powers of two, the starting point of the sliding window falls inside some bucket, and it will provide a 2-approximation to the number of ones seen in the sliding window even though it does not know exactly where the starting point is. Datar and Motwani [DM07] show that the exponential histogram framework is applicable to the class of "weakly additive" functions. Namely, if we let A and B be adjacent buckets and assume that $0 \leq f(A) \leq \text{poly}(N)$, where N is the length of the data stream, then a function is weakly additive if there exists some fixed constant $C_f \geq 1$ such that $f(A) + f(B) \leq f(A \cup B) \leq C_f(f(A) + f(B))$ holds for arbitrary A and B, where we recall that $A \cup B$ represents the concatenation of A and B. Moreover, if there exists a sketch of f, as well as a "composition" function that computes the sketch of $f(A \cup B)$ from the sketches of $f(A)$ and $f(B)$, then the exponential histogram framework provides a sliding window algorithm to approximate f.

Since the buckets in the exponential histogram data structure consist of disjoint elements, a crucial underlying requirement is that an approximation of f must be deducible from the merger of the information from these buckets. For example, it is not clear how to maintain buckets for the goal of approximating the geometric mean of a sliding window. To that effect, Braverman and Ostrovsky [BO07] define the notion of a smooth function, and provide the smooth histogram data structure as a framework for approximating smooth functions of sliding windows. They also show that the class of smooth functions contains the class of weakly additive functions, as well as a number of other functions, such as the geometric mean.

Smooth Histogram. Given adjacent buckets A, B, and C, a smooth function demands that if $(1-\beta)f(A \cup B) \leq f(B)$, then $(1-\alpha)f(A \cup B \cup C) \leq f(B \cup C)$ for some constants $0 < \beta \leq \alpha < 1$. Informally, a smooth function has the property that once a suffix of a data stream becomes a good approximation, then it *always* remains a good approximation, even with the arrival of new elements in the stream. With this definition of smooth function in mind, the smooth histogram data structure maintains a number of "checkpoints" throughout the data stream. Each checkpoint corresponds to a sketch of all the elements seen from the checkpoint until the most recently arrived element. Unlike the exponential histogram, the most recently arrived element impacts all sketches in the smooth histogram. A checkpoint is created with the arrival of each new element and checkpoints are discarded when their corresponding sketches get "too close" to the next checkpoint. That is, when the corresponding sketches of two checkpoints produce values that are within $(1-\beta)$ of each other, the later checkpoint is discarded, since by the property of smooth functions, the two checkpoints would thereafter always produce values that are within $(1-\alpha)$ of each other. This implies that, if the function is polynomially bounded, then the smooth histogram data structure only needs a logarithmic number of checkpoints. Moreover, Braverman and Ostrovsky [BO07] extend their results to the case where the sketch only provides an approximation to the evaluation of the function.

Smooth Functions. We use the notation $B \subseteq_s A$ if the stream of elements indexed by B are a suffix of stream of elements indexed by A (see Figure 5 for an example).

Definition A.1 (Smooth function)**.** *[BO07] A function $f \geq 1$ is (α, β)-smooth if it has the following properties:*

Monotonicity *$f(A) \geq f(B)$ for $B \subseteq_s A$ (B is a suffix of A)*

Polynomial boundedness *There exists $c > 0$ such that $f(A) \leq n^c$.*

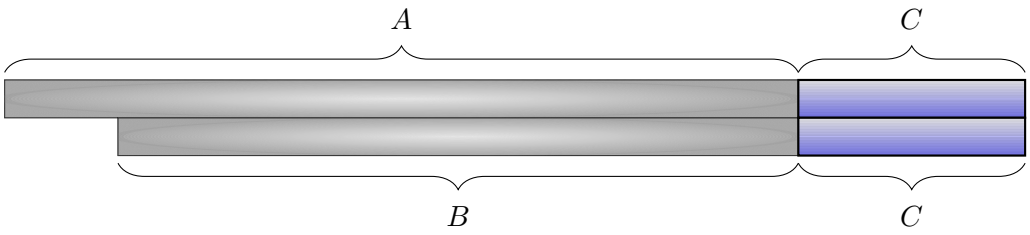

Fig. 5: A, B and C are substreams, where B is a suffix of A and C is adjacent to A and B. Smoothness says that if B is a "good" approximation of A, then $B \cup C$ will be a "good" approximation of $A \cup C$ for any C.

Smoothness *There exists $\alpha \in (0,1)$, $\beta \in (0,\alpha]$ so that if $B \subseteq_s A$ and $(1-\beta)f(A) \leq f(B)$, then $(1-\alpha)f(A \cup C) \leq f(B \cup C)$ for any adjacent C.*

The *smooth histogram* data structure estimates smooth functions in the sliding window model.

Definition A.2 (Smooth Histogram [BO07])**.** *Let g be a function that maintains a $(1+\varepsilon)$-approximation of an (α,β)-smooth function f that takes as input a starting index and ending index in the data stream. The approximate smooth histogram is a structure that consists of an increasing set of indices $X_N = \{x_1, \ldots, x_s = N\}$ and s instances of an algorithm Λ, namely $\Lambda_1, \ldots, \Lambda_s$ with the following properties:*

(1) x_1 corresponds to either the beginning of the data stream, or an expired point.

(2) x_2 corresponds to an active point.

(3) For all $i < s$, one of the following holds:

(a) $x_{i+1} = x_i + 1$ and $g(x_{i+1}, N) < \left(1 - \frac{\beta}{2}\right) g(x_i, N)$.

(b) $(1-\alpha)g_(x_i, N) \leq g(x_{i+1}, N)$ and if $i+2 \leq s$ then $g(x_{i+2}, N) < \left(1 - \frac{\beta}{2}\right) g(x_i, N)$.

(4) $\Lambda_i = \Lambda(x_i, N)$ maintains $g(x_i, N)$.

Unfortunately, despite being quite general, the smooth histogram frameworks cannot be applied to many interesting problems that have been extensively studied in the streaming model, such as clustering [BLLM15, BLLM16, BLUZ19], submodular maximization [CNZ16, ELVZ17], or heavy-hitter detection [LT06, BGO14, BGL+18, Upa19]. We now show that smooth histogram cannot be applied to many interesting numerical linear algebraic functions.

A.2 Lack of (α,β)-smoothness of RandNLA Functions

In this section, we show that the spectral norm, vector induced matrix norms, linear and generalized regression, and low-rank approximation are not amenable for the smooth histogram framework.

We first prove that low-rank approximation is not smooth for any meaningful parameters (α,β) in Definition A.1, even when the best low-rank approximations are nonzero.

Lemma A.3. *Low-rank approximation is not smooth for any meaningful parameters (α,β) in Definition A.1 that gives us a constant factor approximation.*

Proof. Our proof constructs a matrix and a vector explicitly. Let $\mathbf{e}_i$ be the elementary row vector with entry one in the i^{th} position and zero elsewhere. Given $0 < \alpha < 1$, let $d = \frac{2}{\alpha}$. Let S_A be the data stream whose first element is $2d\mathbf{e}_1$, second element is $\mathbf{e}_2$, followed by data stream S_B, which is a suffix of S_A, i.e., $S_B \subseteq_s S_A$. Then suppose B consists of the elements $2d\mathbf{e}_3$, followed by $\mathbf{e}_{i+3}$ for $1 \leq i \leq d$. Finally, let S_C be the data stream consisting of the single element $2d\mathbf{e}_j$, where $j = 4 + d$. Then the corresponding matrices appear as below:

$$\left[\begin{array}{ccccccc}
2\mathrm{d} & & & & & & \\
 & 1 & & & & & \\
\hline
 & & 2\mathrm{d} & & & & \\
 & & & 1 & & & \\
 & & & & 1 & & \\
 & & & & & \ddots & \\
 & & & & & & 1 \\
\hline
 & & & & & 2\mathrm{d} & \\
 & & & & & & 0 \\
 & & & & & & & \ddots
\end{array}\right],$$

where we have the following matrices

$$\mathbf{A} = \left[\begin{array}{ccccccc}
2\mathrm{d} & & & & & & \\
 & 1 & & & & & \\
\hline
 & & 2\mathrm{d} & & & & \\
 & & & 1 & & & \\
 & & & & 1 & & \\
 & & & & & \ddots & \\
 & & & & & & 1
\end{array}\right], \mathbf{B} = \left[\begin{array}{ccccc}
2\mathrm{d} & & & & \\
 & 1 & & & \\
 & & 1 & & \\
 & & & \ddots & \\
 & & & & 1
\end{array}\right], \text{ and } \mathbf{C} = \left[\begin{array}{ccc}
2\mathrm{d} & & \\
 & 0 & \\
 & & \ddots
\end{array}\right].$$

Then for $k = 2$, the best rank k approximation of $\mathbf{A}$ consists of two rows containing $2d$ so that

$$\min_{\substack{\mathbf{X} \in \mathbf{R}^{N \times n} \\ \mathrm{rank}(\mathbf{X}) = 2}} \|\mathbf{A} - \mathbf{X}\|_F = \sqrt{d+1}.$$

The best rank k approximation of matrix formed by S_B consists of the row containing $2d$ and any other elementary row in $\mathbf{B}$ so that

$$\min_{\substack{\mathbf{Y} \in \mathbf{R}^{N \times n} \\ \mathrm{rank}(\mathbf{Y}) = 2}} \|\mathbf{B} - \mathbf{Y}\|_F = \sqrt{d-1}.$$

Hence, the ratio of the best low-rank approximation of $\mathbf{B}$ to the best low-rank approximation of $\mathbf{A}$ is

$$\frac{\sqrt{d-1}}{\sqrt{d+1}} > \frac{d-1}{d+1} = 1 - \frac{2}{d+1} > 1 - \alpha.$$

Now, let $\mathbf{C}$ represent the matrix corresponding to stream $S_{A \cup C}$ and let $\mathbf{D}$ represent the matrix corresponding to stream $S_{B \cup C}$. Then the best rank k approximation of $S_{A \cup C}$ consists of two rows containing $2d$ so that

$$\min_{\substack{\mathbf{X} \in \mathbf{R}^{N \times n} \\ \mathrm{rank}(\mathbf{X}) = 2}} \|\mathbf{C} - \mathbf{X}\|_F = \sqrt{4d^2 + d + 1},$$

while the best rank k approximation of $S_{B\cup C}$ consists of two rows containing $2d$ so that

$$\min_{\substack{\mathbf{Y}\in\mathbf{R}^{N\times n}\\ \text{rank}(\mathbf{Y})=2}} \|\mathbf{D}-\mathbf{Y}\|_F = \sqrt{d}.$$

Thus, the ratio of the best low-rank approximation of $S_{B\cup C}$ to the best low-rank approximation of $S_{A\cup C}$ is at least 2, i.e., $\beta \le -1$. Therefore, low-rank approximation is not smooth. □

We now show that ℓ_p regression is not smooth as per the smooth histogram framework. Recall that for $\mathbf{A} \in \mathbb{R}^{N\times n}$ and $\mathbf{B} \in \mathbb{R}^{N\times d}$, the generalized ℓ_p regression problem is the minimization problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times d}} \|\mathbf{AX}-\mathbf{B}\|_p,$$

where $\|\cdot\|_p$ denotes the entrywise ℓ_p norm, that is, $\|\mathbf{X}\|_p = \left(\sum_{i,j} |\mathbf{X}_{i,j}|^p\right)^{1/p}$.

In our setting, each update to the data stream consists of a row vector $\mathbf{a}_i \in \mathbb{R}^d$ and an element b_i, i.e., the underlying matrix $\mathbf{A}$ represented by a sliding window of size W consists of the rows

$$\mathbf{A} = \begin{bmatrix}\mathbf{a}_{t-W+1} & \mathbf{a}_{t-W+2} & \cdots & \mathbf{a}_t\end{bmatrix}^\top$$
$$\mathbf{b} = \begin{bmatrix}b_{t-W+1} & b_{t-W+2} & \cdots & b_t\end{bmatrix}^\top.$$

We next show that ℓ_p regression is not smooth as per Definition A.1 for any reasonable parameters (α, β) (see Lemma A.4).

Lemma A.4. *ℓ_p regression is not smooth as per Definition A.1 for any reasonable parameters (α, β) that gives us a constant factor approximation.*

Proof. Let $0 < \alpha < 1$. Let A be the data stream whose first element is the row $\{\mathbf{a}_1 = \{100, 0, 0, 0, 0\}, b_1 = 100\}$, second element is the row $\{\mathbf{a}_2 = \{0, \alpha, 0, 0, 0\}, b_2 = 0\}$, followed by data stream B, which is a suffix of A. Then suppose B consists of the elements $\{\mathbf{a}_3 = \{0, 0, 1, 0, 0\}, b_3 = 1\}$, followed by $\{\mathbf{a}_4 = \{0, 0, 0, 1, 0\}, b_4 = 0\}$. Finally, let C be the data stream consisting of the single element $\{\mathbf{a}_5 = \{0, 0, 0, 0, 1000\}, b_5 = 2000\}$ Then the corresponding matrices appear as below:

$$\mathbf{A} = \left[\begin{array}{ccccc} 100 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1000 \end{array}\right], \quad \mathbf{b} = \left[\begin{array}{c} 100 \\ 0 \\ \hline 1 \\ 0 \\ \hline 2000 \end{array}\right],$$

where

$$\mathbf{A}_1 := \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_2 := \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{A}_3 := \begin{bmatrix} 0 & 0 & 0 & 0 & 1000 \end{bmatrix}$$

and $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ are the corresponding rows of the vector b. Let the matrix $\mathbf{A}_1$ and vector $\mathbf{b}_1$ represent data stream S_A, $\mathbf{A}_2$ and $\mathbf{b}_2$ represent data stream S_B, $\mathbf{A}_3$ and $\mathbf{b}_3$ represent $A \cup C$, and $\mathbf{A}_4$ and $\mathbf{b}_4$ represent $S_B \cup S_C$. Finally, let $\mathcal{Z}_i = \text{argmin}_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{A}_i\mathbf{x} - \mathbf{b}_i\|_p$ for $1 \le i \le 4$. Then one can verify that $\mathcal{Z}_1 = 1 + \alpha$ and $\mathcal{Z}_2 = 1$. On the other hand, $\mathcal{Z}_3 > 100$ but $\mathcal{Z}_4 = \sqrt[p]{2^p + 1}$. Thus, ℓ_p regression is not smooth as per Definition A.1 for any reasonable parameters (α, β) that gives us a constant factor approximation. □

Recall that the vector induced ℓ_p matrix norm is defined as

$$\|\mathbf{A}\|_p = \max_{\|\mathbf{x}\|_p=1} \|\mathbf{A}\mathbf{x}\|_p ,$$

and for $p = 2$, it is the same as the spectral norm (Schatten-∞ norm). We now show that vector induced norms are not smooth for all values of p.

Theorem A.5. *The vector induced matrix norm $\|\cdot\|_p$ is not a smooth function as per Definition A.1 for a meaningful parameters (α, β) for constant factor approximation.*

Proof. We give explicit construction of streams to show that there exists a stream for which the vector induced matrix norm is not smooth as per Definition A.1. Let $\mathbf{e}_1, \cdots, \mathbf{e}_4$ be the standard basis of $\mathbf{R}^n$. Let A be the data stream consisting of $\{\mathbf{e}_1\}$, followed by the suffix B, the data stream consisting of $\{\mathbf{e}_2\}$. Let C be the data stream consisting of $\{\mathbf{e}_1\}$. Define $\mathbf{A}$ to be the matrix representing A and $\mathbf{B}$ to be the matrix representing B. Let $\mathbf{R}$ be the matrix representing $A \cup C$ and $\mathbf{S}$ be the matrix representing $B \cup C$. Thus,

$$\mathbf{W} := \mathbf{A}^\top \mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{X} := \mathbf{B}^\top \mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\mathbf{Y} := \mathbf{R}^\top \mathbf{R} = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{Z} := \mathbf{S}^\top \mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Observe that

$$\|\mathbf{W}\|_p = \max_{\|\mathbf{x}\|_p=1} \|\mathbf{W}\mathbf{x}\|_p$$

Thus, $\|\mathbf{W}\|_p = 1$. Similarly, $\|\mathbf{X}\|_p = 1$ so that $(1-\alpha)\|\mathbf{W}\|_p \leq \|\mathbf{X}\|_p$ for any $0 < \alpha < 1$. On the other hand, $\|\mathbf{Y}\|_p = 4$ and $\|\mathbf{Z}\|_p = 1$. Hence, the vector induced matrix norm $\|\cdot\|_p$ is not a smooth function as per Definition A.1. □

A.3 A Generalization of Smooth Histograms for Spectral Approximation

In this section, we give a deterministic sliding window algorithm for spectral approximation that is neither space nor time optimal, but provides a natural generalization of the smooth histogram framework of [BO07] to spectral approximation in the sliding window model.

Theorem A.6. *Let $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$ be a stream of rows and κ be the condition number of the stream. Let $W > 0$ be a window size parameter and $\mathbf{A} = \mathbf{r}_{n-W+1} \circ \ldots \circ \mathbf{r}_n$ be the matrix consisting of the W most recent rows. Given a parameter $\varepsilon > 0$, there exists an algorithm that outputs a matrix $\mathbf{M}$ such that $\mathbf{A}^\top \mathbf{A} \preceq \mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top \mathbf{A}$ and uses $\mathcal{O}\left(\frac{d^3}{\varepsilon} \log \kappa\right)$ words of space.*

Proof. Consider Algorithm 10. Note that either $t_1 = n - W + 1$ or $t_1 < n - W + 1$, since if $t_1 > n - W + 1$, then there would be another matrix with an earlier timestamp. In the first case, we have $\mathbf{M}_1 = \mathbf{A}^\top \mathbf{A}$ since the outer product of each row $\mathbf{r}_i$ with $i \geq n - W + 1$ is added to $\mathbf{M}_1$ at time i.

In the second case, we have that $t_2 > n - W + 1$ or else $\mathbf{M}_1$ would have been deleted. That means at some point t we must have had $\mathbf{M}_1^{(t)} \preceq (1+\varepsilon)\mathbf{M}_2^{(t)}$ to delete all matrices with timestamps

Algorithm 10 Projection-cost preservation for low-rank matrices in the sliding window model

Input: A stream of rows $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{R}^d$, window size W, and an accuracy parameter $\varepsilon > 0$
Output: Projection-cost preservation for low-rank matrices in the sliding window model.

$\mathbf{M}_0 \leftarrow 0^{d \times d}$
for each row $\mathbf{r}_t$ **do**
  Suppose $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_s$ are defined
  $\mathbf{M}_{s+1} \leftarrow \mathbf{r}_t^\top \mathbf{r}_t$, $\mathbf{t}_{s+1} = t$ ▷*Keep timestamp for* $\mathbf{M}_{s+1}$
  for $i = s$ to $i = 1$ **do** ▷*Update sketches with* $\mathbf{r}_t$
    $\mathbf{M}_i \leftarrow \mathbf{M}_i + \mathbf{r}_t^\top \mathbf{r}_t$
  for $i = s$ to $i = 2$ **do**
    if $\mathbf{M}_{i-1}^\top \mathbf{M}_{i-1} \preceq (1+\varepsilon)\mathbf{M}_{i+1}^\top \mathbf{M}_{i+1}$ **then**
      Delete $\mathbf{M}_i$ and t_i and relabel indices
  if $t_2 \leq t - W + 1$ **then** ▷$\mathbf{M}_2$ *has expired*
    Delete $\mathbf{M}_1$ and t_1 and relabel indices
return $\mathbf{M}_1$

between t_1 and t_2, where $\mathbf{M}_j^{(t)} := \sum_{i=t_j}^{t} \mathbf{r}_i^\top \mathbf{r}_i$ is the value at time t of the sketch that is $\mathbf{M}_j$ at the end of the stream. Hence,

$$\mathbf{M}_1 = \mathbf{M}_1^{(t)} + \sum_{i=t}^{n} \mathbf{r}_i^\top \mathbf{r}_i \preceq (1+\varepsilon)\mathbf{M}_2^{(t)} + \sum_{i=t}^{n} \mathbf{r}_i^\top \mathbf{r}_i \preceq (1+\varepsilon)\mathbf{M}_2.$$

Since $\mathbf{M}_1 \succeq \mathbf{A}^\top \mathbf{A} \succeq \mathbf{M}_2$, it follows that $\mathbf{A}^\top \mathbf{A} \preceq \mathbf{M} \preceq (1+\varepsilon)\mathbf{A}^\top \mathbf{A}$.

Each matrix has dimension $d \times d$. Moreover, $\mathbf{M}_i \succeq (1+\varepsilon)\mathbf{M}_{i+2}$ for each index i, so that some singular value has increased by a factor of $(1+\varepsilon)$. Since there are at most d singular values and they can increase by a factor of at most κ, then there are at most $\frac{d}{\varepsilon} \log \kappa$ such matrices, and the space complexity follows. □

We now give the proof of Theorem 2.14.

Reminder of Theorem 2.14.

There exists a deterministic algorithm in the sliding window model that outputs a rank k projection-cost preservation of an input matrix $\mathbf{A}$ whose rank is at most $2k$. If the entries of $\mathbf{A}$ are integers that are bounded in magnitude by $\mathrm{poly}(\mathbf{A})$, then this algorithm stores $\mathcal{O}(k)$ rows of $\mathbf{A}$ and uses $\mathcal{O}\left(\frac{k^4}{\varepsilon} \log n\right)$ additional words of space.

Proof. Consider Algorithm 4. Similar to Algorithm 10, $\mathbf{r}_t^\top \mathbf{r}_t$ is added to each sketch and sketch i is deleted if $\mathbf{M}_{i-1} \preceq (1+\varepsilon)\mathbf{M}_{i+1}$. Hence we have $\mathbf{A}^\top \mathbf{A} \preceq \mathbf{M}_1 \preceq (1+\varepsilon)\mathbf{A}^\top \mathbf{A}$ for the output $\mathbf{M}_1$ of Algorithm 4. Moreover, ome singular value has increased by a factor of $(1+\varepsilon)$ between $\mathbf{M}_{i-1}$ and $\mathbf{M}_{i+1}$, by similar reasoning to Theorem A.6. Since there are at most $2k$ singular values and they can increase by a factor of at most κ, then there are at most $\frac{2k}{\varepsilon} \log \kappa$ such matrices. Thus $\mathbf{R}_1$ contains exactly the row span of $\mathbf{A}$ and since $\mathbf{R}_1 \supseteq \mathbf{R}_2 \supseteq \ldots$, it suffices to store $2k$ rows of $\mathbf{A}$ to track all the matrices $\mathbf{R}_i$. On the other hand, $\mathbf{U}_i \in \mathbb{R}^{2k \times 2k}$, but there are $\mathcal{O}\left(\frac{k}{\varepsilon} \log \kappa\right)$ such matrices, so the total additional working space to maintain the matrices is $\mathcal{O}\left(\frac{k^3}{\varepsilon} \log \kappa\right)$.

Finally, note that since $\mathbf{A}$ has integer entries bounded in magnitude by $n^{\mathcal{O}(1)}$, then the characteristic polynomial has coefficients that are bounded in magnitude by $n^{\mathcal{O}(k)}$. Moreover, $n^{\mathcal{O}(1)} \geq \|\mathbf{A}\|_F \geq \sigma_{\max}(\mathbf{A})$ so then $\sigma_{\min}(\mathbf{A}) \geq \frac{1}{n^{\mathcal{O}(k)}}$. Thus, $\log \kappa = \mathcal{O}(k \log n)$ and the space complexity follows. □

B Extended Proofs

B.1 Online Whack-a-Mole

For a given matrix $\mathbf{A}$ prepended by a regularization matrix $\mathbf{B}$, which is some copies of the identity matrix, the goal is to give weights to the rows of $\mathbf{A}$ so that the online leverage score of each row of $\mathbf{A}$ is uniformly bounded by some $\frac{\gamma T}{n}$, where $\gamma > 1$ and T is some upper bound on the sum of the online leverage scores of a matrix. We give the online whack-a-mole algorithm in Algorithm 11. The

Algorithm 11 Online Whack-a-Mole algorithm for uniformly bounded online leverage scores

Input: Matrix $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$, regularization parameter $\lambda > 0$, parameter $\gamma > 1$
Output: Reweighting matrix $\mathbf{W}$

$\mathbf{B}_0 \leftarrow \lambda \mathbb{I}_d$, $\mathbf{B} = \underbrace{\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_0}_{n \text{ times}}$
Let T be the sum of the online leverage scores of $\mathbf{B} \circ \mathbf{A}$.
$\mathbf{W} \leftarrow \mathbb{I}_n$
for $i = 1$ to $i = n$ **do**
    Let $\tau^{\mathsf{OL}}_{nd+i}(\mathbf{B} \circ (\mathbf{WA}))$ denote the online leverage score of row $nd + i$ of $\mathbf{B} \circ (\mathbf{WA})$.
    if $\tau^{\mathsf{OL}}_{nd+i}(\mathbf{B} \circ (\mathbf{WA})) > \frac{\gamma T}{n}$ **then**
        Decrease $\mathbf{W}_{i,i}$ so that $\tau^{\mathsf{OL}}_{nd+i}(\mathbf{B} \circ (\mathbf{WA})) = \frac{\gamma T}{n}$
return $\mathbf{W}$

intuition by the online whack-a-mole algorithm is simple. We start with a weight matrix $\mathbf{W} = \mathbb{I}_n$ and iterate through the rows of $\mathbf{A}$. If there exists a row $\mathbf{a}_i$ of $\mathbf{A}$ whose online leverage score with respect to $\mathbf{B} \circ (\mathbf{WA})$ is larger than $\frac{\gamma T}{n}$, then $\mathbf{W}_{i,i}$ is decreased until $\mathbf{a}_i$ has online leverage score $\frac{\gamma T}{n}$. Because (online) leverage scores are lower semi-continuous [CLM+15], such a weight $\mathbf{W}_{i,i}$ must exist. Moreover, lowering the weight of a row $\mathbf{a}_j$ does not affect the online leverage score of a row $\mathbf{a}_i$ for $i < j$. Thus, it remains to show that the number of rows whose weights are lowered is at most $\frac{n}{\gamma}$. We now prove Lemma 4.5.

Reminder of Lemma 4.5.

[CLM+15] Let $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$. Let $\lambda > 0$, $\mathbf{B}_0 = \lambda \mathbb{I}_d$, $\mathbf{B} = \underbrace{\mathbf{B}_0 \circ \ldots \circ \mathbf{B}_0}_{n \text{ times}}$ and T be the sum of the online leverage scores of $\mathbf{B} \circ \mathbf{A}$. For any $\gamma \geq 1$, there exists a diagonal matrix $\mathbf{W}$ with entries in $[0, 1]$ that contains at most $\frac{n}{\gamma}$ entries strictly less than 1 so that the online leverage score of any row of $\mathbf{WA}$ with respect to $\mathbf{C} := \mathbf{B} \circ (\mathbf{WA})$ is at most $\frac{\gamma T}{n}$. That is, if $\tau^{\mathsf{OL}}_i(\mathbf{C})$ denotes the leverage score of the i^{th} row of $\mathbf{WA}$ with respect to $\mathbf{C}$, then $\tau^{\mathsf{OL}}_i(\mathbf{C}) \leq \frac{\gamma T}{n}$ for all $i \in [n]$.

Proof. By the above argument, it suffices to show that the number of entries i such that $\mathbf{W}_{i,i} \neq 1$ is at most $\frac{n}{\gamma}$. Let $\mathbf{X} = \mathbf{B} \circ \mathbf{A}$ and for each $i \in [n]$, let $\zeta^{\mathsf{OL}}_i(\mathbf{X})$ denote the online leverage score of row $\mathbf{a}_i$ with respect to matrix $\mathbf{X}$. Note that decreasing the weight of a row $\mathbf{a}_i$ increases the online

leverage score of any subsequent rows $\mathbf{a}_j$ with $j > i$. Thus we have $\sum_{i \in S} \frac{\gamma T}{n} \leq \sum_{i \in S} \zeta_i^{\mathsf{OL}}(\mathbf{X}) \leq T$, so that the number of indices in S is at most $\frac{n}{\gamma}$. □

B.2 Derandomization of OnlineBSS

In this section, we formally derandomize the OnlineBSS algorithm of [CMP16] with a $\mathcal{O}(\log n)$ overhead. The OnlineBSS algorithm, described in Algorithm 12, takes parameters $\varepsilon \in (0,1)$ and $\lambda \geq 0$, as well as the rows of $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n \times d}$ in a stream. The algorithm uses the upper barrier and lower barrier method of [BSS12] to design a probability distribution for each row that depends on the previously sampled rows but will *always* output a matrix $\mathbf{M}$ such that $(1-\varepsilon)(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}) \preceq \mathbf{M}^\top \mathbf{M} + \lambda \mathbb{I} \preceq (1+\varepsilon)(\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I})$. Rather than show the correctness over the randomness of the algorithm, [CMP16] must instead bound the expected number of rows sampled by OnlineBSS over the randomness of the algorithm.

Algorithm 12 OnlineBSS

Input: Stream of rows $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^{1 \times d}$, accuracy $\varepsilon \in (0,1)$, regularization $\lambda \geq 0$.
Output: Additive-multiplicative spectral approximation to $\mathbf{A} := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$.

$c_U \leftarrow \frac{2}{\varepsilon} + 1$, $c_L =\leftarrow \frac{2}{\varepsilon} - 1$
$\mathbf{M} \leftarrow \emptyset$, $\mathbf{B}^U \leftarrow \lambda \mathbb{I}$, $\mathbf{B}^L \leftarrow -\lambda \mathbb{I}$
for each row $\mathbf{a}_t$ **do**
  $\mathbf{X}^U \leftarrow \mathbf{B}^U - \mathbf{M}^\top \mathbf{M}$, $\mathbf{X}^L \leftarrow \mathbf{M}^\top \mathbf{M} - \mathbf{B}^L$
  $p_t \leftarrow \min\left(1, c_U \mathbf{a}_i (\mathbf{X}^U)^{-1} \mathbf{a}_i^\top + c_L \mathbf{a}_i (\mathbf{X}^L)^{-1} \mathbf{a}_i^\top\right)$
  $\mathbf{B}^U \leftarrow \mathbf{B}^U + (1+\varepsilon)\mathbf{a}_i^\top \mathbf{a}_i$, $\mathbf{B}^L \leftarrow \mathbf{B}^U + (1-\varepsilon)\mathbf{a}_i^\top \mathbf{a}_i$
  With probability p_t, $\mathbf{M} \leftarrow \mathbf{M} \circ \frac{\mathbf{a}_t}{\sqrt{p_t}}$
return $\mathbf{M}$.

[CMP16] start with the upper and lower barrier matrices $\mathbf{B}^U = \lambda \mathbb{I}$ and $\mathbf{B}^L = -\lambda \mathbb{I}$, in the Loewner order sense. Upon the arrival of each row $\mathbf{a}_i$, $\mathbf{B}^U$ and $\mathbf{B}^L$ are incremented by $(1+\varepsilon)\mathbf{a}_i^\top \mathbf{a}_i$ and $(1-\varepsilon)\mathbf{a}_i^\top \mathbf{a}_i$ respectively, as to remain upper and lower barriers for $\mathbf{M}^\top \mathbf{M} + \lambda \mathbb{I}$ as long as it remains a good approximation to $\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}$ as rows of $\mathbf{A}$ arrive. These upper and lower barrier matrices are then used in conjunction with $\mathbf{M}$ to compute the sampling probability p_i of row $\mathbf{a}_i$. Namely, $p_i = \min\left(1, c_U \mathbf{a}_i (\mathbf{X}^U)^{-1} \mathbf{a}_i^\top + c_L \mathbf{a}_i (\mathbf{X}^L)^{-1} \mathbf{a}_i^\top\right)$, where $\mathbf{X}^U = \mathbf{B}^U - \mathbf{M}^\top \mathbf{M}$ and $\mathbf{X}^L = \mathbf{M}^\top \mathbf{M} - \mathbf{B}^L$ describe how far $\mathbf{M}^\top \mathbf{M} + \lambda \mathbb{I}$ is from the barriers $\mathbf{X}^U$ and $\mathbf{X}^L$, respectively. As $\mathbf{M}^\top \mathbf{M} + \lambda \mathbb{I}$ approaches one of the barriers, $\mathbf{X}^U$ or $\mathbf{X}^L$ approaches zero, so that the following rows have high sampling probability, which forces $\mathbf{M}^\top \mathbf{M} + \lambda \mathbb{I}$ to remain a good approximation to $\mathbf{A}^\top \mathbf{A} + \lambda \mathbb{I}$.

To analyze the expected number of rows sampled, [CMP16] defines

$$\mathbf{C}_{i,j}^U = \lambda \mathbb{I} + \frac{\varepsilon}{2} \mathbf{A}_i^\top \mathbf{A}_i + \left(1 + \frac{\varepsilon}{2}\right) \mathbf{A}_j^\top \mathbf{A}_j$$
$$\mathbf{C}_{i,j}^L = -\lambda \mathbb{I} - \frac{\varepsilon}{2} \mathbf{A}_i^\top \mathbf{A}_i + \left(1 - \frac{\varepsilon}{2}\right) \mathbf{A}_j^\top \mathbf{A}_j,$$

where $\mathbf{A}_i = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_i$ for each $i \in [n]$. Then by definition, $\mathbf{C}_{i,i}^U = \mathbf{B}_i^U$ and $\mathbf{C}_{i,i}^L = \mathbf{B}_i^L$, where $\mathbf{B}_i^U$ and $\mathbf{B}_i^L$ are the matrices $\mathbf{B}^U$ and $\mathbf{B}^L$ at time i. Finally, denoting $\mathbf{M}$ at time i by $\mathbf{M}_i$, the matrices

$\mathbf{Y}_{i,j}^U$ and $\mathbf{Y}_{i,j}^L$ are defined by:

$$\mathbf{Y}_{i,j}^U = \mathbf{C}_{i,j}^U - \mathbf{M}_j^\top \mathbf{M}_j$$
$$\mathbf{Y}_{i,j}^L = \mathbf{M}_j^\top \mathbf{M}_j - \mathbf{C}_{i,j}^L,$$

so that $\mathbf{Y}_{i,i}^U = \mathbf{X}_i^U$ and $\mathbf{Y}_{i,i}^L = \mathbf{X}_i^L$ are the matrices that represent the "distance" of $\mathbf{M}^\top\mathbf{M}+\lambda\mathbb{I}$ from the upper and lower barriers, to determine the sampling probability p_i. It suffices for [CMP16] to bound $\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j+1}^U)^{-1}\mathbf{a}_i\right] \le \mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{a}_i\right]$ and $\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j+1}^L)^{-1}\mathbf{a}_i\right] \le \mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{a}_i\right]$, since $\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i$ and $\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i$ are proportional to the online ridge leverage score τ_i^{OL} of $\mathbf{a}_i$. Specifically, they show that

$$\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i = \mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i \le \frac{2}{\varepsilon}\tau_i^{\mathsf{OL}}.$$

Thus by Lemma 2.2, the expected number of rows sampled is at most

$$\begin{aligned}\sum_{i=1}^n c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i\right] &= c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,i-1}^L)^{-1}\mathbf{a}_i\right]\\ &\le \sum_{i=1}^n c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i\right]\\ &\le \frac{2}{\varepsilon}(c_U+c_L)\sum_{i=1}^n \tau_i^{\mathsf{OL}} = \mathcal{O}\left(\frac{d}{\varepsilon^2}\log\kappa\right).\end{aligned}$$

We make the following modifications to OnlineBSS. We first round the sampling probability for each row up to a power of 2 between $\frac{1}{n}$ and 1, which only incurs $\mathcal{O}(1)$ additional rows sampled in expectation with those with sampling probability less than $\frac{1}{n}$ and multiplicative $\mathcal{O}(1)$ for those with samping probability at least $\frac{1}{n}$. We then pick a random threshold $t \in [0,1]$ and then sample row $\mathbf{a}_i$ if $p_i > t$. Since there are only $\log n$ distinct sampling probabilities, we can derandomize this process by trying all $\log n$ values of t. It remains to be shown that the expectation of p_i is still at most the online (ridge) leverage score τ_i^{OL}.

However, the bounds $\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j+1}^U)^{-1}\mathbf{a}_i\right] \le \mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j}^U)^{-1}\mathbf{a}_i\right]$ and $\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j+1}^L)^{-1}\mathbf{a}_i\right] \le \mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,j}^L)^{-1}\mathbf{a}_i\right]$ hold even when conditioned on the first j steps of the algorithm. The analysis of [CMP16] only requires that row $\mathbf{a}_j$ is sampled with probability p_j at step $j+1$ for whatever value of p_j results from the randomness of the algorithm. Thus by the same reasoning, it still follows that the expected number of rows sampled is at most

$$\begin{aligned}\sum_{i=1}^n c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{X}_{i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{X}_{i-1}^L)^{-1}\mathbf{a}_i\right] &= c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,i-1}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,i-1}^L)^{-1}\mathbf{a}_i\right]\\ &\le \sum_{i=1}^n c_U\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^U)^{-1}\mathbf{a}_i\right] + c_L\mathbb{E}\left[\mathbf{a}_i^\top(\mathbf{Y}_{i-1,0}^L)^{-1}\mathbf{a}_i\right]\\ &\le \frac{2}{\varepsilon}(c_U+c_L)\sum_{i=1}^n \tau_i^{\mathsf{OL}},\end{aligned}$$

which is at most $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log\kappa\right)$ by Lemma 2.2.