Near Optimal Linear Algebra in the Online and Sliding Window Models

Vladimir Braverman
Johns Hopkins University
Email: vova@cs.jhu.edu

Petros Drineas
Purdue University
Email: pdrineas@purdue.edu

Cameron Musco
University of Massachusetts Amherst
Email: cmusco@cs.umass.edu

Christopher Musco
New York University
Email: cmusco@nyu.edu

Jalaj Upadhyay
Apple
Email: jalaj.upadhyay@apple.com

David P. Woodruff
Carnegie Mellon University
Email: dwoodruf@cs.cmu.edu

Samson Zhou
Carnegie Mellon University
Email: samsonzhou@gmail.com

***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.

***Keywords*-streaming algorithms, online algorithms, sliding window model, numerical linear algebra**

I. 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 [8], [9], [18]–[20], event detection in social media [32], and data summarization [11], [25]. To model such settings, Datar *et al.* [21] 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 [2], [30], [34], [37]. 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 [33], [35]. 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 [6] from the data. Thus in applications such as clustering, the denoised projection can perform better than the original input. The online model has also been extensively used in many other applications, such as learning [3], [5], [28], (prophet) secretary problems [24], [26], [36], ad allocation [31], and a variety of graph applications [12], [13], [27].

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!

A. *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) [4]. 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.

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. In the full version of the paper [7], 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 [10] 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 I.1 (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 II.4.)*

(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 $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

$$\|\mathbf{M}-\mathbf{MP}\|_F^2 \in (1\pm\varepsilon)\|\mathbf{A}-\mathbf{AP}\|_F^2,$$

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 II.10.)

(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 $(1-\varepsilon)\|\mathbf{Ax}\|_1 \le \|\mathbf{Mx}\|_1 \le (1+\varepsilon)\|\mathbf{Ax}\|_1$ 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 IV.5.)*

Here we say the stream has condition number κ if the ratio of the largest to smallest nonzero singular values of any matrix formed by consecutive rows of the stream is at most κ.

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 $\mathrm{poly}(n)$ 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 I.1 the first such algorithms for these problems in the sliding window model, but also Theorem II.4 and Theorem II.10 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 [1], [23].

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 I.2 (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 $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

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

(See Theorem III.1.)

Theorem III.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 [4].

Theorem I.3 (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*

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

(See Theorem III.2.)

By comparison, the online row subset selection algorithm of [4] 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 [4] 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 [4] 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 $\|\mathbf{A}-\mathbf{MX}\|_F^2 \le (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. Our online row subset selection can also be adjoined with the online PCA algorithm of [4] to offer the promise of the existence of a submatrix $\mathbf{Y} \in \mathbb{R}^{k\times d}$ within $\mathbf{X}$ such that there exists a matrix $\mathbf{B}$ with $\|\mathbf{A}-\mathbf{BY}\|_F^2 \le (1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$.

Theorem I.4 (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*

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

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*

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

(See Theorem III.3.)

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

Theorem I.5 (Online ℓ_1-Subspace Embedding)**.** *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*

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

for all $\mathbf{x} \in \mathbb{R}^d$ *with high probability. (See Theorem IV.4.)*

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.

Framework I.6 (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 V.5).*
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* $\mathbf{P} \in \mathbb{R}^{d \times d}$*,*
$$\|\mathbf{M} - \mathbf{MP}\|_F^2 \in (1 \pm \varepsilon) \|\mathbf{A} - \mathbf{AP}\|_F^2 ,$$
while storing $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log^4 n \log^2 \kappa\right)$ *rows at any time. (See Theorem V.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* $(1-\varepsilon)\|\mathbf{Ax}\|_1 \le \|\mathbf{Mx}\|_1 \le (1+\varepsilon)\|\mathbf{Ax}\|_1$ *for all* $\mathbf{x} \in \mathbb{R}^d$*, while storing* $\mathcal{O}\left(\frac{d}{\varepsilon^2} \log^5 n \log \kappa\right)$ *rows at any time. (See Theorem V.11).*

All of the results presented using Framework I.6 are space optimal, up to lower order terms [1], [17], [23]. 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 I.1 nor our deterministic framework Framework I.6 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. We outline our methods in this extended abstract and defer all proofs to [7].

II. 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 $t+2$, i.e., the next update in the stream.

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 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.
 1: $\mathbf{M}_0 \leftarrow \emptyset$.
 2: $\alpha \leftarrow \frac{C}{\varepsilon^2}\log n$ with sufficiently large constant $C > 0$
 3: for each row $\mathbf{r}_t$ do
 4:     $\mathbf{M}_t \leftarrow \mathbf{r}_t$
 5:     Let $\mathbf{M}_{t-1} = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_{m_{t-1}}$.
 6:     for $i = m_{t-1}$ down to $i = 1$ do
 7:         $\tau_i \leftarrow$ SCORE$(\mathbf{m}_i, \mathbf{M}_t)$
 8:         $p_i \leftarrow \min(1, \alpha\tau_i)$
 9:         With probability $p_i$, $\mathbf{M}_t \leftarrow \frac{\mathbf{m}_i}{\sqrt{p_i}} \circ \mathbf{M}_t$
10:     Delete $\mathbf{M}_{t-1}$.
11: $\mathbf{M} \leftarrow \emptyset$
12: Let $\mathbf{M}_n = \mathbf{m}_1 \circ \ldots \circ \mathbf{m}_{m_n}$.
13: for $i = 1$ to $i = m_n$ do
14:     if timestamp of $\mathbf{m}_i$ is at least $n - W + 1$ then
15:         $\mathbf{M} \leftarrow \mathbf{M} \circ \mathbf{m}_i$
16: return $\mathbf{M}$
```

A. 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. [16] defined 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 [16], as well as introduce reverse online ridge leverage scores.

Definition II.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 $\operatorname{rank}(\mathbf{A}_i) > \operatorname{rank}(\mathbf{A}_{i-1})$ (respectively if $\operatorname{rank}(\mathbf{Z}_i) > \operatorname{rank}(\mathbf{Z}_{i+1})$).*

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.

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 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.

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 $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}(\mathbf{A}\circ\mathbf{r})$ then
2:     return $2\mathbf{r}(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{r}^\top$
3: else
4:     return 1
```

However, these quantities are related; [16] 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. We present these results for the case where the input matrix has full rank, noting that similar bounds can be provided if the input matrix is not full rank by replacing the smallest singular value with the smallest nonzero singular value.

Lemma II.2 (Bound on Sum of Online Ridge Leverage Scores)**.** *[16] 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 $\lambda \leq \sigma_{\min}(\mathbf{A})$.*

We 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 II.3 (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*

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

Theorem II.4 (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.*

B. 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 $\mathbf{A}_{(k)} := \operatorname{argmin}_{\operatorname{rank}(\mathbf{X})\le k} \|\mathbf{A}-\mathbf{X}\|_F^2$. Recall the following definition of a projection-cost preservation, from which it follows that obtaining a projection-cost preservation of $\mathbf{A}$ suffices to produce a low-rank approximation of $\mathbf{A}$.

Definition II.5 (Rank k Projection-Cost Preservation [15])**.** *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 $\mathbf{P} \in \mathbb{R}^{d\times d}$,*

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

[15] showed 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 II.6. *[15] Let $\mathbf{A} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n \in \mathbb{R}^{n\times d}$ and $\lambda \le \frac{\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}$. 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 [15] 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:

Lemma II.7. *[15] 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 $\sigma_p(\mathbf{A})^2 \ge \lambda$ and let $\mathbf{X} = \mathbf{A} - \mathbf{A}_{(p)}$. Let $\mathbf{S} \in \mathbb{R}^{m\times n}$ be a sampling matrix so that 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 $\left|\|\mathbf{SX}\|_F^2 - \|\mathbf{X}\|_F^2\right| \le \varepsilon\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$ with high probability.*

On the other hand, our space analysis in Section II-A relied on bounding the sum of the online leverage scores by $\mathcal{O}(d\log\kappa)$ through Lemma II.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 $\lambda \le \frac{\left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2}{k}$ and immediately apply Lemma II.3 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 [14]. 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 II.6.

We could set $\lambda = 0$ and still obtain a rank k projection-cost preservation of $\mathbf{A}$, but smaller values of λ correspond to larger 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 $\left\|\mathbf{B}-\mathbf{B}_{(k)}\right\|_F^2 \le \left\|\mathbf{A}-\mathbf{A}_{(k)}\right\|_F^2$. 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 II.3.

Lemma II.8 (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 $q_i = \min(1, \alpha \cdot$*

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

$\mathbf{r}_i(\mathbf{Z}_{i+1}{}^\top \mathbf{Z}_{i+1} + \lambda \mathbb{I})^{-1} \mathbf{r}_i^\top$). *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*

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

We now give a tighter bound on the sum of the online λ-ridge leverage scores l_i for $\lambda = \frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_2^2}{k}$.

Lemma II.9 (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* $\lambda = \frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_2^2}{\beta k}$. *Then* $\sum_{i=1}^n l_i = \mathcal{O}(k \log \kappa)$.

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 II.10 (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 with high probability, 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.*

We describe how to achieve nearly input sparsity runtime and elaborate on bounded precision and condition number in [7].

III. 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.

A. 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 $\lambda_i = \frac{\|\mathbf{A}_{i-1} - (\mathbf{A}_{i-1})_{(k)}\|_F^2}{k}$, 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 $\lambda = \frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2}{k}$. Although [16] was 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 $\lambda = \frac{\|\mathbf{A} - \mathbf{A}_{(k)}\|_F^2}{k}$, our algorithm outputs a rank k projection-cost preservation of $\mathbf{A}$ by Lemma II.6 and Lemma II.7. Moreover, our bounds for the sum of the online ridge leverage scores in Lemma II.9 show that our algorithm only samples a small number of rows, optimal up to lower order factors.

Theorem III.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*

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

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

B. Online Row Subset Selection

We next describe how to perform row subset selection in the online model. Our starting point is an offline algorithm by [15] and the paradigm of adaptive sampling [22], [23], [29], which is the procedure of repeatedly sampling rows of $\mathbf{A}$ with probability proportional to their squared distances to the subspace spanned by $\mathbf{Z}$. [15] first obtained a matrix $\mathbf{Z}$ that is a constant factor low-rank approximation to the underlying matrix $\mathbf{A}$ through ridge-leverage score sampling. They observed that a theorem by [22] 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.

[15] then adapted 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 [22] if $\mathbf{Z}$ is given to show that 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, [15] showed 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 II.9 to bound the number of rows sampled in this manner.

Theorem III.2 (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 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*

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

C. 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

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

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

[4] gave 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 [6]. 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 provably 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

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

Recall that our online row subset selection algorithm 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 [4] to output matrices $\mathbf{M}$ and $\mathbf{W}$ such that $\mathbf{M}\mathbf{W}$ 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 [4] 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 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 III.3 (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 $\mathbf{W} \in \mathbb{R}^{m\times d}$ at the end of the stream such that*

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

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

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

IV. ℓ_1-Subspace Embeddings

In this section, we consider ℓ_1-subspace embeddings in both the online model and the sliding window model.

Definition IV.1 ((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 $\max_{\mathbf{x}\in\mathbb{R}^d}\frac{|\mathbf{a}_i^\top\mathbf{x}|}{\|\mathbf{A}\mathbf{x}\|_1}$ 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 $\mathrm{rank}(\mathbf{A}_i) > \mathrm{rank}(\mathbf{A}_{i-1})$.*

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.

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

Lemma IV.2. *For $\varepsilon > \frac{1}{n}$, Algorithm 4 returns a matrix $\mathbf{M}$ such that with high probability, we have for all $\mathbf{x} \in \mathbb{R}^d$,*

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

To analyze the space complexity, it remains to bound the

Algorithm 4 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: $\ell_1$-subspace embedding of $\mathbf{A} := \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$.
1: $\mathbf{M} \leftarrow \emptyset$
2: $\alpha \leftarrow \frac{Cd}{\varepsilon^2} \log n$ with sufficiently large constant $C > 0$
3: for each row $\mathbf{a}_t$ do
4:     $\tau_t \leftarrow 4 \cdot \max_{\mathbf{x} \in \mathbb{R}^d} \frac{|\mathbf{a}_t^\top \mathbf{x}|}{\|\mathbf{M}\mathbf{x}\|_1}$
5:     $p_t \leftarrow \min(1, \alpha \tau_t)$
6:     With probability $p_t$, $\mathbf{M} \leftarrow \mathbf{M} \circ \frac{\mathbf{a}_t}{p_t}$
7: return $\mathbf{M}$.
```

Algorithm 5 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 $\ell_1$ sensitivity of $\mathbf{r}$ with respect to $\mathbf{A}$.
1: if $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A} \circ \mathbf{r})$ then
2:     return $4d \cdot \max_{\mathbf{x} \in \mathbb{R}^d} \frac{|\mathbf{r}^\top \mathbf{x}|}{\|\mathbf{A}\mathbf{x}\|_1}$
3: else
4:     return 1
```

sum of the online ℓ_1 sensitivities. Much like the proof of Theorem II.4, we first bound the sum of regularized online ℓ_1 sensitivities and then relate these quantities. We then require a few structural results relating online leverage scores and their relationships to regularized online ℓ_1 sensitivities.

Lemma IV.3 (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 $\sum_{i=1}^n \zeta_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d \log n \log \kappa)$.*

We now give the full guarantees of Algorithm 4.

Theorem IV.4 (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*

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

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

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 4. 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 $\text{rank}(\mathbf{Z}_{i-1}) = \text{rank}(\mathbf{Z}_i)$ 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 5 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 II.4 and Theorem II.10, 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 V 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 IV.5 (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 $(1 - \varepsilon) \|\mathbf{A}\mathbf{x}\|_1 \le \|\mathbf{M}\mathbf{x}\|_1 \le (1+\varepsilon) \|\mathbf{A}\mathbf{x}\|_1$ 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.*

We detail how to efficiently provide constant factor approximations to the sensitivities in [7].

V. A Coreset Framework for Deterministic Sliding Window Algorithms

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 V.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 before.*

We can use deterministic online coresets for deterministic sliding window algorithms using a merge and reduce framework. The idea is to store the $\mathsf{m_{space}}$ most recent rows in a block $\mathbf{B}_0$, for some parameter $\mathsf{m_{space}}$ 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 6.

Algorithm 6 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 $\mathbf{A}\in\mathbb{R}^{W\times d}=\mathbf{r}_{n-W+1}\circ\ldots\circ\mathbf{r}_n$

```
1:  Initialize blocks $\mathbf{B}_0,\mathbf{B}_1,\ldots,\mathbf{B}_{\log n}\leftarrow\emptyset$
2:  for each row $\mathbf{r}_t$ do
3:      if $\mathbf{B}_0$ does not contain $\mathsf{m}_{\mathsf{space}}$ rows then
4:          $\mathbf{B}_0\leftarrow\mathbf{r}_t\circ\mathbf{B}_0$
5:      else
6:          Let $i>0$ be the minimal index such that $\mathbf{B}_i=\emptyset$.
7:          $\mathbf{B}_i\leftarrow$ CORESET$\left(\mathbf{M},\frac{\varepsilon}{\log n}\right)$, where $\mathbf{M}=\mathbf{B}_0\circ\ldots\circ\mathbf{B}_{i-1}$
8:          for $j=0$ to $j=i-1$ do
9:              $\mathbf{B}_j\leftarrow\emptyset$
10:         $\mathbf{B}_0\leftarrow\mathbf{r}_t$
11:     if there exists a row $\mathbf{r}$ in a block $\mathbf{B}_i$ with timestamp before $t-W+1$ then
12:         Delete $\mathbf{r}$ from $\mathbf{B}_i$
13: return $\mathbf{B}_{\log n}\circ\ldots\circ\mathbf{B}_1\circ\mathbf{B}_0$
```

Lemma V.2. *$\mathbf{B}_i$ in Algorithm 6 is a $\left(1+\frac{\varepsilon}{\log n}\right)^i$ online coreset for $2^{i-1}\mathsf{m}_{\mathsf{space}}$ rows.*

Theorem V.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 $f(\mathbf{A})$.*

The online row sampling algorithm of [16] shows the existence of an online coreset for spectral approximation. This coreset can be inefficiently but explicitly computed by computing the online leverage scores, enumerating over sufficiently small subsets of scaled rows, and checking whether a subset is an online coreset for spectral approximation.

Theorem V.4 (Online Coreset for Spectral Approximation). *[16] 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* 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 before. Then $(1-\varepsilon)\left\|\mathbf{A}_i\mathbf{x}\right\|_2\le\left\|\mathbf{M}_i\mathbf{x}\right\|_2\le(1+\varepsilon)\left\|\mathbf{A}_i\mathbf{x}\right\|_2$ for all $\mathbf{x}\in\mathbb{R}^d$, where $\mathbf{A}_i=\mathbf{a}_1\circ\ldots\circ\mathbf{a}_i$.*

Then Theorem V.3 and Theorem V.4 imply:

Theorem V.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 $(1-\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_2\le\left\|\mathbf{M}\mathbf{x}\right\|_2\le(1+\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_2$ for all $\mathbf{x}\in\mathbb{R}^d$.*

Theorem III.1 shows that the existence of an online coreset for computing a rank k projection-cost preservation.

Theorem V.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* 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 before. 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 V.3 and Theorem V.6 give:

Theorem V.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 $(1-\varepsilon)\left\|\mathbf{A}-\mathbf{A}\mathbf{P}\right\|_F\le\left\|\mathbf{M}-\mathbf{M}\mathbf{P}\right\|_F\le(1+\varepsilon)\left\|\mathbf{A}-\mathbf{A}\mathbf{P}\right\|_F$ 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 IV.4, but in fact [17] showed the existence of an offline coreset for ℓ_1-subspace embeddings that stores a smaller number of rows. The offline coreset of [17] is based on sampling rows proportional to their Lewis weights. We define a corresponding online version of Lewis weights:

Definition V.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* $\operatorname{rank}(\mathbf{A}_i) > \operatorname{rank}(\mathbf{A}_{i-1})$.

Theorem V.9 (Online Coreset for ℓ_1-Subspace Embedding)**.** *[17] Let* $\mathbf{A} \in \mathbb{R}^{n\times d} = \mathbf{a}_1 \circ \ldots \circ \mathbf{a}_n$*, If there exists an upper bound* $C > 0$ *on the sum of the online Lewis weights of* $\mathbf{A}$*, then there exists a deterministic algorithm* $\textsc{Coreset}(\mathbf{A}, \varepsilon)$ *that outputs an online coreset of* $\mathcal{O}\left(\frac{C}{\varepsilon^2}\log n\right)$ *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 before. 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* $(1+\varepsilon)$.

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

Lemma V.10 (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* $\sum_{i=1}^n w_i^{\mathsf{OL}}(\mathbf{A}) = \mathcal{O}(d \log n \log \kappa)$.

Then from Theorem V.3, Theorem V.9, and Lemma V.10:

Theorem V.11 (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* $\mathcal{O}\left(\frac{d}{\varepsilon^2}\log^5 n \log\kappa\right)$ *rows and outputs a matrix* $\mathbf{M}$ *such that* $(1-\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1 \le \left\|\mathbf{M}\mathbf{x}\right\|_1 \le (1+\varepsilon)\left\|\mathbf{A}\mathbf{x}\right\|_1$ *for all* $\mathbf{x} \in \mathbb{R}^d$.

Note that Theorem V.3 also provides an approach for a *randomized* ℓ_1-subspace embedding sliding window algorithm that improves upon the space requirements of Theorem IV.5, 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 V.10 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 IV.4. We provide further details on efficient online coreset construction by derandomizing the OnlineBSS algorithm of [16] in [7].

Acknowledgements

Vladimir Braverman is supported in part by NSF CAREER grant 1652257, ONR Award N00014-18-1-2364 and the Lifelong Learning Machines program from DARPA/MTO. Petros Drineas is supported in part by NSF FRG-1760353 and NSF AF-1814041. This work was done before Jalaj Upadhyay joined Apple. David P. Woodruff is supported by the National Institute of Health (NIH) grant 5R01 HG 10798-2, subaward through Indiana University Bloomington, as well as Office of Naval Research (ONR) grant N00014-18-1-2562. Samson Zhou is supported in part by NSF CCF-1649515 and by a Simons Investigator Award of D. Woodruff.

References

[1] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319, 2016.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. 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.

[3] M. Balcan, T. Dick, and E. 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.

[4] A. Bhaskara, S. Lattanzi, S. Vassilvitskii, and M. Zadimoghaddam. Residual based sampling for online low rank approximation. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1596–1614, 2019.

[5] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202–204, 2003.

[6] C. Boutsidis, D. Garber, Z. S. Karnin, and E. Liberty. Online principal components analysis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 887–901, 2015.

[7] V. Braverman, P. Drineas, C. Musco, C. Musco, J. Upadhyay, D. P. Woodruff, and S. Zhou. Near optimal linear algebra in the online and sliding window models. *CoRR*, abs/1805.03765, 2018.

[8] V. Braverman, E. Grigorescu, H. Lang, D. P. Woodruff, and S. 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.

[9] V. Braverman, H. Lang, E. Ullah, and S. Zhou. Improved algorithms for time decay streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 27:1–27:17, 2019.

[10] V. Braverman and R. Ostrovsky. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS) Proceedings*, pages 283–293, 2007.

[11] J. Chen, H. L. Nguyen, and Q. Zhang. Submodular maximization over sliding windows. *CoRR*, abs/1611.00129, 2016.

[12] I. R. Cohen, B. Peng, and D. 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.

[13] I. R. Cohen and D. 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.

[14] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. 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.

[15] M. B. Cohen, C. Musco, and C. 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.

[16] M. B. Cohen, C. Musco, and J. W. Pachocki. Online row sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:18, 2016.

[17] M. B. Cohen and R. 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.

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

[19] G. Cormode and M. 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.

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

[21] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

[22] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006.

[23] A. Deshpande and S. S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 292–303, 2006.

[24] S. Ehsani, M. Hajiaghayi, T. Kesselheim, and S. 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.

[25] A. Epasto, S. Lattanzi, S. Vassilvitskii, and M. Zadimoghaddam. Submodular optimization over sliding windows. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 421–430, 2017.

[26] H. Esfandiari, M. Hajiaghayi, V. Liaghat, and M. Monemizadeh. Prophet secretary. *SIAM J. Discrete Math.*, 31(3):1685–1701, 2017.

[27] B. Gamlath, M. Kapralov, A. Maggiori, O. Svensson, and D. Wajc. Online matching with general arrivals. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 26–37. IEEE Computer Society, 2019.

[28] E. Hazan and T. 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.

[29] S. Mahabadi, I. Razenshteyn, D. P. Woodruff, and S. Zhou. Non-adaptive adaptive sampling on turnstile streams. In *Symposium on Theory of Computing Conference, STOC*, 2020.

[30] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. *PVLDB*, 5(12):1699, 2012.

[31] J. S. Naor and D. Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3-4):16:1–16:20, 2018.

[32] M. Osborne, S. Moran, R. McCreadie, A. V. Lunen, M. Sykora, E. Cano, N. Ireson, C. MacDonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, and A. 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.

[33] S. Papadimitriou and P. 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.

[34] O. Papapetrou, M. N. Garofalakis, and A. Deligiannakis. Sketching distributed sliding-window data streams. *VLDB J.*, 24(3):345–368, 2015.

[35] A. A. Qahtan, B. Alharbi, S. Wang, and X. 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.

[36] A. 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.

[37] Z. Wei, X. Liu, F. Li, S. Shang, X. Du, and J. Wen. Matrix sketching over sliding windows. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference*, pages 1465–1480, 2016.