Residual Based Sampling for Online Low Rank Approximation

Aditya Bhaskara
School of Computing
University of Utah
Salt Lake City, UT, USA
bhaskaraaditya@gmail.com

Silvio Lattanzi
Google Research Zürich
Zürich, Switzerland
silviol@google.com

Sergei Vassilvitskii
Google Research NYC
New York, USA
sergeiv@google.com

Morteza Zadimoghaddam
Google Research Zürich
Zürich, Switzerland
zadim@google.com

***Abstract*—We propose online algorithms for Column Subset Selection (CSS) and Principal Component Analysis (PCA), two methods that are widely employed for data analysis, summarization, and visualization. Given a data matrix A that is revealed one column at a time, the *online CSS* problems asks to keep a small set of columns, S, that best approximates the space spanned by the columns of A. As each column arrives, the algorithm must irrevocably decide whether to add it to S, or to ignore it. In the *online PCA* problem, the goal is to output a projection of each column to a low dimensional subspace. In other words, the algorithm must provide an *embedding* for each column as it arrives, which cannot be changed as new columns arrive.**

While both of these problems have been studied in the online setting, only additive approximations were known prior to our work. The core of our approach is an adaptive sampling technique that gives a practical and efficient algorithm for both of these problems. We prove that by sampling columns using their "residual norm" (i.e. their norm orthogonal to directions sampled so far), we end up with a significantly better dependence between the number of columns sampled, and the desired error in the approximation.

We further show how to combine our algorithm "in series" with prior algorithms. In particular, using the results of Boutsidis et al. [5] and Frieze et al. [15] that have additive guarantees, we show how to improve the bounds on the error of our algorithm.

***Keywords*-low rank approximation; online algorithms; column subset selection**

I. Introduction

Principal Component Analysis (PCA) and Column Subset Selection (CSS) are workhorses of modern machine learning and data analysis methods, and have been widely used widely for tasks as diverse as dimension reduction, denoising, data summarization, and data visualization.

Given a dataset of n observations with d features each, represented as a $d \times n$ data matrix A, and a parameter, k, the goal of PCA is to find a k-dimensional subspace V of $\mathbb{R}^d$, so as to minimize the sum of squared distances between the points and the corresponding projections to V. Formally, it can also be stated as finding a k-dimensional embedding Y_i for each input point A_i along with a subspace V (defined by a $d \times k$ matrix Φ whose columns form an orthonormal basis for V) so as to minimize $\|A - \Phi Y\|_F^2$. (Here Y is the matrix whose columns are Y_i.)

When A is given, it is well known that the optimum Φ is given by the top k left singular vectors of A. Moreover, given Φ, the optimum Y is a column by column projection of the corresponding data columns onto Φ. As the subspace and the embedding are closely related, the PCA problem has been studied either with an emphasis on the embedding Y_i (as in the case of applications involving denoising, dimension reduction, spectral clustering, etc.), or with an emphasis on the subspace (in applications that involve understanding the *structure* of the data).

The other problem we consider is column subset selection (CSS). While the goal of PCA is to find the k-dimensional space that minimizes the projection error, the aim of CSS is to find a small subset of the *columns* such that the projection error of the matrix to the span of the chosen columns is minimized. Formally, given A, the goal is to find a subset $S \in \mathbb{R}^{d \times k}$ of the columns of A that minimizes:

$$\min_{X \in \mathbb{R}^{k \times n}} \|A - SX\|_F^2.$$

The restriction of S to come from columns of A is important for many applications such as data summarization, "interpretable" dimension reduction, and feature selection (where we work with the transpose of the matrix). The PCA and CSS problems turn out to be closely related. The results of [4], [18] show that the objective values of the two problems are related (for slightly differing values of k). Further, it turns out that many of the algorithms for CSS involve first computing the singular value decomposition.

Given the importance of the two problems and their numerous applications, several fast approximation algorithms have been proposed [4], [6], [7], [11], [13], [14], [15], [16], [18], [22], [24], [29], [30], [31], [33]. One class of works considers the offline case (where the full matrix A is known) and nearly linear time algorithms have been designed. Another class of results considers the case in which A is distributed across machines, with the goal of sub-linear total time. The problems have also been studied in the streaming model of computation, as we discuss below. As the literature here is so extensive, we will only survey some of the most closely related works in Section I-B.

Online model.: Online algorithms are a powerful paradigm in algorithm design. In the typical scenario, the

input arrives in increments (depending on the problem), and the goal is to make decisions on the input as it arrives. The goal is to compete with the optimum *offline* solution that has access to the entire input.

The online model is extensively studied for problems arising in data structures, caching and paging, packing and scheduling, secretary problems, and so on. More relevant to our work, online algorithms have also been used in many settings in learning and data analysis. A relevant work in this are is that of Meyerson [25] on online facility location. In this setting, points in a dataset arrive one after another, and the goal is to either assign a point to an existing cluster or form a new cluster of its own. The objectives here are the cost (e.g. a k-means type objective) and the number of clusters formed. Works on online clustering also provide inspiration for our techniques, as we will describe.

Our focus in this work is to consider the PCA and CSS problems in the online model. Let us define the problems and provide some motivation.

Definition I.1 (Online PCA)**.** *We are given an integer $k \geq 1$, and a matrix $A \in \mathbb{R}^{d \times n}$ whose columns $\{A_i\}$ arrive one after another. Once we receive an A_i, we are required to output a "low dimensional embedding" of it, denoted Y_i. At the end, we want the guarantee that there exists a $d \times k$ matrix Φ with orthogonal columns, such that $\|A - \Phi Y\|_F^2$ is minimized.*

The idea behind the formulation is that the embedded vectors should be interpretable "in hindsight" as the projections of the original vectors onto a low-dimensional subspace. The algorithm may or may not maintain such a Φ as it processes the vectors. However the important restriction is that the embedding produced for a point cannot be changed at a later point of time.

The formulation captures the essence of many applications. For instance, in spectral clustering (for which theoretical guarantees are also known [21]), we may have points arriving one by one, and an algorithm for online PCA (if it produces a near-optimal embedding) could be used to perform dimension reduction "on the fly", which could then feed into other algorithms (e.g., a clustering algorithm [25]). The formulation can also be used in related applications such as denoising and dimension reduction. For example well-known applications of PCA embedding in this context are in visualization [32] or in biology [28].

Definition I.2 (Online CSS)**.** *We are given an integer $k \geq 1$, and a matrix $A \in \mathbb{R}^{d \times n}$ whose columns $\{A_i\}$ arrive one after another. At each step, we need to decide whether to keep the current A_i or not. Once kept, the column cannot be discarded. The goal is to keep a subset S of columns of size at most k, while minimizing $\left\|\Pi_S^{\perp} A\right\|_F^2$. Here $\Pi_S^{\perp}$ denotes projection onto the space orthogonal to the span of S (see Section II for details).*

Note that this definition of online CSS is a natural "secretary" style question, in which we are allowed to slightly violate the cardinality constraint. In other words, if we view $\Pi_S^{\perp} A$ as a set function (of S), we wish to find a set S in an online manner, in order to compete in objective value with the best S in hindsight. (Indeed, we attempt to compete with $\left\|A - A^{(k)}\right\|_F^2$, which is only lower than $\min_{|S|=k} \left\|\Pi_S^{\perp} A\right\|_F^2$; here $A^{(k)}$ is the best rank k approximation of A.) As in the PCA case, this formulation has strong motivations. For instance, in applications such as feature selection (or even data summarization), once certain features have been selected (i.e. identified as important), they may be used in other downstream tasks. Thus removing a chosen column later may not be realistic.

Flavor of our results.: For both of the problems, our results are *bi-criteria*, i.e., they involve trade-offs between the error in the approximation and the output dimensionality (in the PCA case, the embedding dimension, and in the CSS case, the number of selected columns). This type of bi-criteria approximation is common in online algorithms (e.g. facility location [25] and clustering [23]). For CSS, we note that even in the offline case, the best known results are bi-criteria: they use roughly k/ϵ columns in order to compete with the best k-column solution (see [18]).

Online vs. streaming.: The online model is similar at a high level to the streaming settings for PCA and CSS that have been well-studied. In both models, the columns arrive one after another, and we need to make some decisions as they arrive. In the streaming setting, the parameter of interest is typically the space and time complexity of each iteration. For instance, in streaming PCA, the goal is usually to maintain the *subspace* corresponding to the top-k SVD, and not to produce an (unchangeable) embedding of each column. Likewise in streaming CSS, the goal is to keep track of a subset of columns (that can be added/removed), with an approximation guarantee at the end. In this sense, the online model is much more restrictive (although traditionally, space and time bounds are not stringent). It is also interesting to note that the online solution is particularly useful in practice when one wants to keep a stable and consistent solution on an evolving dataset. In fact, online algorithms automatically achieve consistency during their execution.

Prior work.: Recently, Boutsidis et al. [5] gave the first online algorithm for the problem of computing the PCA embedding defined above. Their approach yields an *additive* guarantee on the error, of $\epsilon\|A\|_F^2$, and give an embedding into $\mathbb{R}^{k/\mathrm{poly}(\epsilon)}$. Our focus is to obtain algorithms with a much better trade-off between the target error and the embedding dimension, as we will see.

The version of CSS defined above has not been explicitly studied to the best of our knowledge. However, a closely related work is that of Cohen et al. [10], who study the question of approximating the *entire* column span (swapping rows and columns from their work to fit ours) in

an appropriate sense, while using only a small number of columns. While our results are similar at a very high level, the problem there is quite different from ours, and so are the techniques. Another related work is that of sampling based on ridge leverage scores, due to Cohen et al. [9]. However, the paper [9] focuses on the offline and the streaming setting for the problem, which is less restrictive than the online version.

A. Our contributions

We present efficient algorithms for the online variants of both PCA and CSS. Our algorithms rely on sampling procedures that build up a low rank approximation, by adding new directions based on "residual norms". We give a description of these techniques in Section I-C. But first, we state our main results and compare them with prior work.

1) Online PCA:

Theorem I.3. *Suppose the columns $\{A_i\}$ of a matrix $A \in \mathbb{R}^{d\times n}$ arrive in an online manner. Suppose also that we are given an integer $k \geq 1$, a target error ξ, and an upper bound L on the quantity $\log \frac{\|A\|_F^2}{\xi}$. Then for any $\epsilon, \delta > 0$, there exists an efficient algorithm that upon seeing each A_i, outputs an embedding $Y_i \in \mathbb{R}^r$, with the following properties:*

1) *The dimension r satisfies $r \leq O\left(\frac{k}{\epsilon^2}\right)\cdot(L+\log(1/\delta))^4$.*
2) *In the end, with probability $\geq 1-\delta$,*

$$\min_{\Phi\in\mathbb{R}^{d\times \dim(Y)}, \Phi^T\Phi=I} \|A-\Phi Y\|_F^2 \leq (1+\epsilon)\left\|A-A^{(k)}\right\|_F^2 + \epsilon\xi. \quad (1)$$

First, let us compare the bound above with the prior work of Boutsidis et al. [5]. For direct comparison, let us think of the target error ξ as $\gamma\|A\|_F^2$, and let the desired success probability be a constant, say $9/10$. Also suppose that the optimal error $\left\|A-A^{(k)}\right\|_F^2$ is negligible compared to ξ (for intuition, it also helps to think of $\epsilon = 1/2$). Then, to obtain a guarantee as in (1), the work of [5] uses an embedding dimension of $O\left(\frac{k}{\epsilon^2\gamma^2}\right)$. (Because the additive error is $\epsilon\gamma\|A\|_F^2$.) We also note that this is for the *first* algorithm of [5], where $\|A\|_F^2$ is assumed to be known up to a constant; for their main result, the dependence is at least as bad as $k/(\gamma\epsilon)^3$. Now for this setting of ξ, our bound is $O(k/\epsilon^2)\cdot L^4$, where L is $\log(10/\gamma)$ (the factor 10 is due to the choice of success probability). Thus the dependence on $(1/\gamma)$ improves exponentially.

Furthermore it is worth noticing that our algorithm can achieve multiplicative approximation if ξ is $\in O(\left\|A-A^{(k)}\right\|_F^2)$.

Knowing L.: Although we have a much better dependence on the error, note that our algorithm assumes a known bound L on $\log\frac{\|A\|_F^2}{\xi}$ (and the embedding dimension depends on this bound). While this may seem like a strong restriction, we note that it is an estimate of a logarithmic term. Allowing the number to be known to a factor say 10 allows for a huge variation in the allowed range for $\|A\|_F^2$. Thus in many applications, we do not expect this restriction to be a bottleneck. Secondly, we note that this restriction is an artifact of the problem statement (which requires one to commit to the embedding dimension *at the start*). If one is allowed to output embeddings of growing length, with the understanding that 0s are appended in the end, this assumption of a known L can be removed. Indeed, we see this in the case of online CSS.

2) Online CSS:

Theorem I.4. *Suppose the columns $\{A_i\}$ of a matrix $A \in \mathbb{R}^{d\times n}$ arrive in an online manner. Let $k \geq 1$ be an integer, and let ξ be a target value for the error. Then there exists an algorithm, that after seeing each column A_i, decides to either ignore it, or add it to a selection set S (which starts off empty). In the end, the following properties hold with probability $\geq 2/3$:*

1) $|S| \leq O(k/\epsilon^2)\cdot\log^2\frac{\|A\|_F^2}{\xi}$.
2) *The running time of each step is $O(|S|d)$ (or $O(|S|)$ times the number of non-zeros in A_i).*
3) *The set S (at the end) satisfies:*

$$\left\|\Pi_S^{\perp}A\right\|_F^2 \leq (1+\epsilon)\left\|A-A^{(k)}\right\|_F^2 + \epsilon\xi.$$

Note that the theorem for CSS is stronger than the one for PCA both qualitatively and quantitatively. First, we we do not require the knowledge of L. Intuitively, this is because we do not need to output an embedding of each point as it arrives, and hence do not need to know the embedding dimension. Second, the dependence on the logarithmic factor is much weaker. This is an artifact of the fact that our results proceed in two *steps*. First, we obtain a weaker approximation, and then we improve it. The second step for CSS turns out to be much more efficient.

Finally, note that we had a success probability in Theorem I.3 while Theorem I.4 uses a constant. This can be achieved for the CSS case also by simply running multiple copies of the procedure above as we see the columns A_i, and selecting the column iff at least one of the copies selects it. This increases the bound on the size of S by a $\log(1/\delta)$ factor as before.

Necessity of the logarithmic factor: Note that both our bounds, for PCA and CSS, have a dependence on the term $\log\frac{\|A\|_F^2}{\xi}$. This turns out to be an essential consequence of adaptive *residual based* sampling which is key to our algorithm (see the section on our techniques). Lemma VI.1 shows that this is unavoidable in our current style of analysis.

B. Related work

The problem of efficiently computing PCA given a matrix A has a rich and storied history. The closest works (besides [5] which we described earlier) is the work in the

streaming setting. Most of the work here, e.g., [13], [14], [16], [22], [24], [30], has focused on efficiently computing the top singular vectors of A in a single pass through the data. If one needs to output the embedding, of each vector we thus need a second pass.

In addition to these works, the stochastic version of PCA received a lot of attention [1], [3], [26] in the machine learning community. In this setting the columns of the matrix A are drawn from an unknown distribution and are exposed to the algorithm one by one. While this distributional assumption is common in machine learning, removing it is extremely challenging. Another interesting variation of the problem is the "online learning" version analyzed by [27], [34]. In this setting the algorithm has to commit to a rank k subspace before observing the column to project. This problem is related but incomparable to ours: the algorithm has to commit to a subspace before observing a column but at the same time it has the flexibility change the embedding subspace in every step. This line of research it has also been extended recently to the bandit setting [20]. Another related result is that random projections [8], [30] can be used to obtain a result similar to the one we would like to achieve when solving online PCA objective. The main difference between these two problems is that the embedding vectors Y_i can no longer be interpreted as projections onto a low dimensional subspace. Informally, the projections obtained via random projections capture "all" the singular directions well. Thus (at least intuitively) the Y_i cannot be used in applications such as denoising and spectral clustering. Finally, Karnin and Liberty [19] studied the online-PCA version for the spectral norm and provide an additive approximation for this problem.

Similarly to PCA, the CSS problem has received a lot of attention in recent years and several efficient algorithms have been proposed for the offline problem [4], [6], [7], [11], [18], [29], [31], [33]. Despite all this attention, to the best of our knowledge, we give the first online algorithm to the online CSS problem. In addition to the work on streaming CSS, the closest work to ours is the work on online row sampling [10] (discussed earlier). Although the problem studied there is similar, the goal is fundamentally different. In [10] the authors focus on preserving a spectral approximation of the matrix A, i.e., they focus on preserving the entire row span. In our setting, the goal is to find only the best rank k approximation, and this makes the results difficult to compare. Finally, our general approach is also related to the classic adaptive sampling technique [11], [29] but our focus is on single pass online algorithms.

C. Overview of techniques

Our key algorithmic contribution is to tackle the online versions of PCA and CSS problems using ideas similar to adaptive sampling. Our inspiration here are the beautiful results (a) of Meyerson [25] for online facility location, which achieves a constant competitive ratio by opening facilities at a given point with the probability proportional to the service cost of the point, and (b) of Deshpande and Vempala for low rank approximation [13], which shows that to obtain a fast algorithm for low rank approximation it is sufficient to sample in multiple rounds columns with probabilities proportional to their "residual norm", i.e., the distance from the span of the columns selected in previous rounds. Note that sampling based on distances to the *current set* is also the basis for the k-means++ algorithm, as well as the D^2 sampling paradigm for clustering [2]. Given these results, it is natural to ask if one can design efficient algorithms for online PCA and CSS using adaptive sampling based on the residual vectors. We give an affirmative answer to this question at a high level, though the implementation and analysis turns out to be more subtle than algorithms for clustering.

In what follows, let L denote $\log \frac{\|A\|_F^2}{\xi}$, where ξ is the target error, as in the statements of the theorems. For ease of exposition here, let us also suppose that $\xi \geq \left\|A - A^{(k)}\right\|_F^2$. Indeed, we make this assumption in Sections III and IV and remove it in Section V.

First result – logarithmic approximation: Our results for both PCA and CSS rely on first designing a simpler algorithm that achieves logarithmic approximation. I.e., the error will be bounded by roughly $O(L)\xi$. The idea behind this algorithm (say in the case of PCA) is to maintain a set of directions V which have been deemed "significant", and when a new column arrives, make a decision based on its *residual norm* (i.e., its norm orthogonal to the current V). The decision involves either adding the new column to V (which is done if the residual squared norm exceeds roughly ξ/k), or it is added to an "not-so-important" set.

Let us first ask the simple question: can there be many columns which get added directly? I.e., can there be many columns whose projection onto the earlier columns is $\geq \xi/k$? Intuitively, this cannot happen too many times if the rank-k error is smaller than ξ. This is shown formally in our "geometric lemma" (Lemma II.1).

Let us now come to how to treat the "not-so-important" set. Clearly, an online algorithm with small memory cannot keep track of all the unrepresented vectors so far. Instead, the key idea is to create a "running sketch" of the not-so-important vectors. This is done by keeping a single vector $\mathbf{w}$ which keeps track of the sum of these vectors so far, combined with random signs. Once the squared length of $\mathbf{w}$ exceeds ξ/k, we add this vector to V, and reset $\mathbf{w} = 0$. (In other works, the sketch vector is viewed as a proxy for all the not-so-important vectors in the current "phase".)

Then, the same argument as before can be used to conclude that there cannot be too many such $\mathbf{w}$ vectors added to V as the entire algorithm proceeds. Thus the key step is in defining the sense in which the "running sketch"

approximates the vectors that are in the phase. We show that the fact that random signs are used can be used to argue that the squared norms of the vectors as well as projections to certain crucial subspaces are maintained. Putting all of these ideas together leads to an $O(L)\xi$ error bound for online PCA. (See Section III-A.)

In the case of CSS, the above procedure cannot be followed, as $\mathbf{w}$ can be a combination of "too many" of the original vectors. However, we show that a norm-based sampling that chooses only $O(1)$ of the vectors in each phase can *also* act as a proxy for that phase. This key observation leads to an $O(L)\xi$ bound for online CSS. (See Section IV-A.)

Improving the error to $\left\|A - A^{(k)}\right\|_F^2 + \epsilon\xi$*:* The key question we ask is: suppose we are fine with losing an additional factor of L. Can we change the $O(L) \cdot \xi$ error to something smaller? The key observation is that our earlier algorithm, while producing an embedding into the space of the matrix V that it maintains, also keeps track of the error in the embedding. These *new residuals*, by the guarantee of the algorithm, have a total Frobenius norm at most $O(L)\cdot\xi$. The main idea is to now use an *additive approximation* algorithm (e.g., one from the work of [5]) in order to reduce the overall error, by in some sense "recycling" the residuals.

It turns out that this can indeed be made formal. However, due to technical reasons, we need an additive approximation algorithm with a certain *incremental* property. We observe that this holds for one of the algorithms of [5], and thus complete our proof. (See Section III-B.)

Once again for the CSS case, it turns out that this idea needs to be implemented in a slightly different way. But owing to the difference in the nature of the question, we can indeed use classical norm-sampling results of Frieze, Kannan and Vempala [15] in order to obtain the desired guarantees. (See Section IV-B.)

II. Notation and preliminaries

We start with some basic matrix notation we use throughout the paper. Let A be a $d\times n$ matrix. Throughout the paper, we write $A^{(k)}$ to refer to the best rank-k approximation of A (thus it is also a $d \times n$ matrix). For a subset T of the column indices ($T \subseteq [n]$), we denote by $A_{(T)}$ the $d \times |T|$ sub-matrix of A formed by the columns indexed by T. As is standard, we let $\sigma_i(\cdot)$ denote the ith largest singular value of matrix (which we define as the eigenvalues of the matrix times its transpose; note that some works define the singular values as the square roots of these quantities). We also use $\sigma_{\max}$ for the largest singular value, for clarity.

Next, for a set of vectors W, their linear span will be denoted span(W). We use $\Pi_W^{\perp}$ to denote the projection matrix orthogonal to span(W). For a *subspace* W, we abuse notation slightly and denote by $\Pi_W^{\perp}$ the projection matrix to the space orthogonal to W.

Finally, for consistency, we use S throughout the paper to denote subsets of columns, and V to denote linear combinations of subsets of columns.

Note on the logarithms.: Many of our bounds and theorem statements involve logarithmic terms. We do not explicitly make sure that the arguments are > 1 (for example if we choose ξ too large in Theorem I.3, the error bound becomes negative, which does not make sense). To be formally correct for all values of parameters, we need to ensure that we consider the maximum of the corresponding argument and 1. For ease of notation, we will not explicitly do this except in Section V, where it turns out to be important.

A. Key geometric lemma

The following geometric lemma is used crucially in our proof. Intuitively, it says that if we have an ordered set of r vectors with the property that each vector has a large component orthogonal to the previous ones, then either the matrix does not have a good rank k approximation, or the number of columns (r) is small.

Let $v_1, v_2, \ldots, v_r \in \mathbb{R}^d$ be a set of vectors. We denote by Π_i the projection matrix to span$(v_1, \ldots, v_i)$, and by $\Pi_i^{\perp}$ the projection matrix to the orthogonal space, i.e., $\Pi_i^{\perp} = I - \Pi_i$. Our setting implies that $r \leq d$, as the vectors are all linearly independent.

Lemma II.1. *Let $c > 0$ be any constant, and let Γ be a parameter satisfying $\Gamma \geq \frac{1}{c}\cdot\left\|V - V^{(k)}\right\|_F^2$, and suppose that vectors $\{v_i\}$ satisfy the property that for all i, $\|\Pi_{i-1}^{\perp} v_i\| \geq \gamma$. Suppose additionally that $\gamma^2 \geq \frac{2c\Gamma}{k}$. Then the number of columns r satisfies the bound*

$$r \leq 2k \cdot \frac{\log\left(\frac{\|V\|_F^2}{2c\Gamma}\right)}{\log\left(\frac{\gamma^2 k}{c\Gamma}\right)} \leq 2k \cdot \log\left(\frac{\|V\|_F^2}{2c\Gamma}\right). \qquad (2)$$

Remark.: We note that our requirement is similar in spirit with the *leave-one-out* distance that has been considered in other works. The key difference is that we have an ordered set of vectors, and we require the vectors to have a non-trivial orthogonal component to all of the *preceding* vectors, whereas in the standard notion, each vector need to have a non-trivial orthogonal component to *all* of the other vectors in the set. An analogous lemma for the standard leave-one-out distance turns out to be easier to show and has a better dependence on the parameters involved.

Proof: The second inequality in (2) follows from the assumption on γ, and thus we focus on proving the first. Let K be the parallelopiped formed by the columns of V (with the origin). Our proof relies on the well-known fact that:

$$\text{vol}(K) = \sqrt{\det(V^T V)}.$$

The volume of the parallelopiped can also be computed iteratively using the "base times height" formula. If ℓ_i is the length of the projection of v_i orthogonal to

span$\{v_1, \ldots v_{i-1}\}$, then the volume is precisely $\prod_{i=1}^{r} \ell_i$. In our case, this is at least γ^r, by hypothesis.

Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ be the singular values of the matrix V. Now, if $\left\|V - V^{(k)}\right\|_F^2 \leq c\Gamma$, then $\sigma_{2k+1}^2 \leq \frac{c\Gamma}{k}$. For suppose not, then:

$$\sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_{2k+1}^2 \geq (k+1)\sigma_{2k+1}^2 > c\Gamma,$$

contradicting the bound on $\left\|V - V^{(k)}\right\|_F^2$. Next, using the formula for the volume (and the fact that the determinant is the product of the eigenvalues), we have

$$\left(\prod_{i=1}^{2k} \sigma_i\right) \left(\frac{c\Gamma}{k}\right)^{(r-2k)/2} \geq \gamma^r.$$

Now, using standard convexity, we have

$$\prod_{i=1}^{2k} \sigma_i \leq \left(\frac{\sum_{i=1}^{2k} \sigma_i}{2k}\right)^{2k} \leq \left(\frac{\sum_{i=1}^{2k} \sigma_i^2}{2k}\right)^{k} \leq \left(\frac{\|V\|_F^2}{2k}\right)^{k}.$$

Combining the two equations above, we have

$$\left(\frac{\|V\|_F^2}{2c\Gamma}\right)^{k} \geq \left(\frac{\gamma^2 k}{c\Gamma}\right)^{r/2}.$$

Taking logarithms gives the desired bound. ∎

III. Online PCA

As a warm-up and to introduce our high level technique, we start with an algorithm that gives a weaker (logarithmic) error guarantee. Later in Section III-B, we will see how to improve the error bound, thereby proving Theorem I.3 (modulo the assumption $\xi \geq \left\|A - A^{(k)}\right\|_F^2$, which will be removed in Section V).

A. Logarithmic approximation algorithm

In this section, we illustrate some of the main ideas used in our algorithms. We will end up showing the following result.

Theorem III.1. *Suppose the columns of a matrix A arrive in an online manner. Let $k \geq 1$ be an integer, and let ξ be a parameter that satisfies $\xi \geq \left\|A - A^{(k)}\right\|_F^2$. Also, let L be an upper bound on the quantity $\log(\|A\|_F^2/\xi)$. Assume that k, ξ, and L are given. Then for any $\delta > 0$, there exists an algorithm, that after seeing each column $A_i \in \mathbb{R}^d$ of A, outputs a column $Y_i \in \mathbb{R}^r$, such that*

1) *$r \leq O(kL + \log(1/\delta))$.*
2) *The running time of a step is $O(rd)$ (or better, $O(r)$ times the number of non-zeros in A_i).*
3) *In the end, with probability $\geq 1 - \delta$, there exists a projection matrix V (which the algorithm maintains) such that*

$$\|A - VY\|_F^2 \leq O\left(\xi L + \frac{\xi \log(1/\delta)}{k}\right).$$

1) Outline and description of the algorithm: The algorithm maintains a matrix V of dimensions $d \times r'$, for some $r' \leq r$, whose columns are orthogonal. The space spanned by V is the "current guess" for the PCA subspace. When a new column u arrives, either the projection of u onto V (i.e. the vector $V^T u$ is returned), or depending on the error $\left\|u - V^T u\right\|$ (as well as the status of the current "phase", as we will see), a new column is added to V, and $V^T u$ is returned, for the new V. Always, the algorithm pads 0s to the r' dimensional vector, so that a vector in $\mathbb{R}^r$ is returned. The argument shows that with probability $\geq 1 - \delta$, the inequality $r' \leq r$ holds until the end of the input (and thus we do not "overflow" or need to truncate).

The algorithm (see 1 below) proceeds in phases. In each phase, we construct a vector $\mathbf{w}$ which acts as a *sketch* of all the vectors in that phase. We add a corresponding vector $\mathbf{w}'$ to V at the end of that phase.

Algorithm 1 Sampling in phases: procedure Simple-Online-PCA (k, ξ)

Input: Matrix $A \in \mathbb{R}^{d \times n}$ whose columns arrive one by one, guess ξ for the optimum error, parameter k, bound L.
Output: An embedding of columns as they arrive; basis V at the end.

1: Initialize $V = \emptyset$, and set $r = CkL$ ($C = 200$ suffices).
2: Initialize $\mathbf{w} = 0$, and running sum $\sigma = 0$.
3: **while** columns u arrive **do**
4: Compute $\Pi_V^{\perp} u = u - Vu$ (as V has orthonormal columns)
5: Let $p_u := \frac{k\left\|\Pi_V^{\perp} u\right\|^2}{256\xi}$.
6: If $p_u < 1$, do the following:
7: Increment $\sigma \leftarrow \sigma + p_u$, set $\mathbf{w} \leftarrow \mathbf{w} + \chi u$, where χ is ± 1 u.a.r.
8: If $\sigma \geq 1$, add $\mathbf{w}' := \Pi_V^{\perp}\mathbf{w} / \left\|\Pi_V^{\perp}\mathbf{w}\right\|$ to V and reset $\mathbf{w} = 0$ and $\sigma = 0$ (start new phase)
9: Else ($p_u \geq 1$), then declare phase as *special*, and do:
10: Add $\Pi_V^{\perp} u / \left\|\Pi_V^{\perp} u\right\|$ to V
11: *Also* add $\mathbf{w}' = \Pi_V^{\perp}\mathbf{w} / \left\|\Pi_V^{\perp}\mathbf{w}\right\|$ to V (unless $\mathbf{w} = 0$, in which case we do not add it)
12: Reset $\mathbf{w} = 0$ and $\sigma = 0$.
13: Return the embedding $y = Vu$, resized to dimension r by either adding zeroes or truncation.
14: **end while**
15: Output V

Description of the phases.: The columns u of the matrix arrive one by one. To each u, we assign the number p_u, that is proportional to the error in approximating u using the columns of the current V (i.e., the length of the *residual*). A phase is a collection of vectors u in which these p_u values add up to ≥ 1. (We might encounter vectors for which the expression for p_u exceeds 1; in this case, we declare the

phase as special, and treat it differently.) At the end of a phase, we add one vector $\mathbf{w}'$ that is a "sketch" of the vectors in that phase to V. (In the case of special phases, we add two vectors.) The high level idea is that by adding $\mathbf{w}'$ to V, the corresponding directions will incur a smaller error in the future. Further, we do not add a column to V unless we have built up a sufficient amount of "residual mass" (and this is used to bound the number of columns). From the definition of $\mathbf{w}'$, we observe that V is always a matrix with orthonormal columns.

2) Analysis: Let us now analyze the algorithm above, and complete the proof of Theorem III.1.

The first lemma (proof deferred to Section VII-A) is simple yet powerful. It summarizes the sense in which we require $\mathbf{w}$ to be a "proxy" for the phase.

Lemma III.2. *Let $u_1, u_2, \ldots, u_t$ be the vectors in a phase, and let $\mathbf{w}$ be the vector produced at the end of the phase. Then with probability at least $1/4$ (over the choice of the random signs), we have:*

1) $\left\|\Pi_V^{\perp}\mathbf{w}\right\|^2 \geq \frac{1}{8}\sum_{i=0}^{t}\left\|\Pi_V^{\perp}u_i\right\|^2$.
2) *If Π_k is the projection matrix orthogonal to the k-SVD space of the entire matrix A, then*

$$\|\Pi_k\mathbf{w}\|^2 \leq 16\sum_{i=1}^{t}\|\Pi_k u_i\|^2.$$

3) $\|\mathbf{w}\|^2 \leq 16\sum_{i=1}^{t}\|u_i\|^2$.

We point out that Π_k is unknown to the algorithm. However, the random choice of χ ensures that we can still argue about $\Pi_k\mathbf{w}$ (in the analysis).

Definition III.3 (Successful phases)**.** *We will say that a phase is* successful *if all the inequalities in the statement of Lemma III.2 hold. We call a phase unsuccessful otherwise.*

The rest of the argument proceeds as follows. First, we will bound the number of phases in the algorithm. This determines the embedding dimension r (which is also the number of columns of V). Interestingly, the same argument then lets us bound the total error.

Lemma III.4. *The number of special phases is at most $2k\log\frac{\|A\|_F^2}{2\xi}$.*

Proof: Let T be the matrix consisting of all the columns that made the corresponding phases special. Note that any column of T must have squared projection $\geq 4\xi/k$ orthogonal to the span of all the previous columns of T. (This is because every such column is added to the matrix V.)

We also clearly have $\|T\|_F^2 \leq \|A\|_F^2$, and $\left\|T - T^{(k)}\right\|_F^2 \leq \left\|A - A^{(k)}\right\|_F^2 \leq \xi$. Thus the hypotheses of Lemma II.1 all hold. This implies that

$$\#\text{cols}(T) \leq 2k\log\frac{\|T\|_F^2}{2\xi} \leq 2k\log\frac{\|A\|_F^2}{2\xi},$$

thus establishing the lemma. ∎

Lemma III.5. *The number of successful ordinary (i.e., non-special) phases is $\leq 2k\log\frac{\|A\|_F^2}{2\xi}$.*

Proof: The proof is similar to that of previous lemma, but we use the inequalities that characterize a successful phase. Let T be the matrix whose columns are all the vectors $\mathbf{w}$ at the end of successful ordinary phases. We note that vectors of the form $\mathbf{w}' = \Pi_V^{\perp}\mathbf{w}/\left\|\Pi_V^{\perp}\mathbf{w}\right\|$ are added to V, but at any point of time, the span of the columns of T (so far) is a subspace of the span of the columns of V.

By the third inequality in the definition of a successful phase, we have $\|T\|_F^2 \leq 16\|A\|_F^2$. Also, by the second inequality, we have $\left\|T - T^{(k)}\right\|_F^2 \leq 16\left\|A - A^{(k)}\right\|_F^2$. Thus ξ satisfies $\xi \geq (1/16)\left\|T - T^{(k)}\right\|_F^2$.

Next, by the definition of a phase (p_u values add up to ≥ 1), we have that if $u_1, u_2, \ldots, u_t$ are vectors in an ordinary phase and if V denotes the projection matrix maintained by the algorithm at the beginning of the phase, then

$$\sum_{i=1}^{t}\left\|\Pi_V^{\perp}u_i\right\|^2 \geq \frac{256\xi}{k}.$$

This means that if the phase is successful, we have $\left\|\Pi_V^{\perp}\mathbf{w}\right\|^2 \geq \frac{32\xi}{k}$. Thus the columns of T satisfy the hypotheses of Lemma II.1 with $c = 16$. This implies that

$$\#\text{cols}(T) \leq 2k\log\frac{\|T\|_F^2}{32\xi} \leq 2k\log\frac{\|A\|_F^2}{2\xi}.$$

∎

The next lemma (whose proof is postponed to Section VII-B) shows that it is very unlikely that the number of unsuccessful phases is much larger.

Lemma III.6. *For any $\delta > 0$, we have that with probability $\geq 1 - \delta$, the total number of phases is at most*

$$16\left(4k\log\frac{\|A\|_F^2}{2\xi} + \left\lceil 32\ln\frac{2}{\delta}\right\rceil\right).$$

Proof: Lemmas III.4 and III.5 upper bound the number of special and successful ordinary phases. So we only need to upper bound the number of unsuccessful phases. Lemma III.2 implies that every phase is successful with probability at least $1/4$. So we can think of each phase as a coin toss that comes head with probability at least $1/4$. Applying Lemma VII.1 with $p = 1/4$ proves that with probability $1-\delta$, the total number of phases cannot be more than $16(H + \lceil 32\ln(2/\delta)\rceil)$ where H is the number of heads, i.e. the number of successful phases. Lemmas III.4 and III.5 imply that $H \leq 4k\log\frac{\|A\|_F^2}{2\xi}$. Putting together these two upper bounds yields the aggregate upper bound on the total number of phases. ∎

This proves the bound on the dimension of the embedding produced. The next lemma bounds the total error.

Lemma III.7. *Recall that the algorithm receives the columns $\{A_i\}_{i=1}^n$. Let $V^{\langle i \rangle}$ denote the matrix V* after *the vector A_i has been processed. For any $\delta > 0$, with probability $\geq 1 - \delta$, we have*

$$\sum_{i=1}^{n} \left\| \Pi_{V^{\langle i \rangle}}^{\perp} A_i \right\|^2 \leq O\left(\xi \log \frac{\|A\|_F^2}{\xi} + \frac{\xi}{k} \log \frac{1}{\delta} \right).$$

Remark.: Note that for any A_i that is not the last vector in a phase, the $V^{\langle i \rangle}$ is the same as the matrix V at the beginning of the phase. The definition of $V^{\langle i \rangle}$ ensures that for any vector A_i for which the corresponding p_u in the algorithm is ≥ 1, we have $\Pi_{V^{\langle i \rangle}}^{\perp} A_i = 0$, i.e., we do not have to pay for it in the end (as we would have marked the phase as special and added $\Pi_V^{\perp} A_i$ to V).

Proof: Let us consider any phase, and suppose it consists of $A_i, A_{i+1}, \ldots, A_{i+t}$, for some $t \geq 0$. Let V be the matrix that is maintained by the algorithm before the start of the phase. I.e., $V = V^{\langle i-1 \rangle}$.

If the phase is special, then as we noted in the remark above, the algorithm ensures that $\Pi_{V^{\langle i+t \rangle}}^{\perp} A_{i+t} = 0$. For the previous vectors, we have

$$\sum_{j=0}^{t-1} \left\| \Pi_{V^{\langle i+j \rangle}}^{\perp} A_{i+j} \right\|^2 = \sum_{j=0}^{t-1} \left\| \Pi_V^{\perp} A_{i+j} \right\|^2 \leq \frac{256\xi}{k},$$

where the last part follows because the sum of the p_u values is smaller than 1 (else the phase would have ended).

Next, if the phase is not special, then all the p_u values are smaller than 1, and the sum of the values is ≤ 2, and thus

$$\sum_{j=0}^{t} \left\| \Pi_{V^{\langle i+j \rangle}}^{\perp} A_{i+j} \right\|^2 \leq \sum_{j=0}^{t-1} \left\| \Pi_V^{\perp} A_{i+j} \right\|^2 \leq \frac{512\xi}{k}.$$

Thus, combining this with Lemma III.6 that bounds the number of phases, the lemma follows. ∎

Now we are ready to prove Theorem III.1.

Proof of Theorem III.1: Lemmas III.6 and III.7 together show that the desired bounds on the embedding dimension r and the error hold with probability $1 - \delta$. For the running time, note that in each iteration, we simply compute the product Vu, and thus the running time is $O(dr)$. If the matrix is sparse, we only need $r \cdot nnz(A_i)$ time to process the vector A_i resulting in an $O(r \cdot nnz(A))$ running time overall. ∎

B. Online PCA: a $(1+\epsilon)$ approximation

We will now see how to combine the algorithm above with the previous work of [5] in order to obtain a $(1+\epsilon)$ approximation algorithm.

The result of this section is the following. The differences from Theorem III.1 above are the error bound, the value of r, and the running time. (See the comment after Theorem III.9 for a note on the running time.)

Theorem III.8. *Suppose the columns of a matrix A arrive in an online manner. Let $k \geq 1$ be an integer, and let ξ be a parameter that satisfies $\xi \geq \left\| A - A^{(k)} \right\|_F^2$. Also, let L be an upper bound on the quantity $\log(\|A\|_F^2 / \xi)$. Assume that k, ξ, and L are given. Then for any $\delta > 0$, there exists an algorithm, that after seeing each column $A_i \in \mathbb{R}^d$ of A, outputs a column $Y_i \in \mathbb{R}^r$, such that*

1) $r \leq O\left(\frac{k}{\epsilon^2}\right) \cdot (L + \log 1/\delta)^3$.
2) *In the end, with probability $\geq 1 - \delta$, there exists a projection matrix V (which the algorithm maintains) such that*

$$\|A - VY\|_F^2 \leq (1+\epsilon) \left\| A - A^{(k)} \right\|_F^2 + \epsilon\xi.$$

Remark.: This is almost our main result (Theorem III.8), except for the requirement of $\xi \geq \left\| A - A^{(k)} \right\|_F^2$. We remove this assumption in Section V.

1) Main idea: Our key idea for improving the bound from Section III-A is the following: suppose we run Algorithm 1 on the vectors A_i as they arrive; this outputs an embedding that captures the projection of A_i onto the current subspace spanned by V. Instead of ignoring the *residual* vector, we pass the residual to an instance of the *additive* approximation result of [5] (which we use as a blackbox). The algorithm produces an embedding of the residual vectors, and our final output is a *joint* embedding, as we will see.

Intuitively, because the total Frobenius norm of the residuals is only $O(L)\xi$ (as we proved earlier), an additive error of ϵ' times this value, for $\epsilon' \approx \epsilon/L$ suffices to obtain the desired bound.

2) Formal description of the algorithm.: In what follows, we will use the result of Boutsidis et al. [5] as a black box. Their paper presents two algorithms for online PCA. The first is an algorithm that assumes a known upper bound on the Frobenius norm of the entire matrix $\|A\|_F^2$, and achieves an additive error of $\epsilon \|A\|_F^2$. The next algorithm is faster, does not require a bound on $\|A\|_F^2$. However, a key property of the first algorithm is that it is *incremental* in the following sense: the algorithm maintains a subspace U onto which it projects the vectors A_i that it sees, and the space U only grows. In this sense, it is similar in spirit to Algorithm 1. This property does not hold for the second (and main) algorithm of [5]. The property is crucial for our argument, so now we state the theorem corresponding to their first algorithm.

Theorem III.9 (Theorem 1 in [5] restated)**.** *Given an input matrix $A \in \mathbb{R}^{d \times n}$, a parameter $\epsilon > 0$ and an upper bound Γ on $\|A\|_F^2$, there exists an algorithm for online PCA that upon seeing a vector A_i, outputs an embedding $y_i \in \mathbb{R}^\ell$, where $\ell = O(k/\epsilon^2)$. In the end, one has the following guarantee on the error:*

$$\min_{\Phi} \|A - \Phi Y\|_F^2 \leq \left\| A - A^{(k)} \right\|_F^2 + \epsilon\Gamma.$$

Furthermore, at every time i, the algorithm maintains a matrix $U^{\langle i \rangle}$ with d rows and orthonormal columns. $U^{\langle i \rangle}$ is only incremented (by way of adding new columns) as the algorithm proceeds. The embedding y_i of the vector A_i is precisely $(U^{\langle i \rangle})^T A_i$. Also, one has the stronger guarantee that

$$\sum_i \left\| A_i - U^{\langle i \rangle} y_i \right\|^2 \leq \left\| A - A^{(k)} \right\|_F^2 + \epsilon\Gamma.$$

Remarks.: We note that the "furthermore" parts of the statement follow immediately from the proofs in [5]. Also, the statement of Theorem 1 in their paper has an additional restriction of $\|A_i\|^2 \leq \Gamma/\ell$. But by simply adding all such columns to the U (as also noted in Section 4 of their paper), this assumption can be removed. We also remark that one can use any *incremental* algorithm for online-PCA that has an additive guarantee in the place of the theorem above.

We now formally describe the algorithm. The procedure OPCA-Add refers to the additive approximation of [5].

Algorithm 2 Online PCA: a $(1+\epsilon)$ approximation

Input: Matrix $A \in \mathbb{R}^{d \times n}$ whose columns arrive one by one, parameters ξ and L as in the statement of the theorem, parameters k and ϵ. Procedure OPCA-Add refers to the algorithm from Theorem III.9.

Output: An online low-dimensional embedding.

1: Initialize matrices $V', V'' = \emptyset$ (i.e., $d \times 0$ matrices, as in Algorithm 1)
2: Set output dimension $\ell = O(k'/\epsilon'^2) + O(kL + \log 1/\delta)$, where the first term is from Theorem III.9 and Eq (3), and the second from Theorem III.1.
3: **while** columns A_i arrive **do**
4: Execute Algorithm 1 with input A_i; this updates V' and outputs $y'_i = (V')^T A_i$
5: Now execute a step of OPCA-Add$\left(\Pi^{\perp}_{V'} A_i, k', \epsilon', \Gamma\right)$, where k', ϵ', Γ are defined in (3); this updates V'', and outputs $y''_i = (V'')^T (\Pi^{\perp}_{V'} A_i)$
6: Let W be an orthonormal basis for span$(V' \cup V'')$
7: Output the vector $W^T A_i$
8: **end while**

The values of k' and ϵ' we use are the following:

$$k' = 20k(L + \log(1/\delta)) \; ; \quad \epsilon' = \frac{\epsilon}{C(L + \log(1/\delta))}$$
$$\Gamma = C\xi(L + \log(1/\delta)), \tag{3}$$

where C is the constant in the $O()$ term in the error bound of Theorem III.1. We remark that it is important to pick k' larger than k (we will see why in the proof).

3) Analysis: *Proof of Theorem III.8:* Let us start by introducing some notation. Let $r'_{\langle i \rangle}$ denote the residual for the vector A_i from Algorithm 1. Using the notation in the algorithm, we have $r'_{\langle i \rangle} := \Pi^{\perp}_{V'} A_i$ (where V' is the matrix maintained after the ith call to Algorithm 1). Note that $r'_{\langle i \rangle}$ is precisely the input to OPCA-Add in the ith iteration. Let R' denote the matrix whose ith column is $r'_{\langle i \rangle}$. Likewise, let $r''_{\langle i \rangle}$ denote the residual we obtain after OPCA-Add. I.e., $r''_{\langle i \rangle} = r'_{\langle i \rangle} - V'' y''_i$.

A simple consequence of Theorem III.1 is that for our choice of Γ,

$$\sum_i \left\| r'_{\langle i \rangle} \right\|^2 \leq \Gamma, \text{ with probability } \geq 1 - \delta.$$

This means that we can (with probability $1-\delta$) apply the error guarantee from Theorem III.9, and obtain:

$$\sum_i \left\| r''_{\langle i \rangle} \right\|^2 \leq \left\| R' - (R')^{(k')} \right\|_F^2 + \epsilon'\Gamma.$$

Next, we argue that $\left\| R' - (R')^{(k')} \right\|_F^2 \leq \left\| A - A^{(k)} \right\|_F^2$. This is simply because each column $r'_{\langle i \rangle} = A_i - V' y'_i$, and V' at any time contains at most $16k(L + \log(1/\delta))$ columns, from the guarantee of Theorem III.1. Thus the rank-k' approximation to the matrix R' has error at most the rank-k approximation of A_i. (This is because we could "cancel out" the contributions due to the columns of V', if necessary.) This implies that

$$\sum_i \left\| r''_{\langle i \rangle} \right\|^2 \leq \left\| A - A^{(k)} \right\|_F^2 + \epsilon\xi.$$

We note that based on Equation 3, $\epsilon\xi$ is equal to $\epsilon'\Gamma$ which justifies the replacement of the last term in the above inequality. As the final step, we claim that $\left\| A_i - W^T A_i \right\| \leq \left\| r''_{\langle i \rangle} \right\|$. This is because by definition,

$$r''_{\langle i \rangle} = r'_{\langle i \rangle} - V'' y''_i = A_i - V' y'_i - V'' y''_i.$$

This is simply the difference of A_i and a linear combination of the vectors $V' \cup V''$. Thus the length of $r''_{\langle i \rangle}$ is upper bounded by the distance of A_i to span$(V' \cup V'')$, which is precisely $A_i - W^T A_i$.

This completes the proof of the error bound of Theorem III.8. The bound on the number of columns follows because the dominating term is $O(k'/\epsilon'^2)$ from Theorem III.9 (which is the number of columns of V''). Plugging in the values from Eq (3) completes the proof of Theorem III.8. ∎

Remark.: Given the above proof, it may be tempting to ask if one can just apply Theorem III.9 repeatedly, chaining it "in series" with itself to obtain our result. Unfortunately there are few issues with this approach, first the algorithm would be significant more complex, use a larger number of columns and have worst running time. Second one would need to prove that the rank-k approximation error of the residual is not increasing and it is not clear if this is true in our online setting.

IV. Online column subset selection

We next present our algorithms for the online column subset selection problem. Similar to the case of PCA, we first give an algorithm that achieves a poly-logarithmic error bound, and then we use known additive approximation ideas to improve the error bound to $(1+\epsilon)$.

PCA vs CSS.: Before going into the details, note that the first issue we face in adapting our earlier ideas is that the matrix V that is maintained (e.g. in Algorithm 1) has columns that are linear combinations of the columns of A. This is not allowed in the CSS problem. However, unlike online PCA, we do not need to commit to a low dimensional embedding of each column as it arrives. We simply need to make a decision of whether to keep or discard the column. We only need to ensure that *in the end*, the selected columns provide a good approximation to the entire matrix.

A. Logarithmic approximation for CSS

We show now how to adapt Algorithm 1 for the column selection problem. The result we show here is the following.

Theorem IV.1. *Suppose the columns of a matrix A arrive in an online manner. Let $k \geq 1$ be an integer, and let ξ be a (given) parameter that satisfies $\xi \geq \left\|A - A^{(k)}\right\|_F^2$. Then for any $\delta > 0$ there exists an algorithm, that after seeing each column A_i, decides to either ignore it, or add it to a selection set S (which starts off empty), with the following properties:*

1) *In the end, $|S| \leq O\left(k \log \frac{\|A\|_F^2}{\xi} + \log(1/\delta)\right)$.*
2) *The running time of a step is $O(|S|d)$ (or $O(|S|)$ times the number of non-zeros in A_i).*
3) *In the end, with probability $\geq 1-\delta$, we have*

$$\left\|\Pi_S^{\perp} A\right\|_F^2 \leq \xi \cdot O\left(\log \frac{\|A\|_F^2}{\xi} + \frac{\log(1/\delta)}{k}\right).$$

Remark.: As noted in the introduction, we do not need to assume the knowledge of a bound on $\log \|A\|_F^2/\xi$. Indeed, we can even remove the lower bound on ξ, as we will see in Section V.

1) Outline and description of the algorithm: The algorithm is similar in structure to Algorithm 1. It maintains a subspace (with basis V) which now corresponds to the span of the columns S selected so far. The algorithm again proceeds in phases, where columns are collected until there is sufficient "residual mass". The problem now is that we cannot add a random signed combination of the columns in the phase to S (because S is supposed to be a subset). However, the key idea is to show that adding a random *subset* of the vectors in the phase acts as a good enough proxy for the vectors in the phase, with high probability! Furthermore, this subset can be chosen as the phase proceeds (i.e., in an online manner). The process turns out to be equivalent to sampling based on the residual norm, specifically the norm orthogonal to the span of the columns selected *at the beginning* of the phase.

Algorithm 3 gives the formal description.

Algorithm 3 Sampling proportional to residual norm

Input: Matrix $A \in \mathbb{R}^{d \times n}$ whose columns arrive one by one, parameter ξ.
Output: A subset S of the columns.

Initialize $S_{pre} = \emptyset$, $S_{current} = \emptyset$ and running sum $\sigma = 0$.
while columns u arrive **do**
  Let $p_u := \frac{k\left\|\Pi_{S_{pre}}^{\perp}(u)\right\|^2}{160\xi}$.
  With probability $\min(p_u, 1)$, add u to $S_{current}$ (i.e., decide to pick u).
  If $p_u < 1$, do the following:
    Increment $\sigma \leftarrow \sigma + p_u$.
    If $\sigma \geq 1$, set $S_{pre} := S_{pre} \cup S_{current}$ and reset $\sigma = 0$ and $S_{current} = \emptyset$ (start new phase)
  Else ($p_u \geq 1$), then declare phase as *special*, and do:
    Set $S_{pre} := S_{pre} \cup S_{current}$, and reset $\sigma = 0$ and $S_{current} = \emptyset$.
end while
Output $S_{pre} \cup S_{current}$.

Description.: The arriving sequence of columns is partitioned into phases as in Algorithm 1 (based on the total residual mass crossing a threshold). When a column u arrives, we assign a sampling probability p_u, that is proportional to the error in approximating u using the span of the selected columns in all previous phases and not the current phase. The set S_{pre} denotes the set of all selected columns in previous phases, and the set $S_{current}$ denotes the set of selected columns in the current phase. At the end of a phase, $S_{current}$ is appended to S_{pre}.

2) Analysis: Algorithm 3 differs from Algorithm 1 since it selects columns instead of linear combinations of them. However, the analysis shares the same structure: we define a notion of *success* for phases. Then we show that every ordinary phase is successful with at least a constant probability. This lets us conclude that the number of ordinary phases cannot be too large. A more direct argument shows that the number of special phases cannot be too large, which then completes the proof.

The key new notion here is the definition of a successful phase. This is based on the following simple yet important observation about the algorithm:

Observation 1. *The definition of a phase (i.e., which vectors constitute the phase) is independent of $S_{current}$. It only depends on the set S_{pre} at the end of the previous phase.*

Definition IV.2. *Fix a specific phase. Let $u_1, u_2, \ldots, u_t$ be the columns that arrived in that phase, and let $S_{current}$ be*

the columns chosen in the phase. The phase is said to be successful *if there exists a* $\mathbf{w} \in$ *span*$(S_{current})$ *such that:*

1) *For the projection* $\Pi_{S_{pre}}^{\perp}$,
$$\left\|\Pi_{S_{pre}}^{\perp}\mathbf{w}\right\|^2 \geq \frac{1}{2}\sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2.$$
2) *For the projection* Π_k*, the projection* orthogonal to *the span of the top* k *singular vectors of the entire matrix* A*,*
$$\|\Pi_k\mathbf{w}\|^2 \leq 40\sum_i \|\Pi_k u_i\|^2.$$
3) *For the vector itself,*
$$\|\mathbf{w}\|^2 \leq 40\sum_i \|u_i\|^2.$$

Remark.: Recall that in the analysis in Section III-A, we defined $\mathbf{w}$ as a random linear combination of *all the vectors* in the phase, and the combination was explicitly maintained. Here, we only require the *existence* of such a $\mathbf{w}$. The lemma is as before.

Lemma IV.3. *An ordinary (non-special) phase is successful with probability* $\geq 1/80$ *(over the choice of* $S_{current}$*).*

Proof: Let $u_1, \ldots, u_t$ be the columns in the phase. Let Y_i be a random variable that indicates if $u_i \in S_{current}$ or not. From the definition of the algorithm, the Y_i are independent Bernoulli random variables, and $\Pr[y_i = 1] = p_{u_i}$. Let us abuse notation slightly and write $p_i = p_{u_i}$ for $1 \leq i \leq t$.

Now, let χ_i be a uniformly random and independent ± 1 sign for column u_i. The idea is to consider the vector
$$\mathbf{w} = \sum_{i=1}^{t} \chi_i Y_i \hat{u}_i, \tag{4}$$
where $\hat{u}_i$ is the "re-scaled" u_i; i.e., $\hat{u}_i = u_i/\sqrt{p_i}$.

Now, let us compute the probability that a phase is successful. The second and third requirements are easy applications of Markov's inequality. Consider the second one. We have $\Pi_k w = \sum_i \chi_i Y_i \Pi_k u_i/\sqrt{p_i}$. Thus
$$\begin{aligned}\mathbb{E}\left[\|\Pi_k\mathbf{w}\|^2\right] &= \sum_i \mathbb{E}[Y_i^2]\frac{\|\Pi_k u_i\|^2}{p_i} + \\ &\quad \sum_{i\neq j} 2\mathbb{E}[\chi_i\chi_j]\mathbb{E}[Y_iY_j]\frac{\langle \Pi_k u_i, \Pi_k u_j\rangle}{\sqrt{p_ip_j}} \\ &= \sum_i \|\Pi_k u_i\|^2.\end{aligned}$$
This holds because $\mathbb{E}[\chi_i\chi_j]$ is zero for any $i \neq j$, and also $\mathbb{E}[Y_i^2] = E[Y_i] = p_i$ since Y_i is an indicator variable. Thus by Markov's inequality, the second inequality in Definition IV.2 holds with probability $\geq 39/40$. The exact same computation (without Π_k) shows the same bound for the third inequality. Thus parts (2) and (3) in Definition IV.2 hold *together* with probability $\geq 19/20$.

Let us thus compute the probability of part (1). First, as above, we have:
$$\mathbb{E}\left[\left\|\Pi_{S_{pre}}^{\perp}\mathbf{w}\right\|^2\right] = \sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2. \tag{5}$$
The idea now is to apply the Paley-Zygmund inequality (that $\Pr[Z > \frac{1}{2}\mathbb{E}[Z]] \geq \frac{1}{4}\frac{\mathbb{E}[Z]^2}{\mathbb{E}[Z^2]}$), for $Z := \left\|\Pi_{S_{pre}}^{\perp}\mathbf{w}\right\|^2$. For this, we need to compute $\mathbb{E}[Z^2]$.

Note that any term with an χ_i with an odd power has expected value zero. We also note that $\mathbb{E}[Y_i^2] = \mathbb{E}[Y_i]$ given it is an indicator random variable. Using these, we have:
$$\begin{aligned}&\mathbb{E}[Z^2] = \\ &\sum_i p_i \left\|\Pi_{S_{pre}}^{\perp}\hat{u}_i\right\|^4 + \binom{4}{2}\sum_{i\neq j} p_ip_j\langle \Pi_{S_{pre}}^{\perp}\hat{u}_i, \Pi_{S_{pre}}^{\perp}\hat{u}_j\rangle^2 \\ &\leq \sum_i \frac{\left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^4}{p_i} + 6\sum_{i\neq j}\left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2\left\|\Pi_{S_{pre}}^{\perp}u_j\right\|^2. \end{aligned}\tag{6}$$
where the inequality is implied by Cauchy–Schwarz inequality.

We need to upper bound the right side of Equation 6 in terms of square of right side of Equation 5. The second term above is not more than three times larger than square of right side of Equation 5. The first term is a bit harder to upper bound. We recall that, since the phase is ordinary, $p_i = \frac{k}{\xi}\left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2$. So we have:
$$\sum_i \frac{\left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^4}{p_i} = \frac{\xi}{k}\sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2$$
Since the summation is over the columns of a phase, we have $\sum_i p_i \geq 1$ or equivalently $\sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2 \geq \xi/k$. This means:
$$\frac{\xi}{k}\sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2 \leq \left(\sum_i \left\|\Pi_{S_{pre}}^{\perp}u_i\right\|^2\right)^2$$
So we have the desired upper bound on the first term as well. We conclude that $\mathbb{E}[Z^2]$ is at most 4 times larger than $\mathbb{E}[Z]^2$. Plugging this into Paley-Zygmund inequality, we have $\Pr[Z > (1/2)\mathbb{E}[Z]] \geq \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$ which implies the second claim of this lemma. ∎

The rest of the argument is very similar to that in Section III-A (with slightly differing constants). We defer the lemmas and proofs to Section VII-C. The main steps are: we first bound the number of special phases and the number of successful ordinary phases using Lemma II.1; next, we use the fact that an ordinary phase is successful w.p. $\Omega(1)$ (from Lemma IV.3) to conclude that the total number of phases cannot be too large. As before, this also implies a bound on the total error, thus establishing Theorem IV.1.

In the next section we show how to combine this result with previous work to obtain a $(1+\epsilon)$ multiplicative approximation to recover either the embedding or the subspace.

B. $(1+\epsilon)$ approximation for CSS

As we did for online PCA, we improve the basic algorithm from Section IV-A to obtain the following result.

Theorem IV.4. *Suppose the columns of a matrix A arrive in an online manner. Let $k \geq 1$ be an integer, and let ξ be a (given) parameter that satisfies $\xi \geq \left\|A - A^{(k)}\right\|_F^2$. Then for any $\epsilon > 0$ there exists an algorithm, that after seeing each column A_i, decides to either ignore it, or add it to a selection set S (which starts off empty), with the following properties:*

1) *In the end, $|S| \leq O(k/\epsilon^2) \cdot \left(\log \frac{\|A\|_F^2}{\xi}\right)$.*
2) *The running time of a step is $O(|S|d)$ (or $O(|S|)$ times the number of non-zeros in A_i).*
3) *In the end, with probability $\geq 3/4$, we have*

$$\left\|\Pi_S^{\perp} A\right\|_F^2 \leq (1+\epsilon)\left\|A - A^{(k)}\right\|_F^2 + \epsilon\xi.$$

Here again, the high level idea is taking the residual vectors from Algorithm 3 and running an additive approximation algorithm. In this case, instead of using the algorithm from [5] (which does not select columns), we use the classic norm sampling result of Frieze, Kannan and Vempala [15].[1] The problem with carrying this out exactly as in Section III-B is that we do not assume knowledge of the value of L (which was used to define k' and ϵ', the parameters for the second procedure). We thus need a slightly more careful argument, which we now discuss.

Remark.: Since we are using norm sampling, we have a running time that is much better than the one obtained in Section III-B by using the algorithm of [5].

1) Description of the algorithm: In what follows let us denote by $\text{FKV}(k, \epsilon, \Gamma)$ the norm-sampling procedure of [15] that takes as input a matrix A, and samples each column A_i independently with probability $\frac{k}{\epsilon} \cdot \frac{\|A_i\|^2}{\Gamma}$. The guarantee of [15] (stated more generally below) is that if $\Gamma = \|A\|_F^2$, then the chosen set of vectors can be used to approximate A up to an error of $\|A - A_k\|_F^2 + \epsilon \|A\|_F^2$.

Our procedure is described formally in Algorithm 4.

2) Analysis: The main trick in the analysis is to move to a slightly different algorithm (one that can be viewed as a "two-pass" algorithm) and analyze that instead. Observe that in line 3 of the algorithm, we do not use S_{FKV} in any way, and thus it proceeds precisely as in Algorithm 3. Let S_{first} denote the final value (after we have seen all the columns) of $S_{pre} \cup S_{current}$. Now, consider replacing the probability of adding A_i to S_{FKV} (line 5) to $\min\left(1, \frac{20k}{\epsilon} \cdot \frac{\left\|\Pi_{S_{\text{first}}}^{\perp} A_i\right\|^2}{\xi}\right)$. This is no longer an online algorithm, but we will use it only for the sake of analysis.

[1] It is interesting to ask why this cannot be done in the case of PCA. The reason is that norm sampling has a guarantee on the projection error that only holds *after all the columns are chosen*. In online PCA, we need to provide an embedding as the column arrives; this is not possible (at least directly) using norm sampling.

Algorithm 4 Online CSS: a $(1+\epsilon)$ approximation

Input: Matrix $A \in \mathbb{R}^{d \times n}$ whose columns arrive one by one, parameters k, ξ.

Output: An online decision *keep* or *discard* for each point.

1: Initialize sets $S_{pre}, S_{current} = \emptyset$ as in Algorithm 3, initialize $S_{\text{FKV}} = \emptyset$
2: **while** columns A_i arrive **do**
3: Execute Algorithm 3 with input A_i; this updates S_{pre} and $S_{current}$
4: Let $r_{\langle i \rangle}$ be the residual $r_{\langle i \rangle} = \Pi_{S_{pre}}^{\perp} A_i$
5: Now add A_i to S_{FKV} with probability $\min\left(1, \frac{20k}{\epsilon} \cdot \frac{\left\|r_{\langle i \rangle}\right\|^2}{\xi}\right)$
6: If A_i was added to any of the sets maintained, we keep it otherwise, discard it.
7: **end while**

Now, there is a smaller probability of adding each A_i to S_{FKV} (and further, this choice is never again used in the algorithm). Thus the error incurred in Algorithm 4 is upper bounded by the error in this variant. We view the new algorithm as one that has two "passes" over the data. The first pass only executes step 3 and produces S_{first}. The second one samples each i with probability as described above.

For convenience, let us define

$$s_{\langle i \rangle} := \Pi_{S_{\text{first}}}^{\perp} A_i,$$

By the discussion above, $\left\|s_{\langle i \rangle}\right\| \leq \left\|r_{\langle i \rangle}\right\|$. The advantage with this change in algorithm is that we can now use the following result of Frieze, Kannan and Vempala [15] (the version below is a mild variant of Theorem 2.1 of [12].

Lemma IV.5. *Let $c > 1$, $k \geq 1$, and let $A \in \mathbb{R}^{n \times m}$ be a matrix. let S_1 be a subset of columns of A. Suppose we choose each column A_i independently with probability*

$$p_i \geq c \cdot \frac{\left\|\Pi_{S_1}^{\perp} A_i\right\|^2}{\sum_{j \in [m]} \left\|\Pi_{S_1}^{\perp} A_j\right\|^2}, \tag{7}$$

and let S_2 be the set of chosen columns.Then w.p. at least $9/10$, we have

$$\left\|\Pi_{S_1 \cup S_2}^{\perp} A\right\|_F^2 \leq \left\|A - A^{(k)}\right\|^2 + \frac{20k}{c}\left\|\Pi_{S_1}^{\perp} A\right\|_F^2.$$

As the statement is a slight variant, we include the proof in Appendix VII-D. We use the lemma to bound the error after the second pass (which is an upper bound on the error in Algorithm 4).

In order to apply the lemma, we need to ensure that the condition (7) is satisfied. Let Z denote the quantity $\sum_i \left\|s_{\langle i\rangle}\right\|^2$. Then by definition,

$$p_i = \frac{20k}{\epsilon} \cdot \frac{\left\|s_{\langle i\rangle}\right\|^2}{\xi} = \left(\frac{20kZ}{\epsilon\xi}\right) \frac{\left\|s_{\langle i\rangle}\right\|^2}{Z},$$

which satisfies (7) for $c = \frac{20kZ}{\epsilon\xi}$.

Thus with probability at least $9/10$ (over only the second pass), denoting by S' the set of vectors chosen in the second pass,

$$\left\|\Pi^{\perp}_{S_{\text{first}} \cup S'} A\right\|_F^2 \leq \left\|A - A^{(k)}\right\|_F^2 + \epsilon\xi.$$

This completes the proof of the error guarantee in Theorem IV.4. (Note that for this part, we have not used any property of Z.)

Next, consider the expected size of the final set. This time, we need to analyze Algorithm 4 and *not* the two-pass algorithm above (because the original algorithm has more columns!). We have that

$$\mathbb{E}[|S_{\text{FKV}}|] = \sum_{i=1}^{n} \frac{20k}{\epsilon} \frac{\left\|r_{\langle i\rangle}\right\|^2}{\xi}.$$

By the bound on the total error from Theorem IV.1, we have that with probability $\geq 19/20$,

$$\sum_i \left\|r_{\langle i\rangle}\right\|^2 \leq O\left(\xi \log \frac{\|A\|_F^2}{\xi}\right).$$

Thus the above together with Markov's inequality, we have that with probability $\geq 9/10$,

$$|S_{\text{FKV}}| \leq O\left(\frac{k}{\epsilon^2} \log \frac{\|A\|_F^2}{\xi}\right).$$

As the bound on the running time of each step is trivial, this completes the proof of Theorem IV.4.

V. Removing the lower bound on ξ

Our algorithms have all assumed that the target error ξ satisfies the bound $\xi \geq \left\|A - A^{(k)}\right\|_F^2$. We now show a general way to remove this assumption, at the expense of an additional factor of $\log \frac{\left\|A-A^{(k)}\right\|_F^2}{\xi}$.

A. Online PCA

Let us first consider the online PCA problem, and show Theorem I.3. We do this in two steps. In what follows, let us denote $\Psi = \max\{\xi, \left\|A - A^{(k)}\right\|_F^2\}$, and $L_\delta = L + \log(1/\delta)$.

1) We show an analog of Theorem III.1 in which we have no assumption on ξ (there is still the assumption of a known $L \geq \log \frac{\|A\|_F^2}{\xi}$), where the error is $O(\Psi L_\delta)$, and the dimension of the embedding is $O(kL_\delta^2)$.
2) Next, we use the reasoning from Section III-B (in which we run the residuals through an instantiation of the algorithm from [5]) and output the overall embedding.

The first step turns out to be the challenging one; the second is almost identical to Section III-B.

1) Logarithmic approximation: The main idea for the first step is as follows. We start with the given value of ξ, and run Algorithm 1. If the number of phases exceeds kL_δ, we conclude that ξ is too small, and double ξ. The previous analysis gives us that once ξ is above $\left\|A - A^{(k)}\right\|_F^2$, we will no longer exceed the bound on the number of phases, with probability $\geq 1-\delta$. The number of doublings needed is $\log \frac{\left\|A-A^{(k)}\right\|_F^2}{\Psi}$. This is trivially upper bounded by L (and so we keep the output dimension $O(kL^2)$). (See Algorithm 5 for the details.)

Algorithm 5 Online PCA without the assumption on ξ

Input: Matrix $A \in \mathbb{R}^{d\times n}$ whose columns arrive one by one, parameters k and ξ (arbitrary), and upper bound L on $\log \frac{\|A\|_F^2}{\xi}$.
Output: A low dimensional embedding of each point as it arrives

Initialize sets $V_{old} = \emptyset$, $V = \emptyset$
while columns A_i arrive **do**
  Execute Algorithm 1 with input $\Pi^{\perp}_{V_{old}} A_i$ (and the current ξ); this updates V
  Output the embedding $V_{old}^T A_i$ concatenated with $V^T A_i$.
  if (number of phases (i.e., dimension of V) exceeds $12kL_\delta$) **then**
    set $V_{old} \leftarrow V_{old} \cup V$; set $V = \emptyset$, $\xi \leftarrow 2\xi$
  end if
  If the number of columns in $V_{old} \cup V$ exceeds $12kL_\delta^2$ output FAIL
end while

Observation 2. *We have the following observations about the procedure.*

1) *At any point in the execution, columns of V are all orthogonal to columns of V_{old}.*
2) *Consider any suffix $\{A_i, A_{i+1}, \ldots, A_n\}$. Let $\{B_i, B_{i+1}, \ldots, B_n\}$ denote the projections $B_j = \Pi^{\perp}_T$ for any subspace T. Denote the matrices with these columns as A' and B'. Then $\left\|A' - (A')^{(k)}\right\|_F^2 \geq \left\|B' - (B')^{(k)}\right\|_F^2$.*
3) *Suppose the algorithm ends with a particular value ξ. The total residual error so far is bounded by $O(kL_\delta\xi)$.*

Proof: Part 1 follows immediately because we only pass the residual vectors to Algorithm 1.

Part 2 is also an easy consequence of the fact that the rank-k error only reduces upon projection. The key requirement though is that we have the same T for the entire suffix.

To show part 3, suppose that the *initial* value of ξ is ξ_0, and suppose that $\xi = 2^j\xi_0$ (i.e., the doubling, step 6, occurred j times). Because of the way the residual error relates to the phases, and since the number of phases is capped at $12kL_\delta$ for each value of ξ, we conclude that the total residual error is at most

$$12kL_\delta\left(\frac{2\xi_0}{k}+\frac{2^2\xi_0}{k}+\cdots+\frac{2^{j+1}\xi_0}{k}\right)\leq O\left(L_\delta\xi\right).$$

■

Finally, using part 2 of the observation, along with Theorem III.1, we have that once ξ exceeds $\left\|A-A^{(k)}\right\|_F^2$, the algorithm does not increase ξ anymore, and it ends up with the desired guarantees. This establishes the desired analog of Theorem III.1 (with a bound of $O(kL_\delta)^2$ on the embedding dimension).

2) Improving to $(1+\epsilon)$*:* The main observation is that Algorithm 5 is also *incremental* in the sense of Section III-B. I.e., we maintain a subspace V (in this case $V_{old}\cup V$), we always output the projection onto this space, and in the end have an $O(L_\delta)$ approximation to the error.

We can thus combine it with Algorithm 1 from [5], with the following parameters (as Algorithm 2):

$$k'=12kL_\delta^2\ ;\ \epsilon'=\frac{\epsilon}{CL_\delta}\ ;\ \Gamma=C\xi L_\delta.$$

Thus the number of columns used overall is $O(k'/\epsilon'^2)=O\left(\frac{kL_\delta^4}{\epsilon^2}\right)$. This establishes Theorem I.3.

B. Online CSS

In the case of online PCA, the key reason the doubling procedure worked is because we could determine when *too many columns* were chosen. This allowed us to double the value of ξ. The problem now is that we do not have an upper bound L on $\log\frac{\|A\|_F^2}{\xi}$.

Idea, outline of argument.: The main idea is to replace the upper bound on the number of phases in step 6 with a quantity that only depends on the Frobenius norm of the matrix *until now*. Let $F=\max\{\xi,\left\|A_{\{1,2,\ldots,i\}}\right\|_F^2\}$. Then we decide to double ξ if the number of selected columns in the current epoch exceeds

$$O(k\log(\frac{\|A\|_F^2}{\xi})+\log(1/\delta)),\tag{8}$$

where the constant in $O()$ comes from Corollary VII.5. We define an epoch a period of time in the algorithm that the value of ξ is fixed. So every time we double ξ a new epoch starts. The logarithm term is always upper bounded by $\log\frac{\|A\|_F^2}{\xi}$ (assuming $\xi\leq\|A\|_F^2$).

Thus, we are now more aggressive about doubling ξ. This is not a problem if $\xi\leq\left\|A-A^{(k)}\right\|_F^2$ (in this case, we do not care if we "incorrectly" decide to double ξ). Thus, consider the first time we have $\xi\geq\left\|A-A^{(k)}\right\|_F^2$. In this case, we wish to analyze the probability on an incorrect doubling of ξ. Corollary VII.5 implies that probability of starting a new epoch in this case is at most δ. So with probability $1-\delta$, we do not start a new epoch in this regime. So the total number of epochs is upper bounded by $\log\frac{\max\{\xi,\left\|A-A^{(k)}\right\|_F^2\}}{\xi}$. Given the upper bound on the number of columns we pick in each epoch in Equation 8, we can upper bound the total number of phases and also selected columns.

The error can be bounded using the argument from Observation 2 (part 3). The error bound we obtain now is $O\left[\xi\left(\log\frac{\|A\|_F^2}{\xi}+\frac{\log(1/\delta)}{k}\right)\log\frac{\max\{\xi,\left\|A-A^{(k)}\right\|_F^2\}}{\xi}\right]$. We bound this by $O(\xi L(L+\log(1/\delta)))$, where $L=\log\frac{\|A\|_F^2}{\xi}$ as before.

1) Improving to a $(1+\epsilon)$ *approximation:* We can now mimic the argument of Section IV-B. In fact, the parameters to the procedure FKV do not even need to change (this is because the parameters are not important for bounding the error, as we saw). The bound on the number of columns now becomes (since the success probability needed is a constant)

$$O\left(\frac{k}{\epsilon}\right)\cdot L^2.$$

VI. Tightness of our results

It is natural to ask if the bounds we obtain on the residual based sampling performed in our algorithms are optimal. We now give an example in which the term $\log\frac{\|A\|_F^2}{\xi}$ is almost unavoidable for the algorithms we propose.

Lemma VI.1. *Let $k=1$, and let $t,z>2$ be parameters that will be fixed shortly. There exists a matrix A of dimensions $t\times t$, such that $\|A\|_F^2=z^{2t}$, the rank-1 approximation error $\leq 2t^2/z^2$, and further, such that every column of A has a squared projection at least 1 orthogonal to the previous columns.*

Remark.: Let us set $z=2t$, and $\xi=1$. Now we know that $\left\|A-A^{(1)}\right\|_F^2\leq\xi$. If we run an algorithm that samples the columns (as they arrive) with probability $\min\left(1,\|\Pi^\perp A_i\|^2/\xi\right)$ (as in our algorithms, $\Pi^\perp$ is the projection orthogonal to the chosen columns), this algorithm will indeed pick all the columns. Thus if L denotes $\log\frac{\|A\|_F^2}{\xi}$ as before, the algorithm is choosing $\Omega(kL/\log L)$ columns. This matches the upper bound up to a $\log L$ factor.

Proof: The matrix we choose is the following (think of $z>2$):

$$M=\begin{pmatrix}1 & z & z^2 & \cdots & z^{t-1}\\ 0 & 1 & z & \cdots & z^{t-2}\\ & & \cdots & & \\ 0 & 0 & \cdots & 0 & 1\end{pmatrix}$$

It is easy to see that each column has a length 1 orthogonal to the previous columns. The bound on the Frobenius norm is also easy to check (as $z>2$).

We claim that if $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_t > 0$ are the singular values of M, then $\sigma_2^2 \leq \frac{2t}{z^2}$. Note that this immediately implies that $\left\|A - A^{(1)}\right\|_F^2 < \frac{2t^2}{z^2}$.

Bounding σ_2.: We demonstrate an explicit subspace S of dimension $t-1$ such that

$$\max_{x \in S, \|x\|=1} \|Mx\|^2 \leq \frac{2t}{z^2}.$$

By the min-max characterization of singular values, this implies the desired claim. Now define

$$S = \{x \in \mathbb{R}^t : \frac{x_1}{z^{t-1}} + \frac{x_2}{z^{t-2}} + \cdots + x_t = 0\}.$$

Take any unit vector $x \in S$. Consider the ith coordinate of Mx. This is precisely

$$(Mx)_i = x_i + \frac{x_{i+1}}{z} + \cdots + \frac{x_t}{z^{t-i}} - \left(\frac{x_{i-1}}{z} + \frac{x_{i-2}}{z^2} + \cdots + \frac{x_1}{z^{i-1}}\right),$$

where we used the definition of $x \in S$. Thus by Cauchy-Schwartz, we have that

$$(Mx)_i^2 \leq t\left(\frac{x_{i-1}^2}{z^2} + \frac{x_{i-2}^2}{z^4} + \cdots + \frac{x_1^2}{z^{2i-2}}\right).$$

Thus, since $z > 2$, the geometrically decreasing terms are negligible, and thus we have $\sum_i (Mx)_i^2 \leq \frac{2t}{z^2}\sum_i x_i^2$, thus proving the desired bound on σ_2^2. ∎

VII. Proofs of technical lemmas

A. Proof of Lemma III.2

In this section we present the proof of Lemma III.2.

Proof:

By definition, $\mathbf{w}$ is $\sum_{i=1}^t \chi_i u_i$ where χ_i is the uniformly at random sign for vector u_i. Consequently, we have $\Pi_V^\perp \mathbf{w} = \sum_{i=1}^t \chi_i \Pi_V^\perp u_i$. Define random variable Z to be $\left\|\Pi_V^\perp \mathbf{w}\right\|^2$. By the linearity of expectation, we have:

$$\begin{aligned}\mathbb{E}[Z] &= \sum_{1 \leq i,j \leq t} \mathbb{E}[\chi_i \chi_j] \left\|\Pi_V^\perp u_i\right\| \cdot \left\|\Pi_V^\perp u_j\right\| \\ &= \sum_{i=1}^t \left\|\Pi_V^\perp u_i\right\|^2, \end{aligned} \tag{9}$$

where the second equality holds because $\mathbb{E}[\chi_i \chi_j]$ is zero for any $i \neq j$ and, it is 1 for $i = j$. Therefore the first inequality of the lemma statement compares random variable Z with a fraction of its expected value. We can apply the Paley-Zygmund inequality to lower bound the probability of this event:

$$\Pr\left[Z \geq \frac{2-\sqrt{3}}{2}\mathbb{E}[Z]\right] \geq \frac{3\mathbb{E}[Z]^2}{4\mathbb{E}[Z^2]}. \tag{10}$$

Using Equation 9, we can expand $\mathbb{E}[Z]^2$ as:

$$\mathbb{E}[Z]^2 = \sum_{i=1}^t \left\|\Pi_V^\perp u_i\right\|^4 + \sum_{i<j} 2\left\|\Pi_V^\perp u_i\right\|^2 \left\|\Pi_V^\perp u_j\right\|^2$$

To write $\mathbb{E}[Z^2] = \mathbb{E}\left[\left(\sum_i \left\|\Pi_V^\perp u_i\right\|^2 + 2\sum_{i<j}\chi_i\chi_j\langle \Pi_V^\perp u_i, \Pi_V^\perp u_j\rangle\right)^2\right]$ in terms of $\left\|\Pi_V^\perp u_i\right\|^2$, we note that any term with an odd power of χ_i (for any i) has expected value zero. So the relevant terms have coefficients of the form χ_i^4 or $\chi_i^2\chi_j^2$. These coefficients have expected value 1. Therefore we have:

$$\mathbb{E}[Z^2] = \sum_{i=1}^t \left\|\Pi_V^\perp u_i\right\|^4 + 4\sum_{i<j}\langle \Pi_V^\perp u_i, \Pi_V^\perp u_j\rangle^2$$

Thus by applying Cauchy-Schwartz to the inner products above, we conclude that $\mathbb{E}[Z^2] \leq 2\mathbb{E}[Z]^2$ in this setting. Applying this into Equation 10 and observing that the constant is at least $1/8$, we have that with probability at least $3/8$, $\left\|\Pi_V^\perp \mathbf{w}\right\|^2 \geq \frac{1}{8}\sum_i \left\|\Pi_V^\perp u_i\right\|^2$. In other words, with probability at least $3/8$, the first inequality of the lemma statement holds.

Next, it suffices to show that the second and third inequalities of lemma statement also holds with probability at least $15/16$ to conclude that with probability at least $1/4$, both of the inequalities in the lemma statement hold together.

This follows from a simple application of Markov's inequality. Let us take the second inequality. We note that the left side is $Z = \left\|\Pi_k^\perp \mathbf{w}\right\|^2$ and the right side is $16\mathbb{E}[Z]$. Therefore the second inequality of lemma holds with probability at least $15/16$. The proof of the third inequality is identical, and this concludes the proof of this lemma. ∎

B. Chernoff Bound Lemma

Lemma VII.1. *We toss a coin n times. The tosses are independent of each other and in each toss, the probability of seeing a head is at least p. Let H_m and T_m denote the number of heads and tails we observe in the first $m \leq n$ coin tosses. With probability $1-\delta$, we have $H_m \geq \frac{pm}{4} - \lceil 8\ln\left(\frac{2}{\delta}\right)/p\rceil$ for any $1 \leq m \leq n$. We note that although the cliam is about conjunction of all these n events, the probability does not rely on n.*

Proof: While the proof uses only simple concentration bounds, the caveat is that we do not want to have an error probability that depends on n.

We denote the expected number of heads in the first m tosses with μ which is at least pm. Applying lower tail inequality of Theorem 4 in [17] implies

$$Pr[H_m < (1-\frac{1}{2})\mu] \leq e^{-\mu/8} \leq e^{-pm/8}$$

The error probability $e^{-pm/8}$ is at most $\delta/2$ for $m \geq m' = \lceil 8\ln(2/\delta)/p\rceil$. Instead of summing up the error bound

for all values of m, we focus on the smaller geometrically growing sequence $M = \{2^\ell m' | \ell \in \mathbb{Z}^{\geq 0} \text{ AND } 2^\ell m' \leq n\}$. Having the lower bound on H_m for every $m \in M$ helps us achieve a universal lower bound on any $1 \leq m \leq n$ as follows. For any $m \leq m'$, the bound $H_m \geq pm - m'$ holds trivially.

For any other $m \leq n$, there exists an $m'' \in M$ such that $m'' \leq m \leq 2m''$. By definition H_m is at least $H_{m''}$. Assuming $H_{m''} \geq pm''/2$ implies H_m is at least $pm/4$ which proves the claim of the lemma. So we focus on bounding the error probabilities only for values in set M.

For m', the error probability is at most $\delta/2$. The next value in M is $2m'$, so given the exponential form of the error, it is at most $(\delta/2)^2$. Using union bound, the aggregate error probability for set M does not exceed

$$\frac{\delta}{2} + \left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^4 + \cdots \leq \frac{\delta/2}{1 - \delta/2} \leq \delta$$

Therefore with probability at least $1 - \delta$, we have for every $m \in M$, $H_m \geq pm/2$, and consequently for every $1 \leq m \leq n$, $H_m \geq \frac{pm}{4} - m'$ which finishes the proof. ∎

C. Proof of Theorem IV.1

We now complete the argument from Section IV-A, thus proving Theorem IV.1.

Lemma VII.2. *The number of special phases is* $O\left(k \log \frac{\|A\|_F^2}{\xi}\right)$.

Proof: Let T be the set of last columns of all special phases. By algorithm's definition we have that for any column v_i in T it holds that the projection of v_i to the orthogonal space of other columns of T that appeared earlier is at least $\frac{\xi}{k}$. This is because for any $u \in T$, the projection of u onto span(S_{pre}) is at least $\frac{\xi}{k}$ in squared length, and all the previous columns in T must be in S_{pre}.

Thus we can apply Lemma II.1 to obtain the desired conclusion. ∎

Lemma VII.3. *The number of successful ordinary (i.e., non-special) phases is* $\leq 2k \log(\frac{\|A\|_F^2}{\xi})$.

Also in this case the statement is deterministic. The proof is similar to the proof of previous lemma but it uses the inequalities that characterize a successful phase. The proof also uses the following observation (which we will need to analyze the error), so we state it first.

Observation 3. *Let* $\{u_i\}_{i=1}^r$ *be the vectors in any phase. For a special phase, we consider all the vectors in the phase except for the last one. Recall that* S_{pre} *is the set of columns selected in all previous phases. Then, we have*

$$\sum_{i=1}^{r} \left\| \Pi_{S_{pre}}^{\perp} u_i \right\|^2 \leq \frac{320\xi}{k}. \tag{11}$$

Indeed, for non-special phases, we also have

$$\sum_{i=1}^{r} \left\| \Pi_{S_{pre}}^{\perp} u_i \right\|^2 \geq \frac{160\xi}{k}. \tag{12}$$

Proof: These follow by the way we defined the phases (accumulating probabilities until they add up to 1). Unless we are considering the last vector in a special phase, none of the terms in the summation are $\geq \frac{\xi}{k}$, by definition. This completes the proof. ∎

Now we are ready to prove Lemma VII.3.

Proof of Lemma VII.3: The main idea behind the proof is to combine Observation 3, properties of a successful phase (Definition IV.2) and Lemma II.1 to obtain a proof by contradiction. Let W be the matrix that has columns equal to the vectors $\mathbf{w}$ (defined in Equation 4) in successful ordinary (i.e., non-special) phases. First, we note that the second property of a successful phase allows us to relate the best low rank approximation error of matrix A with the best low rank approximation error of matrix W(the former is at most 40 times smaller than the latter).

We know from the first property of a successful phase and also Observation 3 that for every column $\mathbf{w}$ in W, the projection orthogonal to the previous selected columns is at least $\frac{80\xi}{k}$. It is important to observe that each $\mathbf{w}$ is a linear combination of selected columns of its phase. Therefore the projection of each $\mathbf{w}$ to the orthogonal space of the other vectors in W for previous phases is also at least $\frac{80\xi}{k}$.

Therefore we can apply Lemma II.1 on vectors of W by setting $\gamma^2 = \frac{80\xi}{k}$, $c = 40$ and $\Gamma = \xi$. So the number of columns in W (number of ordinary successful phases) cannot exceed

$$2k \cdot \log\left(\frac{\|W\|_F^2}{2c\Gamma}\right) \leq 2k \cdot \log\left(\frac{40\,\|A\|_F^2}{80\xi}\right)$$

where the inequality holds because of the third property of a successful phase. ∎

The next lemma shows that it is very unlikely that the number of unsuccessful phases is asymptotically larger.

Lemma VII.4. *Let* $N^{success}$ *be the number of successful phases in the algorithm. Then with probability at least* $1-\delta$*, the total number of phases is* $O(N^{success} + \log(\frac{1}{\delta}))$.

Proof: The proof of the lemma follows from Lemma VII.1 where each phase is like a coin toss. A head in the coin toss is associated with the phase being a successful one. The only caveat is that we do not know the number of phases a priori. In fact, the existence of a phase depends on the vectors chosen in the earlier phases. However, we still have a trivial upper bound of n on the number of phases.

Each toss has a probability $\geq 1/16 - 1/20 = 1/80$ of being heads. The outcome of each toss is independent of the previous ones, and the number of tosses is $m \leq n$. Therefore the number of successful phases is at least $m/80-$

$O(\log(1/\delta))$ which means the number of phases is $O(H_m + \log(1/\delta))$ proving the claim of the lemma. ∎

These lemmas together establish the following bound on the number of phases. By definition of phases, the expected number of selected columns in each phase is not more than 2, therefore there is a direct relation between the number of phases and the number of selected columns $|S|$.

Corollary VII.5. *The total number of phases, and consequently the number selected of columns, $|S|$, at the end of the algorithm is $O(k\log(\frac{\|A\|_F^2}{\xi}) + \log(1/\delta))$, w.p. at least $1-\delta$.*

Given this, the bound on the error follows.

Lemma VII.6. *Suppose the algorithm receives the vectors $\{u_i\}_{i=1}^n$. Let S_i denote the output matrix $S = S_{pre} \cup S_{current}$ after we see the vector u_i. Then, w.p. at least $1-\delta$, we have*

$$\sum_{i=1}^{n} \left\|\Pi_{S_i}^{\perp} u_i\right\|^2 \leq \xi O\left(\log\left(\frac{\|A\|_F^2}{\xi}\right) + \frac{\log(1/\delta)}{k}\right).$$

Remark. Note that for any u_i, the set S_i is a superset of S_{pre}. The definition of S_i ensures that for any vector u_i for which the "p_u" in the algorithm is ≥ 1, we have $\Pi_{S_i}^{\perp} u_i = 0$ (as we would have marked the phase as special and selected u_i).

Proof: By Observation 1, the error accumulated in each phase is at most $\frac{320\xi}{k}$, so combining this with Corollary VII.5 implies the lemma. ∎

Now we are ready to prove Theorem IV.1.

Proof of Theorem IV.1: Corollary VII.5 and Lemma VII.6 prove the first and third points of the Theorem about the number of selected columns and the error bound. For the second point we note that the most expensive step in the algorithm is to compute the projection $\Pi_{S_{pre}}^{\perp}$ for each column u which requires computing $|S_{pre}| \leq |S|$ projections. To compute this projection efficiently we can keep an orthogonalized version of S_{pre} and we then use it to compute the projections. This can be done easily in time $O(|S|d) \in \widetilde{O}(kd)$ per iteration. ∎

D. Proof of Lemma IV.5

We follow the proof strategy of [12]. Recall that we are given a matrix $A \in \mathbb{R}^{n\times m}$. Let $A^{(k)} = U\Sigma V$ be the rank-k SVD of A, where $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times m}$ have orthogonal columns and rows respectively. Also, let $B \in \mathbb{R}^{n\times m}$ denote $\Pi_{S_1}^{\perp} A$.

Proof of Lemma IV.5: Let Y_i be a 0/1 random variable that indicates if the column A_i is chosen. By hypothesis, we have $p_i := \Pr[Y_i = 1] \geq c \cdot \frac{\|B_i\|^2}{\|B\|_F^2}$, for all $j \in [m]$.

Now, recall from our definition of B above, that $A_j = \Pi_{S_1} A_j + B_j$. The main idea behind the proof is to consider the vector valued random variables, for all $i \in [k]$,

$$w_i := \sum_{j\in[m]} V_{i,j}\Pi_{S_1} A_j + \sum_{j\in[m]} \frac{Y_j}{p_j} V_{i,j} B_j.$$

The first term is independent of the sampling, it simply makes the expected value of w_i something nice. To see this, note that

$$\begin{aligned}\mathbb{E}[w_i] &= \sum_{j\in[m]} V_{i,j}\Pi_{S_1} A_j + \sum_{j\in[m]} V_{i,j} B_j V A^T \\ &= \sigma_i U_i.\end{aligned}$$

In other words, in expectation, w_i is precisely a scaled version of the ith singular vector. Define $z_i := w_i/\sigma_i$, for $i \in [k]$. The key observation is that each z_i is in the space spanned by the chosen columns $\{A_j : Y_j = 1\}$, together with S_1 (as $\Pi_{S_1} A_j \in S_1$ for all A_j).

Now, the approximation for A we consider is $(\sum_{i=1}^k z_i U_i^T)A$. Clearly, the column span of this matrix is contained in span$\{z_i : i \in [k]\}$. Now, suppose we extend the basis $V^{(1)}, \ldots, V^{(k)}$ (recall that these are the rows of V) to a basis of $\mathbb{R}^m$, using the full SVD of A, thus obtaining $V^{(k+1)}, \ldots, V^{(m)}$ (some of the corresponding singular values could be zero). Then, we have

$$\begin{aligned}&\left\|A - (\sum_{i\in[k]} z_i U_i^T)A\right\|_F^2 \\ &= \sum_{j\in[m]} \left\|\left(A - (\sum_{i\in[k]} z_i U_i^T)A\right) V^{(j)}\right\|^2 \\ &= \sum_{j\in[m]} \left\|\sigma_j U_j - (\sum_{i\in[k]} z_i U_i^T)\sigma_j U_j\right\|^2 \\ &= \sum_{j\in[k]} \|\sigma_j U_j - w_j\|^2 + \sum_{j>k} \sigma_j^2 \\ &\qquad \text{(since } U_j \text{ are orthonormal).} \qquad (13)\end{aligned}$$

The second term in (13) is simply $\left\|A - A^{(k)}\right\|_F^2$. Thus, we need to bound the first term. This can be bounded using the variance of w_j, summed over $j \in [k]$. Note that for any j, we have the variance

$$\begin{aligned}&\mathbb{E}\,\|w_j - \mathbb{E}[w_j]\|^2 \\ &= \mathbb{E}\left\|\sum_{j\in[m]} \left(\frac{Y_j}{p_j} - 1\right) V_{i,j} B_j\right\|^2 \\ &= \sum_{j\in[m]} V_{i,j}^2 \|B_j\|^2 \,\mathbb{E}\left(\frac{Y_j}{p_j} - 1\right)^2 \\ &\leq \sum_{j\in[m]} 2V_{i,j}^2 \frac{\|B_j\|^2}{p_j} \qquad (14) \\ &\leq \sum_{j\in[m]} 2V_{i,j}^2 \frac{\|B\|_F^2}{c} = \frac{2\|B\|_F^2}{c}. \qquad (15)\end{aligned}$$

In the last step, we used the lower bound on the probability p_j. Thus, summing over $j \in [k]$, we have that

$$\mathbb{E}\left[\sum_{j\in[k]} \|w_j - \sigma_j U_j\|^2\right] \leq \frac{2k}{c}\|B\|_F^2 .$$

Thus, using Markov's inequality here, and combining with (13), we have that with probability $\geq 9/10$,

$$\left\|A - \sum_{i\in[k]} z_i U_i^T A\right\|^2 \geq \frac{20k}{c}\|B\|_F^2 + \left\|A - A^{(k)}\right\|_F^2 .$$

This completes the proof. ■

REFERENCES

[1] Raman Arora, Andrew Cotter, and Nati Srebro. Stochastic optimization of PCA with capped MSG. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 1815–1823, 2013.

[2] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[3] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The fast convergence of incremental PCA. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3174–3182, 2013.

[4] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 305–314, 2011.

[5] 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 2015, San Diego, CA, USA, January 4-6, 2015*, pages 887–901, 2015.

[6] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 968–977, 2009.

[7] Christos Boutsidis and David P. Woodruff. Optimal CUR matrix decompositions. *SIAM J. Comput.*, 46(2):543–589, 2017.

[8] 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 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 205–214, 2009.

[9] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *SODA*, 2017.

[10] Michael B. Cohen, Cameron Musco, and Jakub W. Pachocki. Online row sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 7:1–7:18, 2016.

[11] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 329–338, 2010.

[12] Amit Deshpande, L. Rademacher Santosh Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1117–1126, 2006.

[13] Amit Deshpande and Santosh 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 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 292–303, 2006.

[14] Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 223–232, 2003.

[15] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 370–378, 1998.

[16] Mina Ghashami and Jeff M. Phillips. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 707–717, 2014.

[17] Michel Goemans. Chernoff bounds, and some applications, February 2015.

[18] Venkatesan Guruswami and Ali Kemal Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1207–1214, 2012.

[19] Zohar Shay Karnin and Edo Liberty. Online with spectral bounds. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 1129–1140, 2015.

[20] Wojciech Kotlowski and Gergely Neu. Bandit principal component analysis. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 1994–2024, 2019.

[21] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 299–308, Washington, DC, USA, 2010. IEEE Computer Society.

[22] Edo Liberty. Simple and deterministic matrix sketching. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 581–588, 2013.

[23] Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pages 81–89, 2016.

[24] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, V. Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences of the United States of America*, 104 51:20167–72, 2007.

[25] Adam Meyerson. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 426–431, 2001.

[26] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. Memory limited, streaming PCA. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2886–2894, 2013.

[27] Jiazhong Nie, Wojciech Kotlowski, and Manfred K. Warmuth. Online PCA with optimal regret. *Journal of Machine Learning Research*, 17:173:1–173:49, 2016.

[28] John Novembre, Toby Johnson, Katarzyna Bryc, Zoltán Kutalik, Adam R Boyko, Adam Auton, Amit Indap, Karen S King, Sven Bergmann, Matthew R Nelson, Matthew Stephens, and Carlos D Bustamante. Genes mirror geography within europe. *Nature*, 456(7218):98, 2008.

[29] Saurabh Paul, Malik Magdon-Ismail, and Petros Drineas. Column selection via adaptive sampling. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 406–414, 2015.

[30] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152, 2006.

[31] Joel A. Tropp. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 978–986, 2009.

[32] Matthew A. Turk and Alex Pentland. Face recognition using eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 1991, 3-6 June, 1991, Lahaina, Maui, Hawaii, USA*, pages 586–591, 1991.

[33] Yining Wang and Aarti Singh. Provably correct algorithms for matrix column subset selection with selectively sampled data. *Journal of Machine Learning Research*, 18:156:1–156:42, 2017.

[34] Manfred K. Warmuth and Dima Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1481–1488, 2006.