Online Principal Component Analysis

Christos Boutsidis [*] Dan Garber [†] Zohar Karnin [‡]

Edo Liberty [§]

Abstract

We consider the *online* version of the well known Principal Component Analysis (PCA) problem. In standard PCA, the input to the problem is a set of vectors $X = [x_1, \ldots, x_n]$ in $\mathbb{R}^{d\times n}$ and a target dimension $k < d$; the output is a set of vectors $Y = [y_1, \ldots, y_n]$ in $\mathbb{R}^{k\times n}$ that minimize $\min_\Phi \|X - \Phi Y\|_\mathrm{F}^2$ where Φ is restricted to be an isometry. The global minimum of this quantity, OPT_k, is obtainable by offline PCA.

In the online setting, the vectors x_t are presented to the algorithm one by one. For every presented x_t the algorithm *must* output a vector y_t before receiving x_{t+1}. The quality of the result, however, is measured in exactly the same way, $\mathrm{ALG} = \min_\Phi \|X - \Phi Y\|_\mathrm{F}^2$. This paper presents the first approximation algorithms for this setting of online PCA. Our algorithms produce $y_t \in \mathbb{R}^\ell$ with $\ell = O(k \cdot \mathrm{poly}(1/\varepsilon))$ such that $\mathrm{ALG} \le \mathrm{OPT}_k + \varepsilon\|X\|_\mathrm{F}^2$.

[*]Yahoo Labs, New York, NY

[†]Technion Israel Institute of Technology and partially at Yahoo Labs

[‡]Yahoo Labs, Haifa, Israel

[§]Yahoo Labs, New York, NY

1 Introduction

Principal Component Analysis (PCA) is one of the most well known and widely used procedures in scientific computing. It is used for dimension reduction, signal denoising, regression, correlation analysis, visualization etc [1]. It can be described in many ways but one is particularly appealing in the context of online algorithms. Given n high-dimensional vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and a target dimension $k < d$, produce n other low-dimensional vectors $y_1, \ldots, y_n \in \mathbb{R}^k$ such that the reconstruction error is minimized. For any set of vectors y_t the reconstruction error is defined as

$$\min_{\Phi \in \mathcal{O}_{d,k}} \sum_{t=1}^{n} \|x_t - \Phi y_t\|_2^2. \tag{1}$$

where $\mathcal{O}_{d,k}$ denotes the set of $d \times k$ isometries

$$\mathcal{O}_{d,k} = \{\Phi \in \mathbb{R}^{d \times k} | \forall y \in \mathbb{R}^k \ \ \|\Phi y\|_2 = \|y\|_2\}. \tag{2}$$

For any $x_1, \ldots, x_n \in \mathbb{R}^d$ and $k < d$ standard offline PCA finds the optimal $y_1, \ldots, y_n \in \mathbb{R}^k$ which minimize the reconstruction error

$$\mathrm{OPT}_k = \min_{y_t \in \mathbb{R}^k} \left(\min_{\Phi \in \mathcal{O}_{d,k}} \sum_{t=1}^{n} \|x_t - \Phi y_t\|_2^2 \right). \tag{3}$$

The solution to this problem goes through computing $W \in \mathcal{O}_{d,k}$ whose columns span the top k left singular vectors of the matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$. Setting $y_t = W^T x_t$ and $\Phi = W$ specifies the optimal solution.

Computing the optimal $y_t = W^T x_t$ naively requires several passes over the matrix X. Power iteration based methods for computing W are memory and CPU efficient but require $\omega(1)$ passes over X. Two passes also naively suffice; one to compute XX^T from which W is computed and one to generate the mapping $y_t = W^T x_t$. The bottleneck is in computing XX^T which demands $\Omega(d^2)$ auxiliary space (in memory) and $\Omega(d^2)$ operations per vector x_t (assuming they are dense). This is prohibitive even for moderate values of d. A significant amount of research went into reducing the computational overhead of obtaining a good approximation for W in one pass [2, 3, 4, 5, 6, 7, 8, 9]. Still, a second pass is needed to produce the reduced dimension vectors y_t.

1.1 Online PCA

In the online setting, the algorithm receives the input vectors x_t one ofter the other and must always output y_t before receiving x_{t+1}. The cost of the online algorithm is measured like in the offline case

$$\mathrm{ALG} = \min_{\Phi \in \mathcal{O}_{d,\ell}} \sum_{t=1}^{n} \|x_t - \Phi y_t\|_2^2 \ .$$

Note that the target dimension of the algorithm, ℓ, is potentially larger than k to compensate for the handicap of operating online.

This is a natural model when a downstream online (rotation invariant) algorithm is applied to y_t. Examples include online algorithms for clustering (k-means, k-median), regression, classification (SVM, logistic regression), facility location, k-server, etc. By operating on the reduced dimensional vectors, these algorithms gain computational advantage but there is a much more important reason to apply them post PCA.

PCA denoises the data. Arguably, this is the most significant reason for PCA being such a popular and successful preprocessing stage in data mining. Even when a significant portion of the Frobenius norm of X is attributed to isotropic noise, PCA can often still recover the signal. This is the reason that clustering, for example, the denoised vectors y_t often gives better qualitative results than clustering x_t directly. Notice that in this setting the algorithm cannot retroactively change past decisions. Furthermore, future decisions should try to stay consistent with past ones, even if those were misguided.

Our model departs from earlier definitions of online PCA. We shortly review three other definitions and point out the differences.

1.1.1 Random projections

Most similar to our work is the result of Sarlos [5]. They generate $y_t = S^T x_t$ where $S \in \mathbb{R}^{d\times\ell}$ is generated randomly and independently from the data. For example, each element of S can be ± 1 with equal probability (Theorem 4.4 in [10]) or drawn from a normal Gaussian distribution (Theorem 10.5 in [11]). Then, with constant probability for $\ell = \Theta(k/\varepsilon)$

$$\min_{\Psi \in \mathbb{R}^{d\times\ell}} \sum_{t=1}^{n} \|x_t - \Psi y_t\|_2^2 \leq (1+\varepsilon)\,\mathrm{OPT}_k \ .$$

Here, the best reconstruction matrix is $\Psi = XY^\dagger$ which is *not* an isometry in general.[1] We claim that this seemingly minute departure from our model is actually very significant. Note that the matrix S exhibits the Johnson

[1]The notation $Y^\dagger$ stands for the Moore Penrose inverse or pseudo inverse of Y.

Lindenstrauss property [12, 13, 14]. Roughly speaking, this means the vectors y_t approximately preserve the lengths, angels, and distances between all the vectors x_t. Thereby, preserving the noise and signal in x_t equally well. This is not surprising given that S is generated independently from the data. Observe that to nullify the noise component $\Psi = XY^\dagger$ must be far from being an isometry and that $\Psi = X(S^T X)^\dagger$ can only be computed after the entire matrix was observed.

For example, let $\Phi \in \mathcal{O}_{d,k}$ be the optimal PCA projection for X. Consider $y_t \in \mathbb{R}^\ell$ whose first k coordinates contain $\Phi^T x_t$ and the rest $\ell - k$ coordinates contain an arbitrary vector $z_t \in \mathbb{R}^{\ell-k}$. In the case where $\|z_t\|_2 \gg \|\Phi^T x_t\|_2$ the geometric arrangement of y_t potentially shares very little with that of signal in x_t. Yet, $\min_{\Psi \in \mathbb{R}^{d\times\ell}} \sum_{t=1}^n \|x_t - \Psi y_t\|_2^2 = \mathrm{OPT}_k$ by setting $\Psi = (\Phi | 0^{d\times(\ell-k)})$. This would have been impossible had Ψ been restricted to being an isometry.

1.1.2 Regret minimization

A regret minimization approach to online PCA was investigated in [15, 16]. In their setting of online PCA, at time t, *before* receiving the vector x_t, the algorithm produces a rank k projection matrix $P_t \in \mathbb{R}^{d\times d}$.[2] The authors present two methods for computing projections P_t such that the quantity $\sum_t \|x_t - P_t^T x_t\|_2^2$ converges to OPT_k in usual no-regret sense. Since each P_t can be written as $P_t = U_t U_t^T$ for $U_t \in \mathcal{O}_{d,k}$ it would seem that setting $y_t = U_t^T x_t$ should solve our problem. Alas, the decomposition $P_t = U_t U_t^T$ (and therefore y_t) is underdetermined. Even if we ignore this issue, each y_t can be reconstructed by a different U_t. To see why this objective is problematic for the sake of dimension reduction, consider our setting where we can observe x_t before outputting y_t. One can simply choose the rank 1 projection $P_t = x_t x_t^T / \|x_t\|_2^2$. On the one hand this gives $\sum_t \|x_t - P_t x_t\|_2^2 = 0$. On the other, it clearly does not provide meaningful dimension reduction.

1.1.3 Stochastic Setting

There are three recent results [17, 18, 19] that efficiently approximate the PCA objective in Equation 1. They assume the input vectors x_t are drawn i.i.d. from a fixed (and unknown) distribution. In this setting, observing n_0 columns x_t one can efficiently compute $U_{n_0} \in \mathcal{O}_{d,k}$ such that it approximately spans the top k singular vectors of X. Returning $y_t = 0^k$ for $t < n_0$ and $y_t = U_{n_0}^T x_t$ for $t \geq n_0$ completes the algorithm. This algorithm is provably

[2] Here, P_t is a square projection matrix $P_t^2 = P_t$

correct if n_0 is independent of n which is intuitively correct but non trivial to show. While the stochastic setting is very common in machine learning (e.g. the PAC model) in online systems the data distribution is *expected* to change or at least drift over time. In systems that deal with abuse detection or prevention, one can expect an almost adversarial input.

1.2 Our contributions

We design a deterministic online PCA algorithm (see Algorithm 1 in Section 2). Our main result (see Theorem 1) shows that, for any $X = [x_1, \ldots, x_n]$ in $\mathbb{R}^{d\times n}$, $k < d$ and $\varepsilon > 0$ the proposed algorithm produces a set of vectors $y_1, \ldots, y_n$ in $\mathbb{R}^\ell$ such that

$$\mathrm{ALG} \le \mathrm{OPT}_k + \varepsilon \|X\|_{\mathrm{F}}^2$$

where $\ell = \lceil 8k/\varepsilon^2 \rceil$ and ALG is the registration error as defined in Equation 1. To the best of our knowledge, this is the first online algorithm in the literature attaining theoretical guarantees for the PCA objective. The description of the algorithm and the proof of its correctness are given in Sections 2 and 3.

While Algorithm 1 solves the main technical and conceptual difficulty in online PCA, it has some drawbacks. I) It must assume that $\max_t \|x_t\|_2^2 \le \|X\|_{\mathrm{F}}^2/\ell$. This is not unlikely because we would expect $\|x_t\|_2^2 \approx \|X\|_{\mathrm{F}}^2/n$. Nevertheless, requiring it is a limitation. II) The algorithm requires $\|X\|_{\mathrm{F}}^2$ as input, which is unreasonable to expect in an online setting. III) It spends $\Omega(d^2)$ floating point operations per input vector and requires auxiliary $\Theta(d^2)$ space in memory.

Section 4 shows that, in the cost of slightly increasing the target dimension and additive error, one can address all the issues above. I) We deal with arbitrary input vectors by special handling of large norm input vectors. This is a simple amendment to the algorithm which only doubles the required target dimension. II) Algorithm 2 avoids requiring $\|X\|_{\mathrm{F}}$ as input by estimating it on the fly. A "doubling argument" analysis shows that the target dimension grows only to $O(k\log(n)/\varepsilon^2)$.[3] Bounding the target dimension by $O(k/\varepsilon^3)$ requires a significant conceptual change to the algorithm and should be considered one of the main contributions of this paper. III) Algorithm 2 spends only $O(dk/\varepsilon^3)$ floating point operations per input vector and uses only $O(dk/\varepsilon^3)$ space by utilizing a streaming matrix approximation technique [8].

[3]Here, we assume that $\|x_t\|$ are polynomial in n.

While the intuitions behind these extensions are quite natural, proving their correctness is technically intricate. We give an overview of these modifications in Section 4 and defer most of the proofs to the appendix.

2 Online PCA algorithm

Algorithm 1 receives as input $X = [x_1, \ldots, x_n]$ one vector at a time. It also receives k, ε and $\|X\|_{\mathrm{F}}^2$ (see Section 4 for an extension of this algorithm that does not require $\|X\|_{\mathrm{F}}^2$ as an input). The parameters k and ε are used only to compute a sufficient target dimension $\ell = \lceil 8k/\varepsilon^2 \rceil$ which insures that $ALG \leq \mathrm{OPT}_k + \varepsilon\|X\|_{\mathrm{F}}^2$.For the sake of simplifying the algorithm we assume that $\max_t \|x_t\|_2^2 \leq \|X\|_{\mathrm{F}}^2/\ell$. This is a reasonable assumption because we expect $\|x_t\|_2^2$ to be roughy $\|X\|_{\mathrm{F}}^2/n$ and $n \gg \ell$. Overcoming this assumption is somewhat cumbersome and uninspiring, see Section 4 for details on that.

Algorithm 1 An online algorithm for Principal Component Analysis

```
input: $X$, $k$, $\varepsilon$, $\|X\|_{\mathrm{F}}$
$\ell = \lceil 8k/\varepsilon^2 \rceil$
$U = 0^{d \times \ell}$; $C = 0^{d \times d}$
for $t = 1, ..., n$ do
  $r_t = x_t - UU^T x_t$
  while $\|C + r_t r_t^T\|_2 \geq 2\|X\|_{\mathrm{F}}^2/\ell$ do
    $[u, \lambda] \leftarrow$ TopEigenVectorAndValueOf($C$)
    Add $u$ to the next all-zero column of $U$
    $C \leftarrow C - \lambda u u^T$
    $r_t \leftarrow x_t - UU^T x_t$
  end while
  $C \leftarrow C + r_t r_t^T$
  yield: $y_t \leftarrow U^T x_t$
end for
```

In Algorithm 1 the matrix U is used to map x_t to y_t and is referred to as the projection matrix.[4] The matrix C accumulates the covariance of the residual errors $r_t = x_t - UU^T x_t$. The algorithm starts with a rank one update of C as $C = C + r_1 r_1^T$. Notice that by the assumption for x_t, we have that $\|r_1 r_1^T\|_2^2 \leq \|X\|_{\mathrm{F}}^2/\ell$, and hence for $t = 1$ the algorithm does not

[4]In linear algebra, a projection matrix is a matrix P such that $P^2 = P$. Notice that U is not a projection matrix in that sense.

enter the while-loop. Then, for the second input vector x_2, the algorithm proceeds by checking the spectral norm of $C + r_2 r_2^T = r_1 r_1^T + r_2 r_2^T$. If this does not exceed the threshold $2\|X\|_{\mathrm{F}}^2/\ell$ the algorithm keeps U unchanged. It can go all the way to $t = n$ if this remains the case for all t. Notice, that in this case, $C = \sum r_t r_t^T = \sum x_t x_t^T = XX^T$. The condition $\|C\|_2 \leq 2\|X\|_{\mathrm{F}}^2/\ell$ translates to $\|X\|_2^2 \leq 2\|X\|_{\mathrm{F}}^2/\ell$. This means the numeric rank[5] of X is at least $4k/\varepsilon^2$. That, in turn, means that OPT_k is large because there is not good rank-k approximation for X.

If, however, for some iterate t the spectral norm of $C + r_t r_t^T$ exceeds the threshold $2\|X\|_{\mathrm{F}}^2/\ell$, the algorithm makes a "correction" to U (and consequently to r_t) in order to ensure that this is not the case. Specifically, it updates U with the principal eigenvector of C at that instance of the algorithm. At the same time it updates C (inside the while-loop) by removing this eigenvector. By controlling $\|C\|_2$ the algorithm enforces that $\|\sum_t r_t r_t^T\|_2^2 \leq 2\|X\|_{\mathrm{F}}^2/\ell$ which turns out to be a key component for online PCA.

Theorem 1. *Let X, k, ε and $\|X\|_{\mathrm{F}}^2$ be inputs to Algorithm 1. Let $\ell = \lceil 8k/\varepsilon^2 \rceil$ and assume that $\max_t \|x_t\|_2^2 \leq \|X\|_{\mathrm{F}}^2/\ell$. The algorithm outputs vectors $y_1, \ldots, y_n$ in $\mathbb{R}^\ell$ such that*

$$ALG = \min_{\Phi \in \mathcal{O}_{d,\ell}} \sum_{t=1}^{n} \|x_t - \Phi y_t\|_2^2 \leq \mathrm{OPT}_k + \varepsilon\|X\|_{\mathrm{F}}^2.$$

Let $Y = [y_1, \ldots, y_n]$ and let R be the $d \times n$ matrix whose columns are r_t. In Lemma 2 we prove that $\min_{\Phi \in \mathcal{O}^{d\times\ell}} \|X - \Phi Y\|_{\mathrm{F}}^2 \leq \|R\|_{\mathrm{F}}^2$. In Lemma 3 we prove that $\|R\|_{\mathrm{F}}^2 \leq \mathrm{OPT}_k + \sqrt{4k}\|X\|_{\mathrm{F}}\|R\|_2$ and in Lemma 9 we prove that $\|R\|_2^2 \leq 2\|X\|_{\mathrm{F}}^2/\ell$. Combining these and setting $\ell = \lceil 8k/\varepsilon^2 \rceil$ completes the proof outline.

To prove the algorithm is correct, we must show that it does not add more than ℓ vectors to U. Let that number be denoted by ℓ'. We show that $\ell' \leq \ell\|R\|_{\mathrm{F}}^2/\|X\|_{\mathrm{F}}^2$ by lower bounding the different values of λ in the algorithm (Lemma 10). Observing that $\|R\|_{\mathrm{F}}^2 \leq \|X\|_{\mathrm{F}}^2$ proves the claim. In fact, by using that $\|R\|_{\mathrm{F}}^2 \leq \mathrm{OPT}_k + \varepsilon\|X\|_{\mathrm{F}}^2$ we get a tiger upper bound $\ell' \leq \ell(\mathrm{OPT}_k / \|X\|_{\mathrm{F}}^2 + \varepsilon)$. Thus, in the typical case of $\mathrm{OPT}_k < (1-\varepsilon)\|X\|_{\mathrm{F}}^2$ the algorithm effectively uses a target dimension lower than ℓ.

[5]The numeric rank, or the stable rank, of a matrix X is equal to $\|X\|_{\mathrm{F}}^2/\|X\|_2^2$.

3 Proofs of main lemmas

Let $X \in \mathbb{R}^{d\times n}$, $Y \in \mathbb{R}^{\ell\times n}$, $\tilde{X} \in \mathbb{R}^{d\times n}$ and $R \in \mathbb{R}^{d\times n}$ denote matrices whose t'th columns are $x_t \in \mathbb{R}^d$, $y_t = U_t^T x_t \in \mathbb{R}^\ell$, $\tilde{x}_t = U_t U_t^T x_t \in \mathbb{R}^d$, and $r_t = (I - U_t U_t^T)x_t \in \mathbb{R}^d$ respectively. Throughout the paper, denote by U_t and C_t the state of U and C *after* the t iteration of the algorithm concluded and before the $t+1$ began. For convenience, unless stated otherwise C and U without a subscript refer to state of these matrices after the algorithm terminated. Let $\ell = \lceil 8k/\varepsilon^2 \rceil$ and $\ell' \le \ell$ be the number of vectors u inserted into U in Algorithm 1. Let Λ be a diagonal matrix holding the values λ on the diagonal such that $\sum_{j=1}^{\ell'} \lambda_j u_j u_j^T = U\Lambda U^T$.

Lemma 2. $\mathrm{ALG} \le \|R\|_{\mathrm{F}}^2$.

Proof.

$$\begin{aligned}
\mathrm{ALG} &= \min_{\Phi\in\mathcal{O}^{d\times\ell}} \|X - \Phi Y\|_{\mathrm{F}}^2 \le \|X - UY\|_{\mathrm{F}}^2 && (4)\\
&= \sum_{t=1}^{n} \|x_t - UU_t^T x_t\|_2^2 = \sum_{t=1}^{n} \|x_t - U_t U_t^T x_t\|_2^2 = \|R\|_{\mathrm{F}}^2 \,. && (5)
\end{aligned}$$

The first inequality holds with equality for any Φ equal to U on its ℓ' non-all-zero columns which are orthogonal unit vectors by Observation 7. The second equality is due to $UU_t^T = U_t U_t^T$ which holds because U restricted to the non all-zero columns of U_t is equal to U_t. □

Lemma 3. $\|R\|_{\mathrm{F}}^2 \le \mathrm{OPT}_k + \sqrt{4k}\|X\|_{\mathrm{F}}\|R\|_2$

Proof. Let $\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_n]$ hold the reconstructed vectors $\tilde{x}_t = U_t U_t^T x_t$. Note that $X = \tilde{X} + R$. From the Pythagorean theorem and the fact that r_t and $\tilde{x}_t$ are orthogonal we have $\|X\|_{\mathrm{F}}^2 = \|\tilde{X}\|_{\mathrm{F}}^2 + \|R\|_{\mathrm{F}}^2$. In what follows, $v_1, \ldots, v_k$ denote the top k left singular vectors of X.

$$\begin{aligned}
\|R\|_{\mathrm{F}}^2 &= \|X\|_{\mathrm{F}}^2 - \|\tilde{X}\|_{\mathrm{F}}^2 \le \|X\|_{\mathrm{F}}^2 - \sum_{i=1}^{k} \|v_i^T \tilde{X}\|_2^2 && (6)\\
&= \|X\|_{\mathrm{F}}^2 - \sum_{i=1}^{k} \|v_i^T X - v_i^T R\|_2^2 && (7)\\
&\le \|X\|_{\mathrm{F}}^2 - \sum_{i=1}^{k} \|v_i^T X\|_2^2 + 2\sum_{i=1}^{k} |v_i^T X R^T v_i| && (8)\\
&\le \mathrm{OPT}_k + 2\|R\|_2 \sum_{i=1}^{k} \|v_i^T X\|_2 \le \mathrm{OPT}_k + 2\|R\|_2 \cdot \sqrt{k}\|X\|_{\mathrm{F}} && (9)
\end{aligned}$$

In Equation 8 we used the fact that for any two vectors α, β: $\|\alpha - \beta\|_2^2 \leq \|\alpha\|_2^2 + 2|\alpha^T \beta|$. In Equation 9 we use that $\text{OPT}_k = \|X\|_{\text{F}}^2 - \sum_{i=1}^k \|v_i^T X\|_2^2$ the Cauchy-Schwarz inequality $\sum_{i=1}^k \|v_i^T X\|_2 \leq (k \sum_{i=1}^k \|v_i^T X\|_2^2)^{1/2} \leq \sqrt{k}\|X\|_{\text{F}}$. □

Lemma 4 (Upper bound for λ_j). *For all $j = 1, 2, ..., \ell'$: $\lambda_j \leq 2\|X\|_{\text{F}}^2/\ell$.*

Proof. The updates to C that increase its largest eigenvalue are $C \leftarrow C + r_t r_t^T$. But, these updates occur after failing the while-loop condition which means that $\|C + r_t r_t^T\|_2 \leq 2\|X\|_{\text{F}}^2/\ell$ at the beginning of every iteration. Also note that the update $C \leftarrow C - \lambda u u^T$ only reduces the two norm of C. □

Observation 5. During the execution of the algorithm the matrix C is symmetric and positive semidefinite. Initially C is the all-zeros matrix which is symmetric and positive semidefinite. The update $C \leftarrow C + r_t r_t^T$ clearly preserves these properties. The update $C \leftarrow C - \lambda u u^T$ also preserves these properties because u is an eigenvector of C at the time of the update with corresponding eigenvalue λ.

Observation 6. After the end of each iteration, $CU = 0$. Note that $CUU^T = 0$ if and only if $CU = 0$. We prove by induction. Let C' and U' be the state of C and U at the end of the last iteration and assume that $C'U' = 0$. If the while loop was not entered $CU = (C' + r_t r_t^T)U' = (C' + (I - U'U'^T)x_t x_t^T(I - U'U'^T))U' = C'U'$. For every iteration of the while loop $CUU^T = (C' - \lambda u u^T)(U'U'^T + uu^T) = C'U'U'^T + (C'u - \lambda u)u^T - \lambda u u^T U'U'^T$. Because u is an eigenvector of C' with eigenvalue λ we have $C'u = \lambda u$. This means $(C'u - \lambda u) = 0$ and $\lambda u u^T U'U'^T = uu^T C'U'U'^T = 0$.

Observation 7. The non all-zero columns of U are mutually orthogonal unit vectors. First, they are eigenvectors of C and thus unit norm by the standard convention. Second, when u is added to U it is an eigenvector of C with eigenvalue λ. Thus, $u^T U = \frac{1}{\lambda} u^T C U = 0$ by Observation 6.

Observation 8. After the conclusion of iteration t we have $C = \sum r_t r_t^T - \sum_j \lambda_j u_j u_j^T$ where j sums over the vectors added thus far. Specifically, when the algorithm terminates $RR^T = C + U\Lambda U^T$.

Lemma 9. $\|R\|_2^2 \leq 2\|X\|_{\text{F}}^2/\ell$.

Proof. Let z be the top eigenvector of RR^T and let z_C and z_U be its (orthogonal) projections into the span of C and $U\Lambda U^T$.

$$\begin{aligned} \|R\|_2^2 &= \|RR^T z\|_2 = \|(C + U\Lambda U^T)z\|_2 = \sqrt{\|Cz_C\|_2^2 + \|U\Lambda U^T z_U\|_2^2} \\ &\leq \sqrt{\|C\|^2\|z_C\|_2^2 + \|U\Lambda U^T\|^2\|z_C\|_2^2} \leq (2\|X\|_{\text{F}}^2/\ell)\sqrt{\|z_C\|_2^2 + \|z_U\|_2^2} \end{aligned}$$

Here we used $RR^T = C + U\Lambda U^T$ (Observation 8), $CU = 0$ (Observation 6), $\|C\|_2 \le 2\|X\|_{\mathrm{F}}^2/\ell$ and $\|U\Lambda U^T\|_2 \le 2\|X\|_{\mathrm{F}}^2/\ell$ (Lemma 4). $\square$

Lemma 10 (Lower bound for λ_j)**.** *For all $j = 1, 2, ..., \ell'$: $\lambda_j \ge \|X\|_{\mathrm{F}}^2/\ell$.*

Proof. Note that the condition in the while loop is $\|C + r_t r_t^T\|_2 \ge 2\|X\|_{\mathrm{F}}^2/\ell$. In the point of the algorithm when this condition is true, we have $\|C\|_2 \ge \|C + r_t r_t^T\|_2 - \|r_t r_t^T\|_2 \ge \|X\|_{\mathrm{F}}^2/\ell$. The first inequality follows by the triangle inequality. The second inequality uses $\|C+r_t r_t^T\|_2 \ge 2\|X\|_{\mathrm{F}}^2/\ell$ and $\|r_t r_t^T\|_2 \le \|X\|_{\mathrm{F}}^2/\ell$. To verify that $\|r_t r_t^T\|_2 \le \|X\|_{\mathrm{F}}^2/\ell$ recall that $r_t = (I - UU^T)x_t$. Note that $I - UU^T$ is a projection matrix and $\|r_t\|_2^2 \le \|x_t\|_2^2 \le \|X\|_{\mathrm{F}}^2/\ell$ by our assumption on the input. $\square$

Lemma 11. *The while loop in the algorithm occurs at most ℓ' times and*

$$\ell' \le \ell \cdot \min\{1, \mathrm{OPT}_k / \|X\|_{\mathrm{F}}^2 + \varepsilon\}$$

Proof. We bound the trace of the matrix $U\Lambda U^T$ from above and below.

$$\|R\|_{\mathrm{F}}^2 = \mathrm{Tr}(RR^T) \ge \mathrm{Tr}(U\Lambda U^T) = \mathrm{Tr}(\Lambda) = \sum_{j=1}^{\ell'} \lambda_j \ge \ell'\|X\|_{\mathrm{F}}^2/\ell \qquad (10)$$

The first inequality is correct because $RR^T = C + U\Lambda U^T$ (from from observation 8) coupled with the fact that C is symmetric and positive semi-definite (Observation 5). The last inequality follows from Lemma 10. Since $\|X\|_{\mathrm{F}}^2 \ge \|R\|_{\mathrm{F}}^2$ we get $\ell' \le \ell$. Also, by Theorem 1 we have that $\|R\|_{\mathrm{F}}^2 \le \mathrm{OPT}_k + \varepsilon\|X\|_{\mathrm{F}}^2$. Combining with Equation 10, $\ell' \le \ell(\mathrm{OPT}_k / \|X\|_{\mathrm{F}}^2 + \varepsilon)$. $\square$

4 Efficient and general online PCA algorithm

In this section we explain the revisions needed to solve the issues raised in Section 1.2. We start by removing the assumption that $\|x_t\|_2^2 \le \|X\|_{\mathrm{F}}^2/\ell$. The idea is, given a vector x_t whose norm is larger than $\|X\|_{\mathrm{F}}^2/\ell$, compute a unit vector, u, in the direction of its corresponding residual. Add u to the U and project C on $(I - uu^T)$. The analysis resulting this change is almost the same to the one above. Observe that no cost is incurred by these vectors. And, that there could be at most ℓ vectors x_t such that $\|x_t\|_2^2 \ge \|X\|_{\mathrm{F}}^2/\ell$. Therefore, the modified algorithm requires at most twice the target dimension.

We can also avoid requiring $\|X\|_{\mathrm{F}}$ as input rather simply. The while-loop condition in Algorithm 1 can be revised to $\|C + r_t r_t\| > \|X_t\|_{\mathrm{F}}^2/\ell$

where $X_t = [x_1, \ldots, x_t]$ is the data matrix observed so far. Using the same analysis as in Section 3 we can argue the following; during the time in which $\|X_t\|_{\mathrm{F}}^2 \in \|x_1\|_2^2(2^\tau, 2^{\tau+1}]$ the number of added directions cannot exceed $O(\ell)$. Thus, assuming $\|x_t\|_2^2 \leq \|x_1\|_2^2 \operatorname{poly}(n)$ the target dimension is bounded by $O(\ell \log(\|X\|_{\mathrm{F}}^2/\|x_1\|_2^2)) = O(k \log(n)/\varepsilon^2)$.

In Algorithm 2 we obtaining a bound on the target dimension that does not depend on n. Below we give the main theorem about Algorithm 2 and some proof sketches. Unfortunately, due to space constraints the full description and proofs are deferred to the appendix.

Theorem 12. *Algorithm 2 guaranties that* $\mathrm{ALG} \leq \mathrm{OPT}_k + \varepsilon\|X\|_{\mathrm{F}}^2$. *For some* $\delta = \min\{\varepsilon, \varepsilon^2\|X\|_{\mathrm{F}}^2/\mathrm{OPT}\}$ *it requires* $O(dk/\delta^3)$ *space and* $O(ndk/\delta^3 + \log(n)dk^3/\delta^6)$ *arithmetic operations assuming* $\|x_t\|_2$ *are polynomial in* n. *The target dimension of Algorithm 2 is at most* k/δ^3.

To obtain the target dimension of k/ε^3 stated in theorem 12 we argue as follows. If many directions were already added to U, instead of adding new directions we can replace the "least useful" existing ones. These replacements introduce the need for two new analyses. One is a modified bound on the total number of times a new direction is added, and the second is a bound on the loss of the algorithm, taking the removals of directions into account. We use a potential function of roughly $\|C\|_{\mathrm{F}}^2$ and show that replacing an old vector with a new one will always incur an increase to the potential function; on the other hand the function is always upper bounded by the total Frobenius norm of the observed data. This allows us to bound the number of new vectors entered to U during the time that $\|X_t\|_{\mathrm{F}}^2 \in \|x_1\|_2^2(2^\tau, 2^{\tau+1}]$ by $O(\ell)$ (for appropriate ℓ). For bounding the loss, we show that in each replacement, because we choose to discard the least useful column of U, we suffer an additive loss of at most $\frac{\varepsilon}{\ell}\|X_t\|_{\mathrm{F}}^2$. These two bounds combined prove that the additional loss suffered due to replacing directions is no more than roughly $\varepsilon\|X\|_{\mathrm{F}}^2$. Since we already suffer a similar additive penalty, this completes our analysis.

To deal with time and memory efficiency issues Algorithm 2 uses existing matrix sketching techniques. We chose the Frequent Directions sketch introduced by Liberty in [8]. In Algorithm 1 we take C to be the actual projection of RR^T onto the space orthogonal to that spanned by U (though the algorithm is not exactly stated this way, this is equivalent to what is done there). In Algorithm 2 we take C to be the projection of ZZ^T onto the space orthogonal to that spanned by U, where Z is a sketch of R with $\|ZZ^T - RR^T\|$ having a bounded spectral norm. The same analysis works with only technical modification.

References

[1] George H Dunteman. *Principal components analysis*. Number 69. Sage, 1989.

[2] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, pages 370–, Washington, DC, USA, 1998. IEEE Computer Society.

[3] Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 223–232, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[4] Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *APPROX-RANDOM*, pages 292–303, 2006.

[5] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

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

[7] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences,*, 104(51):20167–20172, December 2007.

[8] Edo Liberty. Simple and deterministic matrix sketching. In *KDD*, pages 581–588, 2013.

[9] Mina Ghashami and Jeff M. Phillips. Relative errors for deterministic low-rank matrix approximations. In *SODA*, 2014.

[10] 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*, pages 205–214. ACM, 2009.

[11] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[12] W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[13] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical Report TR-99-006, Berkeley, CA, 1999.

[14] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[15] Manfred K. Warmuth and Dima Kuzmin. Randomized online pca algorithms with regret bounds that are logarithmic in the dimension, 2007.

[16] Jiazhong Nie, Wojciech Kotlowski, and Manfred K. Warmuth. Online pca with optimal regrets. In *ALT*, pages 98–112, 2013.

[17] Raman Arora, Andy Cotter, and Nati Srebro. Stochastic optimization of pca with capped msg. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1815–1823. 2013.

[18] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. Memory limited, streaming pca. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2886–2894. 2013.

[19] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The fast convergence of incremental pca. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3174–3182. 2013.

A Implementation of extensions presented in section 4

In this section we modify Algorithm 1 in order to (1) remove the assumption of knowledge of $\|X\|_{\mathrm{F}}^2$ and (2) improve the time and space complexity.

These two improvements are made at the expense of sacrificing the accuracy *slightly*. To sidestep trivial technicalities, we assume some prior knowledge over the quantity $\|X\|_F^2$; we merely assume that we have some quantity w_0 such that $w_0 \leq \|X\|_F^2$ and $w_0 \gg \|x_t\|^2$ for all $\|x_t\|_2^2$. The assumption of knowing w_0 can be removed by working with a buffer of roughly k/eps^3 column. Since we already require this amount of space, the resulting increase in the memory complexity is asymptotically negligible.

Algorithm 2 An efficient online PCA algorithm

```
input: $X$, $k$, $\varepsilon \in (0, \frac{1}{15})$, $w_0$ (which defaults to $w_0 = 0$)
$U = 0^{d \times k/\varepsilon^3}$, $Z = 0^{d \times k/\varepsilon^2}$, $w = 0$, $w_U = 0^{k/\varepsilon^3}$
for $t = 1, ..., n$ do
    $w = w + \|x_t\|_2^2$
    $r_t = x_t - UU^T x_t$
    $C = (I - UU^T)ZZ^T(I - UU^T)$
    while $\|C + r_t r_t^T\|_2 \geq \max\{w_0, w\} \cdot \frac{k}{\varepsilon^2}$ do
        $u, \lambda = \text{topEigenvectorAndValue}(C)$
        $(w_U)_u \leftarrow \lambda$
        If $U$ has a zero column, write $u$ in its place. Otherwise, write $u$
        instead of the column $v$ of $U$ with the minimal quantity of $(w_U)_v$
        $C = (I - UU^T)ZZ^T(I - UU^T)$
        $r_t = x_t - UU^T x_t$
    end while
    Sketch $r_t$ into $Z$
    For each non-zero column $u$ in $U$, $(w_U)_u \leftarrow (w_U)_u + \langle x_t, u \rangle^2$
    yield: $y_t \leftarrow U^T x_t$
end for
```

A.1 Notations

For a variable ξ in the algorithm, where ξ can either be a matrix Z, X, C or a vector r, x we denote by ξ_t the value of ξ at the end of iteration number t. ξ_0 will denote the value of the variable at the beginning of the algorithm. For variables that may change during the while loop we denote by $\xi_{t,z}$ the value of ξ at the end of the z'th while loop in the t'th iteration. In particular, if the while loop was never entered the value of ξ at the end of the iteration t will be equal to $\xi_{t,0}$. For such ξ that may change during the while loop notice that ξ_t is its value at the end of the iteration, meaning after the last while loop has ended.

An exception to the above is for the variable w. Here, we denote by w_t the value of $\max\{w_0, w\}$ at the end of iteration number t. We denote by n the index of the last iteration of the algorithm. In particular ξ_n denotes the value of ξ upon the termination of the algorithm.

A.2 Matrix Sketching

We use the Frequent Direction matrix sketching algorithm presented by Liberty in [8]. This algorithm provides a sketching algorithm for the streaming version where we observe a matrix R column by column.

Lemma 13. Let $R_1, \ldots, R_t, \ldots$ be a sequence of matrices with columns of dimension d where R_{t+1} is obtained by appending a column vector r_{t+1} to R_t. Let $Z_1, \ldots, Z_t, \ldots$ the corresponding sketches of R obtained by adding these columns as they arrive, according to the Frequent Direction algorithm in [8] with parameter ℓ.

1. The worst case time required by the sketch to add a single column is $O(\ell d)$.
2. Each Z_t is a matrix of dimensions $d \times O(\ell)$.
3. Let u be a left singular vector of Z_t with singular value σ and assume that r_{t+1} is orthogonal to u. Then u is a singular vector of Z_{t+1} with singular value $\leq \sigma$.
4. For any vector u and time t it holds that $\|Z_t u\| \leq \|R_t u\|$.
5. For any t there exists a positive semidefinite matrix E_t such that $\|E_t\|_2^2 \leq \|R_t\|_F^2/\ell$ and $Z_t Z_t^T + E = R_t R_t^T$.

A.3 Proof of Theorem 12

Observation 14. *At all times, the matrix $U^T U$ is a diagonal matrix with either zeros or ones across the diagonal. In other words, the non-zero column vectors of U, at any time point are orthonormal.*

Proof. We prove the claim for any $U_{t,z}$ by induction on (t, z). For $t = 0$ the claim is trivial as $U_0 = 0$. For the step we need only to consider $U_{t,z}$ for $z > 0$ since $U_{t,0} = U_{t-1}$. Let u be the new vector in $U_{t,z}$. Since u is defined as an eigenvector of

$$C_{t,z-1} = (I - U_{t,z-1}U_{t,z-1}^T) Z_{t-1} Z_{t-1}^T (I - U_{t,z-1}U_{t,z-1}^T) ,$$

we have that $U_{t,z-1} u = 0$ and the claim immediately follows. $\square$

Lemma 15. *For all t, z, the non-zero columns of $U_{t,z}$ are left singular vectors of Z_{t-1} (possibly with singular value zero). In particular, the claim holds for the final values of the matrices U and Z.*

Proof. We prove the claim by induction on t, z, with a trivial base case of $t = 1$ where $Z_{t-1} = 0$. Let $t > 1$. For $z = 0$, each non-zero column vector u of $U_{t,0}$ is a non-zero column vector of $U_{t-1,z}$ for the largest valid z w.r.t $t-1$. By the induction hypothesis it holds that u is a singular vector of Z_{t-2}. According to Observation 14 we have that r_{t-1}, the vector added to Z_{t-2} is orthogonal to u, hence Lemma 13, item 3 indicates that u is a singular vector of Z_{t-1} as required.

Consider now $z > 0$. In this case u is a vector added in the while loop. Recall that u is defined as an eigenvector of

$$C_{t,z-1} = (I - U_{t,z-1}U_{t,z-1}^T)Z_{t-1}Z_{t-1}^T(I - U_{t,z-1}U_{t,z-1}^T)$$

According to our induction hypothesis, all of the non-zero column vectors of $U_{t,z-1}$ are singular vectors of Z_{t-1}, hence

$$Z_{t-1}Z_{t-1}^T = C_{t,z-1} + U_{t,z-1}U_{t,z-1}^T Z_{t-1}Z_{t-1}^T U_{t,z-1}U_{t,z-1}^T \ .$$

The two equalities above imply that any eigenvector of $C_{t,z-1}$ is an eigenvector of $Z_{t-1}Z_{t-1}^T$ as well. It follows that u is a singular vector of Z_{t-1} as required, thus proving the claim.

□

Lemma 16. *Let v be a column vector of $U_{t,z}$ that is not in $U_{t,z+1}$. Let (t_v, z_v) be the earliest time stamp from which v was a column vector in U consecutively up to time (t, z). It holds that*

$$\|Z_{t-1}v\|_2^2 + \|(X_{t-1} - X_{t_v-1})v\|_2^2 \le \frac{2\varepsilon^3}{k} w_{t-1}$$

Proof. We denote by $(w_U)_u$ the values of the w_U vector for the different directions u at time (t, z). Let λ_v be the eigenvalue associated with v during the time it was entered to U. Then at that time $\|Zv\|_2^2 = \|Cv\|_2 = \lambda_v$ (recall that v is chosen as a vector orthogonal to those of U hence $\|Cv\| = \|Z(I-U)v\|^2 = \|Zv\|^2$). Furthermore, since v was a column in U up to time t we get that all vectors added to R from the insertion of v up to time t are orthogonal to v. Lemma 13, item 3 shows that

$$\|Z_{t-1}v\|_2^2 \le \lambda_v.$$

Since v was a column vector of U from iteration t_v up to iteration t, we have that

$$\|Z_{t-1}v\|_2^2 + \|(X_{t-1} - X_{t_v-1})v\|_2^2 \leq \lambda_v + \sum_{\tau=t_v}^{t} \langle v, x_\tau \rangle^2 = (w_U)_v$$

It remains to bound the quantity of $(w_U)_v$. We will bound the sum $\sum_{u \in U_{t,z}} (w_U)_u$ and use the fact that v is the minimizer of the corresponding expression hence $(w_U)_v$ is upper bounded by $\sum_u (w_U)_u \cdot \frac{\varepsilon^3}{k}$.

Let t_u be the index of the iteration in which u is inserted into U. It is easy to verify that for $\lambda_u = \|C_{t_u-1}u\|$ it holds that

$$w_u = \lambda_u + \|(X_{t-1} - X_{t_u-1})u\|_2^2$$

Now, since $C \preceq Z \preceq R$ we have that

$$w_u = \|R_{t_u-1}u\|_2^2 + \|(X_{t-1} - X_{t_u-1})u\|_2^2$$

Finally,

$$\sum_u (w_U)_u \leq \sum_u \|(X_{t-1} - X_{t_u-1})u\|_2^2 + \|R_{t_u-1}u\|_2^2 \overset{(i)}{\leq}$$

$$\sum_u \|X_{t-1}u\|_2^2 + \|R_{t-1}u\|_2^2 \overset{(ii)}{\leq} \|X_{t-1}\|_F^2 + \|R_{t-1}\|_F^2 \overset{(iii)}{\leq} 2\|X_{t-1}\|_F^2 = 2w_{t-1}$$

Inequality (i) is immediate from the definitions of X_t, R_t as concatenations of t column vectors. Inequality (ii) follows since any orthogonal vector set and any matrix admit $\sum_u \|Au\|_2^2 \leq \|A\|_{\mathrm{F}}^2$. Inequality (iii) is due to the fact that each column of R is obtained by projection a column of X onto some subspace.

□

Lemma 17. *At all times* $\|C_{t,z}\|_2 \leq w_{t-1} \cdot \frac{\varepsilon^2}{k}$,

Proof. We prove the claim by induction over t, z. The base case for $t = 0$ is trivial. For $t > 0$, $z = 0$, if the while loop in iteration t was entered to we have that $C_{t,0} = C_{t-1,z}$ for some z. Since $w_{t-1} \geq w_{t-2}$ the claim holds. If t is such that the while loop of the iteration was not entered the condition of the while loop asserts that $\|C_{t,z}\| = \|C_t\| \leq w_{t-1} \cdot \frac{\varepsilon^2}{k}$.

Consider now $t, z > 0$. We have that

$$C_{t,z-1} = (I - U_{t,z-1}U_{t,z-1}^T)Z_{t-1}Z_{t-1}^T(I - U_{t,z-1}U_{t,z-1}^T)$$

$$C_{t,z} = (I - U_{t,z}U_{t,z}^T)Z_{t-1}Z_{t-1}^T(I - U_{t,z}U_{t,z}^T)$$

If $U_{t,z}$ is obtained by writing u instead of a zero column of $U_{t,z-1}$ then $C_{t,z}$ is a projection of $C_{t,z-1}$ and the claim holds due to the induction hypothesis. If not, u is inserted instead of some vector v. According to Lemmas 15 and 16, v is an eigenvector of $Z_{t-1}Z_{t-1}^T$ with eigenvalue $\lambda_v \leq w_{t-1} \cdot \frac{2\varepsilon^3}{k} \leq \frac{\varepsilon^2}{k}$, assuming $\varepsilon \leq 0.5$. It follows that $C_{t,z}$ is a projection of $C_{t,z-1} + \lambda_v vv^T$. Now, since $C_{t,z-1}v = 0$ (as v is a column vector of $U_{t,z-1}$) we have that

$$\|C_{t,z}\|_2 \leq \|C_{t,z-1} + \lambda_v vv^T\|_2 = \max\{\|C_{t,z-1}\|_2, \|\lambda_v vv^T\|_2\}$$

According to our induction hypothesis and the bound for λ_v, the above expression is bounded by $w_{t-1} \cdot \frac{\varepsilon^2}{k}$ as required. □

Lemma 18. *Let u be a vector that is not in $U_{t,z}$ and in $U_{t,z+1}$. Let λ be the eigenvector associated to it w.r.t $C_{t,z}$. It holds that $\lambda \leq w_{t-1} \cdot \frac{\varepsilon^2}{k}$. Furthermore, if u is a column vector in $U_{t',z'}$ for all $t' \geq t, z' \geq z$, it holds that $\|u^T Z_n\|^2 \leq \lambda \leq \|X_n\|_F^2 \cdot \frac{\varepsilon^2}{k}$.*

Proof. Since u is chosen as the top eigenvector of $C_{t,z}$ we have by Lemma 17 that

$$\lambda = \|C_{t,z}u\|_2 = \|C_{t,z}\|_2 \leq w_{t-1} \cdot \frac{\varepsilon^2}{k}$$

For the second claim in the lemma we note that since u is an eigenvector of $C_{t,z-1} = (I - U_{t,z-1}U_{t,z-1}^T)Z_{t-1}Z_{t-1}^T(I - U_{t,z-1}U_{t,z-1}^T)$, we have that $U_{t,z-1}^T u = 0$, hence

$$\|u^T Z_{t-1}\|_2^2 = \|u^T(I - U_{t,z-1}U_{t,z-1}^T)Z_{t-1}\|_2^2 = u^T C_{t,z} u \leq \|C_{t,z}u\|_2 \leq w_{t-1} \cdot \frac{\varepsilon^2}{k}$$

Since u is assumed to be an element of U throughout the running time of the algorithm, it holds that for all future vectors r added to the sketch Z, u is orthogonal to r. The claim now follows from Lemma 13 item 3. □

Lemma 19. $\|R_n\|_2^2 \leq \frac{2\varepsilon^2}{k}\|X_n\|_{\mathrm{F}}^2$.

Proof. Let $u_1, \ldots, u_\ell$ and $\lambda_1, \ldots, \lambda_\ell$ be the columns of U_n and their corresponding eigenvalues in C at the time of their addition to U. From Lemmas 15 and 18 we have that each u_j is an eigenvector of ZZ^T with eigenvalue $\lambda_j' \leq \lambda_j \leq \frac{\varepsilon^2}{k}\|X_n\|_F^2$. It follows that

$$\|Z_n Z_n^T\|_2 = \max\left\{\frac{\varepsilon^2}{k}\|X_n\|_F^2, \|(I - U_n U_n^T)Z_n Z_n^T(I - U_n U_n^T)\|_2\right\}$$

$$= \max\left\{\frac{\varepsilon^2}{k}\|X_n\|_F^2, \|C_n\|_2\right\} \leq \frac{\varepsilon^2}{k}\|X_n\|_F^2 \ .$$

The last inequality is due to Lemma 17.

Next, by the sketching property (Lemma 13 item 5), for appropriate matrix E: $Z_nZ_n^T = R_nR_n^T + E$, with $\|E\| \leq \frac{\varepsilon^2}{k}\|R_n\|_F^2$. As the columns of R are projections of those of X we have that $\|R_n\|_F^2 \leq \|X_n\|_F^2$, hence

$$\|R_n\|_2^2 = \|R_nR_n^T\|_2 = \|Z_nZ_n^T - E\|_2 \leq \|Z_nZ_n^T\|_2 + \|E\|_2 \leq \frac{2\varepsilon^2}{k}\|X_n\|_F^2$$

□

Lemma 20.

$$\|R_n\|_F^2 \leq \mathrm{OPT}_k + 4\varepsilon\|X_n\|_\mathrm{F}^2$$

Proof. The Lemma can be proven analogically to Theorem 1 as the only difference is the bound over $\|R_n\|_2^2$. □

Lemma 21. *Assume that for all t, $\|x_t\|_2^2 \leq w_t \cdot \frac{\varepsilon^2}{5k}$. Assume that $\varepsilon \leq 0.1$. For $\tau > 0$ consider the iterations of the algorithm during which $w_t \in [2^\tau, 2^{\tau+1})$. During this time, the while loop will be executed at most $5k/\varepsilon^2$ times.*

Proof. For the proof, we define a potential function

$$\Phi_{t,z} = \mathrm{Trace}(R_{t-1}R_{t-1}^T) - \mathrm{Trace}(C_{t,z}) \ .$$

We first notice that since C is clearly PSD,

$$\Phi_{t,z} \leq \mathrm{Trace}(R_{t-1}R_{t-1}^T) = \|R_{t-1}\|_F^2 \leq \|X_{t-1}\|_F^2 \leq 2^{\tau+1} \ .$$

The first inequality is since the columns of R are projections of those of X and the second is since $\|X_{t-1}\|_F^2 \leq w_{t-1}$. We will show that first, Φ is non-decreasing with time and furthermore, for valid $z > 0$, $\Phi_{t,z} \geq \Phi_{t,z-1} + 0.2\frac{\varepsilon^2}{k}2^{\tau+1}$. The result immediately follows.

Consider a pair (t, z) followed by the pair of indices $(t + 1, 0)$. Here, $\Phi_{t+1,0} - \Phi_{t,z} = \|r_t\|_2^2 \geq 0$ hence for such pairs the potential is non-decreasing. Now consider some pair (t, z) for $z > 0$. Since (t, z) is a valid pair it holds that

$$\begin{aligned}
\|C_{t,z-1}\|_2 &\geq \|C_{t,z-1} + r_{t,z-1}r_{t,z-1}^T - r_{t,z-1}r_{t,z-1}^T\|_2 \\
&\geq \|C_{t,z-1} + r_{t,z-1}r_{t,z-1}^T\|_2 - \|r_{t,z-1}r_{t,z-1}^T\|_2 \geq w_t\frac{\varepsilon^2}{k}(1-0.2) \\
&\geq 0.4 \cdot 2^{\tau+1}\frac{\varepsilon^2}{k} \qquad (11)
\end{aligned}$$

Denote by u the column vector in $U_{t,z}$ that is not in $U_{t,z-1}$. Let U' be the matrix obtained by appending the column u to the matrix $U_{t,z-1}$. Let

$$C' = (I - U'(U')^T)Z_{t-1}Z_{t-1}^T(I - U'(U')^T) = (I - uu^T)C_{t,z-1}(I - uu^T)$$

Since u is the top eigenvector of $C_{t,z-1}$ we have by equation (11) that

$$\text{Trace}(C') - \text{Trace}(C_{t,z-1}) = \|C_{t,z-1}\|_2 \geq 0.4 \cdot 2^{\tau+1}\frac{\varepsilon^2}{k}$$

If $U_{t,z-1}$ had a zero column then $C_{t,z} = C'$ and we are done. If not, let v be the vector that was replaced by u. According to Lemma 15, v is a singular vector of Z_{t-1}. According to Lemma 16 and $\varepsilon < 0.1$ we have that

$$\|Z_{t-1}v\|_2^2 \leq \frac{2\varepsilon^3}{k}\|X_{t-1}\|_F^2 \leq \frac{\varepsilon^2}{5k}2^{\tau+1} \ .$$

Hence,

$$C_{t,z} = C' + \|Z_{t-1}v\|^2 \cdot vv^T$$

meaning that

$$\text{Trace}(C_{t,z} - C_{t,z-1}) = \text{Trace}(C_{t,z} - C') + \text{Trace}(C' - C_{t,z-1}) \geq \frac{\varepsilon^2}{5k}2^{\tau+1} \ .$$

We conclude that as required Φ is non-decreasing over time and in each iteration of the while loop, increases by at least $\frac{\varepsilon^2}{5k}2^{\tau+1}$. Since Φ is upper bounded by $2^{\tau+1}$ during the discussed iterations, the lemma follows. □

Lemma 22. *Let $(v_1, t'_1+1, t_1+1), \ldots, (v_j, t'_j+1, t_j+1), \ldots$ be the sequence of triplets of vectors removed from U, the times on which they were added to U and the times on which they were removed from U.*

$$\text{ALG} \leq \left(\|R_n\|_F + 2\sqrt{\sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2}\right)^2 .$$

Proof. For any time t denote by U_t the matrix U in the end of iteration t, by $U_t^{(1)}$ the outcome of zeroing-out every column of U_t that is different from the corresponding column in U_n and by $U_t^{(2)}$ its complement, that is the outcome of zeroing-out every column in U_t that is identical to the corresponding column in U_n. In the same way define $U^{(2)}$ to be the outcome of zeroing-out columns in U_n that are all zeros in $U_t^{(2)}$.

It holds that,

$$\begin{aligned}\|x_t - U_n U_t^T x_t\|_2^2 &= \|x_t - (U_t + U_n - U_t)U_t^T x_t\|_2^2 \\ &\le (\|x_t - U_t U_t^T x_t\|_2 + \|(U_n - U_t)U_t^T x_t\|_2)^2 \\ &= \left(\|r_t\|_2 + \|(U^{(2)} - U_t^{(2)})(U_t^{(2)})^T x_t\|\right)^2 \\ &\le \left(\|r_t\|_2 + 2\|(U_t^{(2)})^T x_t\|_2\right)^2\end{aligned}$$

Summing over all times t we have,

$$\begin{aligned}\text{ALG} &= \sum_{t=1}^{n} \left(\|r_t\|_2 + 2\|(U_t^{(2)})^T x_t\|_2\right)^2 \\ &= \sum_{t=1}^{n} \|r_t\|_2^2 + 4\|r_t\|_2\|(U_t^{(2)})^T x_t\|_2 + 4\|(U_t^{(2)})^T x_t\|_2^2 \\ &\le \|R_n\|_{\text{F}}^2 + 4\sqrt{\sum_{t=1}^{n}\|r_t\|_2^2}\sqrt{\sum_{t=1}^{n}\|(U_t^{(2)})^T x_t\|_2^2} + 4\sum_t \|(U_t^{(2)})^T x_t\|_2^2\end{aligned}$$

Where the last inequality follows from applying the Cauchy-Schwarz inequality to the dot product between the vectors $(\|r_1\|_2, \|r_2\|_2, ..., \|r_n\|_2)$ and $(\|(U_1^{(2)})^T x_1\|_2, \|(U_2^{(2)})^T x_2\|_2, ..., \|(U_n^{(2)})^T x_n\|_2)$.

Since $U_t^{(2)}$ contains only vectors that were columns of U at time t but were replaced later, and are not present in U_n, we have that $\|(U_t^{(2)})^T x_t\|^2 \le \sum_{j: t_j > t > t'_j} (x_t^T v_j)^2$ and so $\sum_{t=1}^{n} \|(U_t^{(2)})^T x_t\|_2^2 \le \sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2$. Thus we have that,

$$\begin{aligned}\text{ALG} &\le \|R_n\|_{\text{F}}^2 + 4\|R_n\|_{\text{F}}\sqrt{\sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2} + 4\sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2 \\ &= \left(\|R_n\|_{\text{F}} + 2\sqrt{\sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2}\right)^2\end{aligned}$$

□

Lemma 23. *Let $(v_1, t'_1 + 1, t_1 + 1), \ldots, (v_j, t'_j + 1, t_j + 1), \ldots$ be the sequence of triplets of vectors removed from U, the times on which they were added to U and the times on which they were removed from U. Then*

$$\sum_j \|(X_{t_j} - X_{t'_j})v_j\|_2^2 \le 20\varepsilon\|X_n\|_F^2.$$

Proof. For some $\tau > 0$ consider the execution of the algorithm during the period in which $w_t \in [2^\tau, 2^{\tau+1})$. According to Lemma 21, at most $\frac{5k}{\varepsilon^2}$ vectors v were removed from the U during that period. According to Lemma 16, for each such v_j it holds that

$$\|(X_{t_j} - X_{t'_j})v\|_2^2 \leq w_{t_j} \cdot \frac{2\varepsilon^3}{k} \leq \frac{2\varepsilon^3}{k} 2^{\tau+1}$$

It follows that the contribution of vectors v thrown from the set during the discussed time period is at most

$$5\frac{k}{\varepsilon^2} \cdot \frac{2\varepsilon^3}{k} 2^{\tau+1} = 10\varepsilon \cdot 2^{\tau+1}$$

The entire sum can now be bounded by a geometric series, ending at $\tau = \log_2(\|X\|_F^2)$ thus proving the lemma. □

Corollary 1.

$$\mathrm{ALG}_{k,\varepsilon} \leq \left(\sqrt{\mathrm{OPT}_k} + \left(\sqrt{4} + \sqrt{20}\right)\sqrt{\varepsilon}\|X_n\|_F\right)^2 \leq \left(\sqrt{\mathrm{OPT}_k} + 6.48\sqrt{\varepsilon}\|X_n\|_F\right)^2$$

In particular, for $\varepsilon = \delta^2 \cdot \mathrm{OPT}_k / \|X_n\|_F^2$,

$$\mathrm{ALG}_{k,\varepsilon} = \mathrm{OPT}_k(1 + O(\delta))$$

Lemma 24 (Time complexity)**.** *Algorithm 2 requires $O(n\frac{dk}{\varepsilon^3} + \log(w_n/w_0)\frac{dk^3}{\varepsilon^6})$ arithmetic operations.*

Proof. We begin by pointing out that in the algorithm we work with the $d \times d$ matrices C and ZZ^T. These matrices have a bounded rank and furthermore, we are always able to maintain a representation of them of the form AA^T with A being a $d \times r$ matrix with r being the rank of C or ZZ^T. Hence, all computations involving them requires $O(dr)$ or $O(dr^2)$ arithmetic operations (corresponding to a product with a vector and computing the eigendecomposition).

Consider first the cost of the operations outside the while loop that do not involve the matrix C. It is easy to see, with Lemma 13, item 1, that the amortized time required in each iteration is $O(d\frac{k}{\varepsilon^3})$. The two operations of computing C and its top singular value may require $O(d\frac{k^2}{\varepsilon^6})$ time. However, these operations do not necessarily have to occur every iteration if we use the following trick: When entering the while loop we will require the norm of $C + r_t r_T^T$ to be bounded by $w_t\frac{\varepsilon^2}{k}$ rather than $w_t\frac{\varepsilon^2}{2k}$. Assume now that we entered the while loop at time t. In this case, we do not need to check the condition of the while loop until we reach an iteration t' where $w_{t'} \geq w_t(1 + \frac{\varepsilon^2}{2k})$. Other than checking the condition of the while loop there is no need to compute C, hence the costly operations of the external loop need only be executed an amount of

$$O(\log(w_n/w_0) \cdot \frac{k}{\varepsilon^2})$$

We now proceed to analyze the running time of the inner while loop. Each such iteration requires $O(d\frac{k^2}{\varepsilon^6})$ arithmetic operations. However, according to Lemma 21 we have that the total number of such iterations is bounded by

$$O(\log(w_n/w_0) \cdot \frac{k}{\varepsilon^2})$$

The lemma immediately follows.

□

Lemma 25 (Space complexity)**.** *Algorithm 1 requires $O(dk/\varepsilon^3)$ space.*

Proof. Immediate from the algorithm and the properties of the Frequent Direction Sketch (Lemma 13 item 2) □