Rivalry of Two Families of Algorithms for Memory-Restricted Streaming PCA

Chun-Liang Li
chunlial@cs.cmu.edu
Carnegie Mellon University

Hsuan-Ten Lin
htlin@csie.ntu.edu.tw
National Taiwan University

Chi-Jen Lu
cjlu@iis.sinica.edu.tw
Academia Sinica

Abstract

We study the problem of recovering the subspace spanned by the first k principal components of d-dimensional data under the streaming setting, with a memory bound of $\mathcal{O}(kd)$. Two families of algorithms are known for this problem. The first family is based on the framework of stochastic gradient descent. Nevertheless, the convergence rate of the family can be seriously affected by the learning rate of the descent steps and deserves more serious study. The second family is based on the power method over blocks of data, but setting the block size for its existing algorithms is not an easy task. In this paper, we analyze the convergence rate of a representative algorithm with decayed learning rate (Oja and Karhunen, 1985) in the first family for the general $k > 1$ case. Moreover, we propose a novel algorithm for the second family that sets the block sizes automatically and dynamically with faster convergence rate. We then conduct empirical studies that fairly compare the two families on real-world data. The studies reveal the advantages and disadvantages of these two families.

1 Introduction

For data points in $\mathbb{R}^d$, the goal of principal component analysis (PCA) is to find the first $k \ll d$ eigenvectors (principal components) that correspond to the top-k eigenvalues of the $d \times d$ covariance matrix. For a batch of stored data points with a moderate d, efficient algorithms like the power method (Golub and Van Loan, 1996) can be run on the empirical covariance matrix to compute the solution.

In addition to the batch algorithms, the stream setting (streaming PCA) is attracting much research attention in recent years (Arora et al., 2012; Mitliagkas et al., 2013; Hardt and Price, 2014). Streaming PCA assumes that each data point $\mathbf{x} \in \mathbb{R}^d$ arrives sequentially and it is not feasible to store all data points. If d is moderate, the empirical covariance matrix can again be computed and fed to an eigenproblem solver to compute the streaming PCA solution. When d is huge, however, it is not feasible to store the $\mathcal{O}(d^2)$ empirical covariance matrix. The situation arises in many modern applications of PCA. Those applications call for memory-restricted streaming PCA, which will be the main focus of this paper. We shall consider restricting to only $\mathcal{O}(kd)$ memory usage, which is of the same order as the minimum amount needed for the PCA solution. In addition, we aim to develop streaming PCA algorithms that can keep improving the goodness of the solution as more data points arrive. Such algorithms are free from a pre-specified goal of goodness and match the practical needs better.

There are two measurements for the goodness of the solution. One is the reconstruction error that measures the expected squared error when projecting a data point to the solution, which is based on the fact that the actual principal components should result in the lowest reconstruction error. The other is the spectral error that measures the difference between the subspace spanned by the solution and the subspace spanned by the actual principal components, which will be formally defined in Section 2. The spectral error enjoys a wide range of practical applications (Sa et al., 2015). In addition, note that when the k^{th} and $(k+1)^{th}$ eigenvalues are close, the solution that wrongly includes the $(k+1)^{th}$ engenvector instead of the k^{th} one may still reach a small reconstruction error, but the spectral error can be large. That is, the spectral error is somewhat harder to knock down and will be the main focus of this paper.

There are several existing streaming PCA algorithms, but not all of them focus on the spectral error *and* meet the memory restriction. For instance, Karnin and Liberty (2015) proposed an algorithm which considers the spectral error, but its space complexity is at least $\Omega(kd \log n)$, where n is the number of data points received. Warmuth and Kuzmin (2008); Nie et al. (2013); Boutsidis et al. (2015) propose the online algorithm with regret guarantee on the reconstruction error. Also, the space complexity

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

of (Nie et al., 2013) grows in the order of d^2. Arora et al. (2013) extended Arora et al. (2012) to derive convergence analysis for minimizing the reconstruction error along with a memory-efficient implementation, but the space complexity is not precisely guaranteed to meet $\mathcal{O}(kd)$. Shamir (2015) also focuses on the reconstruction-based error. That is, those works do not match the focus of this paper.

There are two families of algorithms that tackle the spectral error while respecting the memory restriction, the family of stochastic gradient descent (SGD) algorithms for PCA, and the family of block power methods. The SGD family solves a non-convex optimization problem that minimizes the reconstruction error, and applies SGD (Oja and Karhunen, 1985) under the memory restrictions to design streaming PCA algorithms. Interestingly, although the non-convex problem does not match standard convergence assumptions of SGD (Rakhlin et al., 2012), minimizing the reconstruction error for the special case of $k = 1$ allows Balsubramani et al. (2013) to derive spectral-error guarantees on the classic stochastic-gradient-descent PCA (SPCA) algorithm (Oja and Karhunen, 1985). Recently, Sa et al. (2015) derive a spectral-error minimization algorithm for the general $k > 1$ cases based on SGD along with strong theoretical guarantees. Nevertheless, different from Balsubramani et al. (2013), Sa et al. (2015) require a pre-specified error goal, which is taken to determine a *fixed* learning rate of the descent step. The pre-specified goal makes the algorithm inflexible in taking more data points to further decrease the error. Furthermore, the fixed learning rate is inevitably conservative to keep the algorithm stable, but the conservative nature results in slow convergence in practice, as will be revealed from the experimental results in Section 5.

The other family, namely the block power methods (Mitliagkas et al., 2013), extends the batch power method (Golub and Van Loan, 1996) for the memory-restricted streaming PCA by defining blocks (periods) on the time line. The key of the block power methods is to efficiently compute the product of the estimated covariance matrices in different blocks. The product serves as an approximation to the power of the empirical covariance matrix, which is a core element of the batch power method. This family could also be viewed as the mini-batch SGD algorithms but with different update rule from the SGD family. The original block-power-method PCA (BPCA; Mitliagkas et al., 2013) is proved to converge under some restricted distributions, which is later generalized by Hardt and Price (2014) to a broader class of distributions. The convergence proof of BPCA in both works, however, depends on determining the block size from the total number of data points or a pre-specified error goal, which again make the works inflexible for further decreasing the error with more data points.

From the theoretical perspective, SPCA lacks convergence proof for the general $k > 1$ case without depending on the pre-specified error goal nor the fixed learning rate, and it is non-trivial to directly extend the fixed-learning-rate result of Sa et al. (2015) to realize the proof; from the algorithmic perspective, BPCA needs more algorithmic study on deciding the block size without depending on the pre-specified error goal; from the practical perspective, it is not clear which family should be preferred in real-world applications. This paper makes contributions on all the three perspective. We first prove the convergence of SPCA for $k > 1$ with a decaying learning rate scheme in Section 3. The convergence result turns out to be asymptotically similar to the result of Sa et al. (2015) while not relying on the fixed learning rate. For convenience, we also call the proposed algorithm with the dynamic learning rate decay as SPCA. Then in Section 4, we propose a dynamic block power method (DBPCA) that automatically decides the block size to not only allow easier algorithmic use but also guarantee better convergence rate. Finally, we conduct experiments on real-world datasets and provide concrete recommendations in Section 5.

2 Preliminaries

Let us first introduce some notations which will be used later. First, let $x \leq \mathcal{O}(y)$ and $x \geq \Omega(y)$ denote that for some universal constant c, independent of all our parameters, $x \leq cy$ and $x \geq cy$, respectively, for a large enough y. Next, let $\lceil x \rceil$ denote the smallest integer that is at least x. Finally, for a vector $\mathbf{x}$, we let $\|\mathbf{x}\|$ denote its ℓ_2-norm, and for a matrix M, we let $\|M\| = \max_{\mathbf{x}} \frac{\|M\mathbf{x}\|}{\|\mathbf{x}\|}$, which is the spectral norm.

In this paper, we study the streaming PCA problem, in which with each input data point $\mathbf{x}_n \in \mathbb{R}^d$ is received at step n within a stream. Following previous works (Mitliagkas et al., 2013; Balsubramani et al., 2013), we make the following assumption on the data distribution.

Assumption 1. *Assume that each $\mathbf{x}_n$ is sampled independently from some distribution $\mathcal{X}$ with mean zero and covariance matrix A, which has eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, with $\lambda_k > \lambda_{k+1}$. Moreover, assume that $\|\mathbf{x}\| \leq 1$ for any $\mathbf{x}$ in the support of $\mathcal{X}$,[1] which implies that $\|A\| \leq 1$ and $\|\mathbf{x}_n\mathbf{x}_n^\top\| \leq 1$ for each n.*

Our goal is to find a $d \times k$ matrix Q_n at each step n, with its column-space quickly approaching that spanned by the first k eigenvectors of A. For convenience, we let $\lambda = \lambda_k$ and $\hat{\lambda} = \lambda_{k+1}$, and moreover, let U denote the $d \times k$ matrix with the first k eigenvectors of A as its columns. One common way to measure the distance between such two spaces is

$$\Phi_n = \max_{\mathbf{v} \in \mathbb{R}^k} \left(1 - \frac{\|U^\top Q_n \mathbf{v}\|^2}{\|\mathbf{v}\|^2}\right), \tag{1}$$

[1] We can relax this condition to that of having a small $\|\mathbf{x}\|$ with a high probability as Hardt and Price (2014) do, but we choose this stronger condition to simplify our presentation.

Algorithm 1 SPCA

$S_0 \sim \mathcal{N}(0,1)^{d\times k}$
$S_0 = Q_0 R_0$ (QR decomposition)
$n \leftarrow 1$
while receiving data **do**
  $S_n \leftarrow Q_{n-1} + \gamma_n \mathbf{x}_n \mathbf{x}_n^\top Q_{n-1}$
  $S_n = Q_n R_n$ (QR decomposition)
  $n \leftarrow n+1$
end while

which can be used as an error measure for Q_n. It is known that $\Phi_n = \sin\theta_k(U,Q_n)^2$, where $\theta_k(U,Q_n)$ is the k-th principal angle between these two spaces. For simplicity, we will denote $\sin\theta_k(U,Q_n)$ by $\sin(U,Q_n)$. Moreover, let $\cos(U,Q_n) = \sqrt{1-\sin(U,Q_n)^2}$ and $\tan(U,Q_n) = \sin(U,Q_n)/\cos(U,Q_n)$. It is also known that $\cos(U,Q_n)$ equals the smallest singular value of the matrix $U^\top Q_n$. More can be found in, e.g., Golub and Van Loan (1996).

Our algorithms will generate an initial matrix $S_0 \in \mathbb{R}^{d\times k}$ by sampling each of its entries independently from the normal distribution $\mathcal{N}(0,1)$. Let $S_0 \sim \mathcal{N}(0,1)^{d\times k}$ denote this process, and we will rely on the following guarantee.

Lemma 1. *(Mitliagkas et al., 2013) Suppose we sample $S_0 \sim \mathcal{N}(0,1)^{d\times k}$ and let $S_0 = Q_0 R_0$ be its QR decomposition. Then for a large enough constant $\bar{c}$, there is a small enough constant δ_0 such that $\Pr\left[\cos(U,Q_0) \le \sqrt{\bar{c}/(dk)}\right] \le \delta_0$.*

3 Stochastic Gradient Descent

In this section, we study the classic PCA algorithm framework of Oja and Karhunen (1985) for the general rank-k case, which can be seen as performing stochastic gradient descent. Our algorithm, called SPCA, is given in Algorithm 1. The key component is to determine the learning rate, which is related to the error analysis. We choose the step size at step n as

$$\gamma_n = \frac{c}{n}, \text{ with } c = \frac{c_0}{\lambda - \hat{\lambda}} \text{ for a constant } c_0 \ge 12.$$

The algorithm has a space complexity of $\mathcal{O}(kd)$, by noting that the computation of $\mathbf{x}_n \mathbf{x}_n^\top Q_{n-1}$ can be done by first computing $\mathbf{x}_n^\top Q_{n-1}$ and then multiplying the result by $\mathbf{x}_n$. The sample complexity of our algorithm is guaranteed by the following, which we prove in Subsection 3.1. Our analysis is inspired by and follows closely that of Balsubramani et al. (2013) for the rank-one case, but there are several new hurdles which we need to overcome in the general rank-k case.

Theorem 1. *For any $\rho \in (0,1)$, there is some $N \le (2^{1/(\lambda-\hat{\lambda})}kd)^{\mathcal{O}(1)} + \mathcal{O}\left(\frac{k\log(1/\rho)}{\rho(\lambda-\hat{\lambda})^3}\right)$, such that our algorithm with high probability can achieve $\Phi_n \le \rho$ for any $n \ge N$.*

Let us remark that we did not attempt to optimize the first term in the bound above, as it is dominated by the second term for a small enough ρ. Note that Sa et al. (2015) provided a better bound, which only has quadratic dependence of the eigengap $\lambda - \hat{\lambda}$, for a similar algorithm called Alecton. Alecton is restricted to taking a *fixed* learning rate that comes from a pre-specified error goal on a fixed amount of to-be-received data points. The restriction makes Alecton less practical in the streaming setting, because one may not always be able to know the amount of to-be-received data points in advance. If one receives fewer points than needed, Alecton cannot achieve the error goal; if one receives more than needed, Alecton cannot fully exploit the additional points for a smaller error. The decaying learning rate used by our proposed SPCA algorithm, on the other hand, does not suffer from such a restriction.

3.1 Proof of Theorem 1

The analysis of Balsubramani et al. (2013) works for the rank-one case by using a potential function $\Psi_n = 1 - (U^\top Q_n)^2$, where U and Q_n are both vectors instead of matrices. To work in the general rank-k case, we choose the function Φ_n defined in (1) as a generalization of their Ψ_n, and our goal is to bound Φ_n.

Following Balsubramani et al. (2013), we divide the steps into epochs, with epoch i ranging from step n_{i-1} to step $n_i - 1$, where we choose $n_0 = \hat{c}^c k^3 d^2 \log d$, for a large enough constant $\hat{c}$, and $n_i = \lceil e^{5/c_0} \rceil (n_{i-1}+1) - 1$ for $i \ge 1$.

Remark 1. *This gives us $(n_i+1) \ge e^{5/c_0}(n_{i-1}+1)$ and $n_i \le c_1 n_{i-1}$ for some constant c_1.*

As in Balsubramani et al. (2013), we also use the convention of starting from step n_0. For each epoch i, we would like to establish an upper bound ρ_i on Φ_n for each step n in that epoch. To start with, we know the following from Lemma 1, using the fact that $\Phi_0 = \sin(U,Q_0)^2 = 1 - \cos(U,Q_0)^2$.

Lemma 2. *Let Γ_0 denote the event that $\Phi_0 \le \rho_0$, where $\rho_0 = 1 - \bar{c}/(kd)$ for the constant $\bar{c}$ in Lemma 1. Then we have $\Pr[\neg\Gamma_0] \le \delta_0$.*

Next, for each epoch $i \ge 1$, we consider the event

$$\Gamma_i : \sup_{n_{i-1} \le n < n_i} \Phi_n \le \rho_i,$$

for some ρ_i to be specified later. Then our goal is to show that $\Pr[\neg\Gamma_{i+1} | \Gamma_i]$ is small, for $i \ge 0$. This can be done for the rank-one case, but it relies crucially on the property that the potential function Ψ_n of Balsubramani et al. (2013) satisfies a nice recurrence relation. Unfortunately, this does not appear so for our function Φ_n, mainly because it takes an additional maximization over $v \in \mathbb{R}^k$. To overcome this problem, we take the following approach.

Consider an epoch i and a step n in the epoch. Let us define a new matrix $Y_n = (I + \gamma_n \mathbf{x}_n \mathbf{x}_n^\top) Y_{n-1} = Q_n R_n R_{n-1} \cdots R_{n_{i-1}+1}$, with $Y_{n_{i-1}} = Q_{n_{i-1}}$. Let $\mathcal{S} = \{\mathbf{v} \in \mathbb{R}^k : \|\mathbf{v}\| = 1\}$. Then for any $\mathbf{v} \in \mathcal{S}$, define

$$\Phi_n^{(\mathbf{v})} = 1 - \frac{\|U^\top Y_n \mathbf{v}\|^2}{\|Y_n \mathbf{v}\|^2},$$

and note that $\Phi_n = \max_{\mathbf{v} \in \mathcal{S}} \Phi_n^{(\mathbf{v})}$. Now for each such new function $\Phi_n^{(\mathbf{v})}$, with a fixed $\mathbf{v}$, we can establish a similar recurrence relation as follows, but for our purpose later we show a better upper bound on $|Z_n|$ than that in Balsubramani et al. (2013). We give the proof in Appendix A.

Lemma 3. *For any $n > n_0$ and any $\mathbf{v} \in \mathcal{S}$, we have $\Phi_n^{(\mathbf{v})} \le \Phi_{n-1}^{(\mathbf{v})} + \beta_n - Z_n$, where*

1. $\beta_n = 5\gamma_n^2 + 2\gamma_n^3$
2. $|Z_n| \le 2\gamma_n \sqrt{\Phi_{n-1}^{(\mathbf{v})}}$
3. $\mathbb{E}\left[Z_n | \mathcal{F}_{n-1}\right] \ge 2\gamma_n (\lambda - \hat{\lambda}) \Phi_{n-1}^{(\mathbf{v})} (1 - \Phi_{n-1}^{(\mathbf{v})}) \ge 0$.[2]

With this lemma, the analysis of Balsubramani et al. (2013) can be used to show that $\mathbb{E}[\Phi_n^{(\mathbf{v})}]$ decreases as n grows, but only for each individual $\mathbf{v}$ separately. This alone is not sufficient to guarantee the event Γ_{i+1} as it requires small $\Phi_n^{(\mathbf{v})}$ for all $\mathbf{v}$'s simultaneously. To deal with this, a natural approach is to show that each $\Phi_n^{(\mathbf{v})}$ is large with a small probability, and then apply a union bound, but an apparent difficulty is that there are infinitely many $\mathbf{v}$'s. We will overcome this difficulty by showing how it is possible to apply a union bound only over a finite set of "ϵ-net" for these infinitely many $\mathbf{v}$'s. Still, for this approach to work, we need the probability of having a large $\Phi_n^{(\mathbf{v})}$ to be small enough, compared to the size of the ϵ-net. However, the beginning steps of the first epoch seem to have us in trouble already as the probability of their $\Phi^{(\mathbf{v})}$ values exceeding $\Phi_{n_0}^{(\mathbf{v})}$ is not small. This seems to prevent us from having an error bound $\rho_1 < \rho_0$, and without this to start, it is not clear if we could have smaller and smaller error bounds for later epochs. To handle this, we sacrifice the first epoch by using an error bound ρ_1 slightly larger than ρ_0, but still small enough. The hope is that once Γ_1 is established, we then have a period of small errors, and later epochs could then start to have decreasing ρ_i's. More precisely, we have the following for the first epoch, which we prove in Appendix B.

Lemma 4. *Let $\rho_1 = 1 - \bar{c}/(c_1^{6c} kd)$, for the constant c_1 given in Remark 1. Then $\Pr\left[\neg\Gamma_1 \mid \Gamma_0\right] = 0$.*

It remains to set the error bounds for later epochs appropriately so that we can actually have small $\Pr[\neg\Gamma_{i+1} | \Gamma_i]$, for $i \ge 1$. We let the error bounds decrease in three phases as follows. In the first phase, we let $\rho_i = 1 - 2(1 - \rho_{i-1})$, so that $\eta_i = 1 - \rho_i$ doubles each time. It ends at the first epoch i, denoted by π_1, such that $\rho_i < 3/4$. Note that $\pi_1 \le \mathcal{O}(\log d)$ and at this point, ρ_{π_1} is still much larger than $1/n_{\pi_1}$. Then in the second phase, we let $\rho_i = \rho_{i-1} / \lceil e^{5/c_0} \rceil^2$, which decreases in a faster rate than n_i increases. It ends at the first epoch i, denoted by π_2, such that $\rho_i \le c_2 (c^3 k \log n_{i-1})/(n_{i-1}+1)$, for some constant c_2.[3] Note that $\pi_2 \le \mathcal{O}(\log d)$ and at this point, ρ_{π_2} reaches about the order of $1/n_{\pi_2}$. Finally in phase three, we let $\rho_i = c_2 (c^3 k \log n_{i-1})/(n_{i-1}+1)$, which decreases in about the rate as n_i increases.

With these choices, the events Γ_i's are now defined, and our key lemma is the following, which we prove in Appendix C. The proof handles the difficulties above by showing how a union bound can be applied only on a small "ϵ-net" of $\mathcal{S}$ along with proper choices of ρ_i to guarantee that each $\Phi_n^{(\mathbf{v})}$ is large with a small enough probability.

Lemma 5. *For any $i \ge 1$, $\Pr\left[\neg\Gamma_{i+1} \mid \Gamma_i\right] \le \frac{\delta_0}{2(i+1)^2}$.*

From these lemmas, we can bound the failure probability of our algorithm as

$$\begin{aligned} \Pr\left[\exists i \ge 0 : \neg\Gamma_j\right] &\le \Pr\left[\neg\Gamma_0\right] + \textstyle\sum_{i \ge 0} \Pr\left[\neg\Gamma_{i+1} \mid \Gamma_i\right] \\ &\le \delta_0 + \textstyle\sum_{i \ge 0} \frac{\delta_0}{2(i+1)^2}, \end{aligned}$$

which is at most $2\delta_0$ using the fact that $\sum_{i \ge 1} 1/i^2 \le 2$.

To complete the proof, it remains to determine the number of samples needed by our algorithm to achieve an error bound ρ. This amounts to determine the number n_i of an epoch i with $\rho_i \le \rho$. With $n_{\pi_2} \ge n_0$, it is not hard to check that $\rho_{\pi_2} \le 1/(2^c kd)^{\mathcal{O}(1)}$ and $n_{\pi_2} \le (2^c kd)^{\mathcal{O}(1)}$. Then if $\rho \ge \rho_{\pi_2}$, we can certainly use n_{π_2} as an upper bound. If $\rho \le \rho_{\pi_2}$, it is not hard to check that with $n_i \le \mathcal{O}(c^3 k (1/\rho) \log(1/\rho))$, we can have $\rho_i \le \rho$. As $c = c_0/(\lambda - \hat{\lambda})$, this proves Theorem 1.

4 Block-Wise Power Method

In this section, we turn to study a different approach based on block-wise power methods. Our algorithm is modified from that of Mitliagkas et al. (2013) (referred as BPCA), which updates the estimate Q_n with a more accurate estimate of A using a block of samples, instead of one single sample as in our first algorithm. Our algorithm differs from BPCA by allowing different block sizes, instead of a fixed size. More precisely, we divide the steps into blocks, with block i consisting of steps from some interval I_i, and we use this block of $|I_i|$ samples to update our estimate from Q_{i-1} to Q_i. We will specify $|I_i|$ later in (3), which basically grows exponentially after some initial blocks. We call our algorithm DBPCA, as described in Algorithm 2.

[2] As in Balsubramani et al. (2013), $\mathcal{F}_{n-1}$ here denotes the σ-field of all outcomes up to and including step $n-1$.

[3] Determined later in the proof of Lemma 5 in Appendix C for

Algorithm 2 DBPCA

$S_0 \sim \mathcal{N}(0,1)^{d\times k}$
$S_0 = Q_0 R_0$ (QR-decomposition)
$i \leftarrow 1$
while receiving data **do**
  $S_i \leftarrow 0$
  for $n \in I_i$ **do**
    $S_i \leftarrow S_i + \frac{1}{|I_i|}\mathbf{x}_n\mathbf{x}_n^\top Q_{i-1}$
  end for
  $S_i = Q_i R_i$ (QR-decomposition)
  $i \leftarrow i+1$
end while

This algorithm, as our first algorithm SPCA, also has a space complexity of $\mathcal{O}(kd)$. The sample complexity is guaranteed by the following, which we will prove in Subsection 4.1. To have a easier comparison with the results of Mitliagkas et al. (2013) and Hardt and Price (2014), we use $\sqrt{\Phi_n} = \sin(U, Q_n)$ as the error measure in this section.

Theorem 2. *Given any $\varepsilon \le 1/\sqrt{kd}$, our algorithm can achieve an error ε with high probability after L iterations with a total of N samples, for some $L \le \mathcal{O}\left(\frac{\lambda}{\lambda-\hat{\lambda}}\log\frac{d}{\varepsilon}\right)$ and $N \le \mathcal{O}\left(\frac{\lambda\log(dL)}{\varepsilon^2(\lambda-\hat{\lambda})^3}\right)$.*

Let us make some remarks about the theorem. First, the error ρ in Theorem 1 corresponds to the error ε^2 here, and one can see that the bound in Theorem 2 is better than those in Theorem 1 and Mitliagkas et al. (2013); Hardt and Price (2014) in general. We summarize the sample complexity in terms of the error ε in Table 1. Next, the condition $\varepsilon \le 1/\sqrt{kd}$ in the theorem is only used to simplify the error bound. One can check that our analysis also works for any $\varepsilon \le 1$, but the resulting bound for N has the factor ε^2 replaced by $\min(1/(kd), \varepsilon^2)$. Finally, from Theorem 2, one can also express the error in terms of the number of samples n as $\varepsilon(n) \le \mathcal{O}\left(\sqrt{\frac{\lambda}{n(\lambda-\hat{\lambda})^3}\log\left(\frac{\lambda d\log n}{\lambda-\hat{\lambda}}\right)}\right)$.

4.1 Proof of Theorem 2

Recall that after the i-th block, we have the estimate Q_i, and we would like it to be close to U, with a small error $\sin(U, Q_i)$. To bound this error, we follow Hardt and Price (2014) and work on bounding a surrogate error $\tan(U, Q_i)$, which suffices as $\sin(U, Q_i) \le \tan(U, Q_i)$.

To start with, we know from Lemma 1 that for $\varepsilon_0 = \sqrt{\bar{c}/(kd)}$ with some constant $\bar{c}$, $\Pr[\tan(U, Q_0) > \varepsilon_0] \le \delta_0$, using the fact that $\tan(U, Q_0)^2 = 1/\cos(U, Q_0)^2 - 1$.

Next, we would like to bound each $\tan(U, Q_i)$ in terms of the previous $\tan(U, Q_{i-1})$. For this, recall that with $F_i = \sum_{n\in I_i}\mathbf{x}_n\mathbf{x}_n^\top/|I_i|$, we have $Q_i R_i = F_i Q_{i-1}$, which can be rewritten as $AQ_{i-1} + (F_i - A)Q_{i-1}$. Using the notation $G_i = (F_i - A)Q_{i-1}$, we have $Q_i R_i = AQ_{i-1} + G_i$, where G_i can be seen as the noise arising from estimating A by F_i using the i-th block of samples. Then, we rely on the following lemma from Hardt and Price (2014), with the parameters:

$$\bar{\lambda} = \max(\hat{\lambda}, \lambda/4), \gamma = \left(\bar{\lambda}/\lambda\right)^{1/4} \text{ and } \triangle = (\lambda - \bar{\lambda})/4. \tag{2}$$

Lemma 6. *(Hardt and Price, 2014) Suppose $\|G\| \le \triangle \cdot \min(\cos(U, Q), \beta)$, for some $\beta > 0$. Then*

$$\tan(U, AQ + G) \le \max(\beta, \max(\beta, \gamma)\tan(U, Q)).$$

From this, we can have the following lemma, proved in Appendix D, which provides an exponentially-decreasing upper bound on $\tan(U, Q_i)$, for the parameters:

$$\varepsilon_i = \varepsilon_0\gamma^i \text{ and } \beta_i = \min\left(\gamma/\sqrt{1+\varepsilon_{i-1}^2}, \gamma\varepsilon_{i-1}\right)$$

where $\varepsilon_0 = \sqrt{\bar{c}/(dk)}$ with the constant $\bar{c}$ in Lemma 1.

Lemma 7. *Suppose $\tan(U, Q_{i-1}) \le \varepsilon_{i-1}$ and $\|G_i\| \le \triangle\beta_i$. Then $\tan(U, Q_i) \le \varepsilon_i$.*

The key which sets our approach apart from that of Mitliagkas et al. (2013); Hardt and Price (2014) is the following observation. According to Lemma 7, for earlier iterations, one can in fact tolerate a larger $\|G_i\|$ and thus a larger empirical error for estimating A. This allows us to have smaller blocks at the beginning to save the number of samples, while still keeping the failure probability low. More precisely, we have the following lemma, proved in Appendix E, with the parameters:

$$\delta_i = \delta_0/(2i^2) \text{ and } |I_i| = (c/(\triangle\beta_i)^2)\log(d/\delta_i), \tag{3}$$

where δ_0 is the error probability given in Lemma 1 and c is a large enough constant.

Lemma 8. *For any $i \ge 1$, given $|I_i|$ samples in iteration i, we have $\Pr[\|G_i\| > \triangle\beta_i] \le \delta_i$.*

Algorithm	Complexity	Restriction
SGD family		
Balsubramani et al. (2013)	$\mathcal{O}(\frac{\log(1/\varepsilon)}{\varepsilon^2})$	only for $k=1$
Sa et al. (2015)	$\mathcal{O}(\frac{\log(1/\varepsilon)}{\varepsilon^2})$	pre-specified ε
our proposed SPCA	$\mathcal{O}(\frac{\log(1/\varepsilon)}{\varepsilon^2})$	none
block power method family		
Hardt and Price (2014)	$\mathcal{O}(\frac{\log(1/\varepsilon)}{\varepsilon^2})$	pre-specified ε
our proposed DBPCA	$\mathcal{O}\left(\frac{\log(\log(1/\varepsilon))}{\varepsilon^2}\right)$	none

Table 1: sample complexity and restriction

the bound (8) there to hold.

With this, we can bound the failure probability of our algorithm as

$$\Pr\left[\exists i \geq 0 : \tan(U, Q_i) > \varepsilon_i\right] \leq$$
$$\Pr\left[\tan(U, Q_0) > \varepsilon_0\right] + \textstyle\sum_{i\geq 1} \Pr\left[\|G_i\| > \triangle\beta_i\right]$$

which by Lemma 1 and Lemma 8 is at most $\delta_0 + \sum_{i\geq 1} \delta_i = \delta_0 + \sum_{i\geq 1} \frac{\delta_0}{2i^2} \leq 2\delta_0$.

To complete the proof of Theorem 2, it remains to bound the number of samples needed for achieving error ε. For this, we rely on the following lemma which we prove in Appendix F.

Lemma 9. *For some* $L \leq \mathcal{O}\left(\frac{\lambda}{\lambda-\bar{\lambda}} \log \frac{d}{\varepsilon}\right)$, *we have* $\varepsilon_L \leq \varepsilon$ *and* $\sum_{i=1}^{L} |I_i| \leq \mathcal{O}\left(\frac{\lambda \log(dL)}{\varepsilon^2(\lambda-\bar{\lambda})^3}\right)$.

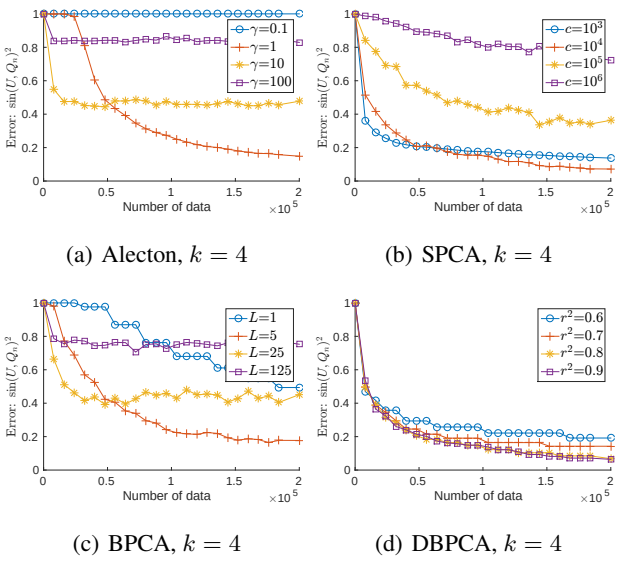
Finally, as $\bar{\lambda} = \max(\hat{\lambda}, \lambda/4)$, we have $\lambda - \bar{\lambda} \geq \Omega(\lambda - \hat{\lambda})$, and putting this into the bound above yields the sample complexity bound stated in the theorem.

5 Experiment

We conduct experiments on two large real-world datasets NYTimes and PubMed (Bache and Lichman, 2013) as used by Mitliagkas et al. (2013). The dimension d of the data points in the datasets are 102 and 141 thousands, respectively, which match our memory-restricted setting. The features of both datasets are normalized into $[0, 1]$.

Parameter tuning is generally difficult for streaming algorithms. Instead of tuning the parameters extensively and reporting with the most optimistic (but perhaps unrealistic) parameter choice for each algorithm, we consider a thorough range of parameters but report the results of four parameter choices per algorithm, which cover the best parameter choice, to understand each algorithm more deeply.

We compare the proposed SPCA and DBPCA with Alecton (fixed-learning-rate; Sa et al., 2015) and BPCA (fixed-block-size; Hardt and Price, 2014). For Alecton, we report the results of the learning rate $\gamma \in \{10^{-1}, \cdots, 10^{3}\}$, with reasons to be explained in Subsection 5.1. For SPCA, we follow its existing work (Balsubramani et al., 2013) to fix $n_0 = 0$ while considering $c \in \{10^{1}, \cdots, 10^{8}\}$. Then we report the results of $c \in \{10^{3}, 10^{4}, 10^{5}, 10^{6}\}$. For BPCA, we follow its existing works (Mitliagkas et al., 2013; Hardt and Price, 2014) and let the block size be $\lfloor N/T \rfloor$, where N is the size of the dataset and T is the number of blocks. Theoretical results of BPCA (Hardt and Price, 2014) suggest $T = \mathcal{O}\left(\frac{\lambda}{\lambda-\hat{\lambda}} \log d\right)$. Because λ and $\hat{\lambda}$ are unknown in practice, Mitliagkas et al. (2013); Hardt and Price (2014) set $T = \lfloor L \log d \rfloor$ with $L = 1$. Instead, we extend the range to $L \in \{5^{-1}, \cdots, 5^{6}\}$ and report the result of $\{5^0, 5^1, 5^2, 5^3\}$.



(a) Alecton, $k = 4$ (b) SPCA, $k = 4$

(c) BPCA, $k = 4$ (d) DBPCA, $k = 4$

Figure 1: Performance of different algorithms on NYTimes when $k = 4$

	$T = 10^5$	$T = 2 \times 10^5$
Alecton	0.232 ± 0.028	0.148 ± 0.008
SPCA	0.159 ± 0.008	0.079 ± 0.006
BPCA	0.234 ± 0.021	0.177 ± 0.012
DBPCA	$\mathbf{0.138 \pm 0.008}$	$\mathbf{0.064 \pm 0.005}$

Table 2: Performance of different algorithms with the best parameter on NYTimes when $k = 4$

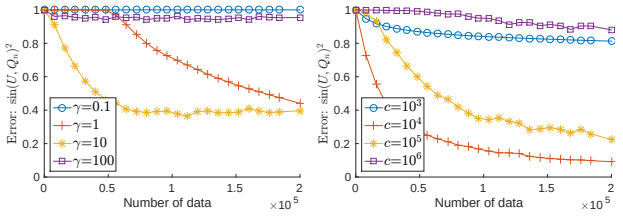
For the proposed DBPCA, we set the initial block size as $2k$ to avoid being rank-insufficient in the first block. Then, we consider the ratio $\gamma^2 \in \{0.6, 0.7, 0.8, 0.9\}$ for enlarging the block size. [4]

We run each algorithm 60 times by randomly generating data streams from the dataset. We consider $\sin(U, Q_n)^2$, which is the error function used for the convergence analysis, as the performance evaluation criterion. The average performance on the two datasets for $k = 4$ and $k = 10$ are shown in Figure 1, 2, 3 and 4, respectively. We observe the similar results on other k values and do not included here because of the space limit. Also, we report the mean and the standard error of each algorithm with the best parameters in Tables 2, 3, 4 and 5. To visualize the results clearly, we crop the figures up to $n = 200,000$, which is sufficient for checking the convergence of most of the parameter choices on the algorithms.

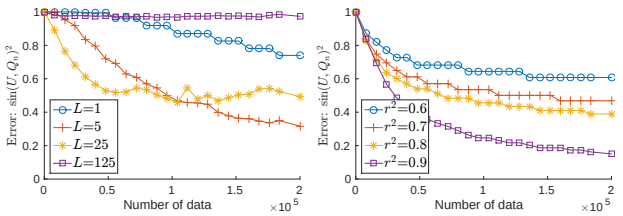
5.1 Comparison between SPCA and Alecton

The main difference between SPCA and Alecton is the rule of determining the learning rate. The learning rate of SPCA will decay along with the number of iterations,

[4] Note that (2) suggests $\gamma^2 \geq 0.5$.

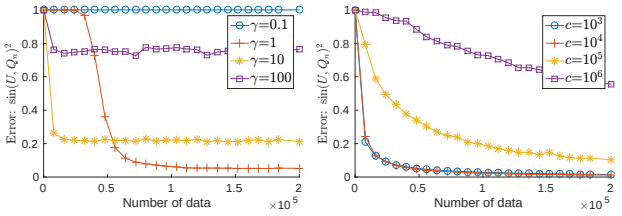


(a) Alecton, $k = 10$ (b) SPCA, $k = 10$

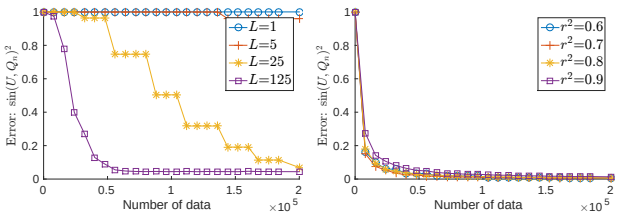


(c) BPCA, $k = 10$ (d) DBPCA, $k = 10$

Figure 2: Performance of different algorithms on NYTimes when $k = 10$



(a) Alecton, $k = 4$ (b) SPCA, $k = 4$



(c) BPCA, $k = 4$ (d) DBPCA, $k = 4$

Figure 3: Performance of different algorithms on PubMed when $k = 4$

	$T = 10^5$	$T = 2 \times 10^5$
Alecton	0.385 ± 0.013	0.386 ± 0.012
SPCA	$\mathbf{0.170 \pm 0.023}$	$\mathbf{0.102 \pm 0.018}$
BPCA	0.487 ± 0.042	0.317 ± 0.034
DBPCA	0.207 ± 0.028	0.151 ± 0.022

Table 3: Performance of different algorithms with the best parameter on NYTimes when $k = 10$

	$T = 10^5$	$T = 2 \times 10^5$
Alecton	0.051 ± 0.007	0.042 ± 0.000
SPCA	0.033 ± 0.000	0.022 ± 0.000
BPCA	0.045 ± 0.001	0.044 ± 0.000
DBPCA	$\mathbf{0.026 \pm 0.000}$	$\mathbf{0.013 \pm 0.000}$

Table 4: Performance of different algorithms with the best parameter on PubMed when $k = 4$

which means it could achieve arbitrarily small error when we have more data. On the other hand, Alecton needs to pre-specify the desired error to determine a fixed learning rate. To achieve the same error, from Table 1, SPCA and Alecton have the same asymptotic convergence rate theoretically. Next, we aim to study their empirical performance.

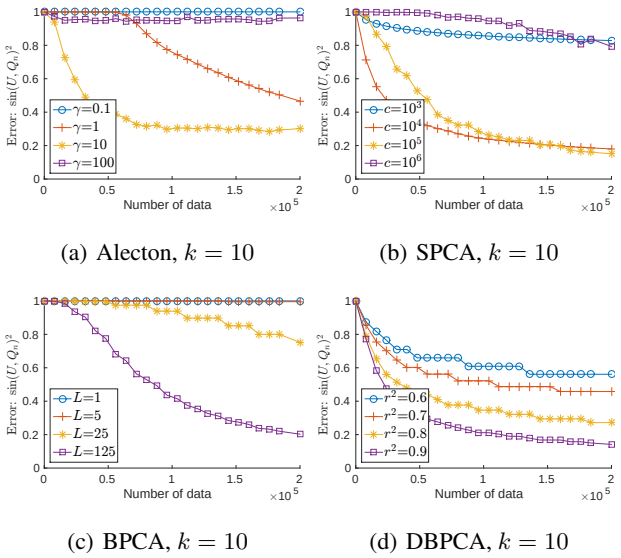
Sa et al. (2015) use a conservative rule to determine the learning rate. The upper bound of the learning rate γ suggested in Sa et al. (2015) is smaller than 10^{-5} for both datasets. However, this conservative and fixed learning rate scheme takes millions of iterations to converge to the competitive performance with SPCA. Similar results can also be found in Sa et al. (2015).

Although the suggested learning rate should be small, we still study performance of Alecton with larger learning rates, which are from 10^{-4} to 10^4. We report the results of $\{10^{-1}, 10^0, 10^1, 10^2\}$, which contain the optimal choices of the used datasets. Obviously, SPCA is generally better than Alecton, such as the case in Figure 1. From Tables 2, 3, 4 and 5, SPCA outperforms Alecton with the best parameters, which demonstrates the advantage of the decayed learning rate used by SPCA. From all figures, although Alecton with a larger learning rate ($\gamma = 10$) has a faster convergence behaviour at the beginning, it is stuck at a suboptimal point and can not utilize the new incoming data. The smaller learning rate could usually results in better performance in the end, but it takes more iterations than the number SPCA needs.

5.2 Comparison between DBPCA and BPCA

From Figure 1 and Figure 2, DBPCA outperforms BPCA under most parameter choices when $k = 4$, and is competitive to BPCA when $k = 10$. The edge of DBPCA over BPCA is even more remarkable in Figure 3 and Figure 4. From the result of the best parameters, DBPCA is significantly better than BPCA by t-test at 95% confidence.

BPCA has the similar drawback to Alecton. As can be observed from Figures 1, 2, 3 and 4, if L is too small (larger block), BPCA only sees one or two blocks of data within $n = 200,000$, and cannot reduce the error much. BPCA typically needs $L > 1$ (smaller blocks) to achieve lower error in the end. $L = 125$ gives the best performance of BPCA in Figures 3 and 4. However, sometimes large L (small blocks) in BPCA allows reducing the error in the beginning, the error cannot converge to a competitive level in the long run. For instance, in Figure 2(c), $L = 125$ con-



(a) Alecton, $k = 10$ (b) SPCA, $k = 10$

(c) BPCA, $k = 10$ (d) DBPCA, $k = 10$

Figure 4: Performance of different algorithms on PubMed when $k = 10$

	$T = 10^5$	$T = 2 \times 10^5$
Alecton	0.291 ± 0.007	0.292 ± 0.009
SPCA	0.274 ± 0.007	0.190 ± 0.040
BPCA	0.415 ± 0.037	0.203 ± 0.030
DBPCA	$\mathbf{0.212 \pm 0.024}$	$\mathbf{0.141 \pm 0.031}$

Table 5: Performance of different algorithms with the best parameter on PubMed when $k = 10$

verges fast but cannot improve much after $n = 50,000$; $L = 25$ converges slower but keeps going towards the lowest error after $n = 200,000$. Also, using smaller blocks cannot ensure reducing the error after each update, and hence BPCA with larger L results in less stable curves even after averaging over 60 runs. The results shows the difficulty of setting parameters of BPCA by the strategy proposed in Mitliagkas et al. (2013); Hardt and Price (2014).

On the other hand, DBPCA achieves better results by using a smaller block in the beginning to make improvements and a larger block later to further reduce the error. In both datasets and under all parameter choices, DBPCA stably reduces the error after each update, which matches our theoretical analysis that guarantees error reduction with a high probability. In addition, DBPCA is quite stable with respect to the choice of γ^2 across the two datasets, making it easier to tune in practice. The properties make DBPCA favorable over BPCA in the family of block power methods.

5.3 Comparison between DBPCA and SPCA

As observed, DBPCA is less sensitive to the parameter γ that corresponds to the theoretical suggestion of $\max(\hat{\lambda}/\lambda, 1/4)^{\frac{1}{4}}$. Somehow SPCA is rather sensitive to the parameter c that corresponds to the theoretical suggestion of $\frac{c_0}{\lambda - \hat{\lambda}}$. For instance, setting $c = 10^3$ results in strong performance when $k = 4$ in Figure 1(b), but the worst performance when $k = 10$ in Figure 2(b). Similar results can be observed in Figure 3(b) and Figure 4(b) when $c = 10^3$. Furthermore, the parameter c in SPCA directly affects the step size of each gradient descent update. Thus, compared with the best parameter choice, larger c leads to less stable performance curve, while smaller c sometimes results in significantly slower convergence. The results suggest that SPCA needs a more careful tuning and/or some deeper studies on proper parameter ranges.

From Tables 2, 3, 4 and 5, DBPCA significantly outperforms SPCA in 6 out of 8 cases by t-test under 95% confidence. The result supports the theoretical study that DBPCA has better converges rate guarantee than SPCA.

However, the benefit of SPCA is its immediate use of new data point. DBPCA, as a representative of the block-power-method family, cannot update the solution until the end of the growing block. Then, the latter points in the larger blocks may be effectively unused for a long period of time. For instance, in Figure 2, DBPCA uses larger blocks than the necessary size. After $N = 150,000$, the block size is near to $20,000$, which is less efficient.

6 Conclusion

We strengthen two families of streaming PCA algorithms, and compare the two strengthened families fairly from both theoretical and empirical sides. For the SGD family, we analyze the convergence rate of the famous SPCA algorithm for the multiple-principal-component cases without specifying the error in advance; for the family of block power methods, we propose a dynamic-block algorithm DBPCA that enjoys faster convergence rate than the original BPCA. Then, the empirical studies demonstrate that DBPCA not only outperforms BPCA often by dynamically enlarging the block sizes, but also converges to competitive results more stably than SPCA in many cases. Both the theoretical and empirical studies thus justify that DBPCA is the best among the two families, with the caveat of stalling the use of data points in larger blocks.

Our work opens some new research directions. Empirical results seem to suggest SPCA is competitive to or slightly worse than DBPCA. It is worth studying whether it is resulted from the substantial difference between $\log 1/\epsilon$ and $\log \log 1/\epsilon$ or caused by the hidden constants in the bounds. So one conjecture is that the bound in Theorem 1 can be further improved. On the other hand, although (2) suggests $\gamma^2 \geq 0.5$, the empirical results show that larger γ^2 generally results in better performance. Hence, it is also worth studying whether the lower bound could be further improved.

Acknowledgements

We thank the anonymous reviewers for valuable suggestions. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4012, and by the Ministry of Science and Technology of Taiwan under numbers MOST 103-2221-E-001-015-MY3 and 103-2221-E-002-148-MY3. The work was partially done when the first author was a joint research assistant at Academia Sinica and National Taiwan University.

References

Arora, R., Cotter, A., Livescu, K., and Srebro, N. (2012). Stochastic optimization for pca and pls. In *Annual Allerton Conference on Communication, Control, and Computing*.

Arora, R., Cotter, A., and Srebro, N. (2013). Stochastic optimization of pca with capped msg. In *Advances in Neural Information Processing Systems*.

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Balsubramani, A., Dasgupta, S., and Freund, Y. (2013). The fast convergence of incremental pca. In *Advances in Neural Information Processing Systems*.

Boutsidis, C., Garber, D., Karnin, Z. S., and Liberty, E. (2015). Online principal components analysis. In *Proceedings of the Symposium on Discrete Algorithms*.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press.

Hardt, M. and Price, E. (2014). The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems*.

Karnin, Z. and Liberty, E. (2015). Online pca with spectral bounds. In *Conference on Learning Theory*.

Mitliagkas, I., Caramanis, C., and Jain, P. (2013). Memory limited, streaming pca. In *Advances in Neural Information Processing Systems*.

Nie, J., Kotlowski, W., and Warmuth, M. K. (2013). Online pca with optimal regrets. In *International Conference on Algorithmic Learning Theory*.

Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*.

Rakhlin, A., Shamir, O., and Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In *International Conference on Machine Learning*.

Sa, C. D., Re, C., and Olukotun, K. (2015). Global convergence of stochastic gradient descent for some non-convex matrix problems. In *International Conference on Machine Learning*.

Shamir, O. (2015). Convergence of stochastic gradient descent for PCA. *CoRR*.

Warmuth, M. K. and Kuzmin, D. (2008). Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension. *Journal of Machine Learning Research*.