Faster Eigenvector Computation via Shift-and-Invert Preconditioning

Dan Garber DGARBER@TTIC.EDU
Toyota Technological Institute at Chicago

Elad Hazan EHAZAN@CS.PRINCETON.EDU
Princeton University

Chi Jin CHIJIN@BERKELEY.EDU
University of California, Berkeley

Sham M. Kakade SHAM@CS.WASHINGTON.EDU
University of Washington

Cameron Musco CNMUSCO@MIT.EDU
Massachusetts Institute of Technology

Praneeth Netrapalli PRANEETH@MICROSOFT.COM
Aaron Sidford ASID@MICROSOFT.COM
Microsoft Research, New England

Abstract

We give faster algorithms and improved sample complexities for the fundamental problem of estimating the top eigenvector. Given an explicit matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we show how to compute an ϵ approximate top eigenvector of $\mathbf{A}^\top \mathbf{A}$ in time $\widetilde{O}\left(\left[\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}(\mathbf{A})}{\mathrm{gap}^2}\right] \cdot \log 1/\epsilon\right)$. Here $\mathrm{nnz}(\mathbf{A})$ is the number of nonzeros in $\mathbf{A}$, $\mathrm{sr}(\mathbf{A})$ is the stable rank, and gap is the relative eigengap.

We also consider an online setting in which, given a stream of i.i.d. samples from a distribution $\mathcal{D}$ with covariance matrix $\mathbf{\Sigma}$ and a vector x_0 which is an $O(\mathrm{gap})$ approximate top eigenvector for $\mathbf{\Sigma}$, we show how to refine x_0 to an ϵ approximation using $O\left(\frac{\mathrm{v}(\mathcal{D})}{\mathrm{gap} \cdot \epsilon}\right)$ samples from $\mathcal{D}$. Here $\mathrm{v}(\mathcal{D})$ is a natural notion of variance. Combining our algorithm with previous work to initialize x_0, we obtain improved sample complexities and runtimes under a variety of assumptions on $\mathcal{D}$.

We achieve our results via a robust analysis of the classic *shift-and-invert* preconditioning method. This technique lets us reduce eigenvector computation to *approximately* solving a series of linear systems with fast stochastic gradient methods.

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

1. Introduction

Given $\mathbf{A} \in \mathbb{R}^{n \times d}$, computing the top eigenvector of $\mathbf{A}^\top \mathbf{A}$ is a fundamental problem in numerical linear algebra, applicable to principal component analysis (Jolliffe, 2002), spectral clustering and learning (Ng et al., 2002; Vempala & Wang, 2004), pagerank computation, and many other graph computations (Page et al., 1999; Koren, 2003; Spielman, 2007). For instance, a degree-k principal component analysis is nothing more than performing k leading eigenvector computations. Given the ever-growing size of modern datasets, it is thus a key challenge to come up with more efficient algorithms for this basic computational primitive.

In this work we provide improved algorithms for computing the top eigenvector, both in the *offline* case, when $\mathbf{A}$ is given explicitly and in the *online* or *statistical* case where we access samples from a distribution $\mathcal{D}$ over $\mathbb{R}^d$ and wish to estimate the top eigenvector of the covariance matrix $\mathbb{E}_{a \sim \mathcal{D}}\left[aa^\top\right]$. In the offline case, our algorithms are the fastest to date in a wide and meaningful regime of parameters. Notably, while the running time of most popular methods for eigenvector computations is a product of the size of the dataset (i.e. number of non-zeros in $\mathbf{A}$) and certain spectral characteristics of $\mathbf{A}$, which both can be quite large in practice, we present running times that actually split the dependency between these two quantities, and as a result may yield significant speedups. In the online case, our results yield improved sample complexity bounds and allow for very efficient *streaming* implementations with memory and processing-time requirements that are proportional to the size of a single sample.

On a high-level, our algorithms are based on a robust analysis of the classic idea of *shift-and-invert* preconditioning (Saad, 1992), which allows us to efficiently reduce eigenvector computation to *approximately* solving a *short* sequence of *well-conditioned* linear systems in $\lambda\mathbf{I}-\mathbf{A}^\top\mathbf{A}$ for some shift $\lambda \approx \lambda_1(\mathbf{A})$. We apply state-of-the-art stochastic gradient methods to approximately solve these linear systems. We believe our results suggest the general effectiveness of shift-and-invert based approaches and imply that further computational gains may be reaped in practice.

1.1. Our Approach

The well known power method for computing the top eigenvector of $\mathbf{A}^\top\mathbf{A}$ starts with an initial vector x and repeatedly multiplies by $\mathbf{A}^\top\mathbf{A}$, eventually causing x to converge to the top eigenvector. For a random start vector, convergence requires $O(\log(d/\epsilon)/\mathrm{gap})$ iterations, where $\mathrm{gap} = (\lambda_1 - \lambda_2)/\lambda_1$, λ_i denotes the i^{th} largest eigenvalue of $\mathbf{A}^\top\mathbf{A}$, and we assume a high-accuracy regime where $\epsilon < \mathrm{gap}$. The dependence on this gap ensures that the largest eigenvalue is significantly amplified in comparison to the remaining values.

If the eigenvalue gap is small, one approach is to replace $\mathbf{A}^\top\mathbf{A}$ with a preconditioned matrix – i.e. a matrix with the same top eigenvector but a much larger gap. Specifically, let $\mathbf{B} = \lambda\mathbf{I} - \mathbf{A}^\top\mathbf{A}$ for some shift parameter λ. If $\lambda > \lambda_1$, we can see that the smallest eigenvector of $\mathbf{B}$ (the largest eigenvector of $\mathbf{B}^{-1}$) is equal to the largest eigenvector of $\mathbf{A}^\top\mathbf{A}$. Additionally, if λ is close to λ_1, there will be a constant gap between the largest and second largest values of $\mathbf{B}^{-1}$. For example, if $\lambda = (1+\mathrm{gap})\lambda_1$, then we will have $\lambda_1\left(\mathbf{B}^{-1}\right) = \frac{1}{\lambda-\lambda_1} = \frac{1}{\mathrm{gap}\cdot\lambda_1}$ and $\lambda_2\left(\mathbf{B}^{-1}\right) = \frac{1}{\lambda-\lambda_2} = \frac{1}{2\cdot\mathrm{gap}\cdot\lambda_1}$.

This constant factor gap ensures that the power method applied to $\mathbf{B}^{-1}$ converges to the top eigenvector of $\mathbf{A}^\top\mathbf{A}$ in just $O(\log(d/\epsilon))$ iterations. Of course, there is a catch – each iteration of this *shifted-and-inverted power method* must solve a linear system in $\mathbf{B}$, whose condition number is proportional $\frac{1}{\mathrm{gap}}$. For small gap, solving this system via iterative methods is more expensive.

Fortunately, linear system solvers are incredibly well studied and there are many efficient iterative algorithms we can adapt to apply $\mathbf{B}^{-1}$ approximately. In particular, we show how to accelerate the iterations of the shifted-and-inverted power method using variants of Stochastic Variance Reduced Gradient (SVRG) (Johnson & Zhang, 2013). Due to the condition number of $\mathbf{B}$, we will not entirely avoid a $\frac{1}{\mathrm{gap}}$ dependence, however, we can separate this dependence from the input size $\mathrm{nnz}(\mathbf{A})$.

Typically, stochastic gradient methods are used to optimize convex functions that are given as the sum of many convex components. To solve a linear system $(\mathbf{M}^\top\mathbf{M})x = b$ we minimize the convex function $f(x) = \frac{1}{2}x^\top(\mathbf{M}^\top\mathbf{M})x - b^\top x$ with components $\psi_i(x) = \frac{1}{2}x^\top\left(m_i m_i^\top\right)x - \frac{1}{n}b^\top x$ where m_i is the i^{th} row of $\mathbf{M}$. Such an approach can be used to solve systems in $\mathbf{A}^\top\mathbf{A}$, however solving systems in $\mathbf{B} = \lambda\mathbf{I} - \mathbf{A}^\top\mathbf{A}$ requires more care. We require an analysis of SVRG that guarantees convergence even when some of our components are *non-convex*. We give a simple analysis for this setting, generalizing recent work in the area (Shalev-Shwartz, 2015; Csiba & Richtárik, 2015).

Given fast approximate solvers for $\mathbf{B}$, the second main piece of our algorithmic framework is a new error bound for the shifted-and-inverted power method, showing that it is robust to approximate linear system solvers, such as SVRG. We give a general analysis, showing exactly what accuracy each system must be solved to, allowing for faster implementations using linear solvers with weaker guarantees. Our proofs center around the potential function: $G(x) \stackrel{\mathrm{def}}{=} \left\|\mathbf{P}_{v_1^\perp}x\right\|_{\mathbf{B}} / \left\|\mathbf{P}_{v_1}x\right\|_{\mathbf{B}}$, where $\mathbf{P}_{v_1}$ and $\mathbf{P}_{v_1^\perp}$ are the projections onto the top eigenvector and its complement respectively. This function resembles tangent based potential functions used in previous work (Hardt & Price, 2014) except that we use the $\mathbf{B}$ norm rather than the ℓ_2 norm. For the exact power method, this is irrelevant – progress is identical in both norms (see Lemma 37 of our full version). However, $\|\cdot\|_{\mathbf{B}}$ is a natural norm for measuring the progress of linear system solvers for $\mathbf{B}$, so our potential function makes it possible to extend analysis to the case when $\mathbf{B}^{-1}x$ is computed up to error ξ with bounded $\|\xi\|_{\mathbf{B}}$.

1.2. Our Results

Our algorithmic framework described above offers several advantages. We obtain improved running times for computing the top eigenvector in the offline model. In Theorem 16 we give an algorithm running in time

$$O\left(\left[\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}\,\mathbf{A}}{\mathrm{gap}^2}\right]\cdot\left[\log\frac{1}{\epsilon} + \log^2\frac{d}{\mathrm{gap}}\right]\right)$$

where $\mathrm{sr}(\mathbf{A}) = \|\mathbf{A}\|_F^2 / \|\mathbf{A}\|_2^2 \le \mathrm{rank}(\mathbf{A})$ is the stable rank and $\mathrm{nnz}(\mathbf{A})$ is the number of non-zero entries. Up to log factors, our runtime is in many settings proportional to the input size $\mathrm{nnz}(\mathbf{A})$, and so is very efficient for large matrices. In the case when $\mathrm{nnz}(\mathbf{A}) \le \frac{d\,\mathrm{sr}(\mathbf{A})}{\mathrm{gap}^2}$ we apply the results of (Frostig et al., 2015b; Lin et al., 2015) to provide an accelerated runtime of:

$$O\left(\left[\frac{\mathrm{nnz}(\mathbf{A})^{\frac{3}{4}}(d\,\mathrm{sr}(\mathbf{A}))^{\frac{1}{4}}}{\sqrt{\mathrm{gap}}}\right]\left[\log\frac{d}{\mathrm{gap}}\log\frac{1}{\epsilon} + \log^3\frac{d}{\mathrm{gap}}\right]\right).$$

Finally, in the case when $\epsilon > \mathrm{gap}$, our results easily extend to give gap-free bounds (Theorems 34 and 35 of our full paper), identical to those shown above but with gap replaced by ϵ. Note that our offline results hold for any $\mathbf{A}$ and require no initial knowledge of the top eigenvector. In Section 6 we discuss how to estimate the parameters λ_1, gap, with modest additional runtime cost. Our algorithms return an approximate top eigenvector x with $x^\top \mathbf{A}^\top \mathbf{A} x \geq (1-\epsilon)\lambda_1$. By choosing error $\epsilon \cdot \mathrm{gap}$, we can ensure that x is actually close to v_1 – i.e. that $|x^\top v_1| \geq 1-\epsilon$. Further, we obtain the same asymptotic runtime since $O\left(\log \frac{1}{\epsilon \cdot \mathrm{gap}} + \log^2 \frac{d}{\mathrm{gap}}\right) = O\left(\log \frac{1}{\epsilon} + \log^2 \frac{d}{\mathrm{gap}}\right)$. We compare our runtimes with previous work in Table 1.

In the online case, in Theorem 25, we show how to improve an $O(\mathrm{gap})$ approximation to the top eigenvector to an ϵ approximation with constant probability using $O\left(\frac{\mathrm{v}(\mathcal{D})}{\mathrm{gap}\cdot\epsilon}\right)$ samples where $\mathrm{v}(\mathcal{D})$ is a natural variance measure. Our algorithm is based on the streaming SVRG algorithm of (Frostig et al., 2015a). It requires just $O(d)$ amortized time per sample, uses just $O(d)$ space, and is easily parallelized. We can apply our result in a variety of regimes, using existing algorithms to obtain the initial $O(\mathrm{gap})$ approximation and our algorithm to refine this solution. As shown in Table 2, this gives improved runtimes and sample complexities over existing work. Notably, we give better asymptotic sample complexity than known matrix concentration results for general distributions, and give the first streaming algorithm that is asymptotically optimal in the popular Gaussian spike model.

Our robust shifted-and-inverted power method analysis provides new understanding of this widely implemented technique. It gives a means of obtaining provably accurate results when each iteration is implemented using solvers with weak accuracy guarantees. In practice, this reduction between approximate linear system solving and eigenvector computation shows that optimized regression libraries can be leveraged for faster eigenvector computation in many cases. Furthermore, in theory we believe that the reduction suggests computational limits inherent in eigenvector computation as seen by the often easier-to-analyze problem of linear system solving. Indeed, in Section 7 of our full paper we provide evidence that in certain regimes our statistical results are optimal.

1.3. Previous Work

Offline Eigenvector Computation

Due to its universal applicability, eigenvector computation in the offline case is extremely well studied. Classical methods, such as the QR algorithm, take roughly $O(nd^2)$ time to compute a full eigendecomposition. They can be accelerated using fast matrix multiplication (Williams, 2012; Le Gall, 2014), however remain prohibitively expensive for large matrices. Hence, faster iterative methods are often employed, especially when only the top eigenvector (or a few of the top eigenvectors) is desired.

As discussed, the popular power method requires $O\left(\log(d/\epsilon)/\mathrm{gap}\right)$ iterations to converge to an ϵ approximate top eigenvector. Using Chebyshev iteration or the Lanczos method, this bound can be improved to $O\left(\log(d/\epsilon/\sqrt{\mathrm{gap}}\right)$ (Saad, 1992), giving total runtime of $O\left(\mathrm{nnz}(\mathbf{A}) \cdot \log(d/\epsilon)/\sqrt{\mathrm{gap}}\right)$. When $\epsilon > \mathrm{gap}$, the gap terms in these runtimes can be replaced by ϵ. We focus on the high-precision regime when $\epsilon < \mathrm{gap}$, but also give gap-free bounds in Section 8 of our full paper.

Unfortunately, if $\mathrm{nnz}(\mathbf{A})$ is very large and gap is small, the above runtimes can still be quite expensive, and there is a natural desire to separate the $1/\sqrt{\mathrm{gap}}$ dependence from the $\mathrm{nnz}(\mathbf{A})$ term. One approach is to use random subspace embedding matrices (Ailon & Chazelle, 2009; Clarkson & Woodruff, 2013) or fast row sampling algorithms (Cohen et al., 2015), which can be applied in $O(\mathrm{nnz}(\mathbf{A}))$ time and yield a matrix $\tilde{\mathbf{A}}$ which is a good spectral approximation to the original. The number of rows in $\tilde{\mathbf{A}}$ depends only on the stable rank of $\mathbf{A}$ and the error of the embedding. Applying such a subspace embedding and then computing the top eigenvector of $\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}}$ requires runtime $O\left(\mathrm{nnz}(\mathbf{A}) + \mathrm{poly}(\mathrm{sr}(\mathbf{A}), \epsilon, \mathrm{gap})\right)$, achieving the goal of reducing runtime dependence on the input size $\mathrm{nnz}(\mathbf{A})$. Unfortunately, the dependence on ϵ is significantly suboptimal – such an approach cannot be used to obtain a linearly convergent algorithm. Further, the technique does not extend naturally to the online setting.

Another approach, which we follow more closely, is to apply stochastic optimization techniques, which iteratively update an estimate to the top eigenvector, considering a random row of $\mathbf{A}$ with each update step. Such algorithms naturally extend to the online setting and have led to improved dependence on the input size for a variety of problems (Bottou, 2010). Using variance-reduced stochastic gradient techniques, (Shamir, 2015c) achieves a runtime bound assuming an upper bound on the squared row norms of $\mathbf{A}$. In the *best case*, when row norms are uniform, this runtime can be simplified to $O\left(\left(\mathrm{nnz}(\mathbf{A}) + d\,\mathrm{sr}(\mathbf{A})^2/\mathrm{gap}^2\right) \cdot \log(1/\epsilon)\log\log(1/\epsilon)\right)$. This result makes an important contribution in separating input size and gap dependencies using stochastic optimization techniques. Unfortunately, the algorithm requires an approximation to the eigenvalue gap and a starting vector that has a constant dot product with the top eigenvector. Initializing with a random vector loses polynomial factors in d (Shamir, 2015b), on top of the already suboptimal dependencies on $\mathrm{sr}(\mathbf{A})$ and ϵ.

Algorithm	**Runtime**
Power Method	$O\left(\mathrm{nnz}(\mathbf{A})\frac{\log(d/\epsilon)}{\mathrm{gap}}\right)$
Lanczos Method	$O\left(\mathrm{nnz}(\mathbf{A})\frac{\log(d/\epsilon)}{\sqrt{\mathrm{gap}}}\right)$
Fast Subspace Embeddings (Clarkson & Woodruff, 2013) + Lanczos	$O\left(\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}(\mathbf{A})}{\max\{\mathrm{gap}^{2.5}\epsilon, \epsilon^{2.5}\}}\right)$
SVRG (Shamir, 2015c) (assuming bounded row norms and warm-start)	$O\left(\left(\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}(\mathbf{A})^2}{\mathrm{gap}^2}\right)\cdot\log(1/\epsilon)\log\log(1/\epsilon)\right)$
Theorem 16	$O\left(\left[\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}(\mathbf{A})}{\mathrm{gap}^2}\right]\cdot\left[\log\frac{1}{\epsilon} + \log^2\frac{d}{\mathrm{gap}}\right]\right)$
Theorem 17 (full paper)	$O\left(\left[\frac{\mathrm{nnz}(\mathbf{A})^{3/4}(d\,\mathrm{sr}(\mathbf{A}))^{1/4}}{\sqrt{\mathrm{gap}}}\right]\cdot\left[\log\frac{d}{\mathrm{gap}}\log\frac{1}{\epsilon} + \log^3\frac{d}{\mathrm{gap}}\right]\right)$

Table 1. Comparision to previous work on Offline Eigenvector Estimation. We give runtimes for computing a unit vector x such that $x^\top \mathbf{A}^\top \mathbf{A} x \geq (1-\epsilon)\lambda_1$ in the regime $\epsilon = O(\mathrm{gap})$.

ONLINE EIGENVECTOR COMPUTATION

In the online, or statistical setting, research often looks beyond runtime. One focus is on minimizing the sample size required to achieve a given accuracy. Another common focus is on obtaining *streaming algorithms*, which use just $O(d)$ space - proportional to the size of a single sample.

In this section, in order to more easily compare to previous work, we normalize $\lambda_1 = 1$ and assume we have the row norm bound $\|a\|_2^2 \leq O(d)$ which then gives us the variance bound $\left\|\mathbb{E}_{a\sim\mathcal{D}}\left[(aa^\top)^2\right]\right\|_2 = O(d)$. Additionally, we compare runtimes for computing some x such that $|x^\top v_1| \geq 1-\epsilon$, as this is the most popular guarantee studied in the literature. Theorem 25 is easily extended to this setting as obtaining x with $x^T \mathbf{A}\mathbf{A}^\top x \geq (1-\epsilon\cdot\mathrm{gap})\lambda_1$ ensures $|x^\top v_1| \geq 1-\epsilon$. Our algorithm requires $O(d/(\mathrm{gap}^2\epsilon))$ samples to find such a vector under the above assumptions.

The simplest algorithm in this setting is to take n samples from $\mathcal{D}$ and compute the leading eigenvector of the empirical estimate $\widehat{\mathbb{E}}[aa^\top] = \frac{1}{n}\sum_{i=1}^n a_i a_i^\top$. By a matrix Bernstein bound (Tropp, 2015), $O\left(\frac{d\log d}{\mathrm{gap}^2\epsilon}\right)$ samples is enough to insure $\left\|\widehat{\mathbb{E}}[aa^\top] - \mathbb{E}[aa^\top]\right\|_2 \leq \sqrt{\epsilon}\cdot\mathrm{gap}$. By Lemma 36 in our full version, this ensures that, if x is set to the top eigenvector of $\widehat{\mathbb{E}}[aa^\top]$ it will satisfy $|x^\top v_1| \geq 1-\epsilon$.

A large body of work focuses on improving this simple algorithm. In Table 2 we give a sampling of results, all which rely on distributional assumptions at least as strong as those given above. Note that, in each setting, we can use the cited algorithm to first compute an $O(\mathrm{gap})$ approximate eigenvector, and then refine this approximation with our streaming from Theorem 25 using $O\left(\frac{d}{\mathrm{gap}^2\epsilon}\right)$ samples. This gives us improved runtime and sample complexity results. Notably, by the lower bound in Section 7 of our full paper, in all settings considered in Table 2, we achieve optimal asymptotic sample complexity - as our sample size grows large, ϵ decreases at an optimal rate. To save space, we do not show our improved runtime bounds, but they are easy to derive by adding the runtime required by the given algorithm to achieve $O(\mathrm{gap})$ accuracy to $O\left(\frac{d^2}{\mathrm{gap}^2\epsilon}\right)$ – the runtime of our streaming algorithm.

The bounds given for the simple matrix Bernstein based algorithm described above, Krasulina/Oja's Algorithm (Balsubramani et al., 2013), and SGD (Shamir, 2015a) require no additional assumptions, aside from those given at the beginning of this section. The streaming results cited for (Mitliagkas et al., 2013) and (Hardt & Price, 2014) assume a is generated from a Gaussian spike model, where $a_i = \sqrt{\lambda_1}\gamma_i v_1 + Z_i$ and $\gamma_i \sim \mathcal{N}(0,1), Z_i \sim \mathcal{N}(0, I_d)$. We note that under this model, the matrix Bernstein results improve by a $\log d$ factor and so match our results in achieving asymptotically optimal convergence rate. The results of (Mitliagkas et al., 2013) and (Hardt & Price, 2014) sacrifice this optimality in order to operate under the streaming model. Our work gives the best of both works – a streaming algorithm giving asymptotically optimal results.

(Sa et al., 2015) assumes $\mathbb{E}\left\|aa^\top \mathbf{W} aa^\top\right\| \leq O(1)\mathrm{tr}(\mathbf{W})$ for any symmetric $\mathbf{W}$ that commutes with $\mathbb{E}aa^\top$. This is much stronger than the assumption above that $\left\|\mathbb{E}_{a\sim\mathcal{D}}\left[(aa^\top)^2\right]\right\|_2 = O(d)$ and there are easy examples where the above assumption holds while theirs does not.

1.4. Organization

While our general approach is simple, our proofs are quite involved, and hence, most are omitted from the main paper. They can be found in our full paper. In Section 2 we review problem definitions and parameters. In Section 3 we describe the shifted-and-inverted power method and show how it can be implemented using approximate system solvers. In Section 4 we instantiate this framework, showing how to apply SVRG to solve systems in our shifted matrix and giving our main offline results. In Section 5 we discuss extending these techniques to the online setting. Finally, in Section 6 we discuss how to efficiently estimate the shift parameters required by our algorithms.

Algorithm	Sample Size	Runtime	Streaming?	Our Sample Complexity
Matrix Bernstein + Lanczos (explicitly forming matrix)	$O\left(\frac{d\log d}{gap^2\epsilon}\right)$	$O\left(\frac{d^3\log d}{gap^2\epsilon}\right)$	$\times$	$O\left(\frac{d\log d}{gap^3}+\frac{d}{gap^2\epsilon}\right)$
Matrix Bernstein + Lanczos (iteratively applying matrix)	$O\left(\frac{d\log d}{gap^2\epsilon}\right)$	$O\left(\frac{d^2\log d\cdot\log(d/\epsilon)}{gap^{2.5}\epsilon}\right)$	$\times$	$O\left(\frac{d\log d}{gap^3}+\frac{d}{gap^2\epsilon}\right)$
Memory-efficient PCA (Mitliagkas et al., 2013), (Hardt & Price, 2014)	$O\left(\frac{d\log(d/\epsilon)}{gap^3\epsilon}\right)$	$O\left(\frac{d^2\log(d/\epsilon)}{gap^3\epsilon}\right)$	$\surd$	$O\left(\frac{d\log(d/\mathrm{gap})}{gap^4}+\frac{d}{gap^2\epsilon}\right)$
Alecton (Sa et al., 2015)	$O(\frac{d\log(d/\epsilon)}{gap^2\epsilon})$	$O\left(\frac{d^2\log(d/\epsilon)}{gap^2\epsilon}\right)$	$\surd$	$O(\frac{d\log(d/\mathrm{gap})}{gap^3}+\frac{d}{gap^2\epsilon})$
Krasulina / Oja's Algorithm (Balsubramani et al., 2013)	$O\left(\frac{d^{c_1}}{gap^2\epsilon^{c_2}}\right)$	$O\left(\frac{d^{c_1+1}}{gap^2\epsilon^{c_2}}\right)$	$\surd$	$O\left(\frac{d^{c_1}}{gap^{2+c_2}}+\frac{d}{gap^2\epsilon}\right)$
SGD (Shamir, 2015a)	$O\left(\frac{d^3\log(d/\epsilon)}{\epsilon^2}\right)$	$O\left(\frac{d^4\log(d/\epsilon)}{\epsilon^2}\right)$	$\surd$	$O\left(\frac{d^3\log(d/\mathrm{gap})}{\mathrm{gap}^2}+\frac{d}{\mathrm{gap}^2\epsilon}\right)$

Table 2. Summary of existing Online Eigenvector Estimation results. Bounds are for computing a unit vector x with $|x^\top v_1| \geq 1-\epsilon$. For each result, we obtain improved bounds by running the algorithm to compute an $O(\mathrm{gap})$ approximate eigenvector, and then using our algorithm to obtain an ϵ approximation using an additional $O\left(\frac{d}{\mathrm{gap}^2\epsilon}\right)$ samples, $O(d)$ space, and $O(d)$ work per sample.

2. Preliminaries

In the *Offline Setting* we are given $\mathbf{A} \in \mathbb{R}^{n\times d}$ with rows $a^{(1)}, ..., a^{(n)}$ and wish to compute an approximation to the top eigenvector of $\mathbf{\Sigma} \stackrel{\text{def}}{=} \mathbf{A}^\top\mathbf{A}$. Specifically, for error ϵ we want a unit vector x with $x^\top\mathbf{\Sigma}x \geq (1-\epsilon)\lambda_1(\mathbf{\Sigma})$.

In the *Online Setting* we access an oracle returning independent samples from distribution $\mathcal{D}$ on $\mathbb{R}^d$. We wish to approximate the top eigenvector of $\mathbf{\Sigma} \stackrel{\text{def}}{=} \mathbb{E}_{a\sim\mathcal{D}}\left[aa^\top\right]$, specifically, to find unit x with $x^\top\mathbf{\Sigma}x \geq (1-\epsilon)\lambda_1(\mathbf{\Sigma})$.

2.1. Problem Parameters

Let $\lambda_1, ..., \lambda_d$ denote the eigenvalues of $\mathbf{\Sigma}$ and $v_1, ..., v_d$ denote their corresponding eigenvectors. We define the *eigenvalue gap* by $\mathrm{gap} \stackrel{\text{def}}{=} (\lambda_1-\lambda_2)/\lambda_1$. Our bounds also employ the following parameters:

In the *Offline Setting*: Let $\mathrm{sr}(\mathbf{A}) \stackrel{\text{def}}{=} \|\mathbf{A}\|_F^2 / \|\mathbf{A}\|_2^2$ denote the stable rank of $\mathbf{A}$. Note that we always have $\mathrm{sr}(\mathbf{A}) \leq \mathrm{rank}(\mathbf{A})$. Let $\mathrm{nnz}(\mathbf{A})$ denote the number of nonzero entries in $\mathbf{A}$.

In the *Online Setting*: Let $\mathrm{v}(\mathcal{D}) \stackrel{\text{def}}{=} \frac{\left\|\mathbb{E}_{a\sim\mathcal{D}}\left[(aa^\top)^2\right]\right\|_2}{\|\mathbb{E}_{a\sim\mathcal{D}}(aa^\top)\|_2^2} = \frac{\left\|\mathbb{E}_{a\sim\mathcal{D}}\left[(aa^\top)^2\right]\right\|_2}{\lambda_1^2}$ denote a natural upper bound on the variance of $\mathcal{D}$ in various settings. Note that $\mathrm{v}(\mathcal{D}) \geq 1$.

3. Algorithmic Framework

Here we overview our robust shift-and-invert framework, focusing on intuition, with proofs relegated to our full paper. We let $\mathbf{B} \stackrel{\text{def}}{=} \lambda\mathbf{I} - \mathbf{\Sigma}$, and assume that we have a crude estimate of λ_1 and gap so can set λ to satisfy $\left(1+\frac{\mathrm{gap}}{150}\right)\lambda_1 \leq \lambda \leq \left(1+\frac{\mathrm{gap}}{100}\right)\lambda_1$. (See Section 6 for how we can compute such a λ). Note that $\lambda_i\left(\mathbf{B}^{-1}\right) = \frac{1}{\lambda_i(\mathbf{B})} = \frac{1}{\lambda-\lambda_i}$ and so $\frac{\lambda_1(\mathbf{B}^{-1})}{\lambda_2(\mathbf{B}^{-1})} = \frac{\lambda-\lambda_2}{\lambda-\lambda_1} \geq \frac{\mathrm{gap}}{\mathrm{gap}/100} = 100$. This large gap ensures that, assuming the ability to apply $\mathbf{B}^{-1}$, the power method will converge very quickly.

3.1. Potential Function

Our eigenvector algorithms aim to maximize the Rayleigh quotient, $x^\top\mathbf{\Sigma}x$ for unit x. However, to track the progress of our algorithm we use a different potential function. We define for $x \neq 0$:

$$G(x) \stackrel{\text{def}}{=} \frac{\left\|\mathbf{P}_{v_1^\perp}x\right\|_{\mathbf{B}}}{\left\|\mathbf{P}_{v_1}x\right\|_{\mathbf{B}}} = \frac{\sqrt{\sum_{i\geq 2}\alpha_i^2/\lambda_i(\mathbf{B}^{-1})}}{\sqrt{\alpha_1^2/\lambda_1(\mathbf{B}^{-1})}}. \qquad (1)$$

where $\mathbf{P}_{v_1}$ and $\mathbf{P}_{v_1^\perp}$ denote the projections onto v_1 and the subspace orthogonal to v_1 and $\alpha_i = v_i^\top x$.

When the Rayleigh quotient error $\epsilon = \lambda_1 - x^\top\mathbf{\Sigma}x$ is small, we can show that $G(x)$ closely tracks ϵ, so we can analyze our algorithms exclusively in terms of $G(x)$ and then convert the resulting bounds to Rayleigh quotient error (see Lemma 3 of full paper).

3.2. Power Iteration

It is easy to see that the shifted-and-inverted power iteration makes progress with respect to our objective function given an exact linear system solver for $\mathbf{B}$.

Theorem 4. *Let x be a unit vector with $\langle x, v_1\rangle \neq 0$ and let $\widetilde{x} = \mathbf{B}^{-1}x$, i.e. the power method update of $\mathbf{B}^{-1}$ on x. Then, under our assumption on λ, we have:*

$$G(\widetilde{x}) \leq \frac{\lambda_2\left(\mathbf{B}^{-1}\right)}{\lambda_1\left(\mathbf{B}^{-1}\right)}G(x) \leq \frac{1}{100}G(x).$$

Proof. Writing x in the eigenbasis, we have $x = \sum_i \alpha_i v_i$ and $\widetilde{x} = \sum_i \alpha_i\lambda_i\left(\mathbf{B}^{-1}\right)v_i$. Since $\langle x, v_1\rangle \neq 0$, $\alpha_1 \neq 0$

and by the equivalent formulation of $G(x)$ given in (1):

$$G(\widetilde{x}) = \frac{\sqrt{\sum_{i\geq 2} \alpha_i^2 \lambda_i(\mathbf{B}^{-1})}}{\sqrt{\alpha_1^2 \lambda_1(\mathbf{B}^{-1})}}$$
$$\leq \frac{\lambda_2\left(\mathbf{B}^{-1}\right)}{\lambda_1\left(\mathbf{B}^{-1}\right)} \cdot \frac{\sqrt{\sum_{i\geq 2} \alpha_i^2 / \lambda_i(\mathbf{B}^{-1})}}{\sqrt{\alpha_1^2/\lambda_1(\mathbf{B}^{-1})}} = \frac{\lambda_2\left(\mathbf{B}^{-1}\right)}{\lambda_1\left(\mathbf{B}^{-1}\right)} G(x).$$

Recalling that $\lambda_1\left(\mathbf{B}^{-1}\right)/\lambda_2\left(\mathbf{B}^{-1}\right) \geq \frac{\mathrm{gap}}{\mathrm{gap}/100} = 100$ yields the result. □

In the next section we show that Theorem 4 can be made robust – we still make progress on our objective function even if we only approximate $\mathbf{B}^{-1}x$ using a fast linear system solver.

3.3. Approximate Power Iteration

We can show that each iteration of the shifted-and-inverted power method makes constant expected progress on $G(x)$ assuming we:

1. Start with a sufficiently good x and an approximation of λ_1
2. Can apply $\mathbf{B}^{-1}$ approximately using a system solver such that the function error (i.e. distance to $\mathbf{B}^{-1}x$ in the $\mathbf{B}$ norm) is sufficiently small *in expectation*.
3. Can estimate Rayleigh quotients over $\mathbf{\Sigma}$ well enough to only accept updates that do not hurt progress on the objective function too much.

Note that the second assumption is very weak. An expected progress bound allows, for example, the solver to occasionally return a solution that is entirely orthogonal to v_1, causing us to make unbounded backwards progress. The third assumption allows us to reject possibly harmful updates and ensure that we still make progress in expectation. In the offline setting, we can access $\mathbf{A}$ and are able to compute Rayleigh quotients trivially. However, we only assume the ability to estimate quotients, since in the online setting we only have access to $\mathbf{\Sigma}$ through samples from $\mathcal{D}$.

Theorem 5 (Approximate Shifted-and-Inverted Power Iteration – Warm Start)**.** *Let $x = \sum_i \alpha_i v_i$ be a unit vector such that $G(x) \leq \frac{1}{\sqrt{10}}$. Suppose we have an estimate $\widehat{\lambda}_1$ of λ_1 such that $10/11\left(\lambda - \lambda_1\right) \leq \lambda - \widehat{\lambda}_1 \leq \lambda - \lambda_1$. Furthermore, suppose we have a subroutine* $\mathrm{solve}(\cdot)$ *such that on any input x*

$$\mathbb{E}\left[\left\|\mathrm{solve}\left(x\right) - \mathbf{B}^{-1}x\right\|_{\mathbf{B}}\right] \leq \frac{c_1}{1000}\sqrt{\lambda_1(\mathbf{B}^{-1})},$$

for some $c_1 < 1$, and a subroutine $\widehat{\mathrm{quot}}\left(\cdot\right)$ that on any input $x \neq 0$ satisfies $\left|\widehat{\mathrm{quot}}\left(x\right) - \mathrm{quot}(x)\right| \leq \frac{1}{30}\left(\lambda - \lambda_1\right)$ where $\mathrm{quot}(x) \stackrel{\mathrm{def}}{=} \frac{x^\top \mathbf{\Sigma} x}{x^\top x}$. Let $\widehat{x} = \mathrm{solve}\left(x\right)$. Then the following update procedure:

$$\text{Set } \widetilde{x} = \begin{cases} \widehat{x} & \text{if } \begin{cases} \widehat{\mathrm{quot}}\left(\widehat{x}\right) \geq \widehat{\lambda}_1 - \left(\lambda - \widehat{\lambda}_1\right)/6 \text{ and} \\ \left\|\widehat{x}\right\|_2 \geq \frac{2}{3}\frac{1}{\lambda - \widehat{\lambda}_1} \end{cases} \\ x & \text{otherwise,} \end{cases}$$

satisfies: $G(\widetilde{x}) \leq \frac{1}{\sqrt{10}}$ and $\mathbb{E}\left[G(\widetilde{x})\right] \leq \frac{3}{25}G(x) + \frac{c_1}{500}$.

That is, not only do we decrease our potential function by a constant factor in expectation, but we are guaranteed that the potential function will never increase beyond $1/\sqrt{10}$. Roughly, the proof of Theorem 5 shows that, conditioning on accepting our iterative step:

$$\mathbb{E}\left[G(\widehat{x})\right] = \mathbb{E}\left[\frac{\left\|\mathbf{P}_{v_1^\perp}\mathbf{B}^{-1}x\right\|_{\mathbf{B}}}{\left\|\mathbf{P}_{v_1}\mathbf{B}^{-1}x\right\|_{\mathbf{B}}}\right]$$
$$\leq \frac{G(x)}{50} + 2\frac{\mathbb{E}\left[\left\|\widehat{x} - \mathbf{B}^{-1}x\right\|_{\mathbf{B}}\right]}{\sqrt{\lambda_1(\mathbf{B}^{-1})}}$$

That is, the potential function decreases as in the exact case (Theorem 4) with additional additive error due to the inexact linear system solve.

Theorem 5 assumes that we can solve linear systems to some absolute accuracy in expectation. However, system solvers typically only guarantee relative progress bounds with respect to an initial estimate of $\mathbf{B}^{-1}x$. Fortunately, we can show that approximating $\mathbf{B}^{-1}x$ with $\frac{1}{x^\top \mathbf{B} x}x$, and applying a linear system solver that improves this estimate by a constant factor in expectation gives small enough error to make progress in each power iteration:

Corollary 6 (Relative Error Linear System Solvers)**.** *For any unit vector x, we have:*

$$\left\|\frac{1}{x^\top \mathbf{B} x}x - \mathbf{B}^{-1}x\right\|_{\mathbf{B}} \leq \alpha_1\sqrt{\lambda_1(\mathbf{B}^{-1})} \cdot G(x),$$

so instantiating Theorem 5 with $c_1 = \alpha_1 G(x)$ gives $\mathbb{E}[G(\widetilde{x})] \leq \frac{4}{25}G(x)$ as long as:

$$\mathbb{E}\left[\left\|\mathrm{solve}\left(x\right) - \mathbf{B}^{-1}x\right\|_{\mathbf{B}}\right] \leq \frac{\left\|\frac{1}{\lambda - x^\top \mathbf{\Sigma} x}x - \mathbf{B}^{-1}x\right\|_{\mathbf{B}}}{1000}.$$

3.4. Initialization

Theorem 5 and Corollary 6 show that, given a good enough approximation to v_1, we can rapidly refine this approximation by applying the shifted-and-inverted power method with very coarse approximate linear system solves. To obtain the initial approximation, we rely on a 'burn-in' period in which we solve each linear system to higher accuracy. During burn-in, we may have a very small component of x in the direction of v_1, and so require higher accuracy to ensure that we do not 'lose' this component.

Theorem 8 (Approximate Shifted-and-Inverted Power Method – Burn-In)**.** *Suppose we initialize $x_0 \sim \mathcal{N}(0, \mathbf{I})$ and suppose we have access to a subroutine* solve$(\cdot)$ *such that*

$$\mathbb{E}\left[\left\|\text{solve}(x) - \mathbf{B}^{-1}x\right\|_{\mathbf{B}}\right] \leq \frac{1}{3000\kappa(\mathbf{B}^{-1})d^{21}} \cdot \left\|\frac{1}{\lambda - x^\top \mathbf{\Sigma} x}x - \mathbf{B}^{-1}x\right\|_{\mathbf{B}},$$

where $\kappa(\mathbf{B}^{-1}) = \frac{\lambda_1(\mathbf{B}^{-1})}{\lambda_d(\mathbf{B}^{-1})} = O(\frac{1}{\text{gap}})$. Then iteratively computing: $x_t = \text{solve}(x_{t-1}) / \|\text{solve}(x_{t-1})\|_2$, for $T = O(\log(d/\text{gap}))$, we have $G(x_T) \leq \frac{1}{\sqrt{10}}$ with probability $1 - O(\frac{1}{d^{10}})$.

4. Offline Eigenvector Computation

We now discuss how to instantiate the framework of Section 3 in the offline setting. We adapt Stochastic Variance Reduced Gradient (SVRG) (Johnson & Zhang, 2013) to solve linear systems in $\mathbf{B}$. To solve a system in the positive definite matrix $\mathbf{A}^\top\mathbf{A}$, one optimizes the objective function $f(x) = \frac{1}{2}x^\top\mathbf{A}^\top\mathbf{A}x - b^\top x$. This function is the sum of n *convex components*, $\psi_i(x) = \frac{1}{2}x^\top\left(a_i a_i^\top\right)x - \frac{1}{n}b^\top x$. In each iteration of traditional gradient descent, one computes the full gradient of $f(x_i)$ and takes a step in that direction. In stochastic gradient methods, at each iteration, a single component is sampled, and the step direction is based only on the gradient of the sampled component. Hence, a full gradient computation is avoided at each iteration, leading to runtime gains.

Unfortunately, while we have access to the rows of $\mathbf{A}$ and so can solve systems in $\mathbf{A}^\top\mathbf{A}$, it is less clear how to solve systems in $\mathbf{B} = \lambda\mathbf{I} - \mathbf{A}^\top\mathbf{A}$. To do this, we will split our function into components of the form $\psi_i(x) = \frac{1}{2}x^\top\left(w_i\mathbf{I} - a_i a_i^\top\right)x - \frac{1}{n}b^\top x$ for some set of weights w_i with $\sum_{i\in[n]} w_i = \lambda$.

Importantly, $w_i\mathbf{I} - a_i a_i^\top$ may not be positive semidefinite. We are minimizing a sum of functions which is convex, but consists of *non-convex components*. While recent results for minimizing such functions could be applied directly (Shalev-Shwartz, 2015; Csiba & Richtárik, 2015), in our full paper we obtain stronger results by using a more general form of SVRG and analyzing the specific properties of our function.

Our analysis shows that we can make constant factor progress in solving linear systems in $\mathbf{B}$ in time $O\left(\text{nnz}(\mathbf{A}) + \frac{d\,\text{sr}(\mathbf{A})}{\text{gap}^2}\right)$. If $\frac{d\,\text{sr}(\mathbf{A})}{\text{gap}^2} \leq \text{nnz}(\mathbf{A})$ this gives a runtime proportional to the input size – the best we could hope for. If not, we show that it is possible to *accelerate* our system solver using the results of (Frostig et al., 2015b; Lin et al., 2015), achieving runtime $O\left(\frac{\text{nnz}(\mathbf{A})^{3/4}(d\,\text{sr}(\mathbf{A}))^{1/4}}{\sqrt{\text{gap}}} \cdot \log\left(\frac{d}{\text{gap}}\right)\right)$. We show how to use the unaccelerated system solvers to obtain our main offline result. The analysis is identical in the accelerated case.

Theorem 16 (Shifted-and-Inverted Power Method With SVRG)**.** *Let $\mathbf{B} = \lambda\mathbf{I} - \mathbf{A}^\top\mathbf{A}$ for $\left(1 + \frac{\text{gap}}{150}\right)\lambda_1 \leq \lambda \leq \left(1 + \frac{\text{gap}}{100}\right)\lambda_1$ and let $x_0 \sim \mathcal{N}(0, \mathbf{I})$ be a random initial vector. Running the inverted power method on $\mathbf{B}$ initialized with x_0, using the SVRG solver from Theorem 12 (in our full paper) to approximately apply $\mathbf{B}^{-1}$ at each step, returns x such that with probability $1 - O\left(\frac{1}{d^{10}}\right)$, $x^\top\mathbf{\Sigma}x \geq (1-\epsilon)\lambda_1$ in total time*

$$O\left(\left(\text{nnz}(\mathbf{A}) + \frac{d\,\text{sr}(\mathbf{A})}{\text{gap}^2}\right) \cdot \left(\log^2\left(\frac{d}{\text{gap}}\right) + \log\left(\frac{1}{\epsilon}\right)\right)\right).$$

Instantiating the theorem with $\epsilon' = \epsilon \cdot \text{gap}$, we can find a unit vector x with $|v_1^\top x| \geq 1 - \epsilon$ in the same asymptotic running time (an extra $\log(1/\text{gap})$ term is absorbed into the $\log^2(d/\text{gap})$ term).

Proof. By Theorem 8, if we start with $x_0 \sim \mathcal{N}(0, \mathbf{I})$ we can run $O\left(\log\left(\frac{d}{\text{gap}}\right)\right)$ iterations of the inverted power method, to obtain x_1 with $G(x_1) \leq 1/\sqrt{10}$ with probability $1 - O(1/d^{10})$. Each iteration requires applying an linear solver that decreases initial error in expectation by a factor of $\frac{1}{\text{poly}(d, 1/\text{gap})}$. Such a solver is given by applying our constant factor solver $O(\log(d/\text{gap}))$ times. So overall in order to find x_1 with $G(x_1) \leq 1/\sqrt{10}$, we require time $O\left(\left(\text{nnz}(\mathbf{A}) + \frac{d\,\text{sr}(\mathbf{A})}{\text{gap}^2}\right) \cdot \log^2\left(\frac{d}{\text{gap}}\right)\right)$.

After this initial 'burn-in' period we can apply Corollary 6 of Theorem 5. In each iteration, we only need to use a solver that decreases initial error by a constant factor in expectation so requires time $O\left(\text{nnz}(\mathbf{A}) + \frac{d\,\text{sr}(\mathbf{A})}{\text{gap}^2}\right)$. With $O(\log(d/\epsilon))$ iterations, we can obtain x with $\mathbb{E}\left[G(x)^2\right] = O(\epsilon/d^{10})$, which is sufficient for the result. Note that the second stage requires $O\left(\log\left(\frac{d}{\epsilon}\right)\right) = O(\log d + \log(1/\epsilon))$ iterations to achieve the high probability bound. However, the $O(\log d)$ term is smaller than the $O\left(\log^2\left(\frac{d}{\text{gap}}\right)\right)$ term, so is absorbed into the asymptotic notation. □

5. Online Eigenvector Computation

We now discuss how to extend our results to the online setting. This setting is somewhat more difficult since there is no canonical $\mathbf{A}$, – we only have access to the distribution $\mathcal{D}$ through samples. In order to apply Theorem 5 we must show how to both estimate the Rayleigh quotient as well as solve the requisite linear systems in expectation.

Our Rayleigh quotient estimation procedure is standard – we first approximate the Rayleigh quotient by its empirical value on a batch of k samples and prove using Chebyshev's inequality that the error on this sample is small with constant probability. We then repeat this procedure $O(\log(1/p))$ times and output the median, obtaining a good estimate probability $1-p$.

Theorem 18 (Online Rayleigh Quotient Estimation)**.** *Given $\epsilon \in (0,1]$, $p \in [0,1]$, and unit vector x set $k = \lceil 4\,\mathrm{v}(\mathcal{D})\epsilon^{-2}\rceil$ and $m = O(\log(1/p))$. For all $i \in [k]$ and $j \in [m]$ let $a_i^{(j)}$ be drawn independently from $\mathcal{D}$ and set $R_{i,j} = x^\top a_i^{(j)}(a_i^{(j)})^\top x$ and $R_j = \frac{1}{k}\sum_{i\in[k]} R_{i,j}$. If we let z be median value of the R_j then with probability $1-p$ we have $\left|z - x^\top \mathbf{\Sigma} x\right| \leq \epsilon\lambda_1$.*

The next challenge is to solve linear systems in $\mathbf{B}$ in the streaming setting. We follow the general strategy of the offline algorithms given in Section 4, replacing traditional SVRG with the streaming SVRG algorithm of (Frostig et al., 2015a). Whereas in the offline case, we could ensure that our initial error $\|x_0 - x^{\text{opt}}\|_{\mathbf{B}}^2$ is small by simply scaling by the Rayleigh quotient (Corollary 6) in the online case estimating the Rayleigh quotient to sufficient accuracy would require too many samples. Instead, we simply show how to use streaming SVRG to solve the desired linear systems to absolute accuracy without an initial point. Ultimately, due to the different error dependences in the online case this guarantee suffices and the lack of an initial point is not a bottleneck.

Corollary 24 (Streaming SVRG Solver)**.** *Given a linear system $\mathbf{B}x = b$ with unit vector b there is a streaming algorithm that returns x with $\mathbb{E}\,\|x - x^{\text{opt}}\|_{\mathbf{B}}^2 \leq 10c\lambda_1(\mathbf{B}^{-1})$ using $O(\frac{\mathrm{v}(\mathcal{D})}{\mathrm{gap}^2\cdot c})$ samples from $\mathcal{D}$.*

Note that convergence for this streaming algorithm is significantly worse than for offline SVRG algorithms. The number of samples we take is proportional to $1/c$, so, if we wanted to for example, apply Theorem 8 we would need $\mathrm{poly}(1/\mathrm{gap}, d)$ samples (compared with just $\log(d/\mathrm{gap})$ iterations in the online case). This is why we only give a warm-start algorithm in the online case – one that operates in the regime where coarse linear solves are sufficient. Our main theorem is:

Theorem 25 (Online Shifted-and-Inverted Power Method – Warm Start)**.** *Let $\mathbf{B} = \lambda\mathbf{I} - \mathbf{A}^\top\mathbf{A}$ for $\left(1+\frac{\mathrm{gap}}{150}\right)\lambda_1 \leq \lambda \leq \left(1+\frac{\mathrm{gap}}{100}\right)\lambda_1$ and let x_0 be some vector with $G(x_0) \leq \frac{1}{\sqrt{10}}$. Running the shifted-and-inverted power method on $\mathbf{B}$ initialized with x_0, using the streaming SVRG solver of Corollary 24 to approximately apply $\mathbf{B}^{-1}$ at each step, returns x such that $x^\top\mathbf{\Sigma}x \geq (1-\epsilon)\lambda_1$ with constant probability for any target $\epsilon < \mathrm{gap}$. The algorithm uses $O(\frac{\mathrm{v}(\mathcal{D})}{\mathrm{gap}\cdot\epsilon})$ samples and amortized $O(d)$ time per sample.*

6. Parameter Estimation for Offline Eigenvector Computation

In Section 4, in order to invoke Theorems 5 and 8 we assumed knowledge of some λ with $(1+\mathrm{gap}/150)\lambda_1 \leq \lambda \leq (1+\mathrm{gap}/100)\lambda_1$. Here we mention that it is possible to efficiently estimate this parameter, incurring a modest additional runtime cost. Algorithm 1 of our full paper uses the gap-free eigenvalue estimation algorithm of (Musco & Musco, 2015), applying the shifted-and-inverted power method with the SVRG based solver of Section 4 to two vectors simultaneously to compute estimates of both λ_1 and λ_2. Using the gap between these estimates, the algorithm iteratively refines its approximation of gap. Overall:

Theorem 26. *There is an algorithm that, with probability $1 - O(1/d^{10})$ returns λ with $(1+\mathrm{gap}/150)\lambda_1 \leq \lambda \leq (1+\mathrm{gap}/100)\lambda_1$ ($\hat{\lambda}_1$) in time*

$$O\left(\left[\mathrm{nnz}(\mathbf{A}) + \frac{d\,\mathrm{sr}(\mathbf{A})}{\mathrm{gap}^2}\right]\cdot\log^3\left(\frac{d}{\mathrm{gap}}\right)\right).$$

Note that, by using the accelerated solver discussed in Section 4 we can also accelerate this to $\tilde{O}\left(\frac{\mathrm{nnz}(\mathbf{A})^{3/4}(d\,\mathrm{sr}(\mathbf{A}))^{1/4}}{\sqrt{\mathrm{gap}}}\right)$. The runtime of Theorem 26 is within a $O(\log(d/\mathrm{gap}))$ factor of our runtimes that assume knowledge of λ. Additionally, note that this extra cost is separated from the ϵ dependencies in the runtimes.

References

Ailon, Nir and Chazelle, Bernard. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

Balsubramani, Akshay, Dasgupta, Sanjoy, and Freund, Yoav. The fast convergence of incremental PCA. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pp. 3174–3182, 2013.

Bottou, Léon. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT*, pp. 177–186. Springer, 2010.

Clarkson, Kenneth L and Woodruff, David P. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 81–90, 2013.

Cohen, Michael B, Lee, Yin Tat, Musco, Cameron, Musco, Christopher, Peng, Richard, and Sidford, Aaron. Uniform sampling for matrix approximation. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, pp. 181–190, 2015.

Csiba, Dominik and Richtárik, Peter. Primal method for ERM with flexible mini-batching schemes and non-convex losses. *arXiv:1506.02227*, 2015.

Frostig, Roy, Ge, Rong, Kakade, Sham M, and Sidford, Aaron. Competing with the empirical risk minimizer in a single pass. In *Proceedings of the 28th Annual Conference on Computational Learning Theory (COLT)*, pp. 728–763, 2015a.

Frostig, Roy, Ge, Rong, Kakade, Sham M, and Sidford, Aaron. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015b.

Hardt, Moritz and Price, Eric. The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pp. 2861–2869, 2014.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pp. 315–323, 2013.

Jolliffe, Ian. *Principal component analysis*. Wiley Online Library, 2002.

Koren, Yehuda. On spectral graph drawing. In *Computing and Combinatorics*, pp. 496–508. Springer, 2003.

Le Gall, François. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pp. 296–303. ACM, 2014.

Lin, Hongzhou, Mairal, Julien, and Harchaoui, Zaid. A universal catalyst for first-order optimization. *arXiv:1506.02186*, 2015.

Mitliagkas, Ioannis, Caramanis, Constantine, and Jain, Prateek. Memory limited, streaming PCA. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pp. 2886–2894, 2013.

Musco, Cameron and Musco, Christopher. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Ng, Andrew Y, Jordan, Michael I, and Weiss, Yair. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pp. 849–856, 2002.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The PageRank citation ranking: bringing order to the Web. 1999.

Sa, Christopher D, Re, Christopher, and Olukotun, Kunle. Global convergence of stochastic gradient descent for some non-convex matrix problems. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 2332–2341, 2015.

Saad, Yousef. *Numerical methods for large eigenvalue problems*. SIAM, 1992.

Shalev-Shwartz, Shai. SDCA without duality. *arXiv:1502.06177*, 2015.

Shamir, Ohad. Convergence of stochastic gradient descent for PCA. *arXiv:1509.09002*, 2015a.

Shamir, Ohad. Fast stochastic algorithms for SVD and PCA: Convergence properties and convexity. *arXiv:1507.08788*, 2015b.

Shamir, Ohad. A stochastic PCA and SVD algorithm with an exponential convergence rate. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 144–152, 2015c.

Spielman, Daniel A. Spectral graph theory and its applications. In *null*, pp. 29–38. IEEE, 2007.

Tropp, Joel A. An introduction to matrix concentration inequalities. *arXiv:1501.01571*, 2015.

Vempala, Santosh and Wang, Grant. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

Williams, Virginia Vassilevska. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 887–898, 2012.