Fast Stochastic Algorithms for SVD and PCA: Convergence Properties and Convexity

Ohad Shamir OHAD.SHAMIR@WEIZMANN.AC.IL
Weizmann Institute of Science, Rehovot, Israel

Abstract

We study the convergence properties of the VR-PCA algorithm introduced by (Shamir, 2015) for fast computation of leading singular vectors. We prove several new results, including a formal analysis of a block version of the algorithm, and convergence from random initialization. We also make a few observations of independent interest, such as how pre-initializing with just a single exact power iteration can significantly improve the analysis, and what are the convexity and non-convexity properties of the underlying optimization problem.

1. Introduction

We consider the problem of computing the subspace spanned by the top k left singular vectors of a $d \times n$ matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $k \ll \min\{n, d\}$. This is equivalent to computing the subspace of the top k eigenvectors of $XX^\top$, or equivalently, solving the optimization problem

$$\min_{W \in \mathbb{R}^{d\times k}: W^\top W = I} -\text{Trace}\left(W^\top \left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right) W\right). \tag{1}$$

This is one of the most fundamental matrix computation problems, and has numerous uses (such as low-rank matrix approximation and principal component analysis).

For large-scale matrices X, where exact eigendecomposition is infeasible, standard deterministic approaches are based on power iterations or variants thereof (e.g. the Lanczos method) (Golub & Van Loan, 2012). Alternatively, one can exploit the structure of Eq. (1) and apply stochastic iterative algorithms, where in each iteration we update a current $d \times k$ matrix W based on one or more randomly-drawn columns $\mathbf{x}_i$ of X. Such algorithms have been known for several decades ((Krasulina, 1969; Oja, 1982)), and enjoyed renewed interest in recent years, e.g. (Arora et al., 2012; Balsubramani et al., 2013; Arora et al., 2013; Hardt & Price, 2014; De Sa et al., 2015). Another stochastic approach is based on SVD of a random projection of the data matrix, e.g. (Halko et al., 2011; Woodruff, 2014).

Unfortunately, each of these algorithms suffer from a different disadvantage: The deterministic algorithms are accurate (runtime logarithmic in the required accuracy ϵ, under an eigengap condition), but require a full pass over the matrix for each iteration, and in the worst-case many such passes would be required (polynomial in the eigengap). On the other hand, each iteration of the stochastic algorithms is cheap, and their number is independent of the size of the matrix, but on the flip side, their noisy stochastic nature means they are not suitable for obtaining a high-accuracy solution (the runtime scales polynomially with ϵ).

Recently, (Shamir, 2015) proposed a new practical algorithm, VR-PCA, for solving Eq. (1), which has a "best-of-both-worlds" property: The algorithm is based on cheap stochastic iterations, yet the algorithm's runtime is logarithmic in the required accuracy ϵ. More precisely, for the case $k = 1$, $\mathbf{x}_i$ of bounded norm, and when there is an eigengap of λ between the first and second leading eigenvalues of the covariance matrix $\frac{1}{n}XX^\top$, the required runtime was shown to be on the order of

$$d\left(n + \frac{1}{\lambda^2}\right)\log\left(\frac{1}{\epsilon}\right). \tag{2}$$

The algorithm is therefore suitable for obtaining high accuracy solutions (the dependence on ϵ is logarithmic), essentially at the cost of only $\mathcal{O}(\log(1/\epsilon))$ passes over the data, assuming $\lambda = \Omega(1/\sqrt{n})$. The algorithm is based on the variance-reduction technique introduced in (Johnson & Zhang, 2013) to speed up stochastic algorithms for *convex* optimization problems, even though the optimization problem in Eq. (1) is inherently non-convex. See Section 3 for a more detailed description of this algorithm, and (Shamir, 2015) for more discussions as well as experiments.

These results left several questions open. For example, it is not clear if the quadratic dependence on $1/\lambda$ in Eq. (2)

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

is necessary, since it is worse than the linear (or better) dependence that can be obtained with the deterministic algorithms mentioned earlier, as well as analogous results that can be obtained with similar techniques for convex optimization problems (where λ is the strong convexity parameter). Also, the analysis was only shown for the case $k = 1$, whereas often in practice, we may want to recover $k > 1$ singular vectors simultaneously. Although (Shamir, 2015) proposed a variant of the algorithm for that case, and studied it empirically, no analysis was provided. Finally, the convergence guarantee assumed that the algorithm is initialized from a point closer to the optimum than what is attained with standard random initialization. Although one can use some other, existing stochastic algorithm to do this "warm-start", no end-to-end analysis of the algorithm, starting from random initialization, was provided.

In this paper, we study these and related questions, and make the following contributions:

- We propose a variant of VR-PCA to handle the $k > 1$ case, and formally analyze its convergence (Section 3). The generalization to $k > 1$ is far from trivial, and requires carefully tracking the evolution of the subspace spanned by the current iterate.
- In Section 4, we study the convergence of VR-PCA starting from a random initialization. And show that with a slightly smarter initialization – essentially, random initialization followed by a *single* power iteration – the convergence (at least in terms of the analysis) can be substantially improved.
- In Section 5, we study whether functions similar to Eq. (1) have hidden convexity properties, which would allow applying existing convex optimization tools as-is, and possibly improve the required runtime. For the $k = 1$ case, we show that this is in a sense true: Close enough to the optimum, and on a suitably-designed convex set, such a function is indeed λ-strongly convex. Unfortunately, the distance from the optimum has to be $\mathcal{O}(\lambda)$, which is quite small and is not clear how to utilize in most practical regimes.

Following the initial publication of this paper, both (Garber & Hazan, 2015) and (Jin et al., 2015) proposed a different family of algorithms for the same problem as we do, in the special case where $k = 1$. The algorithms are based on repeatedly solving a series of convex problems, using fast stochastic methods, and have better theoretical runtime bounds in certain parameter regimes and when starting from random initialization. On the flip side, besides having an analysis specific to the $k = 1$ case, the algorithms are considerably more complex than the simple schemes we consider here, and their practical efficiency on actual datasets remains to be seen. Moreover, to get an algorithm requiring a logarithmic number of passes over the data, the eigengap λ must in general be $\Omega(1/\sqrt{n})$, which in that respect is no better than the runtime bound in Eq. (2). Thus, it is still of interest to understand whether this dependence between λ and n is inherent to this non-convex problem.

2. Some Preliminaries and Notation

We consider a $d \times n$ matrix X composed of n columns $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, and let

$$A = \frac{1}{n} X X^\top = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top .$$

Thus, Eq. (1) is equivalent to finding the k leading eigenvectors of A.

We generally use bold-face letters to denote vectors, and capital letters to denote matrices. We let $\mathrm{Tr}(\cdot)$ denote the trace of a matrix, $\|\cdot\|_F$ to denote the Frobenius norm, and $\|\cdot\|_2$ to denote the spectral norm. A symmetric $d \times d$ matrix B is positive semidefinite, if $\inf_{\mathbf{z} \in \mathbb{R}^d} \mathbf{z}^\top B \mathbf{z} \geq 0$. A is positive definite if the inequality is strict. Following standard notation, we write $B \succeq 0$ to denote that A is positive semidefinite, and $B \succeq C$ if $B - C \succeq 0$. $B \succ 0$ means that B is positive definite.

A twice-differentiable function F on a subset of $\mathbb{R}^d$ is convex, if its Hessian is alway positive semidefinite. If it is always positive definite, and $\succ \lambda I$ for some $\lambda > 0$, we say that the function is λ-strongly convex. If the Hessian is always $\prec sI$ for some $s \geq 0$, then the function is s-smooth.

3. The VR-PCA Algorithm and a Block Version

We begin by recalling the algorithm of (Shamir, 2015) for the $k = 1$ case (Algorithm 1), and then discuss its generalization for $k > 1$.

The basic idea of the algorithm is to perform stochastic updates using randomly-sampled columns $\mathbf{x}_i$ of the matrix, but interlace them with occasional exact power iterations, and use that to gradually reduce the variance of the stochastic updates. Specifically, the algorithm is split into epochs $s = 1, 2, \ldots$, where in each epoch we do a single exact power iteration with respect to the matrix A (by computing $\tilde{\mathbf{u}}$), and then perform m stochastic updates, which can be re-written as

$$\mathbf{w}'_t = (I + \eta A)\mathbf{w}_{t-1} + \eta \left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)(\mathbf{w}_{t-1} - \tilde{\mathbf{w}}_{s-1})$$
$$\mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}'_t,$$

Algorithm 1 VR-PCA: Vector version ($k = 1$)

Parameters: Step size η, epoch length m
Input: Data matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$; Initial unit vector $\tilde{\mathbf{w}}_0$
for $s = 1, 2, \ldots$ **do**
  $\tilde{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{x}_i^\top \tilde{\mathbf{w}}_{s-1}\right)$
  $\mathbf{w}_0 = \tilde{\mathbf{w}}_{s-1}$
  for $t = 1, 2, \ldots, m$ **do**
    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $\mathbf{w}'_t = \mathbf{w}_{t-1} + \eta\left(\mathbf{x}_{i_t}\left(\mathbf{x}_{i_t}^\top \mathbf{w}_{t-1} - \mathbf{x}_{i_t}^\top \tilde{\mathbf{w}}_{s-1}\right) + \tilde{\mathbf{u}}\right)$
    $\mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}'_t$
  end for
  $\tilde{\mathbf{w}}_s = \mathbf{w}_m$
end for

The first term is essentially a power iteration (with a finite step size η), whereas the second term is zero-mean, and with variance dominated by $\|\mathbf{w}_{t-1} - \tilde{\mathbf{w}}_{s-1}\|^2$. As the algorithm progresses, $\mathbf{w}_{t-1}$ and $\tilde{\mathbf{w}}_{s-1}$ both converge toward the same optimal point, hence $\|\mathbf{w}_{t-1} - \tilde{\mathbf{w}}_{s-1}\|^2$ shrinks, eventually leading to an exponential convergence rate.

To handle the $k > 1$ case (where more than one eigenvector should be recovered), one simple technique is deflation, where we recover the leading eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ one-by-one, each time using the $k = 1$ algorithm. However, a disadvantage of this approach is that it requires a positive eigengap between all top k eigenvalues, otherwise the algorithm is not guaranteed to converge. Thus, an algorithm which simultaneously recovers all k leading eigenvectors is preferable.

We will study a block version of Algorithm 1, presented as Algorithm 2. Roughly speaking, we replace the d-dimensional vectors $\mathbf{w}_{t-1}, \tilde{\mathbf{w}}_{s-1}, \mathbf{u}$ by $d \times k$ matrices $W_{t-1}, \tilde{W}_{s-1}, \tilde{U}$, and normalization is replaced by orthogonalization[1]. Indeed, Algorithm 1 is equivalent to Algorithm 2 when $k = 1$. The main twist in Algorithm 2 is that instead of using $\tilde{W}_{s-1}, \tilde{U}$ as-is, we perform a unitary transformation (via the $k \times k$ orthogonal matrix B_{t-1}) which maximally aligns them with W_{t-1}. Note that B_{t-1} is a $k \times k$ matrix, and since k is assumed to be small, this does not introduce significant computational overhead.

We now turn to provide a formal analysis of Algorithm 2:

Theorem 1. *Define the $d \times d$ matrix A as $\frac{1}{n}XX^\top = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top$, and let V_k denote the $d \times k$ matrix composed of the eigenvectors corresponding to the largest k eigenvalues. Suppose that*

[1]The normalization $W_t = W'_t\left(W_t'^\top W'_t\right)^{-1/2}$ ensures that W_t has orthonormal columns. We note that in our analysis, η is chosen sufficiently small so that $W_t'^\top W'_t$ is always invertible, hence the operation is well-defined.

Algorithm 2 VR-PCA: Block version

Parameters: Rank k, Step size η, epoch length m
Input: Data matrix $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$; Initial $d \times k$ matrix $\tilde{W}_0$ with orthonormal columns
for $s = 1, 2, \ldots$ **do**
  $\tilde{U} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{x}_i^\top \tilde{W}_{s-1}\right)$
  $W_0 = \tilde{W}_{s-1}$
  for $t = 1, 2, \ldots, m$ **do**
    $B_{t-1} = VU^\top$, where $USV^\top$ is an SVD decomposition of $W_{t-1}^\top \tilde{W}_{s-1}$
    ▷ Equivalent to

$$B_{t-1} = \arg\min_{B^\top B = I} \|W_{t-1} - \tilde{W}_{s-1}B\|_F^2$$

    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $W'_t = W_{t-1} +$
      $\eta\left(\mathbf{x}_{i_t}\left(\mathbf{x}_{i_t}^\top W_{t-1} - \mathbf{x}_{i_t}^\top \tilde{W}_{s-1}B_{t-1}\right) + \tilde{U}B_{t-1}\right)$
    $W_t = W'_t\left(W_t'^\top W'_t\right)^{-1/2}$
  end for
  $\tilde{W}_s = W_m$
end for

- $\max_i \|\mathbf{x}_i\|^2 \le r$ *for some* $r > 0$.
- *A has eigenvalues $s_1 > s_2 \ge \ldots \ge s_d$, where $s_k - s_{k+1} = \lambda$ for some $\lambda > 0$.*
- $k - \|V_k^\top \tilde{W}_0\|_F^2 \le \frac{1}{2}$.

Let $\delta, \epsilon \in (0, 1)$ be fixed. If we run the algorithm with any epoch length parameter m and step size η, such that

$$\eta \le \frac{c\delta^2}{r^2}\lambda \quad , \quad m \ge \frac{c' \log(2/\delta)}{\eta\lambda},$$
$$km\eta^2 r^2 + rk\sqrt{m\eta^2 \log(2/\delta)} \le c'' \tag{3}$$

(where c, c', c'' designate certain positive numerical constants), and for $T = \left\lceil \frac{\log(1/\epsilon)}{\log(2/\delta)} \right\rceil$ epochs, then with probability at least $1 - \lceil \log_2(1/\epsilon) \rceil \delta$, it holds that

$$k - \|V_k^\top \tilde{W}_T\|_F^2 \ \le \ \epsilon.$$

For any orthogonal W, $k - \|V_k^\top W\|_F^2$ lies between 0 and k, and equals 0 when the column spaces of V_k and W are the same (i.e., when W spans the k leading singular vectors). According to the theorem, taking appropriate[2]

[2]Specifically, we can take $m = c' \log(2/\delta)/\eta\lambda$ and $\eta = a\delta^2/r^2\lambda$, where a is sufficiently small to ensure that the first and third condition in Eq. (3) holds. It can be verified that it's enough to take $a = \min\left\{c, \frac{c''}{4\delta^2 ck\log(2/\delta)}, \frac{1}{4\delta^2 c}\left(\frac{c''}{k\log(2/\delta)}\right)^2\right\}$.

$\eta = \Theta(\lambda/(kr)^2)$, and $m = \Theta((rk/\lambda)^2)$, the algorithm converges with high probability to a high-accuracy approximation of V_k. Moreover, the runtime of each epoch of the algorithm equals $\mathcal{O}(mdk^2 + dnk)$. Overall, we get the following corollary:

Corollary 1. *Under the conditions of Theorem 1, there exists an algorithm returning $\tilde{W}_T$ such that $k - \|V_k^\top \tilde{W}_T\|_F^2 \leq \epsilon$ with arbitrary constant accuracy, in runtime*

$$\mathcal{O}\left(dk\left(n + \frac{r^2k^3}{\lambda^2}\right)\log(1/\epsilon)\right).$$

This runtime bound coincides with the existing one for the $k = 1$ case [3].

The proof of Theorem 1 appears in Subsection A.1, and relies on a careful tracking of the evolution of the potential function $k - \|V_k^\top \tilde{W}_t\|_F^2$. An important challenge compared to the $k = 1$ case is that the matrices W_{t-1} and $\tilde{W}_{s-1}$ do not necessarily become closer over time, so the variance-reduction intuition discussed earlier no longer applies. However, the column space of W_{t-1} and $\tilde{W}_{s-1}$ do become closer, and this is utilized by introducing the transformation matrix B_{t-1}. We note that although B_{t-1} appears essential for our analysis, it isn't clear that using it is necessary in practice: In (Shamir, 2015), the suggested block algorithm was Algorithm 2 with $B_{t-1} = I$, which seemed to work well in experiments. In any case, using this matrix doesn't affect the overall runtime beyond constants, since the additional runtime of computing and using this matrix ($\mathcal{O}(dk^2)$) is the same as the other computations performed at each iteration.

A limitation of the theorem above is the assumption that the initial point $\tilde{W}_0$ is such that $k - \|V_k^\top \tilde{W}_0\|_F^2 \leq \frac{1}{2}$. This is a non-trivial assumption, since if we initialize the algorithm from a random $d \times \mathcal{O}(1)$ orthogonal matrix $\tilde{W}_0$, then with overwhelming probability, $\|V_k^\top \tilde{W}_0\|_F^2 = \mathcal{O}(1/d)$. However, experimentally the algorithm seems to work well even with random initialization (Shamir, 2015). Moreover, if we are interested in a theoretical guarantee, one simple solution is to warm-start the algorithm with a purely stochastic algorithm for this problem (such as (De Sa et al., 2015; Hardt & Price, 2014; Balsubramani et al., 2013)), with runtime guarantees on getting such a $\tilde{W}_0$. The idea is that $\tilde{W}_0$ is only required to approximate V_k up to constant accuracy, so purely stochastic algorithms (which are good in obtaining a low-accuracy solution) are quite suitable. In the next section, we further delve into these issues, and show that in our setting such algorithms in fact can be substantially improved.

[3] (Shamir, 2015) showed that it's possible to further improve the runtime for sparse X, replacing d by the average column sparsity d_s. This is done by maintaining parameters in an implicit form, but it's not clear how to implement a similar trick in the block version, where $k > 1$.

4. Warm-Start and the Power of a Power Iteration

In this section, we study the runtime required to compute a starting point satisfying the conditions of Theorem 1, starting from a random initialization. Combined with Theorem 1, this gives us an end-to-end analysis of the runtime required to find an ϵ-accurate solution, starting from a random point. For simplicity, we will only discuss the case $k = 1$, i.e. where our goal is to compute the single leading eigenvector $\mathbf{v}_1$, although our observations can be generalized to $k > 1$. In the $k = 1$ case, Theorem 1 kicks in once we find a vector $\mathbf{w}$ satisfying $\langle \mathbf{v}_1, \mathbf{w}\rangle^2 \geq \frac{1}{2}$. However, when we have no prior knowledge on the optimal solution $\mathbf{v}_1$, the standard initialization heuristic is to choose a starting vector uniformly at random from the unit sphere, in which case we have $\langle \mathbf{v}_1, \mathbf{w}_0\rangle^2 \approx \frac{1}{d}$. Thus, we begin with a vector almost orthogonal to the leading eigenvector $\mathbf{v}_1$ (depending on d), and this leads to polynomial dependencies on d in the analysis.

However, it turns out that it is possible to improve the analysis by a smarter initialization: Sample $\mathbf{w}$ from the standard Gaussian distribution on $\mathbb{R}^d$, perform a *single* power iteration w.r.t. the covariance matrix $A = \frac{1}{n}XX^\top$, i.e. $\mathbf{w}_0 = A\mathbf{w}/\|A\mathbf{w}\|$, and initialize from $\mathbf{w}_0$. For such a procedure, we have the following simple observation:

Lemma 1. *For $\mathbf{w}_0$ as above, it holds for any δ that with probability at least $1 - \frac{1}{d} - \delta$,*

$$\langle \mathbf{v}_1, \mathbf{w}_0\rangle^2 \geq \frac{\delta^2}{12\log(d)\ \mathit{nrank}(A)},$$

where $\mathit{nrank}(A) = \frac{\|A\|_F^2}{\|A\|_2^2}$ is the numerical rank of A.

The numerical rank (see e.g. (Rudelson & Vershynin, 2007)) is a relaxation of the standard notion of rank: For any $d \times d$ matrix A, nrank(A) is at most the rank of A (which in turn is at most d). However, it will be small even if A is just close to being low-rank. In many if not most machine learning applications, we are interested in matrices which tend to be approximately low-rank, in which case nrank(A) is much smaller than d or even a constant. Therefore, by a single power iteration, we get an initial point $\mathbf{w}_0$ for which $\langle \mathbf{v}_1, \mathbf{w}_0\rangle^2$ is on the order of $1/\text{nrank(A)}$, which can be much larger than the $1/d$ given by a random initialization, and is never substantially worse.

Proof. Let $s_1 \geq s_2 \geq \ldots \geq s_d \geq 0$ be the d eigenvalues

of A, with eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$. We have

$$\begin{aligned}\langle \mathbf{v}_1, \mathbf{w}_0 \rangle^2 &= \frac{\langle \mathbf{v}_1, A\mathbf{w} \rangle^2}{\|A\mathbf{w}\|^2} \\ &= \frac{\left(s_1 \langle \mathbf{v}_1, \mathbf{w} \rangle\right)^2}{\left(\sum_{i=1}^d s_i \mathbf{v}_i \langle \mathbf{v}_i, \mathbf{w} \rangle\right)^2} \\ &= \frac{s_1^2 \langle \mathbf{v}_1, \mathbf{w} \rangle^2}{\sum_{i=1}^d s_i^2 \langle \mathbf{v}_i, \mathbf{w} \rangle^2}.\end{aligned}$$

Since $\mathbf{w}$ is distributed according to a standard Gaussian distribution, which is rotationally symmetric, we can assume without loss of generality that $\mathbf{v}_1, \ldots, \mathbf{v}_d$ correspond to the standard basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_d$, in which case the above reduces to

$$\frac{s_1^2 w_1^2}{\sum_{i=1}^d s_i^2 w_i^2} \geq \frac{s_1^2}{\sum_{i=1}^d s_i^2} \cdot \frac{w_1^2}{\max_i w_i^2},$$

where $w_1, \ldots, w_d$ are independent and scalar random variables with a standard Gaussian distribution.

First, we note that s_1^2 equals $\|A\|_2^2$, the spectral norm of A, whereas $\sum_{i=1}^d s_i^2$ equals $\|A\|_F^2$, the Frobenius norm of A. Therefore, $\frac{s_1^2}{\sum_i s_i^2} = \frac{\|A\|_2^2}{\|A\|_F^2} = \frac{1}{\text{nrank}(A)}$, and we get overall that

$$\langle \mathbf{v}_1, \mathbf{w}_0 \rangle^2 \geq \frac{1}{\text{nrank}(A)} \cdot \frac{w_1^2}{\max_i w_i^2}. \tag{4}$$

We consider the random quantity $w_1^2 / \max_i w_i^2$, and independently bound the deviation probability of the numerator and denominator. First, for any $t \geq 0$ we have

$$\begin{aligned}\Pr(w_1^2 \leq t) &= \Pr(w_1 \in [-\sqrt{t}, \sqrt{t}]) \\ &= \int_{z=-\sqrt{t}}^{\sqrt{t}} \sqrt{\frac{1}{2\pi}} \exp\left(-\frac{z^2}{2}\right) \\ &\leq \sqrt{\frac{1}{2\pi}} * 2\sqrt{t} \\ &= \sqrt{\frac{2}{\pi}} t\,. \end{aligned} \tag{5}$$

Second, by combining two standard Gaussian concentration results (namely, that if $W = \max\{|w_1|, \ldots, |w_d|\}$, then $0 \leq \mathbb{E}[W] \leq 2\sqrt{2\log(d)}$, and by the Cirelson-Ibragimov-Sudakov inequality, $\Pr(W - \mathbb{E}[W] > t) \leq \exp(-t^2/2)$), we get that

$$\Pr(\max_i |w_i| > 2\sqrt{2\log(d)} + t) \leq \exp(-t^2/2),$$

and therefore

$$\Pr(\max_i w_i^2 > (2\sqrt{2\log(d)} + t)^2) \leq \exp(-t^2/2). \tag{6}$$

Combining Eq. (5) and Eq. (6), with a union bound, we get that for any $t_1, t_2 \geq 0$, it holds with probability at least $1 - \sqrt{\frac{2}{\pi}} t_1 - \exp(-t_2^2/2)$ that

$$\frac{w_1^2}{\max_i w_i^2} \geq \frac{t_1}{(2\sqrt{2\log(d)} + t_2)^2}.$$

To slightly simplify this for readability, we take $t_2 = \sqrt{2\log(d)}$, and substitute $\delta = \sqrt{\frac{2}{\pi}} t_1$. This implies that with probability at least $1 - \delta - 1/d$,

$$\frac{w_1^2}{\max_i w_i^2} \geq \frac{\frac{\pi}{2}\delta^2}{18\log(d)} > \frac{\delta^2}{12\log(d)}.$$

Plugging back into Eq. (4), the result follows. □

When an exact power iteration is possible, this result can be plugged into the existing analyses of streaming PCA/SVD algorithms, and can often improve the dependence on the d factor in the iteration complexity bounds to a dependence on the numerical rank of A. Moreover, even in a purely streaming setting, where an exact power iteration is not possible, an approximate power iteration can still be performed and apparently lead to similar results.

To give a concrete example of this, we provide a convergence analysis of the VR-PCA algorithm (Algorithm 1), starting from an arbitrary initial point, bounding the total number of stochastic iterations required by the algorithm in order to produce a point satisfying the conditions of Theorem 1 (from which point the analysis of Theorem 1 takes over). Combined with Theorem 1, this analysis also justifies that VR-PCA indeed converges starting from a random initialization.

Theorem 2. *Using the notation of Theorem 1 (where λ is the eigengap, $\mathbf{v}_1$ is the leading eigenvector, and $r = \max_i \|\mathbf{x}_i\|^2$), and for any $\delta \in (0, \frac{1}{2})$, suppose we run Algorithm 1 with some initial unit-norm vector $\tilde{\mathbf{w}}_0$ such that*

$$\langle \mathbf{v}_1, \tilde{\mathbf{w}}_0 \rangle^2 \geq \zeta > 0,$$

and a step size η satisfying

$$\eta \leq \frac{c\delta^2 \lambda \zeta^3}{r^2 \log^2(2/\delta)} \tag{7}$$

(for some universal constant c). Then with probability at least $1 - \delta$, after

$$T = \left\lfloor \frac{c' \log(2/\delta)}{\eta \lambda \zeta} \right\rfloor$$

stochastic iterations (lines $6-10$ in the pseudocode, where c' is again a universal constant), we get a point $\mathbf{w}_T$ satisfying $1 - \langle \mathbf{v}_1, \mathbf{w}_T \rangle^2 \leq \frac{1}{2}$. Moreover, if η is chosen on the same order as the upper bound in Eq. (7), then

$$T = \Theta\left(\frac{r^2 \log^3(2/\delta)}{\delta^2 \lambda^2 \zeta^4}\right).$$

Note that the analysis does not depend on the choice of the epoch size m, and does not use the special structure of VR-PCA (in fact, the technique we use is applicable to any algorithm which takes stochastic gradient steps to solve this type of problem). The proof of the theorem appears in Section A.2.

we get that the runtime required by VR-PCA to find a point $\mathbf{w}$ such that $1 - \langle \mathbf{v}_1, \mathbf{w}_T \rangle^2 \leq \frac{1}{2}$ is $\mathcal{O}(d/\lambda^2 \zeta^4)$ where ζ is a lower bound on $\langle \mathbf{v}_1, \tilde{\mathbf{w}}_0 \rangle^2$. As discussed earlier, if $\tilde{\mathbf{w}}_0$ is a result of random initialization followed by a power iteration (requiring $\mathcal{O}(nd)$ time), and the covariance matrix A has small numerical rank, then $\zeta = \langle \mathbf{v}_1, \tilde{\mathbf{w}}_0 \rangle^2 = \tilde{\Omega}(1/\log(d))$, and the runtime is

$$\mathcal{O}\left(nd + \frac{d}{\lambda^2}\log^4(d)\right) = \mathcal{O}\left(d\left(n + \left(\frac{\log^2(d)}{\lambda}\right)^2\right)\right).$$

By Corollary 1, the runtime required by VR-PCA from that point to get an ϵ-accurate solution is

$$\mathcal{O}\left(d\left(n + \frac{1}{\lambda^2}\right)\log\left(\frac{1}{\epsilon}\right)\right),$$

so the sum of the two expressions (which is $d\left(n + \frac{1}{\lambda^2}\right)$ up to log-factors), represents the total runtime required by the algorithm.

Very recently (and following the initial publication of this paper), (Jain et al., 2016) managed to obtain an improved analysis of *standard* stochastic gradient for our problem, which requires $\tilde{\mathcal{O}}(1/\lambda^2\epsilon)$ iterations (ignoring log factors) to achieve a point $\mathbf{w}$ such that $\langle \mathbf{v}_1, \mathbf{w} \rangle^2 \geq 1 - \epsilon$, with constant probability. Using this new analysis, if we warm-start VR-PCA with $\tilde{\mathcal{O}}(1/\lambda^2)$ iterations of stochastic gradient descent (each requiring $\mathcal{O}(d)$ runtime), we again get an algorithm attaining a high-accuracy solution in $\tilde{\mathcal{O}}(d\left(n + \frac{1}{\lambda^2}\right))$ runtime, starting from random initialization. Note that here, we do not need to assume the covariance matrix A has low numerical rank, and the runtime of the warm-start phase scales as d/λ^2 rather than dn. Moreover, we suspect that the analysis of (Jain et al., 2016) can be extended to VR-PCA (which is also a stochastic gradient method), although the structure of the updates makes the analysis of (Jain et al., 2016) inapplicable as-is. If that is the case, a standard random initialization of VR-PCA will get the same runtime, without the need for any special initialization or warm-start procedures. We note that this would accord with the empirical evidence in (Shamir, 2015), in which VR-PCA appeared to work well from standard random initialization.

5. Convexity and Non-Convexity of the Rayleigh Quotient

As mentioned in the introduction, an intriguing open question is whether the requirement of $\lambda = \Omega(1/\sqrt{n})$ is necessary or can be further improved, if we are interested in an algorithm performing a logarithmic number of passes over the data. Indeed, in the world of convex optimization from which our algorithmic techniques are derived, where we attempt to minimize convex functions of the form $F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{w})$, the analogue of λ is the strong convexity parameter of F, and it is only required that $\lambda = \Omega(1/n)$ to get a similar type of performance (see e.g. (Johnson & Zhang, 2013; Shalev-Shwartz & Zhang, 2014; Roux et al., 2012; Konečnỳ & Richtárik, 2013; Frostig et al.) in the context of the variance-reduction technique we use). To phrase this in a different way, we showed runtime bounds of order $d(n+1/\lambda^2)\log(1/\epsilon)$ runtime bounds for the PCA/SVD problem, but for λ-strongly convex problems, the known runtime bounds are of order $d(n + 1/\lambda)\log(1/\epsilon)$. Is it possible to get such runtimes (with linear rather than quadratic dependence on $1/\lambda$) for our problem as well?

Another question is whether the non-convex problem that we are tackling (Eq. (1)) is really that non-convex. Clearly, it has a nice structure (since we can solve the problem in polynomial time), but perhaps it actually has hidden convexity properties, at least close enough to the optimal points? We note that Eq. (1) can be "trivially" convexified, by re-casting it as an equivalent semidefinite program (Boyd & Vandenberghe, 2004). However, that would require optimization over $d \times d$ matrices, leading to poor runtime and memory requirements. The question here is whether we have any convexity with respect to the *original* optimization problem over "thin" $d \times k$ matrices.

In fact, the two questions of improved runtime and convexity are closely related: If we can show that the optimization problem is convex in some domain containing an optimal point, perhaps this can lead to faster algorithms for this problem, by utilizing such convexity.

To discuss these questions, we will focus on the $k = 1$ case for simplicity (i.e., our goal is to find a leading eigenvector of the matrix $A = \frac{1}{n}XX^\top = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$), and study potential convexity properties of the negative Rayleigh quotient,

$$F_A(\mathbf{w}) = -\frac{\mathbf{w}^\top A\mathbf{w}}{\|\mathbf{w}\|^2} = \frac{1}{n}\sum_{i=1}^n\left(-\frac{\langle \mathbf{w}, \mathbf{x}_i\rangle^2}{\|\mathbf{w}\|^2}\right).$$

Note that for $k = 1$, this function coincides with Eq. (1) on the unit Euclidean sphere, and with the same optimal points, but has the nice property of being defined on the entire Euclidean space (thus, at least its domain is convex).

At a first glance, such functions F_A appear to potentially be convex at some bounded distance from an optimum, as illustrated for instance in the case where $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ (see Figure 1). Unfortunately, it turns out that the figure is

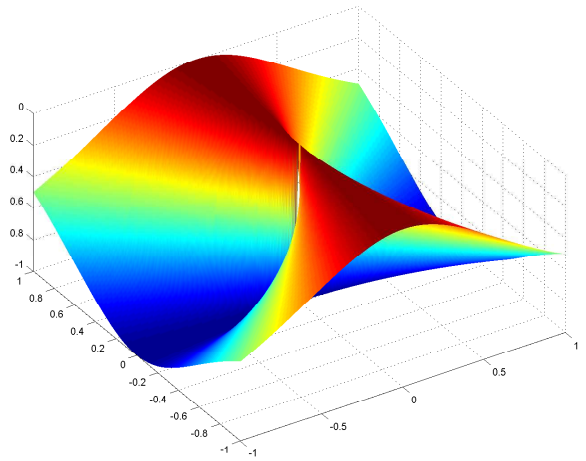

Figure 1. The function $(w_1, w_2) \mapsto -\frac{w_1^2}{w_1^2+w_2^2}$, corresponding to $F_A(\mathbf{w})$ where $A = (1\,0\,;0\,0)$ (best seen in color). It is invariant to re-scaling of $\mathbf{w}$, and attains a minimum at $(a, 0)$ for any $a \neq 0$.

misleading, and in fact the function is *not* convex almost everywhere:

Theorem 3. *For the matrix A above, the Hessian of F_A is not positive semidefinite for all but a measure-zero set.*

Proof. The leading eigenvector of A is $\mathbf{v}_1 = (1, 0)$, and $F_A(\mathbf{w}) = -\frac{w_1^2}{w_1^2+w_2^2}$. The Hessian of this function at some $\mathbf{w}$ equals

$$\frac{2}{(w_1^2+w_2^2)^3}\begin{pmatrix} w_2^2(3w_1^2-w_2^2) & -2w_1w_2(w_1^2-w_2^2) \\ -2w_1w_2(w_1^2-w_2^2) & w_1^2(w_1^2-3w_2^2) \end{pmatrix}.$$

The determinant of this 2×2 matrix equals

$$\frac{4\left(w_1^2w_2^2(3w_1^2-w_2^2)(w_1^2-3w_2^2) - 4w_1^2w_2^2(w_1^2-w_2^2)^2\right)}{(w_1^2+w_2^2)^6}$$
$$= \frac{4w_1^2w_2^2}{(w_1^2+w_2^2)^6}\left((3w_1^2-w_2^2)(w_1^2-3w_2^2) - 4(w_1^2-w_2^2)^2\right)$$
$$= \frac{4w_1^2w_2^2}{(w_1^2+w_2^2)^6}\left(-(w_1^2+w_2^2)^2\right) = -\frac{4w_1^2w_2^2}{(w_1^2+w_2^2)^4},$$

which is always non-positive, and strictly negative for $\mathbf{w}$ for which $w_1w_2 \neq 0$ (which holds for all but a measure-zero set of $\mathbb{R}^d$). Since the determinant of a positive semidefinite matrix is always non-negative, this implies that the Hessian isn't positive semidefinite for any such $\mathbf{w}$. □

The theorem implies that we indeed cannot use convex optimization tools as-is on the function F_A, even if we're close to an optimum. However, the non-convexity was shown for F_A as a function over the entire Euclidean space, so the result does not preclude the possibility of having convexity on a more constrained, lower-dimensional set. In fact, this is what we are going to do next: We will show that if we are given some point $\mathbf{w}_0$ close enough to an optimum, then we can explicitly construct a simple convex set, such that

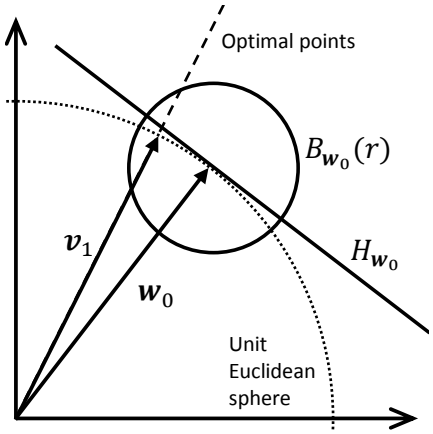

Figure 2. Illustration of the construction of the convex set on which F_A is strongly convex and smooth. $\mathbf{v}_1$ is the leading eigenvector of A, and a minimum of F_A (as well as any re-scaling of $\mathbf{v}_1$). $\mathbf{w}_0$ is a nearby unit vector, and we consider the intersection of a hyperplane orthogonal to $\mathbf{w}_0$, and an Euclidean ball centered at $\mathbf{w}_0$.

- The set includes an optimal point of F_A.
- The function F_A is $\mathcal{O}(1)$-smooth and λ-strongly convex in that set.

Unfortunately, this has a catch: To make it work, we need to have $\mathbf{w}_0$ *very* close to the optimum – in fact, we require $\|\mathbf{v}_1 - \mathbf{w}_0\| \leq \mathcal{O}(\lambda)$, and we show (in Theorem 5) that such a dependence on the eigengap λ cannot be avoided (perhaps up to a small polynomial factor). The issue is that the runtime to get such a $\mathbf{w}_0$, using stochastic-based approaches we are aware of, would scale at least quadratically with $1/\lambda$, but getting dependence better than quadratic was our problem to begin with. So, at the moment we do not know how to utilize such convexity to get actual better runtime performance. However, these results may still indicate that a better runtime and dependence on λ is possible

To explain our construction, we need to consider two convex sets: Given a unit vector $\mathbf{w}_0$, define the hyperplane tangent to $\mathbf{w}_0$,

$$H_{\mathbf{w}_0} = \{\mathbf{w} : \langle \mathbf{w}, \mathbf{w}_0\rangle = 1\}$$

as well as a Euclidean ball of radius r centered at $\mathbf{w}_0$:

$$B_{\mathbf{w}_0}(r) = \{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq r\}$$

The convex set we use, given such a $\mathbf{w}_0$, is simply the intersection of the two, $H_{\mathbf{w}_0} \cap B_{\mathbf{w}_0}(r)$, where r is a sufficiently small number (see Figure 2).

The following theorem shows that if $\mathbf{w}_0$ is $\mathcal{O}(\lambda)$-close to an optimal point (a leading eigenvector $\mathbf{v}_1$ of A), and we choose the radius of $B_{w_0}(r)$ appropriately, then $H_{\mathbf{w}_0} \cap B_{\mathbf{w}_0}(r)$ contains an optimal point, and the function F_A is indeed λ-strongly convex and smooth on that set. For simplicity, we will assume that A is scaled to have spectral norm of 1, but the result can be easily generalized.

Theorem 4. *For any positive semidefinite A with spectral norm 1, eigengap λ and a leading eigenvector $\mathbf{v}_1$, and any unit vector $\mathbf{w}_0$ such that $\|\mathbf{w}_0 - \mathbf{v}_1\| \leq \frac{\lambda}{44}$, the function $F_A(\mathbf{w})$ is 20-smooth and λ-strongly convex on the convex set $H_{\mathbf{w}_0} \cap B_{\mathbf{w}_0}\left(\frac{\lambda}{22}\right)$, which contains a global optimum of F_A.*

The proof of the theorem appears in Subsection A.3. Finally, we show below that a polynomial dependence on the eigengap λ is unavoidable, in the sense that the convexity property is lost if $\mathbf{w}_0$ is significantly further away from $\mathbf{v}_1$.

Theorem 5. *For any $\lambda, \epsilon \in \left(0, \frac{1}{2}\right)$, there exists a positive semidefinite matrix A with spectral norm 1, eigengap λ, and leading eigenvector $\mathbf{v}_1$, as well as a unit vector $\mathbf{w}_0$ for which $\|\mathbf{v}_1 - \mathbf{w}_0\| \leq \sqrt{2(1+\epsilon)\lambda}$, such that F_A is not convex in any neighborhood of $\mathbf{w}_0$ on $H_{\mathbf{w}_0}$.*

Proof. Let

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1-\lambda & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

for which $\mathbf{v}_1 = (1, 0, 0)$, and take

$$\mathbf{w}_0 = (\sqrt{1-p^2}, 0, p),$$

where $p = \sqrt{(1+\epsilon)\lambda}$ (which ensures $\|\mathbf{v}_1 - \mathbf{w}_0\|^2 = \sqrt{2p^2} = \sqrt{2(1+\epsilon)\lambda}$). Consider the ray $\{(\sqrt{1-p^2}, t, p) : t \geq 0\}$, and note that it starts from $\mathbf{w}_0$ and lies in $H_{\mathbf{w}_0}$. The function F_A along that ray (considering it as a function of t) is of the form

$$-\frac{(1-p^2)+(1-\lambda)t^2}{(1-p^2)+t^2+p^2} = -\frac{1-p^2+(1-\lambda)t^2}{1+t^2}.$$

The second derivative with respect to t equals

$$-2\frac{(3t^2-1)(\lambda-p^2)}{(t^2+1)^3} = 2\frac{(3t^2-1)\epsilon\lambda}{(t^2+1)^3},$$

where we plugged in the definition of p. This is a negative quantity for any $t < \frac{1}{\sqrt{3}}$. Therefore, the function F_A is strictly concave (and not convex) along the ray we have defined and close enough to $\mathbf{w}_0$, and therefore isn't convex in any neighborhood of $\mathbf{w}_0$ on $H_{\mathbf{w}_0}$. □

6. Discussion and Open Questions

In this paper, we studied the direct applicability of variance-reduced gradient descent methods to the non-convex SVD and PCA problem. We showed that even in the block case, where we attempt to recover the subspace of $k > 1$ eigenvectors simultaneously, such methods indeed enjoys a convergence rate comparable to the $k = 1$ case. Moreover, we studied the convergence of these methods starting from a random initialization, and observed that preceding the algorithm with a single power iteration can significantly improve the resulting bounds. Finally, we studied the geometry of the optimization problem, and showed that on a suitably chosen convex neighborhood of a global optimum, the function is indeed strongly-convex. Unfortunately, the radius around the optimum where this happens scales with the eigengap, which precludes a better runtime with currently available algorithms. Nevertheless, it does indicate that a better runtime and dependence on λ may be possible.

This work leaves several questions open. In particular, it is still not clear what is the optimal runtime for the PCA/SVD problem (at least in terms of the objective function in Eq. (1)), when using first-order, $\mathcal{O}(d)$-memory methods. In particular, we do not know whether the quadratic dependence on $1/\lambda$ is inevitable, or whether a linear dependence might be possible (similar to existing runtime bounds for λ-strongly convex problems with a finite-sum structure). This is true even for the $k = 1$ case. Another question for future research is understanding whether similar methods can be applied to other non-convex optimization problems, with provable convergence guarantees. A more specific open question is understanding the exact radius around the optimum where strong convexity occurs, by closing the gap between the radius lower bound in Theorem 4 and the upper bound in Theorem 5.

Acknowledgements: This research is supported in part by an FP7 Marie Curie CIG grant, the Intel ICRI-CI Institute, and Israel Science Foundation grant 425/13.

References

Agarwal, A. and Bottou, L. A lower bound for the optimization of finite sums. In *ICML*, 2015.

Arora, R., Cotter, A., Livescu, K., and Srebro, N. Stochastic optimization for PCA and PLS. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

Arora, R., Cotter, A., and Srebro, N. Stochastic optimization of PCA with capped MSG. In *NIPS*, 2013.

Balsubramani, A., Dasgupta, S., and Freund, Y. The fast convergence of incremental PCA. In *NIPS*, 2013.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

De Sa, C., Olukotun, K., and Ré, C. Global convergence of stochastic gradient descent for some nonconvex matrix problems. In *ICML*, 2015.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. Unregularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization.

Garber, D. and Hazan, E. Fast and simple pca via convex optimization. *arXiv preprint arXiv:1509.05647*, 2015.

Golub, G. H and Van Loan, C. *Matrix computations*, volume 3. John Hopkins University Press, 2012.

Halko, N., Martinsson, P., and Tropp, J. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Hardt, M. and Price, E. The noisy power method: A meta algorithm with applications. In *NIPS*, 2014.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Horn, R. and Johnson, C. *Matrix analysis*. Cambridge university press, 2012.

Jain, Prateek, Jin, Chi, Kakade, Sham M, Netrapalli, Praneeth, and Sidford, Aaron. Matching matrix bernstein with little memory: Near-optimal finite sample guarantees for oja's algorithm. *arXiv preprint arXiv:1602.06929*, 2016.

Jin, C., Kakade, S., Musco, C., Netrapalli, P., and Sidford, A. Robust shift-and-invert preconditioning: Faster and more sample efficient algorithms for eigenvector computation. *arXiv preprint arXiv:1510.08896*, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

Konečnỳ, Jakub and Richtárik, Peter. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

Krasulina, T.P. The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix. *USSR Computational Mathematics and Mathematical Physics*, 9(6):189–195, 1969.

Oja, E. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Roux, Nicolas L, Schmidt, Mark, and Bach, Francis R. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pp. 2663–2671, 2012.

Rudelson, M. and Vershynin, R. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21, 2007.

Shalev-Shwartz, Shai and Zhang, Tong. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pp. 1–41, 2014.

Shamir, O. A stochastic PCA and SVD algorithm with an exponential convergence rate. In *ICML*, 2015.

Woodruff, D. Sketching as a tool for numerical linear algebra. *Theoretical Computer Science*, 10(1-2):1–157, 2014.