A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate

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

Abstract

We describe and analyze a simple algorithm for principal component analysis and singular value decomposition, VR-PCA, which uses computationally cheap stochastic iterations, yet converges exponentially fast to the optimal solution. In contrast, existing algorithms suffer either from slow convergence, or computationally intensive iterations whose runtime scales with the data size. The algorithm builds on a recent variance-reduced stochastic gradient technique, which was previously analyzed for strongly convex optimization, whereas here we apply it to an inherently non-convex problem, using a very different analysis.

1. Introduction

We consider the following fundamental matrix optimization problem: Given a matrix $X \in \mathbb{R}^{d\times n}$, we wish to recover its top k left singular vectors (where $k \ll d$) by solving

$$\max_{W\in\mathbb{R}^{d\times k}:W^\top W=I} \frac{1}{n}\|X^\top W\|_F^2, \tag{1}$$

$\|\cdot\|_F$ being the Frobenius norm and I being the identity matrix[1]. A prominent application in machine learning and statistics is Principal Component Analysis (PCA), which is one of the most common tools for unsupervised data analysis and preprocessing: Given a data matrix X whose columns consist of n instances in $\mathbb{R}^d$, we are interested in finding a k-dimensional subspace (specified by a $d \times k$ matrix W), on which the projection of the data has largest possible variance. Finding this subspace has numerous uses, from dimensionality reduction and data compression to data visualization, and the problem is extremely well-studied.

[1]The top k right singular values can also be extracted, by considering the matrix $X^\top$ in lieu of X.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Letting $\mathbf{x}_1,\ldots,\mathbf{x}_n$ denote the columns of X, Eq. (1) can be equivalently written as

$$\min_{W\in\mathbb{R}^{d\times k}:W^\top W=I} -W^\top\left(\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top\right)W, \tag{2}$$

which reveals that the solution is also the top k eigenvectors of the covariance matrix $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$. In this paper, we will mostly focus on the simplest possible form of this problem, where $k=1$, in which case the above reduces to

$$\min_{\mathbf{w}:\|\mathbf{w}\|_2=1} -\mathbf{w}^\top\left(\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{w}, \tag{3}$$

and our goal is to find the top eigenvector $\mathbf{v}_1$. However, as discussed later, the algorithm to be presented can be readily extended to solve Eq. (2) for $k > 1$.

When the data size n and the dimension d are modest, this problem can be solved exactly by a full singular value decomposition of X. However, the required runtime is $\mathcal{O}\left(\min\{nd^2, n^2d\}\right)$, which is prohibitive in large-scale applications. A common alternative is to use iterative methods such as power iterations or more sophisticated variants (Golub & van Loan, 2013). If the covariance matrix has bounded spectral norm and an eigengap λ between its first and second eigenvalues, then these algorithms can be shown to produce a unit vector which is ϵ-far from $\mathbf{v}_1$ (or $-\mathbf{v}_1$) after $\mathcal{O}\left(\frac{\log(1/\epsilon)}{\lambda^p}\right)$ iterations (where e.g. $p=1$ for power iterations). However, each iteration involves multiplying one or more vectors by the covariance matrix $\frac{1}{n}\sum_i \mathbf{x}_i\mathbf{x}_i^\top$. Letting $d_s \in [0,d]$ denote the average sparsity (number of non-zero entries) in each $\mathbf{x}_i$, this requires $\mathcal{O}(d_s n)$ time by passing through the entire data. Thus, the total runtime is $\mathcal{O}\left(\frac{d_s n\log(1/\epsilon)}{\lambda^p}\right)$. When λ is small, this is equivalent to many passes over the data, which can be prohibitive for large datasets.

An alternative to these deterministic algorithms are stochastic and incremental algorithms (e.g. (Krasulina, 1969; Oja, 1982; Oja & Karhunen, 1985) and more recently, (Arora et al., 2012; Mitliagkas et al., 2013; Arora et al., 2013; De Sa et al., 2014)). In contrast to the algorithms above, these algorithms perform much cheaper iterations by choosing some $\mathbf{x}_i$ (uniformly at random or otherwise), and updating the current iterate using only $\mathbf{x}_i$. In general, the runtime of each iteration is only $\mathcal{O}(d_s)$. On the flip side, due to their stochastic and incremental nature, the convergence rate (when known) is quite slow, with the number of required iterations scaling linearly with $1/\epsilon$ and additional problem parameters. This is useful for getting a low to medium-accuracy solution, but is prohibitive when a high-accuracy solution is required.

In this paper, we propose a new stochastic algorithm for solving Eq. (3), denoted as VR-PCA [2], which for bounded data and under suitable assumptions, has provable runtime of

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

This algorithm combines the advantages of the previously discussed approaches, while avoiding their main pitfalls: On one hand, the runtime depends only logarithmically on the accuracy ϵ, so it is suitable to get high-accuracy solutions; while on the other hand, the runtime scales as the *sum* of the data size n and a factor involving the eigengap parameter λ, rather than their product. This means that the algorithm is still applicable when λ is relatively small. In fact, as long as $\lambda \geq \Omega(1/\sqrt{n})$, this runtime bound is better than those mentioned earlier, and equals $d_s n$ up to logarithmic factors: Proportional to the time required to perform a single scan of the data.

VR-PCA builds on a recently-introduced technique for stochastic gradient variance reduction (see (Johnson & Zhang, 2013) as well as (Mahdavi et al., 2013; Konecný & Richtárik, 2013), and (Frostig et al., 2014) in a somewhat different context). However, the setting in which we apply this technique is quite different from previous works, which crucially relied on the strong convexity of the optimization problem, and often assume an unconstrained domain. In contrast, our algorithm attempts to minimize the function in Eq. (3), which is nowhere convex, let alone strongly convex (in fact, it is *concave* everywhere). As a result, the analysis in previous papers is inapplicable, and we require a new and different analysis to understand the performance of the algorithm.

2. Algorithm and Analysis

The pseudo-code of our algorithm appears as Algorithm 1 below. We refer to a single execution of the inner loop as an *iteration*, and each execution of the outer loop as an *epoch*. Thus, the algorithm consists of several epochs, each of which consists of running m iterations.

[2]VR stands for "variance-reduced".

Algorithm 1 VR-PCA

```
Parameters: Step size $\eta$, 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
```

To understand the structure of the algorithm, it is helpful to consider first the well-known Oja's algorithm for stochastic PCA optimization (Oja, 1982), on which our algorithm is based. In our setting, this rule is reduced to repeatedly sampling $\mathbf{x}_{i_t}$ uniformly at random, and performing the update

$$\mathbf{w}'_t = \mathbf{w}_{t-1} + \eta_t\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top\mathbf{w}_{t-1} \ , \quad \mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}_t.$$

Letting $A = \frac{1}{n}XX^\top = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top$, this can be equivalently rewritten as

$$\begin{aligned}\mathbf{w}'_t &= (I+\eta_t A)\mathbf{w}_{t-1} + \eta_t\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)\mathbf{w}_{t-1}\\ \mathbf{w}_t &= \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}_t.\end{aligned} \tag{4}$$

Thus, at each iteration, the algorithm performs a power iteration (using a shifted and scaled version of the matrix A), adds a stochastic zero-mean term $\eta_t\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)\mathbf{w}_{t-1}$, and projects back to the unit sphere. Recently, (Balsubramani et al., 2013) gave a rigorous finite-time analysis of this algorithm, showing that if $\eta_t = \mathcal{O}(1/t)$, then under suitable conditions, we get a convergence rate of $\mathcal{O}(1/T)$.

The reason for the relatively slow convergence rate of this algorithm is the constant variance of the stochastic term added in each step. Inspired by recent variance-reduced stochastic methods for convex optimization (Johnson & Zhang, 2013), we change the algorithm in a way which encourages the variance of the stochastic term to decay over time. Specifically, we can rewrite the update in each iteration of our VR-PCA algorithm as

$$\begin{aligned}\mathbf{w}'_t &= (I+\eta A)\mathbf{w}_{t-1} + \eta\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)(\mathbf{w}_{t-1} - \tilde{\mathbf{w}}_{s-1})\\ \mathbf{w}_t &= \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}_t,\end{aligned} \tag{5}$$

where $\tilde{\mathbf{w}}_{s-1}$ is the vector computed at the beginning of each epoch. Comparing Eq. (5) to Eq. (4), we see that our algorithm also performs a type of power iteration, followed by adding a stochastic term zero-mean term. However, our algorithm picks a fixed step size η, which is more aggressive that a decaying step size η_t. Moreover, the variance of the stochastic term is no longer constant, but rather controlled by $\|\mathbf{w}_{t-1}-\tilde{\mathbf{w}}_{s-1}\|$. As we get closer to the optimal solution, we expect that both $\tilde{\mathbf{w}}_{s-1}$ and $\mathbf{w}_{t-1}$ will be closer and closer to each other, leading to decaying variance, and a much faster convergence rate, compared to Oja's algorithm.

Before continuing to the algorithm's analysis, we make two important remarks:

Remark 1. *To generalize the algorithm to find multiple singular vectors (i.e. solve Eq. (2) for $k > 1$), one option is to replace the vectors $\mathbf{w}_t, \mathbf{w}'_t, \tilde{\mathbf{w}}, \tilde{\mathbf{u}}$ by $d \times k$ matrices $W_t, W'_t, \tilde{W}, \tilde{U}$, and replace the normalization step $\frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}'_t$ by an orthogonalization step[3]. This generalization is completely analogous to how iterative algorithms such as power iterations and Oja's algorithm are generalized to the $k > 1$ case, and the same intuition discussed above still holds. This is also the option used in our experiments. Another option is to recover the singular vectors one-by-one via matrix deflation: First recover the leading vector $\mathbf{v}_1$, and then iteratively recover the leading eigenvector of the deflated matrix $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top - \sum_{l=1}^{j-1}\mathbf{v}_l\mathbf{v}_l^\top$, which is precisely $\mathbf{v}_j$. This is a standard method to extend power iteration algorithms to recover multiple eigenvectors, and our algorithm can be applied to solve it. Algorithmically, one simply needs to replace each computation of the form $\mathbf{x}\mathbf{x}^\top\mathbf{w}$ with $\left(\mathbf{x}\mathbf{x}^\top - \sum_{l=1}^{j-1}\mathbf{v}_l\mathbf{v}_l^\top\right)\mathbf{w}$. A disadvantage of this approach is that it requires a positive eigengap between all top k singular values, otherwise our algorithm may not converge.*

Remark 2. *Using a straightforward implementation, the runtime of each iteration is $\mathcal{O}(d)$, and the total runtime of each epoch is $\mathcal{O}(dm + d_s n)$, where d_s is the average sparsity of the data points $\mathbf{x}_i$. However, a more careful implementation can improve this to $\mathcal{O}(d_s(m+n))$. The trick is to maintain each $\mathbf{w}_t$ as $\alpha\mathbf{g} + \beta\tilde{\mathbf{u}}$, plus a few additional scalars, and in each iteration perform only a sparse update of $\mathbf{g}$, and updates of the scalars, all in $\mathcal{O}(d_s)$ amortized time. See the supplementary material, Appendix B for more details.*

A formal analysis of the algorithm appears as Thm. 1 below. See Sec. 3 for further discussion of the choice of parameters in practice.

[3]I.e. given W'_t, return W_t with the same column space such that $W_t^\top W_t = I$. Note that the algorithm relies on W_t remaining parameterically close to previous iterates, and W'_t is a relatively small perturbation of of an orthogonal W_{t-1}. Therefore, it's important to use an orthogonalization procedure such that W_t is close to W'_t if W'_t is nearly orthogonal, such as Gram-Schmidt.

Theorem 1. *Define A as $\frac{1}{n}XX^\top = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top$, and let $\mathbf{v}_1$ be an eigenvector corresponding to its largest eigenvalue. Suppose that*

- *$\max_i \|\mathbf{x}_i\|^2 \le r$ for some $r > 0$.*
- *A has eigenvalues $s_1 > s_2 \ge \ldots \ge s_d$, where $s_1 - s_2 = \lambda$ for some $\lambda > 0$.*
- *$\langle\tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle \ge \frac{1}{\sqrt{2}}$.*

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

$$\eta \le \frac{c_1\delta^2}{r^2}\lambda \quad , \quad m \ge \frac{c_2\log(2/\delta)}{\eta\lambda}$$
$$m\eta^2r^2 + r\sqrt{m\eta^2\log(2/\delta)} \le c_3, \tag{6}$$

(where c_1, c_2, c_3 designates 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

$$\langle\tilde{\mathbf{w}}_T, \mathbf{v}_1\rangle^2 \;\ge\; 1-\epsilon.$$

The proof of the theorem is provided in Sec. 4. It is easy to verify that for any fixed δ, Eq. (6) holds for any sufficiently large m on the order of $\frac{1}{\eta\lambda}$, as long as η is chosen to be sufficiently smaller than λ/r^2. Therefore, by running the algorithm for $m = \Theta\left((r/\lambda)^2\right)$ iterations per epoch, and $T = \Theta(\log(1/\epsilon))$ epochs, we get accuracy ϵ with high probability[4] $1 - \lceil\log_2(1/\epsilon)\rceil\delta$. Since each epoch requires $\mathcal{O}(d_s(m+n))$ time to implement, we get a total runtime of

$$\mathcal{O}\left(d_s\left(n + \left(\frac{r}{\lambda}\right)^2\right)\log\left(\frac{1}{\epsilon}\right)\right), \tag{7}$$

establishing an exponential convergence rate. If $\lambda/r \ge \Omega(1/\sqrt{n})$, then the runtime is $\mathcal{O}(d_s n\log(1/\epsilon))$ – up to log-factors, proportional to the time required just to scan the data once.

The theorem assumes that we initialize the algorithm with $\tilde{\mathbf{w}}_0$ for which $\langle\tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle \ge \frac{1}{\sqrt{2}}$. This is not trivial, since if we have no prior knowledge on $\mathbf{v}_1$, and we choose $\tilde{\mathbf{w}}_0$ uniformly at random from the unit sphere, then it is well-known that $|\langle\tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle| \le \mathcal{O}(1/\sqrt{d})$ with high probability. Thus, the theorem should be interpreted as analyzing the

[4]Strictly speaking, this statement is non-trivial only in the regime of ϵ where $\log\left(\frac{1}{\epsilon}\right) \ll \frac{1}{\delta}$, but if δ is a reasonably small ($\ll 1$), then this is the practically relevant regime. Moreover, as long as the success probability is positive, we can get an algorithm which succeeds with exponentially high probability by an amplification argument: Simply run several independent instantiations of the algorithm, and pick the solution $\mathbf{w}$ for which $\mathbf{w}^\top\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{w}$ is largest.

algorithm's convergence after an initial "burn-in" period, which results in some $\tilde{\mathbf{w}}_0$ with a certain constant distance from $\mathbf{v}_1$. This period requires a separate analysis, which we leave to future work. However, since we only need to get to a constant distance from $\mathbf{v}_1$, the runtime of that period is independent of the desired accuracy ϵ. Moreover, we note that in our experiments (see Sec. 3), even when initialized from a random point, no "burn-in" period is discernable, and the algorithm seems to enjoy the same exponential convergence rate starting from the very first epoch. Finally, since the variance-reduction technique only kicks in once we are relatively close to the optimum, it is possible to use some different stochastic algorithm with finite-time analysis, such as Oja's algorithm (e.g. (Balsubramani et al., 2013)) or (De Sa et al., 2014) to get to this constant accuracy, from which point our algorithm and analysis takes over (for example, the algorithm of (De Sa et al., 2014) would require at most $\mathcal{O}(d/\lambda^2)$ iterations, starting from a randomly chosen point, according to their analysis). In any case, note that some assumption on $\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1 \rangle$ being bounded away from 0 must hold, otherwise the algorithm may fail to converge in the worst-case (a similar property holds for power iterations, and follows from the non-convex nature of the optimization problem).

3. Experiments

We now turn to present some experiments, which demonstrate the performance of the VR-PCA algorithm. Rather than tuning its parameters, we used the following fixed heuristic: The epoch length m was set to n (number of data points, or columns in the data matrix), and η was set to $\eta = \frac{1}{\bar{r}\sqrt{n}}$, where $\bar{r} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i\|^2$ is the average squared norm of the data. The choice of $m = n$ ensures that at each epoch, the runtime is about equally divided between the stochastic updates and the computation of $\tilde{\mathbf{u}}$. The choice of η is motivated by our theoretical analysis, which requires η on the order of $1/(\max_i \|\mathbf{x}_i\|^2\sqrt{n})$ in the regime where m should be on the order of n. Also, note that this choice of η can be readily computed from the data, and doesn't require knowledge of λ.

First, we performed experiments on several synthetic random datasets (where $n = 200000, d = 10000$), with different choices of eigengap[5] λ. For comparison, we also implemented Oja's algorithm, using several different step sizes, as well as power iterations[6]. All algorithms were initialized from the same random vector, chosen uniformly at random from the unit ball. Note that compared to our analysis, this makes things harder for our algorithm, since we require it to perform well also in the 'burn-in' phase. The results are displayed in figure 1, and we see that for all values of λ considered, VR-PCA converges much faster than all versions of Oja's algorithm, on which it is based, as well as power iterations, even though we did not tune its parameters. Moreover, since the y-axis is in logarithmic scale, we see that the convergence rate is indeed exponential in general, which accords with our theory. In contrast, the convergence rate of Oja's algorithm (no matter which step size is chosen) appears to be sub-exponential. This is not surprising, since the algorithm does not leverage the finite nature of the training data, and the inherent variance in its updates does not decrease exponentially fast. A similar behavior will occur with other purely stochastic algorithms in the literature, such as (Arora et al., 2012; Mitliagkas et al., 2013; De Sa et al., 2014).

Next, we performed a similar experiment using the training data of the well-known MNIST and CCAT datasets. The MNIST data matrix size is 784×70000, and was pre-

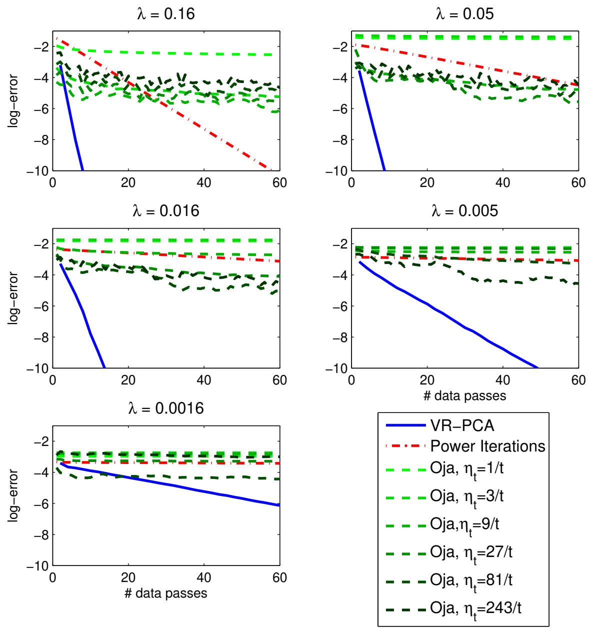

Figure 1. Results for synthetic data. Each plot represents results for a single dataset with eigengap λ, and compares the performance of VR-PCA to power iterations and Oja's algorithm with different step sizes η_t. In each plot, the x-axis represents the number of effective data passes (assuming 2 per epoch for VR-PCA), and the y-axis equals $\log_{10}\left(1 - \frac{\|X^\top \mathbf{w}\|^2}{\max_{\mathbf{v}:\|\mathbf{v}\|=1}\|X^\top \mathbf{v}\|^2}\right)$, where $\mathbf{w}$ is the vector obtained so far.

[5]For each choice of λ, we constructed a $d \times d$ diagonal matrix D, with diagonal $(1, 1-\lambda, 1-1.1\lambda, \ldots, 1-1.4\lambda, q_1, q_2, \ldots)$ where $q_i = |g_i|/d$ and each g_i was chosen according to a standard Gaussian distribution. We then let $X = UDV^\top$, where U and V are random $d \times d$ and $n \times d$ orthogonal matrices. This results in a data matrix X whose spectrum is the same as D.

[6]We note that more sophisticated iterative algorithms, such as the Lanczos method, can attain better performance than power iterations. However, they are not directly comparable to power iterations and VR-PCA, since they are inherently more complex and can require considerably more memory.

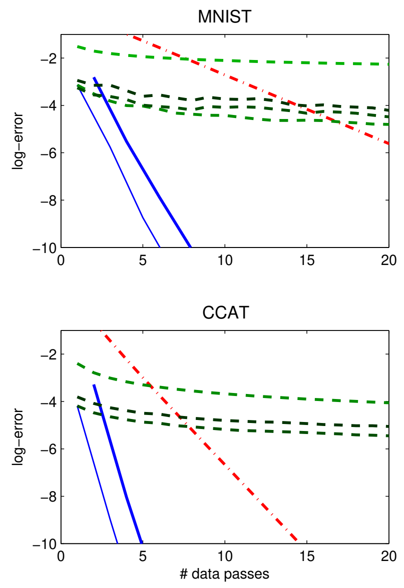

Figure 2. Results for the MNIST and CCAT datasets, using the same algorithms as in Fig. 1, as well as the hybrid method described in the text (represented by a thinner plain line). See Fig. 1 for a legend.

processed by centering the data and dividing each coordinate by its standard deviation times the squared root of the dimension. The CCAT data matrix is sparse (only 0.16% of entries are non-zero), of size 23149×781265, and was used as-is. The results appear in figure 2. We also present the results for a simple hybrid method, which initializes the VR-PCA algorithm with the result of running n iterations of Oja's algorithm. The decaying step size of Oja's algorithm is more suitable for the initial phase, and the resulting hybrid algorithm can perform better than each algorithm alone.

Finally, we present a similar experiment on the MNIST and CCAT datasets, where this time we attempt to recover $k > 1$ singular vectors using the generalization of VR-PCA discussed in remark 1. A similar generalization was also employed with the competitors. The results are displayed in figure 3, and are qualitatively similar to the $k = 1$ case.

4. Proof of Thm. 1

To simplify the presentation of the proof, we use a few important conventions:

- Note that the algorithm remains the same if we divide each $\mathbf{x}_i$ by $\sqrt{r}$, and multiply η by r. Since $\max_i \|\mathbf{x}_i\|^2 \leq r$, this corresponds to running the algorithm with step-size ηr rather than η, on a re-scaled dataset of points with squared norm at most 1, and with an eigengap of λ/r instead of λ. Therefore, we can simply analyze the algorithm assuming that $\max_i \|\mathbf{x}_i\|^2 \leq 1$, and in the end plug in λ/r instead of λ, and ηr instead of η, to get a result which holds for data with squared norm at most r.

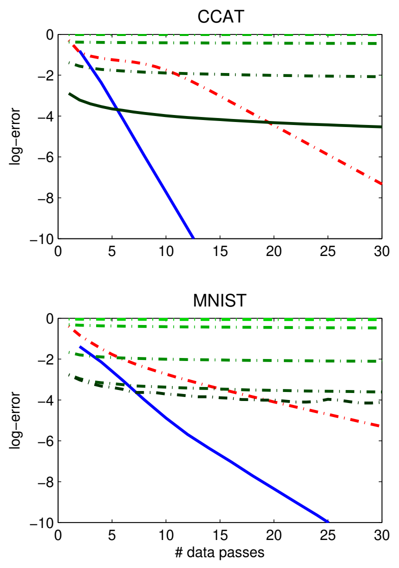

Figure 3. Results for MNIST (for $k = 6$ singular vectors) and CCAT (for $k = 3$ singular vectors). The y-axis here equals $\log_{10}\left(1 - \frac{\|X^\top W\|_F^2}{\max_{V:V^\top V=I}\|X^\top V\|_F^2}\right)$, with $W \in \mathbb{R}^{d\times k}$ being the current iterate. This directly generalizes the performance measure used in previous figures for the $k > 1$ case. See Fig. 1 for a legend.

- Let $A = \sum_{i=1}^d s_i \mathbf{v}_i \mathbf{v}_i^\top$ be an eigendecomposition of A, where $s_1 > s_2 \geq \ldots \geq s_d$, $s_1 - s_2 = \lambda > 0$, and $\mathbf{v}_1, \ldots, \mathbf{v}_d$ are orthonormal vectors. Following the discussion above, we assume that $\max_i \|\mathbf{x}_i\|^2 \leq 1$ and therefore $\{s_1, \ldots, s_d\} \subset [0, 1]$.
- Throughout the proof, we use c to designate positive numerical constants, whose value can vary at different places (even in the same line or expression).

Part I: Establishing a Stochastic Recurrence Relation

We begin by focusing on a single epoch of the algorithm, and a single iteration t, and analyze how $1 - \langle \mathbf{w}_t, \mathbf{v}_1\rangle^2$ evolves during that iteration. The key result we need is the following lemma:

Lemma 1. *Suppose that* $\langle \mathbf{w}_t, \mathbf{v}_1\rangle \geq \frac{1}{2}$, *and that* $\langle \tilde{\mathbf{w}}_{s-1}, \mathbf{v}_1\rangle \geq 0$. *If* $\eta \leq c\lambda$, *then*

$$\mathbb{E}\left[\left(1 - \langle \mathbf{w}_{t+1}, \mathbf{v}_1\rangle^2\right)\middle|\mathbf{w}_t\right] \leq \left(1 - \frac{\eta\lambda}{16}\right)\left(1 - \langle \mathbf{w}_t, \mathbf{v}_1\rangle^2\right) + c\eta^2\left(1 - \langle \tilde{\mathbf{w}}_{s-1}, \mathbf{v}_1\rangle^2\right)$$

for certain positive numerical constants c.

The proof is rather technical, and appears in the supplementary material, Appendix A. Below, we provide an abridged version without some of the technical details.

Since we focus on a particular epoch s, let us drop the subscript from $\tilde{\mathbf{w}}_{s-1}$, and denote it simply at $\tilde{\mathbf{w}}$. Rewriting the update equations from the algorithm, we have that $\mathbf{w}_{t+1} = \frac{\mathbf{w}'_{t+1}}{\|\mathbf{w}'_{t+1}\|}$, where

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

and $\mathbf{x}$ is the random instance chosen at iteration t.

It is easy to verify that $\langle \mathbf{w}'_{t+1}, \mathbf{v}_i \rangle = a_i + z_i$, where

$$a_i = (1 + \eta s_i)\langle \mathbf{w}_t, \mathbf{v}_i \rangle \;\;,\;\; z_i = \eta \mathbf{v}_i^\top (\mathbf{x}\mathbf{x}^\top - A)(\mathbf{w}_t - \tilde{\mathbf{w}}).$$

Moreover, since $\mathbf{v}_1, \ldots, \mathbf{v}_d$ form an orthonormal basis in $\mathbb{R}^d$, we have

$$\|\mathbf{w}'_{t+1}\|^2 = \sum_{i=1}^d \langle \mathbf{v}_i, \mathbf{w}'_{t+1} \rangle^2 = \sum_{i=1}^d (a_i + z_i)^2.$$

Let $\mathbb{E}$ denote expectation with respect to $\mathbf{x}$, conditioned on $\mathbf{w}_t$. Based on the above, we have

$$\mathbb{E}\left[\langle \mathbf{w}_{t+1}, \mathbf{v}_1 \rangle^2\right] = \mathbb{E}\left[\left\langle \frac{\mathbf{w}'_{t+1}}{\|\mathbf{w}'_{t+1}\|}, \mathbf{v}_1 \right\rangle^2\right]$$
$$= \mathbb{E}\left[\frac{\langle \mathbf{w}'_{t+1}, \mathbf{v}_1 \rangle^2}{\|\mathbf{w}'_{t+1}\|^2}\right] = \mathbb{E}\left[\frac{(a_1 + z_1)^2}{\sum_{i=1}^d (a_i + z_i)^2}\right]. \qquad (8)$$

Note that conditioned on $\mathbf{w}_t$, the quantities $a_1 \ldots a_d$ are fixed, whereas $z_1 \ldots z_d$ are random variables (depending on the random choice of $\mathbf{x}$) over which we take an expectation.

The first step of the proof is to simplify Eq. (8), by pushing the expectations inside the numerator and the denominator. An analysis based on a second-order Taylor expansion reveals that Eq. (8) can be lower bounded by

$$\frac{\mathbb{E}\left[(a_1 + z_1)^2\right]}{\mathbb{E}\left[\sum_{i=1}^d (a_i + z_i)^2\right]} - c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2,$$

and since each z_i is zero-mean, this equals

$$\frac{\mathbb{E}\left[a_1^2 + z_1^2\right]}{\mathbb{E}\left[\sum_{i=1}^d (a_i^2 + z_i^2)\right]} - c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2 \qquad (9)$$

It can be shown that $\sum_{i=1}^d z_i^2 \leq c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2$, hence we can lower bound the above by

$$\frac{a_1^2}{\sum_{i=1}^d a_i^2 + c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2} - c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2. \qquad (10)$$

Plugging in $a_i = (1 + \eta s_i)\langle \mathbf{w}_t, \mathbf{v}_i \rangle$ and performing several simplifications, it can be shown that Eq. (10) (and hence $\mathbb{E}[\langle \mathbf{w}_{t+1}, \mathbf{v}_1 \rangle^2]$) is lower bounded by

$$\langle \mathbf{w}_t, \mathbf{v}_1 \rangle^2 \left(1 + \frac{\eta\lambda}{2}\left(1 - \langle \mathbf{w}_t, \mathbf{v}_1 \rangle^2\right) - c\eta^2 \|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2\right). \qquad (11)$$

We now get rid of the $\|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2$ term, by noting that since $(x+y)^2 \leq 2(x^2 + y^2)$ and $\|\mathbf{w}_t\| = \|\mathbf{v}_1\| = 1$,

$$\begin{aligned}\|\mathbf{w}_t - \tilde{\mathbf{w}}\|^2 &\leq \left(\|\mathbf{w}_t - \mathbf{v}_1\| + \|\tilde{\mathbf{w}} - \mathbf{v}_1\|\right)^2 \\ &\leq 2\left(\|\mathbf{w}_t - \mathbf{v}_1\|^2 + \|\tilde{\mathbf{w}} - \mathbf{v}_1\|^2\right) \\ &= 2\left(2 - 2\langle \mathbf{w}_t, \mathbf{v}_1 \rangle + 2 - 2\langle \tilde{\mathbf{w}}, \mathbf{v}_1 \rangle\right).\end{aligned}$$

Plugging this back, and performing several manipulations, we finally get

$$\begin{aligned}&\mathbb{E}[1 - \langle \mathbf{w}_{t+1}, \mathbf{v}_1 \rangle^2] \\ &\leq \;\left(1 - \langle \mathbf{w}_t, \mathbf{v}_1 \rangle^2\right)\left(1 - \frac{\eta\lambda}{16}\right) + c\eta^2\left(1 - \langle \tilde{\mathbf{w}}, \mathbf{v}_1 \rangle^2\right)\end{aligned}$$

as required. Note that to get this bound, we assumed at several places that η is smaller than either a constant, or a constant factor times λ (which is at most 1). Hence, the bound holds by assuming $\eta \leq c\lambda$ for a sufficiently small numerical c.

Part II: Solving the Recurrence Relation for a Single Epoch

As before, since we focus on a single epoch, we drop the subscript from $\tilde{\mathbf{w}}_{s-1}$ and denote it simply as $\tilde{\mathbf{w}}$.

Suppose that $\eta = \alpha\lambda$, where α is a sufficiently small constant to be chosen later. Also, let

$$b_t = 1 - \langle \mathbf{w}_t, \mathbf{v}_1 \rangle^2 \quad \text{and} \quad \tilde{b} = 1 - \langle \tilde{\mathbf{w}}, \mathbf{v}_1 \rangle^2.$$

Then Lemma 1 tells us that if α is sufficiently small, $b_t \leq \frac{3}{4}$, and $\langle \tilde{\mathbf{w}}, \mathbf{v}_1 \rangle \geq 0$, then

$$\mathbb{E}\left[b_{t+1} | \mathbf{w}_t\right] \;\leq\; \left(1 - \frac{\alpha}{16}\lambda^2\right) b_t + c\alpha^2\lambda^2\tilde{b}. \qquad (12)$$

Lemma 2. *Let B be the event that $b_t \leq \frac{3}{4}$ for all $t = 0, 1, 2, \ldots, m$. Then for certain positive numerical constants c, if $\alpha \leq c$, and $\langle \tilde{\mathbf{w}}, \mathbf{v}_1 \rangle \geq 0$, then*

$$\mathbb{E}[b_m | B, \mathbf{w}_0] \leq \left(\left(1 - \frac{\alpha}{16}\lambda^2\right)^m + c\alpha\right)\tilde{b}.$$

Proof. Recall that b_t is a deterministic function of the random variable $\mathbf{w}_t$, which depends in turn on $\mathbf{w}_{t-1}$ and the random instance chosen at round m. We assume that $\mathbf{w}_0$ (and hence $\tilde{b}$) are fixed, and consider how b_t evolves as a function of t. Using Eq. (12), we have

$$\begin{aligned}&\mathbb{E}[b_{t+1} | \mathbf{w}_t, B] = \mathbb{E}\left[b_{t+1} | \mathbf{w}_t, b_{t+1} \leq \frac{3}{4}\right] \\ &\;\leq\; \mathbb{E}[b_{t+1} | \mathbf{w}_t] \;\leq\; \left(1 - \frac{\alpha}{16}\lambda^2\right) b_t \;+\; c\alpha^2\lambda^2\tilde{b}.\end{aligned}$$

Note that the first equality holds, since conditioned on $\mathbf{w}_t$, b_{t+1} is independent of $b_1, \ldots, b_t$, so the event B is equivalent to just requiring $b_{t+1} \leq 3/4$.

Taking expectation over $\mathbf{w}_t$ (conditioned on B), we get that

$$\begin{aligned}\mathbb{E}[b_{t+1}|B] &\leq \mathbb{E}\left[\left(1-\frac{\alpha}{16}\lambda^2\right)b_t + c\alpha^2\lambda^2\tilde{b}\middle|B\right] \\ &= \left(1-\frac{\alpha}{16}\lambda^2\right)\mathbb{E}\left[b_t|B\right] + c\alpha^2\lambda^2\tilde{b}.\end{aligned}$$

Unwinding the recursion, and using that $b_0 = \tilde{b}$, we therefore get that

$$\begin{aligned}\mathbb{E}[b_m|B] &\leq \left(1-\frac{\alpha}{16}\lambda^2\right)^m\tilde{b} + c\alpha^2\lambda^2\tilde{b}\sum_{i=0}^{m-1}\left(1-\frac{\alpha}{16}\lambda^2\right)^i \\ &\leq \left(1-\frac{\alpha}{16}\lambda^2\right)^m\tilde{b} + c\alpha^2\lambda^2\tilde{b}\sum_{i=0}^{\infty}\left(1-\frac{\alpha}{16}\lambda^2\right)^i \\ &= \left(1-\frac{\alpha}{16}\lambda^2\right)^m\tilde{b} + c\alpha^2\lambda^2\tilde{b}\frac{1}{(\alpha/16)\lambda^2} \\ &= \left(\left(1-\frac{\alpha}{16}\lambda^2\right)^m + c\alpha\right)\tilde{b}.\end{aligned}$$

as required. □

We now turn to prove that the event B assumed in Lemma 2 indeed holds with high probability:

Lemma 3. *For certain positive numerical constants c, suppose that $\alpha \leq c$, and $\langle\tilde{\mathbf{w}}, \mathbf{v}_1\rangle \geq 0$. Then for any $\beta \in (0,1)$ and m, if*

$$\tilde{b} + cm\alpha^2\lambda^2 + c\sqrt{m\alpha^2\lambda^2\log(1/\beta)} \leq \frac{3}{4}, \tag{13}$$

for a certain numerical constant c, then it holds with probability at least $1-\beta$ that

$$b_t \;\leq\; \tilde{b} + cm\alpha^2\lambda^2 + c\sqrt{m\alpha^2\lambda^2\log(1/\beta)} \;\leq\; \frac{3}{4}$$

for some numerical constant c and for all $t = 0, 1, 2, \ldots, m$, as well as $\langle\mathbf{w}_m, \mathbf{v}_1\rangle \geq 0$.

Proof. To prove the lemma, we analyze the stochastic process $b_0(=\tilde{b}), b_1, b_2, \ldots, b_m$, and use a concentration of measure argument. First, we collect the following facts:

- $\tilde{b} = b_0 \leq \frac{3}{4}$: This directly follows from the assumption stated in the lemma.
- *The conditional expectation of b_{t+1} is close to b_t, as long as $b_t \leq \frac{3}{4}$* : Supposing that $b_t \leq \frac{3}{4}$ for some t, and α is sufficiently small, then by Eq. (12),

$$\mathbb{E}\left[b_{t+1}|\mathbf{w}_t\right] \;\leq\; \left(1-\frac{\alpha}{16}\lambda^2\right)b_t + c\alpha^2\lambda^2\tilde{b} \;\leq\; b_t + c\alpha^2\lambda^2\tilde{b}.$$

- $|b_{t+1} - b_t|$ *is bounded by $c\alpha\lambda$*: Since the norm of $\mathbf{w}_t, \mathbf{v}_1$ is 1, we have

$$\begin{aligned}|b_{t+1}-b_t| &= \left|\langle\mathbf{w}_{t+1},\mathbf{v}_1\rangle^2 - \langle\mathbf{w}_t,\mathbf{v}_1\rangle^2\right| \\ &= |\langle\mathbf{w}_{t+1},\mathbf{v}_1\rangle + \langle\mathbf{w}_t,\mathbf{v}_1\rangle| * |\langle\mathbf{w}_{t+1},\mathbf{v}_1\rangle - \langle\mathbf{w}_t,\mathbf{v}_1\rangle| \\ &\leq 2\,|\langle\mathbf{w}_{t+1},\mathbf{v}_1\rangle - \langle\mathbf{w}_t,\mathbf{v}_1\rangle| \;\leq\; 2\|\mathbf{w}_{t+1}-\mathbf{w}_t\|.\end{aligned}$$

  Recalling the definition of $\mathbf{w}_{t+1}$ in our algorithm, and the fact that the instances $\mathbf{x}_i$ and hence the matrix A are assumed to have norm at most 1, it is easily verified that $\|\mathbf{w}_{t+1}-\mathbf{w}_t\| \leq c\eta \leq c\alpha\lambda$ for some constant c.

Armed with these facts, and using the maximal version of the Hoeffding-Azuma inequality (Hoeffding, 1963), it follows that with probability at least $1-\beta$, it holds simultaneously for all $t = 1, \ldots, m$ (and for $t = 0$ by assumption) that

$$b_t \leq \tilde{b} + mc\alpha^2\lambda^2\tilde{b} + c\sqrt{m\alpha^2\lambda^2\log(1/\beta)}$$

for some constants c, as long as the expression above is less than $\frac{3}{4}$. If the expression is indeed less than $\frac{3}{4}$, then we get that $b_t \leq \frac{3}{4}$ for all t. Upper bounding $\tilde{b}$ and λ by 1, and slightly simplifying, we get the statement in the lemma.

It remains to prove that if $b_t \leq \frac{3}{4}$ for all t, then $\langle\mathbf{w}_m,\mathbf{v}_1\rangle \geq 0$. Suppose on the contrary that $\langle\mathbf{w}_m,\mathbf{v}_1\rangle < 0$. Since $|\langle\mathbf{w}_{t+1},\mathbf{v}_1\rangle - \langle\mathbf{w}_t,\mathbf{v}_1\rangle| \leq \|\mathbf{w}_{t+1}-\mathbf{w}_t\| \leq c\alpha\lambda$ as we've seen earlier, and $\langle\mathbf{w}_0,\mathbf{v}_1\rangle \geq 0$, it means there must have been some $\mathbf{w}_t$ such that $\langle\mathbf{w}_t,\mathbf{v}_1\rangle \leq c\alpha\lambda$. But this means that $b_t = (1-\langle\mathbf{w}_t,\mathbf{v}_1\rangle^2) \geq 1 - c^2\alpha^2\lambda^2 > \frac{3}{4}$ (as long as α is sufficiently small, since we assume λ is bounded), invalidating the assumption that $b_t \leq \frac{3}{4}$ for all t. Therefore, $\langle\mathbf{w}_m,\mathbf{v}_1\rangle \geq 0$ as required. □

Combining Lemma 2 and Lemma 3, and using Markov's inequality, we get the following corollary:

Lemma 4. *Let confidence parameters $\beta, \gamma \in (0,1)$ be fixed. Suppose that $\langle\tilde{\mathbf{w}},\mathbf{v}_1\rangle \geq 0$, and that m, α are chosen such that*

$$\tilde{b} + cm\alpha^2\lambda^2 + c\sqrt{m\alpha^2\lambda^2\log(1/\beta)} \leq \frac{3}{4}$$

for a certain numerical constant c. Then with probability at least $1-(\beta+\gamma)$, it holds that $\langle\mathbf{w}_m,\mathbf{v}_1\rangle \geq 0$, and

$$b_m \leq \frac{1}{\gamma}\left(\left(1-\frac{\alpha}{16}\lambda^2\right)^m + c\alpha\right)\tilde{b}.$$

for some numerical constant c.

Part III: Analyzing the Entire Algorithm's Run

Given the analysis in Lemma 4 for a single epoch, we are now ready to prove our theorem. Let

$$\tilde{b}_s = 1 - \langle\tilde{\mathbf{w}}_s,\mathbf{v}_1\rangle^2.$$

By assumption, at the beginning of the first epoch, we have $\tilde{b}_0 = 1 - \langle \tilde{\mathbf{w}}_0, \mathbf{v}_1 \rangle^2 \leq 1 - \frac{1}{2} = \frac{1}{2}$. Therefore, by Lemma 4, for any $\beta, \gamma \in \left(0, \frac{1}{2}\right)$, if we pick any

$$\begin{aligned} &\alpha \leq \frac{1}{2}\gamma^2 \quad \text{and} \quad m \geq \frac{48\log(1/\gamma)}{\alpha\lambda^2} \\ &\text{such that} \quad \frac{1}{2} + cm\alpha^2\lambda^2 + c\sqrt{m\alpha^2\lambda^2\log(1/\beta)} \leq \frac{3}{4}, \end{aligned} \tag{14}$$

then we get with probability at least $1 - (\beta + \gamma)$ that

$$\tilde{b}_1 \;\leq\; \frac{1}{\gamma}\left(\left(1 - \frac{\alpha\lambda^2}{16}\right)^{\frac{48\log(1/\gamma)}{\alpha\lambda^2}} + \frac{1}{2}\gamma^2\right)\tilde{b}_0$$

Using the inequality $(1 - (1/x))^{ax} \leq \exp(-a)$, which holds for any $x > 1$ and any a, and taking $x = 16/(\alpha\lambda^2)$ and $a = 3\log(1/\gamma)$, we can upper bound the above by

$$\begin{aligned} &\frac{1}{\gamma}\left(\exp\left(-3\log\left(\frac{1}{\gamma}\right)\right) + \frac{1}{2}\gamma^2\right)\tilde{b}_0 \\ &= \frac{1}{\gamma}\left(\gamma^3 + \frac{1}{2}\gamma^2\right)\tilde{b}_0 \;\leq\; \gamma\tilde{b}_0. \end{aligned}$$

Therefore, we get that $\tilde{b}_1 \leq \gamma\tilde{b}_0$. Moreover, again by Lemma 4, we have $\langle \tilde{\mathbf{w}}_1, \mathbf{v}_1 \rangle \geq 0$. Since $\tilde{b}_1$ is only smaller than $\tilde{b}_0$, the conditions of Lemma 4 are fulfilled for $\tilde{b} = \tilde{b}_1$, so again with probability at least $1 - (\beta + \gamma)$, by the same calculation, we have $\tilde{b}_2 \leq \gamma\tilde{b}_1 \leq \gamma^2\tilde{b}_0$. Repeatedly applying Lemma 4 and using a union bound, we get that after T epochs, with probability at least $1 - T(\beta + \gamma)$,

$$1 - \langle \tilde{\mathbf{w}}_T, \mathbf{v}_1 \rangle^2 \;=\; \tilde{b}_T \;\leq\; \gamma^T\tilde{b}_0 \;<\; \gamma^T.$$

Therefore, for any desired accuracy parameter ϵ, we simply need to use $T = \left\lceil \frac{\log(1/\epsilon)}{\log(1/\gamma)} \right\rceil$ epochs, and get $1 - \langle \tilde{\mathbf{w}}_T, \mathbf{v}_1 \rangle^2 \leq \epsilon$ with probability at least $1 - T(\beta + \gamma) = 1 - \left\lceil \frac{\log(1/\epsilon)}{\log(1/\gamma)} \right\rceil (\beta + \gamma)$.

Using a confidence parameter δ, we pick $\beta = \gamma = \frac{\delta}{2}$, which ensures that the accuracy bound above holds with probability at least

$$1 - \left\lceil \frac{\log(1/\epsilon)}{\log(2/\delta)} \right\rceil \delta \geq 1 - \left\lceil \frac{\log(1/\epsilon)}{\log(2)} \right\rceil \delta = 1 - \left\lceil \log_2\left(\frac{1}{\epsilon}\right) \right\rceil \delta.$$

Substituting this choice of β, γ into Eq. (14), and recalling that the step size η equals $\alpha\lambda$, we get that $\langle \tilde{\mathbf{w}}_T, \mathbf{v}_1 \rangle^2 \geq 1 - \epsilon$ with probability at least $1 - \lceil \log_2(1/\epsilon) \rceil \delta$, provided that

$$\begin{aligned} \eta \leq c\delta^2\lambda \quad , \quad m \geq \frac{c\log(2/\delta)}{\eta\lambda}, \\ m\eta^2 + \sqrt{m\eta^2\log(2/\delta)} \leq c \end{aligned}$$

for suitable constants c.

To get the theorem statement, recall that this analysis pertains to data whose squared norm is bounded by 1. By the reduction discussed at the beginning of the proof, we can apply it to data with squared norm at most r, by replacing λ with λ/r, and η with ηr, leading to the condition

$$\begin{aligned} \eta \leq \frac{c\delta^2}{r^2}\lambda \quad , \quad m \geq \frac{c\log(2/\delta)}{\eta\lambda}, \\ m\eta^2 r^2 + r\sqrt{m\eta^2\log(2/\delta)} \leq c. \end{aligned}$$

Recalling that the different c's correspond to possibly different numerical constants, we get the theorem statement.

5. Discussion

In this paper, we presented and analyzed a stochastic algorithm for PCA and SVD with an exponential convergence rate. Under suitable assumptions, the runtime scales as the *sum* of the data size n and an eigengap factor $\frac{1}{\lambda^2}$, and *logarithmically* in the required accuracy ϵ. In contrast, the runtime of previous iterative methods scale either as the *product* of n and an eigengap factor, or *polynomially* in ϵ.

This work leaves several open questions. First, we note that in the regime of moderate data size n (in particular, when n is dominated by $(r/\lambda)^2$), the required runtime scales with $1/\lambda^2$, which is inferior to the deterministic methods discussed in Sec. 1. Second, in the context of strongly convex optimization problems, the variance-reduced technique we use leads to algorithms with runtime $\mathcal{O}\left(d\left(n + \frac{1}{\lambda}\right)\log\left(\frac{1}{\epsilon}\right)\right)$, where λ is the strong convexity parameter of the problem (Johnson & Zhang, 2013). Comparing this with our algorithm's runtime, and drawing a parallel between strong convexity and the eigengap in PCA problems, it is tempting to conjecture that the $1/\lambda^2$ in our runtime analysis can be improved at least to $1/\lambda$. However, we don't know if this is true, or whether the $1/\lambda^2$ factor is necessary in our setting. Third, it remains to analyze the behavior of the algorithm starting from a randomly initialized point, before we obtain some $\tilde{\mathbf{w}}_0$ sufficiently close to the optimum. Experimentally, this does not seem to be an issue, but a full analysis would be more satisfactory, and might give more guidance on how to optimally choose the step size. Finally, we believe our formal analysis should be extendable to the $k > 1$ case (see remark 1), and that the dependence on the maximal squared norm of the data can be relaxed to a dependence on the average squared norm or some weaker moment conditions.

Acknowledgments

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. We thank Huy Nguyen for spotting a bug in the proof of lemma 1 in an earlier version of this paper.

References

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.

De Sa, C., Olukotun, K., and Ré, C. Global convergence of stochastic gradient descent for some nonconvex matrix problems. *arXiv preprint arXiv:1411.1134*, 2014.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. Competing with the empirical risk minimizer in a single pass. *CoRR*, abs/1412.6606, 2014.

Golub, G. and van Loan, C. *Matrix computations (4. ed.)*. Johns Hopkins University Press, 2013.

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

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

Konecný, J. and Richtárik, P. Semi-stochastic gradient descent methods. *CoRR*, abs/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.

Mahdavi, M., Zhang, L., and Jin, R. Mixed optimization for smooth functions. In *NIPS*, 2013.

Mitliagkas, I., Caramanis, C., and Jain, P. Memory limited, streaming PCA. In *NIPS*, 2013.

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

Oja, E. and Karhunen, J. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of mathematical analysis and applications*, 106(1):69–84, 1985.