Accelerated Stochastic Power Iteration

Peng Xu[1] **Bryan He**[1] **Christopher De Sa**[2] **Ioannis Mitliagkas**[3] **Christopher Ré**[1]
[1]Stanford University [2]Cornell University [3]University of Montréal

Abstract

Principal component analysis (PCA) is one of the most powerful tools for analyzing matrices in machine learning. In this paper, we study methods to accelerate power iteration in the stochastic setting by adding a momentum term. While in the deterministic setting, power iteration with momentum has optimal iteration complexity, we show that naively adding momentum to a stochastic method does not always result in acceleration. We perform a novel, tight variance analysis that reveals a "breaking-point variance" beyond which this acceleration does not occur. Combining this insight with modern variance reduction techniques yields a simple version of power iteration with momentum that achieves the optimal iteration complexities in both the online and offline setting. Our methods are embarrassingly parallel and can produce wall-clock-time speedups. Our approach is very general and applies to many non-convex optimization problems that can now be accelerated using the same technique.

1 Introduction

Principal Component Analysis (PCA) is a fundamental tool for data processing and visualization in machine learning and statistics [Hot33; Jol02]. PCA captures variable interactions in a high-dimensional dataset by identifying the directions of highest variance: the *principal components*. Modern machine learning problems have become too big for full-pass PCA methods, leading practitioners to *stochastic methods*: algorithms that only ingest a random subset of the available data at every iteration. Stochastic methods have been proposed for the *offline*, or *finite-sample* setting, in which the algorithm is given random access to a finite set of samples, and therefore could perform a full-pass periodically [Sha15]. Other methods target the *online* setting, in which the samples are randomly drawn from a distribution, and full passes are not possible [MCJ13; Bou+15; Jai+16]. Information theoretic bounds [AZL16b] show that the *sample complexity*, which is defined as the number of samples necessary to recover the principal component, is at least $\mathcal{O}(1/\Delta^2)$ in the online setting, where Δ is the eigen-gap. Elegant variants of the power method have been shown to match this lower bound [Jai+16; AZL16b]. However, sample complexity is not a great proxy for run time.

Iteration complexity—the number of outer loop iterations required, when the inner loop is embarrassingly parallel—provides an asymptotic measure of an algorithm's performance on a highly parallel computer. We would like to match the Lanczos algorithm's optimal convergence rate of $\mathcal{O}(1/\sqrt{\Delta})$ iterations from the full-pass setting. Unfortunately, the Lanczos algorithm cannot operate in a stochastic setting and none of the simple stochastic power iteration variants achieve this accelerated iteration complexity. Recently, carefully tuned numerical methods based on approximate matrix inversion [Gar+16; AZL16a] have achieved an accelerated rate in the stochastic setting. However, these methods are significantly more complex than stochastic power iteration and are largely theoretical in nature. This context motivates the question: *is it possible to achieve the optimal sample and iteration complexity with a method as simple as power iteration?*

In this paper, we propose a class of simple PCA algorithms based on the power method that (1) operate in the stochastic setting, (2) have a sample complexity with an asymptotically optimal dependence on the eigen-gap, and (3) have an iteration complexity with an asymptotically optimal dependence on the eigen-gap (i.e. one that matches the worst-case rate for the Lanczos method). As background for our method, we first note that a simple modification of the power iteration, *power iteration with momentum*, achieves the

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

optimal accelerated convergence rate $\mathcal{O}(1/\sqrt{\Delta})$ in the deterministic setting. Our proposed algorithms come from the natural idea of designing an efficient, stochastic version of that method.

We demonstrate that simply adding momentum to a stochastic method like Oja's does not always result in acceleration. Although adding momentum with a fixed learning rate accelerates the initial convergence, it also increases the size of the noise ball. This is because momentum accelerates the convergence of the expected iterates, but also increases the variance, which typically dominates, so no overall acceleration is observed (cf. Section 3). Using Chebyshev polynomials to derive an exact expression for the variance of the iterates of our algorithm, we identify the precise relationship between *sample variance* and *acceleration*. Importantly, we identify the exact break-down point beyond which the variance is too much for acceleration to happen.

Based on this analysis, we can design a stochastic momentum power method that is guaranteed to work. We first propose a *mini-batching* technique to ensure acceleration. However, this algorithm requires increasingly large batch sizes to reach small errors. To remedy this, we additionally propose a *variance reduction* technique, which ensures acceleration with a constant batch size. Both of these techniques are used to speed up computation in stochastic optimization and are embarrassingly parallel. This property allows our method to achieve true *wall-clock time acceleration* even in the online setting, something not possible with state-of-the-art results. Hence, we demonstrate that the more complicated techniques based on approximate matrix inversion are not necessary: *simple momentum-based methods are sufficient to accelerate PCA*.

Our tight variance analysis can apply to general stochastic three-term recurrence (cf. Section 4). It is straightforward to show that randomized Kaczmarz algorithms [SV09; GR15] can be accelerated using momentum for solving linear systems. It also enables many non-convex problems, including matrix completion [JNS13], phase retrieval [CLS15] and subspace tracking [BNR10], to be accelerated using a single technique, and suggests that the same might be true for a larger class of non-convex optimization problems.

Our contributions

- We study the relationship between variance and acceleration by finding an exact characterization of variance for a general class of power iteration variants with momentum in Section 3.1.
- Using this bound, we design an algorithm that uses mini-batching to obtain the optimal iteration and sample complexities for the online setting in Section 3.2.
- We design a second algorithm that uses variance reduction to obtain the optimal rate for the offline setting in Section 3.3. Notably, when operating in the offline setting, we are able to use a batch size that is independent of the target accuracy.
- We demonstrate the acceleration of our variance-reduced methods on real-world network datasets in Section 3.4.

2 Power method with momentum

In this section, we provide background knowledge to help introduce our stochastic method. We begin by describing the basic PCA setup and show that a simple momentum scheme accelerates the standard power method. This momentum scheme, and its connection with the Chebyshev polynomial family, serves as the foundation of our stochastic method.

PCA Let $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ be n data points. The goal of PCA is to find the top eigenvector of the symmetric positive semidefinite (PSD) matrix $\mathbf{A} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T \in \mathbb{R}^{d\times d}$ (the sample covariance matrix) when the data points are centered at the origin. We assume that the target matrix $\mathbf{A}$ has eigenvalues $1 \geq \lambda_1 > \lambda_2 \geq \cdots \lambda_d \geq 0$ with corresponding normalized eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d$. The power method estimates the top eigenvector by repeatedly applying the update step

$$\mathbf{w}_{t+1} = \mathbf{A}\mathbf{w}_t$$

with an initial vector $\mathbf{w}_0 \in \mathbb{R}^d$. After $\mathcal{O}\left(\frac{1}{\Delta}\log\frac{1}{\epsilon}\right)$ steps, the normalized iterate $\mathbf{w}_t/\|\mathbf{w}_t\|$[1] is an ϵ-*accurate estimate* of top principal component. Here ϵ accuracy is measured by the squared sine of the angle between $\mathbf{u}_1$ and $\mathbf{w}_t$, which is $1 - (\mathbf{u}_1^T\mathbf{w}_t)^2/\|\mathbf{w}_t\|^2$.

When λ_1 is close to λ_2 (the eigengap Δ is small), the power method will converge very slowly. To address this, we propose a class of algorithms based on the alternative update step

$$\mathbf{w}_{t+1} = \mathbf{A}\mathbf{w}_t - \beta\mathbf{w}_{t-1}. \qquad (\mathbf{A})$$

We call the extra term, $\beta\mathbf{w}_{t-1}$, the *momentum* term, and β the momentum parameter, in analogy to the heavy ball method [Pol64], which uses the same technique to address poorly conditioned problems in convex optimization. For appropriate settings of β, this *accelerated power method* can converge dramatically faster than the traditional power method; this is not surprising, since the same is true for analogous accelerated methods for convex optimization.

Orthogonal polynomials We now connect the dynamics of the update (A) to the behavior of a family of

[1] The $\|\cdot\|$ in this paper is ℓ_2 norm for vectors and spectral norm for matrices.

Table 1: Asymptotic complexities for variants of the power method to achieve ϵ accuracy, $1-(\mathbf{u}_1^T\mathbf{w})^2 \leq \epsilon$. For momentum methods, we choose the optimal $\beta = \lambda_2^2/4$. Here $\Delta := \lambda_1 - \lambda_2$ is the eigen-gap, σ^2 is the variance of one random sample and r is an a.s. norm bound (see Definition (1)). In $\mathcal{O}$ notation, we omit the factors depending on failure probability δ. Jain et al. [Jai+16] and Shamir [Sha15] give the best known results for stochastic PCA without and with variance reduction respectively. However, neither of these results achieve the optimal iteration complexity. Furthermore, they are not tight in terms of the variance of the problem (i.e. when σ^2 is small, the bounds are loose).

Setting	Algorithm	Number of Iterations	Batch Size	Reference
Deterministic	Power	$\mathcal{O}\left(\frac{1}{\Delta}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	n	[GVL12]
	Lanczos	$\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	n	[GVL12]
	Power+M	$\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	n	**This paper**
Online	Oja's	$\mathcal{O}\left(\frac{\sigma^2}{\Delta^2}\cdot\frac{1}{\epsilon}+\frac{1}{\sqrt{\epsilon}}\right)$	$\mathcal{O}(1)$	[Jai+16]
	Mini-batch Power+M	$\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{\sqrt{d}\sigma^2}{\Delta^{3/2}}\cdot\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$	**This paper**
Offline	VR-PCA	$\mathcal{O}\left(\frac{r^2}{\Delta^2}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}(1)$	[Sha15]
	VR Power+M	$\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\cdot\log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{\sqrt{d}\sigma^2}{\Delta^{3/2}}\right)$	**This paper**

orthogonal polynomials, which allows us to use well-known results about orthogonal polynomials to analyze the algorithm's convergence. Consider the polynomial sequence $p_t(x)$, defined as

$$p_{t+1}(x) = xp_t(x) - \beta p_{t-1}(x), p_0 = 1, p_1 = x/2. \quad (\mathbf{P})$$

According to Favard's theorem [Chi11], this recurrence forms an orthogonal polynomial family—in fact these are scaled Chebyshev polynomials of the first kind. If we use the update (A) with appropriate initialization, then our iterates will be given by

$$\mathbf{w}_t = p_t(\mathbf{A})\mathbf{w}_0 = \textstyle\sum_{i=1}^d p_t(\lambda_i)\mathbf{u}_i\mathbf{u}_i^T\mathbf{w}_0.$$

We use this expression and properties of the Chebyshev polynomials to explicitly bound the convergence rate of the accelerated power method with Theorem 1 (analysis and proof in Appendix A).

Theorem 1. *Given a PSD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ with eigenvalues $1 \geq \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, running update (A) with $\lambda_2 \leq 2\sqrt{\beta} < \lambda_1$ results in estimates with worst-case error*

$$1 - \frac{(\mathbf{u}_1^T\mathbf{w}_t)^2}{\|\mathbf{w}_t\|^2} \leq \frac{4}{\left|\mathbf{w}_0^T\mathbf{u}_1\right|^2}\cdot\left(\frac{2\sqrt{\beta}}{\lambda_1+\sqrt{\lambda_1^2-4\beta}}\right)^{2t}.$$

We can derive the following corollary, which gives the iteration complexity to achieve ϵ error.

Corollary 2. *In the same setting as Theorem 1, update (A) with $\mathbf{w}_0 \in \mathbb{R}^d$ such that $\mathbf{u}_1^T\mathbf{w}_0 \neq 0$, for any $\epsilon \in (0,1)$, after $T = \mathcal{O}\left(\frac{\sqrt{\beta}}{\sqrt{\lambda_1^2-4\beta}}\cdot\log\frac{1}{\epsilon}\right)$ iterations achieves $1 - \frac{(\mathbf{u}_1^T\mathbf{w}_T)^2}{\|\mathbf{w}_T\|^2} \leq \epsilon$.*

Remark. Minimizing $\frac{\sqrt{\beta}}{\sqrt{\lambda_1^2-4\beta}}$ over $[\lambda_2^2/4, \lambda_1^2/4)$ tells us that $\beta = \lambda_2^2/4$ is the optimal setting.

In comparison to power iteration, this algorithm converges at an accelerated rate. In fact, as shown in Table 1, this momentum power method scheme (with the optimal assignment of $\beta = \lambda_2^2/4$) even matches the worst-case rate of the Lanczos method.

Extensions In Appendix B.1, we extend this momentum scheme to achieve acceleration in the setting where we want to recover multiple top eigenvectors of A, rather than just one. In Appendix B.2 we show that this momentum method is numerically stable, whereas the Lanczos method suffers from numerical instability [TBI97; GVL12]. Next, in Appendix B.3 we provide a heuristic for auto-tuning the momentum parameter, which is useful in practice. Finally, in Appendix B.4, we consider a larger orthogonal polynomial family, and we show that given some information about the tail spectrum of the matrix, we can obtain even faster convergence by using a 4-term inhomogeneous recurrence.

3 Stochastic PCA

Motivated by the results in the previous section, we study using momentum to accelerate PCA in the stochastic setting. We consider a streaming PCA setting, where we are given a series of i.i.d. samples, $\tilde{\mathbf{A}}_t$, such that

$$\mathbb{E}[\tilde{\mathbf{A}}_t] = \mathbf{A}, \ \max_t \|\tilde{\mathbf{A}}_t\| \leq r, \ \mathbb{E}[\|\tilde{\mathbf{A}}_t - \mathbf{A}\|^2] = \sigma^2. \quad (1)$$

In the sample covariance setting of Section 2, $\tilde{\mathbf{A}}_t$ can be obtained by selecting $\mathbf{x}_i\mathbf{x}_i^T$, where $\mathbf{x}_i$ is uni-

formly sampled from the dataset. One of the most popular streaming PCA algorithms is Oja's algorithm [Oja82], which repeatedly runs the update[2] $\mathbf{w}_{t+1} = (I + \eta \mathbf{A}_t)\mathbf{w}_t$. A natural way to try to accelerate Oja's algorithm is to directly add a momentum term, which leads to

$$\mathbf{w}_{t+1} = (I + \eta \tilde{\mathbf{A}}_t)\mathbf{w}_t - \beta \mathbf{w}_{t-1}. \tag{2}$$

In expectation, this stochastic recurrence behaves like the deterministic three-term recurrence (A), which can achieve acceleration with proper setting of β. However, we observe empirically that (2) usually does not give acceleration. In Figure 1(a), we see that while adding momentum does accelerate the initial convergence to the noise ball, it also increases the size of the noise ball—and decreasing the step size to compensate for this roughly cancels out the acceleration from momentum. This same counterintuitive phenomenon has independently been observed in Goh [Goh17] for stochastic optimization. The inability of momentum to accelerate Oja's algorithm is perhaps not surprising because the sampling complexity of Oja's algorithm is asymptotically optimal in terms of the eigen-gap [AZL16b].

In Section 4, we will characterize this connection between the noise ball size and momentum in more depth by presenting an exact expression for the variance of the iterates. Our analysis shows that when the sample variance is bounded, momentum can yield an accelerated convergence rate. In this section, we will present two methods that can be used to successfully control the variance: mini-batching and variance reduction. A summary of our methods and convergence rates is presented in Table 1.

3.1 Stochastic power method with momentum

In addition to adding momentum to Oja's algorithm, another natural way to try to accelerate stochastic PCA is to use the deterministic update (A) with random samples $\tilde{\mathbf{A}}_t$ rather than the exact matrix $\mathbf{A}$. Specifically, we analyze the stochastic recurrence

$$\mathbf{w}_{t+1} = \tilde{\mathbf{A}}_t \mathbf{w}_t - \beta \mathbf{w}_{t-1}, \tag{3}$$

where $\tilde{\mathbf{A}}_t$ is an i.i.d. unbiased random estimate of $\mathbf{A}$. We write this more explicitly as Algorithm 1.

When the variance is zero, the dynamics of this algorithm are the same as the dynamics of update (A), so it converges at the accelerated rate given in Theorem 1. Even if the variance is nonzero, but sufficiently small, we can still prove that Algorithm 1 converges at an accelerated rate.

[2]Here we consider a constant step size scheme, in which the iterate will converge to a noise ball. The size of the noise ball depends on the variance.

Algorithm 1 Mini-batch Power Method with Momentum (**Mini-batch Power+M**)

Require: Initial point $\mathbf{w}_0$, Number of Iterations T, Batch size s, Momentum parameter β
  $\mathbf{w}_{-1} \leftarrow \mathbf{0}$,
  for t = 0 **to** T - 1 **do**
    Generate a mini-batch of i.i.d. samples $B = \{\tilde{\mathbf{A}}_{t_1}, \cdots, \tilde{\mathbf{A}}_{t_s}\}$
    Update: $\mathbf{w}_{t+1} \leftarrow (\frac{1}{s}\sum_{i=1}^{s} \tilde{\mathbf{A}}_{t_i})\mathbf{w}_t - \beta \mathbf{w}_{t-1}$
    Normalization: $\mathbf{w}_t \leftarrow \mathbf{w}_t / \|\mathbf{w}_{t+1}\|, \mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1}/\|\mathbf{w}_{t+1}\|$.
  end for
  return $\mathbf{w}_T$

Theorem 3. *Suppose we run Algorithm 1 with* $2\sqrt{\beta} \in [\lambda_2, \lambda_1)$. *Let* $\Sigma = \mathbb{E}\left[(\mathbf{A}_t - \mathbf{A}) \otimes (\mathbf{A}_t - \mathbf{A})\right]$[3]. *Suppose that* $\|\mathbf{w}_0\| = 1$ *and* $\left|\mathbf{u}_1^T \mathbf{w}_0\right| \geq 1/2$. *For any* $\delta \in (0,1)$ *and* $\epsilon \in (0,1)$, *if*

$$\begin{aligned} T &= \frac{\sqrt{\beta}}{\sqrt{\lambda_1^2 - 4\beta}} \log\left(\frac{32}{\delta\epsilon}\right), \\ \|\Sigma\| &\leq \frac{(\lambda_1^2 - 4\beta)\delta\epsilon}{256\sqrt{dT}} = \frac{(\lambda_1^2 - 4\beta)^{3/2}\delta\epsilon}{256\sqrt{d}\sqrt{\beta}} \log^{-1}\left(\frac{32}{\delta\epsilon}\right), \end{aligned} \tag{4}$$

then with probability at least $1 - 2\delta$, *we have* $1 - (\mathbf{u}_1^T \mathbf{w}_T)^2 \leq \epsilon$.

When we compare this to the result of Theorem 1, we can see that as long as the variance $\|\Sigma\|$ is sufficiently small, the number of iterations we need to run in the online setting is the same as in the deterministic setting (up to a constant factor that depends on δ). In particular, this is faster than the power method without momentum in the deterministic setting. Of course, in order to get this accelerated rate, we need some way of getting samples that satisfy the variance condition of Theorem 3. Certain low-noise datasets might satisfy this condition, but this is not always the case. In the next two sections, we discuss methods of getting lower-variance samples.

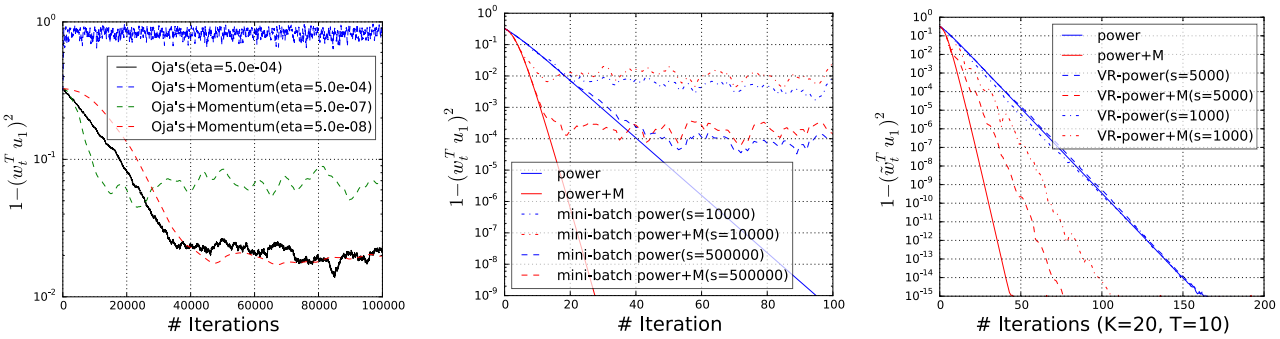
3.2 Controlling variance with mini-batches

In the online PCA setting, a natural way of getting lower-variance samples is to increase the *batch size* (parameter s) used by Algorithm 1. Using the following bound on the variance,

$$\begin{aligned} \|\Sigma\| &= \|\mathbb{E}\left[(\mathbf{A}_t - \mathbf{A}) \otimes (\mathbf{A}_t - \mathbf{A})\right]\| \\ &\leq \mathbb{E}\left[\|(\mathbf{A}_t - \mathbf{A}) \otimes (\mathbf{A}_t - \mathbf{A})\|\right] \\ &= \mathbb{E}\left[\|\mathbf{A}_t - \mathbf{A}\|^2\right] = \frac{\sigma^2}{s}, \end{aligned}$$

we can get an upper bound on the mini-batch size we will need in order to satisfy the variance condition in Theorem 3, which leads to the following corollary.

[3]$\otimes$ denotes the Kronecker product.



(a) Oja's algorithm with Momentum (b) Without Variance Reduction (c) With Variance Reduction

Figure 1: Different PCA algorithms on a synthetic dataset $\mathbf{X} \in \mathbb{R}^{10^6 \times 10}$ where the covariance matrix has eigengap $\Delta = 0.1$. Figure 1(a) shows the performance of Oja's algorithm with momentum. The momentum is set to the optimal $\beta = (1+\eta\lambda_2)^2/4$. Different dashed lines correspond to different step sizes (β changes correspondingly) for momentum methods. Figure 1(b) shows the performance of mini-batch power methods. Increasing the mini-batch size led to a smaller noise ball. Figure 1(c) shows the performance of VR power methods. The epoch length $T = 10$ was estimated according to (7) by setting $\delta = 1\%$ and $c = 1/16$. Stochastic methods report the average performance over 10 runs.

Corollary 4. *Suppose we run Algorithm 1 with* $2\sqrt{\beta} \in [\lambda_2, \lambda_1)$. *Assume that* $\|\mathbf{w}_0\| = 1$ *and* $\left|\mathbf{u}_1^T \mathbf{w}_0\right| \geq 1/2$. *For any* $\delta \in (0,1)$ *and* $\epsilon \in (0,1)$, *if*

$$T = \frac{\sqrt{\beta}}{\sqrt{\lambda_1^2 - 4\beta}} \log\left(\frac{32}{\delta\epsilon}\right),$$
$$s \geq \frac{256\sqrt{d}\sigma^2 T}{(\lambda_1^2 - 4\beta)\delta\epsilon} = \frac{256\sqrt{d}\sqrt{\beta}\sigma^2}{(\lambda_1^2 - 4\beta)^{3/2}\delta\epsilon} \log\left(\frac{32}{\delta\epsilon}\right),$$

then with probability at least $1 - 2\delta$, $1 - (\mathbf{u}_1^T \mathbf{w}_T)^2 \leq \epsilon$.

This means that no matter what the variance of the estimator is, we can still converge at the same rate as the deterministic setting as long as we can compute mini-batches of size s quickly. One practical way of doing this is by using many parallel workers: a mini-batch of size s can be computed in $\mathcal{O}(1)$ time by $\mathcal{O}(s)$ machines working in parallel. If we use a sufficiently large cluster, this means that Algorithm 1 converges in asymptotically less time than any non-momentum power method that uses the cluster for mini-batching, because we converge faster than even the deterministic non-momentum method.

One drawback of this approach is that the required variance decreases as a function of ϵ, so we will need to increase our mini-batch size as the desired error decreases. If we are running in parallel on a cluster of fixed size, this means that we will eventually exhaust the parallel resources of the cluster and be unable to compute the mini-batches in asymptotic $\mathcal{O}(1)$ time. As a result, we now seek methods to reduce the required batch size, and remove its dependence on ϵ.

3.3 Reducing batch size with variance reduction

Another way to generate low-variance samples is the *variance reduction* technique. This technique can be used if we have access to the target matrix $\mathbf{A}$ so that we can occasionally compute an exact matrix-vector product with $\mathbf{A}$. For example, in the offline setting, we can compute $\mathbf{A}\mathbf{w}$ by occasionally doing a complete pass over the data. In PCA, Shamir [Sha15] has applied the standard variance reduction technique that was used in stochastic convex optimization [JZ13], in which the stochastic term in the update is

$$\mathbf{A}\mathbf{w}_t + (\mathbf{A}_t - \mathbf{A})(\mathbf{w}_t - \tilde{\mathbf{w}}), \tag{5}$$

where $\tilde{\mathbf{w}}$ is the (normalized) anchor iterate, for which we know the exact value of $\mathbf{A}\tilde{\mathbf{w}}$. We propose a slightly different variance reduction scheme, where the stochastic term in the update is

$$\begin{aligned} &\left[\mathbf{A} + (\mathbf{A}_t - \mathbf{A})(I - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^T)\right]\mathbf{w}_t \\ &= \mathbf{A}\mathbf{w}_t + (\mathbf{A}_t - \mathbf{A})(I - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^T)\mathbf{w}_t. \end{aligned} \tag{6}$$

It is easy to verify that both (5) and (6) can be computed using only the samples $\mathbf{A}_t$ and the exact value of $\mathbf{A}\tilde{\mathbf{w}}$. In the PCA setting, (6) is more appropriate because progress is measured by the angle between $\mathbf{w}_t$ and $\mathbf{u}_1$, not the ℓ_2 distance as in the convex optimization problem setting: this makes (6) easier to analyze. In addition to being easier to analyze, our proposed update rule (6) produces updates that have generally lower variance because for all unit vectors $\mathbf{w}_t$ and $\tilde{\mathbf{w}}$, $\|\mathbf{w}_t - \tilde{\mathbf{w}}\| \geq \|(I - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^T)\mathbf{w}_t\|$. Using this update step

results in the variance-reduced power method with momentum in Algorithm 2. A number of methods use

Algorithm 2 VR Power Method with Momentum (**VR Power+M**)

Require: Initial point $\mathbf{w}_0$, Number of Iterations T, Batch size s, Momentum parameter β
  $\mathbf{w}_{-1} \leftarrow \mathbf{0}$
  for k = 1 **to** K **do**
    $\tilde{\mathbf{v}} \leftarrow \mathbf{A}\tilde{\mathbf{w}}_k$ (Usually there is no need to materialize $\mathbf{A}$ in practice).
    for t = 1 **to** T **do**
      Generate a mini-batch of i.i.d. samples $B = \{\tilde{\mathbf{A}}_{t_1}, \cdots, \tilde{\mathbf{A}}_{t_s}\}$
      Update: $\alpha \leftarrow \mathbf{w}_t^T \tilde{\mathbf{w}}_k$,
      $\mathbf{w}_{t+1} \leftarrow \frac{1}{s}\sum_{i=1}^{s} \tilde{\mathbf{A}}_{t_i}(\mathbf{w}_t - \alpha\tilde{\mathbf{w}}_k) + \alpha\tilde{\mathbf{v}} - \beta\mathbf{w}_{t-1}$
      Normalization: $\mathbf{w}_t \leftarrow \mathbf{w}_t/\|\mathbf{w}_{t+1}\|$,
      $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1}/\|\mathbf{w}_{t+1}\|$.
    end for
    $\tilde{\mathbf{w}}_{k+1} \leftarrow \mathbf{w}_T$.
  end for
  return $\mathbf{w}_K$

this kind of SVRG-style variance reduction technique, which converges at a linear rate and is not limited by a noise ball. Our method improves upon that by achieving the *accelerated rate* throughout, and only using a mini-batch size that is constant with respect to ϵ.

Theorem 5. *Suppose we run Algorithm 2 with* $2\sqrt{\beta} \in [\lambda_2, \lambda_1)$ *and a initial unit vector* $\mathbf{w}_0$ *such that* $1 - (\mathbf{u}_1^T\mathbf{w}_0)^2 \leq \frac{1}{2}$. *For any* $\delta, \epsilon \in (0,1)$, *if*

$$T = \frac{\sqrt{\beta}}{\sqrt{\lambda_1^2 - 4\beta}} \log\left(\frac{1}{c\delta}\right), s \geq \frac{32\sqrt{d}\sqrt{\beta}\sigma^2}{c(\lambda_1^2 - 4\beta)\delta} \log\left(\frac{1}{c\delta}\right), \tag{7}$$

then after $K = \mathcal{O}(\log(1/\epsilon))$ *epochs, with probability at least* $1 - \log\left(\frac{1}{\epsilon}\right)\delta$, *we have* $1 - (\mathbf{u}_1^T\tilde{\mathbf{w}}_K)^2 \leq \epsilon$, *where* $c \in (0, 1/16)$ *is a numerical constant.*

By comparing to the results of Theorem 1 and Theorem 5, we notice that the VR power method with momentum achieves the same convergence rate, in terms of the total number of iterations we need to run, as the deterministic setting. In contrast to the non-variance-reduced setting, the mini-batch size we need to use does not depend on the desired error ϵ, which allows us to use a fixed mini-batch size throughout the execution of the algorithm. This means that we can use Algorithm 2 together with a parallel mini-batch-computing cluster of fixed size to compute solutions of arbitrary accuracy at a rate faster than any non-momentum power method could achieve. As shown in Table 1, in terms of number of iterations, the momentum methods achieve accelerated linear convergence with proper mini-batching (our results there follow Corollary 4 and Theorem 5, using the optimal momentum $\beta = \lambda_2^2/4$.).

3.4 Experiments

We first use synthetic experiments (details in Appendix E) to illustrate how the variance affects the momentum methods. Figure 1(b) shows that the stochastic power method maintains the same linear convergence as the deterministic power method before hitting the noise ball. Therefore, the momentum method can accelerate the convergence before hitting the noise ball. Figure 1(c) shows that the variance-reduced power method indeed can achieve an accelerated linear convergence with a much smaller batch size on this same synthetic dataset.

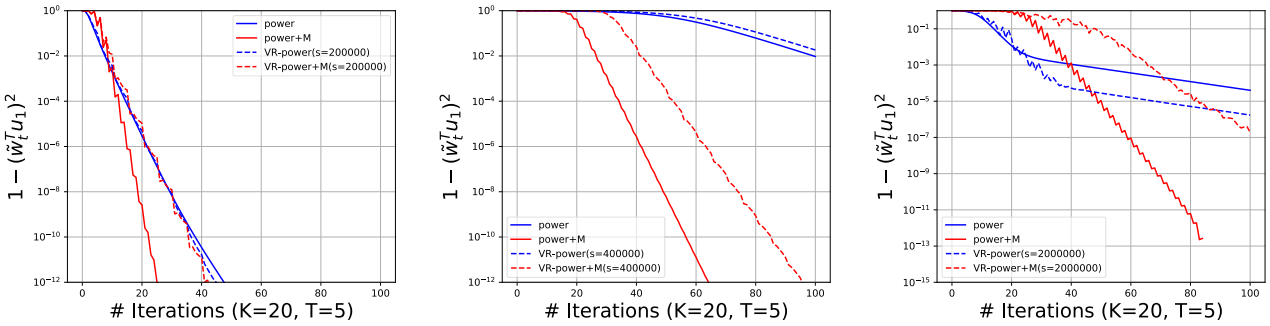
We additionally demonstrate that our variance-reduced method results in accelerated convergence for finding the first eigenvector of the adjacency matrices of several networks. For these experiments, We use the arXiv ASTRO-PH collaboration network [LKF07], the arXiv High-energy physics citation network [LKF05], and the Google web graph [Les+09]. This is the eigenvector centrality task, which is often used in network analysis [Bra05; Bon72]. In the eigenvector centrality task, components of the top eigenvector that are large correspond to nodes that are central to the network. The eigenvector centrality task is closely related the PageRank metric. Figure 2 shows that the variance-reduced power method achieves an accelerated linear convergence on this task. In these experiments, momentum results in larger improvements in convergence when the eigen-gap is small.

4 Convergence analysis

In this section, we sketch the proofs of Theorems 3 and 5. The main idea is to tightly bound the variance of the iterates with properties of the Chebyshev polynomials. Both with mini-batches (Algorithm 1) and variance reduction (Algorithm 2), the dynamics of the stochastic power method with momentum from (3) can be written as $\mathbf{w}_t = \mathbf{F}_t\mathbf{w}_0/\|\mathbf{F}_t\mathbf{w}_0\|$ where $\{\mathbf{F}_t\}$ is a sequence of stochastic matrices in $\mathbb{R}^{d\times d}$ satisfying

$$\mathbf{F}_{t+1} = \mathbf{A}_{t+1}\mathbf{F}_t - \beta\mathbf{F}_{t-1},\ \mathbf{F}_0 = I,\ \mathbf{F}_{-1} = \mathbf{0}. \tag{8}$$

The random matrix $\mathbf{A}_t \in \mathbb{R}^{d\times d}$ has different forms in Algorithm 1 and Algorithm 2. However, in both algorithms, $\mathbf{A}_t$ will be i.i.d. and satisfy $\mathbb{E}[\mathbf{A}_t] = \mathbf{A}$. In fact, this recurrence (8) is general enough to be applied in many other problems, including least-squares regression and the randomized Kaczmarz algorithm [SV09; GR15], as well as some non-convex matrix problems [DSRO15] such as matrix completion [JNS13], phase retrieval [CLS15] and subspace tracking [BNR10]. Since $\mathbf{F}_t$ obeys a linear recurrence, its second moment also follows a linear recurrence (in fact, all its moments do). We decompose



(a) ASTRO-PH collaboration (b) High Energy Physics paper citation (c) Google web graph

Figure 2: Variance-reduced and full-pass PCA algorithms on large network datasets. The dimensions are (a) 18,772, (b) 34,546, and (c) 875,713. The relative eigen-gaps are (a) 0.20, (b) 0.047, and (c) 0.027. Our momentum-based methods result in larger improvements in the convergence rate when the relative eigen-gap is smaller.

this recurrence using Chebyshev polynomials to get a tight bound on the covariance of $\mathbf{F}_t$, which is shown in Lemma 6. This bound is exact in the scalar case.

Lemma 6. *Suppose* $\lambda_1^2 \geq 4\beta$ *and* $\Sigma = \mathbb{E}\left[(\mathbf{A}_t - \mathbf{A}) \otimes (\mathbf{A}_t - \mathbf{A})\right]$. *The norm of the covariance of the matrix* $\mathbf{F}_t$ *is bounded by*

$$\begin{aligned}&\left\|\mathbb{E}\left[\mathbf{F}_t \otimes \mathbf{F}_t\right] - \mathbb{E}\left[\mathbf{F}_t\right] \otimes \mathbb{E}\left[\mathbf{F}_t\right]\right\| \\ &\leq \sum_{n=1}^{t} \|\Sigma\|^n \beta^{t-n} \sum_{\mathbf{k} \in S_{t-n}^{n+1}} \prod_{i=1}^{n+1} U_{k_i}^2\left(\frac{\lambda_1}{2\sqrt{\beta}}\right),\end{aligned}$$

where $U_k(\cdot)$ *is the Chebyshev polynomial of the second kind, and* S_m^n *denotes the set of vectors in* $\mathbb{N}^n$ *with entries that sum to* m, *i.e.*

$$S_m^n = \left\{\mathbf{k} = (k_1, \cdots, k_n) \in \mathbb{N}^n \mid \textstyle\sum_{i=1}^n k_i = m\right\}.$$

For the mini-batch power method without variance reduction (Algorithm 1), the goal is to bound $1 - (\mathbf{u}_1^T \mathbf{w}_t)^2$, which is equivalent to bounding $\sum_{i=2}^d (\mathbf{u}_i^T \mathbf{F}_t \mathbf{w}_0)^2 / (\mathbf{u}_1^T \mathbf{F}_t \mathbf{w}_0)^2$. We use Lemma 6 to get a variance bound for the denominator of this expression, which is

$$\mathbf{Var}\left[\mathbf{u}_1^T \mathbf{F}_t \mathbf{w}_0\right] \leq p_t^2(\lambda_1; \beta) \cdot \frac{8\|\Sigma\| t}{\lambda_1^2 - 4\beta}. \tag{9}$$

With this variance bound and Chebyshev's inequality we get a probabilistic lower bound for $|\mathbf{u}_1^T \mathbf{F}_t \mathbf{w}_0|$. Lemma 6 can also be used to get an upper bound for the numerator, which is

$$\begin{aligned}&\mathbb{E}\left[\sum_{i=2}^{d} (\mathbf{u}_i^T \mathbf{F}_t \mathbf{w}_0)^2\right] \\ &\leq p_t^2(\lambda_1; \beta) \cdot \left(\frac{8\sqrt{d}\|\Sigma\| t}{(\lambda_1^2 - 4\beta)} + \frac{p_t^2(2\sqrt{\beta}; \beta)}{p_t^2(\lambda_1; \beta)}\right)\end{aligned} \tag{10}$$

By Markov's inequality we can get a probabilistic upper bound for $\sum_{i=2}^d (\mathbf{u}_i^T \mathbf{F}_t \mathbf{w}_0)^2$. The result in Theorem 3 now follows by a union bound. The details of the proof appear in Appendix C.1.

Next, we consider the case with variance reduction (Algorithm 2). The analysis contains two steps. The first step is to show a geometric contraction for a single epoch, i.e.

$$1 - (\mathbf{u}_1^T \mathbf{w}_T)^2 \leq \rho \cdot \left(1 - (\mathbf{u}_1^T \mathbf{w}_0)^2\right), \tag{11}$$

with probability at least $1 - \delta$, where $\rho < 1$ is a numerical constant. Afterwards, the second step is to get the final ϵ accuracy of the solution, which trivially requires $\mathcal{O}\left(\log(1/\epsilon)\right)$ epochs. Thus, the analysis boils down to analyzing a single epoch. Notice that in this setting,

$$\mathbf{A}_{t+1} = \mathbf{A} + \left(\tfrac{1}{s}\textstyle\sum_{i=1}^s \tilde{\mathbf{A}}_{t_i} - \mathbf{A}\right)(I - \mathbf{w}_0 \mathbf{w}_0^T), \tag{12}$$

and again $\mathbf{w}_t = \mathbf{F}_t \mathbf{w}_0 / \|\mathbf{F}_t \mathbf{w}_0\|$. Using similar techniques to the mini-batch power method setting, we can prove a variant of Lemma 6 specialized to (12).

Lemma 7. *Suppose* $\lambda_1^2 \geq 4\beta$. *Let* $\mathbf{w}_0 \in \mathbb{R}^d$ *be a unit vector,* $\theta = 1 - (\mathbf{u}_1^T \mathbf{w}_0)^2$, *and*

$$\Sigma = \mathbb{E}\left[\left(\tfrac{1}{s}\textstyle\sum_{i=1}^s \tilde{\mathbf{A}}_{t_i} - \mathbf{A}\right) \otimes \left(\tfrac{1}{s}\textstyle\sum_{i=1}^s \tilde{\mathbf{A}}_{t_i} - \mathbf{A}\right)\right].$$

Then, the norm of the covariance will be bounded by

$$\begin{aligned}&\left\|\mathbb{E}\left[\mathbf{F}_t \mathbf{w}_0 \otimes \mathbf{F}_t \mathbf{w}_0\right] - \mathbb{E}\left[\mathbf{F}_t \mathbf{w}_0\right] \otimes \mathbb{E}\left[\mathbf{F}_t \mathbf{w}_0\right]\right\| \\ &\leq 4\theta \cdot \sum_{n=1}^{t} \|\Sigma\|^n \beta^{t-n} \sum_{\mathbf{k} \in S_{t-n}^{n+1}} \prod_{i=1}^{n+1} U_{k_i}^2\left(\frac{\lambda_1}{2\sqrt{\beta}}\right).\end{aligned}$$

Comparing to the result in Lemma 6, this lemma shows that the covariance is also controlled by the angle between $\mathbf{u}_1$ and $\mathbf{w}_0$ which is the anchor point in each

epoch. Since the anchor point $\tilde{\mathbf{w}}_k$ is approaching $\mathbf{u}_1$, the norm of the covariance is shrinking across epochs—this allows us to prove (11). From here, the proof of Theorem 5 is similar to non-VR case, and the details are in Appendix C.2.

Remark 1. As summarized in Table 1, we achieved the optimal, accelerated iteration complexity, by allowing computation to be wide (massively parallel) instead of deep (many sequential steps). This ability for massive parallelization comes at a cost of an extra $\sqrt{d}$ factor in total computation. It is an interesting open question whether this extra computation is fundamental and unavoidable for massively parallel methods.

Remark 2. Note that Theorems 3 and 5 only state the local convergence complexity. However, it is worth mentioning that there is no technical challenge in obtaining the state of warm initialization in the theorems. For example, starting from random uniform initialization, it will take $\mathcal{O}(d/\Delta^2)$ iterations for Alecton [DSRO15] to get into a constant error ball. This is independent of ϵ and negligible comparing to the local complexity.

5 Related work

PCA A recent spike in research activity has focused on improving a number of computational and statistical aspects of PCA, including tighter sample complexity analysis [Jai+16; Li+16], global convergence [DSRO15; AZL16b; BDF13], memory efficiency [MCJ13] and doing online regret analysis [Bou+15]. Some work has also focused on tightening the analysis of power iteration and Krylov methods to provide gap-independent results using polynomial-based analysis techniques [MM15]. However, that work does not consider the stochastic setting. Some works that study Oja's algorithm [Oja82] or stochastic power methods in the stochastic setting focus on the analysis of a gap-free convergence rate for the *distinct PCA formulation of maximizing explained variance* (as opposed to recovering the strongest direction) [Sha16; AZL16b]. Others provide better dependence on the dimension of the problem [Jai+16]. Garber et al. [Gar+16] and Allen-Zhu and Li [AZL16c] use faster linear system solvers to speed up PCA algorithms such that the convergence rate has the square root dependence on the eigengap in the offline setting. However their methods require solving a series of linear systems, which is not trivially parallelizable. Also none of these results give a convergence analysis that is asymptotically tight in terms of variance. Another line of work has focused on variance control for PCA in the stochastic setting [Sha15] to get a different kind of acceleration. Since this is an independent source of improvement, these methods can be further accelerated using our momentum scheme. In addition [XLS15] study the stochastic power methods for kernel PCA in the random feature space, where our momentum scheme is also applicable.

Stochastic acceleration Momentum is a common acceleration technique in convex optimization [Pol64; Nes83], and has been widely adopted as the de-facto optimization method for non-convex objectives in deep learning [Sut+13]. Provably accelerated stochastic methods have previously been found for convex problems [Cot+11; Jai+17]. However, similar results for non-convex problems remain elusive, despite empirical evidence that momentum results in acceleration for some non-convex problems [Sut+13; KB14].

Orthogonal Polynomials The Chebyshev polynomial family is a sequence of orthogonal polynomials [Chi11] that has been used for analyzing accelerated methods. For example, Chebyshev polynomials have been studied to accelerate the solvers of linear systems [GV61; GVL12] and to accelerate convex optimization [SdB16]. Trefethen and Bau III [TBI97] use Chebyshev polynomials to show that the Lanczos method is quadratically faster than the standard power iteration. The Lanczos method is conventionally considered the accelerated version of power method with momentum [HP14].

6 Conclusion

This paper introduced a very simple accelerated PCA algorithm that works in the stochastic setting. As a foundation, we presented the power method with momentum, an accelerated scheme in the deterministic setting. We proved that the power method with momentum obtains quadratic acceleration like in the convex optimization setting. Then, for the stochastic setting, we introduced and analyzed the stochastic power method with momentum. By leveraging the Chebyshev polynomials, we derived a convergence rate that is asymptotically tight in terms of the variance. Using a tight variance analysis, we demonstrated how the momentum scheme behaves in a stochastic system, which can lead to a better understanding of how momentum interacts with variance in stochastic optimization problems [Goh17]. Specifically, with mini-batching, the stochastic power method with momentum can achieve accelerated convergence to the noise ball. Alternatively, using variance reduction, accelerated convergence at a linear rate can be achieved with a much smaller batch size.

Acknowledgments

We thank Aaron Sidford for helpful discussion and feedback on this work.

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) SIMPLEX program under No. N66001-15-C-4043, D3M program under No. FA8750-17-2-0095, the National Science Foundation (NSF) CAREER Award under No. IIS- 1353606, the Office of Naval Research (ONR) under awards No. N000141210041 and No. N000141310129, a Sloan Research Fellowship, the Moore Foundation, an Okawa Research Grant, Toshiba, and Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, ONR, or the U.S. government.

References

[AZL16a] Zeyuan Allen-Zhu and Yuanzhi Li. "Doubly Accelerated Methods for Faster CCA and Generalized Eigendecomposition". In: *arXiv preprint arXiv:1607.06017* (2016).

[AZL16b] Zeyuan Allen-Zhu and Yuanzhi Li. "First Efficient Convergence for Streaming k-PCA: a Global, Gap-Free, and Near-Optimal Rate". In: *arXiv preprint arXiv:1607.07837* (2016).

[AZL16c] Zeyuan Allen-Zhu and Yuanzhi Li. "LazySVD: Even faster SVD decomposition yet without agonizing pain". In: *Advances in Neural Information Processing Systems*. 2016, pp. 974–982.

[BDF13] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. "The fast convergence of incremental pca". In: *Advances in Neural Information Processing Systems*. 2013, pp. 3174–3182.

[BNR10] Laura Balzano, Robert Nowak, and Benjamin Recht. "Online identification and tracking of subspaces from highly incomplete information". In: *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE. 2010, pp. 704–711.

[Bon72] Phillip Bonacich. "Factoring and weighting approaches to status scores and clique identification". In: *Journal of Mathematical Sociology* 2.1 (1972), pp. 113–120.

[Bou+15] Christos Boutsidis et al. "Online principal components analysis". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2015, pp. 887–901.

[Bra05] Ulrik Brandes. *Network analysis: methodological foundations*. Vol. 3418. Springer Science & Business Media, 2005.

[Chi11] Theodore S Chihara. *An introduction to orthogonal polynomials*. Courier Corporation, 2011.

[CLS15] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. "Phase retrieval via Wirtinger flow: Theory and algorithms". In: *IEEE Transactions on Information Theory* 61.4 (2015), pp. 1985–2007.

[Cot+11] Andrew Cotter et al. "Better mini-batch algorithms via accelerated gradient methods". In: *Advances in neural information processing systems*. 2011, pp. 1647–1655.

[DSRO15] Christopher De Sa, Christopher Ré, and Kunle Olukotun. "Global Convergence of Stochastic Gradient Descent for Some Non-convex Matrix Problems". In: *International Conference on Machine Learning*. 2015, pp. 2332–2341.

[Gar+16] Dan Garber et al. "Faster eigenvector computation via shift-and-invert preconditioning". In: *International Conference on Machine Learning*. 2016, pp. 2626–2634.

[Goh17] Gabriel Goh. "Why Momentum Really Works". In: *Distill* (2017). DOI: 10.23915/distill.00006. URL: http://distill.pub/2017/momentum.

[GR15] Robert M Gower and Peter Richtárik. "Randomized iterative methods for linear systems". In: *SIAM Journal on Matrix Analysis and Applications* 36.4 (2015), pp. 1660–1690.

[GV61] Gene H Golub and Richard S Varga. "Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods". In: *Numerische Mathematik* 3.1 (1961), pp. 147–156.

[GVL12] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[Hot33] Harold Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6 (1933), p. 417.

[HP14] Moritz Hardt and Eric Price. "The noisy power method: A meta algorithm with applications". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2861–2869.

[Jai+16] Prateek Jain et al. "Matching Matrix Bernstein with Little Memory: Near-Optimal Finite Sample Guarantees for Oja's Algorithm". In: *arXiv preprint arXiv:1602.06929* (2016).

[Jai+17] Prateek Jain et al. "Accelerating Stochastic Gradient Descent". In: *arXiv preprint arXiv:1704.08227* (2017).

[JNS13] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. "Low-rank matrix completion using alternating minimization". In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013, pp. 665–674.

[Jol02] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[JZ13] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in Neural Information Processing Systems*. 2013, pp. 315–323.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[Les+09] Jure Leskovec et al. "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters". In: *Internet Mathematics* 6.1 (2009), pp. 29–123.

[Li+16] Chris J Li et al. "Near-Optimal Stochastic Approximation for Online Principal Component Estimation". In: *arXiv preprint arXiv:1603.05305* (2016).

[LKF05] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graphs over time: densification laws, shrinking diameters and possible explanations". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 177–187.

[LKF07] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 2.

[MCJ13] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. "Memory limited, streaming PCA". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2886–2894.

[MM15] Cameron Musco and Christopher Musco. "Randomized block krylov methods for stronger and faster approximate singular value decomposition". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1396–1404.

[Nes83] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate O (1/k2)". In: *Soviet Mathematics Doklady*. Vol. 27. 2. 1983, pp. 372–376.

[Oja82] Erkki Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of mathematical biology* 15.3 (1982), pp. 267–273.

[Pol64] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[SdB16] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. "Regularized Nonlinear Acceleration". In: *Advances In Neural Information Processing Systems*. 2016, pp. 712–720.

[Sha15] Ohad Shamir. "A stochastic PCA and SVD algorithm with an exponential convergence rate". In: *Proc. of the 32st Int. Conf. Machine Learning (ICML 2015)*. 2015, pp. 144–152.

[Sha16] Ohad Shamir. "Convergence of stochastic gradient descent for PCA". In: *International Conference on Machine Learning*. 2016, pp. 257–265.

[Sut+13] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th international conference on machine learning (ICML-13)*. 2013, pp. 1139–1147.

[SV09] Thomas Strohmer and Roman Vershynin. "A randomized Kaczmarz algorithm with exponential convergence". In: *Journal of Fourier Analysis and Applications* 15.2 (2009), pp. 262–278.

[TBI97] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*. Vol. 50. Siam, 1997.

[XLS15] Bo Xie, Yingyu Liang, and Le Song. "Scale up nonlinear component analysis with doubly stochastic gradients". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2341–2349.