A Stochastic PCA Algorithm with an Exponential Convergence Rate

Ohad Shamir
Weizmann Institute of Science
ohad.shamir@weizmann.ac.il

Abstract

We describe and analyze a simple algorithm for principal component analysis, SVR-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 the non-convex PCA problem, using a very different analysis.

Principal Component Analysis (PCA) is one of the most common tools for unsupervised data analysis and preprocessing. In its simplest possible form[1], we are given a dataset of n instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$, and are interested in finding a unit vector $\mathbf{v}$ which minimizes

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

The optimal solution is given by the largest eigenvector $\mathbf{w} = \mathbf{v}_1$ of the covariance matrix $\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top$.

When the data size n and the dimension d are modest, this problem can be solved exactly by computing the $d \times d$ covariance matrix, and performing an eigendecomposition. However, the required runtime is $\mathcal{O}(nd^2 + d^3)$, which is prohibitive in large-scale applications. Moreover, even storing a $d \times d$ matrix in memory can be impossible when d is very large. One possible approach is using iterative methods such as power iteration or the Lanczos method [4]. If the covariance matrix has 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 $p = 1$ for power iterations, and $p = 1/2$ for the Lanczos method). However, each iteration involves multiplying one or more vectors by the covariance matrix, which requires $\mathcal{O}(nd)$ time (by passing through the entire data). Thus, the total runtime is $\mathcal{O}\left(d\frac{n\log(1/\epsilon)}{\lambda^p}\right)$. When λ is small, this runtime 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. [7, 10, 11] and more recently, [1, 9, 2]), which utilize the structure of the covariance method. 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 using only $\mathbf{x}_i$. In general, the runtime of each iteration is just $O(d)$. On the flip side, due to their stochastic and incremental nature, the convergence rate (when known) is quite slow, and the number of required iterations can be prohibitive when a high-accuracy solution is required.

[1] Our approach can be extended to more general cases.

In this work, we propose a new stochastic PCA algorithm, denoted as SVR-PCA [2], which under suitable assumptions, has provable runtime of

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

where λ is the eigengap parameter. This algorithm combines the advantages of the previously discussed approaches, while avoiding some of their 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 dn up to logarithmic factors: Proportional to the time required to perform a single scan of the data.

SVR-PCA builds on a recently-introduced technique for stochastic gradient variance reduction, which has been previously studied (see [5] as well as [8, 6]). 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 (at least locally), and often assume an unconstrained domain. In contrast, our algorithm attempts to minimize the function in Eq. (1), which is nowhere convex, let alone strongly convex (in fact, it is *concave* everywhere), and over a non-convex domain. 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.

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. We note that the runtime of each iteration is $\mathcal{O}(dn)$, and the runtime of each epoch, besides the iterations, is dominated by computing $\tilde{\mathbf{u}}$ in $\mathcal{O}(dn)$ time.

Algorithm 1 SVR-PCA

Parameters: Step size η, epoch length m
Input: Data set $\{\mathbf{x}_i\}_{i=1}^n$, Initial unit vector $\tilde{\mathbf{w}}_0$
for $s = 1, 2, \ldots$ **do**
  $\tilde{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top \tilde{\mathbf{w}}_{s-1}$
  $\mathbf{w}_0 = \tilde{\mathbf{w}}_{s-1}$
  for $t = 1, 2, \ldots, m$ **do**
    Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $\mathbf{w}'_t = \mathbf{w}_{t-1} + \eta\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top(\mathbf{w}_{t-1} - \tilde{\mathbf{w}}_{s-1}) + \tilde{\mathbf{u}}\right)$
    $\mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}'_t$
  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 [10], on which our algorithm is based. In our setting, this rule consists of repeatedly sampling a data point $\mathbf{x}_{i_t}$ at random, and performing the update $\mathbf{w}'_t = \mathbf{w}_{t-1} + \eta_t\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top\mathbf{w}_{t-1}$, $\mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}_t$. Letting $A = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$, this is equivalent to

$$\mathbf{w}'_t = (I + \eta_t A)\mathbf{w}_{t-1} + \eta_t\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)\mathbf{w}_{t-1} \ , \ \mathbf{w}_t = \frac{1}{\|\mathbf{w}'_t\|}\mathbf{w}_t. \tag{2}$$

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, [3] 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

[2] SVR stands for "Stochastic variance-reduced".

for convex optimization [5], 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 of our SVR-PCA algorithm as

$$\mathbf{w}_t' = (I + \eta A)\mathbf{w}_{t-1} + \eta\left(\mathbf{x}_{i_t}\mathbf{x}_{i_t}^\top - A\right)(\mathbf{w}_{t-1} - \tilde{\mathbf{u}}) \quad , \quad \mathbf{w}_t = \frac{1}{\|\mathbf{w}_t'\|}\mathbf{w}_t. \tag{3}$$

Comparing Eq. (3) to Eq. (2), 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 $\|\mathbf{w}_{t-1} - \tilde{\mathbf{u}}\|$. As we get closer to the optimal solution, we expect that both $\tilde{\mathbf{u}}$ 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.

A formal analysis of the algorithm appears in the following theorem, whose proof is omitted in this extended abstract:

Theorem 1. *Let $\mathbf{v}_1$ be an eigenvector of $A = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$ corresponding to the largest singular value. Suppose that the $\max_i \|\mathbf{x}_i\|^2$ (and hence the spectral norm of A) is at most r; That A has singular values $s_1 > s_2 \geq \ldots \geq s_d$, where $s_1 - s_2 = \lambda$ for some $\lambda > 0$; And that $\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle \geq \frac{1}{\sqrt{2}}$.*

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

$$\eta \leq c_1\delta^2\lambda \quad , \quad m \geq \frac{c_2\log(2/\delta)}{\eta\lambda} \quad , \quad m\eta^2 + \sqrt{m\eta^2\log(2/\delta)} \leq c_3, \tag{4}$$

(where c_1, c_2, c_3 designates positive numerical constants), and for $T = \left\lceil \frac{\log(1/\epsilon)}{\log(2/\delta)} \right\rceil$ epochs, then with probability at least $1 - 2\log(1/\epsilon)\delta$, it holds that $\langle \tilde{\mathbf{w}}_T, \mathbf{v}_1\rangle^2 \geq 1 - \epsilon$.

It is easy to verify that for any fixed δ, Eq. (4) 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 $1 - 2\log(1/\epsilon)\delta$. Since each iteration requires $\mathcal{O}(d)$ time to implement, and each epoch requires an additional $\mathcal{O}(dn)$ time to compute $\tilde{\mathbf{u}}$, we get a total runtime of

$$\mathcal{O}\left(d\left(n + \left(\frac{r}{\lambda}\right)^2\right)\log\left(\frac{1}{\epsilon}\right)\right), \tag{5}$$

establishing an exponential convergence rate. If $\lambda/r \geq \Omega(1/\sqrt{n})$, then the runtime is $\mathcal{O}(dn\log(1/\epsilon))$ – 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 $\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle \geq \frac{1}{\sqrt{2}}$. 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 $|\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle| \leq \mathcal{O}(1/\sqrt{d})$ with high probability. Thus, the theorem should be interpreted as analyzing the 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, since the variance-reduction technique only kicks in once we are relatively close to the optimum, there is no reason not to use some different stochastic algorithm with finite-time analysis, such as Oja's algorithm (e.g. [3]) to get to this constant accuracy, from which point our algorithm and analysis takes over. In any case, we note that some assumption on $\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle$ being bounded away from 0 must hold: If we initialize the algorithm with $\tilde{\mathbf{w}}_0$ such that $\langle \tilde{\mathbf{w}}_0, \mathbf{v}_1\rangle = 0$, then the algorithm may fail to converge (a similar property holds for power iterations, and follows from the non-convex nature of the optimization problem).

Finally, we note that 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 [5]. 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 to $1/\lambda$. However, we don't know if this is true, or whether the $1/\lambda^2$ factor is necessary in our setting.

We now turn to present a preliminary experiment, which demonstrates the performance of the SVR-PCA algorithm (additional experiments were performed, but are omitted in this extended abstract). We constructed several synthetic datasets, , where 50,000 examples are drawn i.i.d. from a Gaussian distribution in $\mathbb{R}^{1000}$, with zero mean and covariance matrix $I + \lambda \mathbf{e}_i \mathbf{e}_i^\top$. The spectrum of this matrix equals $(1+\lambda, 1, 1, \ldots, 1)$, corresponding to an eigengap of roughly λ (in practice, due to finite sample effects, the eigengap of the data covariance matrix is slightly different). Each dataset corresponds to a different value of λ. We used a fixed choice of parameters, where $m = n$ and $\eta = 0.05/\sqrt{n}$. This choice of m 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 $1/(r\sqrt{n})$ in the regime where m should be on the order of n. For comparison, we also implemented Oja's algorithm, using several different step sizes. All algorithms were initialized from the same same random vector, chosen uniformly at random from the unit ball. Again, 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, SVR-PCA converges much faster than all versions of Oja's algorithm, on which it is based, 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.

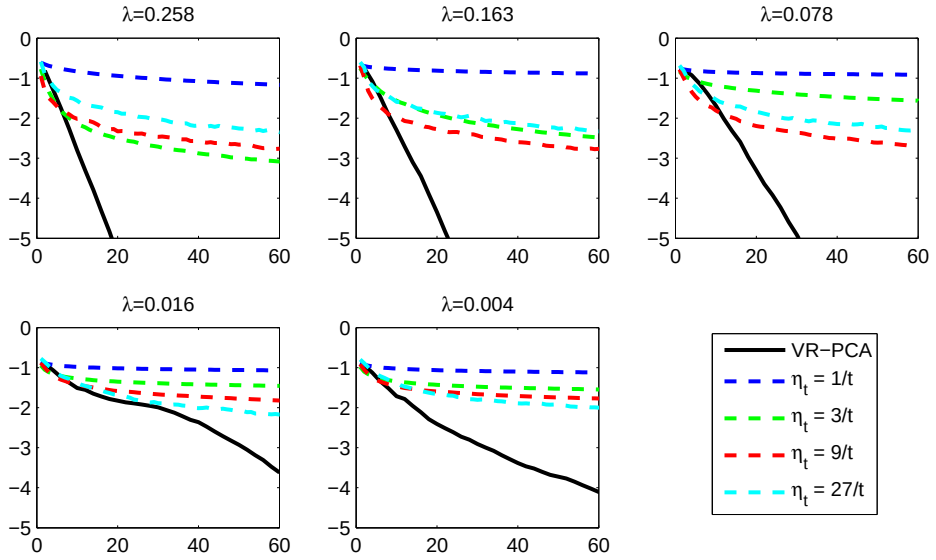

Figure 1: Results for synthetic data. Each plot represents a run on a single dataset with eigengap λ, and compares the performance of SVR-PCA to Oja's algorithm with different step sizes η_t. In each plot, the x-axis represents the number of data accesses divided by the data size (assuming $2n$ accesses per epoch for SVR-PCA, to perform n iterations plus computing $\tilde{\mathbf{u}}$), and the y-axis equals $\log_{10}\left(1 - \frac{\mathbf{w}^\top A \mathbf{w}}{\mathbf{v}_1^\top A \mathbf{v}_1}\right)$, where $\mathbf{w}$ is the vector obtained so far.

References

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

[2] R. Arora, A. Cotter, and N. Srebro. Stochastic optimization of pca with capped msg. In *NIPS*, 2013.

[3] A. Balsubramani, S. Dasgupta, and Y. Freund. The fast convergence of incremental pca. In *NIPS*, 2013.

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

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

[6] J. Konecný and P. Richtárik. Semi-stochastic gradient descent methods. *CoRR*, abs/1312.1666, 2013.

[7] T.P. Krasulina. 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.

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

[9] I. Mitliagkas, C. Caramanis, and P. Jain. Memory limited, streaming pca. In *NIPS*.

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

[11] E. Oja and J. Karhunen. 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.