Final Project – Mathematical Methods in Data Science and Signal Processing

Adi Green (313472417)

A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate by Ohad Shamir [1]

Summary

This paper [1] describes and analyzes an algorithm for performing principal component analysis (PCA) and singular value decomposition (SVD) in a stochastic (randomized) manner, which uses computationally cheap stochastic iterations, yet converges exponentially fast to the optimal solution.

Principal Component Analysis (PCA) is a widely used tool in machine learning and statistics for unsupervised data analysis and pre-processing. It involves finding a k-dimensional subspace in a data matrix $X \in \mathbb{R}^{dxn}$, where the projection of the data has the highest variance. PCA has various applications such as reducing dimensionality, data compression, and data visualization.

We consider the following optimization problem: given a matrix $X \in \mathbb{R}^{dxn}$, and $x_1, \dots, x_n$ being the columns of X:

$$\min_{W \in \mathbb{R}^{dxn}: W^T W = I} - W^T \left(\sum_{i=1}^{n} x_i x_i^T \right) W \tag{1}$$

The solution is the top k left singular values (where $k \ll d$) of the covariance matrix $\sum_{i=1}^{n} x_i x_i^T$. The paper focuses mostly on finding the top eigenvector v$_1$ (k=1) which reduces the above to:

$$\min_{w: \|w\|_2 = 1} - \boldsymbol{w}^T \left(\sum_{i=1}^{n} x_i x_i^T \right) \boldsymbol{w} \tag{2}$$

Deterministic algorithms for calculating a unit vector that is ε-far from v$_1$ are expensive for large datasets, while stochastic algorithms have slow convergence rates for high-accuracy solutions. The new VR-PCA stochastic algorithm combines the advantages of both approaches while avoiding their main disadvantages and is based on a recently introduced technique for stochastic gradient variance reduction. Previous works on this technique are inapplicable to VR-PCA since it attempts to minimize a non-convex and concave function, requiring a new analysis.

VR-PCA, which on bounded data and under suitable assumptions has provable runtime of $O\left(d_s\left(n + \frac{1}{\lambda^2}\right) log(\varepsilon)\right)$ – where d_s is the average sparsity in each x_i, and λ is the eigengap between the first and second eigenvalue.

This algorithm offers a combination of advantages from previous approaches, such as having a logarithmic dependence on accuracy ε for high-accuracy solutions, and a runtime that scales with the sum of data size n and eigengap parameter λ rather than their product, making it applicable even when the eigengap is small. The algorithm has a runtime bound of $d_s n$ up to logarithmic factors, which is proportional to the time required for a single scan of the data and is better than previous approaches as long as $\lambda \geq \Omega\left(\frac{1}{\sqrt{n}}\right)$.

The pseudocode of the algorithm is described at the paper. It is important to note that it has some hyperparameters such as step-size η, epoch length m (execution of the outer loop).

<u>**Experiment**</u>

The paper analyzes 3 types of datasets: A synthetic dataset with different choices of eigengap λ, and the training data of the well-known MNIST and CCAT datasets. However, evaluation of the proposed algorithm on a wider range of datasets could provide more insights into the practical performance of the algorithm, its limitations, and help validate the assumptions and conclusions drawn from the theoretical analysis.

CIFAR-10 is a dataset of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is commonly used in computer vision research and has a different structure and characteristics than the MNIST and CCAT datasets used in the paper. The main differences are that CIFAR-10 consists of color images, which add more dimensionality to the data, and the classes have more variability, which adds additional complexity.

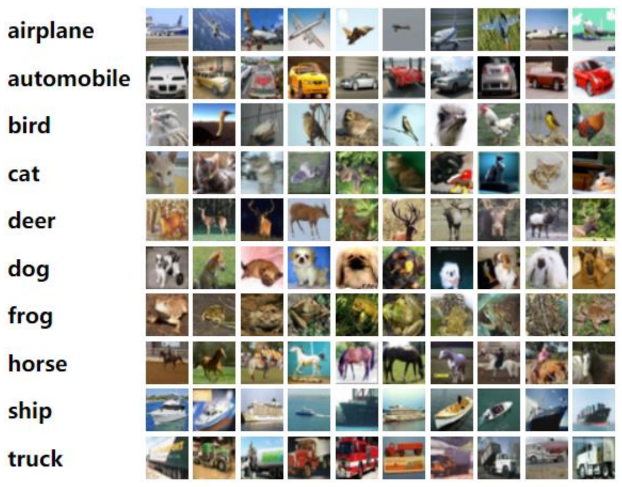

Figure 1 Classes of CIFAR-10 dataset

In addition to the pre-processing that was done on the MNIST dataset (centering the data and dividing each coordinate by its standard deviation times the square root of the dimension), the data was converted to grayscale and flattened, resulting in 32 x 32 x 3 = 3072 features. In total, the data matrix was $X \in \mathbb{R}^{3072x60000}$.

To evaluate if the numerical result of the paper fits this dataset, I implemented VR-PCA, the hybrid method for VR-PCA (initializing the VR-PCA algorithm with the result of running n iterations of Oja's algorithm), power iteration, and Oja's algorithm.

I used all the assumptions and parameters presented in the paper:

1. The epoch length m was set to n (number of data points, or columns in the data matrix)
2. $\eta = \frac{1}{\bar{r}\sqrt{n}}$, where $\bar{r} = \frac{1}{n}\sum_{i=1}^{n}\|x_i\|^2$
3. $\tilde{w}_0$ – initial guess vector, is chosen uniformly at random from the unit sphere and is the same for all algorithms

Assumption number 2 states that the choice of η is based on the theoretical analysis performed by the author of the paper. Specifically, the analysis requires η to be on the order of $\frac{1}{\max_i\|x_i\|^2\sqrt{n}}$ when m should be on the order of n. The main advantage of this choice is that it can be readily computed from the data without requiring knowledge of λ.

While the paper provides a thorough proof of this theorem, a short experiment was conducted on the CIFAR-10 dataset to demonstrate the practical application of this assumption. The experiment involved selecting different values of step size η within the range of $[\frac{\eta}{20}, 20\eta]$ and examining the convergence rate.

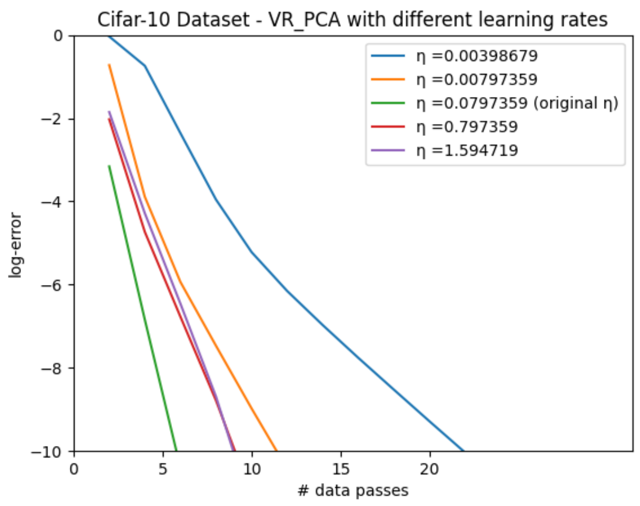

Figure 2 Results for CIFAR-10 dataset, using different step sizes.

The results presented in Figure 2 confirm the article writer's assumption, showing that the chosen value of η had the fastest convergence rate. It is important to note that this experiment was conducted only on the specific dataset with a limited range of η values and should be repeated on a larger dataset to ensure the generalizability of the results.

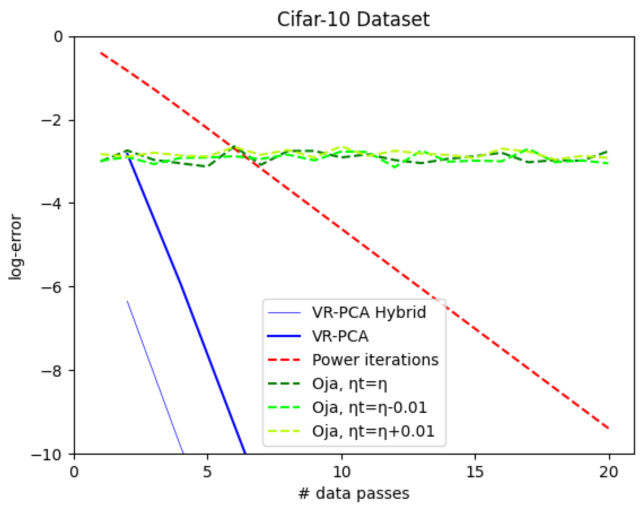

Figure 3 Results for CIFAR-10 dataset, using different algorithms.

The results presented in Figure 3 are consistent with those reported in the paper. VR-PCA exhibits faster convergence than all versions of Oja's algorithm, upon which it is based, as well as power iteration. The hybrid method performs slightly better than VR-PCA. Additionally, the convergence rate of Oja's algorithm appears to be sub-exponential for all step sizes tested, which is consistent with the findings of the original article. It is worth noting that the convergence rate of VR-PCA is linear, meaning it is exponential in a logarithmic scale, as noted in the paper.

While VR-PCA has shown strong performance in some scenarios, there remains significant interest in developing faster and more efficient algorithms for computing the first few principal components of large datasets. One possible avenue for improvement is to reduce the runtime of the algorithm, which currently depends on $\frac{1}{\lambda^2}$, to depend on $\frac{1}{\lambda}$ instead and explore methods to make the algorithm gap-free. By reducing computational costs and guaranteeing convergence to the exact solution, such improvements could help make PCA and related techniques more practical and applicable to a wider range of applications, particularly in the context of big data analysis.

Further Research

In the paper, the authors remarked on how to generalize the algorithm for multiple singular vectors. They suggested two methods: one was replacing the d-dimensional vectors with $d \times k$ matrices, and the normalization step with an orthogonalization step. The other option is to recover the singular vectors one by one via matrix deflation. However, this approach requires a positive eigengap between all top k singular values, or else the algorithm may not converge.

The paper provides a brief example of convergence rate analysis for the MNIST and CCAT datasets, resulting in a similar conclusion to the k = 1 case. While it studied it empirically, no analysis was provided. Further research needs to be conducted to extend the analysis of the existing algorithm to handle the case of k > 1 and make the algorithm more efficient for large-scale applications.

To conduct such research, the first step would be to review the existing literature on randomized algorithms for computing multiple singular vectors, focused on VR-PCA.

In [2], the author proposed a variant of VR-PCA to handle the k > 1 case, and formally analyzed its convergence. It was mentioned that the generalization to k > 1 is far from trivial and requires carefully tracking the evolution of the subspace spanned by the current iterate. This paper provides a formal and detailed analysis of the VR-PCA algorithm for k > 1 and found that these methods achieve convergence rates like the k=1 case, even in the block case. They also discovered that initializing the algorithm with a single power iteration can improve the resulting bounds.

The paper concludes with several open questions, in particular, they do not know whether the quadratic dependence on the $\frac{1}{\lambda}$ (eigengap) is inevitable, or whether a linear dependence or no dependence at all might be possible.

In [3], the authors investigated the k-SVD algorithm to obtain the first k singular vectors of a matrix while comparing its performance with state-of-the-art algorithms: such as the first gap-free k-SVD for block Krylov method, first variance-reduced stochastic k-SVD method [2] by Shamir, and the fastest non-zero running time algorithm.

Focusing on Shamir's work, the authors note that although his algorithm has better (local) performance than power methods, it is not gap-free, not accelerated, and requires a very accurate warm-start, which, in principle, can take a long time. In this paper [3], they develop a new algorithmic framework to solve k-SVD that not only improves upon previous breakthroughs but also relies solely on a simple, unified framework. Compared to Shamir's work, their algorithm is stochastic, accelerated, and gap-free.

The main idea of their algorithm is to choose our favorite 1-SVD algorithm and run it on the original matrix projected to the orthogonal space of previously obtained vectors, appending the new vector obtained to the matrix each time. The main restriction is that the 1-SVD algorithm needs to run in a poly-logarithmic with respect to $\frac{1}{\varepsilon}$ to prevent the error from blowing up, and Shamir's algorithm falls into this category.

To further deepen the research, I would suggest developing this method for algorithms that run in polynomial time, rather than poly-logarithmic time as in streaming settings where we don't have all our data at once. While the method suggested above can still be applied, the error may blow up. Once a modified algorithm is developed, it should be compared to other available methods, and the convergence rate and accuracy of the algorithm analyzed. Experiments could be conducted on various datasets with different characteristics (e.g., size, sparsity, etc.), and the performance of different algorithms compared in terms of convergence rate, accuracy, and computational efficiency.

To summarize, I began by analyzing VR-PCA for the k = 1 case on a different dataset than the ones shown by Shamir [1]. I then proposed developing the original algorithm, VR-PCA, for k > 1. However, I discovered that Shamir had already accomplished this [2] and [3] had even further improved the algorithm to be stochastic, accelerated, and gap-free. Finally, I suggested conducting further research to make the algorithm applicable in a streaming setting.

References

[1] O. Shamir, "A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate," in International Conference on Machine Learning, pp. 143-153, 2015.

[2] O. Shamir, "Fast Stochastic Algorithms for SVD and PCA: Convergence Properties and Convexity," in International Conference on Machine Learning, pp. 248–256, 2016.

[3] Z. A. Z. a. Y. Li, "Even faster SVD decomposition yet without agonizing pain," arXiv, 2016.