# A SURVEY OF METHODS FOR STREAMING PCA

NAME: BERNARDO BIANCO PRADO,
UNIQNAME: BEPRADO,
UM ID: 42685820

## 1. Main References

The main references for this review are [1], [3], [4], [5], [6], [8].

## 2. Introduction

Principal component analysis (PCA) is a commonly used method to understand underlying structures of data. In many cases, it helps reduce the dimensionality of data and thus, the computational cost of manipulating it. PCA is an effective method, but can be computationally expensive since it has a time complexity of $\mathcal{O}(nd^2)$ for each update, where $n$ is the number of data points and $d$ is the number of features of the data [3]. In particular, if one receives new data routinely, performing this algorithm will be impracticable. This is indeed the situation in many important problems in science and engineering. For example, in recommendation systems such as Netflix recommended movies or in the computer vision problem of reconstructing a 3D image from a variety of 2D pictures [3]. In these problems, one must consistently update the principal components as new data becomes available in order to produce improved results. In addition, these problems share the peculiarity that the data available to the algorithm has missing entries. In order to deal with the constant inflow of data and the missing information, one needs a series of improved PCA methods. In this paper, we will survey a series of *streaming PCA* methods that attempt to circumvent these two obstacles and discuss their effectiveness in doing so.

## 3. Problem Statement and Notation

We will survey three categories of streaming PCA: algebraic methods, geometric methods and deep learning methods. All these approaches will arrive at similar results but provide different perspectives and computational approaches. In particular, they will have different run times, convergence rates, or computational and memory complexities.

In this review, we will denote matrices by upper case letters such as $A$ and vectors by lower case letter, such as $x$. A subscript on a matrix or a vector will denote an ordering of the matrices or vectors, for example $x_l$ will mean the $l$-th data point of an ordering. We will write $x_l(i)$ to mean the $i$-th entry of the vector $x_l$. In this article, we will assume vectors $x$ lie in $\mathbb{R}^d$ unless otherwise specified and we will use the letter $n$ for the number of data instances in our problem. We will write $\ell^p$ for the norm $||x||_p = (x(1)^p + ... + x(d)^p)^{1/p}$. For example, $||x||_2$ is just the Euclidean norm of the vector $x \in \mathbb{R}^d$. We will denote by $k$ the dimension of the subspace we are estimating with the PCA algorithms. A boldface $\mathbb{E}$ will denote the expectation operator common in probability theory. For example, $\mathbb{E}[x]$ will mean the expected value of a random vector $x$. Finally, $I_k$ will denote the $k$ by $k$ identity matrix.

## 4. Algorithms for Streaming PCA

4.1. **Fundamental Ideas.** We begin this section with some fundamental ideas that will be recurrent throughout our survey. A first step is to realize that PCA can be framed in a deterministic, probabilistic or unified setting. In our survey, we will look at it mostly through the lens of a minimization problem that unifies both the probabilistic and deterministic paradigms

*Date*: October 4, 2022.

(where the deterministic is seen as a sampled version of the probabilistic). For a classical PCA algorithm, we seek to minimize the following loss function [3] for a random vector $x$

$$J(U) = \mathbb{E}||x - UU^T x||_2^2 \approx \frac{1}{n}\sum_{l=1}^{n} ||x_l - UU^T x_l||_2^2$$

over matrices $U \in \mathbb{R}^{d\times k}$ so that $U^T U = I_k$. For large enough $n$, the functional $J$ is well approximated by the sum on the right. Which can be seen as the sum of the residual vectors from the projection onto the column space of $U$.

Methods that treat PCA as a minimization problem have the advantage of low computational complexity through the use of iterative methods such as gradient descent or stochastic gradient descent. This is particularly useful in the case of streaming PCA since this problem requires one to constantly update the optimal $k$-subspace [3].

In the case in which we don't observe all features in our data, one can usually look at a similar functional without penalty for the unobserved entries. For this purpose, we define [3] the operator $\mathcal{P}_{\Omega_n}$ (here $\Omega_n$ is a vector with entries 0 and 1) that maps a vector $x$ to a new vector $\mathcal{P}_{\Omega_n}(x)$ that has $i$-th entry $x(i)$ if $\Omega_n(i) = 1$ and 0 if $\Omega_n(i) = 0$. Informally, $\mathcal{P}_{\Omega_n}$ ignores unobserved features. One may then look at a new functional that is the center of the Grassmannian Rank-One Update Subspace Estimation (GROUSE) algorithm that we cover next.

4.2. **The GROUSE Algorithm.** The GROUSE algorithm [4] applies the ideas above and solves the streaming PCA problem with missing data by minimizing the loss function

$$F(U) = \sum_{l=1}^{n} ||\mathcal{P}_{\Omega_n}(x_l - UU^T x_l)||_2^2$$

over the constraint set $U^T U = I_k$. It turns out this is equivalent [3], [4] to minimizing the loss function

$$\sum_{l=1}^{n} \min_{w} ||\mathcal{P}_{\Omega_n}(x_l - Uw)||_2^2 = \sum_{l=1}^{n} \mathrm{dist}(x_l, \mathcal{P}_{\Omega_n}(\mathcal{S}))^2$$

where $\mathcal{S}$ is the subspace spanned by the columns of $U$. Thus, minimizing the functional $F$ is equivalent to minimizing the more geometrically intuitive functional which looks at the best average fitting subspace $\mathcal{S}$ to the data at hand.

The loss function $F$ is minimized through a geometric modification of the stochastic gradient descent (SGD) algorithm. In particular, because the constraint set $U^T U = I_k$ is non-Euclidean, we must modify the SGD algorithm so our gradient step does not leave the constraint set. Fortunately [4], this constraint set is well understood and we can explicitly compute its geodesics (shortest distance paths). The SGD algorithm update step is changed to follow a geodesic starting at the current matrix $U_{n-1}$ with initial direction equal to the gradient

$$\nabla_U ||\mathcal{P}_{\Omega_n}(x_l - UU^T x_l)||_2^2 \Big|_{U=U_{n-1}}$$

of a randomly chosen data point $l$. The step size parameter now becomes the geodesic "time" parameter that will control how far the geodesic should go in the update step. The requirement on this step parameter to ensure convergence to at least a local minimum is that

$$\lim_{n\to\infty} \eta_n = 0\,, \quad \sum_{n=1}^{\infty} \eta_n = +\infty\,.$$

So they must decay, but not decay too fast. This algorithm has computational complexity of $\mathcal{O}(dk + |\Omega_n|k^2)$ for each update which is an improvement from standard PCA if $k \ll n$.

This algorithm performs well [4] against state of the art streaming PCA algorithms but has the disadvantage that it can be sensitive to noise in the data [6]. A more robust algorithm that follows a similar framework is the Grassmannian Robust Adaptive Subspace Tracking Algorithm (GRASTA).

4.3. **GRASTA.** The source for the lack of robustness in the GROUSE algorithm lies in the $\ell^2$ norms in the loss function [6]. To remediate that, GRASTA looks at a similar functional with $\ell^2$ norm substituted by the $\ell^1$ norm:

$$F_{grasta}(U) = \sum_{l=1}^{n} ||\mathcal{P}_{\Omega_n}(x_l - UU^T x_l)||_1$$

with the same constraint $U^T U = I_k$. In contrast to the GROUSE algorithm, GRASTA does not simply perform the modified SDG due to the nonlinear constraint [6]. It utilizes a framework that explores the augmented Lagrangian and the dual formulation of the problem. In particular, they write the augmented Lagrangian and perform alternating updates between the dual variables and the constrained matrix $U$. For the dual variable update, they use a method called Alternating Direction Method of Multipliers (ADMM) [2] and for the matrix update step, they use the manifold adaptation of SGD as in the GROUSE algorithm [6].

Another popular algorithm that takes the geometric approach to PCA is called Projection Approximation Subspace Tracking (PAST).

4.4. **The PAST Algorithm.** The PAST algorithm [8] assigns more relevance to current data in comparison to past data, and thus is well suited for the problem of streaming PCA, where the data's distribution may be changing over time. It does so by minimizing the following loss function,

$$L(U) = \sum_{l=1}^{n} \lambda^{n-l} ||x_l - UU^T x_l||_2^2$$

where $0 < \lambda \leq 1$. This allows the algorithm to be more adaptable to changes in the environment. The requirement that $U^T U = I_k$ is dropped because the map $UU^T$ is a good approximation to the constrained projection [3]. In fact, PAST goes one step further and approximates this loss function by

$$L_{past}(U) = \sum_{l=1}^{n} \lambda^{n-l} ||x_l - UU_{l-1}^T x_l||_2^2 \,.$$

In other words, it utilizes previous computed matrices $U_l$ for $l = 0, ..., n-1$ to simplify the functional. In the end, after all these simplifications, it performs [8] standard SGD on $L_{past}$ and achieves a computational complexity of $\mathcal{O}(dk)$ for each update, a significant improvement on the standard PCA algorithm and even on the GROUSE algorithm.

The way it was formulated, the PAST algorithm is not well suited to deal with the missing data case. Fortunately, this can be fixed through a similar process as w outlined earlier in this review. This leads us to the Parallel Estimation and Tracking by Recursive Least-Squares (PETRELS) algorithm.

4.5. **The PETRELS Algorithm.** The main difference between the PETRELS and the PAST algorithms is that PETRELS incorporates the projections $\mathcal{P}_{\Omega_l}$ into the loss function. Namely, it minimizes the function

$$L_{petrels} = \sum_{l=1}^{n} \lambda^{n-l} ||\mathcal{P}_{\Omega_l}(x_l - UU^T x_l)||_2^2 \,.$$

It's worth mentioning that the addition of the weights $\lambda^{n-l}$ makes this algorithm less susceptible to outliers than standard PCA or the GROUSE algorithm. Similarly to previous methods, the PETRELS algorithm performs SGD, but now of second order. This is in contrast with the GROUSE algorithm, which performs first order SGD. This may give faster convergence of the descent at the cost of a larger computational complexity of $\mathcal{O}(|\Omega_n|k^3)$. This algorithm can be paralellized to yield a complexity of $\mathcal{O}(k^3)$.

4.6. **Autoencoder Methods.** We have discussed algorithms for PCA that focus on minimizing a loss function. These have nice complexity problems that are suitable for streaming PCA. We will now cover a method that takes on a different perspective. This perspective comes from the observed equivalence of PCA with certain autoenconders [7]. This provides a path to a nonlinear version of PCA that may be useful in complex problems.

In particular, [1] implemented an autoencoder and compared it to the GROUSE and GRASTA algorithms. This more suitably applied to images since it uses convolutional neural networks (CNN). The autoencoder has is composed of 5 convolutional layers with 64 $5 \times 5$ and one fully connected layer filters. The network uses a tanh activation functon. The initial images had dimensions $576 \times 604$ and the ourput had dimension 2048. The authors used an $\ell^1$ loss function in the network, making it robust to outlier data. The loss function is simply a weighted sum of differences in the original image and the reconstructed image, where the weights are learned in training.

This method was compared to the GROUSE and GRASTA algorithms and achieved a runtime of about 20 times faster than GROUSE and 6 times faster than GRASTA. This comparisons are only empirical since theoretical analysis of the autoencoder can be difficult and was not explored in the paper. On the other hand, this method may prove fruitful for many video applications at the cost of giving up the intuition that we gain from the other algorithms.

## References


[1] A. Akhriev and J. Marecek, "Deep Autoencoders with Value-at-Risk Thresholding for Unsupervised Anomaly Detection," 2019 IEEE International Symposium on Multimedia (ISM), 2019, pp. 208-2083, doi: 10.1109/ISM46123.2019.00045.

[2] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers." *now Publishers, Hanover, Massachusetts*; Delft, The Netherlands, 2011.

[3] L. Balzano, Y. Chi and Y. M. Lu, "Streaming PCA and Subspace Tracking: The Missing Data Case," *in Proceedings of the IEEE*, vol. 106, no. 8, pp. 1293-1310, Aug. 2018, doi: 10.1109/JPROC.2018.2847041.

[4] L. Balzano, R. Nowak and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2010, pp. 704-711, doi: 10.1109/ALLERTON.2010.5706976.

[5] Y. Chi, Y. C. Eldar and R. Calderbank, "PETRELS: Parallel Subspace Estimation and Tracking by Recursive Least Squares From Partial Observations," *in IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5947-5959, Dec.1, 2013, doi: 10.1109/TSP.2013.2282910.

[6] J. He, L. Balzano and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1568-1575, doi: 10.1109/CVPR.2012.6247848.

[7] G. E. Hilton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science (American Association for the Advancement of Science)*, vol. 313, no. 5786. Washington, DC: American Association for the Advancement of Science, pp. 504-507.

[8] Bin Yang, "Projection approximation subspace tracking," *in IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 95-107, Jan. 1995, doi: 10.1109/78.365290.