A Comparison of Stochastic Methods for PCA as Applied to Streaming Facial Recognition[*]

Corbin Rosset
Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218, USA
crosset2@jhu.edu

Edmund Duhaime
Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218, USA
eduhaim1@jhu.edu

Abstract

This study applies a variety of stochastic and incremental techniques for principle component analysis (PCA) to quickly learn maximally informative linear subspaces on a streaming version of the Yale Face Data Set B. We compare the performance of a K-nearest neighbor classifier and a bagged tree learner on the best subspaces learned by each of: Stochastic Power Method (SPM), Incremental PCA (IPCA), Matrix Stochastic Descent (MSG), and Sparse PCA (SPCA). We compare and contrast the theoretical properties such as correctness, space, and iteration complexity. In all cases, the memory required to store a subspace capable of achieving accuracy similar to that of the baseline classifier was 2-4% of the size of the input data. These algorithms play an important role in improving the scalability of facial recognition in a streaming setting.

1 Introduction

The premise of subspace learning is to map a data matrix $X \in \mathbb{R}^{d \times n}$ of n examples each of dimension d to a k dimensional subspace, $k << d$ that preserves maximal "information". The motivation is that most data sets capture redundant, verbose, or noisy features that mask the underlying structure of the data. It is often the case that natural processes are largely controlled by finitely many simpler mechanisms that operate in lower dimensions. For example, latent variables captured by images such as shadows, reflections, perspective can drastically alter pixel values even though such phenomenon are easily parameterized. PCA seeks to find low dimensional linear representations of these latent variables such that, when transformed back into the original space, the loss of information is minimal. In the next section, we will pose PCA as an optimization problem with multiple equivalent interpretations, and then derive its empirical solution. We will then describe and derive solututions to the state of the art stochastic and incremental approximation algorithms to the PCA objective. We applied each of the algorithms on the Yale Face Data Set B in a streaming fashion and trained a K-NN neighbor classifier on the respective learned subspaces.

[*]We would like to thank Dr. Raman Arora for his work on stochastic algorithms for manifold learning.

2 PCA and its Stochastic Variants

Given n data points in $\mathbb{R}^d$, find an orthogonal projection matrix $U \in \mathbb{R}^{d \times k}$ such that the projection $\hat{x} \in \mathbb{R}^k$ of a data vector $x \in \mathbb{R}^d$ given by $\hat{x} = UU^\top x$ minimizes the empirical reconstruction error:

$$\begin{aligned}\text{Error} &= \frac{1}{n}\sum_{i=1}^{n} \|x_i - UU^\top x_i\|_2^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \left(\|x\|_2^2 - \|U^\top x_i\|_2^2\right)\end{aligned} \tag{1}$$

In Equation 1, the term $\frac{1}{n}\sum_{i=1}^{n} \|x_i\|_2^2$ is constant given any data set x. So in order to minimize the reconstruction, or projection, error, we must maximize $\frac{1}{n}\sum_{i=1}^{n} \|U^\top x_i\|_2^2$. This is equivalent to maximizing the trace of $U^\top \left[\frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top\right] U$ where $C_{xx} =$

$\frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$ is the empirical covariance matrix.

Since the i'th diagonal entry of C_{xx} is the variance [1] of x_i, $var(x_i) = \mathbb{E}[x_i^\top x_i]$ the trace of C_{xx} is the variance of the whole data matrix. Hence it is natural to interpret $\mathbb{E}[||U^\top x_i\|_2^2] = \mathbb{E}[U^\top x_i x_i^\top U]$ as the variance of x_i captured by U. Thus, PCA serves to minimize reconstruction error, or maximize the variance of the data captured by the subspace U. Derived in the appendix, it turns out that U is comprised of the top-k eigenvectors of C_{xx} sorted in decreasing order of eigenvalue: $U = V_{:,1:k}$ for $C_{xx} = VSV^\top$.

This solution holds for any distribution $\mathcal{D}$ of the data as long as each of the n samples are drawn i.i.d from it. For a correct empirical covariance matrix, a simple $O(d^2mk)$ time algorithm known as the power iteration method calculates $U \in \mathbb{R}^{d\times k}$ using $O(m)$ iterations per component [2].

The most pressing challenge is the temporal[3] and spatial dependence of these solutions on powers of d, which can easily be intractable for $d \geq 10^5$, which is common for images. Secondly, a batch algorithm may not satisfy modern computational desires, for many modern applications require responses to realtime streaming data arriving at high frequency. In such streaming models, it is also desired that the algorithm adapt to and track the optimal subspace.

Subspace learning (linear or nonlinear) is ubiquitous in data-driven applications such as compression, de-noising, visualization and matrix completion.

2.1 Stochastic Power Method

The optimization problem describing PCA,

$$\begin{aligned} \max_{U\in\mathbb{R}^{d\times k}} \quad & \mathbb{E}_{\mathcal{D}}\left[\text{trace}\left(U^\top x x^\top U\right)\right] \\ \text{subject to} \quad & U^\top U = I \end{aligned} \tag{2}$$

is in fact nonconvex due to the constraint and the objective function being a maximization. A convex relaxation is given by the constraint that $U^\top U \preceq I$ meaning all eigenvalues of C_{xx} are at most one, and optimally equally to one. Assuming $\mathcal{D}$ is unknown but unchanging, the goal is to update U directly as samples x_i arrive sequentially and independently. Since we have access to neither $\mathcal{D}$ nor the population of samples ahead of time, neither C_{xx} nor its empirical estimate can be computed. However, the i.i.d assumption admits $\mathbb{E}[x_t x_t^\top] = C_{xx}$.

In this setting, gradient descent is a viable algorithm, with updates of the form

$$U^{(t+1)} = \mathcal{P}_{orth}\left(U^{(t)} + \eta_t x_t x_t^\top U^{(t)}\right) \tag{3}$$

The projection of the updated U onto the set of orthonormal matrices, $\mathcal{P}_{orth}$ can be accomplished in $O(k^2d)$ time using Gram-Schmidt or QR factorization. Notice that instead of $O(d^2)$ memory, the updates require only $O(kd)$. The time complexity is $O(Tkd)$ for T iterations. While this algorithm converges with probability 1, the rate of convergence is unknown. Obviously, if the true covariance C_{xx} were known, then the objective function would become trace $\left(U^\top C_{xx} U\right)$ with derivative $2C_{xx}U$, which would replace the gradient term in the update above. The

2.2 Incremental PCA

Derived from incremental singular value decomposition (0), a rank-k approximation to the empirical covariance matrix can be updated incrementally:

$$C_{xx}^{(t+1)} = \Pi_k\left(C_{xx}^{(t)} + x_t x_t^\top\right) \tag{4}$$

where Π_k is the projection onto the set of rank-k matrices with respect to the spectral norm. To avoid explicit computation the $d \times d$ matrix $x_t x_t^\top$, we assume by the inductive hypothesis[4] that a rank-ℓ approximation to $C_{xx}^{(t)} = USU^\top$ is given, and make rank one updates to Equation 4 of the form:

$$\begin{bmatrix} U & \frac{x_\perp}{||x_\perp||} \end{bmatrix} \begin{bmatrix} S + \hat{x}\hat{x}^\top & ||x_\perp||_2 \hat{x} \\ ||x_\perp||_2 \hat{x}^\top & ||x_\perp||_2^2 \end{bmatrix} \begin{bmatrix} U \\ \frac{x_\perp}{||x_\perp||} \end{bmatrix} \tag{5}$$

for $\hat{x} = U^\top x_t$ the coefficients of the projection of x_t onto the column space of U, and $x_\perp =$

[1]It is assumed X is centered $\mathbb{E}[x_i] = 0$, eliminating the $(\mathbb{E}[x_i])^2$ term in the variance.

[2]the iteration complexity depends on the eigengap between consecutive components; a larger difference in eigenvalues implies faster convergence

[3]the fastest known algorithm for brute force eigendecomposition of C_{xx} is $O(d^3 + d^2 log^2 d)$

[4]Initialize the algorithm with U and S equal to all zeros.

$x_t - UU^\top x_t$ its orthogonal component. The 2nd matrix in Equation 5, denoted Q, is symmetric of size $\ell + 1 \times \ell + 1$, and therefore Q itself can be eigendecomposed $Q = U'S'U'^\top$ in $O(k^3)$ time[5] with positive eigenvalues that happen to be those S plus one more from the rank-one update. Finally, set $U = \begin{bmatrix} U & \frac{x_\perp}{||x_\perp||} \end{bmatrix} U'$ and $S = S'$ and truncated each to retain the top-k eigenvectors and eigenvalues sorted in decreasing spectral energy.

The memory required is $O(kd)$ and runtime is $O(Tk^2d)$ dominated by the update to U each of the T iterations. There are also no parameters to tune here. In practice, this algorithm is most expeditious to converge, but there are data distributions that arise naturally or adversarially that cause the algorithm to converge to an incorrect subspace with high probability (0). Some proposed remedies have been to store rank-k' matrices, $k' > k$ or use mini-batch updates to lower the probability of failure.

2.3 Matrix Stochastic Gradient

The loss function in Equation 1 can be rewritten as follows:

$$\begin{aligned} \min_{P \in \mathbb{R}^{d\times d}} \quad & \mathbb{E}_{\S\tilde{\mathcal{D}}} \left[x^\top (I - P) x \right] \\ \text{subject to} \quad & \text{rank}(P) \leq k, 0 \preceq P \preceq I \end{aligned} \tag{6}$$

for $P = UU^\top$ an orthogonal projection matrix. Again, a convex relaxation replaces the constraints with: trace$(P) = k$, $0 \preceq P \preceq I$. The iterate updates are:

$$P^{(t+1)} = \Pi_{\text{trace}(P)=k, 0\preceq P \preceq I} \left(P^{(t)} + \eta_t x_t x_t^\top \right) \tag{7}$$

which again is rewritten in the form of incremental SVD to preserve only a rank-k approximation of the true projection matrix, and to avoid the $d \times d$ calculation. Assume an approximation of the eigendecomposition is given as $P^{t)} = USU^\top$ for $P \in \mathbb{R}^{d\times k}$, then for a new example x_t, $P^{(t+1)} + x_t x_t^\top$ has the same update as Equation 5 for the same definition of $x_\perp$ and $\hat{x}$.

However, unlike IPCA, the projection step here $\Pi_{\text{trace}(P)=k,0\preceq P\preceq I}$, is necessary; it can be computed as follows:

[5]it will be enforced that $\ell = k$

$S_{ii} \leftarrow max(0, min(1, S'_{ii} + \mu))$ for $\mu \in \mathbb{R}$ such that $\sum_i S_i i = k$ is a shifting of all eigenvalues by a constant so that their sum is k. This is merely a rescaling of the learned subspace, and as before, $U \leftarrow \begin{bmatrix} U & \frac{x_\perp}{||x_\perp||} \end{bmatrix} U'$, but only those columns corresponding to nonzero S_{ii}.

2.4 Sparse PCA

notes from class

3 K-NN for High-Dimensional Data

discuss johnson lindenstrauss lemma popularly applied to proximity problems to preserve pairwise distances in a lower dimensional space

4 Experiments

The data used for the experiments was Extended Yale Face Dataset B (B+) [6]. This dataset includes images of 38 people with light at various angles to their faces. Each image is of size 168x192 pixels and in grayscale. The total dataset is approximately 2400 images with 64 images for each person.

Some images were deemed to be of too poor quality to include in the data used for the experiment, as no discernible features were present and the classifier would only be guessing. All the images were manually evaluated to determine which sets to take out, and then specific types of lighting angles were removed from the dataset. This reduced the dataset to about 1850 images.

The reduced dataset was then split into 60% train data, 20% test data, and 20% dev data. For each of the previously described algorithms 80 principle components were generated from the reduced dataset training data. Then a KNN classifier was trained on the principle components, starting from a single principle component and going up to 80. The built-in Matlab KNN classifier was used for this. The accuracy of the classifier was recorded and cross validation performed each time a principle component was added. The KNN classifier was also run directly on the whole dataset.

In addition to accuracy of the classifiers, images were reconstructed using the principle components

[6]http://vision.ucsd.edu/content/extended-yale-face-database-b-b

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word “Abstract”	12 pt	bold
section titles	12 pt	bold
document text	11 pt	
abstract text	10 pt	
captions	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 1: Font guide.

to see how well they had captured the faces. “Eigenfaces” (the top five principle components combined and shown as images) were also calculated and stored.

5 Results

NED: pictures of 1) reconstructed faces with 5, 10, 15... principle components 2) table of the best dimension and associated accuracy achieved by each algorithm 3) some examples of “eigenfaces” (the top five principle components shown as images) as well as their thresholded versions. You’ll find these in the figures directory. 4) graphs of accuracy versus number of principle components for some of the algorithms (choose the better looking ones)

Captions: Provide a caption for every illustration; number each one sequentially in the form: “Figure 1. Caption of the Figure.” “Table 1. Caption of the Table.” Type the captions of the figures and tables below the body, using 10 point text.

6 Conclusion

all algorithms for pca should learn the same subspace, up to small rotations and scalings...

discuss applications of streaming subspace learning

there are extensions for incremental kernal pca and dealing with the case of missing data...

Acknowledgments

Do not number the acknowledgment section.

References

Arora, R.; Cotter, A.; Livescu, K.; Srebro, N., ”Stochastic optimization for PCA and PLS,” in Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on , vol., no., pp.861-868, 1-5 Oct. 2012

Brand, Matthew. *Incremental Singular Value Decomposition of Uncertain Data with Missing Values* In ECCV ’02: Proceedings of the 7th European Conference on Computer Vision-Part I (2002), pp. 707-720

the following are examples

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.