Candid Covariance-free Incremental Principal Component Analysis

Juyang Weng, Yilu Zhang and Wey-Shiuan Hwang[*]

Abstract

Appearance-based image analysis techniques require fast computation of principal components of high dimensional image vectors. We introduce a fast incremental principal component analysis (IPCA) algorithm, called candid covariance-free IPCA (CCIPCA), to compute the principal components of a sequence of samples incrementally without estimating the covariance matrix (thus covariance-free). The new method is motivated by the concept of statistical efficiency (the estimate has the smallest variance given the observed data). To do this, it keeps the scale of observations and computes the mean of observations incrementally, which is an efficient estimate for some well known distributions (e.g. Gaussian), although the highest possible efficiency is not guaranteed in our case because of unknown sample distribution. The method is for real-time applications, and thus it does not allow iterations. It converges very fast for high dimensional image vectors. Some links between IPCA and the development of the cerebral cortex are also discussed.

Index items: principal component analysis, incremental principal component analysis, stochastic gradient ascent (SGA), generalized hebbian algorithm (GHA), orthogonal complement.

1 Introduction

A class of image analysis techniques called appearance-based approach has now become very popular. A major reason that leads to its popularity is the use of statistics tools to automatically derive features instead of relying on human to define features. Although principal component analysis is a well known technique, Kirby & Sirovich [1] appears to be among the first who used the technique directly on the characterization of human faces – each image is considered simply as a high dimensional vector, each pixel corresponding to a component. Turk & Pentland [2] were among the first who used this representation for face recognition. The technique has been extended to 3-D object recognition [3], sign recognition [4] and autonomous navigation [5], among many other image analysis problems.

[*]The authors are with Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824. E-mail: {weng, zhangyil, hwangwey}@cse.msu.edu. Please send correspondence to Dr. Juyang Weng.

A well-known computational approach to PCA involves solving an eigensystem problem, i.e., computing the eigenvectors and eigenvalues of the sample covariance matrix, using a numerical method such as the power method and the QR method [6]. This approach requires that all the training images are available before the principal components can be estimated. This is called a batch method. This type of method no longer satisfies an up-coming new trend of computer vision research [7], in which all visual filters are incrementally derived from very long online real-time video stream, motivated by the development of animal vision systems. Online development of visual filters requires that the system performs while new sensory signals flow in. Further, when the dimension of the image is high, both the computation and storage complexity grow dramatically. For example, in the eigenface method, a moderate grey image of 64 rows and 88 columns results in a d-dimensional vector with $d = 5632$. The symmetric covariance matrix requires $d(d+1)/2$ elements, which amounts to 15,862,528 entries! A clever saving method can be used when the number of images is smaller than the number of pixels in the image [1]. However, an online developing system must observe an open number of images and the number is larger than the dimension of the observed vectors. Thus, an incremental method is required to compute the principal components for observations arriving sequentially, where the estimate of principal components are updated by each arriving observation vector. No covariance matrix is allowed to be estimated as an intermediate result. There is evidence that biological neural networks use an incremental method to perform various learning, e.g., Hebbian learning [8].

Several IPCA techniques have been proposed to compute principal components without the covariance matrix [9][10][11]. However, they ran into convergence problems when facing high dimensional image vectors. We explain in this article why. We propose a new method, candid covariance-free IPCA (CCIPCA), based on the work of Oja [10] and Sanger [11]. It is motivated by a well-known statistical concept called efficient estimate. An amnesic average technique is also used to dynamically determine the retaining rate of the old and new data, instead of a fixed learning rate.

2 Derivation of the Algorithm

2.1 The First Eigenvector

Suppose that sample vectors are acquired sequentially, $u(1), u(2), \ldots$, possibly infinite. Each $u(n)$, $n = 1, 2, \ldots$, is a d-dimensional vector and d can be as large as 5000 and beyond. Without loss of generality, we can assume that $u(n)$ has a zero mean (the mean may be incrementally estimated and subtracted out). $A = E\{u(n)u^T(n)\}$ is the $d \times d$ covariance matrix, which is neither known nor allowed to be estimated as an intermediate result.

By definition, an eigenvector x of matrix A satisfies

$$\lambda x = Ax, \tag{1}$$

where λ is the corresponding eigenvalue. By replacing the unknown A with the sample covariance matrix, and replacing the x of Eq. (1) with its estimate $x(i)$ at each time step i, we obtain an illuminating expression for $v = \lambda x$,

$$v(n) = \frac{1}{n}\sum_{i=1}^{n} u(i)u^T(i)x(i). \tag{2}$$

where $v(n)$ is the n-th step estimate of v. As we will see soon, this equation is motivated by statistical efficiency. Once we have the estimate of v, it is easy to get the eigenvector and the eigenvalue since $\lambda = ||v||$ and $x = v/||v||$.

Now the question is how to estimate $x(i)$ in Eq. (2). Considering $x = v/||v||$, we may choose $x(i)$ as $v(i-1)/||v(i-1)||$, which leads to the following incremental expression,

$$v(n) = \frac{1}{n}\sum_{i=1}^{n} u(i)u^T(i)\frac{v(i-1)}{||v(i-1)||}. \tag{3}$$

To begin with, we set $v(0) = u(1)$, the first direction of data spread. For incremental estimation, Eq. (3) is written in a recursive form,

$$v(n) = \frac{n-1}{n}v(n-1) + \frac{1}{n}u(n)u^T(n)\frac{v(n-1)}{||v(n-1)||}, \tag{4}$$

where $(n-1)/n$ is the weight for the last estimate and $1/n$ is the weight for the new data. We have proved that with the algorithm given by Eq. (4), $v_1(n) \to \pm\lambda_1 e_1$ when $n \to \infty$, where λ_1 is the largest eigenvalue of the covariance matrix of $\{u(n)\}$, and e_1 is the corresponding eigenvector [12].

The derivation of Eqs. (2)-(4) is motivated by statistical efficiency. An unbiased estimate $\hat{Q}$ of the parameter Q is said to be the *efficient estimate for the class D of distribution functions* if for every distribution density function $f(u, Q)$ of D the variance $D^2(\hat{Q})$ (squared error) has reached the minimal value given by

$$D^2(\hat{Q}) = E[(\hat{Q} - Q)^2] \geq \frac{1}{n\int_{-\infty}^{+\infty}\left[\frac{\partial \log f(u,Q)}{\partial Q}\right]^2 f(u,Q)du}. \tag{5}$$

The right side of inequality (5) is called Cramér-Rao bound. It says that the efficient estimate is one that has the least variance from the real parameter, and its variance is bounded below by the Cramér-Rao bound. For example, the sample mean, $\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w(i)$, is the efficient estimate of the mean of a Gaussian distribution with a known standard deviation σ [13]. For a vector version of the Cramér-Rao bound, the reader is referred to [14, page 203-204].

If we define $w(i) = u(i)u^T(i)x(i)$, $v(n)$ in Eq. (2) can be viewed as the mean of "samples" $w(i)$. That is exactly why our method is motivated by the statistical efficiency in using averaging in Eq. (2). In other words, statistically, the method tends to converge most quickly or the estimate has the smallest error variance given the currently observed samples. Of course, $w(i)$ is not necessarily drawn from a Gaussian distribution independently and thus the estimate using the sample mean in Eq. (4) is not strictly efficient. However, the estimate $v(n)$ still has a high statistical efficiency and has a fairly low error variance as we will show experimentally.

The Cramér-Rao lower error bound in Eq. (5) can also be used to estimate the error variance, or equivalently the convergence rate, using a Gaussian distribution model, as proposed and experimented with by Weng et al. [14, Section 4.6]. This is a reasonable estimate because of our near optimal statistical efficiency here. Weng et al. [14] demonstrated that actual error variance is not very sensitive to the distribution (e.g. uniform or Gaussian distributions). This error estimator is especially useful to estimate roughly how many samples are needed for a given tolerable error variance.

IPCA algorithms have been studied by several researchers [15] [16] [9] [10]. An early work with a rigorous proof for convergence was given by Oja & Karhunen [9] [10], where they introduced their stochastic gradient ascent (SGA) algorithm. SGA computes,

$$\tilde{v}_i(n) = v_i(n-1) + \gamma_i(n)u(n)u^T(n)v_i(n-1) \tag{6}$$

$$v_i(n) = \text{orthonormalize } \tilde{v}_i(n) \text{ w.r.t. } v_j(n), j = 1, 2, \ldots, i-1 \tag{7}$$

where, $v_i(n)$ is the estimate of the i-th dominant eigenvectors of the sample covariance matrix $A = E\{u(n)u^T(n)\}$, and $\tilde{v}_i(n)$ is the new estimate. In practice, the orthonormalization in Eq.(7) can be done by a standard *Gram-Schmidt Orthonomalization* (GSO) procedure. The parameter $\gamma_i(n)$ is a stochastic approximation gain. The convergence of SGA has been proved under some assumptions of A and $\gamma_i(n)$ [10].

SGA is essentially a gradient method, associated with which is the problem of choosing $\gamma_i(n)$, the learning rate. Simply speaking, the learning rate should be appropriate so that the second term (the correction term) on the right side of Eq. (6) is comparable to the first term, neither too large nor too small. In practice, $\gamma_i(n)$ depends very much on the nature of the data and usually requires a trial-and-error procedure, which is impractical for online applications. Oja gave some suggestions on $\gamma_i(n)$ in [9], which is typically $1/n$ multiplied by some constants. However, procedure (6) is at the mercy of the magnitude of observation $u(n)$, where the first term has a unit norm but the second can take any magnitude. If $u(n)$ has a very small magnitude, the second term will be too small to make any changes in the new estimate. If $u(n)$ has a large magnitude, which is the case with high dimensional images, the second term will dominate the right side before a very large number n and, hence, a small $\gamma_i(n)$, has been reached. In either case, the updating is inefficient and the convergence will be slow.

Contrasted with SGA, the first term on the right side of Eq. (4) is not normalized. In effect, $v(n)$ in Eq. (4) converges to λe instead of e as it does in Eq. (6), where λ is the eigenvalue and e is the eigenvector. In Eq. (4), the statistical efficiency is realized by keeping the scale of the estimate at the same order of the new observations (the first and second terms properly weighted on the right side of Eq. (4) to get sample mean), which allows fully use of every observation in terms of statistical efficiency. Note that the coefficient $(n-1)/n$ in Eq. (4) is as important as the "learning rate" $1/n$ in the second term to realize sample mean. Although $(n-1)/n$ is close to 1 when n is large, it is very important for fast convergence with early samples. The point is that if the estimate does not converge well at the beginning, it is harder to be pulled back later when n is large. Thus, one does not need to worry about the nature of the observations. This is also the reason that we used "candid" in naming the new algorithm.

It is true that the series of parameters, $\gamma_i(n)$, $i = 1, 2, ..., k$, in SGA can be manually tuned in an off-line application so that it takes into account the magnitude of $u(n)$. But a predefined $\gamma_i(n)$ can not accomplish statistical efficiency no matter how $\gamma_i(n)$ is tuned. This is true because all the "observations," i.e., the last term in Eq. (4) and Eq. (6), contribute to the estimate in Eq. (4) with the same weight for statistical efficiency, but they contribute unequally in Eq. (6) due to normalization of $v(n-1)$ in the first term, and, thus, damage the efficiency. Further, the manual tuning is not suited for an online learning algorithm since the user cannot predict signals in advance. An online algorithm must automatically compute data-sensitive parameters.

There is a further improvement to procedure (4). In Eq. (4), all the "samples" ($w(i) = u(i)u^T(i)\frac{v(i-1)}{||v(i-1)||}$), are weighted equally. However, since $w(i)$ is generated by $v(i)$ and $v(i)$ is far away from its real value at a early estimation stage, $w(i)$ is a "sample" with large "noise" when i is small. To speed up the convergence of the estimation, it is preferable to give smaller weight to these early "samples". A way to implement this idea is to use an amnesic average by changing Eq. (4) into,

$$v(n) = \frac{n-1-l}{n}v(n-1) + \frac{1+l}{n}u(n)u^T(n)\frac{v(n-1)}{||v(n-1)||}, \tag{8}$$

where the positive parameter l is called the amnesic parameter. Note that the two modified weights still sum to 1. With the presence of l, larger weight is given to new "samples" and the effect of old "samples" will fade out gradually. Typically, l ranges from 2 to 4.

2.2 Intuitive Explanation

An intuitive explanation of procedure (4) is as follows. Consider a set of 2-dimensional data with a Gaussian probability distribution function (For any other physically arising distribution, we can consider its first two orders of statistics since PCA does so). The data is charactrised by an ellipse as shown in Fig. 1. According to the geometrical meaning of eigenvectors, we know that the first eigenvector is aligned with the long axis (v_1) of the ellipse. Suppose $v_1(n-1)$ is the $(n-1)$th-step estimation of the first eigenvector. Noticing $u^T(n)\frac{v_1(n-1)}{||v_1(n-1)||}$ is a scalar, we know $\frac{1}{n}u(n)u^T(n)\frac{v_1(n-1)}{||v_1(n-1)||}$ is essentially a scaled vector of $u(n)$. According to procedure (4), $v_1(n)$ is a weighted combination of the last estimate, $v_1(n-1)$ and the scaled vector of $u(n)$. Therefore, geometrically speaking, $v_1(n)$ is obtained by pulling $v_1(n-1)$ toward $u(n)$ by a small amount.

A line, l_2, orthogonal to $v_1(n-1)$, divides the whole plane into two halves, the upper and the lower ones. Because every point u_l in the lower half plane has an obtuse angle with $v_1(n-1)$, $u_l^T\frac{v_1(n-1)}{||v_1(n-1)||}$ is a negative scalar. So, for u_l, Eq. (4) may be written as,

$$v(n) = \frac{n-1}{n}v(n-1) + \frac{1}{n}\left|u_l^T\frac{v(n-1)}{||v(n-1)||}\right|(-u_l),$$

where $-u_l$ is an upper half plane point obtained by rotating u_l for 180 degrees w.r.t. the origin. Since the ellipse is centrally symmetric, we may rotate all the lower half plane points to the upper half plane and only consider the pulling effect of upper half plane points. For the points u_u in the upper half plane, the pure force will pull $v_1(n-1)$ towards the direction of v_1 since there are more data points to the right side of $v_1(n-1)$ than those to the left side. As long as the first two eigenvalues are different, this pulling force always exists and the pulling direction is towards the eigenvector corresponding to a larger eigenvalue. $v_1(n-1)$ will not stop moving until it is aligned with v_1 when the pulling forces from both sides are balanced. In other words, $v_1(n)$ in Eq. (4) will converge to the first eigenvector. As we can imagine, the larger the ratio of the first eigenvalue over the second eigenvalue, the more unbalanced the force is, and the faster the pulling or the convergence will be. However, when $\lambda_1 = \lambda_2$, the ellipse degenerates to a circle. The movement will not stop, which seems that the algorithm does not converge. Actually, since any vector in that circle can represent the eigenvector, it does not hurt not converging. We will get back to the cases of equal eigenvalues in Section 2.4.

2.3 Higher-order Eigenvectors

Procedure (4) only estimates the first dominant eigenvector. One way to compute the other higher order eigenvectors is following what SGA does: Start with a set of orthonormalized vectors, update them using the suggested iteration step, and recover the orthogonality using GSO. For real-time

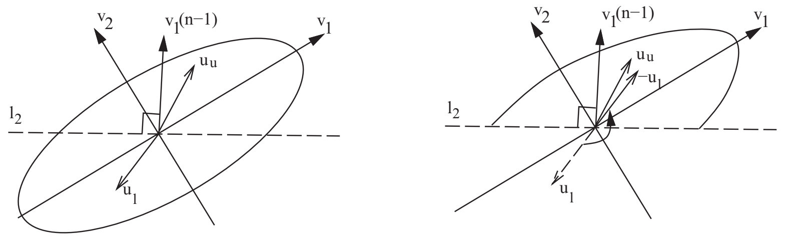

Figure 1: Intuitive explanation of the incremental PCA algorithm.

online computation, we need to avoid the time-consuming GSO. Further, breaking-then-recovering orthogonality slows down the convergence compared with keeping orthogonality all along. We know eigenvectors are orthogonal to each other. So, it helps to generate "observations" only in a complementary space for the computation of the higher order eigenvectors. For example, to compute the second order eigenvector, we first subtract from the data its projection on the estimated first order eigenvector $v_1(n)$, as shown in Eq. (9),

$$u_2(n) = u_1(n) - u_1^T(n)\frac{v_1(n)}{||v_1(n)||}\frac{v_1(n)}{||v_1(n)||}, \tag{9}$$

where $u_1(n) = u(n)$. The obtained residual, $u_2(n)$, which is in the complementary space of $v_1(n)$, serves as the input data to the iteration step. In this way, the orthogonality is always enforced when the convergence is reached, although not exactly so at early stages. This in effect well uses the sample available and thus speeds up the convergence.

A similar idea has been used by some other researchers. Kreyszig proposed an algorithm, which finds the first eigenvector using a method equivalent to SGA and subtracts the first component from the samples before computing the next component [17]. Sanger suggested an algorithm, called generalized hebbian algorithm (GHA), based on the same idea except that all the components are computed at the same time [11]. However, in both cases, the statistical efficiency was not considered.

The new CCIPCA also saves computations. One may notice that the expensive steps in both SGA and CCIPCA are the dot products in the high-dimensional data space. CCIPCA requires one extra dot product, i.e., $u_i^T(n)v_i(n)$ in Eq. (9), for each principal component in each estimation step. For SGA, to do orthonormalization over k new estimates of eigenvectors using GSO, we have totally $k(k+1)/2$ dot products. So, the average number of dot product saved by CCIPCA over SGA for each eigenvector is $(k-1)/2$.

2.4 Equal Eigenvalues

Let us consider the case where there are equal eigenvalues. Suppose ordered eigenvalues between λ_l and λ_m are equal:

$$\lambda_{l-1} > \lambda_l = \lambda_{l+1} = \ldots = \lambda_m > \lambda_{m+1}.$$

According to the explanation in Section 2.2, the vector estimate will converge to the one with a larger eigenvalue first. Therefore, the estimate of eigenvectors e_i, where $i < l$, will not be affected anyway.

The vector estimates of e_l to e_m will converge into the subspace spanned by the corresponding eigenvectors. Since their eigenvalues are equal, the shape of the distribution in Fig. 1 is a hypersphere within the subspace. Thus, the estimates of the multiple eigenvectors will converge to any set of the orthogonal basis of that subspace. Where it converges to depends mainly on the early samples, because of the averaging effect in Eq. (2), where the contribution of new data gets infinitely small when n increases without a bound. That is exactly what we want. The convergence of these eigenvectors are as fast as those in the general case.

2.5 Algorithm Summary

Combining the mechanisms discussed above, we have the candid covariance-free IPCA algorithm as follows.

Procedure 1 *Compute first k dominant eigenvectors, $v_1(n), v_2(n), \ldots, v_k(n)$, directly from $u(n)$, $n = 1, 2, \ldots$.*

```
For $n = 1, 2, \ldots$, do the followings steps,
1. $u_1(n) = u(n)$.
2. For $i = 1, 2, \ldots, \min\{k, n\}$ do,
    (a) If $i = n$, initialize the $i$th eigenvector as $v_i(n) = u_i(n)$.
    (b) Otherwise,
```

$$v_i(n) = \frac{n-1-l}{n}v_i(n-1) + \frac{1+l}{n}u_i(n)u_i^T(n)\frac{v_i(n-1)}{||v_i(n-1)||} \tag{10}$$

$$u_{i+1}(n) = u_i(n) - u_i^T(n)\frac{v_i(n)}{||v_i(n)||}\frac{v_i(n)}{||v_i(n)||} \tag{11}$$

A mathematical proof of the convergence of CCIPCA can be founded in [12].

3 Empirical Results on Convergence

We performed experiments to study the statistical efficiency of the new algorithm as well as the existing IPCA algorithms, especially for high dimensional data such as images. We define *sample-to-dimension ratio* as n/d where n is the number of samples and d is the dimension of the sample space. The lower the ratio, generally the harder a statistical estimation problem becomes.

First presented here are our results on the FERET face data set [18]. This data set has frontal views of 457 subjects. Most of the subjects have two views while 34 of them have four views and two of them have one view, which results in a data set of 982 images. The size of each image is 88-by-64 pixels, or 5632 dimensions. Therefore, this is a very hard problem with a very low sample-to-dimension ratio of $982/5632 = 0.7$.

We computed the eigenvectors using a batch PCA with QR method and used them as our ground truth. The program for batch PCA was adapted from the C Recipes [19]. Since the real mean of the image data is unknown, we incrementally estimated the sample mean $\hat{m}(n)$ by

$$\hat{m}(n) = \frac{n-1}{n}\hat{m}(n-1) + \frac{1}{n}x(n)$$

where $x(n)$ is the nth sample image. The data entering the IPCA algorithms are the scatter vectors, $u(n) = x(n) - \hat{m}(n), n = 1, 2, \ldots$.

To record intermediate results, we divided the entire data set into 20 subsets. When the data went through the IPCA algorithms, the estimates of the eigenvectors were saved after each subset was passed. In SGA, we used the learning rate suggested in [9, page 54]. Since only the first five γ_i were suggested, we extrapolated them to give $\gamma_6(n) = 46/n$, $\gamma_7(n) = 62/n$, $\gamma_8(n) = 80/n$, $\gamma_9(n) = 100/n$, and, $\gamma_{10}(n) = 130/n$. In GHA, we set $\gamma(n)$ as $1/n$. The amnesic parameter l was set to be 2 in CCIPCA.

The correlation between the estimated unit eigenvector v and the one computed by the batch method v', also normalized, is represented by their inner product $v \cdot v'$. Thus, the larger the correlation, the better. Since $||v - v'|| = 2(1 - v \cdot v')$, $v = v'$ iff $v \cdot v' = 1$. As we can see from Fig. 2, SGA does not seem to converge after fed all images. GHA shows a trend to converge but the estimates are still far from the correct ones. In contrast, the proposed CCIPCA converges fast. Although the higher order eigenvectors converges slower than earlier ones, the 10th one still reaches about 70% with the extremely low sample-to-dimension ratio. We will see below that the 10th principal component represents only 3% of the total data variance. So, 70% correlation with the correct one means that only about 1% of the total data variance is lost.

To examine the convergence of eigenvalues, we use the ratio, $\frac{||v_i||}{\lambda_i}$, the length of the estimated eigenvector divided by the estimate computed by the C Recipe batch method. The results for

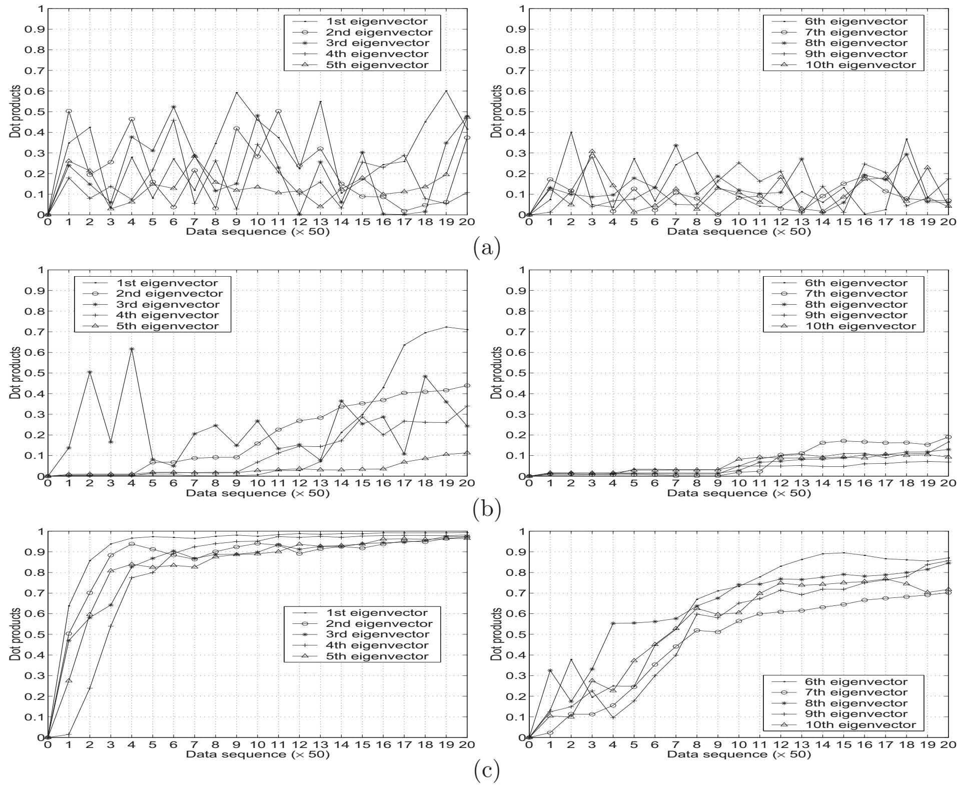

Figure 2: The correctness, or the correlation, represented by dot products, of the first 10 eigenvectors computed by (a) SGA (b) GHA (c) the proposed CCIPCA with the amnesic parameter $l = 2$.

eigenvalues show a similar pattern as in Fig. 2. For conciseness, we only shown the eigenvalue result of the proposed CCIPCA in Fig. 3, together with Fig. 4 showing the first 10 eigenvalues. The ratio between the summation of these 10 eigenvalues and the variance of the data is 58.82%, which means that about 60% of the data variance falls into the subspace spanned by the first 10 eigenvectors.

To demonstrate the effect of amnesic parameter l in Eq. (10), we show the result of eigenvector estimate with $l = 0$. Comparing Fig. 5 with Fig. 2 (c), we can see that the amnesic parameter did help to achieve faster convergence. The amnesic parameter has been made to vary with n in our SAIL robot development program [20], but due to the space limit, the subject is beyond the scope here.

Next, we will show the performance of the algorithm with a much longer data stream. Since the statistics of a real-world image stream may not necessarily be stationary (for example, the mean

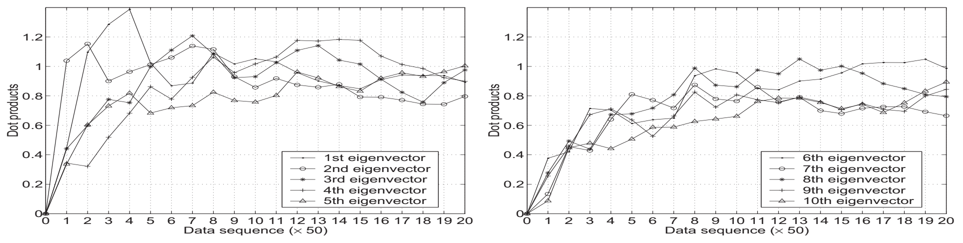

Figure 3: The correctness of the eigenvalue, $\frac{||v_i||}{\lambda_i}$ by CCIPCA.

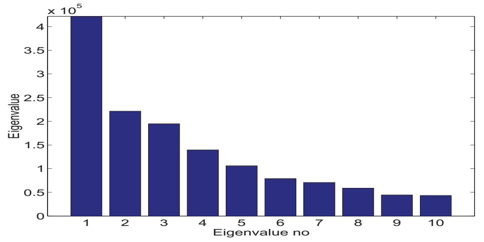

Figure 4: The absolute values of the first 10 eigenvalues.

and variance may change with time), the changing mean and variance make convergence evaluation difficult. To avoid this effect, we simulate a statistically stable long data stream by feeding the images in FERET data set repeatedly into the algorithms. Fig. 6 shows the result after 20 epochs. As expected, all IPCA algorithms converge further while CCIPCA is the quickest.

Shown in Fig. 7 are the first 10 eigenfaces estimated by batch PCA, and CCIPCA (with the amnesic parameter $l = 2$) after one epoch and 20 epochs, respectively. The corresponding eigenfaces computed by the very different methods are very similar.

The average execution time of SGA, GHA, and CCIPCA in each estimation step is shown in Table 1. It is independent of the data. Without doing the GSO procedure, GHA and CCIPCA run

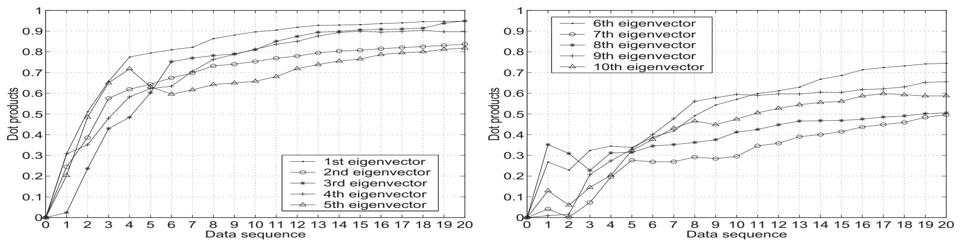

Figure 5: The effect of the amnesic parameter. The correctness of the first 10 eigenvectors computed by CCIPCA, with the amnesic parameter $l = 0$. A comparison with Fig. 2 (c).

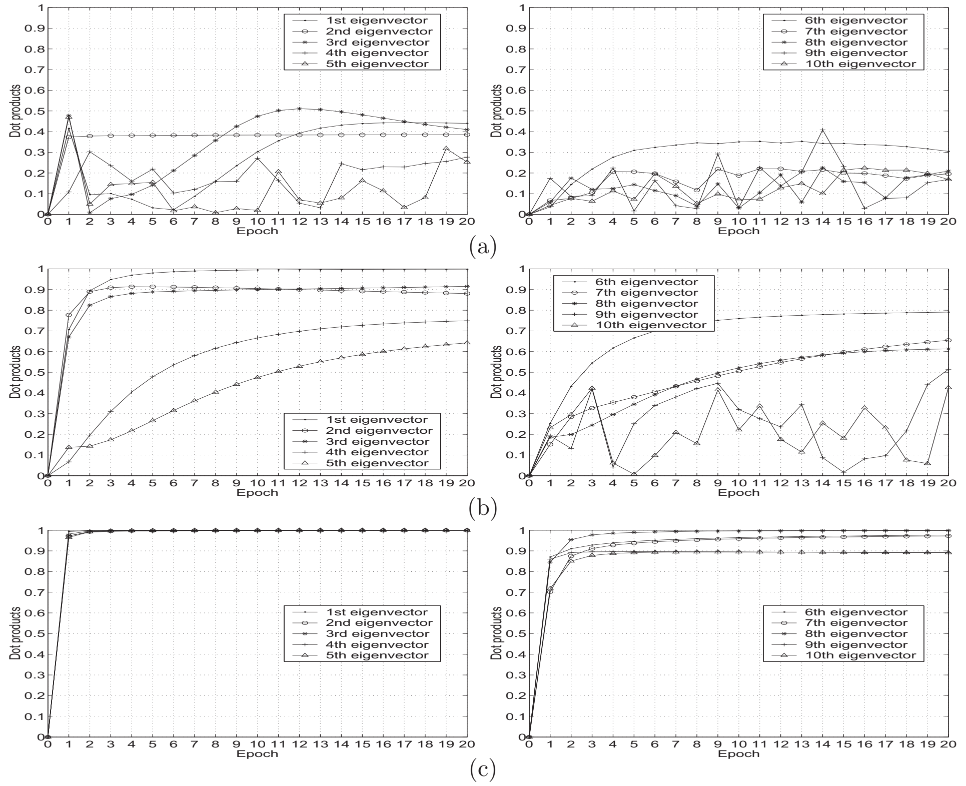

Figure 6: A longer data stream. The correctness of the first 10 eigenvectors computed by (a) SGA (b) GHA (c) CCIPCA (with the amnesic parameter $l = 2$), respectively over 20 epochs.

significantly faster than SGA. CCIPCA has a further computational advantage over GHA because of a saving in normalization.

Table 1: The average execution time for estimating 10 eigenvectors with one new data.

SGA	GHA	CCIPCA
0.42s	0.083s	0.072s

We observed similar efficiency difference for other data sets, such as speech data. For the general readership, an experiment was done on a lower dimension data set. We extracted 10-by-10 pixel subimages around the right eye area in each image of the FERET data set, estimated their sample covariance matrix Σ, and used MATLAB to generate 1000 samples with the Gaussian distribution $N(0, \Sigma)$ in the 100-dimensional space. Thus, the sample-to-dimension ratio is $1000/100 = 10$. The original eye-area subimage sequence is not statistically stationary because the last person's eye-area

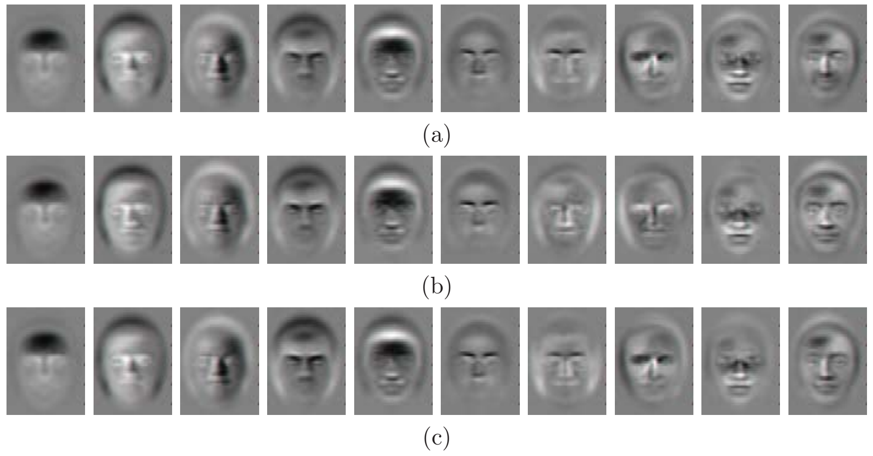

Figure 7: The first 10 eigenfaces obtained by (a) batch PCA, (b) CCIPCA (with amnesic parameter $l = 2$) after one epoch, and (c) CCIPCA (with amnesic parameter $l = 2$) after 20 epochs, shown as images.

image does not necessarily following the distribution defined by the early persons' data. We used the MATLAB-generated data to avoid this non-stationary situation. It turned out all the first ten eigenvectors estimated by CCIPCA reached above 90% correlation with the actual ones.

4 Conclusions and Discussions

This short paper concentrates on a challenging issue of computing dominating eigenvectors and eigenvalues from incrementally arriving high dimensional data stream without computing the corresponding covariance matrix and without knowing data in advance. The proposed CCIPCA algorithm is fast in convergence rate and low in the computational complexity. Our results showed that whether the concept of the efficient estimate is used or not plays a dominating role in convergence speed for high dimensional data. An amnesic average technique is implemented to further improve the convergence rate.

The importance of the result presented here is potentially beyond the apparent technical scope interesting to the computer vision community. As discussed in [7], what a human brain does is not just computing – processing data – but more importantly and more fundamentally, developing the computing engine itself, from real-world, online sensory data streams. Although a lot of studies remain to be done and many open questions are waiting to be answered, the incremental development of a "processor" plays a central role in brain development. The "processor" here is closely related to a procedure widely used now in appearance-based vision: inner product of input scatter vector u with an eigenvector, something that a neuron does before sigmoidal nonlinearity. What is the relationship between IPCA and our brain? A clear answer is not available yet, but Rubner & Schulten [21] proved that the well-known mechanisms of biological Hebbian learning and lateral inhibition between nearby neurons [22] (pages 1020, 376) result in an incremental way of computing PCA. Although we do not claim that the computational steps of the proposed CCIPCA can be found physiologically in the brain, the link between incremental PCA and the developmental mechanisms of our brain is probably more intimate than we can fully appreciate now.

5 Acknowledgements

The work is supported in part by National Science Foundation under grant No. IIS 9815191, and DARPA ETO under contract No. DAAN02-98-C-4025, DARPA ITO under grant No. DABT63-99-1-0014. The authors would like to thank Shaoyun Chen for his codes to do batch PCA.

References

[1] I. Sirovich and M. Kirby, "Low-dimensional procedure for the caracterization of human faces," *Journal of Optical Society of America A*, vol. 4, no. 3, pp. 519–524, March 1987.

[2] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[3] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *Int'l Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, January 1995.

[4] Y. Cui and J. Weng, "Appearance-base hand sign recognition from intensity image sequences," *Computer Vision and Image Understanding*, vol. 78, pp. 157–176, 2000.

[5] S. Chen and J. Weng, "State-based SHOSLIF for indoor visual navigation," *IEEE Trans. Neural Networks*, vol. 11, no. 6, pp. 1300–1314, 2000.

[6] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[7] J. Weng and I. Stockman (eds), *Proceedings of NSF/DARPA Workshop on Development and Learning*, East Lansing, Michigan, April 5-7, 2000.

[8] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction To the Theory of Neural Computation*, Addison-Wesley, Reading, MA, 1991.

[9] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, Letchworth, UK, 1983.

[10] 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 Application*, vol. 106, pp. 69–84, 1985.

[11] T.D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 459–473, 1989.

[12] Y. Zhang and J. Weng, "Convergence analysis of complementary candid incremental principal component analysis," Tech. Rep. MSU-CSE-01-23, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, August 2001.

[13] M. Fisz, *Probability theory and mathematical statistics*, John Wiley & Sons, Inc., New York, third edition, 1963.

[14] J. Weng, T. S. Huang, and N. Ahuja, *Motion and Structure from Image Sequences*, Springer-Verlag, New York, 1993.

[15] N.L. Owsley, "Adaptive data orthogonalization," in *Proc. IEEE Int'l Conf. Acoust., Speech and Signal Processing*, Tulsa, Oklahoma, April 10-12 1978, pp. 109–112.

[16] P.A. Thompson, "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise," in *Proc. 13th Asilomar Conf. on Circuits, System and Computers*, Pacific Grove, CA, 1979, pp. 529–533.

[17] E. Kreyszig, *Advanced engineering mathematics*, Wiley, New York, 1988.

[18] P. J. Phillips, H. Moon, P. Rauss, and S. A. Rizvi, "The FERET evaluation methodology for face-recognition algorithms," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Puerto Rico, June 1997, pp. 137–143.

[19] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, New York, 2nd edition, 1986.

[20] J. Weng, W.S. Hwang, Y. Zhang, C. Yang, and R. Smith, "Developmental humanoids: humanoids that develop skills automatically," in *Proc. The First IEEE-RAS International Conference on Humanoid Robots*, Boston, MA, September 7-8, 2000.

[21] J. Rubner and K. Schulten, "Development of feature detectors by self-organization," *Biological Cybernetics*, vol. 62, pp. 193–199, 1990.

[22] E.R. Kandel, J.H. Schwartz, and T.M. Jessell, Eds., *Principles of Neural Science*, Appleton and Lance, Norwalk, Connecticut, third edition, 1991.