FAST PRINCIPAL COMPONENT ANALYSIS USING EIGENSPACE MERGING

Liang Liu[1], *Yunhong Wang*[2], *Qian Wang*[1], *Tieniu Tan*[1]

[1]National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Computer Science and Engineering
Beihang University, Beijing, China

ABSTRACT

In this paper, we propose a fast algorithm for Principal Component Analysis (PCA) dealing with large high-dimensional data sets. A large data set is firstly divided into several small data sets. Then, the traditional PCA method is applied on each small data set and several eigenspace models are obtained, where each eigenspace model is computed from a small data set. At last, these eigenspace models are merged into one eigenspace model which contains the PCA result of the original data set. Experiments on the FERET data set show that this algorithm is much faster than the traditional PCA method, while the principal components and the reconstruction errors are almost the same as that given by the traditional method.

Index Terms— principal component analysis, eigenspace merging.

1. INTRODUCTION

PCA (Principal Component Analysis) is widely used in dimension reduction, feature extraction, image compression, etc. In 1990s, PCA was used in face recognition and made profound influence in this field. The problem of PCA can be formulated as follows.

For an $m \times n$ matrix $\mathbf{D}$, each column can be viewed as a point in m-dimensional linear space. The task of PCA is to find the center of the n points and c principal orthonormal vectors which expand an eigenspace[1]. In many applications, c is much smaller than both m and n.

For the traditional PCA method [1], the time complexity[2] is $O(mn \cdot \min(m,n)/2)$ and the space complexity is $O(mn)$. When m and n are very large, both the time complexity and the space complexity can be prohibitive in practice. In this paper, a fast algorithm with a possible loss of precision is proposed. For this algorithm, the time complexity is $O((\sqrt{6}+1)cmn)$ and the space complexity is $O(\sqrt{6}cm)$. These make the task much easier.

The proposed method can be viewed as an application of eigenspace merging [2, 3, 4]. The algorithm of eigenspace merging was originally used to merge two eigenspaces without storing covariance matrix or original data. In this paper, we shall show that eigenspace merging can be used to design a fast algorithm for PCA. We will also analyze the error bound introduced by the proposed algorithm.

The remainder of this paper is organized as follows. In Section 2, a fast algorithm for PCA is proposed and discussed in detail. In Section 3, some experimental results are presented. Conclusions are drawn in Section 4.

[1]An eigenspace is an affine subspace of the original m-dimensional space.
[2]For simplicity, assume that $m \gg n$ or $m \ll n$.

2. FAST PCA USING EIGENSPACE MERGING

In Section 2.1, we give a description of eigenspace model. Section 2.2 gives a brief introduction about eigenspace merging. In Section 2.3, we will present the algorithm of Fast PCA in detail. In Section 2.4, we analyze the error bound of the proposed algorithm.

2.1. Eigenspace model description

An eigenspace model is a structure which contains four parameters, namely $\Omega = (\overline{\mathbf{x}}, \mathbf{U}, \mathbf{\Lambda}, N)$ [5], where $\overline{\mathbf{x}}$ is the center of the eigenspace, and $\mathbf{U}$ is a matrix whose columns are orthonormal bases of the eigenspace, namely eigenvectors. $\mathbf{\Lambda}$ is a diagonal matrix whose elements along the diagonal are variances for each principal axis, namely eigenvalues, and N is the number of samples to construct this eigenspace.

In Section 2.2, we shall see that this model is quite convenient for eigenspace merging.

2.2. A brief introduction about eigenspace merging

Skarbek [2] developed an algorithm to compute eigenspace merging which is more concise than Hall's method [3]. Both methods need not store the covariance matrix of previous training samples. Given two eigenspace models Ω_1 and Ω_2, eigenspace merging is used to find the eigenspace model Ω for the union of the original data sets assuming that the original data is not available.

If there are q_1 and q_2 eigenvectors in Ω_1 and Ω_2 respectively and we keep c eigenvectors in Ω, the time complexity

of eigenspace merging is $O(q_1q_2m + (q_1 + q_2 + 1)cm)$ using the computational trick in [4], where m is the dimension of the original feature space.

2.3. A fast algorithm for PCA

Given an $m \times n$ matrix $\mathbf{D}$, we want to compute c principal components of the column vectors in $\mathbf{D}$. To accomplish this task, the time complexity of the traditional PCA method [1] is $O(mn^2/2)$ when $m \gg n$.

Our method is shown in Fig. 1. The matrix $\mathbf{D}$ is firstly divided into g small matrix $\mathbf{D}_1$, $\mathbf{D}_2$, $\cdots$, $\mathbf{D}_g$. The number of columns in $\mathbf{D}_i (i = 1, 2, \cdots, g)$ is at most k. Then the traditional PCA method is applied on each small matrix $\mathbf{D}_i$ and we can obtain g eigenspace models $\Omega_i (i = 1, 2, \cdots, g)$ corresponding to $\mathbf{D}_i (i = 1, 2, \cdots, g)$ respectively, with each eigenspace model containing c eigenvectors. We merge these eigenspace models into one eigenspace model using binary tree structure, while making the tree as short as possible (Fig. 1). The final eigenspace model contains c principal components of the original data set.

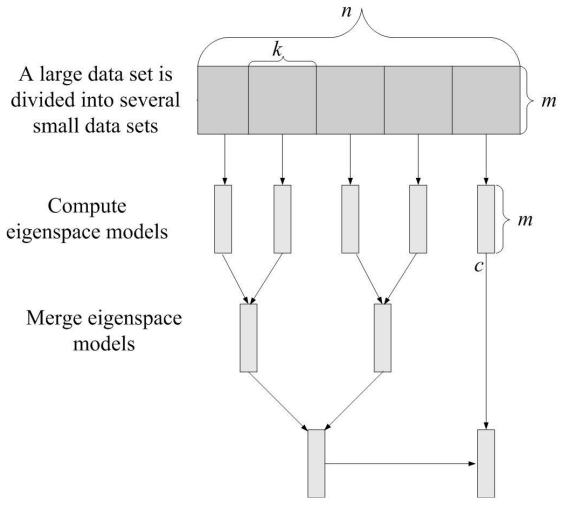

Fig. 1. An illustration of the proposed algorithm. A large data set is firstly divided into several small data sets. Then the traditional PCA method is applied on each small data set and we can get several eigenspace models. These eigenspace models are merged into one eigenspace model which contains the principal components of the original data set.

One critical problem is how to choose k optimally. This needs some careful analysis on the computational time. To compute c principal components of $\mathbf{D}_i$, the computational time is about $O(k^2m/2 + ckm)$. To merge two eigenspace models where each model contains c eigenvectors, it takes time $O(c^2m + (2c + 1)cm)$ (cf. Section 2.2). The total computational time of our method is about

$$T(n,k) = \left\lceil \frac{n}{k} \right\rceil (k^2m/2 + ckm) + (\left\lceil \frac{n}{k} \right\rceil - 1)(3c^2m + cm)$$
$$\approx mn(k/2 + 3c^2/k + c). \quad (1)$$

Solve $\frac{\partial T(n,k)}{\partial k} = 0$ and we can get

$$k = \sqrt{6}c. \quad (2)$$

Notice that k should be an integer. Moreover, a larger k helps to reduce the accumulated error. So we choose $k = \lceil \sqrt{6}c \rceil$ and get $T(n) \approx (\sqrt{6} + 1)cmn$. Thus, the time complexity is $O((\sqrt{6} + 1)cmn)$. The algorithm of Fast PCA is summarized as follows.

Algorithm **Fast PCA**
Input:
$\mathbf{D}$: an $m \times n$ matrix which contains data samples as its columns.
c: the number of principal components we want to compute.
Output:
$\Omega = (\overline{\mathbf{x}}, \mathbf{U}, \mathbf{\Lambda}, N)$: the eigenspace model of $\mathbf{D}$, which contains c principal components.
Method:
1. $k \leftarrow \lceil \sqrt{6}c \rceil$
2. $g \leftarrow \lceil \frac{n}{k} \rceil$, $r \leftarrow n - (g-1)k$
3. $\Omega_1 \leftarrow$ TPCA$(\mathbf{D}, 1, r, c)$
4. **for** $i = 1$ to $g - 1$
5. $\quad \Omega_{i+1} \leftarrow$ TPCA$(\mathbf{D}, r + (i-1)k + 1, r + ik, c)$
6. **end for**
7. **while** $g > 1$ **do**
8. $\quad$ **for** $i = 1$ to $\lfloor g/2 \rfloor$
9. $\quad\quad \Omega_i \leftarrow$ MergeEigenspaces$(\Omega_{2i-1}, \Omega_{2i}, c)$
10. $\quad$ **end for**
11. $\quad$ **if** g is odd
12. $\quad\quad \Omega_{(g+1)/2} \leftarrow \Omega_g$
13. $\quad$ **end if**
14. $\quad g \leftarrow \lceil g/2 \rceil$
15. **end while**
16. $\Omega \leftarrow \Omega_1$

In Line 3 and Line 5, TPCA$(\mathbf{D}, a, b, c)$ applies the traditional PCA method on a submatrix of $\mathbf{D}$ which contains $b - a + 1$ columns from the ath column to the bth column in $\mathbf{D}$, and keeps c eigenvectors in the output eigenspace model. In Line 9, MergeEigenspaces$(\Omega_{2i-1}, \Omega_{2i}, c)$ applies eigenspace merging method (Section 2.2) on Ω_{2i-1} and Ω_{2i}, and keeps c eigenvectors in the output eigenspace model Ω_i.

For this algorithm, the data can be processed "block" by "block". The size of each "block" is at most $\sqrt{6}cm$ in Line 3 and Line 5. The merging operation in Line 9 deals with a data size of $2cm$, which is less than $\sqrt{6}cm$. So the space complexity is $O(\sqrt{6}cm)$.

Another important problem is when it is suitable to use Fast PCA instead of the traditional method. This need a comparison of the running time, namely

$$(\sqrt{6}+1)cmn < mn \cdot \min(m,n)/2. \quad (3)$$

Hence, we can get the range of c when Fast PCA is faster than the traditional PCA method

$$c < \frac{\min(m,n)}{2(\sqrt{6}+1)}. \quad (4)$$

By the algorithm of eigenspace merging [2], the c eigenvectors in Ω obtained in Line 16 are guaranteed to be orthonormal. Moreover, the mean $\overline{\mathbf{x}}$ in Ω is also guaranteed to be the same as the output of the traditional PCA method.

2.4. Error bound analysis

Assuming that the modeling errors of Ω_1 and Ω_2 are ϵ_a and ϵ_b respectively, if the error introduced by eigenspace merging is ϵ_1, the average error for the projected data in the merged model is bounded by [2]

$$\epsilon \leq 2(\alpha_1\epsilon_a + \alpha_2\epsilon_b + \epsilon_1). \quad (5)$$

For simplicity, assume that the modeling error introduced by MergeEigenspace$(\Omega_{2i-1}, \Omega_{2i}, c)$ is ϵ_1 and the error induced by TPCA$(\mathbf{D}, a, b, c)$ is ϵ_2. Then the error bound of Fast PCA can be approximately computed as follows.

$$\begin{aligned}\epsilon(n) &\leq 2\left[\alpha_1\epsilon(2^{\lfloor\log_2 g\rfloor}k) + \alpha_2\epsilon(n - 2^{\lfloor\log_2 g\rfloor}k) + \epsilon_1\right] \\ &\leq 2\left[\epsilon(2^{\lfloor\log_2 g\rfloor}k) + \epsilon_1\right] \\ &< \frac{2n}{\sqrt{6}c}(2\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2). \quad (6)\end{aligned}$$

This is the upper bound of the average error. In many applications, the error can be much lower than this error bound because the samples are often similar to each other in a data set (e.g. face image data set, as we will see in Section 3).

If we merge the eigenspace models sequentially (FPCA2 method in Section 3), the error bound will be

$$\begin{aligned}\epsilon'(n) &\leq 2\left[\frac{n-k}{n}\epsilon'(n-k) + \frac{k}{n}\epsilon'(k) + \epsilon_1\right] \\ &< 2^{n/(\sqrt{6}c)} \cdot \frac{3(\sqrt{6}c+1)}{n}(2\epsilon_1 + \epsilon_2). \quad (7)\end{aligned}$$

We can see that the upper bound of $\epsilon'(n)$ is exponential in n, which is much worse than that of $\epsilon(n)$. This is the reason why we merge the eigenspace models using a shortest binary tree structure instead of merging them sequentially.

If we want to improve the precision of Fast PCA while we cannot afford the computational complexity of the traditional method, we can choose a larger c (cf. Formula (5)) in Fast PCA to balance the precision and computational complexity.

Fig. 2. The first 8 images in the data set.

Table 1. Comparison of the running time of TPCA, IPCA, FPCA2 and FPCA. (s)

Method / c	TPCA [1]	IPCA [5]	FPCA2	FPCA
1	4.88	1.00	0.99	0.97
2	4.91	1.59	1.52	1.44
4	4.91	4.31	1.63	1.61
8	4.95	8.73	1.80	1.80
16	5.02	18.33	2.19	2.23
32	5.17	27.34	3.08	3.20
64	5.48	90.63	4.77	4.73

3. EXPERIMENTS

In order to evaluate the effectiveness of the proposed algorithm, we have conducted some experiments on a subset of the FERET data set. This subset contains 540 face images from 270 subjects with each subject corresponding to 2 face images. All images contain frontal faces with different light conditions. All images are gray-level images with the same size of 142×120. The first 8 images are shown in Fig. 2.

In our experiments, we firstly transform each image into a vector by concatenating all columns in the image. In this way, we get a 17040×540 matrix with each column representing a face image. All experiments are done on this matrix. Apart from the proposed method and the traditional method, we also do experiments using some other methods. The methods we used in our experiments are listed as follows.

-**TPCA** [1], the traditional PCA method.

-**IPCA** [5], namely Incremental PCA, this method can be viewed as a special case of FPCA2 by choosing the size of each "block" as one, but keeping at most c eigenvectors.

-**FPCA2**, a variation of Fast PCA. In this method, the eigenspace models are merged sequentially. In other words, we merge $\Omega_2, \Omega_3, \cdots, \Omega_g$ into Ω_1 one by one.

-**FPCA**, namely Fast PCA, the proposed method (Section 2.3).

We have done experiments using these four methods for $c = 1, 2, 4, 8, \cdots, 64$. Table 1 shows the running time of each method. We can see that the running time of the proposed method and FPCA2 are almost the same. They are both faster than the other two methods when $1 \leq c \leq 64$. In theory, the time complexity and space complexity of the four methods are listed in Table 2.

Table 2. Comparison of time complexity and space complexity of TPCA, IPCA, FPCA2 and FPCA.

Method	Time complexity	Space complexity
TPCA	$mn \cdot \min(m,n)/2$	mn
IPCA	c^2mn	cm
FPCA2	$(\sqrt{6}+1)cmn$	$\sqrt{6}cm$
FPCA	$(\sqrt{6}+1)cmn$	$\sqrt{6}cm$

Table 3. Comparison of the reconstruction errors of TPCA, IPCA, FPCA2 and FPCA.

c \ Method	TPCA	IPCA	FPCA2	FPCA
1	30.719	30.720	30.721	30.732
2	27.955	28.098	28.267	27.994
4	24.983	25.054	25.045	25.064
8	21.710	21.831	21.805	21.803
16	18.294	18.418	18.421	18.401
32	14.837	14.989	14.936	14.918
64	11.352	11.489	11.420	11.402

We have also compared the reconstruction errors of these four methods. The reconstruction error is defined as

$$e = \sqrt{\frac{1}{mn}\left\|\mathbf{D} - \left[\mathbf{U}\mathbf{U}^T\left(\mathbf{D}-\overline{\mathbf{D}}\right)+\overline{\mathbf{D}}\right]\right\|_F^2}, \qquad (8)$$

where $\overline{\mathbf{D}}$ is an $m \times n$ matrix with each column containing the mean vector of all columns in $\mathbf{D}$, and $\mathbf{U}$ is an $m \times c$ matrix with each column containing a principal component. $\|\cdot\|_F$ is the Frobenius norm. Table 3 shows the reconstruction errors of the four methods. We can see that the reconstruction errors are quite close for different methods. When $c = 2, 8, 16, 32, 64$, the reconstruction errors given by FPCA are closer to TPCA than those given by IPCA and FPCA2. This shows that FPCA is superior to IPCA and FPCA2 in precision.

Fig. 3 shows 8 principal components produced by the four methods for $c = 8$. The four rows in Fig. 4 (from top to bottom) are produced by TPCA, IPCA, FPCA2 and FPCA respectively. It is very clear that the results given by FPCA is the closest to that given by TPCA.

4. CONCLUSIONS

In this paper, we have presented a fast algorithm for PCA using eigenspace merging. The time complexity of this algorithm is $O((\sqrt{6}+1)cmn)$, if we want to compute c principal components from an $m \times n$ matrix which contains data samples as its columns. The space complexity of the proposed algorithm is $O(\sqrt{6}cm)$, which is not related to n. This makes the computation much easier for large data set. Although there may be some loss of precision in theory, experiments show that the proposed algorithm produces almost the same principal components as that given by the traditional method.

Fig. 3. 8 principal components produced by TPCA, IPCA, FPCA2 and FPCA respectively (from the top row to the bottom row) for $c = 8$. Pay attention to the first component produced by each method.

5. ACKNOWLEDGEMENT

This work was supported by Program of New Century Excellent Talents in University, National Natural Science Foundation of China (No. 60575003, 60332010, 60335010, 60121302, 60275003, 69825105, 60605008), Joint Project supported by National Science Foundation of China and Royal Society of UK (60710059), the National Basic Research Program (Grant No.2004CB318110), Hi-Tech Research and Development Program of China (2006AA01Z133, 2006AA01Z193) and the Chinese Academy of Sciences.

6. REFERENCES

[1] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, March 1991.

[2] W. Skarbek, "Merging subspace models for face recognition," *In proceedings of the CAIP*, pp. 606–613, 2003.

[3] P. M. Hall, D. Marshall, and R. R. Martin, "Merging and splitting eigenspace models," *IEEE Transacetions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 1042–1049, 2000.

[4] A. Franco, A. Lumini, and D. Maio, "Eigenspace merging for model updating," *The 16th International Conference on Pattern Recognition*, vol. 2, pp. 156–159, 2002.

[5] P. M. Hall, D. Marshall, and R. R. Martin, "Incremental eigenanalysis for classification," *In The British Machine Vision Conference*, pp. 286–295, 1998.