International Journal of Internet, Broadcasting and Communication Vol.9 No.1 9-17 (2017)
https://doi.org/10.7236/IJIBC.2017.9.1.9

IJIBC 17-1-2

Combining Empirical Feature Map and Conjugate Least Squares Support Vector Machine for Real Time Image Recognition : Research with Jade Solution Company

Byung Joo Kim

Department of Computer Engineering, Youngsan University
bjkim@ysu.ac.kr

Abstract

This paper describes a process of developing commercial real time image recognition system with company. In this paper we will make a system that is combining an empirical kernel map method and conjugate least squares support vector machine in order to represent images in a low-dimensional subspace for real time image recognition. In the traditional approach calculating these eigenspace models, known as traditional PCA method, model must capture all the images needed to build the internal representation. Updating of the existing eigenspace is only possible when all the images must be kept in order to update the eigenspace, requiring a lot of storage capability. Proposed method allows discarding the acquired images immediately after the update. By experimental results we can show that empirical kernel map has similar accuracy compare to traditional batch way eigenspace method and more efficient in memory requirement than traditional one. This experimental result shows that proposed model is suitable for commercial real time image recognition system.

Keywords: *Empirical Feature Map, Conjugate LS-SVM, Image Recognition, Kin_ship DB*

1. Introduction

Unsupervised surveillance gadgets aided by hi-tech visual information retrieval and indexing systems use computerized face recognition techniques that can recognizes faces from an image. There are two main approaches for face recognition [1]. The first approach is the feature based matching approach using the relationship between facial features[2]. The second approach is the template matching approach using the holistic features of the face images. Template based techniques often follow the subspace method called eigenface originated by Turk and Pentland[3]. This technique is based on the Karhunen-Loeve transformation, which is also referred as PCA. It has gained great success and become a de facto standard and a common performance benchmark in face recognition. One of the attractive characteristics of PCA is

Manuscript Received: Dec. 10, 2016 / Revised: Dec. 18, 2016 / Accepted: Jan. 13, 2017
Corresponding Author: bjkim@ysu.ac.kr
Tel: +82-055-380-9447, Fax: +82-055-380-9447
Department of Computer Engineering, Youngsan University

that a high dimension vector can be represented by a small number of orthogonal basis vectors. The conventional methods of PCA such as singular value decomposion(SVD) and eigen-decomposition, perform in batch-mode with a computational complexity of O (m^3) when m is the minimum value between the data dimension and the number of training examples. Undoubtedly these methods are computationally expensive when dealing with large scale problems where both the dimension and the number of training examples are large. To address this problem, many researchers have been working on incremental algorithms. Among them Chandrasekaran et al presented an incremental eigenspace update method using SVD[4]. Hall et al derived an eigen-decomposition based incremental algorithm and later extended their work to merge and split eigenspace models[5]. Another problem of PCA is that it only defines a linear projection of the data, the scope of its application is necessarily somewhat limited. It has been shown that most of the data in the real world are inherently non-symmetric and therefore contain higher-order correlation in-formation that could be useful[6]. PCA is incapable of representing such data. For such cases, nonlinear transforms is necessary. Recently kernel trick has been applied to PCA and is based on a formulation of PCA in terms of the dot product matrix instead of the covariance matrix[7]. Kernel PCA(KPCA), however, requires storing and finding the eigenvectors of a $N \times N$ kernel matrix where N is a number of patterns. It is infeasible method for when N is large. This fact has motivated the development of empirical kernel method which does not store the kernel matrix. In this paper we propose a method that allows for incremental eigenspace update method by incremental kernel PCA for vision learning and recognition. Paper is organized as follows. In Section 2 we will briefly explain the incremental PCA method. In Section 3 KPCA is introduced and to make KPCA incrementally, empirical kernel map method is explained. Experimental results to evaluate the performance of proposed method is shown in Section 4. Discussion of proposed method and future work is described in Section 5.

2. Incremental PCA

In this section, we will give a brief introduction to the method of incremental PCA algorithm which overcomes the computational complexity of standard PCA. Before continuing, a note on notation is in order. Vectors are columns, and the size of a vector, or matrix, where it is important, is denoted with subscripts. Particular column vectors within a matrix are denoted with a superscript, while a superscript on a vector denotes a particular observation from a set of observations, so we treat observations as column vectors of a matrix. As an example,

A^i_{mn} is the ith column vector in a $m \times n$ matrix. We denote a column extension to a matrix using square brackets. Thus $[A_{mn} b]$ is an(m × (n + 1)) matrix, with vector b appended to A_{mn} as a last column.

To explain the incremental PCA, we assume that we have already built a set of eigenvectors $U = [u_j, j = 1, \cdots, k]$ after having trained the input images $x_i, i =, \cdots, N$. The corresponding eigenvalues are Λ and $\overline{\mathrm{X}}$ is the mean of input image. Incremental building of eigenspace requires updating these eigenspace to take into account of a new input image. Here we give a brief summarization of the method which is described in [5]. First, we update the mean:

$$\overline{x'} = \frac{1}{N+1}(N\bar{x} + x_{N+1}) \tag{1}$$

We then update the set of eigenvectors to reflect the new input image and to apply a rotational transformation to U . For doing this, it is necessary to compute the orthogonal residual vector $\hat{h} = (Ua_{N+1} + \bar{x}) - x_{N+1}$ where

a_{N+1} is principal component and normalize it to obtain $h_{N+1} = \frac{h_{N+1}}{\|h_{N+1}\|_2}$ for $\|h_{N+1}\|_2 \rangle 0$ and $h_{N+1} = 0$ otherwise. We obtain the new matrix of eigenvectors U' by appending h_{N+1} to the eigenvectors U and rotating them :

$$U' = [U, h_{N+1}]R \tag{2}$$

where R$\in \Re_{(k+1)\times(k+1)}$ is a rotation matrix. R is the solution of the eigenspace of the following form:

$$DR = R\Lambda' \tag{3}$$

where Λ' is a diagonal matrix of new eigenvalues. We compose D $\in \Re_{(k+1)\times(k+1)}$ as:

$$D = \frac{N}{N+1}\begin{bmatrix} \Lambda & 0 \\ 0^T & 0 \end{bmatrix} + \frac{N}{(N+1)^2}\begin{bmatrix} aa^T & \gamma a \\ \gamma a^T & \gamma^2 \end{bmatrix} \tag{4}$$

where $\gamma = h_{N+1}^T(x_{N+1} - \bar{x})$ and $a = U^T(x_{N+1} - \bar{x})$. Though there are other ways to construct matrix D [4][5], the only method ,however, described in [6] allows for the updating of mean.

2.1 Updating Image Representations

The incremental PCA represents the input image with principal components $a_{i(N)}$ and it can be approximated as follows:

$$x_{i(N)} = \hat{U}a_{i(N)} + \bar{x} \tag{5}$$

To update the principal components $a_{i(N)}$ for a new image x_{N+1}, computing an auxiliary vector η is necessary. η is calculated as follows:

$$\eta = \left[U\ \hat{h}_{N+1} \right]^T (\bar{x} - \bar{x}') \tag{6}$$

then the computation of all principal components is

$$a_{i(N+1)}(R')^T \begin{bmatrix} a_{i(N)} \\ 0 \end{bmatrix} + \eta, \quad i = 1, \cdots, N+1 \tag{7}$$

The transformations described above yield a model that represents the input images with the same accuracy as the previous one, therefore we can now discard the old subspace and the coefficients that represent the image in it. x_{N+1} is represented accurately as well, so we can safely discard it. The representation of all N + 1 images are possible because the subspace is spanned by k +1eigenvector. Due to the increase of the dimensionality by one, however, more storage is required to represent the data. If we try to keep a k-dimensional eigenspace, we lose a certain amount of information. In order to balance the storage

requirements with the level of accuracy, it is needed for us to set the criterion on retaining the number of eigenvectors. There is no explicit guideline for retaining a number of eigenvectors.
In this paper we set our criterion on adding an eigenvector as $\lambda'_{K+1} \rangle 0.7\bar{\lambda}$ where $\bar{\lambda}$ is a mean of the λ. Based on this rule, we decide whether adding u'_{K+1} or not.

3. Empirical Feature Map

A prerequisite of the incremental eigenspace update method is that it has to be applied on the data set. Furthermore incremental PCA builds the subspace of eigenvectors incrementally, it is restricted to apply the linear data. But in the case of KPCA this data set $\Phi(x^N)$ is high dimensional and most of the time can not even be calculated explicit ly. For the case of nonlinear data set, applying feature mapping function method to incremental PCA may be one of the solutions. This is performed by so-called kernel-trick , which means an implicit embedding to an infinite dimensional Hilbert space[9](i.e. feature space) F .

$$K(x, y) = \Phi(x) \cdot \Phi(y) \tag{8}$$

Where K is a given kernel function in an input space. When K is semi positive definite, the existence of Φ is proven[7]. Most of the case, however, the mapping Φ is high-dimensional and cannot be obtained explicitly. The vector in the feature space is not observable and only the inner product between vectors can be observed via a kernel function. However, for a given data set, it is possible to approximate Φ by empirical kernel map proposed by Scholkopf[10] and Tsuda[11] which is defined as $\Psi_N : \Re^d \to \Re^N$

$$\Psi_N(x) = [\Phi(x_1)\cdot\Phi(x), \cdots, \Phi(x_N)\cdot\Phi(x)]^T = [K(x_1, x), \cdots, K(x_N, x)]^T \tag{9}$$

A performance evaluation of empirical kernel map was shown by Tsuda. He shows that support vector machine with an empirical kernel map is identical with the conventional kernel map[12]. The empirical kernel map $\Psi_N(x_N)$, however, do not form an orthonormal basis in $\Re^N$, the dot product in this space is not the ordinary dot product. In the case of KPCA , however, we can be ignored as the following argument. The idea is that we have to perform linear PCA on the $\Psi_N(x_N)$ from the empirical kernel map and thus diagonalize its covariance matrix. Let the N × N matrix $\Psi = [\Psi_N(x_1), \Psi_N(x_2), \cdots, \Psi_N(x_N)]$, then from equation (9) and definition of the kernel matrix we can construct Ψ = NK. The covariance matrix of the empirically mapped data is:

$$C_\Psi = \frac{1}{N}\Psi\Psi^T = NKK^T = NK^2 \tag{10}$$

In case of empirical kernel map, we diagonalize NK^2 instead of K as in KPCA. Mika shows that the two matrices have the same eigenvectors $\{u_k\}$ [12]. The eigenvalues $\{\lambda_k\}$ of K are related to the eigenvalues $\{K_k\}$ of NK^2 by

$$\lambda_k = \sqrt{\frac{K_k}{N}} \tag{11}$$

and as before we can normalize the eigenvectors $\{v_k\}$ for the covariance matrix C_Ψ of the data by dividing each $\{u_k\}$ by $\sqrt{\lambda_k N}$. Instead of actually diagonalize the covariance matrix C_Ψ , the incremental KPCA is applied directly on the mapped data Ψ = NK. This makes it easy for us to adapt the incremental eigenspace update method to KPCA such that it is also correctly takes into account the centering of the mapped data in an incremental way. By this result, we only need to apply the empirical map to one data point at a time and do not need to store the N × N kernel matrix.

4. Experiment

To evaluate the performance of accuracy on eiegnspace update for incremental data we take nonlinear data. The disadvantage of incremental method is their accuracy compared to batch method even though it has the advantage of memory efficiency. So we shall apply proposed method to a simple toy data which will show the accuracy and memory efficiency of incremental KPCA compared to APEX model proposed by Kung[13] and batch KPCA. Next we will use images from the KinFaceW-I data).

4.1 Toy Data

To evaluate the eigenspace update accuracy and memory efficiency of incremental KPCA compared to APEX and KPCA we take nonlinear data used by Scholkoff[8]. Totally 41 training data set is generated by:

$$y = x^2 + 0.2\varepsilon : \varepsilon \quad \mathit{from} \quad N(0,1), x = [-1,1] \tag{12}$$

First we compare feature extraction ability of incremental KPCA to APEX model. APEX model is famous principal component extractor based on Hebbian learning rule. Applying toy data to incremental KPCA we finally obtain 2 eigenvectors. To evaluate the performance of two methods on same condition, we set 2 output nodes to standard APEX model.

In table 1 we experimented APEX method on various conditions. Generally neural network based learning model has difficulty in determining the parameters; for example learning rate, initial weight value and optimal hidden layer node. This makes us to conduct experiments on various conditions. $\|\omega\|$ is norm of weight vector in APEX and $\|\omega\|$=1 means that it converges stable minimum. cosθ is angle between eigenvector of KPCA and APEX, incremental

Table 1. Performance evaluation of incremental KPCA(IKPCA) and APEX

Method	Iteration	Learning Rate	$\|\omega_1\|$	$\|\omega_2\|$	$\cos\theta_1$	$\cos\theta_2$	MSE
APEX	50	0.01	0.6827	1.4346	0.9993	0.7084	14.8589
APEX	50	0.05				Do not converge	
APEX	500	0.01	1.0068	1.0014	0.9995	0.9970	4.4403
APEX	500	0.05	1.0152	1.0470	0.9861	0.9432	4.6340
APEX	1000	0.01	1.0068	1.0014	0.9995	0.9970	4.4403
APEX	1000	0.05	1.0152	1.0470	0.9861	0.9432	4.6340
IKPCA	100		1	1	1	1	0.0223

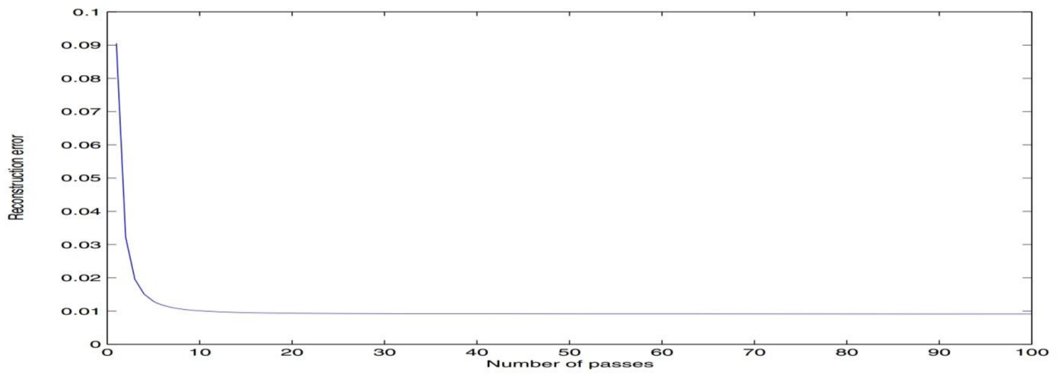

Figure 1. Reconstruction error change by re-learning in incremental KPCA

KPCA respectively. cosθ of eigenvector can be a factor of evaluating accuracy how much incremental KPCA and APEX is close to accuracy of KPCA. Table 1 nicely shows the two advantages of incremental KPCA compared to APEX: first, performance of incremental KPCA is better than APEX; second, the performance of incremental KPCA is easily improved by re-learning. Another factor of evaluating accuracy is reconstruction error. Reconstruction error is defined as the squared distance between the Ψ image of x_N and reconstruction when projected onto the first principal components.

$$\delta = \left|\Psi(x_N) - P_l\Psi(x_N)\right|^2 \tag{13}$$

In here P_l is the first principal component. The MSE(Mean Square Error) value of reconstruction error in APEX is 4.4403 whereas incremental KPCA is 0.0223. This means that the accuracy of incremental KPCA is superior to standard APEX and similar to that of batch KPCA. Figure 1 shows the MSE value change for reconstruction error by re-learning in incremental KPCA. Re-learning is similar meaning of epoch in neural network learning. We can see that the performance of incremental KPCA is easily improved by relearning. Above results of simple toy problem indicate that incremental KPCA is comparable to the batch way KPCA and superior in terms of accuracy. Next we will compare the memory efficiency of incremental KPCA compared to KPCA. In these experiments, incremental KPCA only needs D matrix and R matrix whereas KPCA needs kernel matrix. Table 2 shows the memory requirement of each method. Memory requirement of standard KPCA is 93 times more than incremental KPCA. We can see that incremental KPCA is more efficient in memory requirement than KPCA and has similar ability of eigenspace update accuracy.

Table 2. Memory efficiency of incremental KPCA compared to KPCA on toy data

	KPCA	IKPCA
Kernel matrix	14 X 41	None
R matrix	None	3 X 3
D matrix	None	3 X 3
Efficiency ratio	93.3889	1

By this simple toy problem we can show that incremental KPCA has similar accuracy compare to KPCA and more efficient in memory requirement than KPCA.

4.2 LS-SVM for Large Size Data

Support vector machines(SVM) developed by Vapnik[7] and it is a powerful methodology for solving problems in nonlinear classification. Originally, it has been introduced within the context of statistical learning theory and structural risk minimization. In the methods one solves convex optimization problems, typically by quadratic programming(QP). Solving QP problem requires complicated computational effort and need more memory requirement. LS-SVM[8] overcomes this problem by solving a set of linear equations in the problem formulation. LS-SVM method is computationally attractive and easier to extend than SVM. But traditional batch way LS-SVM requires storing (N+1) × (N+1) matrix where N is a number of patterns. It is infeasible method when dealing with image data because its size is big. For image data sets the use of iterative methods is recommended. In principle, various methods can be used at this point including SOR(Successive Over-Relaxation), CG(Conjugate Gradient), GMRES(Generalized Minimal Residual) etc. However, not all of these iterative methods can be applied to any kind of linear system. For example, in order to apply CG the matrix should be positive definite. Due to the presence of the *b* bias term in the LS-SVM model the resulting matrix is not positive definite. So before we can apply such methods we have to transform the linear system into a positive definite system. The LS-SVM KKT system is of the form

$$\begin{bmatrix} 0 & Y^T \\ Y & H \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{14}$$

more specifically with H = Ω+ I/γ, $\xi_1 = b, \xi_2 = \alpha,\ d_1 = 0, d_2 = I_v$. This can be transformed into

$$\begin{bmatrix} s & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 + H^{-1}y\xi_1 \end{bmatrix} = \begin{bmatrix} -d_1 + y^T H^{-1} d_2 \\ d_2 \end{bmatrix} \tag{15}$$

with S = $y^T H^{-1} y > 0$ ($H = H^{-T} > 0$). Because *s* is positive and *H* positive definite the overall matrix is positive definite. This form is very suitable because different kinds of iterative methods can be applied to problems involving positive definite matrices. This leads to the LS-SVM classifier with conjugate gradient algorithm LS-SVM for big data is as follows.

```
1. Solve $\eta,\nu$ from $H\eta = Y$ and
   $H\nu = 1_v$
2. Compute $s = Y^T \eta$
3. Find solution
   $b = \eta^T 1_v / s$
   $\alpha = \nu - \eta b$
```

4.3 The KinFaceW-I Face Data Set

To validate the above results on a widely used pattern recognition benchmark database, we use the KinFaceW-I data set[9]. There are 156, 134, 116, and 127 pairs of kinship images for these four relations. And there are four folders: father-daughter, father-son, mother-daughter and mother-son, representing four different kinship relations: Father-Daughter (FD), Father-Son (FS), Mother-Daughter (MD), and Mother-Son (MS). Among them we use 156 Father-Daughter images. 70% of image is used for training and rest of 30% is used for testing. Figure 2 shows the sample images of training data. A RBF kernel has been taken with $\sigma_1^2 = 67.416,\ \ \sigma_2^2 = 56.351,$ and are obtained by 10-fold cross-validation.

Table 3. Training and generalization result on KinFaceW-I Data

	Training	Generalization	Eigenvalue update criterion
Standard LS-SVM	100%	95.06%	none
Proposed method	100%	94.06%	$\lambda' \rangle 0.7\bar{\lambda}$

Figure 2. Example of the KinFaceW-I Dataset

The results on the the KinFaceW-I data are given in Table 3. For this widely used pattern recognition problem, we can see that proposed classification system classifies well on the KinFaceW-I data set.

4.4 Comparison with SVM

Recently SVM has been a powerful methodology for solving problems in nonlinear classification. To evaluate the classification accuracy of the proposed system it is desirable to compare with SVM. Generally a disadvantage of the incremental method is its accuracy compared to the batch method even though it has the advantage of memory efficiency. According to Table 4 and Table 5 we can see that the proposed method has better classification performance compared to batch SVM. Through this result we can show that the proposed classifier has remarkable classification accuracy, although it is worked in an incremental way.

Table 4. Performance comparison of proposed method and SVM using all features

	Training	Generalization	Eigenvalue update criterion
Standard SVM	100%	95.01%	none
Proposed method	100%	95.03%	$\lambda' \rangle 0.7\bar{\lambda}$

Table 5. Performance comparison of proposed method and SVM using extracted features

	Training	Generalization	Eigenvalue update criterion
Standard SVM	100%	94.02%	none
Proposed method	100%	94.06%	$\lambda' \rangle 0.7\bar{\lambda}$

5. Conclusion and Remarks

A conjugate based LS-SVM which combining empirical kernel map was presented for dealing with real time image recognition. Such classifier has following advantages. Proposed image recognition system is more efficient in memory requirement than batch LS-SVM. In batch LS-SVM the (N+1) × (N+1) matrix has to be stored, while for our proposed method does not. It is very useful when dealing with large size data. Experimental results on huge data from the KinFaceW-I data, proposed method shows lead to good performance. By this result we will make a commercial image recognition system with Jade Solution Company.

Acknowledgement

This work was supported by a 2016 research grant from Youngsan University, Republic of Korea.

References

[1] R. Chellappa, C.L. Wilson, and S. Sirohey, “Human and machine recognition of faces : a survey,” in *Proc. IEEE*, vol. 83, No. 5, pp. 705-740, May 1995.

[2] R. Brunelli, and T. Poggio, “Face recognition:feature versus templates,” *IEEE Trans. PAMI*, vol. 15, No. 10, pp. 1042-1052, 1993.

[3] M. Turk, and A. Pentland, “Face recognition using eigenfaces,” in *Proc. IEEE Conf. on CVPR*, pp. 586-591, 1991.

[4] J. Winkeler, B.S. Manjunath, and S. Chandrasekaran, “Subset selection for active object recognition,” In *CVPR*, , *IEEE Computer Society Press*, vol. 2, pp.511-516, June 1999.

[5] P. Hall, D. Marshall, and R. Martin, “On-line eigenalysis for classification,” in *British Machine Vision Conference*, vol. 1, pp. 286-295, Sep. 1998.

[6] W.S. Softky, and D.M. Kammen, “Correlation in high dimensional or asymmetric data set: Hebbian neuronal processing,*” Neural Networks* vol. 4, Nov. pp.337-348, 1991.

[7] V. N. Vapnik, *Statistical learning theory*, John Wiley & Sons, New York, 1998.

[8] J.A.K. Suykens, and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Processing Letters*, vol. 9, pp. 293-300, 1999.

[9] Jiwen Lu, Junlin Hu, Xiuzhuang Zhou, Yuanyuan Shang, Yap-Peng Tan and Gang Wang, “ Neighborhood Repulsed Metric Learning for Kinship Verification,” in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'12)*, 2012.