An Experimental Evaluation of Linear and Kernel-Based Methods for Face Recognition

Himaanshu Gupta, Amit K Agrawal, Tarun Pruthi, Chandra Shekhar, Rama Chellappa
Department of Electrical & Computer Engineering
University of Maryland, College Park, MD 20742
Emails: {hgupta, aagrawal, shekhar, rama}@cfar.umd.edu, tpruthi@glue.umd.edu

Abstract

In this paper we present the results of a comparative study of linear and kernel-based methods for face recognition. The methods used for dimensionality reduction are Principal Component Analysis (PCA), Kernel Principal Component Analysis (KPCA), Linear Discriminant Analysis (LDA) and Kernel Discriminant Analysis (KDA). The methods used for classification are Nearest Neighbor (NN) and Support Vector Machine (SVM). In addition, these classification methods are applied on raw images to gauge the performance of these dimensionality reduction techniques. All experiments have been performed on images from UMIST Face Database.

1. Introduction

Automatic face recognition has been a very active research area in the last decade. The common approaches for face recognition are of two types, local feature-based [1][16] and global or "holistic" [7][8][10][17]. In the local feature-based approach, typically a set of features is extracted from the image (e.g. locations of facial features such as nose, eyes etc.) and used to classify the face. The main advantage of this approach is its relative robustness to variations in illumination, contrast, and small amounts of out-of-plane rotation. Its main disadvantage is that there is generally no reliable method to extract an optimal set of features, hence useful information may be lost in the feature extraction step. In contrast, global approaches use the entire image as the pattern to be classified, so no information is lost. However, they tend to be more sensitive to image variations. In this paper we will concentrate on the global approach.

When using global methods, we usually represent an image of size *nxm* pixels as a vector in *nm*-dimensional space. In practice, however, the *nm*-dimensional space is often too large to allow robust and fast recognition. A common way to resolve this problem is to use dimensionality reduction techniques. Two of the most effective techniques for this purpose are PCA and LDA.

Principal component analysis (PCA) is a powerful technique for extracting global structure from high-dimensional data set. It has been widely adopted to reduce dimensionality and extract abstract features of faces for face recognition [7][8]. But the features extracted by PCA are "global" features for all face classes and thus may not be optimal for discriminating one face class from the others. Kernel PCA (KPCA), recently proposed as a nonlinear extension of PCA for face recognition [2][7], computes the principal components in a high-dimensional feature space, which is non-linearly related to the input space and thus can extract non-linear principal components. However, similar to the linear PCA, KPCA provides the directions corresponding to largest variance in the data and these may not necessarily be the best directions for discrimination.

Linear Discriminant Analysis (LDA), on other hand, finds the directions along which the classes can be optimally differentiated. LDA has been widely used for face recognition [10][17]. However, being a linear technique, it may not perform well when severe non-linearity is involved. To remedy this linear limitation of LDA, Kernel Discriminant Analysis (KDA), a nonlinear approach based on the kernel technique has been developed for extracting nonlinear discriminant features. In this paper, we present the results of a comparative study of the dimensionality reduction methods discussed above. These methods are described in detail in Appendix A.

2. Methodology

We choose as our benchmark nearest neighbor classification (NN) on raw input images after preprocessing (hereafter referred to as raw data). This is because NN is the simplest possible classification scheme, and it provides a benchmark for comparison with more complex classification schemes. For kernel based methods, we use a polynomial kernel $k(\boldsymbol{x},\boldsymbol{y}) = (1+\boldsymbol{x}.\boldsymbol{y})^p$ where p denotes the degree of the polynomial, and Gaussian RBF kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp(\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma^2)$. SVM experiments were performed using the publicly available SVM$^{\text{light}}$ software [6][14].

2.1. Database

All the experiments were conducted using the UMIST database [9][18]. This database consists of cropped gray-scale images of 20 subjects covering a range of sex/race/appearance. The database also accounts for pose variation with both profile and frontal views. There are a maximum of 48 images per subject and a minimum of 19. For illustration, the images of a single subject are shown in Figure 1. All images are taken by the same camera under tightly controlled conditions of illumination. Each image in the database is of size 112x92.

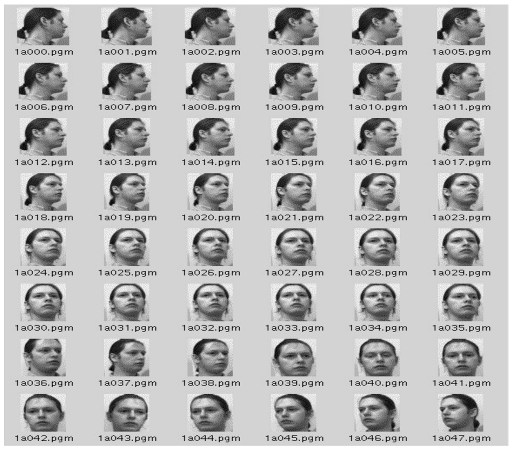

Fig1. Images of a subject from the UMIST database

2.2. Preprocessing

The original images were downsampled by a factor of 4 in each dimension to 28x23. The downsampled images were histogram equalized, and normalized to zero mean and unit variance.

2.3. Generating Training and Test Data

The database has variable number of images for different subjects, but the same number of training and test images for each subject were chosen so that the results are not affected by number of images of any particular subject. Ten runs of each experiment were performed and the results were averaged. In each experiment, nine randomly chosen images for each subject were used in the training set, and nine randomly chosen images for each subject were used in the test set. The test and training set did not have any images in common.

2.4 Dimensionality Reduction

Dimensionality reduction is achieved by linear and kernel-based methods. Linear methods used are PCA and LDA, and kernel-based methods used are KPCA and KDA. The mathematical details of these methods are presented in Appendix A.

2.5 Classification

After global features are extracted using PCA/KPCA or LDA/KDA, classification is performed using two approaches: Nearest Neighbor using the standard L_2–norm for Euclidean distance, and Support Vector Machine [5][11], which is widely used for classification because of certain desirable properties such as provable optimality and low computational cost. Since the classic SVM was formulated for the two-class problem, its basic scheme is extended to multi-class face recognition by adopting a *one-versus-rest* decomposition. This works by constructing c SVMs for a c-class problem. Each SVM is trained to separate its class from all the other classes and then an arbitrator is used between the SVM outputs to produce the final result.

The simplest form of arbitrator is *max-selector* which selects the class whose SVM output for that test vector is maximum. However, the outputs of different SVMs may not be on the same scale. We therefore use the following arbitrator:

- *Clip SVM outputs to [-1,1] and then apply the max-selector.*
- *If more than one SVM outputs are 1, regard it as ambiguous decision*
- *If maximum = -1, regard it as undecided*

We choose to count all the ambiguous and no-decisions as errors so that it reflects the inability of one-versus-rest technique to decide the true class.

3. Experimental Results

3.1. Classification by Nearest Neighbor

3.1.1. PCA/KPCA. In Figures 2 and 3, classification error rates on test data are plotted against the number of principal components (PCs), or equivalently the size of the projected subspace. Since the size of training set is N = 180 (20 subjects x 9 images per subject), the maximum number of PCs is 180.

Figure 2 shows the error plots for PCA and KPCA using polynomial kernel with degree 2,3 and 4. The dashed horizontal line represents the error rate using NN on raw data, which is obviously independent of the number of PCs. Figure 3 shows the plot of classification error rates versus number of PCs for a Gaussian RBF kernel with different values of the kernel parameter (σ). Also plotted is the result using PCA for comparison. An interesting observation is that error performance is worse for polynomial KPCA compared to PCA, and with increasing kernel polynomial degree, error performance degrades even further. This shows that higher order data correlations may not always be useful. This result is in contrast with the results presented in [13] (where better results are obtained with polynomial kernels), which can attributed to differences in the database and the precise form of the polynomial kernel used. For Gaussian RBF kernel, error performance improves with increasing kernel variance, which can be explained as follows: At very low kernel variance, the kernel dot product matrix reduces to the trivial identity matrix for the training set, and for the test set, the dot products between training and test vectors will be very small, or vanish entirely due to numerical underflow. Hence all projections for a particular test vector will be close to zero and classification by NN algorithm will be completely arbitrary. In other words, information in the training set is not utilized. As variance is increased, the above effect is reduced. The best performance using a Gaussian kernel is obtained with $\sigma = 0.26*d$ (where d is the dimension of training vector). As σ is increased beyond this, performance saturates. Another interesting observation is that even the best KPCA performance (obtained by using a Gaussian RBF kernel) is not superior to that obtained using PCA.

In both cases, NN on raw data gives better performance compared to PCA and KPCA. This is a strong argument against using component analysis for face recognition.

3.1.2. LDA/KDA. We would expect LDA, which is a discrimination scheme, to give better classification results compared to PCA, as the latter is primarily a representation scheme. The results obtained support this observation.

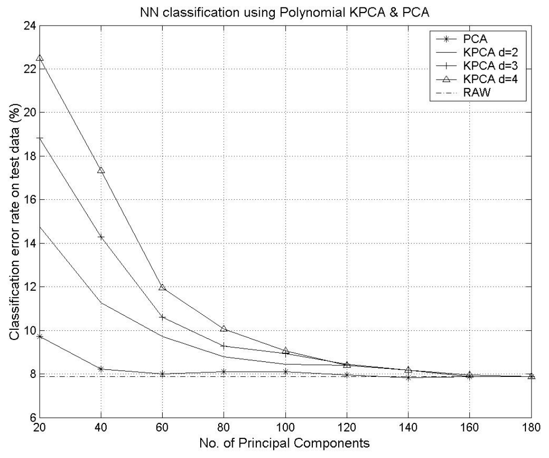

Fig 2. Plot of classification error rates for Linear and Polynomial Kernel PCA with varying number of principal components. d denotes the degree of polynomial kernel.

Figure 4 shows the error rates plotted against the dimension of projected subspace for LDA and KDA with polynomial kernel degree=2,3 and 4. LDA performs better than NN on raw data. But as in PCA, polynomial kernels perform poorly compared to LDA and the performance degrades with increasing kernel degree. Figure 5 shows the results for RBF kernels with different variances. Also plotted is the result using LDA for comparison. The same reasoning as before can be applied to argue that the performance will be poor at low RBF kernel variance. As variance is increased, performance improves and then saturates. In this case also, we observe that even the best KDA (with RBF kernel) performance is not superior to that obtained using LDA, which raises questions about the appropriateness of using kernels for this problem. However, unlike in the case of PCA/KPCA, we do get improved performance over NN on raw data by using LDA and KDA.

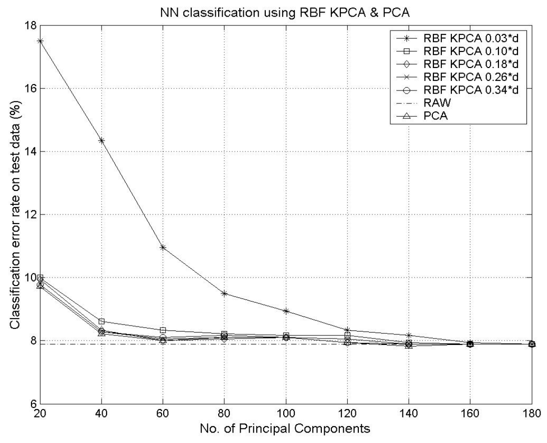

Figure3. Plot of classification error rates for Linear and Gaussian RBF Kernel PCA with varying number of principal components. d is the input vector dimension.

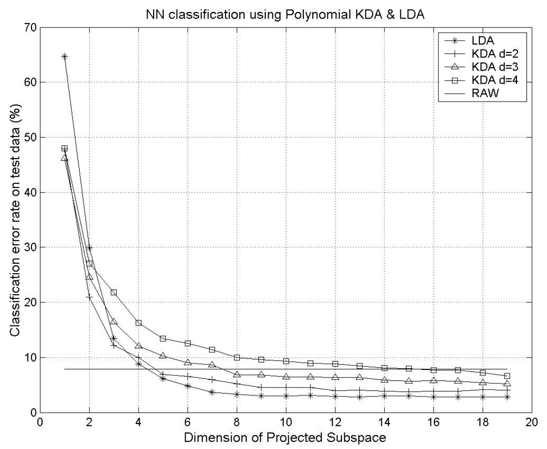

Figure 4. Plot of classification error rates for LDA and Polynomial KDA with varying dimensionality of projected space. d denotes the degree of polynomial kernel.

3.2. Classification using SVMs

Figure 6 shows the classification error rates obtained by using Linear SVM as the classifier after KPCA and KDA using various kernels. Since SVMs have been shown to perform well even when applied directly to raw input without any dimensionality reduction [12], we have showed that result as the "Raw" bar. The figure also shows the classification error rate on raw data using NN. We observe that Gaussian RBF KDA gives the best result in this case over PCA, KPCA and is only marginally better than the results using LDA. Figure 7 shows the classification error rates using Linear PCA and LDA followed by kernel SVMs (using polynomial kernels of degree 2 and 3, and Gaussian RBF kernel). We observe that LDA followed by Gaussian RBF SVM gives the best error performance.

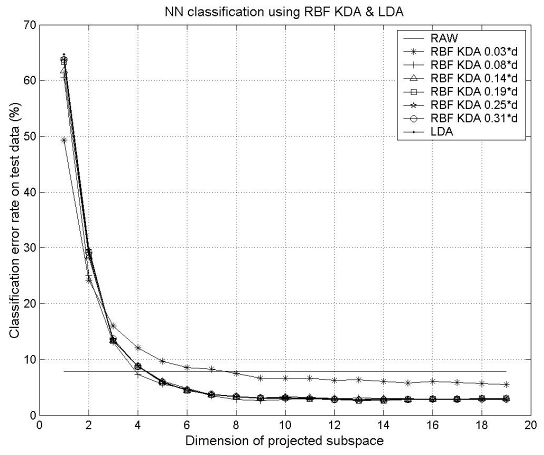

Figure 5. Plot of classification error rates for LDA and Gaussian RBF KDA with varying dimensionality of projected space. The values of the kernel parameter σ are indicated as a multiple of the pattern dimension d.

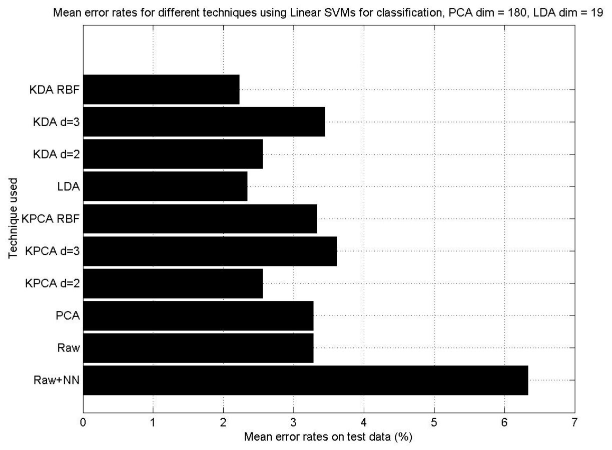

Figure 6. Bar graph showing classification error rates for classification using Linear SVMs. In the above figure, d=2,3 implies degree 2,3 polynomial kernels. σ =0.08*D for Gaussian RBF KPCA. , and σ =0.06*D for Gaussian RBF KDA, where D is the dimension of the input vector.

4. Conclusions and Future Work

After comparison of various linear and non-linear techniques, we arrive at the following conclusions:

- Although kernels enable us to work in a high dimension feature space, they do not always ensure better performance. Introduction of kernels might even degrade the performance if the problem is close to linearly separable. Moreover, the choice of optimal kernel is not always obvious.
- LDA/KDA outperform PCA/KPCA because of the inherent discrimination ability of the former.
- The SVM classifier outperforms the Nearest Neighbor algorithm in all the methods used for feature extraction. In SVM classifier, we find that kernel SVM with linear preprocessing (PCA/LDA) performs better than Linear SVM with non-linear preprocessing (KPCA/KDA), though in spirit both are similar. The lowest error performance of 1.77% was achieved on UMIST database by using LDA with RBF SVM classifier.

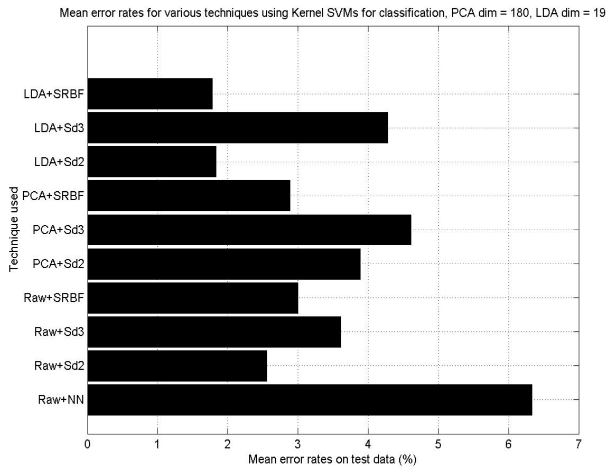

Figure 7. Bar graph showing classification error rates using Kernel SVMs .In the above figure, SRBF=RBF SVM, Sd2=Polynomial SVM with degree 2, Sd3= Polynomial SVM with degree 3.

In future, we plan to perform this study on other publicly available face databases like Yale and CMU-PIE databases, and evaluate the performance of these methods under more difficult conditions. Also, we would like to apply these methods to the problems of Face Detection and Recognition using Gait.

Appendix A

Principal Component Analysis (PCA)

Let $\chi = \{x_1, x_2, ..., x_N\}$ be the training set, where each x_i represents a training vector, N is the size of training set and d is the dimensionality of the input vector. Using linear PCA, the maximum dimension of the projected subspace is min*(N,d)*. Let $X = [x_1 | x_2 | ..... | x_N]$ denote the matrix containing the training vectors. PCA involves finds the eigenvectors of the covariance matrix, solving the following equation

$$XX^T \boldsymbol{e} = \lambda \boldsymbol{e} \quad (1)$$

where $\boldsymbol{e}$ is an eigenvector and λ an eigenvalue. Using the method of Kirby and Sirovich [15], we pre-multiply both sides by X^T, and then equation (1) can be written as

$$K\alpha = \lambda\alpha \quad (2)$$

where $K = X^T X$ and $\alpha = X^T \boldsymbol{e}$. K is referred to as the inner product matrix of the training samples since $K_{ij} = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$.

This is a standard eigenvalue problem, which can be solved for α and λ. From (2), we get $\boldsymbol{e} = X\alpha$ (after normalization $\alpha = \frac{\alpha}{\sqrt{\lambda}}$). The projections on the first q eigenvectors (corresponding to largest q eigenvalues) constitute the feature vector. For a test vector $\boldsymbol{x}$, the principal component y corresponding to eigenvector $\boldsymbol{e}$ is given by

$$y = \boldsymbol{e}^T \boldsymbol{x} = (X\alpha)^{\mathrm{T}} \boldsymbol{x} = \alpha^T X^T \boldsymbol{x} = \sum_{i=1}^{N} \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) \quad (3)$$

where $\boldsymbol{a} \cdot \boldsymbol{b}$ denotes the inner (dot) product of vectors **a** and **b**.

Kernel PCA in Feature Space

Kernel based methods can map the input space to a high (possibly infinite) dimensional feature space in a non-linear way. The projection into higher space is done using inner products without explicit computation of the mapping, which makes it computationally feasible. Essentially, the computations are done in a subspace of dimension N of a higher dimensional space. In contrast to PCA, the dimension of the projected subspace can reach N using KPCA. The basic concept of kernel PCA is to first map the input data χ into a feature space F via a non-linear mapping $\phi(\cdot)$ and then perform a linear PCA in F. The motivation is that a training set, which may not be linearly separable in input space, may be linearly separable in the mapped space.

Again, let $X = [\boldsymbol{x}_1 | \boldsymbol{x}_2 | ... | \boldsymbol{x}_N]$ denote the matrix containing the training vectors, and let $\Phi = [\boldsymbol{\phi}(\boldsymbol{x}_1) | .... | \boldsymbol{\phi}(\boldsymbol{x}_N)]$ be its image in the feature space. Assuming that the mapped data are centered, i.e., $\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_i) = 0$ (the centering method in F is explained at the end of this sub-section) let $K = \Phi^{\mathrm{T}}\Phi$, with $K_{ij} = \boldsymbol{\phi}(\boldsymbol{x}_i).\boldsymbol{\phi}(\boldsymbol{x}_j)$. Then the principal directions satisfy equation (2) and are given by $\boldsymbol{e} = \Phi\alpha$ (after normalization of α). For a test vector $\boldsymbol{x}$, the principal component y corresponding to eigenvector **e** is given by:

$$y = \boldsymbol{e}^T \boldsymbol{\phi}(\boldsymbol{x}) = \alpha^{\mathrm{T}} \Phi^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i).\boldsymbol{\phi}(\boldsymbol{x}) \quad (4)$$

The dot-product matrix K can be computed by choosing a kernel *k(**x**,**y**)* such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i).\boldsymbol{\phi}(\boldsymbol{x}_j) = K_{ij}$, and thus avoiding any computation in the high dimensional feature space. This is referred to as the "kernel trick". Then, the first q principal components (assuming that the eigenvectors are sorted in a descending order of eigenvalues) constitute the q-dimensional feature vector for a face pattern.

In general, the assumption of centered data in feature space made above is not reasonable. A method to center the mapped data is described here: Let $\boldsymbol{\phi}^{\sim}(\boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_j) - 1/N * \sum_i \boldsymbol{\phi}(\boldsymbol{x}_i)$, $1 \le j \le N$ be the centered mapped data in the feature space. $\Phi^{\sim} = [\boldsymbol{\phi}^{\sim}(\boldsymbol{x}_1) | \boldsymbol{\phi}^{\sim}(\boldsymbol{x}_2) | ... | \boldsymbol{\phi}^{\sim}(\boldsymbol{x}_N)] = \Phi(\mathbf{I} - \mathbf{1}_{N\times N}/N)$ be the matrix containing the centered mapped training vectors. Then, the inner product matrix $K^{\sim}$ for the centered mapped data can be obtained from inner product matrix K of non-centered data by

$$K^{\sim} = \Phi^{\sim\mathrm{T}}\Phi^{\sim} = (\mathbf{I} - \frac{\mathbf{1}_{N\times N}}{N})^{\mathrm{T}} K (\mathbf{I} - \frac{\mathbf{1}_{N\times N}}{N}) \quad (5)$$

where $\mathbf{I}_{NxN}$ is a NxN identity matrix and $\mathbf{1}_{NxN}$ is a NxN matrix of all ones.

Fisher's Linear Discriminant

LDA [3] finds a linear transformation by maximizing the between-class variance and minimizing the within-class variance. Assume N_1 training samples $\chi_1 = \{\boldsymbol{x}_1^1, ..., \boldsymbol{x}_{N_1}^1\}$ belong to class 1 and N_2 samples $\chi_2 = \{\boldsymbol{x}_1^2, ..., \boldsymbol{x}_{N_2}^2\}$ belong to class 2 with $N = N_1 + N_2$. Let $X_1 = [\boldsymbol{x}_1^1 | ... | \boldsymbol{x}_{N_1}^1]$, $X_2 = [\boldsymbol{x}_1^2 | ... | \boldsymbol{x}_{N_2}^2]$ and $X_{dxN} = [X_{1_{dxN_1}} | X_{2_{dxN_2}}]$ be the matrix containing all the training vectors. Fisher's linear discriminant is given by the vector $\boldsymbol{w}$, which maximizes

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} \quad (6)$$

where

$$S_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$$
$$S_W = \sum_{i=1,2} \sum_{\boldsymbol{x} \in X_i} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

are the between and within class scatter matrices respectively and $\boldsymbol{m}_i$ are the class sample means given by $\boldsymbol{m}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{x}_j^i$. The maximization of (6) results in a generalized eigenvalue problem $S_B \boldsymbol{w} = \lambda S_W \boldsymbol{w}$. If S_W is non-singular, $\boldsymbol{w}_{opt} = \frac{S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)}{| S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) |}$. Now we will present a mathematical formulation that will express the cost function $J(\boldsymbol{w})$ in terms of the dot-product matrix of the input training samples.

Let us define, $\boldsymbol{v}_1 = \mathbf{1}_{N_1x1} / N_1$, $\boldsymbol{v}_2 = \mathbf{1}_{N_2x1} / N_2$, $\boldsymbol{u}_1^T = \mathbf{1}_{1xN_1}$, $\boldsymbol{u}_2^T = \mathbf{1}_{1xN_2}$, $\boldsymbol{I}_1 = [\mathbf{I}_{N_1} \mid \mathbf{0}_{N_2xN_1}]^T$, and $\boldsymbol{I}_2 = [\mathbf{0}_{N_1xN_2} \mid \mathbf{I}_{N_2}]^T$, where $\mathbf{1}_{mxn}$ ($\mathbf{0}_{mxn}$) denote an *mxn* matrix of all ones (zeroes), and $\mathbf{I}_m$ denotes the *mxm* identity matrix.

Now, we can express the numerator of Equation (6) as $\boldsymbol{w}^T S_B \boldsymbol{w} = \boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} = \alpha^T KQK\alpha$ where $K = X^T X$ is the dot product matrix, and $Q = (\mathbf{I}_1\boldsymbol{v}_1 - \mathbf{I}_2\boldsymbol{v}_2)(\mathbf{I}_1\boldsymbol{v}_1 - \mathbf{I}_2\mathbf{v}_2)^T$. The denominator can be expressed as $\boldsymbol{w}^T S_W \boldsymbol{w} = \alpha^T KRK\alpha$, with $R = R_1 + R_2$ where $R_i = (\mathbf{I}_i - \mathbf{I}_i\boldsymbol{v}_i\boldsymbol{u}_i^T)(\mathbf{I}_i - \mathbf{I}_i\boldsymbol{v}_i\boldsymbol{u}_i^T)^T$, i=1,2. Hence, maximizing (6) is equivalent to maximizing

$$J(\alpha) = \frac{\alpha^T KQK\alpha}{\alpha^T KRK\alpha} \qquad (7)$$

which is in terms of the input dot-product matrix *K*.

Discriminant Analysis in Feature Space

As in the case of PCA, we can use the kernel idea for LDA to find non-linear directions by first mapping the data non-linearly into some feature space *F* and computing Fisher's linear discriminant there, thus implicitly yielding a non-linear discriminant in input space. Let $\phi(\cdot)$ be a non-linear mapping to some feature space *F*. Let $\Phi = [\boldsymbol{\phi}(\boldsymbol{x}_1) \mid ... \mid \boldsymbol{\phi}(\boldsymbol{x}_N)]$ be the matrix which contains the non-linear mappings of all the training samples. To find the linear discriminant in *F* we need to maximize

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B^\Phi \boldsymbol{w}}{\boldsymbol{w}^T S_W^\Phi \boldsymbol{w}} \qquad (8)$$

Again, this problem can be reduced to a eigenvalue problem of the same form as in LDA. Instead of mapping the data explicitly into the feature space, we seek a formulation of the algorithm which uses only the dot-products $\boldsymbol{\phi}(\boldsymbol{x}_i).\boldsymbol{\phi}(\boldsymbol{x}_j)$ of the images of training patterns in feature space. We are then able to compute these dot-products efficiently without mapping explicitly to *F*.

As shown in the LDA case, the cost function in this can be reduced to the form in Equation 7. In this case, the dot-product matrix *K* is $\Phi^T\Phi$, and can be computed by choosing a kernel of the form $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i).\boldsymbol{\phi}(\boldsymbol{x}_j) = K_{ij}$. The projection y_j of a test vector $\boldsymbol{x}$ corresponding to j^{th} eigenvector is given by

$$y_j = \sum_{i=1}^N \alpha^j{}_i k(\boldsymbol{x}, \boldsymbol{x}_i) = \sum_{i=1}^N \alpha_i^j \boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}_i) \qquad (9)$$

Acknowledgement

We are grateful to UMIST, UK for making the UMIST face database available for public research, and for granting permission to publish images of the subject in this paper.

References

[1] B.S. Manjunath, R. Chellappa, C. von der Malsburg, "A Feature Based Approach to Face Recognition", in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pp.373-378, 1992.

[2] Kwang In Kim, Keechul Jung and Hang Joon Kim, "Face Recognition Using Kernel Principal Component Analysis", *IEEE Signal Processing letters*, Vol 9, No. 2, Feb 2002.

[3] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, Klaus-Robert Müller, "Fisher Discriminant Analysis With Kernels", *Neural Networks for Signal Processing* IX, 1999

[4] Bernhard Scholkpof, Alexander Smola, and Klaus-Robert Muller, "Nonlinear Component Analysis as a kernel Eigenvalue Problem", Tubingen, Germany*, Tech. Report No 44*, Dec 1996.

[5] Christopher J C Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", Bell Laboratories, Lucent Technologies, 1998.

[6] T. Joachims, "Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning", B. Schölkopf and C. Burges and A. Smola (ed.), *MIT-Press*, 1999.

[7] W. Y. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips, "Face recognition: A literature survey", *UMD CfAR Technical Report CAR-TR-948, 2000.*

[8] M.Turk and A.Pentland, "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, 3(1):71-86,1991.

[9] UMIST Face Database, UK. *URL: http://images.ee.umist.ac.uk/danny/database.html.*

[10] K.Etemad and R.Chellappa, "Discriminant Analysis for recognition of human face images", *Journal of the Optical Society of America* A,14(8):1724-1733, 1997.

[11] V.Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, NY, 1995.

[12] G.D Guo, S.Z.Li, and K.L. Chan, "Face recognition by Support Vector Machines", *Proc. Int. Conf. Automatic Face and Gesture Recognition*, pp.196-201, 2000.

[13] Yang, M. H., "Kernel EigenFaces vs Kernel Fisherfaces: Face Recognition using Kernel Methods", *Proc. of the Fifth Int. Conf. on Automatic Face and Gesture Recognition*, pp215-220, May 2002.

[14] SVM^{light} software,URL: *http://svmlight.joachims.org.*

[15] L. Sirovich, M. Kirby, "Low dimensional procedure for the characterization of human faces", *Journal of Optical Society of America*, Vol.4, No.3, pp.519-524, 1987.

[16] L. Wiskott, J..M. Fellous, C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching", *IEEE Trans. On Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 775-779, 1997.

[17] P.N Belhumeur, D.J Kriegman, J.P. Hespanha, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection", *IEEE Conference on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 711-720, 1997.

[18] Daniel B. Graham, Nigel M. Allinson, "Face Recognition: From Theory to Applications", *NATO ASI Series F, Computer and Systems Sciences*, Vol. 163, pp. 446-456, 1998.