IJPRAI VOL. 21, NO. 7, PP. 1213-1231, NOV. 2007

Feature Extraction based on Direct Calculation of Mutual Information

Nojun Kwak

nojunk@ieee.org

Division of Electrical & Computer Engineering, Ajou University

San 5 Wonchun-Dong Yeongtong-Gu, Suwon 443-749 KOREA

Abstract

In many pattern recognition problems, it is desirable to reduce the number of input features by extracting important features related to the problems. By focusing on only the problem-relevant features, the dimension of features can be greatly reduced and thereby can result in a better generalization performance with less computational complexity. In this paper, we propose a feature extraction method for handling classification problems. The proposed algorithm is used to search for a set of linear combinations of the original features, whose mutual information with the output class can be maximized. The mutual information between the extracted features and the output class is calculated by using the probability density estimation based on the Parzen window method. A greedy algorithm using the gradient descent method is used to determine the new features. The computational load is proportional to the square of the number of samples. The proposed method was applied to several classification problems, which showed better or comparable performances than the conventional feature extraction methods.

Keywords

Feature extraction, mutual information, Parzen window, gradient descent, subspace method, optimization, classification.

I. Introduction

For many pattern recognition problems, it is desirable to reduce the number of input features through feature extraction because irrelevant or redundant features tend to complicate the learning process, thereby resulting in a poor performance [1]. Even when the features presented contain sufficient information on the problem, the result may become erroneous because the dimension of the feature space could be too large. Reducing the dimensionality of the feature space may improve the learning process by considering only the most important data representation, possibly with elements retaining the maximum information on the original data and with better generalization capabilities [2]. Dimensionality reduction through feature extraction is quite desirable not only in the aspect of the number of required data, but also in terms of data storage and computational complexity.

Many studies have been done on the feature extraction problems. The principal component analysis (PCA) [3], the linear discriminant analysis (LDA) [4], and the independent component analysis (ICA) [5] have been widely used.

Although PCA is one of the most popular and widely used methods for classification problems such as face recognition problems [6], it can still be improved for classification problems since it is an unsupervised learning method that does not make use of the output class information. PCA is useful in reducing the dimension of a feature space to a manageable size to be used for a classifier. Likewise, ICA, which is another unsupervised learning method, leaves room for improvement in classification problems. Unlike PCA and ICA, LDA focuses on the classification problems to find the optimal linear discriminating functions. Although it is simple and powerful, the application of this method is limited to cases where the means of the classes are separated well.

Another line of research on feature extraction is using the mutual information directly as a measure of extracting good features. Although mutual information is a good measure in pattern recognition problems, researchers have only recently begun to use it due to the computational burden [7] [8] [9] [10]. Ullman et al. applied this to binary classification problems with binary features [7]. Fisher et al. [8] used the quadratic mutual information, which is related to Renyi's entropy, in feature extraction problems.

In feature extraction problems, ICA-FX, which is an extension of the ICA, is another approach [11] [12] [13]. The algorithm attempts to maximize the mutual information between the input features and output classes indirectly under the assumption that the sources are independent of the classes. The performance of this method depends on how the data is distributed to satisfy the assumption.

In this paper, we propose a method of extracting features by using the mutual information based on Shannon's entropy. In computing the mutual information, one needs to know the probability density function, which can be estimated by the Parzen window method [9] [10]. New features can be generated by linear combinations of input vectors. Among these new feature candidates, we find the one that maximizes the mutual information with the output. And this process is repeated until all the necessary features are obtained.

In the following section, the preliminaries are briefly introduced and the feature extraction problem is formalized. A new feature extraction method is proposed in Section 3. In Section 4, the proposed algorithm is applied to several classification problems to show its efficacy. Conclusions follow in Section 5.

II. Preliminaries and Problem Formulation

For clarity, capital letters hereafter represent random variables and small letters are the instances of the corresponding random variables. Boldfaced letters represent vectors.

A. Entropy and Mutual Information

In solving feature extraction problems, this study attempts to find the inputs that contain as much information on the outputs as possible. The information theory provides a means of measuring the information with mutual information [14] [15].

Let $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ be the probability mass function (or probability density function, *pdf*) for random vector $\boldsymbol{X}$ and $\boldsymbol{Y}$. Also let $p(\boldsymbol{x}, \boldsymbol{y})$ be the joint probability mass function (or joint *pdf*) of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The mutual information between discrete random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as

$$I(\boldsymbol{X};\boldsymbol{Y}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{x},\boldsymbol{y}) \log \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})}.$$

where $\mathcal{X}$ and $\mathcal{Y}$ are alphabets of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. If the mutual information between the two random vectors is large (small), the two vectors are closely (not closely) related. If the mutual information becomes zero, the two random vectors are independent.

The other two basic quantities of information theory is the entropy and conditional entropy, which are defined as follows:

$$\begin{aligned}\text{Entropy: } H(\boldsymbol{X}) &= -\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \log p(\boldsymbol{x}) \\ \text{Conditional Entropy: } H(\boldsymbol{Y}|\boldsymbol{X}) &= -\sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{x},\boldsymbol{y}) \log p(\boldsymbol{y}|\boldsymbol{x}).\end{aligned} \tag{1}$$

Among the relationships between the mutual information and entropy, the following relationship will be used later:

$$I(\boldsymbol{X};\boldsymbol{Y}) = H(\boldsymbol{X}) - H(\boldsymbol{X}|\boldsymbol{Y}). \tag{2}$$

For most classifying systems, the output class, C, is represented by a discrete random variable, while the input features are generally continuous. The differential entropy and mutual information for continuous random variables are defined as follows,

$$\begin{aligned}H(\boldsymbol{X}) &= -\int p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x} \\ I(\boldsymbol{X},\boldsymbol{Y}) &= \int p(\boldsymbol{x},\boldsymbol{y}) \log \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} d\boldsymbol{x} d\boldsymbol{y},\end{aligned}$$

but it is very difficult to determine the *pdf*s $(p(\boldsymbol{x}), p(\boldsymbol{y}), p(\boldsymbol{x},\boldsymbol{y}))$ and perform the integrations. Therefore, the continuous input feature space is divided into several discrete partitions and the entropy and mutual information are calculated using the definitions for the discrete random variables. The inherent error that exists in the quantization process poses as a problem in computing the entropy and mutual information of continuous variables. To avoid this, the Parzen window method presented in the following subsection can be used to estimate the *pdf*s of continuous random variables.

B. Density Estimation by the Parzen Window

A density estimation by the Parzen window can be used to approximate the probability density $p(\boldsymbol{x})$ of a vector of continuous random variables, $\boldsymbol{X}$ [16]. It involves the superposition of a normalized window function centered on a set of samples. Given a data set of n N-dimensional training vectors $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, the *pdf* estimate by the Parzen window method is given by the following equation:

$$\hat{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \phi(\boldsymbol{x} - \boldsymbol{x}_i, h),$$

where $\phi(\cdot)$ is the window function and h is the window width parameter.

For window functions, the rectangular and the Gaussian window functions are commonly used. The Gaussian window function is given by

$$\phi(\boldsymbol{z}, h) = \frac{1}{(2\pi)^{N/2} h^N |\Sigma|^{1/2}} \exp\left(-\frac{\boldsymbol{z}^T \Sigma^{-1} \boldsymbol{z}}{2h^2}\right), \tag{3}$$

where Σ is a covariance matrix of an N-dimensional random vector $\boldsymbol{Z}$.

Parzen reported that $\hat{p}(\boldsymbol{x})$ will converge to the true density if $\phi(\cdot)$ and h are selected properly [16]. For non-smooth classes of densities, kernel methods such as the Parzen window density estimator are known to have convergence problems. In this paper, we assume that the probability densities of features are smooth, and do not deal with the non-smooth classes of densities.

C. Feature extraction: Problem formulation

The success of a feature extraction algorithm for classification problems is determined by the probability of misclassification. When *pdf*s are known, the Bayes classifier gives the minimum classification error known as the Bayes error. The Bayes error is upper and lower bounded by the mutual information as follows [15] [17]:

(Bayes error bounds) Let $\boldsymbol{F}$ and C be random variables that represent the input features and output class, respectively. In addition, let N_c be the number of classes. If the output class, c, is to be estimated using the input features, $\boldsymbol{f}$, the probability of error, P_E, is bounded as follows:

$$\frac{H(C|\boldsymbol{F})-1}{\log N_c} = \frac{H(C)-I(\boldsymbol{F};C)-1}{\log N_c} \leq P_E \leq \frac{H(C|\boldsymbol{F})}{2} = \frac{H(C)-I(\boldsymbol{F};C)}{2}. \tag{4}$$

In addition to this, the following inequality is used to formalize the purpose of the feature extraction problems.

(Data processing inequality [15]) Let $\boldsymbol{X}$ and C be random variables, which represent the input features and output class, respectively. For any deterministic function $T(\cdot)$ of $\boldsymbol{x}$, mutual information between $\boldsymbol{F} \triangleq T(\boldsymbol{X})$ and the output class, C, is upper-bounded by mutual information between $\boldsymbol{X}$ and C:

$$I(\boldsymbol{F};C) = I(T(\boldsymbol{X});C) \leq I(\boldsymbol{X};C)$$

where the equality holds if the transformation is invertible.

Suppose that there are N zero mean original input features, $\boldsymbol{X} = [X_1, \cdots, X_N]^T \in \Re^N$, and an output class, C. The purpose of the feature extraction is to extract $M(< N)$ new features, $\boldsymbol{F} = [F_1, \cdots, F_M]^T$, from $\boldsymbol{X}$ containing as much information on the class as possible.

Furthermore, suppose the situation where only linear combinations of the original features are considered as the candidates for the new features. In this case, finding the i-th new feature F_i is equivalent to searching for the optimal weight vector, $\boldsymbol{w}_i^* \in \Re^N$, $i = 1, \cdots, M$, where F_i and $\boldsymbol{w}_i^*$ have the following relationship:

$$F_i = \boldsymbol{w}_i^{*T}\boldsymbol{X}. \tag{5}$$

Since the entropy of class, $H(C)$, and the number of classes, N_c, are fixed in (4), the upper and lower bounds of P_E are minimized when $I(\boldsymbol{F};C)$ becomes a maximum. It is, thus, necessary for good feature extraction methods to extract the features that maximize the mutual information with the output class. The *data processing inequality* shows that no transformation $T(\cdot)$ can increase the mutual information between the input features and the output class. Consequently, the goal of a feature extraction is to extract $M(< N)$ features, $\boldsymbol{F}$, from $\boldsymbol{X}$, so that the mutual information between the newly extracted features $\boldsymbol{F}$ and the output class C, $I(\boldsymbol{F}; C)$, approaches $I(\boldsymbol{X}; C)$.

In order to obtain mutual information in a more direct way, the *pdf*s of input and output variables must be known, but this is difficult in practice. The histogram method has been used in estimating the *pdf*s. However, the histogram method requires extremely large memory space for calculating the mutual information. For example, in extracting M features, if the output classes are composed of N_c classes and the jth input feature space is divided into P_j partitions to obtain the histogram, there must be $N_c \times \Pi_{j=1}^{M} P_j$ cells to compute $I(\boldsymbol{F}; C)$. Therefore calculation of mutual information is difficult by estimating the *pdf*s with a histogram. In order to overcome these problems, Parzen window was used to estimate the mutual information between two continuous variables in [8] [9] [10]. The detailed process of this mutual information calculation follows in the next subsection.

D. Calculation of Mutual Information using Parzen Window

Rewriting (2), the mutual information between the input features, $\boldsymbol{F}$, and the class, C, can be represented as follows:

$$I(\boldsymbol{F}; C) = H(C) - H(C|\boldsymbol{F}). \tag{6}$$

In this equation, the class is a discrete variable, and $H(C)$ can be easily calculated in the same way as in (1). However, the conditional entropy

$$H(C|\boldsymbol{F}) = -\int_{\boldsymbol{F}} p(\boldsymbol{f}) \sum_{c=1}^{N_c} p(c|\boldsymbol{f}) \log p(c|\boldsymbol{f}) d\boldsymbol{f}, \tag{7}$$

is difficult to obtain because it is not easy to estimate $p(c|\boldsymbol{f})$.

An estimate of the conditional *pdf* $\hat{p}(\boldsymbol{f}|c)$ of each class can be obtained using the Parzen window method as

$$\hat{p}(\boldsymbol{f}|c) = \frac{1}{n_c} \sum_{i \in I^c} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h), \tag{8}$$

where $c = 1, \cdots, N_c$, n_c is the number of the training samples belonging to class c, and I^c is the set of indices of the training samples belonging to class c.

According to the Bayesian rule, the conditional probability $p(c|\boldsymbol{f})$ can be written as

$$p(c|\boldsymbol{f}) = \frac{p(c|\boldsymbol{f})}{\sum_{k=1}^{N_c} p(k|\boldsymbol{f})} = \frac{p(c)p(\boldsymbol{f}|c)}{\sum_{k=1}^{N_c} p(k)p(\boldsymbol{f}|k)}. \tag{9}$$

Using (8), the estimate of the conditional probability becomes

$$\hat{p}(c|\boldsymbol{f}) = \frac{\sum_{i \in I^c} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h_c)}{\sum_{k=1}^{N_c} \sum_{i \in I^k} \phi(\boldsymbol{f} - \boldsymbol{f}_i, h_k)}, \tag{10}$$

where h_c and h_k are window width parameters corresponding to class c and class k respectively.

Using the Gaussian window function (3) for ϕ with the same window width parameter, h, and the same covariance matrix, $\Sigma_{\boldsymbol{F}}$, for each class, (10) becomes

$$\hat{p}(c|\boldsymbol{f}) = \frac{\sum_{i \in I^c} \exp(-\frac{(\boldsymbol{f}-\boldsymbol{f}_i)^T \Sigma_{\boldsymbol{F}}^{-1} (\boldsymbol{f}-\boldsymbol{f}_i)}{2h^2})}{\sum_{k=1}^{N_c} \sum_{i \in I^k} \exp(-\frac{(\boldsymbol{f}-\boldsymbol{f}_i)^T \Sigma_{\boldsymbol{F}}^{-1} (\boldsymbol{f}-\boldsymbol{f}_i)}{2h^2})}. \tag{11}$$

Using the same covariance matrix for each class is justified because for multiclass classification problems, there may not be enough samples, and the error for the estimate of the class specific covariance matrix can be large.

Replacing the integration with a summation of the sample points and assuming that each sample has equal probability in (7), it becomes

$$\hat{H}(C|\boldsymbol{F}) = -\sum_{j=1}^{n} \frac{1}{n} \sum_{c=1}^{N_c} \hat{p}(c|\boldsymbol{f}_j) \, \log \hat{p}(c|\boldsymbol{f}_j), \tag{12}$$

where $\boldsymbol{f}_j$ is the jth sample of the n training samples. With (6) and (11), the estimate of the mutual information is obtained as follows:

$$\hat{I}(\boldsymbol{F}; C) = -\sum_{c=1}^{N_c} \hat{p}(c) \log \hat{p}(c) + \sum_{j=1}^{n} \frac{1}{n} \sum_{c=1}^{N_c} \hat{p}(c|\boldsymbol{f}_j) \, \log \hat{p}(c|\boldsymbol{f}_j).$$

The computational complexity is proportional to the square of the number of the samples, n^2, and the dimension of the input feature space, N, regardless of the number of the classes. Unlike the calculation of the mutual information using discrete quantization, this method has no problem with memory allocation. If the sample size is too large, either the clustering method [18] or the sample selection method [19] may be used to reduce the memory and to expedite the calculation.

With this estimation, a new method of feature extraction is proposed in the following section.

III. Feature extraction by maximizing mutual information

In this section, a new feature extraction algorithm, PWFX, that directly maximizes the mutual information between the input features and the output class is presented.

A. PWFX

In this paper, only linear combinations of the original features, $\boldsymbol{X}$, are considered as candidates for the new features and the optimal weight vectors, $\boldsymbol{w}_i^* \in \Re^N$'s, $i = 1, \cdots, M$, are searched for.

As stated in Section 2, the goal is to extract the features with the maximum mutual information on the class, but it is very difficult to search for the optimal solution among all the sets of linear transformations in (5). In order to alleviate this problem, new features are extracted one by one until the number of extracted features reaches M. The procedure of PWFX, which is based on the greedy algorithm, can be described as follows:

Step 1. (Initialization) Set $\mathcal{F} \longleftarrow$ "empty set."

Step 2. (Preprocessing) Preprocess the data.

Step 3. (Greedy extraction) Repeat the following until the desired number, M, of the features are extracted.

(a) (Searching for the weights of the maximal mutual information) Search for the weight vector $\boldsymbol{w} \in \Re^N$ that maximizes $I(\boldsymbol{F}^*_{i-1}, \boldsymbol{w}^T\boldsymbol{X}; C)$ and denote it as $\boldsymbol{w}^*_i$.

(b) (Extraction of the next feature) Set $F_i = \boldsymbol{w}^{*T}_i\boldsymbol{X}$ and $\mathcal{F} \longleftarrow \mathcal{F} \cup \{F_i\}$.

Step 4. Output the set $\mathcal{F}$ containing the extracted features.

Here, $\boldsymbol{F}^*_k \in \Re^k$ is the vector whose components are the elements of $\mathcal{F}$ when $|\mathcal{F}|= k$. Note that $\boldsymbol{F} = \boldsymbol{F}^*_M$ in this notation.

Henceforth, for convenience, $\boldsymbol{F}_k$ will be used to denote the k dimensional vector whose first $(k-1)$ components are those of $\boldsymbol{F}^*_{k-1}$ and the last component is $\boldsymbol{w}^T\boldsymbol{X}$. In addition, the matrices $W^*_k = [\boldsymbol{w}^*_1, \cdots, \boldsymbol{w}^*_k]$ and $W_k = [\boldsymbol{w}^*_1, \cdots, \boldsymbol{w}^*_{k-1}, \boldsymbol{w}]$ shall denote the set of weights such that $\boldsymbol{F}_k = W_k^T\boldsymbol{X}$ and $\boldsymbol{F}^*_k = W^{*T}_k\boldsymbol{X}$, respectively.

Because $H(C)$ remains constant throughout the calculation, in order to extract the i-th feature, maximizing $I(\boldsymbol{F}^*_{i-1}, \boldsymbol{w}^T\boldsymbol{X}; C) = I(\boldsymbol{F}_i; C)$ is equivalent to minimizing $H(C|\boldsymbol{F}_i)$ with respect to $\boldsymbol{w}$, and the problem will now be to find

$$\boldsymbol{w}^*_i = \arg\min_{\boldsymbol{w}} H(C|\boldsymbol{F}_i) = \arg\min_{\boldsymbol{w}} H(C|W_i^T\boldsymbol{X}). \tag{13}$$

After computing $H(C|W_i^T\boldsymbol{X})$ with (12), the gradient based algorithms such as gradient descent method can be used to find the solution. The detailed process for Steps 2 and 3 is described in the following subsections.

Regarding how to determine the number of extracted feature M, one can use the mutual information as a criterion. For example, in Step 3, if the mutual information $I(\boldsymbol{F}^*_k; C)$ does not increase a lot for the addition of a new feature F_k, feature extraction can be stopped and M can be determined to be $k-1$.

B. Decorrelation of data with PCA

In order to alleviate the computational complexity in applying the Parzen window method, PCA is utilized in Step 2 as follows:

Step 2. (Sphering by PCA) Transform the original features $\boldsymbol{X}$ into $\boldsymbol{Y} = W_{pca}^T\boldsymbol{X}$ to have a zero mean and an $N' \times N'$ identity covariance matrix; $\Sigma_{\boldsymbol{Y}} = I_{N'}$.

Since a matrix inversion is used in the calculation of $H(C|\boldsymbol{F}_i)$, it would be better to have the covariance matrix, $\Sigma_{\boldsymbol{F}_i}$, to take a special form. To this end, the N dimensional feature vector, $\boldsymbol{X}$, is first transformed into an $N'(\leq N)$ dimensional vector $\boldsymbol{Y} = W_{pca}^T\boldsymbol{X}$ using PCA. Note that the rank N' of W_{pca} can be equal to N if $\Sigma_{\boldsymbol{X}}$ is nonsingular. Also note that for a dataset with a large number of original features, this step serves as a filtering step which reduces the dimensionality from N to N'.

After performing PCA, the covariance matrix of $\boldsymbol{Y}$ becomes an N' dimensional identity matrix; $\Sigma_{\boldsymbol{Y}} = I_{N'}$. If $\boldsymbol{Y}$ is used instead of $\boldsymbol{X}$, the problem in (13) then becomes finding successive $\boldsymbol{v}_i^* \in \Re^{N'}$, $i = 1, \cdots, M$, such that

$$\boldsymbol{v}_i^* = \arg\min_{\boldsymbol{v}} H(C|\boldsymbol{F}_i) = \arg\min_{\boldsymbol{v}} H(C|V_i^T \boldsymbol{Y}),$$

where $V_i \triangleq [\boldsymbol{v}_1^*, \cdots, \boldsymbol{v}_{i-1}^*, \boldsymbol{v}]$.

The search space of $\boldsymbol{w}$ is restricted to an N' dimensional subspace such that $\boldsymbol{w} = W_{pca}\boldsymbol{v}$, and the i-th extracted feature F_i becomes

$$F_i = \boldsymbol{v}_i^{*T}\boldsymbol{Y} = \boldsymbol{v}_i^{*T} W_{pca}^T \boldsymbol{X}.$$

Henceforth, the covariance matrix of $\boldsymbol{F}_i$ becomes

$$\Sigma_{\boldsymbol{F}_i} \triangleq E\{\boldsymbol{F}_i \boldsymbol{F}_i^T\} = V_i^T \Sigma_Y V_i = V_i^T V_i.$$

In order to simplify the process of taking the inverse of $\Sigma_{\boldsymbol{F}_i}$, $\boldsymbol{v} \in \Re^{N'}$, the last column of V_i, is restricted to the orthogonal direction to all the other $(i-1)$ columns of V_i. In addition, because the scaling of $\boldsymbol{v}$ does not change the values of $\hat{p}(c|V_i^T\boldsymbol{y})$ and $\hat{H}(C|V_i^T\boldsymbol{Y})$ in (11) and (12), $\boldsymbol{v}$ are always normalized such that $\boldsymbol{v}^T\boldsymbol{v} = 1$. Subsequently, the covariance matrix of $\boldsymbol{F}_i$ becomes an i dimensional identity matrix I_i. This orthonormalization of the weight matrix alleviates the computational effort greatly by avoiding the process of matrix inversion in the calculation of the conditional entropy (12) and its derivative.

C. Feature extraction using Parzen window

In this part, Step 3 of the greedy extraction algorithm described in Section 3.1 is elaborated. After preprocessing by PCA described in the previous subsection, features with maximum mutual information with the class is added one by one in Step 3. In this process, the mutual information is calculated in the same way as in Section 2 by using the density estimation by the Parzen window method. Step 3 is carried out as follows:

Step 3. (Greedy extraction) For $i = 1, \cdots, M$, repeat the following.

(a1) (Randomize weight) Generate an N' dimensional random weight $\boldsymbol{v}$.

(a2) (Orthonormalization) Orthonormalize the weight $\boldsymbol{v}$ by the Gram-Schmidt method;

$$\begin{aligned} \boldsymbol{v} &\longleftarrow \boldsymbol{v} - \sum_{j=1}^{i-1} (\boldsymbol{v}^T\boldsymbol{v}_j^* / ||\boldsymbol{v}_j^*||^2)\boldsymbol{v}_j^* \\ \boldsymbol{v} &\longleftarrow \boldsymbol{v}/||\boldsymbol{v}||. \end{aligned} \tag{14}$$

(a3) (Weight update) Update weight by the gradient descent method;

i. (Gradient calculation) Calculate $\nabla_{\boldsymbol{v}} H(C|V_i^T\boldsymbol{Y})$.

ii. (Weight update) Reserve the old weight and update the weight with the learning rate μ.

$$\begin{aligned} \boldsymbol{v}_{old} &\longleftarrow \boldsymbol{v} \\ \Delta\boldsymbol{v} &\longleftarrow -\mu\nabla_{\boldsymbol{v}} H(C|V_i^T\boldsymbol{Y}) \\ \boldsymbol{v} &\longleftarrow \boldsymbol{v} + \Delta\boldsymbol{v}. \end{aligned}$$

iii. (Orthonormalization) Orthonormalize the weight $\boldsymbol{v}$ to make $\Sigma_{\boldsymbol{F}_i} = E\{\boldsymbol{F}_i\boldsymbol{F}_i^T\} = I_i$, using procedure (14).

iv. (Convergence check) If $||\boldsymbol{v} - \boldsymbol{v}_{old}|| < \epsilon$ or the number of iterations reaches MAX_ITER, go to Step (b). Otherwise, go to Step 3(a3)-i.

(b) (Extraction of the next feature) Set $\boldsymbol{v}_i^* = \boldsymbol{v}$, $F_i = \boldsymbol{v}_i^{*T}\boldsymbol{Y}$ and $\boldsymbol{w}_i^* = W_{pca}\boldsymbol{v}_i^*$. $\mathcal{F} \longleftarrow \mathcal{F} \cup \{F_i\}$.

In Step 3(a3)-ii, we use the gradient descent method, but we may use other optimization methods such as conjugate gradient or Levenberg-Marquardt. The objective function may be highly nonlinear and the solution may converge to a local minimum. To alleviate this problem, the learning rate μ can be set to higher values in earlier iterations and then scaled down as the number of iterations increases. In addition, in Step 3(a1), the conditional entropy $H(C|V_i^T\boldsymbol{Y})$ can be calculated for various initial weight $\boldsymbol{v}$'s and we can select the best $\boldsymbol{v}$ that results in the minimum value of $H(C|V_i^T\boldsymbol{Y})$.

D. *Calculation of the gradient*

Before discussing the calculation of the gradient in Step 3(a3)-i in detail, let us consider how to choose the window width parameter, h, for different number of extracted features. In the calculation of the conditional probability (11), $(\boldsymbol{f} - \boldsymbol{f}_i)^T\Sigma_{\boldsymbol{F}}^{-1}(\boldsymbol{f} - \boldsymbol{f}_i)$ increases in proportion to the number of extracted features when $\Sigma_{\boldsymbol{F}} = I$. To make sure that $\phi(\boldsymbol{f} - \boldsymbol{f_i}, h)$ does not change too much for a given sample point $\boldsymbol{f} = \boldsymbol{f_j}$ when the dimension of $\boldsymbol{f}$ increases, the denominator $2h^2$ of the exponential should also be increased. Thus we set $\hbar_k = \hbar_1\sqrt{k}$ where $\hbar_k$ denotes h when the dimension of $\boldsymbol{F}$ is k. This worked well for the experiments reported in Section 4.

All that remains is the calculation of the gradient $\nabla_{\boldsymbol{v}} H(C|V_i^T\boldsymbol{Y}) \in \Re^{N'}$ in Step 3(a3)-i. Replacing $\boldsymbol{F}$ with $V_i^T\boldsymbol{Y}$, (12) becomes

$$\hat{H}(C|V_i^T\boldsymbol{Y}) = -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\hat{p}(c|V_i^T\boldsymbol{y}_j)\,\log\hat{p}(c|V_i^T\boldsymbol{y}_j),$$

where $\boldsymbol{y}_j$ represents the j-th sample of the original data transformed by PCA.

The last column of V_i can be obtained by differentiating this with respect to $\boldsymbol{v}$,

$$\begin{aligned} &\nabla_{\boldsymbol{v}}\hat{H}(C|V_i^T\boldsymbol{Y}) \\ &= -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\{\nabla_{\boldsymbol{v}}\hat{p}(c|V_i^T\boldsymbol{y}_j)\,\log\hat{p}(c|V_i^T\boldsymbol{y}_j) + \nabla_{\boldsymbol{v}}\hat{p}(c|V_i^T\boldsymbol{y}_j)\} \\ &= -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\nabla_{\boldsymbol{v}}\hat{p}(c|V_i^T\boldsymbol{y}_j)\{1 + \log\hat{p}(c|V_i^T\boldsymbol{y}_j)\}. \end{aligned} \tag{15}$$

TABLE I

CLASSIFICATION PROBLEMS

Dataset		No. of classes	No. of features	No. of samples	Performance test
Linearly inseparable		2	4	1000	train/test (500/500)
UCI	sonar	2	60	208	13-fold cross validation
	wine	3	13	178	train/test (90/88)
face	Yale	15	21 × 30	165	leave-one-out
	AT&T	40	23 × 28	400	leave-one-out

Rewriting (11) leads to

$$\hat{p}(c|V_i^T\boldsymbol{y}) = \frac{\sum_{l\in I^c}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)}{\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)},$$

where, $\varphi(\boldsymbol{z}) \triangleq \exp(-\frac{\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z}}{2h^2})$ and $\tilde{\boldsymbol{y}}_l \triangleq \boldsymbol{y} - \boldsymbol{y}_l$.

By differentiating this w.r.t. $\boldsymbol{v}$, the following is obtained:

$$\begin{aligned}\nabla_{\boldsymbol{v}}\hat{p}(c|V_i^T\boldsymbol{y}) =& \frac{\sum_{l\in I^c}\nabla_{\boldsymbol{v}}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)}{\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)} \\ &- \frac{[\sum_{l\in I^c}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)][\sum_{k=1}^{N_c}\sum_{l\in I^k}\nabla_{\boldsymbol{v}}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)]}{[\sum_{k=1}^{N_c}\sum_{l\in I^k}\varphi(V_i^T\tilde{\boldsymbol{y}}_l)]^2}.\end{aligned}$$

Since the new feature candidates are orthogonal to the previously extracted features in Step 3(a3)-iii, $\Sigma_{\boldsymbol{F}_i} = I_i$. Replacing Σ in $\varphi(\cdot)$ with the identity matrix, and differentiating it w.r.t. $\boldsymbol{v}$, $\nabla_{\boldsymbol{v}}\varphi(V_i^T\tilde{\boldsymbol{y}}_l) \in \Re^{N'}$ can be obtained as follows:

$$\begin{aligned}\nabla_{\boldsymbol{v}}\varphi(V_i^T\tilde{\boldsymbol{y}}_l) &= \nabla_{\boldsymbol{v}}\exp(-\frac{\tilde{\boldsymbol{y}}_l^T V_i V_i^T\tilde{\boldsymbol{y}}_l}{2h^2}) \\ &= -\frac{1}{h^2}\exp(-\frac{\tilde{\boldsymbol{y}}_l^T V_i V_i^T\tilde{\boldsymbol{y}}_l}{2h^2})(\tilde{\boldsymbol{y}}_l\boldsymbol{v}^T\tilde{\boldsymbol{y}}_l).\end{aligned}$$

Finally, the calculation of (15) is completed and the PWFX algorithm can be implemented.

As in PWFS, the computational complexity of PWFX is proportional to the square of the number of samples, n^2, and the dimension of the input feature space, N, in addition, it is also proportional to the number of iterations.

IV. Experimental Results

In this section, PWFX was applied to several classification problems shown in Table I and the classification performance was compared with those of other methods which are either popular or give good performance.

In the proposed feature extraction algorithm, deciding the window width parameter, h, is very important. Not knowing the optimal value for the width parameter, we tried PWFX for several values of $\hbar_1$ and found out that $\hbar_1 = 0.1 \sim 0.5$ was generally acceptable. We set $\hbar_1 = 0.3$ for all the experiments below. In addition, in order to expedite the computation, the influence range of a sample point was restricted to $2\sigma \cdot h$ for each dimension, i.e., the influence was set to zero in the outer domain of $2\sigma \cdot h$ from the sample point. Here σ is the standard deviation of the corresponding feature and is always equal to 1 in the simulation. This reduces the computational effort greatly.

In all the experiments, the learning rate, μ, was set to 0.1 and the convergence parameter, ϵ, in Step 3(c)-iv was set to 10^{-3} and MAX_ITER was set to 300.

A. Simple linearly inseparable problem

Suppose there are four independent input features, $x_1 \sim x_4$, uniformly distributed on [-1,1] for a binary classification, and the output class, c, is determined as follows:

$$c = \begin{cases} 0 & \text{if } |x_1 + 2x_2| < 1 \\ 1 & \text{if } |x_1 + 2x_2| \geq 1. \end{cases}$$

Although this problem is linearly inseparable, it can still be classified with one feature if $x_1 + 2x_2$ is extracted as a new feature.

For this problem, 1,000 samples were randomly generated and LDA, ICA-FX and PWFX were performed on this dataset. The training and test sets contained 500 samples each. For this dataset, the normalized weights corresponding to the first features extracted by LDA, ICA-FX and PWFX are $\boldsymbol{w}_{lda} = [-0.42, 0.34, 0.83, -0.17]^T$, $\boldsymbol{w}_{icafx} = [0.52, -0.41, -0.58, 0.49]^T$ and $\boldsymbol{w}_{pwfx} = [0.46, 0.88, -0.02, 0.00]^T$ respectively. We can see that $\boldsymbol{w}_{PWFX}$ is very close to the optimal weight $\boldsymbol{w}^* = [1, 2, 0, 0]^T$.

Table II shows the classification results of the feature extraction algorithms with various numbers of extracted features. Multi-layer perceptron (MLP) with one hidden layer and three hidden nodes was used for the classification. The learning rate, momentum, and the number of iterations were set to 0.1, 0.9, and 100 respectively. In the table, the results are the averages of 10 experiments and the numbers in the parentheses are the standard deviations. Because there are two classes, the LDA extracts only one feature for this problem. The classification performances of the LDA and ICA-FX can be seen to be relatively poor giving almost 50% of error rates for most cases. In contrast, the PWFX is seen to perform well for this problem.

Since the LDA assumes that the distribution of each class has one peak like a Gaussian distribution, it does not perform well if the centers of the classes are not separated well. For this problem, the centers of 'class 0' and 'class 1' are both approximately at (0,0). Likewise, the ICA-FX cannot effectively deal with the linearly inseparable problems. On the other hand, the PWFX does not make any assumption on the class distribution because it makes use of the Parzen window, and thus can better deal with inseparable problems. Note that the performance of the PWFX does not vary much as the number of extracted features increases. This suggests that the first feature contains almost

TABLE II

CLASSIFICATION PERFORMANCE FOR THE INSEPARABLE $|x_1 + 2x_2| \gtrless 1$ PROBLEM.

No. of	Classification error (%) (MLP)		
features	LDA	ICA-FX	**PWFX**
1	49.44 (0.08)	49.80 (0.21)	**3.96 (0.26)**
2	–	49.42 (0.38)	**2.72 (0.19)**
3	–	45.92 (0.81)	**3.98 (0.06)**
4	–	39.48 (0.90)	**4.38 (0.11)**

all the information on the class which was expected as the first weight of PWFX is very close to the optimal weight.

B. Sonar target dataset

The sonar target classification problem is described in [20]. The aspect angle independent dataset were used in this experiment. The classification performances of the PWFX were compared with those of the LDA and ICA-FX for various numbers of extracted features.

A thirteen-fold cross validation was used to obtain the performances, and the MLP, C4.5, and SVM were used as classifier systems. For all the classifiers, input values of the data were normalized to have zero means and standard deviations of one. In training the MLP, the standard BP algorithm was used with three hidden nodes, two output nodes, a learning rate of 0.05 and a momentum of 0.95. The neural networks were trained for 1,000 iterations. The C4.5 parameters were set to default values in [21]. For the SVM, the radial (Gaussian) kernel was used and the other parameters were set to the default values [22]. Because the performance of the radial kernel SVM depends critically on the parameter γ, the SVM was conducted with various values of γ ranging from 0.01 to 1, and the reported classification performance on the test data corresponds to the best γ that showed the best classification performance on the training data.

Table III shows the experimental results. The results of the LDA, and ICA-FX were obtained from [13]. For comparison, the performances using all the 60 original features are also reported in the table. To save the space, this is reported in the last row on the LDA column.

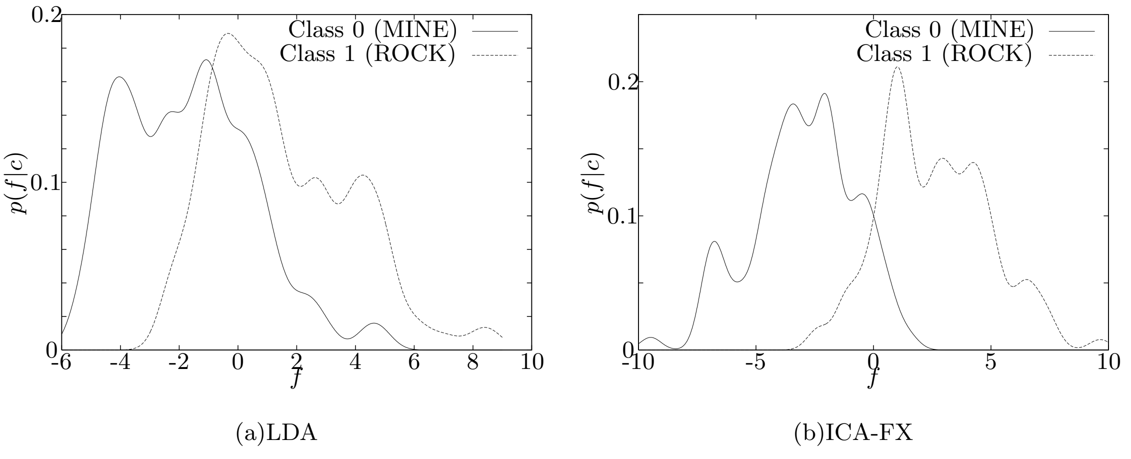
The performance for MLP is an average of the 10 experiments and the numbers in the parentheses denote its standard deviation. The PWFX can be seen to perform far better than the other methods regardless of the classifying system and with only 3 or 6 features. These results indicate that most of the information on the class is contained in the first few features of the PWFX.

To observe the characteristics of the PWFX, the probability density of the first features of LDA, ICA-FX with $M = 1$, and PWFX are plotted in Fig. 1. These are estimates of the conditional densities $p(f|c)$'s (class-specific density estimates) calculated by the Parzen window method with $h = 0.2$. In Figures 1 (a),(b), and (c), if the domain for $p(f|c = 0) \neq 0$ and the domain for

TABLE III

Classification performance for Sonar Target data. (The last row of the LDA column shows the performance using all the 60 original features.)

No. of	Classification performance (%) (C4.5/MLP/SVM)		
features	LDA	ICA-FX	PWFX
1	71.2/75.2(0.37)/74.1	87.5/87.3(0.17)/87.1	**90.4/91.8(0.14)/90.8**
3	–	86.1/88.1(0.37)/89.0	**95.7/92.3(0.15)/95.7**
6	–	85.6/86.4(0.42)/87.1	**95.7/91.1(0.23)/94.2**
9	–	83.2/85.0(0.83)/88.8	**96.6/89.9(0.30)/92.3**
12	–	78.2/83.4(0.49)/86.6	**96.6/89.2(0.24)/92.3**
60	73.1/76.4(0.89)/82.7	73.1/80.0(0.78)/84.2	**96.3/86.7(0.56)/88.2**



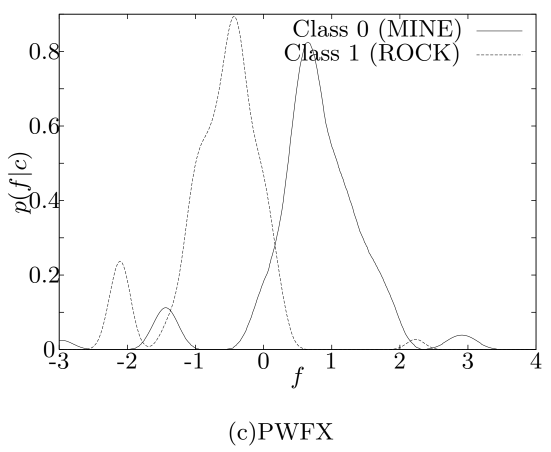


Fig. 1. Probability density estimates for a given feature.

TABLE IV

Classification performance for Wine data. (The last row of the LDA column shows the performance using all the 13 original features.)

No. of	Classification performance (%) (C4.5/MLP)		
features	LDA	ICA-FX	PWFX
1	56.81/56.81(0.00)	64.77/63.63(0.00)	**94.31/93.18(0.00)**
2	56.81/96.01(0.97)	63.63/95.11(1.86)	**97.72/96.59(0.54)**
3	–	78.40/95.45(0.76)	**97.72/96.59(0.54)**
13	79.15/85.23(2.12)	87.50/94.31(1.42)	**97.72/96.59(0.76)**

$p(f|c = 1) \neq 0$ do not overlap, there will be no error in the classification. It can be seen that the overlapping region of the two classes is the smallest in PWFX and the largest in LDA. Unlike the other methods, the domain where $p(f|c = 1)$ is greater than $p(f|c = 0)$ is comprised of several separated regions in the PWFX, i.e., the regions around [-3,-2.5], [-1.8,-1.4], and [0.2,2.2]. The PWFX searches for a direction that directly maximizes the mutual information, whereas the LDA and ICA-FX implicitly assume that the centers of each class are well separated. Therefore, it is expected that the PWFX will perform better than the LDA and ICA-FX for non-separable classification problems.

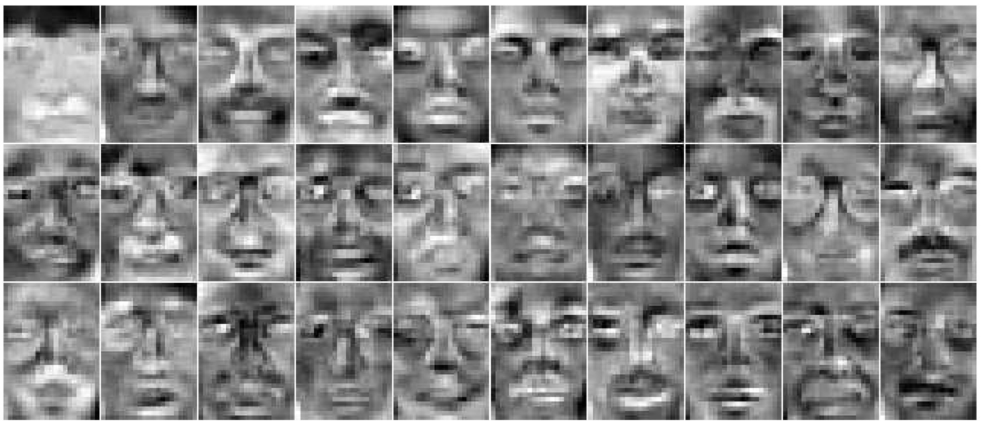
Addison et al. [23] reported that the error rates of MLP with $47 \sim 60$ features for the sonar data are $2 \sim 5\%$. However, these results cannot be directly compared to those of PWFX because the experimental settings [23] might be different.

C. Wine dataset

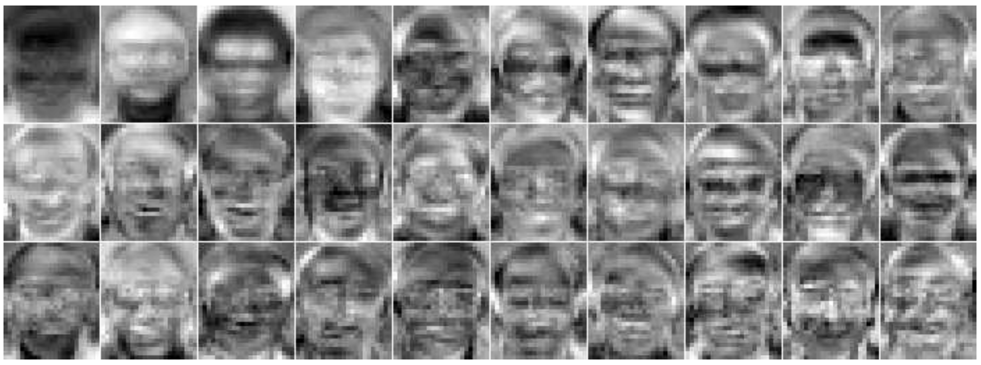
The wine data are the results of a chemical analysis of wines grown in the same region in Italy but were derived from three different cultivars [24]. Each sample is composed of quantities of the 13 constituents. There are 178 samples of which 59, 79, and 48 samples are in *Class 1*, *Class 2*, and *Class 3*, respectively. Among each class, 30 samples were used as the training data and the remaining ones were reserved for the test data.

Table IV shows the performance of the LDA, ICA-FX and PWFX methods. C4.5 and the MLP were used as classifier systems. The default parameters were used for C4.5, and a learning rate of 0.1 and a momentum of 0.9 were used for the MLP. The number of maximum iterations was set to 300 and the number of hidden nodes was set to 3 for the MLP with one hidden layer. For comparison, the performances using all the 13 original features are also reported in the table. To save the space, this is reported in the last row on the LDA column.

The performance of the PWFX can be seen to be much better than those of the other methods by approximately 30% when one feature was extracted. This gap decreases when more than 2 features were extracted. Also, note that the PWFX worked well on both classifiers, whereas the others did not.



(a)Yale



(b)AT&T

Fig. 2. Weights of various methods for the Yale and AT&T datasets (1st row: LDA (Fisherfaces), 2nd row: ICA-FX, 3rd row: PWFX.)

D. Face recognition problem

In this subsection, the PWFX, LDA and ICA-FX were applied to the face recognition problem on the Yale [25] and AT&T [26] face databases.

To obtain the original features, $\boldsymbol{X}$, each image was downsampled into a manageable size. Each downsampled pixel was transformed to have a zero mean and a unit variance. The PCA was then performed both as a whitening process and for the purpose of further reducing the dimensions of the feature space. Finally, the PWFX was applied to extract the valuable features for the classification. As a comparison, the LDA and ICA-FX were also used after the PCA was performed. The performance was tested using the leave-one-out scheme and the classification was performed using the nearest neighborhood classifier.

The Yale face database consisted of 165 grayscale images of 15 individuals and the AT&T database consisted of 400 images, which were ten different images for 40 distinct individuals. The Yale face images were downsampled into 21×30 pixels and the AT&T images were downsampled into 23×28

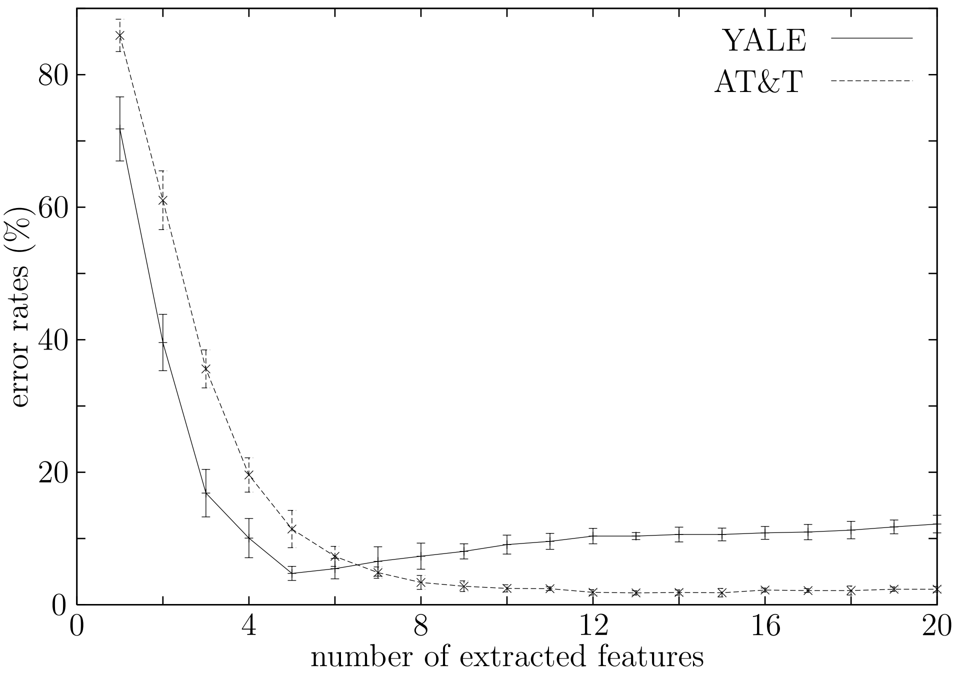


Fig. 3. Error rates of PWFX for Yale and AT&T Databases.

TABLE V

Error rates of various feature extraction methods.

	Method	Dim. of Reduced Space	No. of Errors	Error Rate (%)
YALE	LDA	14	14	8.48
	ICA-FX	10	7	4.24
	PWFX	**5**	**7.8**	**4.73 (1.06)**
AT&T	LDA	39	16	4.00
	ICA-FX	10	4	1.00
	PWFX	**13**	**7.2**	**1.80 (0.32)**

pixels. For the original features, the largest 30 principal components from the 630 pixels were used for the Yale, and 40 principal components from 644 pixels were used for the AT&T dataset.

Figures 2 (a) and (b) represent the typical weights of LDA, ICA-FX, and PWFX for the Yale and AT&T data. From the top, each row shows the first 10 weights of LDA, ICA-FX, and PWFX, respectively.

Figure 3 shows the error rates of the PWFX averaged on the ten experiments with various numbers of extracted features. The bars at each data point denote the standard deviations of the ten experiments. In that figure, the performances of the PWFX for the Yale and AT&T databases are best when the number of extracted features is 5 and 13, respectively.

In Table V, the performance of the PWFX is slightly worse than that of the ICA-FX, but is better than those of the LDA. The results other than those of the PWFX were obtained from [27]. For the AT&T data, Yang [28] reported the error rates of the kernel Fisherface method with 39 features and kernel eigenface method with 40 features are 1.25% and 2.0%, respectively. The error rate of the PWFX is nearly the same, but the number of features used is about one third of those in [28].

V. Conclusions

This paper proposes a new feature extraction method for dealing with classification problems. The proposed method provides a way of using the mutual information directly in extracting features for classification problems by overcoming computational limitations.

Although the restriction that the newly extracted feature of the PWFX is orthogonal to the already extracted features is unnecessary, it greatly simplifies the calculation of the gradient. Once the gradient was calculated, the gradient descent method was used to maximize the mutual information and the greedy extraction scheme was used to determine the new features. The proposed algorithm does not make any assumption with respect to the distribution of features, and this fits well for complex classification problems where the classes have multiple peaks in the feature space. The computational complexity of the proposed method is proportional to the square of the sample size and it is also proportional to the size of the input space. In the bench mark problems, it showed better or comparable performances than those of other conventional methods such as the LDA and ICA-FX. This shows that the PWFX can extract good features for classification problems.

But, there still remains several issues to be dealt with in future work. These include the method of feature extraction based on direct calculation of Bayes error, the resolution of the computational complexities for the problems in high dimensional input space, and the study of the effect of feature extraction on the required amount of training data.

Acknowledgments

This work was supported by the new faculty research fund of Ajou University (20071040).

References

[1] P.A. Devijver and J. Kittler, *Pattern recognition: A statistical approach*, Prentice Hall, London, 1982.

[2] K.J. Cios, W. Pedrycz, and R.W. Swiniarski, *Data mining methods for knowledge discovery*, chapter 9, Kluwer Academic Publishers, 1998.

[3] I.T. Joliffe, *Principal Component Analysis*, Springer-Verlag, 1986.

[4] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, second edition, 1990.

[5] A.J. Bell and T.J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, June 1995.

[6] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[7] S. Ullman, M. Vidal-Naquet, and E. Sali, "Visual features of intermediate complexity and their use in classification," *Nature Neuroscience*, , no. 7, pp. 682–687, July 2002.

[8] J.W. Fisher III and J.C. Principe, "A methodology for information theoretic feature extraction," in *Proc. Int'l Joint Conf. on Neural Networks 1998*, Anchorage, Alasca, May 1998.

[9] P. Viola and W.M. Wells III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[10] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on Parzen window," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1667–1671, Dec. 2002.

[11] N. Kwak and C.-H. Choi, "A new method of feature extraction and its stability," in *Proc. Int'l Conf. on Artificial Neural Networks 2002*, Madrid Spain, Aug. 2002, pp. 480–485.

[12] N. Kwak, C.-H. Choi, and N. Ahuja, "Face recognition using feature extraction based on independent component analysis," in *Proc. Int'l Conf. on Image Processing 2002*, Rochester, NY, Sep. 2002, pp. 337–340.

[13] N. Kwak and C.-H. Choi, "Feature extraction based on ica for binary classification problems," *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1374–1388, Nov. 2003.

[14] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL, 1949.

[15] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[16] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statistics*, vol. 33, pp. 1065–1076, Sept. 1962.

[17] M.E. Hellman and J. Raviv, "Probability of error, equivocation and chernoff bound," *IEEE Trans. on Information Theory*, vol. 16, pp. 368–372, 1970.

[18] G.A. Babich and O.I. Camps, "Weighted Parzen window for pattern classification," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 18, no. 5, pp. 567–570, May 1996.

[19] K. Fukunaga and R.R. Hayes, "The reduced Parzen classifier," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 11, no. 4, pp. 423–425, April 1989.

[20] R.P. Gorman and T.J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.

[21] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[22] Stefan Ruping, "mysvm – a support vector machine," For more information contact http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

[23] D. Addison, S. Wermter, and G. Arevian, "A comparison of feature extraction and selection techniques," in *Proc. Int'l Conf. on Artificial Neural Networks 2003*, Istanbul, Turkey, June 2003, pp. 212–215.

[24] S. Aeberhard, D. Coomans, and O. de Vel, "The classification performance of rda," Tech. Rep. 92-01, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, 1992.

[25] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, July 1997.

[26] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, Dec. 1994.

[27] N. Kwak, *Feature selection and extraction based on mutual information for classification*, Ph.D. thesis, Seoul National University, Seoul, Korea, 2003.

[28] M.H. Yang, "Kernel eigneface vs. kernel fisherface: Face recognition using kernel methods," in *Proc. 5th Int'l Conf. Automatic Face Gesture Recognition*, Washington D.C., May 2002, pp. 215–220.