工學博士學位論文

Feature Selection and Extraction Based on Mutual Information for Classification
분류를 위한 공유정보 기반 특징 선택 및 추출

2003年 2月

서 울 大 學 校　大 學 院

電氣·컴퓨터 工學部

郭　魯　濬

Feature Selection and Extraction Based on Mutual Information for Classification

분류를 위한 공유정보 기반 특징 선택 및 추출

指導敎授 崔 悰 鎬

이 論文을 工學博士學位論文으로 提出함

2002年 10月

서울大學校 大學院
電氣·컴퓨터 工學部
郭 魯 濬

郭 魯 濬의 工學博士學位論文을 認准함

2002年 12月

委 員 長	최 진 영	(인)
副委員長	최 종 호	(인)
委 員	장 병 탁	(인)
委 員	조 남 익	(인)
委 員	이 창 수	(인)

Abstract

The advent of internet and development of new computer technologies made it easy to create huge databases with a large number of features that are sometimes called as 'attributes' or 'fields'. Among these, there are features that are relevant or irrelevant to the concerning problem and there may be redundant ones also. From the viewpoint of managing and analyzing a database, reducing the number of features by selecting only the relevant ones or extracting new features, which are relevant to the problem, from the original ones is desirable. Using only problem-relevant features, the dimension of the feature space can be greatly reduced in line with the principle of parsimony, resulting better generalization.

This thesis deals with the problem of feature selection and extraction for classification problems. Throughout the thesis, mutual information is used as a measure of correlation between class labels and features. In the first part of the dissertation, the feature selection problem is studied and a new method of feature selection is proposed. In order to calculate the mutual information between input features and class labels, a new method based on the Parzen window is proposed, and it is applied to a greedy feature selection algorithm for classification problems. In the second part, the feature extraction problem is dealt with and a new method of feature extraction is proposed. It is shown how standard algorithms for independent component analysis (ICA) can be appended with class labels to produce a number of features that carry whole the information about the class labels that was contained in the original features. A local stability analysis of the proposed algorithm is also provided. The advantage of the proposed method is that general ICA algorithms become available to a task of feature extraction for classification problems by maximizing the joint mutual information between the class labels and the new features. Using the new features, the dimension of the feature space can be greatly reduced without degrading the classification performance.

The proposed feature selection and extraction methods are applied to various pattern recognition problems such as face recognition and the performances of the proposed methods are compared with those of other conventional methods. The experimental results show that the proposed methods outperform the other

methods using small numbers of features.

Keywords: Feature selection, feature extraction, dimensionality reduction, Parzen window, mutual information, independent component analysis, face recognition, classification, data mining, pattern recognition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been an explosive growth in men's capability to generate and collect data. According to some estimates, the amount of data in the world is doubling every twenty months [1]. Consequently, it is becoming more and more important to interpret, digest, and analyze this data in order to extract useful knowledge out of them. Therefore, there exists a significant need for new techniques and tools with the ability of assisting human beings intelligently and automatically in analyzing mountains of data for nuggets of knowledge. The overall efforts toward this ends are collectively referred to as *knowledge discovery in database* (KDD) [2].

Recently, the term *data mining* is widespread to refer to the subject [2] [3] [4]. There are arguments whether the term should be used in a narrow sense to indicate a single step in the KDD process that involves finding patterns in the data as in [2], where it is defined as a step in the KDD process consisting a particular algorithms that produces a particular enumeration of patterns over data, or it should be used in a wide sense to refer to the entire process of KDD [3], where it is defined as the entire process of discovering advantageous patterns in data. Especially, all the pattern recognition problems can be viewed as data mining problems in a wide sense. Both viewpoints are adopted in this dissertation and if necessary, it will be clearly indicated whether the term is used in a wide sense or in a narrow sense in the following.

There are several ways of classifying data mining processes. One is to cate-

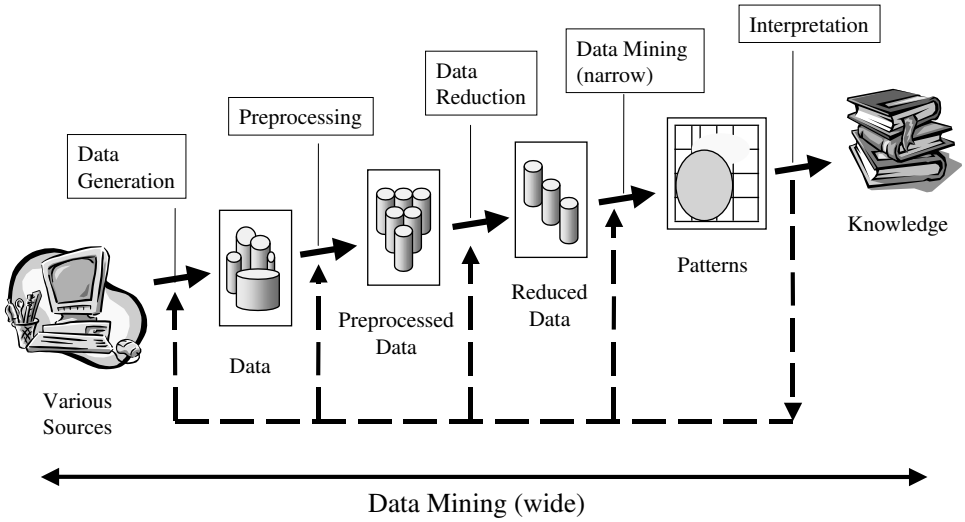


Figure 1.1: Data Mining Process

gorize the processes sequentially. The other is to categorize them by the characteristics of the problem in question. For the first case, there exist various steps in the data mining process and Fig. 1.1 shows typical steps in the data mining process. It is a modified version of the Figure 1.3 in [2]. Broad outlines of their basic functions are as follows:

1. Data generation: generating or collecting domain specific data that contain information about the goals of the end-user, on which discovery is to be performed.

2. Preprocessing: basic operations such as the removal of noise or outliers if appropriate. Normalization or quantization may occur in this step.

3. Data reduction: finding useful features or samples to represent the data depending on the goal of the task. Dimensionality reduction or sample selection methods are used in this step to reduce the effective number of variables or samples under consideration or to find invariant representation of the data.

4. Data mining (narrow sense): choosing and applying an appropriate algorithms in searching for the patterns in the data, depending on whether the goal of the KDD process is classification, regression, clustering, etc. The resultant patterns may be classification rules or trees, regression equations, clusters, etc.

5. Interpretation: interpreting and evaluating the resultant patterns to get the knowledge. After evaluating the performance, any of the previous steps may be revisited for the next iteration.

Among the five steps above, *data reduction* and *data mining* steps have been studied actively and many algorithms have been proposed on these areas. For example, neural networks, decision trees, example based methods such as nearest neighborhood classifiers, and many statistical methods are in popular usage in the data mining step. On the other hand, feature selection, feature extraction, and sample selection methods are typically used for the data reduction.

For the second case, the entire data mining process can be divided into *supervised* and *unsupervised* learning depending whether there are target (output) values or not. In supervised learning, one tries to investigate the relationship between the inputs and the targets using given input-target (output) patterns. On the other hand, in unsupervised learning, there are no distinction between attributes and the purpose is to investigate the underlying structure of the data such as the distribution of the data or the minimum variance direction and these are mostly related to the *clustering* problems. Supervised learning can be further divided into the *classification* and the *regression* problems depending on the characteristics of the target values. Typically it is classified as a classification problem if the targets are categorical, while it is referred to as a regression problem if the targets are continuous numerical values.

In this dissertation, the main focus is on the *data reduction step* and the *feature selection and extraction for classification problems* are extensively studied.

1.1 Feature Selection and Extraction

In data mining problems, one is given an array of attributes or data fields to search for the underlying patterns in the data. These attributes are called *features*, and there may exist irrelevant or redundant features to complicate the learning process, thus leading to an erroneous result. Even when the features presented contain enough information about the problem, the resultant patterns after the data mining process may not be relevant because the dimension of feature space can be so large that it may require numerous instances to investigate the patterns. This problem is commonly referred to as the *curse of dimensionality* [5]. Especially in supervised learning, where the purpose is to investigate the input-output relationship and predict output class using input features, some experiments have also reported that the performance of classifier systems deteriorates as irrelevant features are added [3].

Though some of the modern classifiers, such as the support vector machines (SVM), are surprisingly tolerant to extra irrelevant information, this problem can be avoided by selecting only the relevant features or extracting new features containing the maximal information about the problem in question from the original ones. The former methodology is called as the *feature selection* or the *subset selection*, while the latter is named as the *feature extraction* which includes all the methods that takes any functions, logical or numerical, of the original features to extract new features.

Reduction of pattern dimensionality may improve the data mining process by considering only compact, the most important data representation, possibly with elements retaining maximum information about the original data and with better generalization abilities [6]. Not only in the aspect of curse of dimensionality, but also in the viewpoint of data storage and computational complexity, dimensionality reduction through feature selection or extraction is quite desirable.

1.2 Previous Works for Feature Selection

Feature selection is usually defined as a process of finding a subset of features, from the original set of features forming patterns of a given data, optimal

according to the goal and criterion of feature selection [6].

Feature selection algorithms can be classified as a *filter* model or a *wrapper* model, depending on whether it is treated as a preprocess or interwinded with the learning task. More precisely, in a filter model, the data mining (in narrow sense) process is performed after the features are selected, while in a wrapper model, features are selected in the process of data mining. A wrapper approach generally outperforms a filter model because it directly optimizes the evaluation measure of the learning task while removing irrelevant features, but the time needed to complete feature selection is much longer than that of a filter approach [4].

The feature selection problem has been dealt with intensely, and some solutions have been proposed [7] – [15]. Among these, one of the most important contributions has been made using the decision tree method. This method can be classified as a wrapper model and it uncovers relevant attributes one by one iteratively [13], [14]. Setiono and Lui [13] proposed a feature selection algorithm based on a decision tree by excluding the input features of the neural network one by one and retraining the network repeatedly. It has many attractive characteristics, but it basically requires a process of retraining for almost every combinations of input features. To overcome this shortcoming, a fast training algorithm other than the BP (back-propagation) is used, but nevertheless it requires a considerable amount of time. The CDP (classifier with dynamic pruning) of Agrawal et al. [15] is also based on the decision tree which makes use of the mutual information between inputs and outputs. It is very efficient in finding rules that map inputs to outputs, but as a downside, requires a great deal of memory because it generates and counts all the possible input-output pairs. MIFS (mutual information feature selector) by Battiti [7] uses mutual information between inputs and outputs like the CDP but it is a filter method. Batitti demonstrated that mutual information can be very useful in feature selection problems, and the MIFS can be used in any classifying systems for its simplicity whatever the learning algorithm may be. Because the computation of mutual information between continuous variables is a very difficult job requiring probability density functions (*pdf*) and involving integration of those functions, Battiti used histograms to avoid these complexities. Thus, the performance can be degraded as a result of large er-

rors in estimating the mutual information. Kwak and Choi [8], [9] proposed an extended version of MIFS that can provide better estimation of mutual information between inputs and outputs. Though easy to implement without degrading the performance much, the MIFS methods have another limitation in that these methods do not provide a direct measure to judge whether to add additional features or not. More direct calculation of mutual information is attempted using the quadratic mutual information in [16] – [18].

Regarding the topic of selecting appropriate number of features, the stepwise regression [19] and the best-first search by Winston [20] are considered as standard techniques. The former uses a statistical partial F-test in deciding whether to add a new feature or not. The latter searches the space of attribute subsets by greedy hillclimbing augmented with backtracking facility. Since it does not care how the performance of subsets are evaluated, the sucess of the algorithm usually depends on the subset evaluation scheme.

1.3 Previous Works for Feature Extraction

Feature extraction is a process of revealing a number of descriptors from raw data of an object, representing information of an object, suitable for further data mining process. Usually feature extraction is realized via transformations of the raw data into condensed representation in a feature space [6].

Many researches have been made on the feature extraction problems. Though the principal component analysis (PCA) is the most popular [21], by its nature, it is not well-fitted for supervised learning since it does not make use of any output class information in deciding the principal components. The main drawback of this method is that the extracted features are not invariant under transformation. Merely scaling the attributes changes resulting features.

Unlike PCA, Fisher's linear discriminant analysis (LDA) [22] focuses on classification problems to find optimal linear discriminating functions. Though it is a very simple and powerful method for feature extraction, the application of this method is limited to the case in which classes have significant differences between means, since it is based on the information about the differences between means.

In addition, the original LDA cannot produce more than $N_c - 1$ features, where N_c is the number of classes though an extension has been made for this problem in [23].

Another common method of feature extraction is to use a feedforward neural network such as multilayer perceptron (MLP). This method uses the fact that in the feedforward structure the output class is determined through the hidden nodes which produce transformed forms of original input features. This notion can be understood as squeezing the data through a bottleneck of a few hidden units. Thus, the hidden node activations are interpreted as new features in this approach. This line of research includes [24] - [27]. Fractal encoding [28] and wavelet transformation [29] have also been used for feature extraction.

Recently, in neural networks and signal processing circles, independent component analysis (ICA), which was devised for blind source separation problems, has received a great deal of attention because of its potential applications in various areas. Bell and Sejnowski [30] have developed an unsupervised learning algorithm performing ICA based on entropy maximization in a single-layer feedforward neural network. ICA can be very useful as a dimension-preserving transform because it produces statistically independent components, and some have directly used ICA for feature extraction and selection [31] - [34]. Recent researches [17], [35] are focused on extraction of output relevant features based on mutual information maximization methods. In these researches, Renyi's entropy measure was used instead of that of Shannon.

1.4 Organization of the Dissertation

In this dissertation, new methods for the feature selection and extraction for classification problems are presented and the proposed methods are applied to various problems including face recognition problems. Throughout the dissertation, the mutual information is used as a measure in determining the relevance of features.

In the first part of the dissertation, a new feature selection method with the mutual information maximization scheme is proposed for the classification prob-

lem. In calculating the mutual information between the input features and the output class, instead of discretizing the input space, the Parzen window method is used to estimate the input distribution. With this method, more accurate mutual information is calculated. It has been used for measuring the relative importance of input features in determining the class labels, and this feature selection method gives better performances than other conventional methods.

In the second part of the dissertation, the feature extraction problem is dealt with and it is shown how standard algorithms for ICA can be appended with class labels to extract good features for classification. The proposed method produces a number of features that do not carry information about the class label – these features will be discarded – and a number of features that do. The advantage is that general ICA algorithms become available to a task of feature extraction by maximizing the joint mutual information between class labels and new features. It is an extended version of [36] and this method is well-suited for classification problems. The algorithm is originally developed for binary-class classification problems and then it is extended to multi-class classification problems. A stability analysis is also provided for this method.

The proposed feature selection and extraction methods are applied to several classification problems. The proposed algorithms greatly reduces the dimension of feature space while improving classification performance.

The remainder of this dissertation is organized as follows. In the following chapter, the basics of information theory, Parzen window method, and ICA are briefly presented. In Chapter 3, a new feature selection method based on Parzen window method is proposed. In Chapter 4, a new feature extraction algorithm based on ICA is proposed and a local stability analysis of the algorithm is also provided. At the end of Chapter 3 and 4, the proposed algorithms are applied to several classification problems to show their effectiveness. And finally, conclusions follow in Chapter 5.

Chapter 2

Preliminaries

In this section, some basic concepts and notations of the information theory and the Parzen window that are used in the development of the proposed algorithms are briefly introduced. A brief review of ICA is also presented.

To make things clear, from now on, capital letters represent random variables and small letters are instances of the corresponding random variables. Boldfaced letters represent vectors.

2.1 Entropy and Mutual Information

A classifying system maps input features onto output classes. There are relevant features that have important information on outputs, whereas irrelevant ones contain little information on outputs. In solving the feature selection and extraction problems, one tries to find inputs that contain as much information on the outputs as possible and need tools for measuring the information. Fortunately, the information theory provides a way to measure the information of random variables with entropy and mutual information [37], [38].

The entropy is a measure of uncertainty of random variables. If a discrete random variable X has $\mathcal{X}$ alphabets and the *pdf* is $p(x) = \Pr\{X = x\}$, $x \in \mathcal{X}$, the entropy of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.1}$$

Here the base of log is 2 and the unit of entropy is the *bit*. For two discrete random variables X and Y with their joint *pdf* $p(x, y)$, the joint entropy of X and Y is defined as

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\,\log p(x,y). \tag{2.2}$$

The joint entropy measures the total uncertainty of random variables.

When certain variables are known and others are not, the remaining uncertainty is measured by the conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x)\,H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\,\log p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\,\log p(y|x). \end{aligned} \tag{2.3}$$

In the equation above, $H(Y|X)$ represents the remaining information of Y when X is known. As shown in (2.3) the conditional entropy is defined as the conditional expectation of the entropy of an unknown variable given a known random variable. The joint entropy and the conditional entropy has the following relation:

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y). \end{aligned} \tag{2.4}$$

This, known as the *chain-rule*, implies that the total entropy of random variables X and Y is the entropy of X plus the remaining entropy of Y for a given X.

The information found commonly in two random variables is of importance in this thesis, and this is defined as the mutual information between two variables:

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\,\log\frac{p(x,y)}{p(x)p(y)}. \tag{2.5}$$

If the mutual information between two random variables is large (small), it means two variables are closely (not closely) related. If the mutual information becomes

zero, the two random variables are totally unrelated and the two variables are independent. The mutual information and the entropy have the following relation:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ I(X;Y) &= H(Y) - H(Y|X) \\ I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ I(X;Y) &= I(Y;X) \\ I(X;X) &= H(X). \end{aligned} \tag{2.6}$$

Until now, definitions of the entropy and the mutual information of discrete random variables have been presented. For many classifying systems the output class C can be represented with a discrete random variable, while the input features are generally continuous. For continuous random variables, though the differential entropy and mutual information are defined as

$$\begin{aligned} H(X) &= -\int p(x)\, \log p(x) dx \\ I(X;Y) &= \int p(x,y)\, \log \frac{p(x,y)}{p(x)p(y)} dxdy, \end{aligned} \tag{2.7}$$

it is very difficult to find *pdf*s $(p(x), p(y), p(x,y))$ and to perform the integrations. Therefore the continuous input feature space is divided into several discrete partitions and the entropy and the mutual information is calculated using the definitions for discrete cases. The inherent error that exists in the quantization process is of great concern in the computation of entropy and mutual information of continuous variables.

2.2 The Parzen Window Density Estimate

To calculate the mutual information between the input features and the output class, one need to know the *pdf*s of the inputs and the output. The Parzen window density estimate can be used to approximate the probability density $p(\boldsymbol{x})$ of a vector of continuous random variables $\boldsymbol{X}$ [39]. It involves the superposition of a normalized window function centered on a set of random samples. Given a

set of n d-dimensional training vectors $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, the *pdf* estimate of the Parzen window is given by

$$\hat{p}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x} - \boldsymbol{x}_i, h), \tag{2.8}$$

where $\phi(\cdot)$ is the window function and h is the window width parameter. Parzen showed that $\hat{p}(\boldsymbol{x})$ converges to the true density if $\phi(\cdot)$ and h are selected properly [39]. The window function is required to be a finite-valued non-negative density function such that

$$\int \phi(\boldsymbol{y}, h) d\boldsymbol{y} = 1, \tag{2.9}$$

and the width parameter is required to be a function of n such that

$$\lim_{n \to \infty} h(n) = 0, \tag{2.10}$$

and

$$\lim_{n \to \infty} n h^d(n) = \infty. \tag{2.11}$$

The selection of h is always crucial in the density estimator by the Parzen window. Despite significant efforts in the past, it is still unclear how to optimize the value of h. Some authors [40], [41] recommended the method of selecting experimentally the best h for a particular data set. In the Parzen window classifier system [42], h was selected by varying it over several orders of magnitude and choosing the values h_{opt} corresponding to the minimum error. In [43], h was set to $\frac{1}{\log n}$.

For window functions, the rectangular and the Gaussian window functions are commonly used. In this dissertation, the Gaussian window function of the following is used:

$$\phi(\boldsymbol{z}, h) = \frac{1}{(2\pi)^{d/2} h^d |\Sigma|^{1/2}} \exp(-\frac{\boldsymbol{z}^T \Sigma^{-1} \boldsymbol{z}}{2h^2}), \tag{2.12}$$

where Σ is a covariance matrix of a d-dimensional random vector $\boldsymbol{Z}$ whose instance is $\boldsymbol{z}$.

In the density estimation by the Parzen window, the ratio of the sample size to the dimensionality may be too small or too large. If it is too small, the covariance

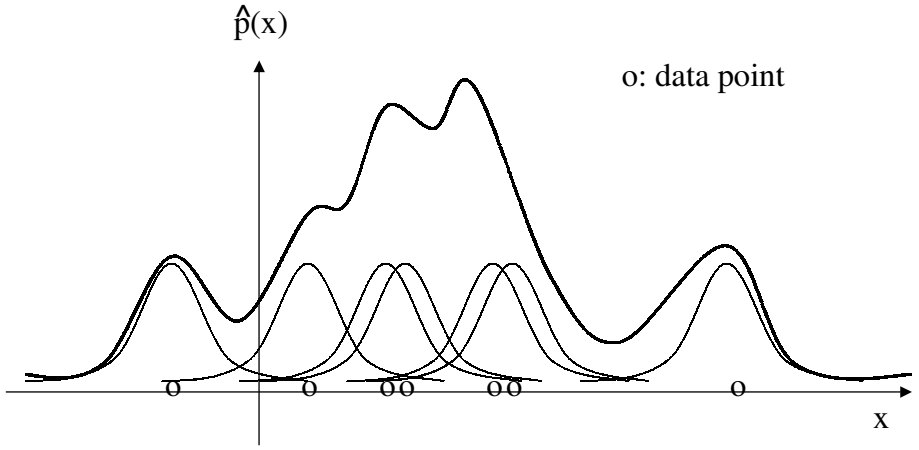


Figure 2.1: An example of Parzen window density estimate

matrix becomes singular and Muto et al. [44] devised a method to avoid this situation. On the other hand, if the ratio is too large, the computational burden becomes heavier and the clustering method [42] or the sample selection method [45] can be used in estimating the density function by the Parzen window. Figure 2.1 is a typical example of the Parzen window density estimate. In the figure, a Gaussian kernel is placed on top of each data point to produce the density estimate $\hat{p}(x)$.

2.3 Review of ICA

The problem of linear independent component analysis for blind source separation was developed in the literature [46] - [48]. In parallel, Bell and Sejnowski [30] have developed an unsupervised learning algorithm based on entropy maximization of a feedforward neural network's output layer, which is referred to as the Infomax algorithm. The Infomax approach, maximum likelihood estimation (MLE) approach, and negentropy maximization approach were shown to lead to identical methods [49] - [51].

The problem setting of ICA is as follows. Assume that there is an L-dimensional

zero-mean non-Gaussian source vector $\boldsymbol{s}(t) = [s_1(t), \cdots, s_L(t)]^T$, such that the components $s_i(t)$'s are mutually independent, and an observed data vector $\boldsymbol{x}(t) = [x_1(t), \cdots, x_N(t)]^T$ is composed of linear combinations of sources $s_i(t)$ at each time point t, such that

$$\boldsymbol{x}(t) = A\boldsymbol{s}(t) \tag{2.13}$$

where A is a full rank $N \times L$ matrix with $L \leq N$. The goal of ICA is to find a linear mapping W such that each component of an estimate $\boldsymbol{u}$ of the source vector

$$\boldsymbol{u}(t) = W\boldsymbol{x}(t) = WA\boldsymbol{s}(t) \tag{2.14}$$

is as independent as possible. The original sources $\boldsymbol{s}(t)$ are exactly recovered when W is the inverse of A up to some scale changes and permutations. For a derivation of an ICA algorithm, one usually assumes that $L = N$, because he has no idea about the number of sources. In addition, sources are assumed to be independent of time t and are drawn from independent identical distribution $p_i(s_i)$.

Bell and Sejnowski [30] have used a feed-forward neural processor to develop the Infomax algorithm, one of the popular algorithms for ICA. The overall structure of the Infomax is shown in Fig. 2.2. This neural processor takes $\boldsymbol{x}$ as an input vector. The weight W is multiplied to the input $\boldsymbol{x}$ to give $\boldsymbol{u}$ and each component u_i goes through a bounded invertible monotonic nonlinear function $g_i(\cdot)$ to match the cumulative distribution of the sources. Let $y_i = g_i(u_i)$ as shown in the figure.

From the view of information theory, maximizing the statistical independence among variables u_i's is equivalent to minimizing mutual information among u_i's. This can be achieved by minimizing mutual information between y_i's, since the nonlinear transfer function $g_i(\cdot)$ does not introduce any dependencies.

In [30], it has been shown that by maximizing the joint entropy $H(\boldsymbol{Y})$ of the output $\boldsymbol{y} = [y_1, \cdots, y_N]^T$ of a processor, the mutual information among the output components Y_i's

$$I(\boldsymbol{Y}) \triangleq I(Y_1; Y_2; \cdots; Y_N) = \int p(\boldsymbol{y}) \log \frac{p(\boldsymbol{y})}{\prod_{i=1}^{N} p_i(y_i)} \, d\boldsymbol{y} \tag{2.15}$$

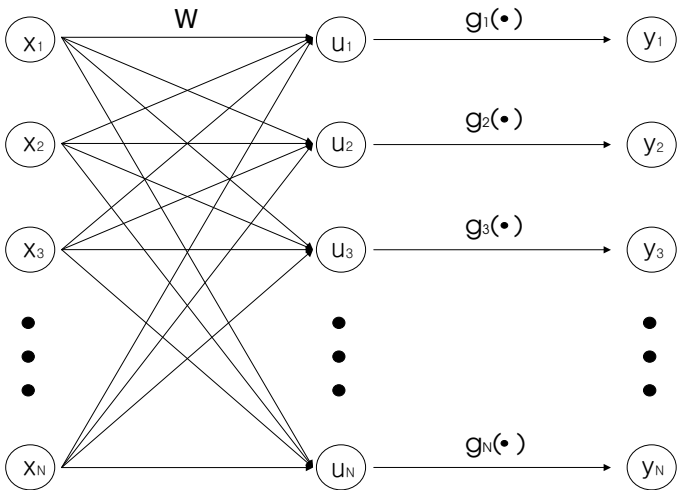


Figure 2.2: Feedforward structure for ICA

can be approximately minimized. Here, $p(\boldsymbol{y})$ is the joint *pdf* of a random vector $\boldsymbol{Y}$, and $p_i(y_i)$ is the marginal *pdf* of the random variable Y_i.

The joint entropy of the outputs of this processor is

$$\begin{aligned} H(\boldsymbol{Y}) &= -\int p(\boldsymbol{y}) \log p(\boldsymbol{y}) d\boldsymbol{y} \\ &= -\int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{|\det J(\boldsymbol{x})|} d\boldsymbol{x} \end{aligned} \tag{2.16}$$

where $J(\boldsymbol{x})$ is the Jacobian matrix whose (i,j)th element is the partial derivative $\partial y_j / \partial x_i$. Note that $J(\boldsymbol{x}) = W$. Differentiating $H(\boldsymbol{Y})$ with respect to W leads to the learning rule for ICA:

$$\Delta W \propto W^{-T} - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{x}^T. \tag{2.17}$$

By multiplying $W^T W$ on the right, the natural gradient [52] is obtained speeding up the convergence rate

$$\Delta W \propto [I - \boldsymbol{\varphi}(\boldsymbol{u})u^T]W \tag{2.18}$$

where

$$\boldsymbol{\varphi}(\boldsymbol{u}) = \left[-\frac{\frac{\partial p_1(u_1)}{\partial u_1}}{p_1(u_1)}, \cdots, -\frac{\frac{\partial p_N(u_N)}{\partial u_N}}{p_N(u_N)} \right]^T. \tag{2.19}$$

The parametric density estimation $p_i(u_i)$ plays an important role in the success of the learning rule in (2.18). If $p_i(u_i)$ is assumed to be Gaussian, $\varphi_i(u_i) = -\dot{p}_i(u_i)/p_i(u_i)$ becomes a linear function of u_i with a positive coefficient and the learning rule (2.18) becomes unstable. This is why non-Gaussian sources are assumed in ICA.

There is a close relation between the assumption on the source distribution and the choice of the nonlinear function $g_i(\cdot)$. By simple computation with (2.15) and (2.16), the joint entropy $H(\boldsymbol{Y})$ becomes

$$H(\boldsymbol{Y}) = \sum_{i=1}^{N} H(Y_i) - I(\boldsymbol{Y}). \tag{2.20}$$

The maximal value for $H(\boldsymbol{Y})$ is achieved when the mutual information among the outputs is zero and their marginal distributions are uniform. For a uniform distribution of Y_i, the distribution of U_i must be

$$p_i(u_i) \propto \left| \frac{\partial g_i(u_i)}{\partial u_i} \right| \tag{2.21}$$

because the relation between the pdf of Y_i and that of U_i is

$$p_i(y_i) = p_i(u_i) / \left| \frac{\partial g_i(u_i)}{\partial u_i} \right|, \quad \text{for } p_i(y_i) \neq 0. \tag{2.22}$$

By the relationship (2.21), the estimate u_i of the source has a distribution that is approximately the form of the derivative of the nonlinearity.

Note that if the sigmoid function is used for $g_i(\cdot)$ as in [30], $p_i(u_i)$ in (2.21) becomes super-Gaussian, which has longer tails than the Gaussian pdf. Some research [52], [53] relaxes the assumption on the source distribution to be sub-Gaussian or super-Gaussian and [52] leads to the extended Infomax learning rule:

$$\begin{aligned} \Delta W &\propto [I - D \tanh(\boldsymbol{u})\boldsymbol{u}^T - \boldsymbol{u}\boldsymbol{u}^T]W \\ &\begin{cases} d_i = 1 & : \text{super-Gaussian} \\ d_i = -1 & : \text{sub-Gaussian.} \end{cases} \end{aligned} \tag{2.23}$$

Here d_i is the ith element of the N-dimensional diagonal matrix D, and it switches between sub- and super-Gaussian using a stability analysis.

In this dissertation, the extended Infomax algorithm in [52] is adopted because it is easy to implement with less strict assumptions on the source distribution.

Chapter 3

Feature Selection Based on Parzen Window

Various feature selection methods can be devised depending on the goal and the criterion of a data mining problem. In this chapter, a new input feature selection algorithm by maximizing the mutual information between input features and the output class is presented for classification problems. In the previous feature selection algorithms such as the mutual information feature selector (MIFS) [7] and the mutual information feature selection under uniform information distribution (MIFS-U) [8], [9], an extension of the MIFS, the mutual information of continuous variables is calculated using discrete quantization method. This quantization step inherently involves some errors in computation of mutual information and feature subset selected with this criterion may contain erroneous features. The proposed method, called Parzen window feature selector (PWFS), computes the mutual information between input features which take on continuous values and categorical output class directly using Parzen window method [54]. Before presenting the algorithm, the feature selection problems are formalized in the following.

3.1 Problem Formulation

The success of a feature selection algorithm for classification problems depends critically on how much information about the output class is contained in the selected features. A useful theorem in relation to this is Fano's inequality [38] in information theory.

> *(Fano's inequality)* Let $\boldsymbol{X}$ and C be random variables that represent input features and output class, respectively. If one tries to estimate the output class C using the input features $\boldsymbol{X}$, the minimal probability of incorrect estimation P_E satisfies the following inequality:
>
> $$P_E \geq \frac{H(C|\boldsymbol{X}) - 1}{\log N_c} = \frac{H(C) - I(\boldsymbol{X}; C) - 1}{\log N_c}. \tag{3.1}$$

Because the entropy of class $H(C)$ and the number of classes N_c is fixed, the lower bound of P_E is minimized when $I(\boldsymbol{X}; C)$ becomes the maximum. Thus it is necessary for good feature selection methods to maximize the mutual information $I(\boldsymbol{X}; C)$.

Battiti [7] formalized this concept of selecting the most relevant k features from a set of n features as a "feature reduction" problem:

> *FRn-k (feature reduction from n to k)* : Given an initial set $\mathcal{F}$ with n features and an output class C, find the subset $\mathcal{S} \subset \mathcal{F}$ with k features that minimizes $H(C|\boldsymbol{S})$, i.e., that maximizes the mutual information $I(\boldsymbol{S}; C)$. Where $\boldsymbol{S}$ is a k-dimensional feature vector whose components are the elements of $\mathcal{S}$.

There are three key strategies for solving this *FRn-k* problem. The first strategy is the *generate and test*. All the feature subsets $\mathcal{S}$ are generated and their $I(\boldsymbol{S}; C)$ are compared. Theoretically, this can find the optimal subset, but it is almost impossible due to the large number of combinations when the number of features are reasonably large. The second strategy is the *backward elimination*. In this strategy, from the full feature set $\mathcal{F}$ that contains n elements, the worst

features are eliminated one by one until k elements remain. This method also has many drawbacks in computing $I(\boldsymbol{S}; C)$ because the dimension of feature space can be too large in calculating the joint *pdf*s. The final strategy is the *greedy selection*. In this method, starting from the empty set of selected features, the best available input feature is added to the selected feature set one by one until the size of the set reaches k. This ideal greedy selection algorithm using the mutual information as the relevance criterion is realized as follows:

1. (Initialization) set $\mathcal{F} \longleftarrow$ "initial set of n features," $\mathcal{S} \longleftarrow$ "empty set."
2. (Computation of the MI with the output class) $\forall F_i \in \mathcal{F}$, compute $I(F_i; C)$.
3. (Selection of the first feature) find the feature that maximizes $I(F_i; C)$, set $\mathcal{F} \longleftarrow \mathcal{F} \backslash \{F_i\}$, $\mathcal{S} \longleftarrow \{F_i\}$.
4. (Greedy selection) repeat until desired number of features are selected.
   (a) (Computation of the joint MI between variables) $\forall F_i \in \mathcal{F}$, compute $I(F_i, \boldsymbol{S}; C)$.
   (b) (Selection of the next feature) choose the feature $F_i \in \mathcal{F}$ that maximizes $I(F_i, \boldsymbol{S}; C)$, and set $\mathcal{F} \longleftarrow \mathcal{F} \backslash \{F_i\}$, $\mathcal{S} \longleftarrow \{F_i\}$.
5. Output the set $\mathcal{S}$ containing the selected features.

To compute the mutual information, the *pdf*s of input and output variables must be known, but this is difficult in practice, so the histogram method has been used in estimating the *pdf*s. But the histogram method needs extremely large memory space in calculating the mutual information. For example, in selecting k features problem, if the output classes are composed of K_c classes and the jth input feature space is divided into P_j partitions to get the histogram, there must be $K_c \times \Pi_{j=1}^{k} P_j$ cells to compute $I(F_i, \boldsymbol{S}; C)$. In this case, even for a simple problem of selecting 10 important features, $K_c \times 10^{10}$ memories are needed if each feature space is divided into 10 partitions. Furthermore, to get a correct mutual information, the number of samples must be at least in the same order as the number of cells. Therefore realization of the ideal greedy selection algorithm

is practically impossible by estimating the *pdf*s with histogram. To overcome this practical obstacle, alternative methods have been devised [7] [8] [9]. In the following section, these methods are briefly reviewed. Thereafter, in the Section 3.3, a new method of feature selection using Parzen window density estimation is proposed.

3.2 Previous Works (MIFS, MIFS-U)

The mutual information feature selector (MIFS) algorithm [7] is the same as the ideal greedy selection algorithm except for Step 4. Instead of calculating $I(F_i, \boldsymbol{S}; C)$, the mutual information between a *candidate for newly selected feature* F_i plus *already selected features* $\boldsymbol{S}$ and output classes C, Battiti [7] used only $I(F_i; C)$ and $I(F_i; F_j)$. To be selected, a feature which cannot be predictable from the already selected features in $\mathcal{S}$, must be informative regarding the class. In the MIFS, Step 4 in ideal greedy selection algorithm was replaced as follows [7]:

> 4. (Greedy selection) repeat until desired number of features are selected.
>
> > (a) (Computation of the MI between variables) for all couples of variables (F_i, F_s) with $F_i \in \mathcal{F}, F_s \in \mathcal{S}$ compute $I(F_i; F_s)$, if it is not yet available.
> >
> > (b) (Selection of the next feature) choose the feature $F_i \in \mathcal{F}$ that maximizes $I(F_i; C) - \beta \sum_{F_s \in \mathcal{S}} I(F_i; F_s)$; set $\mathcal{F} \longleftarrow \mathcal{F} \backslash \{F_i\}$, $\mathcal{S} \longleftarrow \{F_i\}$.

Here β is a redundancy parameter which is used in considering the redundancy among input features. If $\beta = 0$, the mutual informations among input features are not taken into consideration and the algorithm selects features in the order of the mutual information between an input feature and output classes, the redundancy between input features is never reflected. As β grows, the mutual informations between input features begin to influence the selection procedure and the redundancy becomes reduced. But in the case where β is too large, the

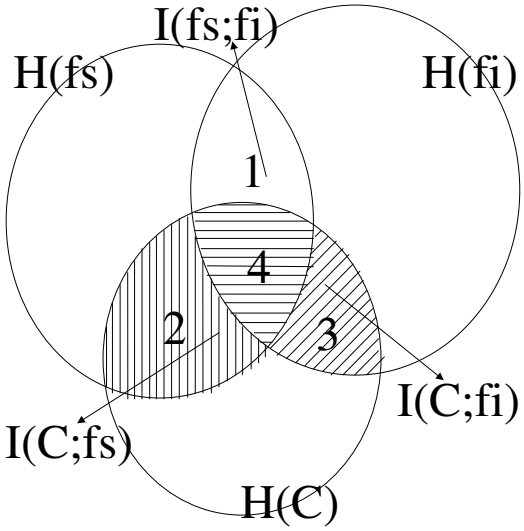


Figure 3.1: The relation between input features and output classes

algorithm only considers the relation between inputs and does not reflect the input-output relation well.

The relation between input features and output classes can be represented as shown in Fig. 3.1. The ideal greedy feature selection algorithm using the mutual information chooses the feature F_i that maximizes joint mutual information $I(F_i, F_s; C)$ which is the area 2,3, and 4, represented by the dashed area in Fig. 3.1. Because $I(F_s; C)$ (area 2 and 4) is common for all the unselected features F_i in computing the joint mutual information $I(C; F_i, F_s)$, the ideal greedy algorithm selects the feature F_i that maximizes the area 3 in Fig. 3.1. On the other hand, the MIFS selects the feature that maximizes $I(C; F_i) - \beta I(F_i; F_s)$. For $\beta = 1$, it corresponds to area 3 subtracted by area 1 in Fig. 3.1.

Therefore if a feature is closely related to the already selected feature F_s, the area 1 in Fig. 3.1 is large and this can degrade the performance of MIFS. For this reason, the MIFS does not work well in nonlinear problems such as the following example.

Example *Two independent random variables X and Y are uniformly distributed on [-0.5,0.5], and assume that there are 3 input features $X, X - Y$ and Y^2. The*

Table 3.1: Feature Selection by MIFS for the *Example*

(a) MI between input and output classes ($I(F_i; C)$)

X	$X-Y$	Y^2
0.8459	0.2621	0.0170

(b) MI between input features ($I(F_i; F_j)$)

	X	$X-Y$	Y^2
X	–	0.6168	0.0610
$X-Y$	0.6168	–	0.5624
Y^2	0.0610	0.5624	–

(c) $I(f_i; C) - I(F_i; F_s)$

$X-Y$	$I(X-Y;Z) - I(X-Y;X) = -0.3537$
Y^2	$I(Y^2;Z) - I(Y^2;X) = -0.0439$

(d) Order of Selection

	X	$X-Y$	Y^2
Ideal Greedy	1	2	3
MIFS ($\beta = 1$)	1	3	2

output belongs to class Z

$$Z = \begin{cases} 0 & \textit{if } X + 0.2Y < 0 \\ 1 & \textit{if } X + 0.2Y \geq 0. \end{cases}$$

When 1,000 samples are taken and each input feature space is partitioned into ten, the mutual information between each input feature and the output classes and those between input features are shown in Table 3.1. The order of selection by the MIFS($\beta = 1$) is X, Y^2, and $X - Y$ in that order.

As shown in Table 3.1(c) the MIFS selects Y^2 rather than the more important feature $X - Y$ as the second choice. Note that Y can be calculated exactly by a linear combination of X and $X-Y$. Because the output class Z can be computed exactly by X and $X-Y$, one can say $X-Y$ rather than Y^2 is more informative about the Z for a given X. To verify that $X - Y$ is a more important feature than Y^2, neural networks were trained with (X,$X - Y$) and (X,Y^2) as input features respectively. The neural networks were trained with sets of 200 training data and the classification rates are on the test data of 800 patterns. Two hidden

nodes were used with a learning rate of 2.0 and momentum of 0.1. The number of epochs at the time of termination was 200. As expected, the results are 99.8% when X and $X - Y$ are selected, and 93.4% when X and Y^2 are selected.

This is due to the relatively large β, and is a good example showing a case where the relations between inputs are weighted too much.This is due to the difference of the algorithm from the ideal greedy selection algorithm described ahead. The MIFS handles redundancy at the expense of classifying performance.

The mutual information feature selection under uniform information distribution (MIFS-U) [8] [9] that is closer to the ideal one than the MIFS is now reviewed. The ideal greedy algorithm tries to maximize $I(C; F_i, F_s)$ (area 2, 3, and 4 in Fig. 3.1) and this can be rewritten as

$$I(C; F_i, F_s) = I(C; F_s) + I(C; F_i|F_s). \tag{3.2}$$

Here $I(C; F_i|F_s)$ represents the remaining mutual information between the output class C and the feature F_i for a given F_s. This is shown as area 3 in Fig. 3.1, whereas the area 2 plus area 4 represents $I(C; F_s)$. Since $I(C; F_s)$ is common for all the candidate features to be selected in the ideal feature selection algorithm, there is no need to compute this. So the ideal greedy algorithm now tries to find the feature that maximizes $I(C; F_i|F_s)$ (area 3 in Fig. 3.1). However, calculating $I(C; F_i|F_s)$ requires as much work as calculating $H(F_i, F_s, C)$.

So $I(C; F_i|F_s)$ will be approximated with $I(F_s; F_i)$ and $I(C; F_i)$, which are relatively easy to calculate. The conditional mutual information $I(C; F_i|F_s)$ can be represented as

$$I(C; F_i|F_s) = I(C; F_i) - \{I(F_s; F_i) - I(F_s; F_i|C)\}. \tag{3.3}$$

Here $I(F_s; F_i)$ corresponds to area 1 and 4 and $I(F_s; F_i|C)$ corresponds to area 1. So the term $I(F_s; F_i) - I(F_s; F_i|C)$ corresponds to area 4 in Fig. 3.1. The term $I(F_s; F_i|C)$ means the mutual information between the already selected feature F_s and the candidate feature F_i for a given class C. If conditioning by the class C does not change the ratio of the entropy of F_s and the mutual information between F_s and F_i, i.e., if the following relation holds,

$$\frac{H(F_s|C)}{H(F_s)} = \frac{I(F_s; F_i|C)}{I(F_s; F_i)}, \tag{3.4}$$

Table 3.2: Validation of (3.4) for the *Example*

$H(F_s\|C)/H(F_s)$	
$H(X)$	3.3181
$H(X\|Z)$	2.4723
$H(X\|Z)/H(X)$	0.745

$I(F_s;F_i\|C)/I(F_s;F_i)$			
$I(X-Y;X)$	0.6168	$I(Y^2;X)$	0.0610
$I(X-Y;X\|Z)$	0.4379	$I(Y^2;X\|Z)$	0.0491
$I(X-Y;X\|Z)/I(X-Y;X)$	0.709	$I(Y^2;X)/I(Y^2;X\|Z)$	0.805

$I(F_s;F_i|C)$ can be represented as

$$I(F_s;F_i|C) = \frac{H(F_s|C)}{H(F_s)} I(F_s;F_i). \tag{3.5}$$

Using the equation above and (3.3)

$$\begin{aligned} I(C;F_i|F_s) &= I(C;F_i) - (1 - \frac{H(F_s|C)}{H(F_s)}) I(F_s;F_i) \\ &= I(C;F_i) - \frac{I(C;F_s)}{H(F_s)} I(F_s;F_i). \end{aligned} \tag{3.6}$$

If it is assumed that each region in Fig. 3.1 corresponds to its corresponding information, condition (3.4) is hard to satisfied when information is concentrated on one of the four regions in Fig. 3.1, i.e., $H(F_s|F_i, C)$, $I(F_s;F_i|C)$, $I(C;F_s|F_i)$, or $I(C;F_s;F_i)$. It is more likely that the condition (3.4) holds when information is distributed uniformly throughout the region of $H(F_s)$ in Fig. 3.1. Because of this, the algorithm is referred to as the MIFS-U (mutual information feature selector under uniform information distribution). The ratio in (3.4) is computed for the *Example* and the values of several pieces of mutual information are shown in Table 3.2. It shows that the relation (3.4) holds with less than 10% of error.

With this formula, the Step 4 in the ideal greedy selection algorithm is revised as follows:

> 4. (Greedy selection) repeat until desired number of features are selected.

(a) (Computation of entropy) $\forall F_s \in \mathcal{S}$, compute $H(F_s)$ if it is not already available.

(b) (Computation of the MI between variables) for all couples of variables (F_i, F_s) with $F_i \in \mathcal{F}, F_s \in \mathcal{S}$, compute $I(F_s; F_i)$, if it is not yet available.

(c) (Selection of the next feature) choose a feature $F_i \in \mathcal{F}$ that maximizes $I(C; F_i) - \beta \sum_{F_s \in S} \frac{I(C;F_s)}{H(F_s)} I(F_i; F_s)$; set $\mathcal{F} \longleftarrow \mathcal{F} \backslash \{F_i\}$, $S \longleftarrow \{F_i\}$.

Here the entropy $H(F_s)$ can be computed in the process of computing the mutual information with output class C, so there is little change in computational load with respect to the MIFS. In the calculation of mutual informations and entropies, there are two mainly used approaches of partitioning the continuous feature space: equi-distance partitioning [7] and equi-probable partitioning [55]. The equi-distance partitioning method is used for the MIFS-U as in [7]. The detail of partitioning method is as follows: If the distribution of the values in a variable F_i is not known *a priori*, its mean μ and the standard deviation σ are computed and the interval $[\mu - 2\sigma, \mu + 2\sigma]$ is divided into p_i equally spaced segments. The points falling outside are assigned to the extreme left (right) segment.

Parameter β offers flexibility to the algorithm as in the MIFS. If β is set to zero, the proposed algorithm chooses features in the order of the mutual information with the output. As β grows, it excludes the redundant features more efficiently. In general β can be set to 1 in compliance with (3.6). For all the experiments to be discussed later, β is set to 1 if there is no comment.

In computing mutual information $I(F_s; F_i)$, a second order joint probability distribution which can be computed from a joint histogram of variables F_s and F_i is required. Therefore, if there are n features and each feature space is divided into p partitions to get a histogram, p^2 memories are needed for each of $\binom{n}{2}$ histograms to use MIFS-U. The computational effort therefore increases in the order of n^2 as the number of features increases for given numbers of examples and partitions. This implies that the computational complexity of MIFS-U is not greater than that of MIFS.

Although the MIFS and MIFS-U methods report good results on some problems, these are somewhat heuristic because they do not use the mutual information $I(F_i, \boldsymbol{S}; C)$ directly. To overcome these problems, a new method for computing the mutual information between continuous input features and discrete output class is proposed in the following section.

3.3 Parzen Window Feature Selector (PWFS)

In classification problems, the class has discrete values while the input features are usually continuous variables. In this case, rewriting the relation of (2.6), the mutual information between the input features $\boldsymbol{X}$ and the class C can be represented as follows:

$$I(\boldsymbol{X}; C) = H(C) - H(C|\boldsymbol{X}).$$

In this equation, because the class is a discrete variable, the entropy of the class variable $H(C)$ can be easily calculated as in (2.16). But the conditional entropy

$$H(C|\boldsymbol{X}) = -\int_{\boldsymbol{X}} p(\boldsymbol{x}) \sum_{c=1}^{N_c} p(c|\boldsymbol{x}) \log p(c|\boldsymbol{x}) d\boldsymbol{x}, \tag{3.7}$$

where N_c is the number of classes, is hard to get because it is not easy to estimate $p(c|\boldsymbol{x})$.

Now, a new method is presented to estimate the conditional entropy and the mutual information by the Parzen window method. By the Bayesian rule, the conditional probability $p(c|\boldsymbol{x})$ can be written as

$$p(c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|c)p(c)}{p(\boldsymbol{x})}. \tag{3.8}$$

If the class has N_c values, say $1, 2, \cdots, N_c$, the estimate of the conditional *pdf* $\hat{p}(\boldsymbol{x}|c)$ of each class is obtained using the Parzen window method as

$$\hat{p}(\boldsymbol{x}|c) = \frac{1}{n_c} \sum_{i \in I_c} \phi(\boldsymbol{x} - \boldsymbol{x}_i, h), \tag{3.9}$$

where $c = 1, \cdots, N_c$; n_c is the number of the training examples belonging to class c; and I_c is the set of indices of the training examples belonging to class c.

Because the summation of the conditional probability equals one, i.e.,

$$\sum_{k=1}^{N_c} p(k|\boldsymbol{x}) = 1,$$

the conditional probability $p(c|\boldsymbol{x})$ is

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

The second equality is by the Bayesian rule (3.8). Using (3.9), the estimate of the conditional probability becomes

$$\hat{p}(c|\boldsymbol{x}) = \frac{\sum_{i\in I_c} \phi(\boldsymbol{x}-\boldsymbol{x}_i, h_c)}{\sum_{k=1}^{N_c}\sum_{i\in I_k} \phi(\boldsymbol{x}-\boldsymbol{x}_i, h_k)}, \tag{3.10}$$

where h_c and h_k are the class specific window width parameters. Here $\hat{p}(k) = n_k/n$ is used instead of the true density $p(k)$.

If the Gaussian window function (2.12) is used with the same window width parameter and the same covariance matrix for each class, (3.10) becomes

$$\hat{p}(c|\boldsymbol{x}) = \frac{\sum_{i\in I_c} \exp(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x}_i)}{2h^2})}{\sum_{k=1}^{N_c}\sum_{i\in I_k} \exp(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x}_i)}{2h^2})}. \tag{3.11}$$

Note that for multi-class classification problems, there may not be enough samples such that the error for the estimate of class specific covariance matrix can be large. Thus, the same covariance matrix is used for each class throughout this thesis.

Now in the calculation of the conditional entropy (3.7) with n training samples, if the integration is replaced with a summation of the sample points and it is assumed that each sample has the same probability, $\hat{H}(C|\boldsymbol{X})$ can be obtained as follows:

$$\hat{H}(C|\boldsymbol{X}) = -\sum_{j=1}^{n}\frac{1}{n}\sum_{c=1}^{N_c}\hat{p}(c|\boldsymbol{x}_j)\,\log\hat{p}(c|\boldsymbol{x}_j). \tag{3.12}$$

Here $\boldsymbol{x}_j$ is the jth sample of the training data. With (3.11) and (3.12), the estimate of the mutual information is obtained.

The computational complexity for (3.12) is propotional to $n^2 \times d$. When there is a computational problem because of large n, one may use the clustering method [42] or the sample selection method [45] to speed up the calculation. The methods

based on histograms require computational complexity and memory proportional to q^d, where q represents number of quantization levels. Note that the proposed method does not require excessive memory, unlike the histogram based methods.

With the estimation of mutual information described in the previous section, the *FRn-k* problem can be solved by the greedy selection algorithm represented in Section 3.1. Note that the dimension of a input feature vector $\boldsymbol{x}$ starts from one at the beginning and increases one by one as a new feature is added to selected feature set $\mathcal{S}$. For convenience, the proposed method is referred to as the PWFS (Parzen window feature selector) from now on.

In the proposed mutual information estimation, the selection of the window function and the window width parameter is very important. As mentioned in Section II, the rectangular window and the Gaussian window is normally used for the Parzen window function. In the simulation, the Gaussian window is used rather than the rectangular window because it does not contain any discontinuity. For the window width parameter h, $k/\log n$ is used as in [43], where k is a positive constant and n is the number of the samples. This choice of h satisfies the conditions (2.10) and (2.11).

To see the properties of the proposed algorithm, let us consider the typical four points XOR problem. Let $\boldsymbol{X} = (X_1, X_2)$ be a continuous input feature vector and the samples for $\boldsymbol{X}$ are given (0,0), (0,1), (1,0), (1,1). The term C is the discrete output class which takes a value in $\{0, 1\}$. In the Parzen window method, each sample point influences the conditional probability throughout the entire feature space. The influence $\phi(\boldsymbol{x} - \boldsymbol{x}_i, h)$ of a sample point $\boldsymbol{x}_i$ has the polarity of its corresponding class. It is named as a *class specific influence field*, which is similar to an electric field produced by a charged particle. The influence fields generated by given four sample points in the XOR problem are shown in Fig. 3.2. In the figure, the slope and the range of the influence field is determined by the window width parameter h. The smaller h is, the sharper the slope and the narrower the range of influence becomes. Figure 3.2 was drawn with $h = \frac{1}{2\log n}$ where n is the number of sample points which is four in this case. With this h, the higher (lower) estimate for the conditional probability of class C being 0 or 1 for each sample point is 0.90 (0.10) by (3.11). With (3.12), the

conditional entropy estimate $\hat{H}(C|X_1, X_2)$ becomes 0.465, and the entropy $H(C)$ is 1 by (2.16). Thus, the estimate of the mutual information between two input features and the output class $\hat{I}(C; X_1, X_2)$ $(= H(C) - \hat{H}(C|X_1, X_2))$ is 0.535. The significance of $\hat{I}(C; X_1, X_2)$ being greater than zero will become clear later.

In Fig. 3.3, the conditional probability of class 1 calculated by (3.11) is provided on the input feature space. Note that one can get a Baye's classifier if one classify a given input to class 1 when $p(c = 1|\boldsymbol{x}) > 0.5$ and to class 0 when $p(c = 1|\boldsymbol{x}) < 0.5$. This classfier system is a type of Parzen classifier [42], [44], [45], [56]. Since the classifier system is not my concern, this issue is not further dealt with.

In the process of the greedy selection scheme, the mutual informations $I(X_1; C)$, $I(X_2; C)$ between the variables X_1, X_2 and the class C is zero, while the estimate of the mutual information $\hat{I}(C; X_1, X_2)$ between the output class and both input features is far greater than zero. Thus, it is known that using both features gives more information about the output class than using only one of the variables in the greedy selection scheme with the Parzen window. But, in the conventional feature selection methods such as MIFS [7] and MIFS-U [8] [9], this knowledge can not be obtained because these methods do not use the mutual information of multiple variables. Instead, to avoid using too many memory cells in calculating mutual information with the discrete quantization method, they make use of some measure on redundancy between variables information which can be obtained by calculating the mutual information between two input features. These methods report good performances in several problems, but they are prone to errors in highly nonlinear problems like XOR problem and have to resort to some other methods like Taguchi method [9].

One more advantage of the PWFS is that it provides a measure that indicates whether to use additional features or not. Though it is quite difficult to estimate how much the performance will increase with one more feature by the increase of the mutual information, one can at least get a lower bound of error probability by the Fano's inequality and can compare the increseas in mutual information or the error probability which will aid the decision whether to add more features or not.

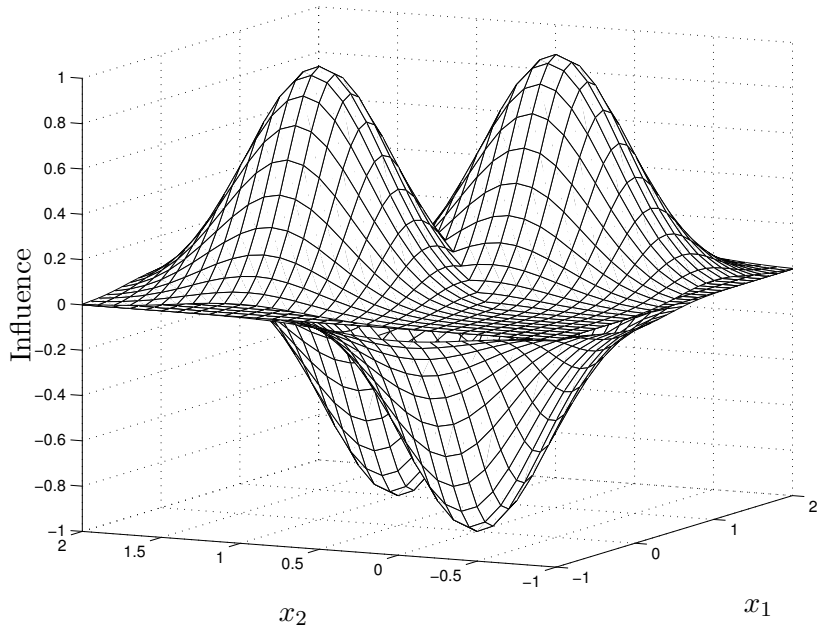


Figure 3.2: Influence fields generated by four sample points in the XOR problem

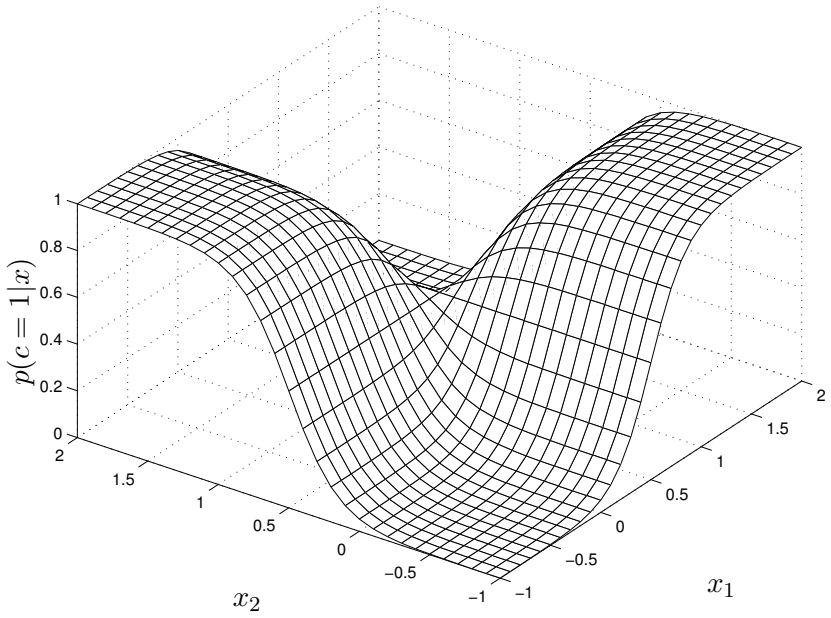


Figure 3.3: Conditional probability of class 1 $p(c = 1|\boldsymbol{x})$ in XOR problem

3.4 Experimental Results

In this section, the PWFS is applied to some of the classification problems and the performance of PWFS is compared with those of MIFS and MIFS-U to show the effectiveness of the PWFS.

In all the following experiments, h is set to $\frac{1}{\log n}$ where n is the sample size of a particular data set as in [43]. Because the off diagonal terms in the covariance matrix can be prone to large errors and need great computational efforts, only diagonal terms are used in the covariance matrix for simplicity if not otherwise stated.

In addition, to expedite the computation, the influence range of a sample point is restricted to $2\sigma \cdot h$ for each dimension, i.e., the influence is made to zero in the outer domain of $2\sigma \cdot h$ from the sample point, where σ is a standard deviation of the corresponding feature. This can greatly reduce the computational effort, especially when there are already enough selected features.

IBM dataset

These datasets were generated by Agrawal et al. [15] to test their data mining algorithm CDP. They were also used in [8], [9], and [13] for testing the performances of each feature selection method. Each of the datasets has nine attributes, which are *salary, commission, age, education level, make of the car, zipcode of the town, value of the house, years house owned,* and *total amount of the loan.* All of them have two classes Group A and Group B. The four classification functions are shown in Table 3.3. For convenience, the four datasets generated using each function in Table 3.3 are referred to as IBM1, IBM2, IBM3, IBM4 and nine input features as $F1, F2, \cdots, F9$, respectively. From the table, it can be seen that only a small fraction of the original features completely determine the output class for these datasets. Thus feature selection can be very useful for these datasets if appropriate features are selected. Although these datasets are artificial, the same argument is true for many real world datasets; there are many irrelevant features and only a small number of features can be used to solve the given problem.

For each dataset, 1,000 input-output patterns are generated and the window

Table 3.3: IBM Classification Functions

Function 1
Group A: $((\text{age} < 40) \ \wedge \ (50\text{K} \leq \text{salary} \leq 100\text{K})) \vee$
$((40 \leq \text{age} < 60) \ \wedge (75\text{K} \leq \text{salary} \leq 125\text{K})) \vee$
$((\text{age} \geq 60) \ \wedge (25\text{K} \leq \text{salary} \leq 75\text{K}))$.
Group B: Otherwise.
Function 2
Group A: $((\text{age} < 40) \wedge$
$(((\text{elevel} \in [0 \ldots 2] \ ? \ (25\text{K} \leq \text{salary} \leq 75\text{K})) : (50\text{K} \leq \text{salary} \leq 100\text{K})))\vee$
$((40 \leq \text{age} < 60) \wedge$
$(((\text{elevel} \in [1 \ldots 3] \ ? \ (50\text{K} \leq \text{salary} \leq 100\text{K})) : (75\text{K} \leq \text{salary} \leq 125\text{K})))\vee$
$((\text{age} \geq 60) \wedge$
$(((\text{elevel} \in [2 \ldots 4] \ ? \ (50\text{K} \leq \text{salary} \leq 100\text{K})) : (25\text{K} \leq \text{salary} \leq 75\text{K})))$.
Group B: Otherwise.
Function 3
Group A: disposable > 0, where
disposable $= (0.67 \times (\text{salary} + \text{commission}) - 5000 \times \text{elevel} - 0.2 \times \text{loan} - 10000)$.
Group B: Otherwise.
Function 4
Group A: $((\text{age} < 40) \wedge$
$(((50\text{K} \leq \text{salary} \leq 100\text{K}) \ ? \ (100\text{K} \leq \text{loan} \leq 300\text{K})) : (200\text{K} \leq \text{loan} \leq 400\text{K})))\vee$
$((40 \leq \text{age} < 60) \wedge$
$(((75\text{K} \leq \text{salary} \leq 125\text{K}) \ ? \ (200\text{K} \leq \text{loan} \leq 400\text{K})) : (300\text{K} \leq \text{loan} \leq 500\text{K})))\vee$
$((\text{age} \geq 60) \wedge$
$(((25\text{K} \leq \text{salary} \leq 75\text{K}) \ ? \ (300\text{K} \leq \text{loan} \leq 500\text{K})) : (100\text{K} \leq \text{loan} \leq 300\text{K})))\vee$
Group B: Otherwise.

Table 3.4: Feature Selection for IBM datasets. The boldfaced features are the relevant ones in the classification.

IBM 1									
	F1	F2	**F3**	F4	F5	F6	F7	F8	F9
MIFS / MIFS-U ($\beta = 0$)	**1**	3	**2**	8	7	9	6	4	5
MIFS ($\beta = 1$)	**1**	9	**2**	3	5	4	8	6	7
MIFS-U ($\beta = 1$)	**1**	9	**2**	3	6	8	7	4	5
PWFS	**1**	3	**2**	4	6	7	8	5	9
IBM 2									
	F1	F2	**F3**	**F4**	F5	F6	F7	F8	F9
MIFS / MIFS-U ($\beta = 0$)	**2**	3	**1**	**8**	5	6	7	9	4
MIFS ($\beta = 1$)	**2**	9	**1**	**3**	5	4	8	6	7
MIFS-U ($\beta = 1$)	**2**	9	**1**	**3**	5	6	7	8	4
PWFS	**1**	9	**3**	**2**	7	6	8	5	4
IBM 3									
	F1	**F2**	F3	**F4**	F5	F6	F7	F8	**F9**
MIFS / MIFS-U ($\beta = 0$)	**2**	**3**	6	**4**	8	9	7	5	**1**
MIFS ($\beta = 1$)	**2**	**9**	7	**3**	5	4	8	6	**1**
MIFS-U ($\beta = 1$)	**2**	**3**	5	**4**	8	7	9	6	**1**
PWFS	**2**	**4**	8	**3**	6	5	9	7	**1**
IBM 4									
	F1	F2	**F3**	F4	F5	F6	F7	F8	**F9**
MIFS / MIFS-U ($\beta = 0$)	**4**	8	**2**	3	5	9	6	7	**1**
MIFS ($\beta = 1$)	**4**	8	**2**	3	5	9	7	6	**1**
MIFS-U ($\beta = 1$)	**4**	9	**2**	3	5	7	6	8	**1**
PWFS	**2**	9	**3**	4	5	6	8	7	**1**

width parameter h is set to $\frac{1}{\log n}$. The proposed algorithm is compared with MIFS [7] and MIFS-U [9]. In MIFS and MIFS-U, each input space was divided into 10 partitions to compute the entropies and the mutual information and redundency parameter β was set to 0 and 1 as in [9].

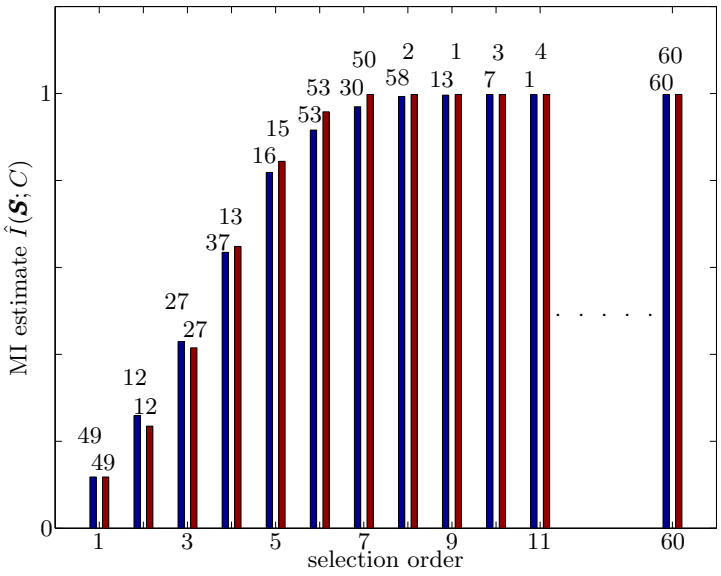


Figure 3.4: Selection order and mutual information estimate of PWFS for sonar dataset (Left bar: Type I, Right bar: Type II. The number on top of each bar is the selected feature index.)

Table 3.4 is the order of selection by each feature selection method. The features used in the classification functions are written in boldface in Table 3.4. In the table, it can be seen that the PWFS performs well for all the four datasets, while the MIFS and MIFS-U fails to identify F1 (*salary*) as one of the important three features in IBM4 dataset.

Sonar dataset

This dataset [57] was constructed to discriminate between the sonar returns bounced off a metal cylinder and those bounced off a rock, and it was used in [7] and [9] to test the performances of their feature selection methods. It consists of 208 patterns including 104 training and testing patterns each. It has 60 input features and two output classes: *metal* and *rock*. As in [7], the input features are normalized to have the values in [0,1] and one node is allotted per each output class for the classification.

For comparison, two types of PWFS are used for this dataset; first one only

Table 3.5: Classification Rates with Different Numbers of Features for Sonar Dataset (%) (The numbers in the parentheses are the standard deviations of 10 experiments)

Number of features	PWFS (Type I)	PWFS (Type II)	MIFS	MIFS-U	Stepwise regression
3	70.23 (1.2)	70.23 (1.2)	51.71 (2.1)	65.23 (1.6)	68.19 (1.1)
6	79.80 (0.8)	77.82 (0.6)	74.81 (1.4)	77.03 (0.4)	76.12 (0.3)
9	80.01 (0.9)	80.44 (1.1)	76.45 (2.4)	78.98 (0.7)	–
10	81.42 (1.4)	–	77.12 (3.1)	78.94 (0.8)	–
12	–	–	78.12 (1.8)	81.51 (0.4)	–
All (60)	87.92 (0.2)				

uses diagonal terms in the covariance matrix (Type I), and the other uses full covariance matrix (Type II). The selection order and the mutual information estimate $\hat{I}(\boldsymbol{S}; C)$ for PWFS are presented in Fig. 4.7. In the figure, the left bars show the results of Type I and the right bars show those of Type II. Here, C and $\boldsymbol{S}$ are as defined in Section III-A. In the figure, the number on top of each bar represents the index of selected feature. It can be seen that the estimate of the mutual information is saturated after 10 (9) features were selected with Type I (Type II); thus, 10 (9) features were used and any more features were not used in PWFS. Note that the selected features of Type I and Type II give nearly the same $\hat{I}(\boldsymbol{S}; C)$ and are the same when the number of selected features is small.

In Table 3.5, the performances of PWFS are compared with those of the conventional MIFS and MIFS-U. In addition, the result of stepwise regression [19] is also reported. Because the importance of each feature is not known *a priori*, $3 \sim 12$ features (top $5\% \sim 20\%$) are selected among the 60 features, and the neural networks are trained with the set of training patterns using these input features. Multilayer perceptrons (MLP) with one hidden layer are used and the hidden layer had three nodes as in [7]. The conventional back-propagation (BP) learning algorithm is used with the momentum of 0.0 and learning rate of 0.2. The networks are trained for 300 epochs in all cases as Battiti did [7]. Each input feature space is divided into ten partitions to calculate the entropies and mutual information by MIFS and MIFS-U. The results of MIFS, MIFS-U and stepwise

regression are from [9]. In the table, all the resulting classification rates are the average values of 10 experiments and the corresponding standard deviations are shown in the parentheses.

From the table, it can be seen that PWFS produced better performances than the others and the performances of Type I and Type II do not differ much.

Vehicle dataset

This dataset comes from the Turing Institute, Glasgow, Scotland [58]. The purpose of the dataset is to classify a given silhouette as one of the four types of vehicle, "Opel," "Saab," "bus," and "van," using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. There are 18 numeric features that were extracted from the silhouettes. Total number of examples are 946, which includes 240 Opel, 240 Saab, 240 bus, and 226 van. Among these, 200 data are used as a training set and the other 746 as a test set.

The PWFS was compared with MIFS and MIFS-U. The stepwise regression cannot be used, because this is a classification problem with more than two classes. The classification was performed using MLP with the standard BP algorithm. Three hidden nodes were used with learning rate of 0.2 and zero momentum. The MLP was trained for 300 iterations, 10 times for each experiment. Table 3.6 is the classification rates of various numbers of selected features. The numbers in the parentheses are the standard deviations of 10 experiments. The result show that PWFS is better than the other algorithms for vehicle dataset.

Other UCI datasets

The PWFS was used for various datasets in the UC-Irvine repository [59] and the performances were compared with those of MIFS and MIFS-U. Table 3.7 is the brief information of the datasets used in this thesis.

For these datasets, several features have been selected, and the results are shown in Tables 3.8 $\sim$ 3.11. As classifier systems, the decision tree classifier C4.5 [14] was used for "letter" and "breast cancer" datasets and the nearest neighborhood classifier with neighborhood size of three was used for "waveform"

Table 3.6: Classification Rates with Different Numbers of Features for Vehicle Dataset (%) (The numbers in the parentheses are the standard deviations of 10 experiments)

Number of features	PWFS	MIFS	MIFS-U
2	58.77 (0.5)	40.23 (0.6)	57.53 (2.5)
4	62.50 (0.5)	57.32 (0.7)	59.97 (2.2)
6	68.89 (1.3)	65.50 (1.7)	63.94 (1.1)
8	71.59 (1.5)	70.04 (1.2)	70.35 (2.5)
10	73.20 (0.8)	71.57 (1.5)	72.70 (1.9)
All (18)		76.45 (1.0)	

Table 3.7: Brief Information of the Datasets Used

Name	# features	# instances	# classes
Letter	16	20,000	26
Breast Cancer	9	699	2
Waveform	21	1,000	3
Glass	9	214	6

Table 3.8: Classification Rates for Letter Dataset

Number of features	PWFS	MIFS	MIFS-U
2	36.36	35.44	35.44
4	67.58	62.46	68.56
6	82.86	81.00	80.50
8	84.72	84.94	83.18
All (16)		87.68	

Table 3.9: Classification Rates for Breast Cancer Dataset

Number of features	PWFS	MIFS	MIFS-U
1	92.28	92.28	92.28
2	95.71	93.42	95.71
3	96.00	93.42	95.00
4	96.57	93.71	94.28
All (9)		96.28	

Table 3.10: Classification Rates for Waveform Dataset

Number of features	PWFS	MIFS	MIFS-U
2	67.71	65.85	58.85
4	75.42	67.57	73.85
6	75.42	67.14	71.57
8	78.85	66.28	77.24
10	79.10	67.71	79.57
All (21)		76.57	

Table 3.11: Classification Rates for Glass Dataset

Number of features	PWFS	MIFS	MIFS-U
1	48.13	48.13	48.13
2	62.61	57.94	57.94
3	68.22	64.95	65.42
4	71.49	66.35	66.35
All (9)		70.56	

and "glass" datasets. In the experiments, 75% was used as the training set and the other 25% was used as the test set for "letter" data, 50% as the training set and the other 50% as the test set for "breast cancer", 30% as the training set and 70% as the test set for "waveform". Since the number of instances is relatively small in "glass" dataset, the 10-fold cross-validation was used for this dataset. In most experiments, it can be seen that the PWFS exhibits better performances than MIFS and MIFS-U. Note also that except for the "letter" dataset, the performances with a smaller number of features are better than those with all features. This result clearly shows the necessity and advantages of feature selection in data mining process.

Chapter 4

Feature Extraction Based on ICA (ICA-FX)

As stated in Introduction, feature extraction is a processing of revealing a number of descriptors from raw data of an object (a sample), representing information about an object, suitable for further data mining processing. For real valued attributes, subspace methods such as PCA and LDA have been used intensively. Usually, different feature extraction methods are preferred depending on the goal and the criterion of a data mining problem. In this chapter, feature extraction for classification problems, where the goal is to extract features that result in a good classification performance with a reduced dimension of a feature space, is dealt with. As in the feature selection method in the previous chapter, mutual information is adopted as the criterion of extracting new features where the features whose mutual information with the output class are the largest are searched for.

Recently, ICA, one of the subspace methods, has been focused in many areas. ICA outputs a set of maximally independent vectors which are linear combinations of observed data. Although these vectors may find some applications in such areas as blind source separation [30] and data visualization [31], it does not fit for feature extraction for classification problems, because it is an unsupervised learning that does not use class information. In this chapter, a feature extraction algorithm is proposed for the classification problem by incorporating

standard ICA algorithms with binary class labels and it is extended to multi-class problems.

The main idea of the proposed feature extraction algorithm is simple. In applying standard ICA algorithms to feature extraction for classification problems, it makes use of the binary class labels to produce two sets of new features; one that does not carry information about the class label (these features will be discarded) and the other that does (these will be useful for classification). The advantage is that general ICA algorithms become available to a task of feature extraction by maximizing the joint mutual information between class labels and new features. Before the algorithm ICA-FX [60] [61] is presented, the purpose of feature extraction is formalized.

4.1 Problem Formulation

The formulation of feature extraction for classification problems is almost the same as that of feature selection. Suppose that there are N normalized input features $\boldsymbol{X} = [X_1, \cdots, X_N]^T$ and a output class label C. The purpose of feature extraction is to extract $M(\leq N)$ new features $\boldsymbol{F}_a = [F_1, \cdots, F_M]^T$ from $\boldsymbol{X}$ containing maximal information of the class.

Using the same argument as in the feature selection formulation, it is necessary for good feature extraction methods to extract features maximizing mutual information with the output class. But there is no transformation $T(\cdot)$ that can increase the mutual information between input features and output class as shown by the following data processing inequality [38].

> *(Data processing inequality)* Let $\boldsymbol{X}$ and C be random variables that represent input features and output class, respectively. For any deterministic function $T(\cdot)$ of $\boldsymbol{X}$, the mutual information between $T(\boldsymbol{X})$ and output class C is upper-bounded by the mutual information between $\boldsymbol{X}$ and C:
>
> $$I(T(\boldsymbol{X}); C) \leq I(\boldsymbol{X}; C) \tag{4.1}$$
>
> where the equality holds if the transformation is invertible.

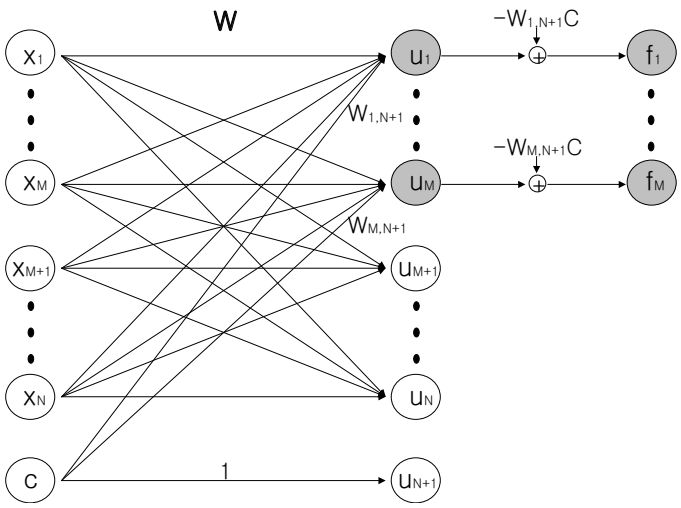


Figure 4.1: Feature extraction algorithm based on ICA (ICA-FX)

Thus, the purpose of a feature extraction is to extract $M(\leq N)$ features $\boldsymbol{F}_a$ from $\boldsymbol{X}$, such that $I(\boldsymbol{F}_a; C)$, the mutual information between newly extracted features $\boldsymbol{F}_a$ and output class C, becomes as close as to $I(\boldsymbol{X}; C)$, the mutual information between original features $\boldsymbol{X}$ and output class C.

4.2 Algorithm: ICA-FX for Binary-Class Problems

In this section, a feature extraction method ICA-FX is proposed for binary classification problems [60] by modifying a standard ICA algorithm for the purpose presented in the previous section. It is an extension of [36]. The main idea of the proposed method is to incorporate the binary class labels into the structure of standard ICA to extract a set of new features that are highly correlated with given class labels, as LDA does but using a method other than orthogonal projection.

Consider the structure shown in Fig. 4.1. Here, the original feature vector $\boldsymbol{X} = [X_1, \cdots, X_N]^T$ is fully connected to $\boldsymbol{U} = [U_1, \cdots, U_N]$, class label C is connected to $\boldsymbol{U}_a = [U_1, \cdots, U_M]$, and $U_{N+1} = C$. In the figure, the weight

matrix $\boldsymbol{W} \in \Re^{(N+1)\times(N+1)}$ becomes

$$\boldsymbol{W} = \begin{pmatrix} w_{1,1} & \cdots & w_{1,N} & w_{1,N+1} \\ \vdots & & \vdots & \vdots \\ w_{M,1} & \cdots & w_{M,N} & w_{M,N+1} \\ w_{M+1,1} & \cdots & w_{M+1,N} & 0 \\ \vdots & & \vdots & \vdots \\ w_{N,1} & \cdots & w_{N,N} & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{4.2}$$

And let us denote the upper left $N \times N$ matrix of $\boldsymbol{W}$ as W.

Now our aim is to separate the input feature space $\boldsymbol{X}$ into two linear subspaces: one that is spanned by $\boldsymbol{F}_a = [F_1, \cdots, F_M]^T$ that contains maximal information about the class label C, and the other spanned by $\boldsymbol{F}_b = [F_{M+1}, \cdots, F_N]^T$ that is independent of C as much as possible.

The condition for this separation can be derived as follows. If it is assumed that the weight matrix $\boldsymbol{W}$ is nonsingular, it can be seen that $\boldsymbol{X}$ and $\boldsymbol{F} = [F_1, \cdots, F_N]^T$ span the same linear space and it can be represented with a direct sum of $\boldsymbol{F}_a$ and $\boldsymbol{F}_b$. Then by the *data processing inequality*, the following inequality is obtained:

$$\begin{aligned} I(\boldsymbol{X}; C) =& I(W\boldsymbol{X}; C) \\ =& I(\boldsymbol{F}; C) \\ =& I(\boldsymbol{F}_a, \boldsymbol{F}_b; C) \\ \geq& I(\boldsymbol{F}_a; C). \end{aligned} \tag{4.3}$$

The first equality holds because W is nonsingular and in the inequality on the last line, a necessary condition for the equality is $I(\boldsymbol{F}_b; C) = I(U_{M+1}, \cdots, U_N; C) = 0$.

If this is possible, one can reduce the dimension of input feature space from N to $M(< N)$ by using only $\boldsymbol{F}_a$ instead of $\boldsymbol{X}$, without losing any information about the target class.

To solve this problem, the feature extraction problem is interpreted in the structure of the blind source separation (BSS) problem as shown in Fig. 4.2. The detailed description of each step is as follows:

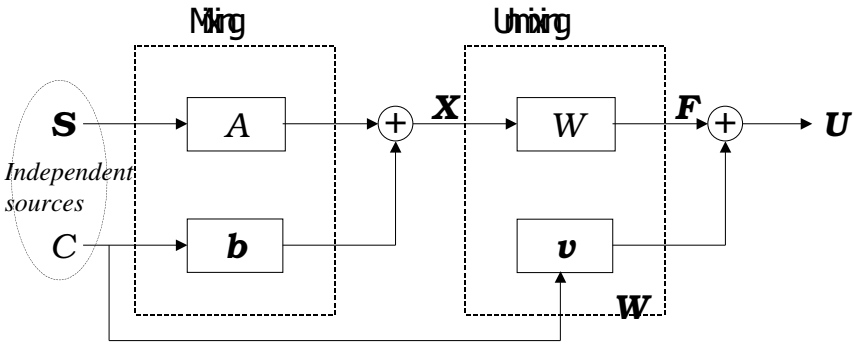


Figure 4.2: Interpretation of Feature Extraction in the BSS structure

(Mixing) Assume that there exist N independent sources $\boldsymbol{S} = [S_1, \cdots, S_N]^T$ which are also independent of class label C. Assume also that the observed feature vector $\boldsymbol{X}$ is the linear combination of the sources $\boldsymbol{S}$ and C with the mixing matrix $A \in \Re^{N \times N}$ and $\boldsymbol{b} \in \Re^{N \times 1}$; i.e.,

$$\boldsymbol{X} = A\boldsymbol{S} + \boldsymbol{b}C. \tag{4.4}$$

(Unmixing) Our unmixing stage is a little different from the BSS problem as shown in Fig. 4.1. Let us denote the last column of $\boldsymbol{W}$ without the $(N+1)$th element as $\boldsymbol{v} \in \Re^{N \times 1}$. Then the unmixing equation becomes

$$\boldsymbol{U} = W\boldsymbol{X} + \boldsymbol{v}C. \tag{4.5}$$

Suppose that $\boldsymbol{U}$ has been made somehow equal to $\boldsymbol{E}$, the scaled and permuted version of source $\boldsymbol{S}$; i.e.,

$$\boldsymbol{E} \triangleq \Lambda\Pi\boldsymbol{S} \tag{4.6}$$

where Λ is a diagonal matrix corresponding to an appropriate scale and Π is a permutation matrix. Then, U_i's $(i = 1, \cdots, N)$ are independent of class C, and among the elements of $\boldsymbol{F} = W\boldsymbol{X} (= \boldsymbol{U} - \boldsymbol{v}C)$, $\boldsymbol{F}_b = [F_{M+1}, \cdots, F_N]^T$ will be independent of C because $v_i = w_{i,N+1} = 0$ for $i = M+1, \cdots, N$. Thus, one can extract $M (< N)$ dimensional new feature vector $\boldsymbol{F}_a$ by a linear transformation of $\boldsymbol{X}$ containing the maximal information about the class if the relation $\boldsymbol{U} = \boldsymbol{E}$ holds.

Now that the feature extraction problem is set in a similar form as the standard BSS or ICA problem, a learning rule for $\boldsymbol{W}$, can be derived using the the similar approach for the derivation of a learning rule for ICA. Because the Infomax approach, the MLE approach, and the negentropy maximization approach were shown to lead to the identical learning rule for ICA problems, as mentioned in Section 2.3, any approach can be used for the derivation. In this thesis, the MLE approach is used to obtain a learning rule.

If it is assumed that $\boldsymbol{U} = [U_1, \cdots, U_N]^T$ is a linear combination of the source $\boldsymbol{S}$; i.e., it is made to be equal to $\boldsymbol{E}$, a scaled and permutated version of the source $\boldsymbol{S}$ as in (4.6), and that each element of $\boldsymbol{U}$ is independent of other elements of $\boldsymbol{U}$ and it is also independent of class C, the log likelihood of the given data becomes

$$L(\boldsymbol{x}, c|\boldsymbol{W}) = \log|\det \boldsymbol{W}| + \sum_{i=1}^{N} \log p_i(u_i) + \log p(c) \tag{4.7}$$

because

$$p(\boldsymbol{x}, c|\boldsymbol{W}) = |\det \boldsymbol{W}|\ p(\boldsymbol{u}, c) = |\det \boldsymbol{W}|\ \prod_{i=1}^{N} p_i(u_i)\ p(c). \tag{4.8}$$

Now, L is to be maximized, and this can be achieved by the steepest ascent method. Because the last term in (4.7) is a constant, differentiating (4.7) with respect to $\boldsymbol{W}$ leads to

$$\begin{aligned} \frac{\partial L}{\partial w_{i,j}} &= \frac{adj(w_{j,i})}{|\det \boldsymbol{W}|} - \varphi_i(u_i)x_j \qquad 1 \le i, j \le N \\ \frac{\partial L}{\partial w_{i,N+1}} &= -\varphi_i(u_i)c \qquad\qquad\qquad 1 \le i \le M \end{aligned} \tag{4.9}$$

where $adj(\cdot)$ is adjoint and $\varphi_i(u_i) = -\frac{dp_i(u_i)}{du_i}/p_i(u_i)$. Note that c has binary numerical values corresponding to the two categories.

It can be seen that $|\det \boldsymbol{W}| = |\det W|$ and $adj(w_{j,i})/|\det \boldsymbol{W}| = W_{i,j}^{-T}$. Thus the learning rule becomes

$$\begin{aligned} \Delta W &\propto\ W^{-T} - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{x}^T \\ \Delta \boldsymbol{v}_a &\propto\ -\boldsymbol{\varphi}(\boldsymbol{u}_a)c. \end{aligned} \tag{4.10}$$

Since the two terms in (4.10) have different tasks regarding the update of separate matrices W and W_{N+1}, the learning process can be divided, and applying

natural gradient on updating W, it is obtained:

$$\begin{aligned} W^{(t+1)} =& W^{(t)} + \mu_1[I_N - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{f}^T]W^{(t)} \\ \boldsymbol{v}_a^{(t+1)} =& \boldsymbol{v}_a^{(t)} - \mu_2\boldsymbol{\varphi}(\boldsymbol{u}_a)c. \end{aligned} \tag{4.11}$$

Here $\boldsymbol{v}_a \triangleq [w_{1,N+1}, \cdots, w_{M,N+1}]^T \in \Re^M$, $\boldsymbol{\varphi}(\boldsymbol{u}) \triangleq [\varphi_1(u_1), \cdots, \varphi_N(u_N)]^T$, $\boldsymbol{\varphi}(\boldsymbol{u}_a) \triangleq [\varphi_1(u_1), \cdots, \varphi_M(u_M)]^T$, I_N is a $N \times N$ identity matrix, and μ_1 and μ_2 are learning rates that can be set differently. By this updating rule, the assumption that u_i's are independent of one another and of c will most be likely fulfilled by the resulting u_i's.

Note that the learning rule for W is the same as the original ICA learning rule [30], and also note that $\boldsymbol{F}_a$ corresponds to the first M elements of $W\boldsymbol{X}$. Therefore, one can extract the optimal features $\boldsymbol{F}_a$ by the proposed algorithm when it finds the optimal solution for W by (4.11).

4.3 Stability of ICA-FX

In this part, the conditions of local stability of the ICA-FX algorithm shown in [60], [62] is presented. The local stability analysis in this thesis undergoes almost the same procedure as that of general ICA algorithms in [63].

Stationary points

To begin with, let us first investigate the stationary point of the learning rule given in (4.11). Let us define

$$A_\star \triangleq A(\Lambda\Pi)^{-1}. \tag{4.12}$$

Now assuming that the output $\boldsymbol{U}$ is made to be equal to $\boldsymbol{E}$, then (4.4), (4.5), and (4.6) become

$$\begin{aligned} \boldsymbol{X} &= A_\star\boldsymbol{E} + \boldsymbol{b}C \\ \boldsymbol{E} &= W\boldsymbol{X} + \boldsymbol{v}C \end{aligned} \tag{4.13}$$

and it becomes

$$(I_N - WA_\star)\boldsymbol{E} = (W\boldsymbol{b} + \boldsymbol{v})C. \tag{4.14}$$

Because C and $\boldsymbol{E}$ are assumed to be independent of each other, W and $\boldsymbol{v}$ must satisfy

$$\begin{aligned} W &= A_\star^{-1} = \Lambda\Pi A^{-1} \\ \boldsymbol{v} &= -W\boldsymbol{b} = -A_\star^{-1}\boldsymbol{b} = -\Lambda\Pi A^{-1}\boldsymbol{b} \end{aligned} \tag{4.15}$$

if $\boldsymbol{U}$ were made to be equal to $\boldsymbol{E}$. This solution is a stationary point of learning rule (4.11) by the following theorem.

Theorem 1 *The W and $\boldsymbol{v}$ satisfying (4.15) is a stationary point of the learning rule (4.11), and the scaling matrix Λ is uniquely determined up to a sign change in each component.*

Proof: See appendix A. ■

In most cases, odd increasing activation functions φ_i are used for ICA, and if the same is done for the ICA-FX, one can get the unique scale up to a sign and W and $\boldsymbol{v}$ in (4.15) is a stationary point.

Local asymptotic stability

Now let us investigate the condition for the stability of the stationary point given in (4.15). In doing so a new version of weight matrix Z and a set of scalars k_i's are introduced such that

$$\begin{aligned} W^{(t)} &= Z^{(t)}W^* \\ v_i^{(t)} &= k_i^{(t)}v_i^* \ (\neq 0), \qquad 1 \le i \le M \end{aligned} \tag{4.16}$$

to follow the same procedure as in [63]. Here W^* and v_i^* are the optimal values of W and v_i which are $A_\star^{-1}$ and $-(A_\star^{-1}\boldsymbol{b})_i$, respectively. Note that the stability of W and v_i in the vicinity of W^* and v_i^* is equivalent to the stability of Z and k_i in the vicinity of the identity matrix I_N and 1.

If W^{*-1} is multiplied to both sides of the learning rule for W in (4.11), it becomes

$$Z^{(t+1)} = \{I_N - \mu_1 G(Z^{(t)}, \boldsymbol{k}^{(t)})\}Z^{(t)} \tag{4.17}$$

where the (i,j)th element of $G \in \Re^{N\times N}$ is

$$\begin{aligned} G(Z^{(t)}, \boldsymbol{k}^{(t)})_{ij} &= \varphi_i(u_i) f_j - \delta_{ij} \\ &= \begin{cases} \varphi_i((Z^{(t)}W^*\boldsymbol{x})_i + k_i^{(t)} v_i^* c)(Z^{(t)}W^*\boldsymbol{x})_j - \delta_{ij} & \text{if } \ 1 \le i \le M \\ \varphi_i((Z^{(t)}W^*\boldsymbol{x})_i)(Z^{(t)}W^*\boldsymbol{x})_j - \delta_{ij} & \text{if } \ M < i \le N \end{cases}. \end{aligned} \tag{4.18}$$

Here, it is denoted $\boldsymbol{k} = [k_1, \cdots, k_M]^T$ for convenience.

In the learning rule for $\boldsymbol{v}_a$, to avoid difficulties in the derivation of the stability condition, the notation of the weight update rule for $\boldsymbol{v}_a$ in (4.11) near the stable point $\boldsymbol{v}_a^*$ is modified a little as follows:

$$v_i^{(t+1)} = v_i^{(t)} - \mu_i^{(t)} \varphi_i(u_i) c v_i^* v_i^{(t)}, \qquad 1 \le i \le M. \tag{4.19}$$

Here it is assumed that the learning rate $\mu_i^{(t)}(>0)$ changes over time t and varies with different index i such that it satisfies $\mu_i^{(t)} v_i^{(t)} v_i^* = \mu_2$. The modification is justified because $v_i^{(t)} v_i^* \cong v_i^{*2}$ is positive when $v_i^{(t)}$ is near a stationary point v_i^*. Note that the modification applies only after $\boldsymbol{v}_a$ has reached sufficiently near a stable point $\boldsymbol{v}_a^*$.

Using the fact that $v_i^{(t)} = k_i^{(t)} v_i^*$ (4.19) can be rewritten as

$$k_i^{(t+1)} = [1 - \mu_i^{(t)} g_i(Z^{(t)}, \boldsymbol{k}^{(t)})] k_i^{(t)}, \qquad 1 \le i \le M \tag{4.20}$$

where

$$\begin{aligned} g_i(Z^{(t)}, \boldsymbol{k}^{(t)}) &= \varphi_i(u_i) c \\ &= \varphi_i((Z^{(t)}W^*\boldsymbol{x})_i + k_i^{(t)} v_i^* c) v_i^* c \end{aligned} \tag{4.21}$$

Using the weight update rules (4.17) and (4.20) for the new variables Z and K, the local stability condition is obtained in the following theorem.

Theorem 2 *The local asymptotic stability of the stationary point of the proposed algorithm is governed by the nonlinear moment*

$$\kappa_i = \mathrm{E}\{\dot{\varphi}_i(E_i)\}\mathrm{E}\{E_i^2\} - \mathrm{E}\{\varphi_i(E_i)E_i\} \tag{4.22}$$

and it is stable if

$$1 + \kappa_i > 0, \quad 1 + \kappa_j > 0, \quad (1 + \kappa_i)(1 + \kappa_j) > 1 \tag{4.23}$$

for all $1 \leq i, j \leq N$. *Thus the sufficient condition is*

$$\kappa_i > 0, \quad 1 \leq i \leq N. \tag{4.24}$$

Here $\mathrm{E}\{\cdot\}$ *is the expectation.*

Proof: See appendix B. ■

Because the condition for the stability of the ICA-FX in Theorem 2 is identical to that of the standard ICA in [63], the interpretation of the nonlinear moment κ_i can be consulted to [63]. Just stating the key point here, the local stability is preserved when the activation function $\varphi_i(e_i)$ is chosen to be positively correlated with the true activation function $\varphi_i^*(e_i) \triangleq -\dot{p}_i(e_i)/p_i(e_i)$.

Thus, as the standard ICA algorithm, the choice of activation function $\varphi_i(e_i)$ is of great importance, and the performance of ICA-FX depends heavily on the function $\boldsymbol{\varphi}(\boldsymbol{e})$, which is determined by the densities $p_i(e_i)$'s. But in practical situations, these densities are mostly unknown, and true densities are approximated by some model densities, generally given by (i) momentum expansion, (ii) a simple parametric model not far from Gaussian, or (iii) a mixture of simple parametric models [64]. In this work, one does not need an exact approximation of the density $p_i(u_i)$ because we do not have physical sources like in BSS problems. Therefore, the extended Infomax algorithm [52], one of the approximation methods belonging to type (ii), is used because of its computational efficiency and wide applications.

4.4 Extension of ICA-FX to Multi-Class Problems

In this section, ICA-FX is extended to multi-class problems [61]. The problem to be solved in this section is as follows:

> **(Problem statement)** Assume that there are a normalized input feature vector, $\boldsymbol{X} = [X_1, \cdots, X_N]^T$, and an output class, $\mathcal{C} \in \{\mathcal{C}_1, \cdots, \mathcal{C}_{N_c}\}$.

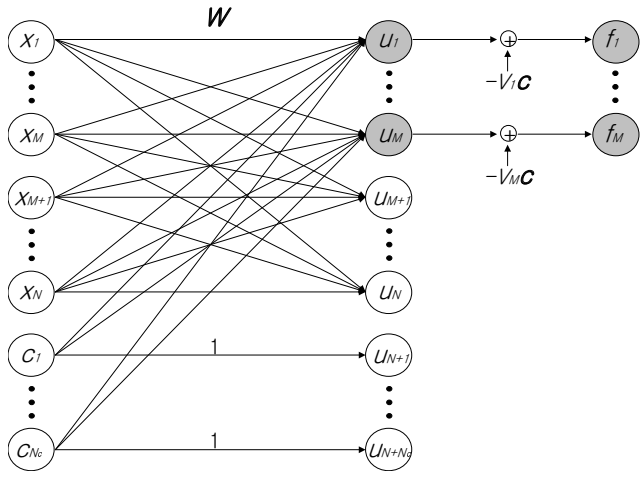


Figure 4.3: ICA-FX for multi-class problems

The purpose of feature extraction is to extract $M(\leq N)$ new features $\boldsymbol{F_a} = [F_1, \cdots, F_M]^T$ from $\boldsymbol{X}$, by a linear combination of the X_i's, containing the maximum information on class $\mathcal{C}$.

First, suppose $N_c(\geq 2)$ denotes the number of classes. To incorporate the class labels in the ICA structure, the discrete class labels need to be encoded into numerical variables. The 1-of-N_c scheme is used in coding classes, i.e., a class vector, $\boldsymbol{C} = [C_1, \cdots, C_{N_c}]^T$, is introduced and if a class label, $\mathcal{C}$, belongs to the lth value $\mathcal{C}_l$, then C_l is activated as 1 and all the other C_i's, $i \neq l$, are set to -1. After all the training examples are presented, each $C_i, i = 1, \cdots, N_c$, is shifted in order to have zero mean and are scaled to have a unit variance.

Now consider the structure shown in Fig. 4.3. Here, the original feature vector $\boldsymbol{X}$ is fully connected to $\boldsymbol{U} = [U_1, \cdots, U_N]$, the class vector $\boldsymbol{C}$ is connected only to $\boldsymbol{U}_a = [U_1, \cdots, U_M]$, and $U_{N+l} = C_l$, $l = 1, \cdots, N_c$. In the figure, the

weight matrix $\boldsymbol{W} \in \Re^{(N+N_c)\times(N+N_c)}$ becomes

$$\begin{aligned}\boldsymbol{W} &= \left(\begin{array}{c|c} W & V \\ \hline \mathbf{0}_{N_c,N} & I_{N_c} \end{array}\right) \\ &= \left(\begin{array}{ccc|ccc} w_{1,1} & \cdots & w_{1,N} & w_{1,N+1} & \cdots & w_{1,N+N_c} \\ \vdots & & \vdots & \vdots & & \vdots \\ w_{M,1} & \cdots & w_{M,N} & w_{M,N+1} & \cdots & w_{M,N+N_c} \\ w_{M+1,1} & \cdots & w_{M+1,N} & & & \\ \vdots & & \vdots & & \mathbf{0}_{N-M,N_c} & \\ w_{N,1} & \cdots & w_{N,N} & & & \\ \hline \multicolumn{3}{c|}{\mathbf{0}_{N_c,N}} & \multicolumn{3}{c}{I_{N_c}} \end{array}\right).\end{aligned} \tag{4.25}$$

where $W \in \Re^{N\times N}$ and $V = [V_a^T, \mathbf{0}_{N-M,N_c}^T]^T \in \Re^{N\times N_c}$. Here the first nonzero M rows of V is denoted as $V_a \in \Re^{M\times N_c}$.

The mixing and unmixing stages and the underlying assumptions for the multi-class ICA-FX is almost the same as the binary ICA-FX. Consequently, the derivation of the learning rule for this takes exact the same steps as in the ICA-FX for binary-class problems and the learning rule becomes as follows:

$$\begin{aligned} W^{(t+1)} &= W^{(t)} + \mu_1[I_N - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{f}^T]W^{(t)} \\ V_a^{(t+1)} &= V_a^{(t)} - \mu_2\boldsymbol{\varphi}(\boldsymbol{u}_a)\boldsymbol{c}^T. \end{aligned} \tag{4.26}$$

The stability condition of the ICA-FX for binary classification problems in [60] [62] can also be easily extended to multi-class ICA-FX as follows:

Theorem 3 *The local asymptotic stability of the ICA-FX around the stationary point* $(W = \Lambda\Pi A^{-1}, V = -\Lambda\Pi A^{-1}B)$ *is governed by the nonlinear moment*

$$\kappa_i = E\{\dot{\varphi}_i(E_i)\}E\{E_i^2\} - E\{\varphi_i(E_i)E_i\} \tag{4.27}$$

and it is stable if

$$1 + \kappa_i > 0, \quad 1 + \kappa_j > 0, \quad (1 + \kappa_i)(1 + \kappa_j) > 1 \tag{4.28}$$

for all $1 \le i, j \le N$. *Therefore, the sufficient condition is*

$$\kappa_i > 0, \quad 1 \le i \le N. \tag{4.29}$$

Because the proof of the theorem is almost the same as that for binary-class problems, the proof is omitted.

Now, the properties of the ICA-FX is discussed in terms of the suitability of the proposed algorithm for the classification problems.

4.5 Properties of ICA-FX

In ICA-FX, given a new instance consisting of N features $\boldsymbol{X} = [X_1, \cdots, X_N]$, it is transformed into an M-dimensional new feature vector $\boldsymbol{F}_a = [F_1, \cdots, F_M]$ and it is used to estimate which class the instance belongs to. In the following, it is discussed why ICA-FX is suitable for the classification problems in the statistical sense.

Consider a normalized zero-mean binary output class C, with its density

$$p_c(c) = p_1\delta(c - c_1) + p_2\delta(c - c_2), \tag{4.30}$$

where $\delta(\cdot)$ is a dirac delta function, and p_1, p_2 are the probabilities that class C takes values c_1 and c_2, respectively.

Suppose that U_i $(i = 1, \cdots, N)$ has density $p_i(u_i)$, which is sub-Gaussian $(p_i(u_i) \propto N(\mu, \sigma^2) + N(-\mu, \sigma^2))$ or super-Gaussian $(p_i(u_i) \propto N(0, \sigma^2)\mathrm{sech}^2(u_i))$ as in [52], where $N(\mu, \sigma^2)$ is the normal density with mean μ and variance σ^2. Then the density of F_i $(i = 1, \cdots, M)$ is proportional to the convolution of two densities $p_i(u_i)$ and $p_c(-c/w_{i,N+1})$ by the assumption that U_i's and C are independent; i.e.,

$$\begin{aligned} p(f_i) =& \frac{1}{|w_{i,N+1}|} p_i(u_i) * p_c(-\frac{c}{w_{i,N+1}}) \\ \propto& \begin{cases} p_1 N(-w_{i,c}c_1, \sigma^2)\mathrm{sech}^2(f_i + w_{i,N+1}c_1) \\ \quad + p_2 N(-w_{i,N+1}c_2, \sigma^2)\mathrm{sech}^2(f_i + w_{i,N+1}c_2) \\ \qquad\qquad \text{if } p_i(u_i)\text{: super-Gaussian} \\ p_1 N(\mu - w_{i,N+1}c_1, \sigma^2) + p_2 N(\mu - w_{i,N+1}c_2, \sigma^2) \\ \quad + p_1 N(-\mu - w_{i,N+1}c_1, \sigma^2) + p_2 N(-\mu - w_{i,N+1}c_2, \sigma^2) \\ \qquad\qquad \text{if } p_i(u_i)\text{: sub-Gaussian} \end{cases} \end{aligned} \tag{4.31}$$

because $f_i = u_i - w_{i,N+1}c$.

Figure 4.4 shows the densities of super- and sub-Gaussian models of u_i and the corresponding densities of F_i for varying $w_{i,N+1} = [0 \cdots 4]$. In the figure, it is set $\mu = 1$, $\sigma = 1$, $p_1 = p_2 = 0.5$, and $c_1 = -c_2 = 1$. It can be seen in Fig. 4.4 that super-Gaussian is sharper than sub-Gaussian at peak. For the super-Gaussian model of U_i, it can be seen that as $w_{i,N+1}$ grows, the density of F_i has two peaks, which are separated from each other, and the shape is quite like a sub-Gaussian model with a large mean. For the sub-Gaussian model of U_i, it also takes two peaks as the weight $w_{i,N+1}$ grows, though the peaks are smoother than those of super-Gaussian. In both cases, as $w_{i,N+1}$ grows, the influence of output class C becomes dominant in the density of F_i, and the classification problem becomes easier: for a given F_i check if it is larger than zero and then associate it with the corresponding class C.

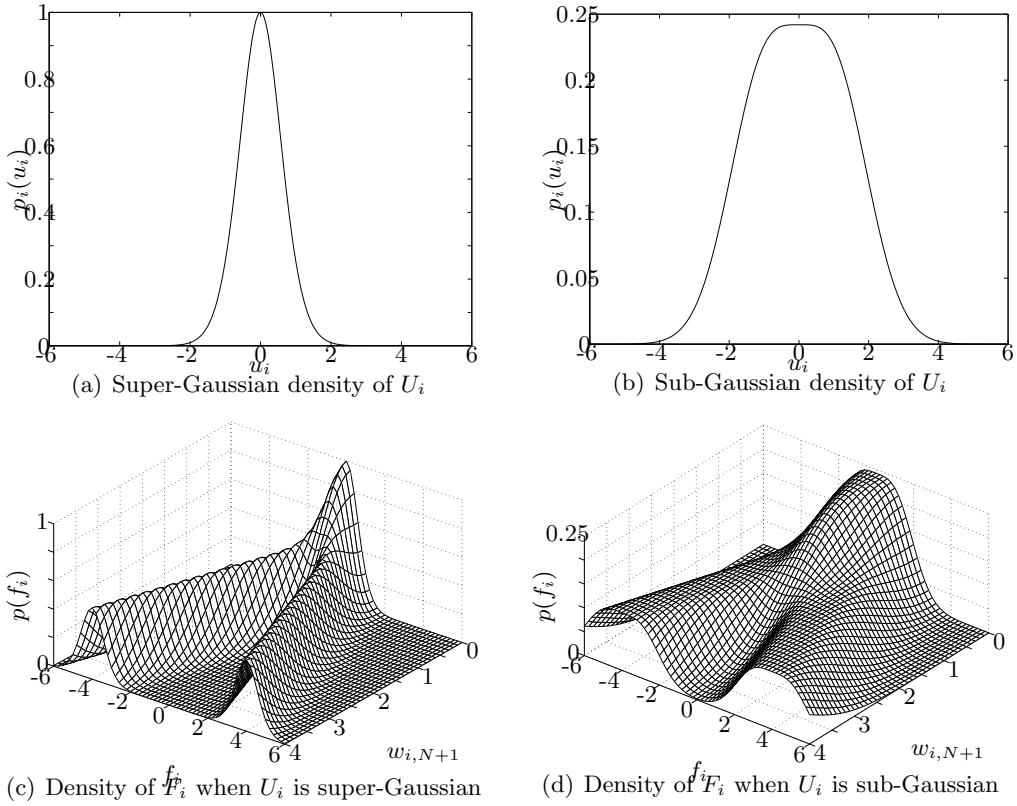
This phenomenon can be interpreted as a discrete source estimation problem in a noisy channel, as shown in Fig. 4.5. If the class C is regarded as an input and U_i as noise, the goal is to estimate C through channel output F_i. Because it is assumed that C and U_i's are independent, the higher the signal-to-noise ratio (SNR) becomes, the more class information is conveyed in the channel output F_i. The SNR can be estimated using powers of source and noise, which in this case leads to the following estimation:

$$SNR = \frac{\mathrm{E}\{C^2\}}{\mathrm{E}\{(U_i/w_{i,N+1})^2\}}. \tag{4.32}$$

Therefore, if one can make large $w_{i,N+1}$, the noise power in Fig. 4.5 is suppressed and the source C can be easily estimated.

In many real-world problems, as the number of input features increases, the contribution of class C to U_i becomes small; i.e., $w_{i,N+1}$ becomes relatively small such that the density of F_i is no longer bimodal. Even if this is the case, the density has a flatter top that looks like a sub-Gaussian density model, which is easier to estimate classes than those with normal densities.

As in standard ICA, applying PCA before conducting the ICA-FX can enhance the performance of the ICA-FX much more. Therefore, the PCA was used in all the following experimental results before applying the ICA-FX.



(a) Super-Gaussian density of U_i

(b) Sub-Gaussian density of U_i

(c) Density of F_i when U_i is super-Gaussian

(d) Density of F_i when U_i is sub-Gaussian

Figure 4.4: Super- and sub-Gaussian densities of U_i and corresponding densities of F_i ($p_1 = p_2 = 0.5$, $c_1 = -c_2 = 1$, $\mu = 1$, and $\sigma = 1$).

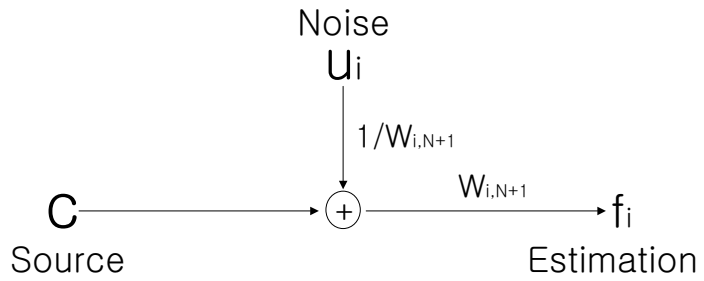


Figure 4.5: Channel representation of feature extraction

4.6 Experimental Results of ICA-FX for Binary Classification Problems

In this section some experimental results of ICA-FX for binary classification problems will be presented to show the characteristics of the ICA-FX. In order to show the effectiveness of the proposed algorithm, the same number of features were selected from both the original features and the extracted features and the classification performances were compared. For comparison, the PCA and LDA were also performed to extract features. In the selection of features for original data, the MIFS-U [8], [9], which makes use of the mutual information between input features and output class in ordering the significance of features, was used. It is noted that the simulation results can vary depending on the initial condition of the rate updating rule because there may be many local optimum solutions.

Simple problem

Suppose there are two input features x_1 and x_2 uniformly distributed on [-1,1] for a binary classification, and the output class y is determined as follows:

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

Here, $y = 0$ corresponds to $c = -1$ and $y = 1$ corresponds to $c = 1$.

Plotting this problem on a three-dimensional space of (x_1, x_2, y) leads to Fig. 4.6 where the class information, as well as the input features, correspond to each axis, respectively. The data points are located in the shaded areas in this problem. As can be seen in the figure, this problem is linearly separable and clearly shows the necessity of feature extraction; if $x_1 + x_2$ is extracted as a new feature, perfect classification is possible with only one feature. But feature extraction algorithms based on conventional unsupervised learning, such as the conventional PCA and ICA, cannot extract $x_1 + x_2$ as a new feature because they only consider the input distribution; i.e., they only examine (x_1, x_2) space. For problems of this kind, feature selection methods in [8], [9] also fail to find adequate features because they have no ability to construct new features by themselves. Note that other

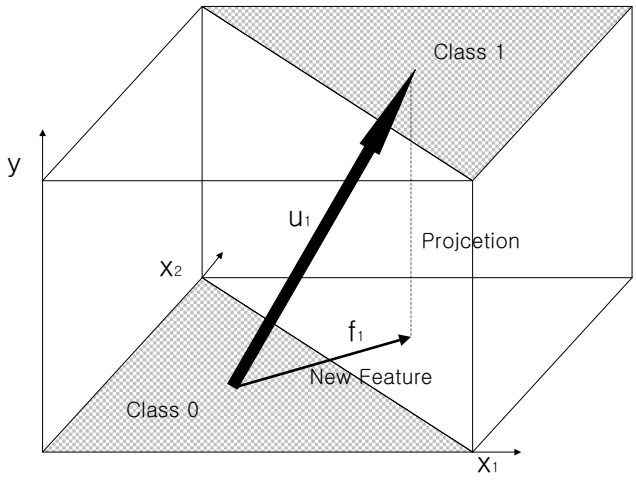


Figure 4.6: ICA-FX for a simple problem

feature extraction methods using supervised algorithms such as LDA and MMI can solve this problem.

For this problem, the ICA-FX was performed with $M = 1$ and could get $u_1 = 43.59x_1 + 46.12x_2 + 36.78y$ from which a new feature $f_1 = 43.59x_1 + 46.12x_2$ is obtained. To illustrate the characteristic of ICA-FX on this problem, u_1 is plotted as a thick arrow in Fig. 4.6 and f_1 is the projection of u_1 onto the (x_1, x_2) feature space.

IBM datasets

The IBM datasets were generated by Agrawal et al. [15] as stated in Section 3.4. Each of the datasets has nine attributes: *salary, commission, age, education level, make of the car, zipcode of the town, value of the house, years house owned,* and *total amount of the loan.* The data generation code was downloaded from [65]. It can generate about 100 different datasets among which three datasets are used in this part. They are *pred 4*, *pred 5*, and *pred 9* which correspond to IBM2, IBM3, and IBM1 in Table 4.1 respectively. Note that these datasets are different from those used in Section 3.4. The proposed algorithm ICA-FX was tested for these datasets and the classification performances were compared with those of

Table 4.1: IBM Data sets

IBM1	
Group A:	$0.33 \times (salary + commission) - 30000 > 0$
Group B:	Otherwise.
IBM2	
Group A:	$0.67 \times (salary + commission) - 5000 \times ed\_level - 20000 > 0$
Group B:	Otherwise.
IBM3	
Group A:	$0.67 \times (salary + commission) - 5000 \times ed\_level - loan/5 - 10000 > 0$
Group B:	Otherwise.

other feature extraction methods.

As can be seen from Table 4.1, these datasets are linearly separable and use only a few features for classification. Total 1000 instances were generated for each dataset with noise of zero mean and either 0% or 10% of SNR added to the attributes, among which 66% were used as training data while the others were reserved for test. In the training, C4.5 [14], one of the most popular decision-tree algorithms which gives deterministic classification rules, and a three-layered MLP were used. To show the effectiveness of our feature extraction algorithm, the performance of ICA-FX was compared with PCA, LDA, and the original data with various number of features. For the original data, the feature selection algorithm MIFS-U, which selects good features among candidate features, was used before training. In training C4.5, all the parameters were set as the default values in [14], and for MLP, three hidden nodes were used with a standard back-propagation (BP) algorithm with zero momentum and a learning rate of 0.2. After 300 iterations, the training of the network was stopped.

The experimental results are shown in Table 4.2. In the table, the performance of the original features selected with MIFS-U and the newly extracted features were compared with those of PCA, LDA, and ICA-FX. Because this is a binary classification problem, standard LDA extracts only one feature for all cases. The classification performances on the test set trained with C4.5 and BP are presented in Table 4.2. The parentheses after the classification performance of C4.5 contain the sizes of the decision trees.

Table 4.2: Experimental results for IBM data (Parentheses are the sizes of the decision trees of c4.5)

IBM1					
Noise power	No. of features	Classification performance (%) (C4.5/MLP)			
		MIFS-U	PCA	LDA	ICA-FX
0%	1	87.6(3)/85.8	53.0(3)/55.6	82.2(3)/84.0	96.8(3)/97.0
	2	97.8(25)/97.8	85.4(21)/85.8	–	99.6(3)/97.6
	all	97.8(27)/97.6	89.4(49)/90.2		99.6(3)/97.8
10%	1	82.0(3)/81.4	53.0(3)/56.2	81.2(3)/81.4	92.6(3)/91.8
	2	89.4(21)/90.2	81.6(37)/81.6	–	92.6(11)/92.8
	all	87.6(47)/87.8	87.4(49)/88.0	–	92.4(17)/92.2
IBM2					
Noise power	No. of features	Classification performance (%) (C4.5/MLP)			
		MIFS-U	PCA	LDA	ICA-FX
0%	1	89.4(5)/91.0	87.0(3)/87.2	96.4(3)/96.6	97.8(7)/98.0
	2	96.6(5)/97.0	89.6(13)/89.4	–	98.8(15)/98.4
	3	98.8(25)/98.8	89.6(13)/89.8	–	98.8(17)/98.8
	all	98.8(23)/98.6	93.8(33)/95.2	–	99.0(25)/98.8
10%	1	90.0(5)/90.6	87.0(3)/87.0	94.6(9)/95.2	96.2(5)/96.8
	2	94.8(13)/95.6	85.6(19)/86.0	–	94.8(13)/96.8
	3	96.0(13)/95.2	85.6(23)/85.0	–	95.2(19)/97.0
	all	95.0(21)/94.6	92.2(23)/92.4	–	95.8(29)/97.4
IBM3					
Noise power	No. of features	Classification performance (%) (C4.5/MLP)			
		MIFS-U	PCA	LDA	ICA-FX
0%	1	85.0(3)/85.0	55.4(3)/55.4	92.2(3)/92.2	93.2(3)/94.2
	2	91.2(31)/91.4	61.8(7)/63.8	–	93.6(15)/96.4
	3	90.6(29)/91.8	65.8(23)/66.0	–	97.0(3)/97.0
	4	90.2(33)/92.0	65.8(27)/66.4	–	96.8(21)/97.4
	all	92.4(65)/98.2	88.8(113)/89.6	–	97.8(39)/100.0
10%	1	84.8(3)/84.4	52.2(3)/52.2	89.0(3)/90.0	92.2(3)/93.0
	2	88.4(21)/89.6	58.8(11)/61.4	–	93.4(5)/93.2
	3	86.8(31)/88.8	63.0(11)/64.0	–	94.4(15)/94.0
	4	87.4(41)/87.0	63.0(15)/64.2	–	93.4(19)/94.2
	all	89.4(57)/92.6	79.8(103)/81.8	–	92.4(49)/93.6

As can be seen from Table 4.2, C4.5 and BP produce similar classification performances on these data sets. For all three of the problems, ICA-FX outperformed other methods. It also can be seen that PCA performed worst in all cases, even worse than the original features selected with MIFS-U. This is because PCA can be thought as an unsupervised feature extraction method, and the ordering of its principle components has nothing to do with the classification. Note that the performances with 'all' features are different for different feature extraction/selection methods, although they operate on the same space of all the features. They operate on the same amount of information about the class. But the classifier systems do not make full use of the information. Also note that for all the three datasets with 10% noise, the performances of ICA-FX with one feature is quite better than those with all nine original features of MIFS-U. This clearly shows the advantages of feature extraction; by feature extraction, better generalization performance is expected with reduced dimensionality. The comparison of the tree sizes of C4.5 on these datasets also shows that the computational complexity can also be greatly reduced by feature extraction. For example, the tree size of ICA-FX with one feature is 3 while that corresponds to all original features is 47 for the case of IBM1 with 10% noise.

In the cases of 0% noise power, very good performance was achieved for all the cases with only one feature. In fact, in IBM1 and IBM2, the first feature selected among the original ones was *salary*, while the newly extracted feature with $M = 1$ corresponds to $(salary + commission)$ and $(salary + commission - 6500 \times ed\_level)$, respectively. Comparing these with Table 4.1, one can see these are very good features for classification. The small numbers of tree size for extracted features compared to that for the other methods show our feature extraction algorithm can be utilized to generate oblique decision trees resulting in rules easy to understand. For the case of 10% SNR, ICA-FX also performed better than others in most cases. From these results, it is seen that ICA-FX performs excellently, especially for linearly separable problems.

Table 4.3: Brief Information of the UCI Data sets Used

Name	No. of features	No. of instances	No. of classes
Sonar	60	208	2
Breast Cancer	9	699	2
Pima	8	768	2

UCI datasets

The UCI machine learning repository contains many real-world data sets that have been used by numerous researchers [59]. In this subsection, experimental results of the proposed extraction algorithm was presented for some of these data sets. Table 4.3 shows the brief information of the data sets used in this thesis. Conventional PCA, ICA, and LDA algorithms were conducted on these datasets and various numbers of features were extracted and the classification performances were compared with that of the ICA-FX. Because there is no measure on relative importance among independent components from ICA, the MIFS-U was used in selecting the important features for the classification. For comparison,the MIFS-U was also conducted on the original datasets and the performance were reported.

As classifier systems, MLP, C4.5, and SVM were used. For all the classifiers, input values of the data were normalized to have zero means and standard deviations of one. In training 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 networks were trained for 1,000 iterations. The parameters of C4.5 were set to default values in [14]. For SVM, the 'mySVM' program by Stefan Ruping of University of Dortmund [66] was used. For the kernel function, radial (Gaussian) kernel was used and the other parameters were set as default. Because the performance of the radial kernel SVM critically depends on the value of γ, SVM has been conducted with various values of $\gamma = 0.01 \sim 1$ and the maximum classification rate was reported. Thirteen-fold cross-validation was used for the sonar dataset and ten-fold cross-validation was used for the others. For MLP, ten experiments were conducted for each dataset and the averages and the standard

deviations are reported in this thesis.

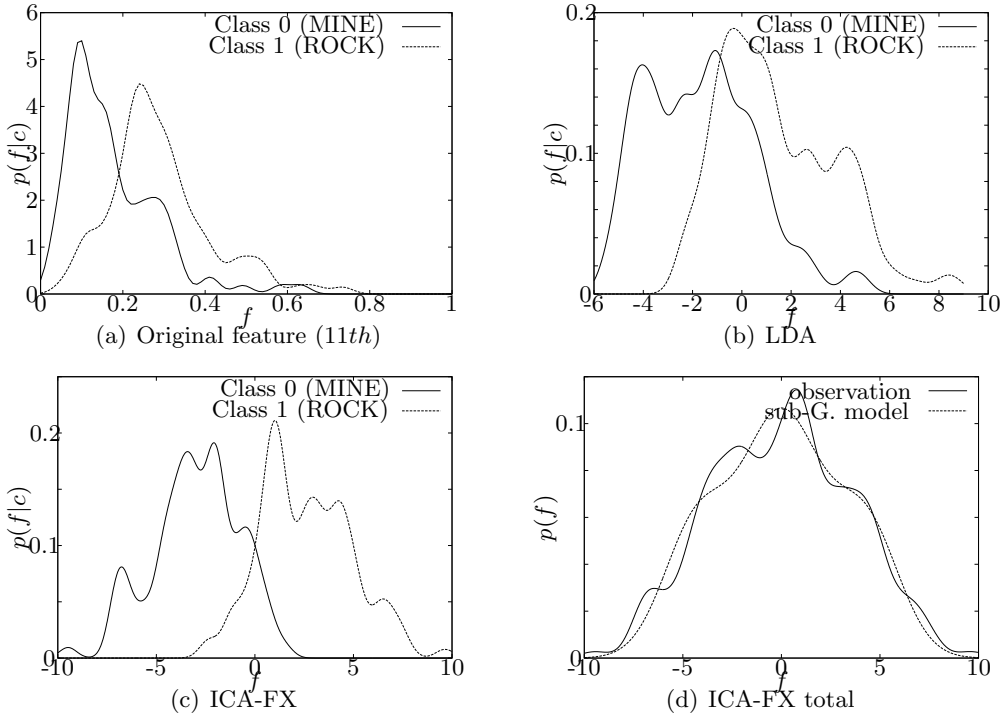
Sonar dataset

Here, the same dataset used in Section 3.4 is also used. It consists of 208 instances, with 60 features and two output classes: *mine/rock*. In this experiment, 13-fold cross validation were used in getting the performances as follows. The 208 instances were divided randomly into 13 disjoint sets with 16 cases in each. For each experiment, 12 of these sets are used as training data, while the 13th is reserved for testing. The experiment is repeated 13 times so that every case appears once as part of a test set.

The training was conducted with MLP, C4.5, and SVM for various numbers of features. Table 4.4 shows the result of our experiment. The reported performance for MLP is an average over the 10 experiments and the numbers in parentheses denote the standard deviation. The result shows that the extracted features from ICA-FX perform better than the original ones, especially when the number of features to be selected is small. In the table, one can see that the performances of ICA-FX are almost the same for small numbers of features and far better than when all the 60 features were used. From this phenomenon, it can be inferred that all the available information about the class is contained in the first feature. As in the IBM datasets, this shows the advantage of feature extraction. The performance of ICA-FX with one feature is more than 4% better than that with all the original features when SVM is used. Considering the fact that SVM is very insensitive to noises or outliers and it does not suffer from the 'curse of dimensionality' much, the generalization performance of feature extraction by ICA-FX can be concluded very effective for this dataset. The differences become over 10% when C4.5 and MLP are used as classifier systems.

Note that the performances of unsupervised feature extraction methods PCA and ICA are not as good as expected. From this, one can see that the unsupervised methods of feature extraction are not good choices for the classification problems.

The first three figures in Fig. 4.7 are the estimates of conditional densities $p(f|c)$'s (class-specific density estimates) of the first selected feature among the



(a) Original feature ($11th$) (b) LDA

(c) ICA-FX (d) ICA-FX total

Figure 4.7: Probability density estimates for a given feature (Parzen window method with window width 0.2 was used)

Table 4.4: Classification performance for Sonar Target data (Parentheses are the standard deviations of 10 experiments)

No. of features	Classification performance (%) (C4.5/MLP/SVM)				
	MIFS-U	PCA	ICA	LDA	ICA-FX
1	73.1/74.8(0.32)/ 74.8	52.4/59.3(0.41)/ 58.6	65.9/67.9(0.25)/ 67.2	71.2/75.2(0.37)/ 74.1	87.5/87.3(0.17)/ 87.1
3	70.2/72.9(0.58)/ 75.5	51.0/57.9(0.42)/ 54.7	63.0/71.1(0.45)/ 69.7	–	86.1/88.1(0.37)/ 89.0
6	69.7/77.5(0.24)/ 80.8	64.9/63.8(0.72)/ 63.0	61.2/69.9(0.63)/ 70.2	–	85.6/86.4(0.42)/ 87.1
9	81.7/80.1(0.61)/ 79.9	69.7/71.2(0.67)/ 70.2	61.5/68.7(0.62)/ 68.7	–	83.2/85.0(0.83)/ 88.8
12	79.3/79.5(0.53)/ 81.3	73.1/74.0(0.64)/ 75.1	60.1/71.4(0.71)/ 71.7	–	78.2/83.4(0.49)/ 86.6
60	73.1/76.4(0.89)/ 82.7	73.1/75.5(0.96)/ 82.7	63.9/74.1(1.43)/ 77.0	–	73.1/80.0(0.78)/ 84.2

original features by MIFS-U (which is the 11th of 60 features), the feature extracted by LDA, and the feature extracted by ICA-FX with $M = 1$. The density estimates were used with the well known Parzen window method [39] using both training and test data. In applying Parzen window, the window width parameter was set to 0.2. The result shows that the conditional density of the feature from ICA-FX is much more balanced than those of the original and LDA in the feature space. In the figures of 6.(a),(b),(c), if the domain for $p(f|c = 0) \neq 0$ and the domain for $p(f|c = 1) \neq 0$ do not overlap, then no error can be made in classification. One can see that the overlapping region of the two classes is much smaller in ICA-FX than the other two. This is why the performance of ICA-FX is far better than the others with only one feature. The density estimate $p(f)$ of the feature from ICA-FX is presented in Fig. 4.7(d). Note that in Fig. 4.7(d), the distribution of the feature from ICA-FX is much flatter than the Gaussian distribution and looks quite like the density of feature f_i obtained with sub-Gaussian model. The dotted line of Fig. 4.7(d) is the density of sub-Gaussian model shown in Fig. 4.4(d) with $w_{i,N+1} = 1.5$.

Table 4.5: Classification performance for Breast Cancer data (Parentheses are the standard deviations of 10 experiments)

No. of features	Classification performance (%) (C4.5/MLP/SVM)				
	MIFS-U	PCA	ICA	LDA	ICA-FX
1	91.1/92.4(0.03)/ 92.7	85.8/86.1(0.05)/ 85.8	84.7/81.5(0.29)/ 85.1	96.8/96.6(0.07)/ 96.9	97.0/97.1(0.11)/ 97.0
2	94.7/95.8(0.17)/ 95.7	93.3/93.8(0.07)/ 94.7	87.3/85.4(0.31)/ 90.3	–	96.5/97.1(0.09)/ 97.1
3	95.8/96.2(0.15)/ 96.1	93.8/94.7(0.11)/ 95.9	89.1/85.6(0.33)/ 91.3	–	96.7/96.9(0.12)/ 96.9
6	95.0/96.1(0.08)/ 96.7	94.8/96.6(0.15)/ 96.6	90.4/90.0(0.59)/ 94.3	–	95.9/96.7(0.27)/ 96.7
9	94.5/96.4(0.13)/ 96.7	94.4/96.8(0.16)/ 96.7	91.1/93.0(0.84)/ 95.9	–	95.5/96.9(0.13)/ 96.6

Wisconsin Breast Cancer dataset

This database was obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg [67]. The data set consists of nine numerical attributes and two classes, which are benign and malignant. It contains 699 instances with 458 benign and 241 malignant. There are 16 missing values in our experiment and these were replaced with average values of corresponding attributes.

The performances of ICA-FX were compared with those of PCA, ICA, LDA, and the original features selected with MIFS-U. The classification results are shown in Table 4.5. As in the sonar dataset, the data were trained with C4.5, MLP, and SVM. The meta-parameters for C4.5, MLP, and SVM are the same as those for the sonar problem. For verification, 10-fold cross validation is used. In the table, classification performances are present and the numbers in parentheses are standard deviations of MLP over 10 experiments.

The result shows that with only one extracted feature, one can get nearly the maximum classification performance that can be achieved with at least two or three original features. It shows that feature extraction is desirable for this classification problem. The reduced feature space by feature extraction is a lot

Table 4.6: Classification performance for Pima data (Parentheses are the standard deviations of 10 experiments)

No. of features	Classification performance (%) (C4.5/MLP/SVM)				
	MIFS-U	PCA	ICA	LDA	ICA-FX
1	72.8/74.1(0.19)/ 74.5	67.8/66.2(0.17)/ 66.3	69.7/71.6(0.17)/ 73.2	74.5/75.2(0.23)/ 75.6	76.0/78.6(0.11)/ 78.7
2	74.2/76.7(0.13)/ 75.8	75.0/74.4(0.23)/ 75.1	72.7/76.8(0.24)/ 76.7	–	75.2/78.2(0.25)/ 78.1
3	74.1/76.3(0.27)/ 76.8	74.2/75.1(0.23)/ 75.5	72.7/76.7(0.54)/ 76.8	–	75.7/76.7(0.18)/ 77.8
5	73.3/75.3(0.64)/ 76.6	73.7/75.2(0.39)/ 75.5	72.9/76.4(0.55)/ 77.2	–	77.2/77.8(0.38)/ 78.3
8	74.5/76.5(0.45)/ 78.1	74.5/76.6(0.31)/ 78.1	72.3/77.0(0.62)/ 77.9	–	72.9/76.7(0.48)/ 78.0

easier to work with in the sense of computational complexity and data storage. The performance of LDA is almost the same as ICA-FX for this problem.

Pima Indian Diabetes dataset

This dataset consists of 768 instances in which 500 are class 0 and the other 268 are class 1. It has 8 numeric features with no missing value.

For this data, PCA, ICA, LDA, and ICA-FX were applied, and their performances were compared. Original features selected by MIFS-U were also compared. In training, C4.5, MLP, and SVM were used. The meta-parameters for the classifiers were set to be equal to the previous cases. For verification, 10-fold cross validation was used.

In Table 4.6, classification performances are presented. As shown in the table, the performance of ICA-FX is better than those of other methods regardless of what classifier system was used when the number of features is small. It is also seen that the performances of different methods get closer as the number of extracted features becomes large. Note also that for ICA-FX, the classification rate of one feature is as good as those of the other cases where more features are used.

4.7 Face Recognition by Multi-Class ICA-FX

Face recognition is one of the most actively studied pattern recognition fields where the problem is that the dimension of the raw data is so large that feature extraction is inevitable. Because all the pattern recognition problems can be considered as data mining problems as mentioned in Introduction, the feature extraction method described in the early part of this chapter can be directly used for face recognition. In addition to the previously mentioned advantages of feature extraction, feature extraction for face recognition may be used as a coding scheme for the compression of face images. Before the experimental results of ICA-FX for face recognition problems are presented, some of the most popular face recognition techniques are briefly reviewed.

Many subspace methods have been successfully applied to construct features of an image [68] – [71]. Among these, the Eigenface [68] (based on PCA) and Fisherface [69] (based on LDA) methods are popular, because they allow the efficient characterization of a low-dimensional subspace whilst preserving the perceptual quality of a very high-dimensional raw image.

Though it is the most popular, the Eigenface method [68], by its nature, is not suitable for classification problems since it does not make use of any output class information in computing the principal components (PC). The main drawback of this method is that the extracted features are not invariant under the transformation. Merely scaling the attributes changes resulting features. In addition, it does not use higher order statistics and it has been reported that the performance of the Eigenface method is severely affected by the level of illumination [69].

Unlike the Eigenface method, the Fisherface method [69] focuses on the classification problems to determine optimal linear discriminating functions for certain types of data whose classes have a Gaussian distribution and the centers of which are well separated. Although it is quite simple and powerful for classification problems, it cannot produce more than $N_c - 1$ features, where N_c is the number of classes. As in the Eigenface method, it only uses second order statistics in representing the images. Some researchers have proposed subspace methods using higher order statistics such as the evolutionary pursuit and kernel methods for

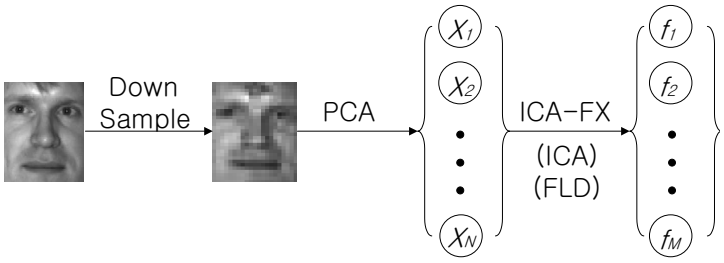


Figure 4.8: Experimental procedure

face recognition [70], [71].

Recently, some researchers have shown that ICA is more powerful for face recognition than the PCA [72] [73]. Unlike PCA and LDA, ICA uses higher order statistics and has been applied successfully in recognizing faces with changes in pose [72], and classifying facial actions [73]. This method was applied to face recognition and facial expression problems. The proposed algorithm greatly reduces the dimension of feature space while improving the classification performance.

In this section, the ICA-FX for multi-class classification problems is applied to face recognition problems and the performance is compared with those of the other methods such as PCA, pure ICA, and LDA. This is an extension of [74] where face recognition problems were viewed as multiple binary classification problems and the binary version of the ICA-FX [60] [62] was used to tackle the problems.

To apply the ICA-FX to face recognition problems, firstly the original features $\boldsymbol{X}$ of an image, which will be used to obtain new features $\boldsymbol{F}_a$, need to be determined. There are several methods for determining the features of an image, such as wavelets, Fourier analysis, fractal dimensions, and many other methods [75]. Among them, one can easily come up with an idea of using each pixel as one feature. Though this is the most simple method without losing any information of an image, the dimension of feature space by this method becomes too large to be handled easily. In this thesis, each image is downsampled into a manageable size in order to reduce the computational complexity. Subsequently, each down-

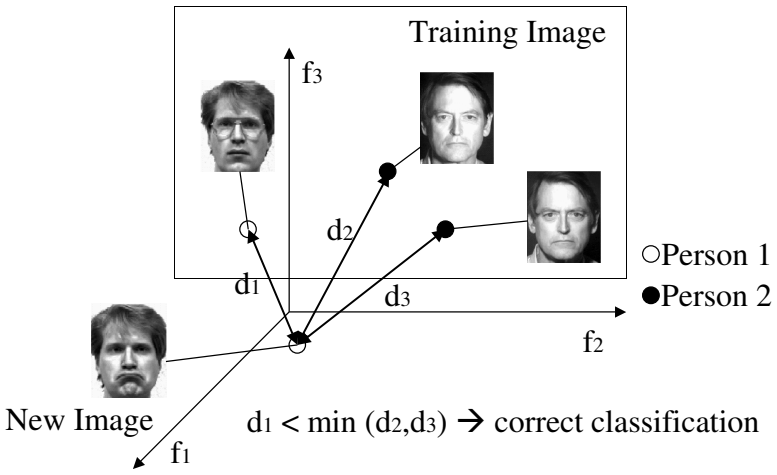


Figure 4.9: Example of one nearest neighborhood classifier

sampled pixel is transformed to have a zero mean and unit variance, and PCA is then performed both as a whitening process of the ICA-FX and for the purpose of further reducing the dimension of the feature space. Therefore, X_i corresponds to the coefficient of the ith principal component of a given image. Finally, the main routine of the ICA-FX is applied to extract the valuable features for classification. Figure 4.8 shows the experimental procedure used in this thesis. For comparison, ICA and LDA are also used after PCA is performed, as shown in the figure. The performances are tested with the leave-one-out scheme and the classifications are performed using the one nearest neighborhood classifier. That is, to test the ith image among the total n images, all the other $(n-1)$ images are used for training and the ith image is classified as the identity of the image whose Euclidean distance from the ith image is the closest among the $(n-1)$ images. Figure 4.9 is a typical example of classification using one nearest neighborhood. In the figure, each image is projected onto the feature space (f_1, f_2, f_3) and if the features are good, the distance of d_1 will be the smallest among the three distances $d_1 \sim d_3$ and the classification will be correct.

The ICA-FX is applied to the Yale [69] and AT&T [76] face databases for face recognition, and to the Japanese Female Facial Expression (JAFFE) [77] database

Figure 4.10: Yale Database

for classifying facial expressions. Throughout the experiments, the learning rates μ_1 and μ_2 for the ICA-FX are set to 0.002 and 0.1 respectively and the number of iterations for learning is set to 300. The ICA results are obtained by an extended infomax algorithm [52] with a learning rate of 0.002 and 300 iterations.

Yale Database

The Yale face database consists of 165 grayscale images of 15 individuals. There are 11 images per subject with different facial expressions or configurations. In [69], the authors report two types of databases: a closely cropped set and a full face set. In this thesis, the closely cropped set was used and the images were downsampled into 21 $\times$ 30 pixels. Figure 4.10 represents the downsampled images of the first three individuals of the dataset.

For the data, PCA was first performed on 630 downsampled pixels and various numbers of principal components were used as the inputs of the ICA, LDA and ICA-FX. Figure 4.11 represents the typical weights of PCA, ICA, LDA, and ICA-FX. The top row is the first 10 principal components (PC) among 165 PC's, which are generally referred to as Eigenfaces. The third row is the first 10 out of 14 Fisherfaces that are the weights of LDA. The second and the fourth rows are the weights of ICA and ICA-FX respectively. Here, the first 30 principal components were used as inputs to ICA, LDA, and ICA-FX and ten features were extracted using the ICA-FX.

Figure 4.12 shows the performances of PCA, ICA, LDA, and ICA-FX when different numbers of principal components were used in the face classification.

Figure 4.11: Weights of various subspace methods for Yale dataset. (1st row: PCA (Eigenfaces), 2nd row: ICA, 3rd row: LDA (Fisherfaces), 4th row: ICA-FX)

Note that the number of features produced by the LDA is 14, because there are 15 subjects in this dataset, while the number of features by ICA is the same as that of PCA. In the ICA-FX, the number of features was set to 10. Because ICA and the ICA-FX can have different results according to the initial weight randomization, the results of ICA and ICA-FX are averages of two experiments. From the figure, it can be seen that the performance of ICA-FX is better than those of the other methods regardless of the number of principal components that are used as inputs to the ICA-FX.

Note that the error rate decreases as the number of principal components increases in ICA-FX. In other methods, the error rates decrease in the beginning as the number of features increases but they increase as the number of features further increases.

Figure 4.13 shows the performance of the ICA-FX with various numbers of extracted features (M in Section III) when the number of principal components (N in Section III) was fixed to 30, 40, and 50. In the figure, it can be seen that the performances are better when $10 \sim 20$ features are extracted and the error rates tend to grow as the number of extracted features increases for all the three cases. This phenomenon can be explained by what is referred to as 'the law of parsimony', or 'Occam's razor' [78]. The unnecessarily large number of features

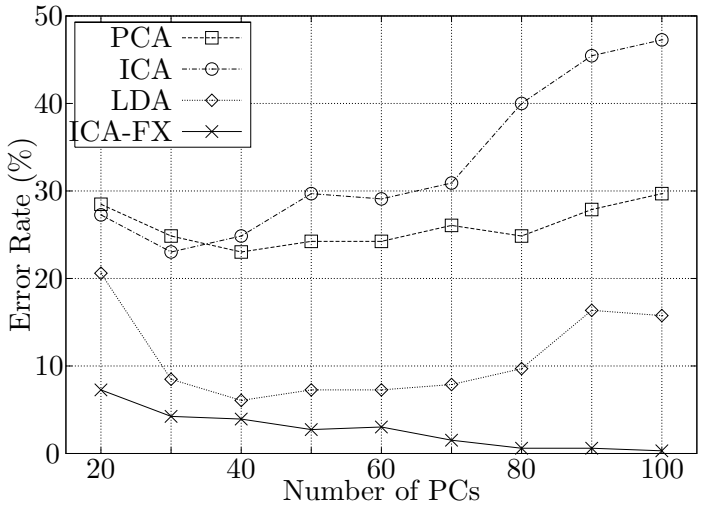


Figure 4.12: Comparison of performances of PCA, ICA, LDA, and ICA-FX on Yale database with various number of PC's. (The numbers of features for LDA and ICA-FX are 14 and 10 respectively. The number of features for ICA is the same as that of PCA)

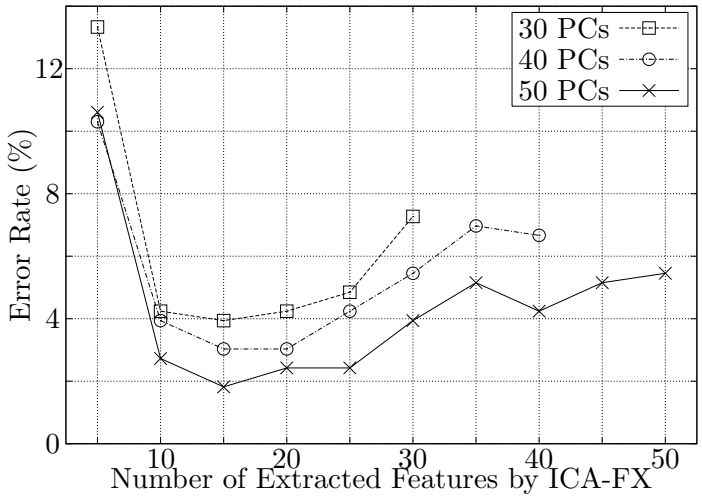


Figure 4.13: Performances of ICA-FX on Yale database with various number of features used. (30, 40, and 50 principal components were used as inputs to ICA-FX.)

Table 4.7: Experimental results on Yale database

Method	Dim. of Reduced Space	No. of Error	Error Rate (%)
Eigenface (PCA)	30	41	24.85
ICA	30	38	23.03
Fisherface (LDA)	14	14	8.48
Kernel Eigenface (d=3)	60	40	24.24
Kernel Fisherface (G)	14	10	6.06
ICA-FX (binary)	14	6	3.64
ICA-FX (multi)	**10**	**7**	**4.24**

degrades the classification performance.

To provide insights on how the ICA-FX simplifies the face pattern distribution, each face pattern is projected onto the two dimensional feature space in Fig. 4.14. This figure provides a low-dimensional representation of the data, which can be used to capture the structure of the data. In the figure, the PCA, ICA, LDA, and ICA-FX were used to generate features using all the 165 face images. Thirty principal components are used as inputs of ICA, LDA, and ICA-FX. For ICA-FX 10 features are extracted. The most significant two features are selected as bases for PCA and LDA cases, and the first two features are selected as bases for ICA and ICA-FX. For the sake of simplicity in visualization, the first 7 identities among the total 15 identities are shown in the figure, that is total 77 images are used for the plots. Before plotting, features are normalized to have zero means and unit variances. Seven different symbols such as '+', and '*' are used to represent different identities. Note that the same symbols cluster more closely in the cases of LDA and ICA-FX than those of PCA and ICA as expected.

In Table 4.7, the performance of the ICA-FX was compared with those of the other algorithms: PCA (Eigenface), LDA (Fisherface), ICA, and the kernel methods presented in [71]. In the experiments, PCA was initially conducted on 630 pixels and the first 30 principal components were used as in [69]. Subsequently, the LDA, ICA, and ICA-FX were applied to these 30 principal components. Table 4.7 shows the classification error rates of each methods. In the test, the error rates were determined by the 'leave-one-out' strategy and recognition was

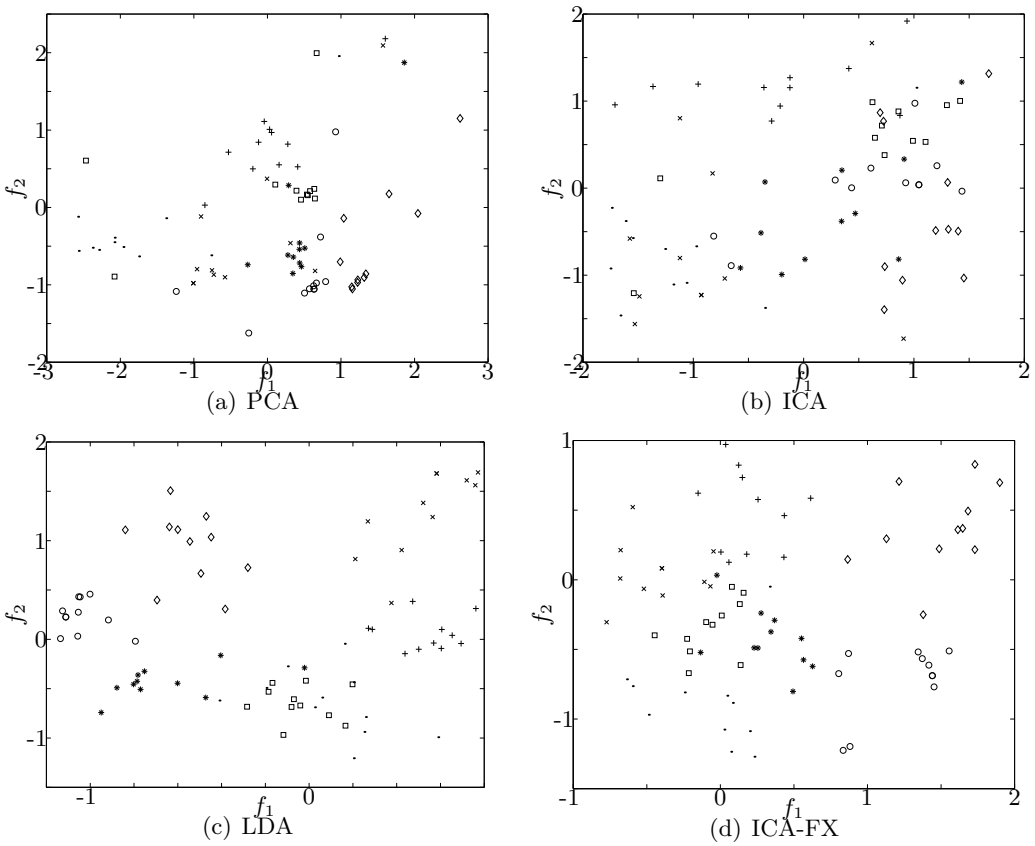


Figure 4.14: Distribution of 7 identities $(\cdot, \circ, *, \times, +, \diamond, \square)$ of Yale data drawn on 2 dimensional subspaces of PCA, ICA, LDA, and ICA-FX.

performed using the one nearest neighbor classifier as in [69]. In the table, the performances of the kernel Eigenface, and the kernel Fisherface are from [71]. From the table it can be seen that ICA-FX outperforms the other methods using a smaller number of features.

For comparison, the performance of the binary version ICA-FX is also reported in the table [74]. In this experiment, the face recognition problem is viewed as multiple binary classification problems or face authentication problems where the purpose is to accept or reject a person as a targeted person. Each image among 165 images is reserved for test and the remaining 164 images are used to train the binary version ICA-FX. Because the binary-class ICA-FX is for two-class problems, the identity of the test image is set as *class 0* and all the other identities are classified as *class 1*. After the features are extracted using binary-class ICA-FX, one nearest neighborhood method is used to test the performance. This procedure is performed for every 165 images and the error rate is 3.64% with 14 features. Note that the performance of multi-class ICA-FX is compatible with that of binary-class ICA-FX. Considering that the face authentication problem is generally easier than the face recognition problems where one must decide who the person really is, it can be concluded that the multi-class ICA-FX is very good at extracting features for face recognition problems.

AT&T Database

The AT&T database of faces (formerly 'The ORL Database of Faces') [76], consists of 400 images, which are ten different images for 40 distinct individuals. It includes various lighting conditions, facial expressions, and facial details. The images were downsampled into 23 $\times$ 28 pixels for efficiency. Figure 4.15 shows the downsampled images of the first three individuals.

The experiments were performed exactly the same way as in the Yale database. Leave-one-out strategy was used with the one nearest neighborhood classifier throughout the experiments. Averages of two experiments for the ICA and ICA-FX are reported here. Figure 4.16 shows the weights of the PCA, ICA, LDA, and ICA-FX for this dataset respectively.

Figure 4.17 shows the error rates of the PCA, ICA, LDA, and ICA-FX when

Figure 4.15: AT&T Database

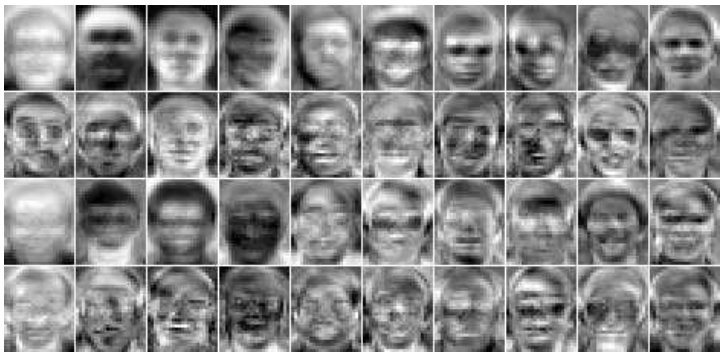


Figure 4.16: Weights of various subspace methods for AT&T dataset. (1st row: PCA (Eigenfaces), 2nd row: ICA, 3rd row: LDA (Fisherfaces), 4th row: ICA-FX)

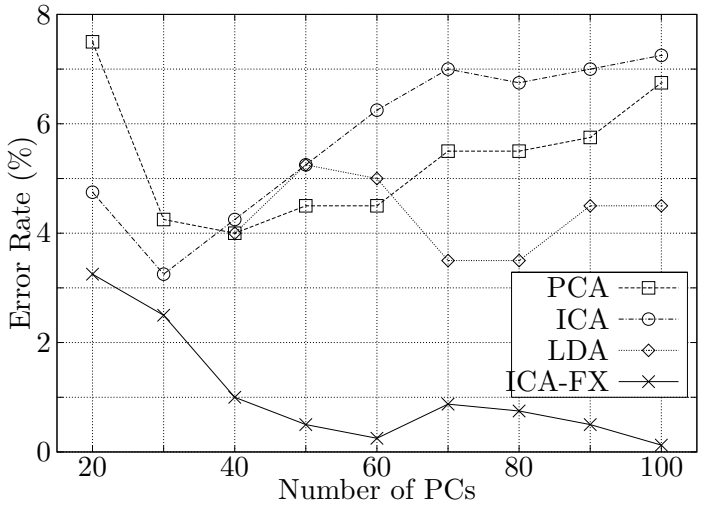


Figure 4.17: Comparison of performances of PCA, ICA, LDA, and ICA-FX on AT&T database with various number of PC's. (The numbers of features for LDA and ICA-FX are 39 and 10 respectively. The number of features for ICA is the same as that of PCA)

different numbers of principal components were used. Note that the number of extracted features by LDA is 39, because there are 40 classes. Because there must be at least 40 PC's to get 39 Fisherfaces, the error rates of the LDA for 20 and 30 PC's are not reported. The number of extracted features for the ICA-FX was set to 10.

Figure 4.18 shows the error rates of the ICA-FX when different numbers of features were used with 40, 50, or 60 principal components. It can be seen that there are little differences in the performance when 10 or more features are extracted and the error rates gradually increase as an increase number of features are extracted in all the cases. This phenomenon is the same as that for the Yale database.

In Fig. 4.19, each face pattern is projected onto the two dimensional feature space. In the figure, the PCA, ICA, LDA, and ICA-FX were used to generate features using all the 400 face images. Forty principal components are used as inputs of ICA, LDA, and ICA-FX. For ICA-FX 10 features are extracted. The most significant two features are selected as bases for PCA and LDA cases, and

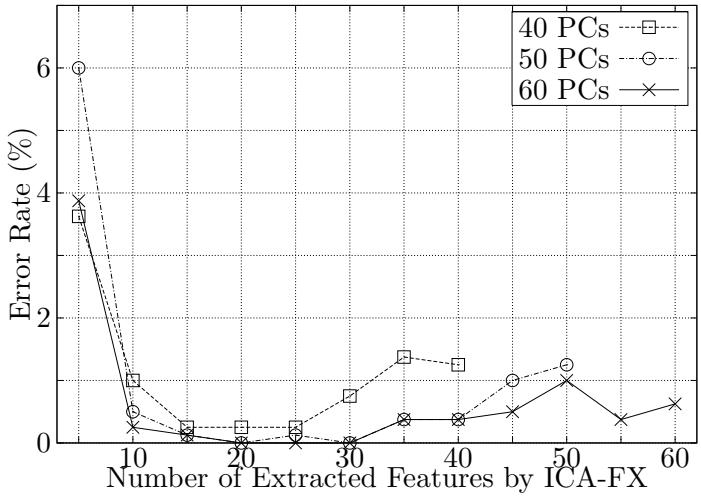


Figure 4.18: Performances of ICA-FX on AT&T database with various number of features used. (40, 50, and 60 principal components were used as inputs to ICA-FX.)

the first two features are selected as bases for ICA and ICA-FX. As in Yale databases, for the sake of simplicity in visualization, the first 7 identities among the total 40 identities are shown in the figure, that is total 70 images are used for the plots. Before plotting, features are normalized to have zero means and unit variances. Seven different symbols are used to represent different identities. In the figure, one can easily separate one cluster of a symbol from another in the cases of LDA and ICA-FX, while it is hard to do so for the cases of PCA and ICA.

Table 4.8 shows the error rates of the PCA, ICA, LDA, the kernel methods, and ICA-FX. For ICA, LDA and ICA-FX, 40 principal components were used for the input vector as in [71]. The performances of the kernel methods are those from [71]. As shown in the table, it can be seen that the ICA-FX outperforms the other methods with significantly less features.

As in the Yale face recognition problem, the performance of the binary version ICA-FX is also reported in the table [74]. The detailed procedure for this experiment is the same as that described in the Yale problem. The performance of the multi-class ICA-FX is nearly the same as that of the binary-class ICA-FX using

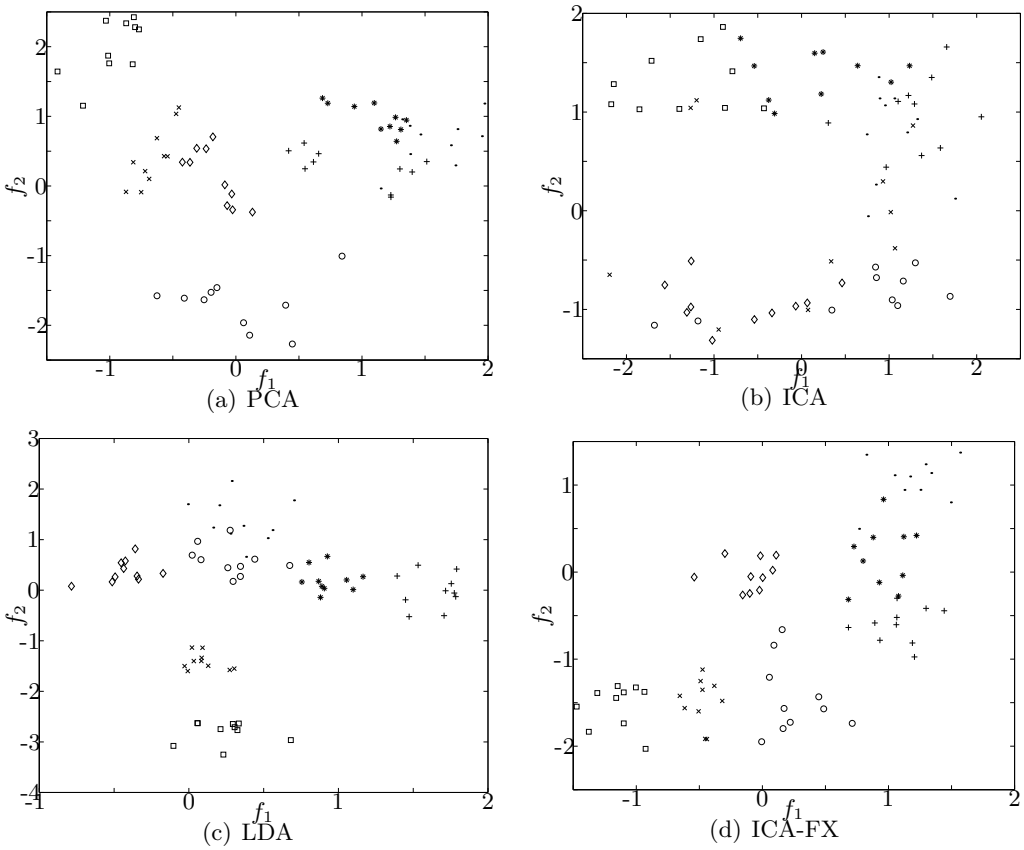


Figure 4.19: Distribution of 7 identities $(\cdot, \circ, *, \times, +, \diamond, \square)$ of AT&T data drawn on 2 dimensional subspaces of PCA, ICA, LDA, and ICA-FX.

Table 4.8: Experimental results on AT&T database

Method	Dim. of Reduced Space	No. of Error	Error Rate (%)
Eigenface (PCA)	40	16	4.00
ICA	40	17	4.25
Fisherface (LDA)	39	16	4.00
Kernel Eigenface (d=3)	40	8	2.00
Kernel Fisherface (G)	39	5	1.25
ICA-FX (binary)	10	4	1.00
ICA-FX (multi)	**10**	**4**	**1.00**

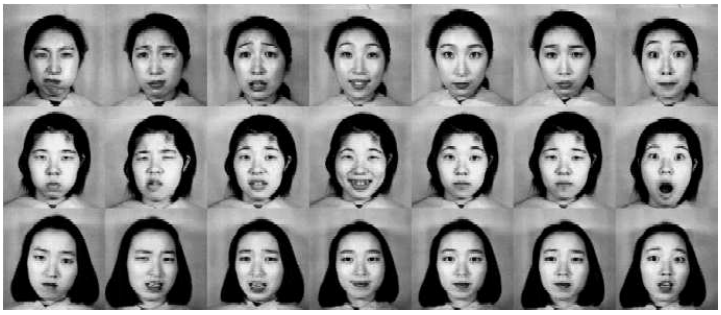


Figure 4.20: JAFFE Database

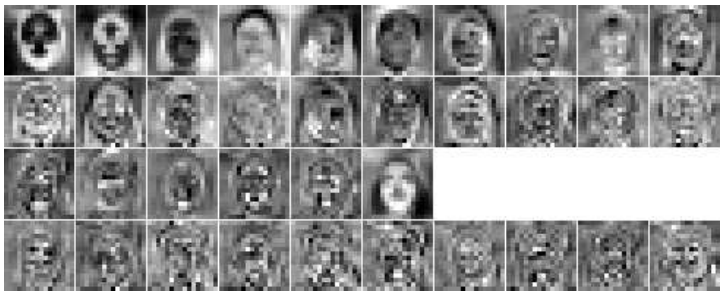


Figure 4.21: Weights of various subspace methods for JAFFE dataset. (1st row: PCA (Eigenfaces), 2nd row: ICA, 3rd row: LDA (Fisherfaces), 4th row: ICA-FX)

the same number of features and it can be concluded again that the multi-class ICA-FX is very good at extracting features for face recognition problems.

JAFFE Database

This database consists of 213 images of 7 facial expressions (angry, disappointed, fearful, happy, sad, surprised, and neutral) posed by 10 Japanese female models [77]. The number of images belonging to each category is shown in Table 4.9. Figure 4.20 shows samples of the images. For the experiments, each image was downsampled to 16×16 and a total of 256 pixels were used. The PCA, ICA, LDA, and ICA-FX were used for recognizing 7 facial expressions and the weights for each method are shown in Fig. 4.21. Note that there are 6 Fisherfaces, because there are 7 categories of facial expression.

The performances of the various methods are shown in Fig. 4.22. In the figure,

Table 4.9: Distribution of JAFFE database

Category	No. of Images	Total Images
Angry	30	
Disappointed	29	
Fearful	32	
Happy	31	213
Sad	31	
Surprised	30	
Neutral	30	

various numbers of principal components were used as the inputs to ICA, LDA and ICA-FX. For ICA-FX, 10 features were extracted. It can be seen that the performances of the LDA is even worse than those of the PCA. The performance of the ICA-FX is much better than those of the other methods. The error rates for ICA-FX decreases consistently as the number of principal components increases, while those of the others do not.

Figure 4.23 shows the error rates of the ICA-FX when different numbers of features were used with 50, 60, or 70 principal components. It is expected from Fig. 4.22 and 4.23, that error rates can be reduced to below 5% with more principal components and 15 features extracted by the ICA-FX.

In Fig. 4.24, each face pattern is projected onto the two dimensional feature space. Features are generated using all the 213 face images. The PCA, ICA, LDA, and ICA-FX were compared. Forty principal components are used as inputs of ICA, LDA, and ICA-FX. For ICA-FX 10 features are extracted. The most significant two features are selected as bases for PCA and LDA cases, and the first two features are selected as bases for ICA and ICA-FX. Before plotting, features are normalized to have zero means and unit variances. Seven different symbols are used to represent seven expressions. Though not clear as in Fig. 4.19, one can see that the localization properties of LDA and ICA-FX is quite better than those of PCA and ICA.

Table 4.10 shows the performances of the PCA, ICA, LDA, and ICA-FX when the first 60 principal components were used. In this table, the number of features by the ICA-FX was set to 10. The experimental results show that the

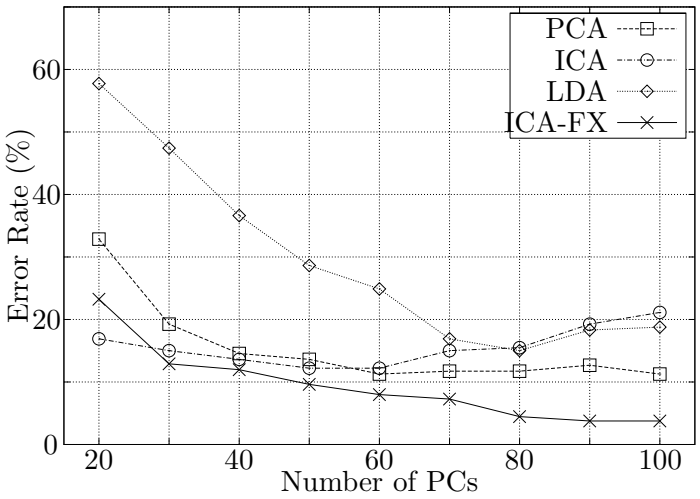


Figure 4.22: Comparison of performances of PCA, ICA, LDA, and ICA-FX on JAFFE database with various number of PC's. (The numbers of features for LDA and ICA-FX are 6 and 10 respectively. The number of features for ICA is the same as that of PCA)

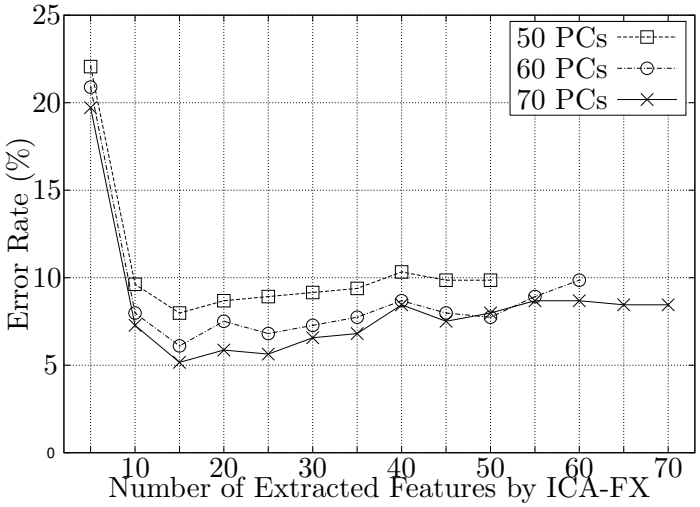


Figure 4.23: Performances of ICA-FX on JAFFE database with various number of features used. (50, 60, and 70 principal components were used as inputs to ICA-FX.)

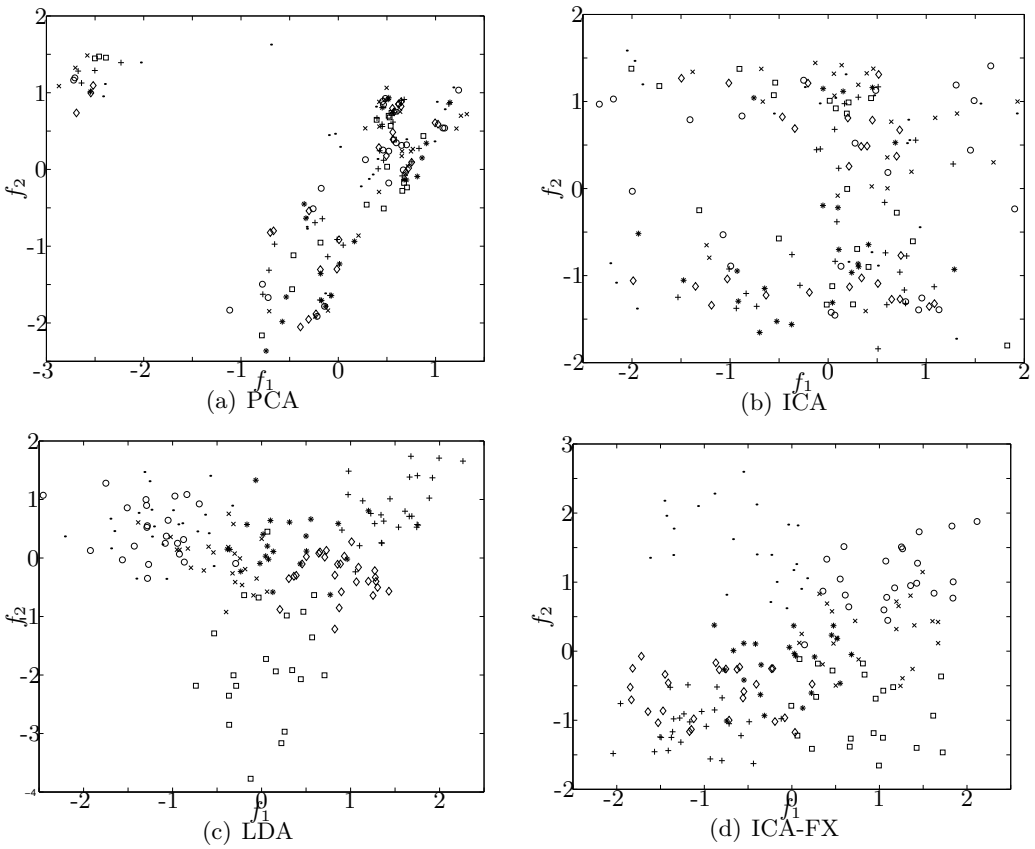


Figure 4.24: Distribution of 7 identities ($\cdot, \circ, *, \times, +, \diamond, \square$) of JAFFE data drawn on 2 dimensional subspaces of PCA, ICA, LDA, and ICA-FX.

Table 4.10: Experimental results on JAFFE database

Method	Dim. of Reduced Space	No. of Error	Error Rate (%)
Eigenface (PCA)	60	24	11.27
ICA	60	26	12.20
Fisherface (LDA)	6	53	24.88
ICA-FX	**10**	**17**	**7.98**

classification rates of the ICA-FX are better than those of the other methods. Furthermore, the performance of the LDA is unsatisfactory in this case. The reason is that the number of extracted features by the LDA is too small to contain sufficient information on the class. When five features are extracted by the ICA-FX, the classification errors are approximately $19 \sim 23\%$ in Fig 4.23, and are close to that of the LDA in Table 4.10.

Chapter 5

Conclusions

In this dissertation, the problems of feature selection and extraction for classification problems have been dealt with. Dimensionality reduction with feature selection or extraction is desirable in the aspect that it can resolve the so called 'curse of dimensionality' problem with better generalization performance. In addition, it also reduces the data storage and the computational complexity in the data mining process afterwards.

Mutual information has been used as a measure of importance of a feature throughout the dissertation. Although the mutual information is a very good indicator of the relevance between variables, the reasons why it is not widely used is its computational difficulties, especially for continuous multi-variables.

In the first part of the dissertation, a method for calculating mutual information between continuous input features and discrete output class is proposed and it is applied to a greedy input feature selection algorithm for classification problems. The proposed method makes use of the Parzen window in getting the conditional density in a feature space. With this method, the mutual information between output class and multiple input features can be computed without requiring a large amount of memory.

The computational complexity of the proposed method is proportional to the square of the given sample size. This might be a limiting factor for huge data sets, but with a simple modification that confines each influence field in a finite area, the computational efforts can be greatly reduced. Furthermore, it is

expected that a clustering or sample selection method can be used to overcome this limitation.

The proposed feature selection method PWFS was applied for several classification problems and better performances were obtained compared to the conventional feature selection methods such as MIFS, MIFS-U, and the stepwise regression.

In the second part of the dissertation, an algorithm ICA-FX is proposed for feature extraction and the stability condition for the proposed algorithm is also provided. Firstly, the ICA-FX is developed for binary-class classification problems and then it is extended to multi-class classification problems. The proposed algorithm is based on the standard ICA and can generate very useful features for classification problems.

Although ICA can be directly used for feature extraction, it does not generate useful information because of its unsupervised learning nature. In the proposed algorithm, class information was added in performing ICA. The added class information plays a critical role in the extraction of useful features for classification. With the additional class information new features containing maximal information about the class can be obtained. The number of extracted features can be arbitrarily chosen.

The stability condition for the proposed algorithm suggests that the activation function $\varphi_i(\cdot)$ should be chosen to well represent the true density of the source. If a squashing function such as sigmoid or logistic is used as an activation function, the true source density should not be Gaussian. If it is so, the algorithm diverges as in standard ICA.

Since it uses the standard feed-forward structure and learning algorithm of ICA, it is easy to implement and train. Experimental results for several data sets show that the proposed algorithm generates good features that outperform the original features and other features extracted from other methods for classification problems. Because the original ICA is ideally suited for processing large datasets such as biomedical ones, the proposed algorithm is also expected to perform well for large-scale classification problems.

The proposed feature selection and extraction algorithms were applied to

several classification problems including face recognition problems. The performances of the proposed methods clearly show the necessity of dimensionality reduction in the data mining process. They outperformed the other compared methods and it can be concluded that the proposed methods is good for selecting or extracting features for classification problems.

Bibliography

[1] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus, "Knowledge discovery in databases: An overview," *AI Magazine*, , no. 3, pp. 57–70, 1992.

[2] U.M Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, American Association for Artificial Intelligence and The MIT Press, 1996.

[3] G.H. John, *Enhancements to the data mining process*, Ph.D. thesis, Computer Science Dept., Stanford University, 1997.

[4] H. Liu and H. Motoda, *Feature Extraction Construction and Selection*, chapter 1, pp. 3–11, Kluwer Academic Publishers, Boston, 1998.

[5] V.S. Cherkassky and I.F. Mulier, *Learning from Data*, chapter 5, John Wiley & Sons, 1998.

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

[7] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. Neural Networks*, vol. 5, no. 4, July 1994.

[8] N. Kwak and C.-H. Choi, "Improved mutual information feature selector for neural networks in supervised learning," in *Proc. Int'l Joint Conf. on Neural Networks 1999*, Washington D.C., July 1999.

[9] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Trans. on Neural Networks*, vol. 13, no. 1, pp. 143–159, Jan. 2002.

[10] K.L Priddy et al., "Bayesian selection of important features for feedforward neural networks," *Neurocomputing*, vol. 5, no. 2 and 3, 1993.

[11] L.M. Belue and K.W. Bauer, "Methods of determining input features for multilayer perceptrons," *Neural Computation*, vol. 7, no. 2, 1995.

[12] J.M. Steppe, K.W. Bauer Jr., and S.K. Rogers, "Integrated feature and architecture selection," *IEEE Trans. Neural Networks*, vol. 7, no. 4, pp. 1007–1014, July 1996.

[13] R. Setiono and H. Liu, "Neural network feature selector," *IEEE Trans. Neural Networks*, vol. 8, no. 3, May 1997.

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

[15] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE Trans. Know. and Data Eng.*, vol. 5, no. 6, Dec. 1993.

[16] D. Xu and J.C. Principe, "Learning from examples with quadratic mutual information," in *Proc. of the 1998 IEEE Signal Processing Society Workshop*, 1998, pp. 155–164.

[17] K. Torkkola and W.M. Campbell, "Mutual information in learning feature transformations," in *Proc. Int'l Conf. Machine Learning*, Stanford, CA, 2000.

[18] K. Torkkola, "Nonlinear feature transforms using maximum mutual information," in *Proc. 2001 Int'l Joint Conf. on Neural Networks*, Washington D.C., July 2001.

[19] N.R. Draper and H. Smith, *Applied Regression Analysis*, John Wiley & Sons, New York, 2nd edition, 1981.

[20] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, MA, 1992.

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

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

[23] T. Okada and S. Tomita, "An optimal orthonormal system for discriminant analysis," *Pattern Recognition*, vol. 18, pp. 139–144, 1985.

[24] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *IEEE Trans. Know. and Data Eng.*, vol. 8, no. 6, Dec. 1996.

[25] K.J. McGarry, S. Wermter, and J. MacIntyre, "Knowledge extraction from radial basis function networks and multi-layer perceptrons," in *Proc. Int'l Joint Conf. on Neural Networks 1999*, Washington D.C., July 1999.

[26] R. Setiono and H. Liu, "A connectionist approach to generating oblique decision trees," *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 29, no. 3, June 1999.

[27] Q. Li and D.W. Tufts, "Principal feature classification," *IEEE Trans. Neural Networks*, vol. 8, no. 1, Jan. 1997.

[28] M. Baldoni, C. Baroglio, D. Cavagnino, and L. Saitta, *Towards automatic fractal feature extraction for image recognition*, pp. 357 – 373, Kluwer Academic Publishers, 1998.

[29] Y. Mallet, D. Coomans, J. Kautsky, and O. De Vel, "Classification using adaptive wavelets for feature extraction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, Oct. 1997.

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

[31] A. Hyvarinen, E. Oja, P. Hoyer, and J. Hurri, "Image feature extraction by sparse coding and independent component analysis," in *Proc. Fourteenth International Conference on Pattern Recognition*, Brisbane, Australia, Aug. 1998.

[32] M. Kotani et. al, "Application of independent component analysis to feature extraction of speech," in *Proc. Int'l Joint Conf. on Neural Networks 1999*, Washington D.C., July 1999.

[33] A.D. Back and T.P. Trappenberg, "Input variable selection using independent component analysis," in *Proc. Int'l Joint Conf. on Neural Networks 1999*, Washington D.C., July 1999.

[34] H.H. Yang and J. Moody, "Data visualization and feature selection: new algorithms for nongaussian data," *Advances in Neural Information Processing Systems*, vol. 12, 2000.

[35] 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.

[36] N. Kwak, C.-H. Choi, and C.-Y. Choi, "Feature extraction using ica," in *Proc. Int'l Conf. on Artificial Neural Networks 2001*, Vienna Austria, Aug. 2001.

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

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

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

[40] K. Fukunaga and D.M. Hummels, "Bayes error estimation using parzen and k-nn procedures," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 9, pp. 634–644, 1987.

[41] S.J. Raudys and A.K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 13, no. 3, pp. 252–264, March 1991.

[42] 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.

[43] C. Meilhac and C. Nastar, "Relevance feedback and catagory search in image databases," in *Proc. IEEE Int'l Conf. on Content-based Access of Video and Image databases*, Florence, Italy, June 1999.

[44] Y. Muto, H. Nagase, and Y. Hamamoto, "Evaluation of a modified parzen classifier in high-dimensional spaces," in *Proc. 15th Int'l Conf. Pattern Recognition*, 2000, vol. 2, pp. 67–70.

[45] 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.

[46] J. Herault and C. Jutten, "Space or time adaptive signal provessing by neural network models," in *Proc. AIP Conf. Neural Networks Computing*, Snowbird, UT, USA, 1986, vol. 151, pp. 206–211.

[47] J. Cardoso, "Source separation using higher order moments," in *Proc. ICASSP*, 1989, pp. 2109–2112.

[48] P. Comon, "Independent component analysis, a new concept?," *Signal Processing*, vol. 36, pp. 287–314, 1994.

[49] D. Obradovic and G. Deco, "Blind source seperation: are information maximization and redundancy minimization different?," in *Proc. IEEE Workshop on Neural Networks for Signal Processing 1997*, Florida, Sept. 1997.

[50] J. Cardoso, "Infomax and maximum likelifood for blind source separation," *IEEE Signal Processing Letters*, vol. 4, no. 4, April 1997.

[51] T.-W. Lee, M. Girolami, A.J. Bell, and T.J. Sejnowski, "A unifying information -theretic framework for independent component analysis," *Computers and Mathematics with Applications*, vol. 31, no. 11, March 2000.

[52] T-W. Lee, M. Girolami, and T.J. Sejnowski, "Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources," *Neural Computation*, vol. 11, no. 2, Feb. 1999.

[53] L. Xu, C. Cheung, and S.-I. Amari, "Learned parametric mixture based ica algorithm," *Neurocomputing*, vol. 22, no. 1-3, pp. 69 – 80, 1998.

[54] 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.

[55] A.M. Fraser and H.L. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical Review A*, vol. 33, no. 2, 1986.

[56] Y. Hamamoto, S. Uchimura, and S. Tomita, "On the behavior of artificial neural network classifiers in high-dimensional spaces," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 18, no. 5, pp. 571–574, May 1996.

[57] 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.

[58] J.P. Siebert, "Vehicle recognition using rule based methods," Tech. Rep., Turing Institute, March 1987, Research Memorandum TIRM-87-018.

[59] P. M. Murphy and D. W. Aha, "Uci repository of machine learning databases," 1994, For more information contact ml-repository@ics.uci.edu or http://www.cs.toronto.edu/∼delve/.

[60] N. Kwak and C.-H. Choi, "Feature extraction based on ica for binary classification problems," *To appear in IEEE Trans. on Knowledge and Data Engineering*, 2003.

[61] N. Kwak, C.-H. Choi, and N. Ahuja, "Feature extraction for classification problems and its application to face recognition," *Submitted to IEEE Trans. Pattern Analysis and Machine Intelligence*, 2002.

[62] 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.

[63] J. Cardoso, "On the stability of source separation algorithms," *Journal of VLSI Signal Processing Systems*, vol. 26, no. 1, pp. 7 – 14, Aug. 2000.

[64] N. Vlassis and Y. Motomura, "Efficient source adaptivity in independent component analysis," *IEEE Trans. Neural Networks*, vol. 12, no. 3, May 2001.

[65] Quest Group at IBM Almaden Research Center, "Quest synthetic data generation code for classification," 1993, For information contact http://www.almaden.ibm.com/cs/quest/.

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

[67] W.H. Wolberg and O.L.Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. National Academy of Sciences*, vol. 87, Dec. 1990.

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

[69] 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.

[70] C. Liu and H. Wechsler, "Evolutionary pursuit and its application to face recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 570–582, June 2000.

[71] M.-H. Yang, "Face recognition using kernel methods," *Advances of Neural Information Processing Systems*, vol. 14, 2002.

[72] M.S. Bartlett and T.J. Sejnowski, "Viewpoint invariant face recognition using independent component analysis and attractor networks," *Neural Information Processing Systems-Natural and Synthetic*, vol. 9, pp. 817–823, 1997.

[73] G. Donato, M.S. Bartlett, J.C. Hager, P. Ekman, and T.J. Sejnowski, "Classifying facial actions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 974–989, Oct. 1999.

[74] 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.

[75] E. Micheli-Tzanakou, *Supervised and unsupervised pattern recognition*, CRC Press, 2000.

[76] 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.

[77] M. J. Lyons, J. Budynek, and S. Akamatsu, "Automatic classification of single facial images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1357 – 1362, Dec. 1999.

[78] T.M. Mitchell, *Machine learning*, McGraw-Hill, 1997.

Appendix A

Proof of Theorem 1

If (4.15) is to be a stationary point of learning rule (4.11), $\Delta W \triangleq W^{(t+1)} - W^{(t)}$ and $\Delta \boldsymbol{v} \triangleq \boldsymbol{v}^{(t+1)} - \boldsymbol{v}^{(t)}$ must be zero in the statistical sense. Thus

$$\begin{aligned} \mathrm{E}\{[I_N - \boldsymbol{\varphi}(\boldsymbol{U})\boldsymbol{F}^T]W\} &= 0 \\ \mathrm{E}\{\boldsymbol{\varphi}(\boldsymbol{U}_a)C\} &= 0 \end{aligned} \tag{A.1}$$

must be satisfied. The second equality is readily satisfied because of the independence of $\boldsymbol{U}_a$ and C and the zero mean assumption on C. The first equality holds if

$$\mathrm{E}\{I_N - \boldsymbol{\varphi}(\boldsymbol{U})\boldsymbol{F}^T\} = I_N - \mathrm{E}\{\boldsymbol{\varphi}(\boldsymbol{U})\boldsymbol{U}^T\} - E\{\boldsymbol{\varphi}(\boldsymbol{U})C\}\boldsymbol{v}^T = 0. \tag{A.2}$$

In the equation the last term $\mathrm{E}\{\boldsymbol{\varphi}(\boldsymbol{U})C\} = 0$ because $\boldsymbol{U}$ and C are independent and C is a zero mean random variable. Thus, the condition (A.2) holds if

$$\mathrm{E}\{\varphi_i(U_i)U_j\} = \delta_{ij}, \tag{A.3}$$

where δ_{ij} is a Kronecker delta. When $i \neq j$, this condition is satisfied because of the independence assumption on $U_i(= E_i)$'s, and the remaining condition is

$$\mathrm{E}\{\varphi_i(U_i)U_i\} = \mathrm{E}\{\varphi_i(\lambda_i S_{\Pi(i)})\lambda_i S_{\Pi(i)}\} = 1, \qquad \forall 1 \leq i \leq N. \tag{A.4}$$

Here the fact that $U_i = E_i = \lambda_i S_{\Pi(i)}$ is used, where λ_i is the ith diagonal element of scaling matrix Λ and $S_{\Pi(i)}$ is the ith signal permuted through Π.

Assuming that S_i has an even pdf, then U_i has an even pdf and $\varphi_i(= \dot{p}_i(u_i)/p_i(u_i))$ is an odd function. Therefore, λ_i that satisfies (A.4) always comes in pairs: if λ is a solution, so is $-\lambda$. Furthermore if we assume that φ_i is an increasing differentiable function, (A.4) has a unique solution λ_i^* up to a sign change.

Appendix A. Proof of Theorem 1

Appendix B

Proof of Theorem 2

For the proof, a standard tool for analyzing the local asymptotic stability of a stochastic algorithm is used. It makes use of the derivative of the mean field at a stationary point. In our problem, $Z \in \Re^{N\times N}$ and $\boldsymbol{k} \in \Re^{M}$ constitute an $N \times N + M$ dimensional space, and this space can be denoted as a direct sum of Z and $\boldsymbol{k}$; i.e., $Z \oplus \boldsymbol{k}$. Then the derivative considered here is that of a mapping $H : Z \oplus \boldsymbol{k} \rightarrow \mathrm{E}\{G(Z,\boldsymbol{k})Z\} \oplus \mathrm{E}\{g_1(Z,\boldsymbol{k})k_1\} \oplus \cdots \oplus \mathrm{E}\{g_M(Z,\boldsymbol{k})k_M\}$ at the stationary point $(Z^*, \boldsymbol{k}^*)$ where $Z^* = I_N$ and $\boldsymbol{k}^* = \mathbf{1}_M = [1, \cdots, 1]^T$. The derivative is of $(N \times N + M)^2$ dimension, and if it is positive definite, the stationary point is a local asymptotic stable point. As written in [63], because the derivative of the mapping H is very sparse, the first-order expansion of H at the point $(Z^*, \boldsymbol{k}^*)$ can be used rather than trying to use the exact derivatives.

For convenience, let us split H into two functions H^1 and H^2 such that

$$\begin{aligned} H^1 &: Z \oplus \boldsymbol{k} \rightarrow \mathrm{E}\{G(Z,\boldsymbol{k})Z\} \in \Re^{N\times N} \\ H_i^2 &: Z \oplus \boldsymbol{k} \rightarrow \mathrm{E}\{g_i(Z,\boldsymbol{k})k_i\}, \qquad 1 \le i \le M. \end{aligned} \tag{B.1}$$

Note that $H = H^1 \oplus H^2$. To get the first order linear approximation of the function at a stationary point $(Z^*, \boldsymbol{k}^*)$, H^1 and H^2 is evaluated near a small variation of the stationary point $(Z, \boldsymbol{k}) = (Z^* + \mathcal{E}, \boldsymbol{k}^* + \boldsymbol{\varepsilon})$, where $\mathcal{E} \in \Re^{N\times N}$ and

Appendix B. Proof of Theorem 2

$\boldsymbol{\varepsilon} \in \Re^M$.

$$
\begin{aligned}
&H^1_{ij}(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon}) \\
&\quad = [\mathrm{E}\{G(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\}(I_N + \mathcal{E})]_{ij} \\
&\quad = [\mathrm{E}\{G(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\}]_{ij} + [\mathrm{E}\{G(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\}\mathcal{E}]_{ij} \\
&\quad = \mathrm{E}\{G_{ij}\} + \sum_{n=1}^{N}\sum_{m=1}^{N} \mathrm{E}\{\frac{\partial G_{ij}}{\partial Z_{nm}}\}\mathcal{E}_{nm} + \sum_{m=1}^{M} \mathrm{E}\{\frac{\partial G_{ij}}{\partial k_m}\}\varepsilon_m + \sum_{m=1}^{N} \mathrm{E}\{G_{im}\}\mathcal{E}_{mj} \\
&\qquad + o(\mathcal{E}) + o(\boldsymbol{\varepsilon}).
\end{aligned}
\tag{B.2}
$$

and

$$
\begin{aligned}
&H^2_i(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon}) \\
&\quad = \mathrm{E}\{g_i(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\}(1 + \varepsilon_i) \\
&\quad = \mathrm{E}\{g_i(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\} + \mathrm{E}\{g_i(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon})\}\varepsilon_i \\
&\quad = \mathrm{E}\{g_i\} + \sum_{n=1}^{N}\sum_{m=1}^{N} \mathrm{E}\{\frac{\partial g_i}{\partial Z_{mn}}\}\mathcal{E}_{mn} + \sum_{m=1}^{M} \mathrm{E}\{\frac{\partial g_i}{\partial k_m}\}\varepsilon_m + \mathrm{E}\{g_i\}\varepsilon_i + o(\mathcal{E}) + o(\boldsymbol{\varepsilon}).
\end{aligned}
\tag{B.3}
$$

Using the independence and zero mean assumptions on e_i's and c, these can be further expanded as

$$
\begin{aligned}
&H^1_{ij}(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon}) \\
&= \begin{cases}
\mathcal{E}_{ij}\mathrm{E}\{\dot{\varphi}_i(E_i)E_j^2\} + \mathrm{E}\{\varphi_i(E_i)E_i\}\mathcal{E}_{ji} + v_j^* \sum_{m=1}^{M} \mathcal{E}_{im} v_m^* \mathrm{E}\{\dot{\varphi}_i(E_i)C^2\} \\
\qquad -\varepsilon_i v_i^* v_j^* \mathrm{E}\{\dot{\varphi}_i(E_i)C^2\} + o(\mathcal{E}) + o(\boldsymbol{\varepsilon}) & \text{if} \quad 1 \le i, j \le M \\
\mathcal{E}_{ij}\mathrm{E}\{\dot{\varphi}_i(E_i)E_j^2\} + \mathrm{E}\{\varphi_i(E_i)E_i\}\mathcal{E}_{ji} + v_j^* \sum_{m=1}^{M} \mathcal{E}_{im} v_m^* \mathrm{E}\{\dot{\varphi}_i(E_i)C^2\} \\
\qquad +o(\mathcal{E}) + o(\boldsymbol{\varepsilon}) & \text{if} \quad M < i \le N,\ 1 \le j \le M \\
\mathcal{E}_{ij}\mathrm{E}\{\dot{\varphi}_i(E_i)E_j^2\} + \mathrm{E}\{\varphi_i(E_i)E_i\}\mathcal{E}_{ji} + o(\mathcal{E}) + o(\boldsymbol{\varepsilon}) & \text{if} \quad M < i, j \le N
\end{cases}
\end{aligned}
\tag{B.4}
$$

and

$$
\begin{aligned}
&H^2_i(I_N + \mathcal{E}, \mathbf{1}_M + \boldsymbol{\varepsilon}) \\
&= -v_i^* \sum_{m=1}^{M} \mathcal{E}_{im} v_m^* \mathrm{E}\{\dot{\varphi}_i(E_i)C^2\} + \varepsilon_i v_i^{*2} \mathrm{E}\{\dot{\varphi}_i(E_i)C^2\} + o(\mathcal{E}) + o(\boldsymbol{\varepsilon}) \qquad 1 \le i \le M.
\end{aligned}
\tag{B.5}
$$

Now, the local stability conditions are developed case by case.

(Case 1) $i, j > M$

In this case, H_{ij}^1 and H_{ji}^1 only depend on $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ and are represented as

$$\begin{bmatrix} H_{ij}^1 \\ H_{ji}^1 \end{bmatrix} = \begin{bmatrix} \mathrm{E}\{\dot{\varphi}_i(E_i)\}\mathrm{E}\{E_j^2\} & \mathrm{E}\{\varphi_i(E_i)E_i\} \\ \mathrm{E}\{\varphi_i(E_j)E_j\} & \mathrm{E}\{\dot{\varphi}_j(E_j)\}\mathrm{E}\{E_i^2\} \end{bmatrix} \begin{bmatrix} \mathcal{E}_{ij} \\ \mathcal{E}_{ji} \end{bmatrix} \triangleq D_{ij} \begin{bmatrix} \mathcal{E}_{ij} \\ \mathcal{E}_{ji} \end{bmatrix} \quad \text{if} \quad i \neq j$$
$$H_{ii}^1 = [\mathrm{E}\{\dot{\varphi}_i(E_i)E_i^2\} + \mathrm{E}\{\varphi_i(E_i)E_i\}]\mathcal{E}_{ii} \triangleq d_i \mathcal{E}_{ii}. \tag{B.6}$$

Thus for $i \neq j$, Z_{ij} and Z_{ji} are stabilized when D_{ij} is positive definite. And if $i = j$, Z_{ii} is stabilized when d_i is positive. Using the fact that $\mathrm{E}\{\varphi_i(E_i)E_i\} = 1$ $\forall i = 1, \cdots, N$, it can be shown that the local stability condition for the pair (i, j) when $i, j > M$ is (4.28).

(Case 2) $i \leq M, j > M$

In this case, H_{ij}^1 and H_{ji}^1 are dependent not only on $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ but also on all $\mathcal{E}_{jm}$, $m = 1, \cdots, M$. Thus for a fixed j, all the H_{ij}^1 and H_{ji}^1, $i = 1, \cdots, M$ are augmented, and a $2M$-dimensional vector $\boldsymbol{H}_j \triangleq [H_{1j}^1, \cdots, H_{Mj}^1, H_{j1}^1, \cdots, H_{jM}^1]^T$ is constructed. Now this augmented vector $\boldsymbol{H}_j$ depends only on $\boldsymbol{\mathcal{E}}_j \triangleq [\mathcal{E}_{1j}^1, \cdots, \mathcal{E}_{Mj}, \mathcal{E}_{j1}, \cdots, \mathcal{E}_{jM}]^T$ and can be represented as a linear equation $\boldsymbol{H}_j = \boldsymbol{D}_j \boldsymbol{\mathcal{E}}_j$, using an appropriate matrix $\boldsymbol{D}_j \in \Re^{2M \times 2M}$. The stability of $\boldsymbol{Z}_j = [Z_{1j}, \cdots, Z_{Mj}, Z_{j1}, \cdots, Z_{jM}]^T$ for $j > M$ is equivalent to the positive definiteness of $\boldsymbol{D}_j$ and it can be checked by investigating the sign of the $\boldsymbol{H}_j^T \boldsymbol{\mathcal{E}}_j$.

Substituting (B.4) and using $\mathrm{E}\{\varphi_i(E_i)E_i\} = 1$ $\forall i = 1, \cdots, N$, it leads

$$\begin{aligned} \boldsymbol{H}_j^T \boldsymbol{\mathcal{E}}_j &= \sum_{i=1}^{M} (H_{ij}^1 \mathcal{E}_{ij} + H_{ji}^1 \mathcal{E}_{ji}) \\ &= \sum_{i=1}^{M} [\mathrm{E}\{\dot{\varphi}_i(E_i)E_j^2\}\mathcal{E}_{ij}^2 + 2\mathcal{E}_{ij}\mathcal{E}_{ji} + \mathrm{E}\{\dot{\varphi}_j(E_j)E_i^2\}\mathcal{E}_{ji}^2] \\ &\quad + \mathrm{E}\{\dot{\varphi}_j(E_j)\}\mathrm{E}\{C^2\}(\sum_{i=1}^{M} \mathcal{E}_{ji} v_i^*)^2. \end{aligned} \tag{B.7}$$

If it is assumed that $\dot{\varphi}_j(\cdot)$ is nonnegative, as in the proof of the uniqueness of the scalar λ_j, the last term becomes nonnegative. Thus, a sufficient condition for this equation to be positive is to make the first term positive, and this condition

is satisfied if and only if equation (4.28) holds. Therefore, (4.28) becomes a sufficient condition for the local stability of $\boldsymbol{Z}_j$.

(Case 3) $i, j \leq M$

In this case, because H_{ij}^1 and H_i^2 are dependent both on $\mathcal{E}$ and $\boldsymbol{\varepsilon}$, a new vector is constructed and the stability condition of the vector is investigated as in the previous case.

Consider the $M \times M + M$ dimensional vectors $\boldsymbol{H} \triangleq [H_{11}^1, H_{12}^1, \cdots, H_{MM}^1, H_1^2, \cdots, H_M^2]^T$ and $\boldsymbol{\mathcal{E}} \triangleq [\mathcal{E}_{11}, \mathcal{E}_{12}, \cdots, \mathcal{E}_{MM}, \varepsilon_1, \cdots, \varepsilon_M]^T$. Using (B.4) and (B.5), $\boldsymbol{H}$ can be represented as the linear equation $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{\mathcal{E}}$, where $\boldsymbol{D}$ is an appropriate matrix. Thus, the stability of the $\mathcal{Z} = [Z_{11}, Z_{12} \cdots, Z_{MM}]^T$ and $\boldsymbol{k}$ can be checked using the same procedure as the previous case.

$$\begin{aligned}
\boldsymbol{H}^T\boldsymbol{\mathcal{E}} &= \sum_{i=1}^{M}\sum_{j=1}^{M} H_{ij}^1 \mathcal{E}_{ij} + \sum_{i=1}^{M} H_i^2 \varepsilon_i \\
&= \sum_{i=1}^{M}\sum_{j=1}^{M} (\mathcal{E}_{ij}^2 \mathrm{E}\{\dot{\varphi}_i(E_i) E_j^2\} + \mathcal{E}_{ij}\mathcal{E}_{ji}) \\
&\quad + \sum_{i=1}^{M} [\mathrm{E}\{\dot{\varphi}_i(E_i)\}\mathrm{E}\{C^2\}(v_i^* \varepsilon_i - \sum_{j=1}^{M} v_j^* \mathcal{E}_{ij})^2]
\end{aligned} \tag{B.8}$$

The last term is nonnegative with the assumption of $\dot{\varphi}_i(\cdot) \geq 0$, and a sufficient condition for the double summation to be positive is (4.28). Thus, $\mathcal{Z} \oplus \boldsymbol{k}$ is locally stable if condition (4.28) holds.

Combining the stability conditions for the case 1, 2, and 3, it is concluded that the learning rule (4.11) for ICA-FX is locally asymptotically stable at the stationary point (4.15) if condition (4.28) holds.

초 록

인터넷과 새로운 컴퓨터 기술의 발달로 거대한 데이터베이스를 구축하는 것이 가능해졌다. 이러한 데이터베이스에는 일반적으로 attribute혹은 field라고 불리우기도 하는 많은 입력 특징(feature)들이 존재하며 이들 특징들 중에는 풀려고 하는 문제에 적합한 특징이 있는가 하면 그렇지 않은 특징도 있고, 서로 중복적인 특징들도 존재한다. 데이터베이스를 관리하고 분석하는 입장에서 보면 원래의 입력 특징들로부터 문제를 풀기에 적합한 특징만을 선택하거나 문제에 적합한 새로운 특징을 추출해서 입력 특징의 수를 줄이는 것이 바람직하다. 문제에 적합한 특징만을 이용함으로써 입력 특징의 수가 크게 줄 수 있으며, 이는 절약의 원리(principle of parsimony)와도 일맥상통하여 보다 낳은 일반화 성능을 가져올 수 있다.

이 논문에서는 분류 문제를 위한 입력 특징 선택과 추출 문제를 다루고 있다. 논문 전반에 걸쳐 출력 클래스와 입력 특징들 사이의 밀접도를 재는 척도로 공유 정보(mutual information)를 사용하였다. 먼저 논문의 전반부에서는 입력 특징 선택 문제를 연구하였고 새로운 특징 선택 방법을 제안하였다. 입력 특징과 출력 클래스간의 공유 정보를 계산함에 있어서 Parzen window에 기반한 새로운 방법을 제시하였고 이를 분류 문제를 위한 greedy 선택 알고리즘에 적용하였다. 논문의 후반부에서는 입력 특징 추출 문제를 다루었고 새로운 입력 특징 추출 알고리즘을 제안하였다. 제안된 알고리즘은 기존의 독립 성분 분석(independent component analysis; ICA) 알고리즘을 분류 문제에 적합하도록 변형하여 원래 특징들이 가지고 있던 출력 클래스에 관한 정보량을 그대로 갖고 있는 적은 수의 새로운 특징들을 추출하는 방법이다. 논문에서는 제안된 알고리즘의 국부 안정도(local stability) 조건도 함께 제시하였다. 제안된 방법의 장점은 클래스와 새로운 입력 특징들 사이의 공유 정보를 최대화함으로써 일반적인 ICA 알고리즘을 분류 문제를 위한 특징 추출 방법으로 이용할 수 있다는 점이다. 새로운 특징을 사용함으로써 분류 성능을 떨어뜨리지 않으면서 입력 특징 공간의 차원을 많이 줄일 수 있다.

제안된 특징 선택 및 추출 방법들을 얼굴 인식과 같은 여러 가지 패턴 인식과 분류 문제에 적용해 제안된 방법들의 성능을 기존 방법들의 성능과 비교해 보았다. 실험 결과를 통해 제안된 방법들이 적은 수의 특징을 이용하면서 기존 방법들의 분류율보다 높은 성능을 보여 우수함을 확인할 수 있었다.

제시어: 특징 선택, 특징 추출, 입력차원 감소(dimensionality reduction), Parzen window, 공유 정보, 독립 성분 분석, 얼굴 인식, 분류, 데이터 마이닝, 패턴 인식

감사의 글

관악에서의 처음을 맞이한지 딱 십년이 흘렀습니다. 잘해봐야 고등학생으로 밖에 안 보이던 앳된 청년의 모습을 벗고 어느덧 매일 면도를 하지 않으면 수염이 거뭇거뭇해지는 나이가 되었습니다. 처음 아버지와 입학 원서를 접수하러 온 걸 시작으로 석사 박사과정을 마치기까지 그 동안 거의 매일 오르내리던 관악인데 돌아보니 구석구석 낯선 곳이 참 많습니다. 막상 감사의 글을 쓰려고 생각해보니 그 동안 많은 사람들을 만났지만 떠올리면 이름이 생각나지 않는 수많은 사람들이 남았습니다. 건방지다 말할지 모르겠지만 인생이란 이런 것인가 봅니다. 모두에게 기억 될 수는 없겠지만 언제나 좋은 사람으로 기억되고 싶습니다. 이제 학교라는 울타리를 벗어나 그 동안의 삶과는 완전히 다른 또 다른 세상이 저를 기다리고 있습니다. 또 하나의 껍질을 깨어가는 지금, 조금은 두렵지만 그만큼 기대되는 제 인생의 앞날을 모든 분들이 축복해 주십시오.

이렇게 한 편의 논문이 나오기까지 그 동안 많은 분들의 도움이 있었습니다. 가장 먼저 제 육년 동안의 대학원 생활에서 지도교수로서 언제나 학자로서의 품위를 잃지 않으시고 한결같이 연구하는 자세로 제자들을 지도해주신 존경하는 최종호 교수님께 큰 감사를 드립니다. 다음으로 대학원 입학 당시부터 프로젝트를 함께 하면서 저에게 많은 도움을 주신 최진영 교수님과, 바쁘신 중에도 시간을 내서 제 논문을 심사해 주시고 많은 조언을 주신 장병탁 교수님과 조남익 교수님께도 진심으로 감사드립니다. 논문 심사 때문에 수업도 빼먹고 달려와 주신 이창수 선배님도 정말 고맙습니다. 그 외에도 저의 대학과 대학원 생활을 이끌어 주신 전기공학부의 여러 교수님들께도 감사를 드립니다.

제어 및 시스템 연구실의 여러 선배님, 동기, 후배들의 가르침과 조언은 제가 대학원 생활을 무사히 마칠 수 있게 해 주었고 앞으로의 삶에 있어서도 커다란 재산이 될 것입니다. 이 지면을 빌어 황진권, 김상우, 전윤호, 김명찬, 김성원, 최병갑, 고봉준, 김도형, 성완, 이정우 선배님들께 감사드립니다. 동문 선배로서 제가 이 연구실에 들어오는데 결정적인 도움을 주신 구민이형과 실험실의 대소사를 빠짐없이 챙기시고 언제나 후배들에게 충고어린 한마디를 던져주시는 오석이형께도 감사드립니다. 언제나 열심히 연구하시는 혁이형, 즐거운 대학원 생활로 인도해주신 동환이형도 정말 고맙습니다. 대학원 시작을 같이한 실험실 동기, 태환, 태웅, 진우에게도 감사의 마음을 전합니다. 일년 동안 미국에서 같이 살면서 더 친해진 경준이를 비롯해 앞으로 공학관을 이끌어갈 은찬이, 태신이, 주형이, 찬수, 병우에게도 감사드립니다. 매주 테니스를 같이 치던 김경섭 박사

님과 정훈, 주보에게도 고마움과 따뜻한 인사의 말을 드립니다. 그 외에 자동화 연구소의 동영, 윤수, 선호, 채권, 영집 등에게도 고마움을 전합니다. 프로젝트를 같이 수행하며 매주 만났던 최진영 교수님 연구실의 문혁이형, 명수 등 모두에게도 감사드립니다.

대학 입학 때부터 관악 생활을 함께 하면서 서로 힘이 되어주고 위로가 되었던 전기공학부 선후배님께 진심으로 감사합니다. 특히 새터에서 한 방이 된 인연에서 시작하여 바보정경이라는 모임으로 만난 석기형, 영균, 철신, 재훈, 은정과, 동수형, 재호형, 결혼해서 잘 살고 있는 정태, 재용, 결혼이 얼마 안 남은 재인에게도 감사를 드립니다. 포장마차서 술잔을 기울이던 지훈, 동규, 민호, 진태, 일년 동안의 미국 생활에서 큰 힘이 되어 준 윤기에게도 감사의 마음을 전합니다. 그 외 영준형, 영훈, 형진이를 비롯한 93A반 동기들, 일리노이 대학에서 함께 외로움을 달랠 수 있었던 태림, 성욱, 보원, 황남형, 원종형과 모든 선배 후배님들, 초중고등학교 친구인 영신, 은지, 석주, 재국, 동현 등에게도 언제나 고맙다는 말을 하고 싶었습니다.

기쁠 때나 즐거울 때 짜증나거나 힘든 일이 있을 때 언제나 가족이라는 울타리로 저를 감싸주고 보듬어 주며 성원해 주는 누나와 진순이에게 정말 고맙다는 말을 하고 싶고 마지막으로 지금의 제가 있기까지 모든 고생을 아끼지 않으신 아버지, 어머니께 가장 큰 존경과 감사의 마음을 전합니다.