Dimensionality Reduction Based on ICA for Regression Problems

Nojun Kwak [a,*,1] Chunghoon Kim [b,2] Hwangnam Kim [c,3]

[a] *Division of Electrical & Computer Engineering, Ajou University, San 5, Woncheon-dong, Yeongtong-gu, Suwon, 443-749 KOREA.*

[b] *Department of Computer Science, University of California, Santa Barbara, CA 93106 USA.*

[c] *School of Electrical Engineering, Korea University, Anam-dong, Seongbuk-Gu, Seoul, 136-701 KOREA.*

Abstract

In manipulating data such as in supervised learning, we often extract new features from the original input variables for the purpose of reducing the dimensions of input space and achieving better performances. In this paper, we show how standard algorithms for independent component analysis (ICA) can be extended to extract attributes for regression problems. The advantage is that general ICA algorithms become available to a task of dimensionality reduction for regression problems by maximizing the joint mutual information between target variable and new attributes. We applied the proposed method to a couple of real world regression problems as well as some artificial problems and compared the performances with those of other conventional methods. Experimental results show that the proposed method can efficiently reduce the dimension of input space without degrading the regression performance.

Key words: Feature extraction, dimensionality reduction, regression, ICA.

1 Introduction

In supervised learning, one is given an array of input variables to predict the target value and there may exist irrelevant or redundant input variables to complicate the learning process, thus leading to incorrect prediction [1] [2]. Even when the input variables presented contain enough information about the target variables, they may not predict the target correctly because the dimension of the input space may be so large that it may require numerous instances to determine the relationship between inputs and target. This problem is known as the *curse of dimensionality* [1] which can be mitigated by selecting only the relevant inputs or extracting new variables containing maximal information about the target variable from the original inputs. The former methodology is called feature selection or subset selection, while the latter is named feature extraction which includes all the methods that utilize functional mapping to generate new features from the original inputs [4] [3] [4].

* Corresponding author is Nojun Kwak.
Email addresses: nojunk@ieee.org (Nojun Kwak), spinoz@csl.snu.ac.kr (Chunghoon Kim), hnkim@korea.ac.kr (Hwangnam Kim).

[1] Nojun Kwak is an assistant professor in the Division of Electrical & Computer Engineering at Ajou University, Korea.

[2] Chunghoon Kim is a postdoctoral researcher in the Department of Computer Science at the University of California, Santa Barbara.

[3] Hwangnam Kim is an assistant professor in the School of Electrical Engineering at Korea University, Korea.

[4] The terms *input* and *feature* are usually used interchangeably in pattern recognition society. However, in this paper, to avoid complication, the term *input* is used

This paper deals with the feature extraction problems since it often results in improved performance by extracting new features from the original inputs, especially when small input dimension is required. Among the various feature extraction approaches, we focus on finding an appropriate subspace spanned by a set of linear transformations of the original input variables. This subspace method includes popular algorithms such as PCA (principle component analysis) [5], ICA (independent component analysis) [6] [7] [8], and LDA (linear discriminant analysis) [9]. However, most of these methods cannot be used for regression problems since some of them such as PCA and ICA focus on finding features by unsupervised manner and others such as LDA have been mainly developed for classification problems.

Instead of using only up to the second order statistics as in LDA and its variants, methods utilizing higher order statistics such as mutual information have been proposed [10] – [16] . Mutual information is a good metric to measure the closeness between random variables [17], thus fits to the dimensionality reduction tasks. Many algorithms based on mutual information have been proposed for feature selection problems [18] [19] [20], however, researchers have only recently begun to use it for feature extraction problems due to the computational burden [10] [11] [13] [16]. Ullman et al. applied this to binary classification problems with binary features [11]. Fisher et al. [12] [13] [14] used the quadratic mutual information, which is related to Renyi's entropy, in feature extraction for classification problems. On the other hand, Kwak [16] tried to directly maximize the Shannon mutual information between the discrete class variable and the newly extracted features. However, none of these

for the original input variable while the term *feature* is used for the transformed variable.

methods can easily be extended for regression problems because the mutual information between continuous variables is still difficult to compute.

Recently, a feature extraction method ICA-FX which also utilizes mutual information was proposed [15]. Although ICA-FX is an extension of ICA whose structure and the learning process are quite similar to those of standard ICA, unlike ICA, it was developed for classification problems in that it includes the output class information to find appropriate features. The algorithm attempts to maximize the mutual information between the features and output classes indirectly under the assumption that the sources are independent of the classes. By maximizing the mutual information between the class and the features, it can construct new features that contain much information about the output class.

As was overviewed, although many feature extraction methods have been developed for classification problems, relatively little attention has been taken on the feature extraction methods for regression problems in the pattern recognition society.

On the other hand, in statistics, several algorithms have been developed as dimensionality reduction techniques for regression problems among which classical multivariate linear regression (MLR) [21] can be the basic. Although MLR is the optimal in the sense of least squared error, it has a limitation that it can only produce one feature. To overcome this limitation, a local linear dimensionality reduction method based on the nearest neighbor scheme have been proposed [22]. Sliced inverse regression (SIR) [23] and principal Hessian directions (PHD) [24] are also very popular dimensionality reduction techniques for regression problems in statistics.

In this paper, the ICA-FX for classification problems is extended to regression problems. Because the output class label is coded as a numerical value in the original ICA-FX algorithm, the method can be easily extended to regression problems without changing much from the original ICA-FX which was developed for classification problems. We apply the ICA-FX to several regression problems and the performances of ICA-FX are compared with those of other methods.

This paper is organized as follows. In Section 2, we briefly review the ICA algorithm. In Section 3, we develop ICA-FX for regression problems. This follows almost the same steps as we did for classification problems. Experimental results showing the advantages of the proposed algorithm are presented in Section 4 and conclusions are provided in Section 5.

2 Review of ICA

The problem setting of ICA is as follows. Assume that there is an L-dimensional zero-mean non-Gaussian source vector $\boldsymbol{s} = [s_1, \cdots, s_L]^T$, such that the components s_i's are mutually independent, and an observed data vector $\boldsymbol{x} = [x_1, \cdots, x_N]^T$ is composed of linear combinations of sources s_i, such that

$$\boldsymbol{x} = A\boldsymbol{s} \tag{1}$$

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

$$\boldsymbol{u} = W\boldsymbol{x} = WA\boldsymbol{s} \tag{2}$$

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

To find W in (2), Bell and Sejnowski [7] have developed the Infomax algorithm, one of the popular algorithms for ICA, in which they used a feed-forward neural processor. This neural processor takes $\boldsymbol{x}$ as an input vector. The weight W is multiplied by 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.

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 $y_i = g_i(u_i), i = 1 \cdots L$, since the nonlinear transfer function $g_i(\cdot)$ does not introduce any dependencies.

In [7], 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, we can approximately minimize the mutual information among the output components y_i's

$$I(\boldsymbol{y}) = \int p(\boldsymbol{y}) \log \frac{p(\boldsymbol{y})}{\prod_{i=1}^{N} p_i(y_i)} \, d\boldsymbol{y}. \tag{3}$$

Here, $p(\boldsymbol{y})$ is the joint probability density function (pdf) of a vector $\boldsymbol{y}$, and $p_i(y_i)$ is the marginal pdf of the variable y_i.

The joint entropy of the outputs of this processor is

$$H(\boldsymbol{y}) = -\int p(\boldsymbol{y}) \log p(\boldsymbol{y}) d\boldsymbol{y} = -\int p(\boldsymbol{x}) \frac{p(\boldsymbol{x})}{\log |\det J(\boldsymbol{x})|} d\boldsymbol{x} \tag{4}$$

where $J(\boldsymbol{x})$ is a 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 and applying natural gradient by multiplying $W^T W$ on the right, we get the learning rule for ICA:

$$\Delta W \propto [I - \boldsymbol{\varphi}(\boldsymbol{u}) u^T] W \tag{5}$$

where

$$\boldsymbol{\varphi}(\boldsymbol{u}) = \left[-\frac{\frac{\partial p_1(u_1)}{\partial u_1}}{p_1(u_1)}, \cdots, -\frac{\frac{\partial p_N(u_N)}{\partial u_N}}{p_N(u_N)} \right]^T . \tag{6}$$

In this paper, we adopt the extended Infomax approach [8] because it is easy to implement with less strict assumptions on source distribution.

3 ICA-FX for Regression

ICA outputs a set of maximally independent vectors that are linear combinations of the observed data. Although these vectors might have some applications in such areas as blind source separation and data visualization, it is not suitable for feature extraction for supervised learning, because it does not make use of the output target information. The effort to incorporate the standard ICA with supervised learning has been made in our previous work [15], where a new feature extraction algorithm, ICA-FX for classification problems was proposed. In this paper, the original ICA-FX algorithm for classification problems is extended to regression problems.

Before we present our algorithm, we formalize the purpose of feature extraction

for regression problems.

The success of a feature extraction algorithm depends critically on how much information about the target variables is contained in the newly generated features. This can be formalized as follows.

(Problem statement) *Assume that there are a normalized input vector,* $\boldsymbol{x} = [x_1, \cdots, x_N]^T$*, and target variables,* $\boldsymbol{t} = [t_1, \cdots, t_{N_t}]^T$*. The purpose of feature extraction for regression problems is to extract* $M(\leq N)$ *new features* $\boldsymbol{f_a} = [f_1, \cdots, f_M]^T$ *from* $\boldsymbol{x}$*, containing the maximum information on the target variables* $\boldsymbol{t}$*. Here* N_t *is the number of target variables.*

In information theory, the information between random variables is measured by *mutual information* and the above problem statement can be formalized using this information theoretical term as follows:

(Problem statement - information theoretic view) *The purpose of a feature extraction for regression problem is to extract* $M(\leq N)$ *features* $\boldsymbol{f}_a$ *from the original input vector* $\boldsymbol{x}$*, such that* $I(\boldsymbol{f}_a; \boldsymbol{t})$*, the mutual information between newly extracted features* $\boldsymbol{f}_a$ *and target variables* $\boldsymbol{t}$ *becomes maximum.*

The main idea of the proposed method is simple. It tries to apply the standard ICA algorithms to feature extraction for regression problems by making use

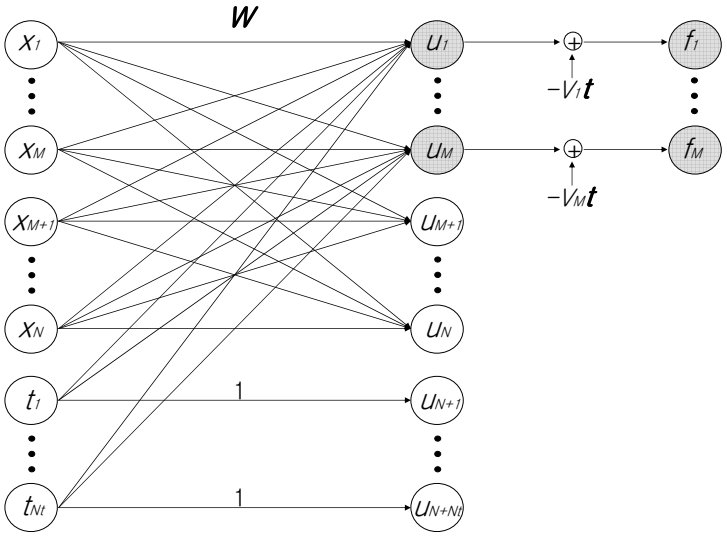

Fig. 1. Feature extraction algorithm based on ICA (ICA-FX)

of the target variables to produce two sets of new features; features that carry as much information on the target variables (these features will be useful for regression) as possible and the others that do not (these will be discarded). The advantage is that the general ICA algorithms can be used for feature extraction by maximizing the joint mutual information between the target variables and new features.

Now consider the structure shown in Fig. 1. Here, the original input vector $\boldsymbol{x}$ is fully connected to $\boldsymbol{u} = [u_1, \cdots, u_N]$, the target vector $\boldsymbol{t}$ is connected only to $\boldsymbol{u}_a = [u_1, \cdots, u_M]$, and $u_{N+l} = t_l$ for $l = 1, \cdots, N_t$. In the figure, the weight

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

$$\boldsymbol{W} = \left(\begin{array}{c|c} W & V \\ \hline \mathbf{0}_{N_t,N} & I_{N_t} \end{array}\right) = \left(\begin{array}{c|ccc} & w_{1,N+1} & \cdots & w_{1,N+N_t} \\ & \vdots & & \vdots \\ W & w_{M,N+1} & \cdots & w_{M,N+N_t} \\ & & & \\ & \multicolumn{3}{c}{\mathbf{0}_{N-M,N_t}} \\ \hline \mathbf{0}_{N_t,N} & \multicolumn{3}{c}{I_{N_t}} \end{array}\right). \tag{7}$$

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

As stated before, in information theoretic view, the aim of feature extraction is to extract M new features $\boldsymbol{f}_a$ from the original N input variables, $\boldsymbol{x}$, such that $I(\boldsymbol{f}_a;\boldsymbol{t})$, the mutual information between newly extracted features $\boldsymbol{f}_a$ and the target variables $\boldsymbol{t}$ is maximized.

Because of the data processing inequality it is maximized when $I(\boldsymbol{f}_a;\boldsymbol{t})$ becomes equal to $I(\boldsymbol{x};\boldsymbol{t})$, the mutual information between the original input variables $\boldsymbol{x}$ and the target variables $\boldsymbol{t}$.

This can be satisfied if we can separate the input space spanned by $\boldsymbol{x}$ into two linear subspaces: one that is spanned by $\boldsymbol{f}_a = [f_1, \cdots, f_M]^T$, which contains the maximum information on the target variables $\boldsymbol{t}$, and the other spanned by $\boldsymbol{f}_b = [f_{M+1}, \cdots, f_N]^T$, which is independent of $\boldsymbol{t}$ as much as possible.

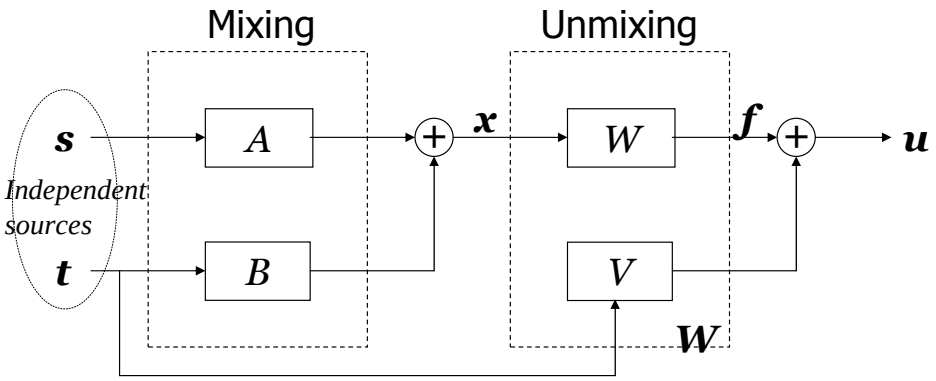

Fig. 2. Interpretation of Feature Extraction in the BSS structure

The condition for this separation can be derived as follows. If it is assumed that $\boldsymbol{W}$ is nonsingular, then $\boldsymbol{x}$ and $\boldsymbol{f} = [f_1, \cdots, f_N]^T$ span the same linear space. Because $\boldsymbol{f}$ can be represented with $\boldsymbol{f}_a \oplus \boldsymbol{f}_b = [\boldsymbol{f}_a^T, \boldsymbol{f}_b^T]^T$, the direct sum of $\boldsymbol{f}_a$ and $\boldsymbol{f}_b$, it becomes

$$I(\boldsymbol{x};\boldsymbol{t}) = I(W\boldsymbol{x};\boldsymbol{t}) = I(\boldsymbol{f};\boldsymbol{t}) = I(\boldsymbol{f}_a \oplus \boldsymbol{f}_b;\boldsymbol{t}) \geq I(\boldsymbol{f}_a;\boldsymbol{t}). \tag{8}$$

The first equality holds because W is nonsingular. The second and the third equalities are from the definitions of $\boldsymbol{f}$, $\boldsymbol{f}_a$ and $\boldsymbol{f}_b$. The last inequality is the data processing inequality [17] which holds if $I(\boldsymbol{f}_b;\boldsymbol{t}) = I([u_{M+1}, \cdots, u_N]^T;\boldsymbol{t}) = 0$.

If this is possible, the dimension of the input space can be reduced from N to $M(< N)$ by using only $\boldsymbol{f}_a$ instead of $\boldsymbol{x}$, without losing any information on the target variables.

To meet the equality condition $I(\boldsymbol{f}_b;\boldsymbol{t}) = 0$ of (8), the feature extraction problem is interpreted in the structure of the blind source separation (BSS) problem as shown in Fig. 2. The detailed description of each step is as follows:

(Mixing) *Assume that there are N independent sources $\boldsymbol{s} = [s_1, \cdots, s_N]^T$ which are also independent of the target variables $\boldsymbol{t}$. Assume also that the*

observed input vector $\boldsymbol{x}$ *is a linear combination of the sources* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *with the mixing matrices* $A \in \Re^{N \times N}$ *and* $B \in \Re^{N \times N_t}$*, i.e.,*

$$\boldsymbol{x} = A\boldsymbol{s} + B\boldsymbol{t}. \tag{9}$$

(Unmixing) *As shown in Fig. 1, the unmixing stage is slightly different from the BSS problem. In the figure, the unmixing equation becomes*

$$\boldsymbol{u} = W\boldsymbol{x} + V\boldsymbol{t}. \tag{10}$$

Suppose $\boldsymbol{u}$ *is somehow made equal to* $\boldsymbol{e}$*, the scaled and permuted version of the source* $\boldsymbol{s}$*, i.e.,*

$$\boldsymbol{e} \triangleq \Lambda\Pi\boldsymbol{s} \tag{11}$$

where Λ *is a diagonal matrix corresponding to an appropriate scale and* Π *is a permutation matrix. The* u_i*'s* ($i = 1, \cdots, N$) *are then independent of the target variables* $\boldsymbol{t}$ *by the assumption. Among the elements of* $\boldsymbol{f} = W\boldsymbol{x} (= \boldsymbol{u} - V\boldsymbol{t})$*,* $\boldsymbol{f}_b = [f_{M+1}, \cdots, f_N]^T$ *will be independent of* $\boldsymbol{t}$ *because the* i*th row of* V *is zero, i.e.,* $V_i = [w_{i,N+1}, \cdots, w_{i,N+N_t}] = \boldsymbol{0}$*, and* $f_i = u_i$ *for* $i = M+1, \cdots, N$*. Therefore, the* $M (< N)$ *dimensional new feature vector* $\boldsymbol{f}_a$ *that contains the most information on the target can be extracted by a linear transformation of the input vector* $\boldsymbol{x}$ *if the relation* $\boldsymbol{u} = \boldsymbol{e}$ *holds.*

The learning rule for the ICA-FX for regression is obtained by the same way as that of ICA-FX for classification problem using the MLE (maximum likelihood estimation) approach as follows.

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 equal to $\boldsymbol{e}$, a scaled and permutated version of the source, $\boldsymbol{s}$,

as in (11), and that each element of $\boldsymbol{u}$ is independent of the other elements of $\boldsymbol{u}$, which is also independent of the target vector $\boldsymbol{t}$, the log likelihood of $\boldsymbol{W}$ for a given data $D = \{(\boldsymbol{x}^l, \boldsymbol{t}^l)\}_{l=1}^n$ becomes the following:

$$L(\boldsymbol{W}|D) = n \log |\det \boldsymbol{W}| + \sum_{l=1}^{n} \sum_{i=1}^{N} \log p_i(u_i^l) + \sum_{l=1}^{n} \log p(\boldsymbol{t}^l). \tag{12}$$

Here, l denotes an index of the observed data and n is the number of samples. The above equation is because

$$L(\boldsymbol{W}|D) = p(D|\boldsymbol{W}) = \prod_{l=1}^{n} p(\boldsymbol{x}^l, \boldsymbol{t}^l|\boldsymbol{W})$$

and

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

Now, by the steepest ascent method L can be maximized. Because the last term in (12) is a constant, differentiating (12) with respect to $\boldsymbol{W}$ leads to

$$\begin{aligned} \frac{\partial L}{\partial w_{i,j}} &= n \frac{adj(w_{j,i})}{|\det \boldsymbol{W}|} - \sum_{l=1}^{n} \varphi_i(u_i^l) x_j^l \qquad 1 \le i, j \le N \\ \frac{\partial L}{\partial w_{i,N+j}} &= -\sum_{l=1}^{n} \varphi_i(u_i^l) t_j^l \qquad 1 \le i \le M, 1 \le j \le N_t \end{aligned} \tag{13}$$

where $adj(\cdot)$ is adjoint and $\varphi_i(\alpha) = -\frac{dp_i(\alpha)}{d\alpha} / p_i(\alpha)$.

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

$$\begin{aligned} \Delta W &\propto \; W^{-T} - \frac{1}{n} \sum_{l=1}^{n} \boldsymbol{\varphi}(\boldsymbol{u}^l) \boldsymbol{x}^{lT} \\ \Delta V_a &\propto \; -\frac{1}{n} \sum_{l=1}^{n} \boldsymbol{\varphi}(\boldsymbol{u}_a^l) \boldsymbol{t}^{lT}. \end{aligned} \tag{14}$$

Here $\boldsymbol{\varphi}(\boldsymbol{u}^l) \triangleq [\varphi_1(u_1^l), \cdots, \varphi_N(u_N^l)]^T$ and $\boldsymbol{\varphi}(\boldsymbol{u}_a^l) \triangleq [\varphi_1(u_1^l), \cdots, \varphi_M(u_M^l)]^T$.

Applying a natural gradient on updating W, by multiplying $W^T W$ on the

right side of the first equation of (14), the following is obtained.

$$\begin{aligned} W^{(n+1)} =& W^{(n)} + \mu_1[I_N - \frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\varphi}(\boldsymbol{u}^l)\boldsymbol{f}^{lT}]W^{(n)} \\ V_a^{(n+1)} =& V_a^{(n)} - \mu_2\frac{1}{n}\sum_{l=1}^{n}\boldsymbol{\varphi}(\boldsymbol{u}_a^l)\boldsymbol{t}^{lT}. \end{aligned} \tag{15}$$

Here μ_1 and μ_2 are the learning rates that can be set differently. By this weight update rule, the resulting u_i's will have a good chance of fulfilling the assumption that u_i's are not only independent of one another but also independent of the target variables $\boldsymbol{t}$. Note that the performance of this method depends on how the data is distributed to satisfy the assumption. Because the learning rule involves matrix multiplications, the computational complexity of the algorithm is $\mathcal{O}(nN^3)$.

Note that the learning rule for W is the same as the original ICA learning rule in [7], and also note that $\boldsymbol{f}_a$ corresponds to the first M elements of $W\boldsymbol{x}$. Therefore, the optimal features $\boldsymbol{f}_a$ can be extracted by the proposed algorithm when it finds the optimal solution for W by (15).

4 Experiment Results

In this section, we have applied ICA-FX to several regression problems and compared the performances of ICA-FX with those of other conventional methods such as PCA, MLR, SIR and PHD.

For all the problems, various regression methods such as multi-layer perceptron (MLP) [25], support vector machine (SVM) [26], and k-nearest neighborhood (k-NN) regression [27] were used to obtain the performance of each feature extraction algorithm.

For MLP, standard MLP with one hidden layer was used. The number of input nodes was set to the number of extracted features and one output node was used. For each problem, various numbers of hidden nodes (2, 4, 6, 8, 10, 15, and 20) were used in the training and the one with best performance on the test data was chosen. As transfer functions of hidden and output layers, *tansig (tangent sigmoid)* and *purelin (pure linear*) were used respectively. As a training rule of the MLP, *trainlm (Levenberg-Marquardt)* was used. The weight update rule of the method is

$$W_{mlp}(k+1) = W_{mlp}(k) - (J^T J + \alpha I)^{-1} J^T \boldsymbol{e}_{mlp}.$$

Here, J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights, and $\boldsymbol{e}_{mlp}$ is a vector of network errors. For adaptive value α, default settings of the *Matlab* were used [25].

For SVM, the *SVM for regression* presented in [26] [28] was used. For each problem, polynomial kernels of degree 1,2, and 3 were trained and the one with best performance on the test data were chose. All the other parameters were set to default values.

For k-NN regression, the following weighted version of the k-NN regression [27] was used

$$\hat{\boldsymbol{t}}(\boldsymbol{z}) = \frac{\sum_{i=1}^{k} q(\boldsymbol{z}, \boldsymbol{z}_i)\boldsymbol{t}_i}{\sum_{i=1}^{k} q(\boldsymbol{z}, \boldsymbol{z}_i)} \tag{16}$$

where $\boldsymbol{z}_i$ indicates the i-th nearest neighbor of the data point $\boldsymbol{z}$ and $\boldsymbol{t}_i$ is the corresponding target vector. The kernel function $q(\boldsymbol{z}, \boldsymbol{z}_i) = \frac{1}{1+\sqrt{||\boldsymbol{z}-\boldsymbol{z}_i||}}$ was used throughout the experiments. Here, $||\cdot||$ is an $L2$-norm of a vector. The number of neighbors k was varied among one, three and five for all the problems.

Note that the experimental results of ICA-FX can vary depending on the initial

condition of the rate updating rule because there may be many local optimum solutions. In ICA-FX, the learning rate μ_1 and μ_2 should be selected carefully. If they are too large, the algorithm diverges, while if they are too small, the learning slows down. Several values were tested and both μ_1 and μ_2 were set to 0.02 in all the experiments. The ICA-FX algorithm terminated if either the maximum number of iterations were reached or the change of log likelihood in (12) was below a threshold ϵ, i.e., $\mid L(\boldsymbol{W}^{(n)}|D) - L(\boldsymbol{W}^{(n-1)}|D) \mid < \epsilon$. In all the experiments, the number of maximum iterations was set to 500 and $\epsilon = 10^{-6}$ was used.

For all the problems below, the number of target variables is one. Therefore, the target variable is denoted by t instead of bold-faced $\boldsymbol{t}$.

4.1 Artificial Problems

4.1.1 Linear Case

Suppose we have five independent input variables $x_1 \sim x_5$ which have Gaussian distributions with zero mean and unit variance. Also suppose that the target output variable t has the following relationship with the input $\boldsymbol{x}$:

$$t = 2x_1 + 3x_3 + z,$$

where z is a Gaussian random noise with zero mean and variance of 0.1.

For this problem, 1000 samples were generated. Ten-fold cross-validation was applied to report the performances of various feature extraction algorithms such as PCA, MLR, SIR, PHD and ICA-FX. For comparison, we also reported the performance of regression with the original 5 inputs. Five regression meth-

ods, MLP, SVM, and k-NN regression with $k = 1, 3, 5$, were tested and the averages of the five methods were reported in Table 1. For MLP, various numbers of hidden nodes (2, 4, 6, 8, 10, 15, and 20) were used in the training and the best performance on the test data was used in the averaging process. Likewise, for SVM, the polynomial kernels of degree 1, 2, and 3 were trained and the best performance on the test data was used in the averaging.

In the table, we also reported the best performance among the five regression methods in the second row of each feature extraction algorithm. The performances are the root mean square (rms) errors on the test data. The numbers in the parentheses are the averages of standard deviations for each feature extraction algorithm.

As can be seen in the table, when the number of extracted features was one, all the feature extraction algorithms except PCA and PHD performed well and resulted almost the same average rms error around 0.12. Note that MLR is optimal in the least squared error sense, but in some regression methods such as MLP, ICA-FX performed slightly better than MLR. This is due to the fact that the reported rms error is on the test data. The reason PHD performed poorly can be explained by the fact that there is no Hessian direction because the target value is linear to the input variables. As the number of extracted features increased, the performances of all the other methods except PCA and PHD became worse while those of PCA and PHD improved. From this we can infer that all the feature extraction algorithms except PCA and PHD extracted a near best feature as the first feature and the added features from the second to the fifth did not contribute to the enhancement of the regression performance. On the other hand, PCA and PHD failed to extract a good feature as the first one and the added information in the additional features

Table 1

Performance for the simple linear dataset (rms error). Averages of five regression methods (MLP, SVM, 1-NN, 3-NN, and 5-NN). Each methods were tested with 10-fold CV and the numbers in the parentheses are the averages of standard deviations corresponding to each regression method. The second row of each feature extraction algorithm shows the best performance among the five regression methods.

No. of Features	1	2	3	4	5
Original	–	–	–	–	0.70 (0.07)
	–	–	–	–	0.09 (MLP)
PCA	3.32 (0.45)	3.05 (0.12)	3.00 (0.16)	3.05 (0.16)	0.70 (0.06)
	2.84 (SVM)	2.61 (SVM)	2.59 (SVM)	2.69 (MLP)	0.09 (MLP)
MLR	0.13 (0.02)	–	–	–	–
	0.10 (MLP)	–	–	–	–
SIR	0.14 (0.02)	0.25 (0.04)	0.40 (0.04)	0.55 (0.04)	0.70 (0.06)
	0.11 (MLP)	0.11 (MLP)	0.10 (MLP)	0.10 (MLP)	0.09 (MLP)
PHD	3.24 (0.49)	3.09 (0.42)	2.64 (0.66)	1.40 (0.72)	0.70 (0.06)
	2.75 (SVM)	2.40 (SVM)	2.32 (MLP)	0.98 (MLP)	0.09 (MLP)
ICA-FX	**0.12 (0.02)**	**0.13 (0.02)**	**0.14 (0.02)**	**0.15 (0.03)**	**0.16 (0.03)**
	0.09 (MLP)	**0.09 (MLP)**	**0.09 (MLP)**	**0.09 (MLP)**	**0.09 (MLP)**

contributed to the performance enhancement of the regression system.

Note that regardless of the number of extracted features, ICA-FX performed the best. In addition, the average rms error of ICA-FX did not degrade much compared to that of SIR as the number of extracted features increased. This is due to the fact that in SIR, the extracted features were orthogonal to one another that the second to fifth extracted features contained little information about the target variable and acted as noises in regression systems. On the other hand, since the structure of weight matrix in (7) de-

pends on the number of extracted features M, the n-th extracted features of ICA-FX were different from one another as the number of extracted features M varied. For example, when we set $M = 1$, the extracted feature was $f_1 = -5.564x_1 + 0.017x_2 - 8.562x_3 + 0.008x_4 + 0.001x_5$, while when $M = 2$, they became $f_1 = 5.973x_1 + 0.062x_2 + 9.189x_3 - 0.019x_4 + 0.006x_5$ and $f_2 = 0.754x_1 - 0.830x_2 + 1.070x_3 + 0.311x_4 - 0.121x_5$. Note that when $M = 1$, the weight vector of f_1 was almost parallel to $[2, 0, 3, 0, 0]^T$ which is the optimal value. Note also that when $M = 2$, the first feature was almost the same as that of $M = 1$ and for the second feature, the ratio between the weights corresponding to x_1 and x_3 was almost $2 : 3$ which was the same as the optimal solution, while the weights corresponding to the other inputs were raised much. This shows that f_2 as well as f_1 contained some information about the target variable when we set $M = 2$.

When the number of extracted features is equal to the original number of features, 5, the average rms errors of different methods except ICA-FX were the same (0.70) and that of ICA-FX was far better than this (0.16). This result was originated from the k-NN regression method. Considering that PCA, SIR, and PHD search for orthogonal transformations, the Euclidian distance between two points in the original input space is preserved in the feature space generated by these methods if the dimension of the feature space is equal to that of the original input space. Therefore, the k-NN regression method, which is based on the Euclidian distance, results in the same rms errors for these methods. On the other hand, because ICA-FX produces non-orthogonal transformations, the Euclidian distance is not preserved by ICA-FX and its rms error by k-NN regression becomes different from those of other methods. In case of MLP and SVM, the rms errors with 5 features were almost the same

regardless of the dimensionality reduction methods.

As can be seen in the table, SVM and MLP performed better than k-NN regression regardless of the number of extracted features but the performance differences among different regression methods were not high.

As a summary, in ICA-FX, all the M extracted features shared information about the target variable and the performance of regression did not degrade much as the number of extracted features increased. However, since the extracted feature corresponding to $M = 1$ was near optimal, the performance of regression system degraded slightly in ICA-FX also.

4.1.2 Nonlinear Case

Suppose we have five independent input variables $x_1 \sim x_5$ which have Gaussian distributions with zero mean and unit variance. Furthermore, suppose that the target output variable t has the following nonlinear relationship with the input $\boldsymbol{x}$:

$$t = sin(x_2 + 2x_4) + z.$$

Here, z is the Gaussian random noise with zero mean and variance of 0.1.

For this problem, 1000 samples were generated. Ten-fold cross-validation was applied to this dataset and the performances of various feature extraction algorithms on the test data are reported in Table 2. The numbers in the table are the average rms errors of five regression methods: MLP, SVM, 1-NN, 3-NN, 5-NN. For comparison, we also showed the performance with the original 5 inputs in the table. The rms error of MLP used in the averaging process was the best one among different numbers of hidden nodes (2, 4, 6, 8, 10, 15, and

Table 2

Performance for the nonlinear dataset (rms error). Averages of five regression methods (MLP, SVM, 1-NN, 3-NN, and 5-NN). Each methods were tested with 10-fold CV and the numbers in the parentheses are the averages of standard deviations corresponding to each regression method. The second row of each feature extraction algorithm shows the best performance among the five regression methods.

No. of Features	1	2	3	4	5
Original	–	–	–	–	0.44 (0.03)
	–	–	–	–	0.07 (MLP)
PCA	0.80 (0.04)	0.80 (0.04)	0.77 (0.05)	0.68 (0.07)	0.44 (0.03)
	0.68 (SVM)	0.70 (SVM)	0.68 (MLP)	0.57 (5-NN)	0.07 (MLP)
MLR	0.65 (0.04)	–	–	–	–
	0.56 (MLP)	–	–	–	–
SIR	0.62 (0.05)	0.63 (0.04)	0.61 (0.04)	0.60 (0.07)	0.44 (0.04)
	0.53 (MLP)	0.52 (MLP)	0.50 (MLP)	0.45 (MLP)	0.07 (MLP)
PHD	0.79 (0.03)	0.76 (0.03)	0.72 (0.05)	0.65 (0.08)	0.44 (0.03)
	0.69 (SVM)	0.67 (MLP)	0.63 (MLP)	0.53 (MLP)	0.07 (MLP)
ICA-FX	**0.59 (0.10)**	**0.55 (0.09)**	**0.49 (0.07)**	**0.45 (0.04)**	**0.43 (0.04)**
	0.50 (MLP)	**0.45 (MLP)**	**0.31 (MLP)**	**0.17 (MLP)**	**0.07 (MLP)**

20). Likewise, the rms error of SVM was set to the best one with polynomial kernels of degree 1,2, and 3. The numbers in the parentheses are the average of the five standard deviations corresponding to each regression method. In the second row, we also show the best rms error among the five regression methods.

As can be seen in the table, regardless of the number of extracted features, ICA-FX outperformed the other methods. It even performed better than MLR which is optimal in the least squared error sense. It is because the rms error

is on the test data and the mutual information criterion used in ICA-FX is better suited for nonlinear problems than the least squared error criterion. In fact, the weight vector for MLR was $[0.036, 0.401, -0.012, 0.847, -0.347]^T$ while it was $[0.000, -0.466, -0.009, -0.935, 0.061]^T$ for ICA-FX with $M = 1$. Note that the weights of x_1, x_3 and x_5 are smaller in ICA-FX than MLR and the ratio between the second and the forth weights is closer to the optimal ratio 1 : 2 in ICA-FX than that in MLR. As in the linear case, compared to the other methods, the performance of ICA-FX did not vary much with different numbers of extracted features. Among the five regression methods, MLP was the best in many cases.

4.2 Real world datasets

4.2.1 Housing - Boston

In this section, we have applied the proposed feature extraction algorithm to the *Housing (Boston)* dataset in the UCI Machine Learning Repository [29].

The dataset contains 13 input variables, 12 continuous and 1 binary, and one continuous target variable. There are total 506 instances. We randomly divided this dataset into 90% training and 10% test sets 10 times and reported the average rms error on the test data in Table 3. As in Table 1 and Table 2, the numbers in the table are the averages of the five regression methods (MLP, SVM, 1-NN, 3-NN, 5-NN) and the numbers in the parentheses are the averages of the standard deviations of 10 experiments corresponding to the five regression methods. We also reported the best rms error and the corresponding regression method in the second row of each feature extraction algorithm.

Table 3

Performance for the Housing dataset (rms error). Averages of five regression methods (MLP, SVM, 1-NN, 3-NN, and 5-NN). Each methods were tested 10 times and the numbers in the parentheses are the averages of standard deviations of the 10 experiments corresponding to each regression method. The second row of each feature extraction algorithm shows the best performance among the five regression methods.

No. of f.	1	3	5	7	9	11	13
Original	–	–	–	–	–	–	3.56 (0.51)
	–	–	–	–	–	–	3.28 (MLP)
PCA	8.19 (1.18)	4.70 (0.85)	4.24 (0.82)	3.78 (0.68)	3.82 (0.70)	3.59 (0.60)	3.53 (0.63)
	7.33 (SVM)	4.43 (5-NN)	3.70 (MLP)	3.36 (SVM)	3.42 (MLP)	3.31 (SVM)	3.27 (MLP)
MLR	4.41 (0.69)	–	–	–	–	–	–
	3.65 (MLP)	–	–	–	–	–	–
SIR	4.61 (0.73)	4.39 (0.69)	3.80 (0.63)	3.73 (0.71)	3.80 (0.81)	3.85 (0.62)	3.60 (0.63)
	3.85 (MLP)	3.65 (MLP)	3.60 (SVM)	3.34 (MLP)	3.37 (MLP)	3.35 (MLP)	3.32 (SVM)
PHD	9.07 (1.41)	5.67 (0.94)	4.89 (0.96)	4.58 (1.03)	4.01 (0.79)	3.95 (0.68)	3.63 (0.91)
	7.85 (MLP)	5.16 (5-NN)	4.57 (5-NN)	4.24 (3-NN)	3.63 (MLP)	3.61 (MLP)	3.41 (MLP)
ICA-FX	**4.32 (0.59)**	**4.09 (0.53)**	**3.74 (0.51)**	**3.37 (0.55)**	**3.48 (0.63)**	**3.61 (0.72)**	**3.61 (0.75)**
	3.59 (MLP)	**3.35 (MLP)**	**3.43 (5-NN)**	**3.25 (3-NN)**	**3.20 (MLP)**	**3.27 (SVM)**	**3.35 (MLP)**

From the table, we can see that for most numbers of extracted features, ICA-FX performed better than other conventional methods. It is so especially when the number of extracted features was small. The performance of ICA-FX was almost constant, irrespective of the number of extracted features. On the other hand, the performances of other methods varied much with different numbers of extracted features. Note that the best average rms error of ICA-FX was 3.48 and it was slightly better than that using all the 13 original inputs. For

this dataset, MLP was generally better than other regression methods.

4.2.2 Orange Juice

Orange juice dataset was obtained from the UCL machine learning database [30]. The purpose of the dataset is to estimate the level of saccharose of an orange juice from its observed near-infrared spectrum. It consists of 150 training and 68 test examples with 700 input variables. The target is a continuous variable which corresponds to the level of saccharose.

As can be seen, this problem is a typical example of small sample size (SSS) problem whose input dimension $d(= 700)$ is much larger than the number of training samples $n(= 150)$. In classification problems, especially in literatures on LDA, many techniques have been proposed to cope with the SSS problem [31] [32] [33] [34]. The simplest approach is to reduce the dimension of the input space from d to the order of the sample size $n(< d)$. This is achieved by preprocessing the data with PCA and normalizing each eigenvector corresponding to the nonzero eigenvalue to have unit variance. This method is usually called as the sphering process. Because the sphering process has the effect of magnifying the eigenvectors corresponding to small eigenvalues, a small diagonal matrix can be added to the covariance matrix to reduce this phenomenon. By doing this, all the eigenvalues of the modified covariance matrix becomes nonzero and the full input dimension can be utilized in the learning process. This is usually called as the regularization technique. After regularization, sphering process can be used to scale each eigenvector. In this experiment, to resolve the SSS problem, for all the feature extraction methods except PCA, we applied sphering process which reduced the dimension of

Table 4

Performance for the Orange Juice dataset (rms error). Averages of five regression methods (MLP, SVM, 1-NN, 3-NN, and 5-NN). The parentheses in ICA-FX are the standard deviations of 10 experiments. The second row of each feature extraction algorithm shows the best performance among the five regression methods.

no. of features	1	5	9	13	700
Original	–	–	–	–	10.34
	–	–	–	–	8.10 (MLP)
PCA	10.73	11.07	10.37	10.45	–
	9.14 (MLP)	9.10 (5-NN)	8.15 (MLP)	8.55 (MLP)	–
MLR	9.42	–	–	–	–
	7.46 (5-NN)	–	–	–	–
SIR	10.56	10.78	10.21	10.45	–
	9.32 (5-NN)	9.14 (5-NN)	6.97 (MLP)	8.45 (MLP)	–
PHD	12.50	10.93	10.62	10.39	–
	9.18 (MLP)	9.36 (5-NN)	8.68 (MLP)	8.25 (MLP)	–
ICA-FX	**8.51 (1.34)**	**7.88 (0.97)**	**7.85 (0.91)**	**8.01 (0.96)**	–
	7.13 (MLP)	**7.00 (5-NN)**	**6.27 (MLP)**	**6.15 (MLP)**	–

input space into $149(= n - 1)$.

Table 4 shows the averages of the rms errors on the test dataset obtained by the five regression methods (MLP, SVM, 1-NN, 3-NN, and 5-NN) with various feature extraction algorithms. Because the result of ICA-FX depends on the initial random weight, we performed ICA-FX with different initial weights 10 times and showed the standard deviations in the parentheses. As in the previous tables, the best rms errors among the five regression methods were shown in the second row of each feature extraction algorithm.

For this problem, the MLR performed better than other methods except ICA-FX. ICA-FX performed best regardless of the number of extracted features. When the number of features was one, the average rms error of ICA-FX was better than MLR by around 0.9 and it was better than the other methods at least by 2.0 regardless of the number of features.

The best rms error of ICA-FX was obtained by MLP when 13 features were extracted. Comparing different regression systems, MLP and 5-NN regression were relatively better than other methods. More specifically, when the number of extracted features is small, the performance of 5-NN was comparable to that of MLP. However, as the number of extracted features increases, MLP outperformed 5-NN. The reason is because we used Euclidian distance ($L2$-norm) in k-NN and if the dimension of input space is too high, the inputs that contain little information on the target variable act as noise, leading to a degraded performance. When MLP was used, the 700 original inputs resulted in rms error of 8.10 while the best rms error of ICA-FX with MLP was 6.15 which was obtained when $M = 13$. From this experiment, we can see that ICA-FX performs good not only for the problems with small numbers (several tens) of original input variables but also for the problems with relatively large numbers of inputs up to several hundreds.

5 Conclusions

In this paper, we have extended the feature extraction algorithm ICA-FX to regression problems. The proposed algorithm is based on the standard ICA and can generate very useful features for regression 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, we added output target information in training ICA. With the additional target information we can extract new features containing maximal information about the target.

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 showed that the proposed algorithm generates good features that outperform the original inputs and other features extracted from other methods. Because the original ICA is suited for processing large datasets such as biomedical ones, the proposed algorithm is also expected to perform well for large-scale regression problems.

In the proposed method, the number of extracted features can be arbitrarily chosen. However, the best number of extracted features are not known in advance. In our future work, we will study how to systematically determine the number of extracted features for the proposed method.

References

[1] V. Cherkassky, I. Mulier, Learning from Data, John Wiley & Sons, 1998, Ch. 5.

[2] G. John, Enhancements to the data mining process, Ph.D. thesis, Computer Science Dept., Stanford University (1997).

[3] A. Webb, Statistical pattern recognition, 2nd Edition, Wiley, 2002, Ch. 9.

[4] H. Liu, H. Motoda, Feature extraction, construction, and selection: A data mining perspective, Kluwer Academic Publishers, 1998.

[5] I. Joliffe, Principal Component Analysis, Springer-Verlag, 1986.

[6] A. Hyvarinen, J. Karhunen, E. Oja, Independent component analysis, John Wiley & Sons, 2001.

[7] A. Bell, T. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Computation 7 (6) (1995) 1129–1159.

[8] T.-W. Lee, M. Girolami, T. Sejnowski, Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources, Neural Computation 11 (2) (1999) 417–441.

[9] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd Edition, Academic Press, 1990.

[10] P. Viola, W. W. III, Alignment by maximization of mutual information, International Journal of Computer Vision 24 (2) (1997) 137–154.

[11] S. Ullman, M. Vidal-Naquet, E. Sali, Visual features of intermediate complexity and their use in classification, Nature Neuroscience (7) (2002) 682–687.

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

[13] J. F. III, J. Principe, A methodology for information theoretic feature extraction, in: Proc. Int'l Joint Conf. on Neural Networks 1998, Anchorage, Alasca, 1998, pp. 1712–1716.

[14] K. Torkkola, Nonlinear feature transforms using maximum mutual information, in: Proc. 2001 Int'l Joint Conf. on Neural Networks, Washington D.C., 2001, pp. 2756 –2761.

[15] N. Kwak, C.-H. Choi, Feature extraction based on ica for binary classification problems, IEEE Trans. on Knowledge and Data Engineering 15 (6) (2003) 1374–1388.

[16] N. Kwak, Feature extraction based on direct calculation of mutual information, International Journal of Pattern Recognition and Artificial IntelligenceAccepted for publication.

[17] T. Cover, J. Thomas, Elements of Information Theory, John Wiley & Sons, 1991.

[18] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Trans. Neural Networks 5 (4) (1994) 537 – 550.

[19] N. Kwak, C.-H. Choi, Input feature selection for classification problems, IEEE Trans. Neural Networks 13 (1) (2002) 143 – 159.

[20] N. Kwak, C.-H. Choi, Input feature selection by mutual information based on parzen window, IEEE Trans. on Pattern Analysis and Machine Intelligence 24 (12) (2002) 1667–1671.

[21] S. Weisberg, Applied linear regression, 2nd Edition, John Wiley, New York, 1985, Ch. 3, p. 324.

[22] M. Loog, Supervised dimensionality reduction and contextual pattern recognition in medical image processing, Ponsen & Looijen, Wageningen, The Netherlands, 2004, Ch. 3.

[23] K. C. Li, Sliced inverse regression for dimension reduction (with discussioin), Journal of the American Statistical Association 86 (1991) 316–342.

[24] K. C. Li, On principal hessian directions for data visualization and dimension reduction: Another application of Stein's lemma, Journal of the American Statistical Assiciation 87 (1992) 1025–1039.

[25] The neural network toolbox for Matlab, http://www.mathworks.com/products/neuralnet/.

[26] A. Smola, B. Scholkopf, A tutorial on support vector regression, Tech. Rep. TR-1998-030, Neuro Colt Royal Holloway College (1998).

[27] E. A. Narayada, On estimating regression, Theory Probab. Appl. 9 (1964) 141–142.

[28] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, SVM and kernel methods Matlab Toolbox, Perception Systemes et Information, INSA de Rouen, Rouen, France (2005).

[29] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998).

[30] M. Meurens, Orange juice data, Universite Catholique de Louvain, BNUT unit., http://www.dice.ucl.ac.be/mlg/DataBases/ORANGE_JUICE/.

[31] J. Lu, K. Plataniotis, A. Venetsanopoulos, Regularization studies of linear discriminant analysis in small sample size scenarios with applications to face recognition, Pattern Recognition Letters 26 (2005) 181–191.

[32] L. Chen, H. Liao, M. Ko, J. Lin, G. Yu, A new lda-based face recognition system which can solve the small sample size problem, Pattern Recognition 33 (2000) 1713–1726.

[33] J. Yang, F. Frangi, J.-Y. Yang, D. Zhang, Z. Jin, Kpca plus lda: A complete kernel fisher discriminant framework for feature extraction and recognition, IEEE Trnasactions on Pattern Analysis and Machine Intelligence 27 (2) (2005) 230–244.

[34] X. S. Zhou, T. S. Huang, Small sample learning during multimedia retrieval

using biasmap, in: Proc. IEEE International Conference on Computer Vision and Pattern Recognition, Vol. 1, Hawaii, USA, 2001, pp. 11–17.