Dimensionality Reduction Based on ICA for Regression Problems

Nojun Kwak[1] and Chunghoon Kim[2]

[1] Samsung Electronics, Suwon P.O. Box 105, Gyeonggi-Do, 442-742 Korea,
nojunk@ieee.org,
http://csl.snu.ac.kr/∼nojunk
[2] School of Electrical Engineering and Computer Science, Seoul National University,
#047, San 56-1, Sillim-dong, Gwanak-gu, Seoul 151-744, Korea
spinoz@csl.snu.ac.kr

Abstract. In manipulating data such as in supervised learning, we often extract new features from the original features for the purpose of reducing the dimensions of feature space and achieving better performance. In this paper, we show how standard algorithms for independent component analysis (ICA) can be applied to extract features for regression problems. The advantage is that general ICA algorithms become available to a task of feature extraction for regression problems by maximizing the joint mutual information between target variable and new features. Using the new features, we can greatly reduce the dimension of feature space without degrading the regression performance.

1 Introduction

In regression problems, one is given an array of attributes to predict the target value. These attributes are called *features*, and there may exist irrelevant or redundant features to complicate the learning process, thus leading to incorrect prediction. Even when the features presented contain enough information about the target variable, they may not predict the target correctly because the dimension of feature space may be so large that it may require numerous instances to determine the relationship between input features and target. This problem can be avoided by selecting only the relevant features or extracting new features containing the maximal information about the target variable from the original ones. The former methodology is called feature selection or subset selection, while the latter is named feature extraction which includes all the methods that compute any functions, logical or numerical, from the original.

This paper considers the feature extraction problem since it often results in improved performance by extracting new features from the original, especially when small number of dimensions is required. Among the various approach, we focus on finding an appropriate subspace spanned by a set of new features that are arbitrary linear combinations of original. PCA (principle component analysis) [1], ICA (independent component analysis) [2] [3], and LDA (linear discriminant analysis) [4] are the most popular subspace methods. However,

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 1–10, 2006.

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.

In our previous work, we have developed ICA-FX (feature extraction based on independent component analysis) [5], a supervised feature extraction method for classification problems, by modifying the learning rule of original ICA. Like ICA, it utilizes higher order statistics, while unlike ICA, it was developed as a supervised method in that it includes the output class information to find an appropriate feature subspace. This method is well-suited for classification problems in the aspect of constructing new features that are strongly related to output class.

In this paper, the ICA-FX for classification problems is extended to regression problems. Because the output class label was coded as a numerical value in the ICA-FX algorithm, the method can be easily applied to regression problems without changing much from original ICA-FX for classification problems.

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

$$\boldsymbol{x}(n) = A\boldsymbol{s}(n) \tag{1}$$

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

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

is as independent as possible. The original sources $\boldsymbol{s}(n)$ are exactly recovered when W is the 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 independent of time n and are drawn from independent identical distribution $p_i(s_i)$.

To find W in (2), Bell and Sejnowski [2] 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 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.

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 [2], 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 the Jacobian matrix whose (i, j)th element is 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 algorithm [3] 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 [5], 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 for 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 output class is contained in the newly generated features. This can be formalized as follows.

(Problem statement) *Assume that there are a normalized input feature 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 output 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 this 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* $\boldsymbol{x}$*, such that* $I(\boldsymbol{f}_a; \boldsymbol{t})$*, the mutual information between newly extracted features* $\boldsymbol{f}_a$ *and target output 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 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 feature 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 $u_{N+l} = t_l$, $l = 1, \cdots, N_t$. 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 features, $\boldsymbol{x}$, such that

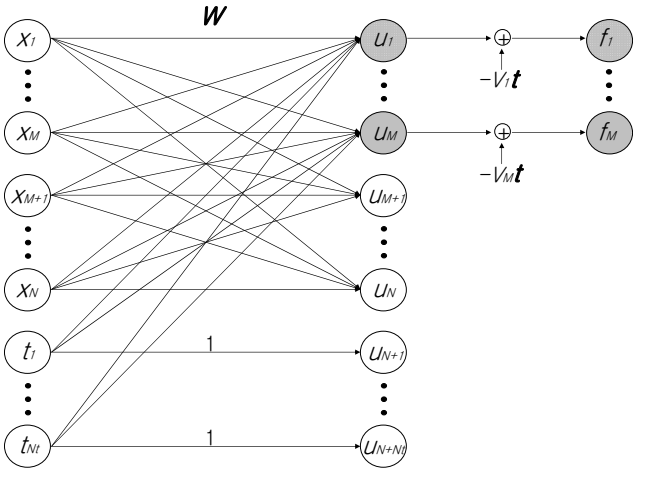

Fig. 1. Feature extraction algorithm based on ICA (ICA-FX)

$I(\boldsymbol{f}_a; t)$, 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 features $\boldsymbol{x}$ and the target variables $\boldsymbol{t}$.

This can be satisfied if we can separate the input feature space $\boldsymbol{x}$ into two linear subspaces: one that is spanned by $\boldsymbol{f}_a = [f_1, \cdots, f_M]^T$, 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.

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, which can be represented with the direct sum of $\boldsymbol{f}_a$ and $\boldsymbol{f}_b$, and then by the data processing inequality,

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

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 $\boldsymbol{f}_b$. In the inequality on the last line, the equality holds if $I(\boldsymbol{f}_b; \boldsymbol{t}) = I(u_{M+1}, \cdots, u_N; t) = 0$.

If this is possible, the dimension of the input feature 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 solve this problem, 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 t. Assume also that the ob-*

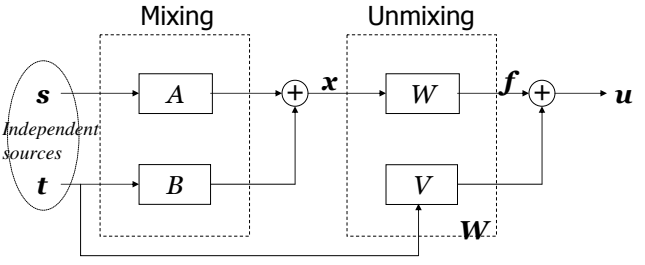

Fig. 2. Interpretation of Feature Extraction in the BSS structure

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

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

(Unmixing) *The unmixing stage is slightly different from the BSS problem as shown in Fig. 1. 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* t *because the* i*th row of* V*,* $V_i = [w_{i,N+1}, \cdots, w_{i,N+N_t}] = \boldsymbol{0}$ *and* $f_i = u_i$ *for* $i = M+1, \cdots, N$*. Therefore, the* $M (< N)$ *dimensional new feature vector* $\boldsymbol{f}_a$ *can be extracted by a linear transformation of* $\boldsymbol{x}$ *containing the most information on the class 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 the data for a given $\boldsymbol{W}$ becomes the following:

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

because

$$p(\boldsymbol{x}, \boldsymbol{t}|\boldsymbol{W}) = |\det \boldsymbol{W}|\ p(\boldsymbol{u}, \boldsymbol{t}) = |\det \boldsymbol{W}|\ \prod_{i=1}^{N} p_i(u_i)\ p(\boldsymbol{t}). \tag{13}$$

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

$$\begin{aligned} \frac{\partial L}{\partial w_{i,j}} &= \frac{adj(w_{j,i})}{|\det \boldsymbol{W}|} - \varphi_i(u_i)x_j \qquad 1 \le i,j \le N \\ \frac{\partial L}{\partial w_{i,N+j}} &= -\varphi_i(u_i)t_j \qquad 1 \le i \le M, 1 \le j \le N_t \end{aligned} \tag{14}$$

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

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

$$\begin{aligned} \Delta W &\propto W^{-T} - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{x}^T \\ \Delta V_a &\propto -\boldsymbol{\varphi}(\boldsymbol{u}_a)\boldsymbol{t}^T. \end{aligned} \tag{15}$$

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

Applying a natural gradient on updating W, by multiplying W^TW on the right side of the first equation of (15), the following is obtained.

$$\begin{aligned} W^{(n+1)} &= W^{(n)} + \mu_1[I_N - \boldsymbol{\varphi}(\boldsymbol{u})\boldsymbol{f}^T]W^{(n)} \\ V_a^{(n+1)} &= V_a^{(n)} - \mu_2\boldsymbol{\varphi}(\boldsymbol{u}_a)\boldsymbol{t}^T. \end{aligned} \tag{16}$$

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 learning rule for W is the same as the original ICA learning rule [2], 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 (16).

4 Experiment Results

In this section, we have applied ICA-FX to several regression problems and show the performance of the method. For all the problems below, we have compared the performance of ICA-FX with those of PCA and original features.

As a regression tool, standard multi-layer perceptron (MLP) with one hidden layer was used. The numbers of nodes in input, hidden, and output layers were set to the number of extracted features, six, and one respectively. As a 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

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

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 μ, default settings of the *Matlab* were used.

4.1 Artificial Problems

Linear Case. Consider the simple problem of the following:

Suppose we have five independent input features $x_1 \sim x_5$ which have normal distribution with zero mean and variance of 1. Also suppose that the target output variable t has the following relationship with the input $\boldsymbol{x}$: $t = 2x_1 + 3x_3$.

For this problem, 1000 samples were generated which were divided into 500 training data and 500 test data. On the training data, ICA-FX was applied where the number of extracted feature M was set to 1. The first row of weight matrix $\boldsymbol{W}$ in (7) after ICA-FX was [7.0904, 0.0004, 10.9933, 0.0002, 0.0003, -13.1630]. Thus the newly extracted feature is $f = 7.0904x_1 + 0.0004x_2 + 10.9933x_3 + 0.0002x_4 + 0.0003x_5$. Note that the coefficients for x_2, x_4 and x_5 are very small compared to those of x_1 and x_3. In addition, note that the ratio of coefficient of x_1 and that of x_3 is approximately 2:3. The extracted weights show that the ICA-FX performs quite well for linear regression problems.

For this problem, we have also performed PCA and compared the performance of the ICA-FX, PCA, and the original 5 features in Table 1. In this experiment, the number of extracted features for ICA-FX was varied from 1 to 5. The performance is the root mean square (rms) error of the test data. Averages of 10 experiments with standard deviations are reported here.

Table 1. Performance for the simple linear dataset (rms error). Averages of 10 experiments. Numbers in the parentheses are the standard deviations.

	no. of features	rms error
Original	5	0.00 (0.00)
PCA	1	2.85 (0.64)
	3	2.17 (0.68)
	5	0.00 (0.00)
ICA-FX	1	0.01 (0.02)
	3	0.00 (0.00)
	5	0.00 (0.00)

The table also shows that the ICA-FX performs well with a small number of features.

Nonlinear Case. Consider the following problems:

As the problem above, suppose we have five independent input features $x_1 \sim x_5$ which have normal distribution with zero mean and variance of 1. Now,

suppose that the target output variable t has the following nonlinear relationship with the input $\boldsymbol{x}$*:* $t = sin(x_2 + 2x_4)$.

For this problem, 1000 samples were generated which were divided into 500 training data and 500 test data. On the training data, ICA-FX was applied where the number of extracted feature M was set to 1. The first row of weight matrix $\boldsymbol{W}$ in (7) after ICA-FX was [-0.0176, -0.5146, -0.0982, -0.9607, 0.0046, 0.4886]. Thus the newly extracted feature is $f = -0.0176x_1 - 0.5146x_2 - 0.0982x_3 - 0.9607x_4 + 0.0046x_5$. Note that the coefficients for x_1, x_3 and x_5 are very small compared to those of x_2 and x_4. This indicates that the resultant weights after ICA-FX can be used to select appropriate features. In addition, note that the ratio of coefficient of x_2 and that of x_4 is approximately 1:2 which is the ratio of the corresponding coefficients inside *sin* term. The extracted weights show that the ICA-FX performs well for this nonlinear regression problem too.

As in the linear case, we have performed PCA for this dataset and compared the rms error of the ICA-FX with those of PCA and the original 5 features in Table 2 The number of extracted features for ICA-FX was varied from 1 to 5.

Table 2. Performance for the simple nonlinear dataset (rms error). Averages of 10 experiments. Numbers in the parentheses are the standard deviations.

	no. of features	rms error
Original	5	0.14 (0.17)
PCA	1	0.73 (0.02)
	3	0.57 (0.20)
	5	0.08 (0.05)
ICA-FX	1	0.15 (0.03)
	3	0.19 (0.27)
	5	0.08 (0.06)

The table shows that the ICA-FX performs better than PCA and the original data for this dataset.

4.2 UCI Dataset - Housing (Boston)

In this section, we have applied ICA-FX to *Housing (Boston)* dataset in UCI Machine Learning Repository [6].

The dataset contains 13 input features, 12 continuous and 1 binary, and one continuous target variable. There are total 506 instances. We have 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. In applying ICA-FX, we have normalised each input feature to have zero mean and unit variance and varied the number of extracted features M from 1 to 13.

Note that ICA-FX performs better than PCA generally and the performance was robust irrespective of the number of extracted features. Note also that the ICA-FX performs better than using 13 original features.

Table 3. Performance for the Housing dataset (rms error). Numbers in the parentheses are the standard deviations.

no. of features	original	PCA	ICA-FX
1	–	7.36 (0.68)	3.59 (0.60)
3	–	4.70 (1.17)	3.60 (0.54)
5	–	4.16 (1.72)	3.60 (0.65)
7	–	3.49 (0.51)	3.50 (0.45)
9	–	4.00 (0.91)	3.80 (1.01)
11	–	3.67 (0.71)	3.42 (0.63)
13	4.20 (0.99)	4.41 (1.73)	3.37 (0.63)

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. The number of extracted features can be arbitrarily chosen.

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. 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 regression problems.

References

1. I.T. Joliffe, *Principal Component Analysis*, Springer-Verlag, 1986.
2. 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.
3. 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.
4. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, second edition, 1990.
5. N. Kwak and C.-H. Choi, "Feature extraction based on ICA for binary classification problems," *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1374–1388, Nov. 2003.
6. http://www.ics.uci.edu/∼mlearn/MLSummary.html