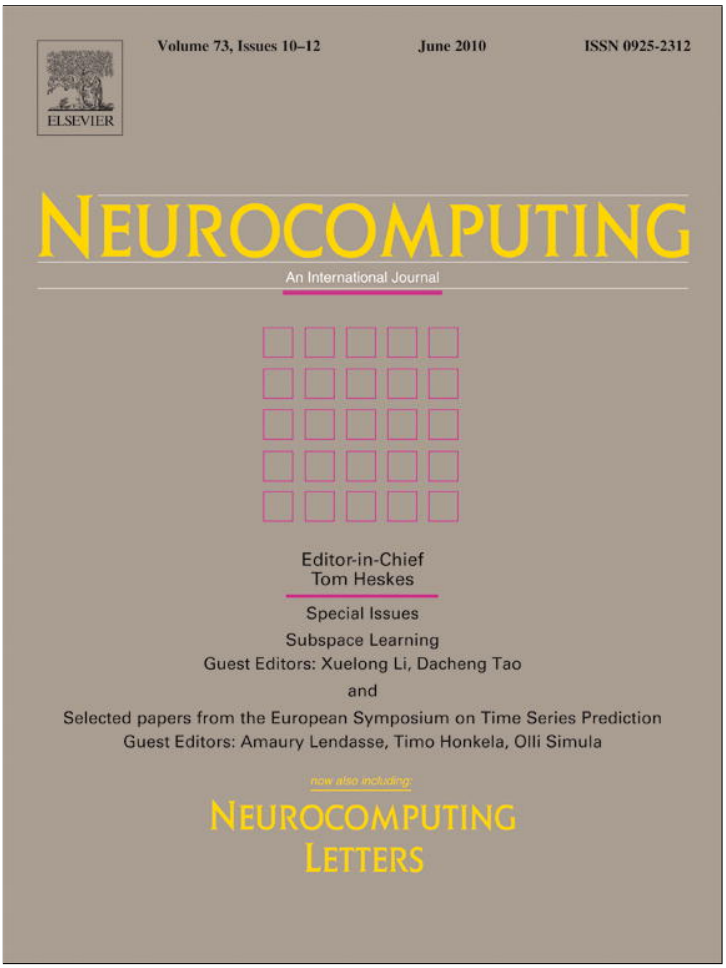

Neurocomputing 73 (2010) 1740–1751

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Feature extraction based on subspace methods for regression problems

Nojun Kwak *, Jung-Won Lee

Division of Electrical and Computer Engineering, Ajou University, San 5, Woncheon-dong, Yeongtong-gu, Suwon 443–749, Republic of Korea

ARTICLE INFO

Available online 12 March 2010

Keywords:
Regression
Feature extraction
Subspace method
Dimensionality reduction
PCA
LDA

ABSTRACT

In this paper, we propose a couple of new feature extraction methods for regression problems. The first one is closely related to the conventional principle component analysis (PCA) but unlike PCA, it incorporates target information in the optimization process and try to find a set of linear transforms that maximizes the distances between points with large differences in target values. On the other hand, the second one is a regressional version of linear discriminant analysis (LDA) which is very popular for classification problems. We have applied the proposed methods to several regression problems and compared the performance with the conventional feature extraction methods. The experimental results show that the proposed methods, especially the extension of LDA, perform well in many regression problems.

1. Introduction

Regression which is the process of estimating a real-valued function based on a finite set of noisy samples is one of the classical problems in the statistics, machine learning and pattern recognition societies. With classification problems, regression problems are classified as supervised learning. In supervised learning, we are given a set data consisting of pairs of input objects and desired outputs. The input objects and the desired outputs are normally called as the *input features* and the *target variables* respectively.

Nowadays many real world problems such as biomedical data analysis and image processing involve very large number of input features. On the other hand, there are many cases where the number of the features may be smaller, but of the same order of magnitude as the number of samples. In both cases, supervised learning faces the so-called *curse of dimensionality* where overfitting easily appears and the supervised learning may be ill-posed. In addition, irrelevant or redundant input features tend to complicate the learning process, thereby resulting in a poor generalization performance [1]. Even when the features presented contain sufficient information on the problem, the result may become erroneous because the dimension of the feature space could be too large.

For these reasons, it is desirable to reduce the number of input features through dimensionality reduction techniques such as feature selection or feature extraction. Reducing the dimensionality of the feature space may improve the learning process by considering only the most important data representation, possibly with elements retaining the maximum information on the original data and with better generalization capabilities [2]. Dimensionality reduction is quite desirable not only in the aspect of the number of required data, but also in terms of data storage and computational complexity.

In this paper, we focus on the linear feature extraction methods for regression problems to reduce the dimensionality of the input feature space.

Many studies have been performed to solve the feature extraction problems among which the principal component analysis (PCA) [3] and the independent component analysis (ICA) [4] have been widely used. Although PCA is one of the most popular and widely used methods, which is very useful in reducing the dimension of a feature space to a manageable size, it can still be improved for supervised learning problems since it is an unsupervised learning method that does not make use of the target information. Likewise, ICA, which is another unsupervised learning method, leaves much room for improvement to be used for supervised learning problems.

Unlike PCA and ICA, linear discriminant analysis (LDA) [5] is originally developed for supervised learning especially for classification problems to find the optimal linear discriminating functions. The LDA tries to find a linear combinations of original input features by maximizing the ratio of between-class scatter and within-class scatter. This optimization problem can be nicely interpreted in the recently developed spectral regression framework [6]. There are quite a lot of variants of LDA for improving the performance and coping with the limitation of LDA that it cannot produce more than N_c-1 features, where N_c is the number of classes [7–10].

Recently, instead of using only up to the second order statistics as in LDA and its variants, ICA-FX which is an extension of ICA that utilizes higher order statistics by maximizing the mutual information between the class and the features has been proposed for classification problems [11]. Neighborhood component analysis

* Corresponding author. Tel./fax: +82 31 219 2480.
E-mail addresses: nojunk@ieee.org (N. Kwak), jungwony@ajou.ac.kr (J.-W. Lee).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.10.025

(NCA) [12] and tensor based algorithms [13–15] have also been proposed as feature extraction methods for classification problems.

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 machine learning society. Recently, a Gaussian process regression (GPR) [16] was introduced and was proved to be a very successful regression method. Though this method can be regarded as a nonlinear feature extraction method, the nature of kernel trick prohibits it from being used as a dimensionality reduction technique. In [12], NCA was extended to regression problems to estimate the degrees of rotation and scaling of a face image.

In statistics, several algorithms have been developed as dimensionality reduction techniques for regression problems among which classical multivariate linear regression (MLR) [17] can be the basic. Although MLR is the optimal in the sense of least squared error, it has the limitation that it can only produce one feature. To overcome this limitation, a local linear dimensionality reduction method based on the nearest neighbor scheme has been proposed [18]. Sliced inverse regression (SIR) [19] and principal Hessian directions (PHD) [20] are also very popular dimensionality reduction techniques for regression problems in statistics.

In our previous work, ICA-FX was extended to regression problems [21] and showed good performance for various regression problems. However, because ICA based methods are iterative methods, the speed of convergence may become a problem for ICA-FX. Another limitation is that it has a good chance of overfitting because it naturally utilizes higher order statistics and the training and test sample distributions are not always the same, especially when the size of training data is small.

In this paper, we propose a couple of linear feature extraction methods for regression problems. The first one is quite related to the conventional PCA but, unlike PCA, it utilizes the target information to find a set of features that maximizes the distances between samples with large differences in target values. Thus, this can be regarded as a weighted version of PCA. The second one is a generalization of LDA to regression problems which tries to maximize the ratio of distances of samples with large differences in target value and those with small differences in target value. The experimental results show that the proposed methods especially the second one perform well for many regression problems. In addition, because both methods involve in the singular value decomposition (SVD), it is relatively faster than iterative methods such as ICA-FX.

The paper is organized as follows. In Section 2, we briefly overview the conventional feature extraction methods especially for regression problems. The two new feature extraction methods are presented in Section 3 and the experimental results are shown in Section4. Finally, in Section 5, the conclusions follow.

2. Conventional methods: linear feature extraction for regression

Consider a set of predictor/response[1] pairs $\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^n$ where $\boldsymbol{x}_i \in \Re^{d\times 1}$ $\boldsymbol{y}_i \in \Re^{t\times 1}$ and n denotes the number of given predictor/response pairs. Here, d is the number of original features and t is the number of target variables which will in most problems be equal to 1.[2]

In this regression setting, we would like to find a set of linear transformations of $\boldsymbol{x}$, that constitute sufficient statistics of the target vector $\boldsymbol{y}$. This transformation can be denoted as $f_i = \boldsymbol{w}_i^T\boldsymbol{x}$, where f_i is the i-th extracted feature and $\boldsymbol{w}_i \in \Re^{d\times 1}$ is the corresponding coefficient or weight vector.

In this section, we would like to introduce several conventional methods developed for this purpose.

2.1. Principal component analysis (PCA)

In PCA, the k-th maximum variance direction $\boldsymbol{w}_k$ is searched by solving the following eigenvalue decomposition problem

$$S_x\boldsymbol{w}_k = \lambda_k\boldsymbol{w}_k, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \tag{1}$$

where $S_x = (1/n)\sum_{i=1}^n(\boldsymbol{x}_i-\overline{\boldsymbol{x}})(\boldsymbol{x}_i-\overline{\boldsymbol{x}})^T$ is the covariance matrix of $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}$ is the mean of $\boldsymbol{x}$.

As can be seen, PCA does not utilize the target information and is very sensitive to the scaling of input features.

2.2. Multivariate linear regression (MLR)

In MLR, we try to find a direction $\boldsymbol{w}$ that results in the least squared error by solving the following optimization problem:

$$\boldsymbol{w} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^n (y_i-\boldsymbol{w}^T\boldsymbol{x}_i)^2 \tag{2}$$

and the solution is simply the vector

$$\boldsymbol{w} = \left(\sum_{i=1}^n \boldsymbol{x}_i\boldsymbol{x}_i^T\right)^{-1}\sum_{i=1}^n \boldsymbol{x}_i(y_i-\overline{y}) \tag{3}$$

where $\overline{y}$ is the mean of the response variable.

In the sense of least squared error, $f = \boldsymbol{w}^T\boldsymbol{x}$ gives the optimal linear approximation to the y and as such $\boldsymbol{w}$ can be interpreted as a good dimension reducing linear transformation. However, using MLR one can only obtain a single feature which will, in many cases, not be sufficient to perform an accurate prediction of the target variable.

2.3. Sliced inverse regression (SIR)

In this dimensionality reduction scheme, unlike conventional forward regression problems where one tries to estimate the target variable $\boldsymbol{y}$ given input variable $\boldsymbol{x}$, inverse regression problem is considered which tries to estimate $\boldsymbol{x}$ for given $\boldsymbol{y}$. It is argued that the inverse regression is much simpler than the forward regression and can effectively alleviate the curse of dimensionality, because in most cases, the dimension of input features d is much larger than that of target variables t [19].

The following is the standard SIR algorithm. For simplicity, let us assume that $t = 1$ and the covariance matrix S_x of input features $\boldsymbol{x}$ is $d \times d$ identity matrix.[3]

Step 1: Sort the data by y in the increasing order.
Step 2: Divide the data set into L slices as equally as possible. Let n_l be the number of examples in slice l.
Step 3: Within each slice, compute the sample mean of $\boldsymbol{x}$, $\overline{\boldsymbol{x}}_l = (1/n_l)\sum_{(i)\in \text{slice } l}\boldsymbol{x}_{(i)}$.
Step 4: Compute the covariance matrix for the slice means of $\boldsymbol{x}$, weighted by the slice sizes

$$S_\eta = \frac{1}{n}\sum_{l=1}^L n_l(\overline{\boldsymbol{x}}_l-\overline{\boldsymbol{x}})(\overline{\boldsymbol{x}}_l-\overline{\boldsymbol{x}})^T \tag{4}$$

[1] Note that instead of the terms *predictor* and *response*, *input* and *target* can be used without notification.

[2] From now on, we will assume $t=1$ and instead of the vector form $\boldsymbol{y}$, the scalar form y will be used without notification.

[3] This is possible by the sphering process that will be discussed in the next section.

Here, $\overline{\boldsymbol{x}}$ denotes sample mean of $\boldsymbol{x}$ such that $\overline{\boldsymbol{x}}=(1/n)\sum_{i=1}^{n}\boldsymbol{x}_i$.

Step 5. Find the k-th SIR direction $\boldsymbol{w}_k$ by conducting the eigenvalue decomposition of S_η:

$$S_\eta \boldsymbol{w}_k=\lambda_k \boldsymbol{w}_k, \quad \lambda_1\geq\lambda_2\geq\cdots\geq\lambda_d \tag{5}$$

Note the similarity of SIR to PCA that it takes L points each of which is the sample mean of n_l points in each slice l and then performs the PCA to this L points. However, the difference is that in generating the L points, $\boldsymbol{x}$'s that are associated with similar y values are averaged out to capture the relationship between the input $\boldsymbol{x}$ and the target y.

2.4. Principal Hessian directions (PHD)

As in SIR, let us assume that $t=1$ and let $f(\boldsymbol{x})$ be the regression function $E(Y|\boldsymbol{x})$, which is a d dimensional function. Here, $E(\cdot)$ denotes expectation. Consider the Hessian matrix $H(\boldsymbol{x})$ of $f(\boldsymbol{x})$ whose (i,j) component is as follows:

$$H_{ij}(\boldsymbol{x})=\frac{\partial^2}{\partial x_i\partial x_j}f(\boldsymbol{x}) \tag{6}$$

where x_k is the k-th component of the vector $\boldsymbol{x}$.

Hessian matrices are important in studying multivariate nonlinear functions and PHD focuses on the utilization of the properties of Hessian matrices for dimensionality reduction. In PHD algorithm, instead of calculating the Hessian matrix at each location of $\boldsymbol{x}$, which is quite difficult and time-consuming, average Hessian $\overline{H}=EH(\boldsymbol{x})$ is considered and principal Hessian directions are defined as the eigenvectors $\boldsymbol{w}_k$'s ($k=1,\ldots,d$) of the matrix $\overline{H}S_x$ where S_x is the covariance matrix of $\boldsymbol{x}$:

$$\overline{H}S_x\boldsymbol{w}_k=\lambda_k\boldsymbol{w}_k, \quad |\lambda_1|\geq|\lambda_2|\geq\cdots\geq|\lambda_d| \tag{7}$$

By using Stein's lemma [22], it is shown in [20] that the average Hessian matrix $\overline{H}$ for normal $\boldsymbol{x}$ with mean $\overline{\boldsymbol{x}}$ and covariance S_x is related to the weighted covariance matrix

$$S_{yxx}=E\{(Y-\overline{y})(\boldsymbol{x}-\overline{\boldsymbol{x}})(\boldsymbol{x}-\overline{\boldsymbol{x}})^T\} \tag{8}$$

through the identity

$$\overline{H}=S_x^{-1}S_{yxx}S_x^{-1} \tag{9}$$

Without loss of generality, we can assume S_x as the $d\times d$ identity matrix[4] and the principal Hessian directions $\boldsymbol{w}_k$'s ($k=1,\ldots,d$) can be obtained by solving the following eigenvalue decomposition problem:

$$S_{yxx}\boldsymbol{w}_k=\lambda_k\boldsymbol{w}_k, \quad |\lambda_1|\geq|\lambda_2|\geq\cdots\geq|\lambda_d| \tag{10}$$

where S_{yxx} in (8) can be estimated by

$$S_{yxx}=\frac{1}{n}\sum_{i=1}^{n}(y_i-\overline{y})(\boldsymbol{x}_i-\overline{\boldsymbol{x}})(\boldsymbol{x}_i-\overline{\boldsymbol{x}})^T \tag{11}$$

2.5. Gaussian process with linear regression model (GPL)

This method is a simplified version of the more general regression method GPR [16]. In this method, the target variable y is assumed to be a linear combination of input variables $\boldsymbol{x}$ and an additive noise ε as follows:

$$y=\boldsymbol{w}^T\boldsymbol{x}+\varepsilon \tag{12}$$

If we assume that the noise is independent, identically distributed Gaussian distribution with zero mean and variance σ_n^2, the likelihood of the observation given the parameters $\boldsymbol{w}$ can be calculated as

$$p(\boldsymbol{y}|X,\boldsymbol{w})=\prod_{i=1}^{n}p(y_i|\boldsymbol{x}_i,\boldsymbol{w})=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_n}\exp\left(-\frac{(y_i-\boldsymbol{w}^T\boldsymbol{x}_i)^2}{2\sigma_n^2}\right)$$
$$=\frac{1}{(2\pi\sigma_n^2)^{n/2}}\exp\left(-\frac{1}{2\sigma_n^2}|\boldsymbol{y}-X^T\boldsymbol{w}|^2\right)=\mathcal{N}(X^T\boldsymbol{w},\sigma_n^2 I) \tag{13}$$

where n is the number of training examples, $X\in\Re^{d\times n}$ is the training data matrix, $\boldsymbol{y}\in\Re^{n\times 1}$ is the target vector of the training data and $|\boldsymbol{z}|$ denotes the Euclidian norm of the vector $\boldsymbol{z}$. In the Bayesian estimation, we assume that $\boldsymbol{w}$ has a zero mean Gaussian *prior* with covariance matrix Σ_p, i.e.,

$$\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\Sigma_p) \tag{14}$$

Using the likelihood (13) and prior (14), the posterior probability of $\boldsymbol{w}$ can be obtained as

$$p(\boldsymbol{w}|X,\boldsymbol{y})\sim\mathcal{N}\left(\frac{1}{\sigma_n^2}A^{-1}X\boldsymbol{y},A^{-1}\right) \tag{15}$$

where $A=\sigma_n^{-2}XX^T+\Sigma_p^{-1}$.

The *maximum a posteriori* estimate of $\boldsymbol{w}$ becomes the mean value of (15), i.e., $\boldsymbol{w}_{MAP}=(1/\sigma_n^2)A^{-1}X$.

From this derivation, $f=\boldsymbol{w}^T\boldsymbol{x}$ in itself can be interpreted as a new feature obtained from the linear combination of input variables $\boldsymbol{x}$. Like MLR, this method can extract only one feature.

The GPR extends this linear regression model to a nonlinear version by replacing (12) with

$$y=\boldsymbol{w}^T\phi(\boldsymbol{x})+\varepsilon \tag{16}$$

where $\phi(\boldsymbol{x})$ is a nonlinear function of the input vector $\boldsymbol{x}$. Because our focus is on the linear feature extraction, we stop the discussion on GPR here. The interested readers may consult [16] for more detail.

2.6. Linear discriminant analysis (LDA)

Unlike other methods previously introduced in this section, LDA focuses on classification problem where instead of continuous target variable y, discrete class identifier $c\in\{1,\ldots,N_c\}$ is used. Here, N_c is the number of classes.

In LDA, we try to optimize the following Fisher's criterion such that the ratio of the between-covariance matrix $S_b=(1/n)\sum_{c=1}^{N_c}n_c(\overline{\boldsymbol{x}}_c-\overline{\boldsymbol{x}})(\overline{\boldsymbol{x}}_c-\overline{\boldsymbol{x}})^T$ and the within-covariance matrix $S_w=(1/n)\sum_{c=1}^{N_c}\sum_{i\in\{\text{class}=c\}}(\boldsymbol{x}_i-\overline{\boldsymbol{x}}_c)(\boldsymbol{x}_i-\overline{\boldsymbol{x}}_c)^T$ is maximized

$$W=\arg\max_W\frac{|W^TS_bW|}{|W^TS_wW|} \tag{17}$$

Here, $\overline{\boldsymbol{x}}=(1/n)\sum_{i=1}^{n}\boldsymbol{x}_i$ is the total mean of the samples, n_c is the number of samples belonging to the class c and $\overline{\boldsymbol{x}}_c=(1/n_c)\sum_{i\in\{\text{class}=c\}}\boldsymbol{x}_i$ is the mean of the samples belonging to the class c.

The optimization problem in (17) is equivalent to the following generalized eigenvalue problem:

$$S_b\boldsymbol{w}_k=\lambda_kS_w\boldsymbol{w}_k, \quad \lambda_1\geq\lambda_2\geq\cdots\geq\lambda_d\geq 0 \tag{18}$$

where $\boldsymbol{w}_1$ is the most discriminant component, $\boldsymbol{w}_2$ is the second and so on.

In [23], the relationship between LDA and MLR was studied and the LS-LDA (least squares LDA) was proposed which showed almost the same classification performances as those of LDA.

[4] This is possible by the sphering process that will be discussed in the next section.

3. Methods

In this section, we propose two new feature extraction methods for regression problems. These are quite related to the conventional methods described in the previous section and the relationship will be discussed.

3.1. Weighted PCA for regression

In this approach, we would like to incorporate the target information in the structure of PCA in order to obtain good features suitable for regression.

For this purpose, we propose to solve the following eigenvalue decomposition problem:

$$S_{yx}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \tag{19}$$

where

$$S_{yx} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} g(y_i - y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \tag{20}$$

Here, weight function $g(\cdot)$ is a positive even function whose output does not decrease as the absolute value of the input increases. Typical examples are $g(x) = |x|$ and $g(x) = \sqrt{|x|}$. Note that the matrix S_{yx} in (20) is the weighted version of the covariance matrix S_x, and replacing S_{yx} with S_x, (19) becomes identical to (1) for PCA. Furthermore, if we use constant weight function, i.e., $g(x)=1$, then S_{yx} becomes identical to the covariance matrix S_x and the proposed algorithm becomes identical to PCA. In this case, by sphering process which will be discussed later, S_x can be made to be I_d and the extracted feature will be of little importance because any random directions will be selected as extracted features. Hereafter, we will refer to the proposed algorithm as *WPCA*.

The intuition of the proposed algorithm is very simple. In computing the weighted covariance matrix S_{yx}, it emphasizes the contribution of pairs of samples whose difference in the target values is large and reduce the effect of pairs of samples which show small differences in the target values.

Note the similarity of the weighted covariance matrix S_{yx} and that of PHD in (11). However, unlike S_{yxx} in PHD which can have negative eigenvalues, S_{yx} is positive semi-definite matrix whose eigenvalues are all non-negative.

Although WPCA takes the target information into account when generating the weighted covariance matrix S_{yx}, the drawback of PCA that the extracted features are not invariant under transformation remains still. By merely scaling one of the input features the extracted features can be different. This is not only true for PCA and WPCA but also applies to SIR and PHD.

To resolve this problem, we preprocess the given data with the well known sphering process as follows:

(Sphering): Solve the eigenvalue decomposition problem of PCA in (1). Rewriting (1) in a matrix form it becomes $S_x W_1 = W_1 \Lambda_1$. Then, scale each column $\boldsymbol{w}_{1k}$ $(k=1,\ldots,d')$ of W_1 to make $\{\hat{\boldsymbol{x}}_{ik} = \hat{\boldsymbol{w}}_{1k}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) : i=1,\ldots,n\}$ have unit variance. Here $\hat{\boldsymbol{w}}_{1k}$ is a scaled version of $\boldsymbol{w}_{1k}$ and d' is the number of non-zero eigenvalues of S_x. If $\|\boldsymbol{w}_{1k}\| = 1$, this is equivalent to setting $\hat{\boldsymbol{w}}_{1k} = \boldsymbol{w}_{1k}/\sqrt{\lambda_{1k}}$ for $\lambda_{1k} \neq 0$.[5]

[5] We can get d non-zero eigenvalues only if S_x is nonsingular. If S_x is singular, the sphering process can be performed only for the eigenvectors whose corresponding eigenvalues are non-zero. By this, the original d-dimensional input space is reduced to $d' < d$ dimensional space. If we do not want to reduce the original dimension by the preprocessing, the regularization technique which will be treated in the next subsection can be utilized.

By the above sphering process, the original input features are transformed to a set of orthonormal random variables which does not show any preference for a specific direction and WPCA can be performed on this new set of features reflecting the effect of target information as much as possible while suppressing the negative effect of underlying structure of the original input features.

The computational complexity of WPCA can be decomposed into two parts. The first part is related to obtaining the covariance matrix shown in (20) and it is proportional to the square of the number of examples and the square of the input dimension, i.e., $C_1 = O(n^2d^2)$. The second part is related to solving the eigenvalue decomposition problem in (19) and it is typically proportional to the cubic of the input dimension, i.e., $C_2 = O(d^3)$.

Comparing this to the complexity of PCA, because the second part is common to PCA and WPCA, we can see that WPCA is more computationally complex than PCA which requires $O(nd^2)$ operations in obtaining the covariance matrix. However, in case where n is very large, a subset of samples can be selected to obtain the weighted covariance matrix S_{yx}.

3.2. LDA for regression

3.2.1. Algorithm

In the classification problems, LDA has been a very successful dimensionality reduction method and many variants have been developed so far. As described in the previous section, the gist of LDA lies in maximizing the Fisher's criterion which tries to maximize the between-class scatter while minimizing the within-class scatter simultaneously.

In this section, we extend this idea to the regression problems and a new feature extraction algorithm for regression is proposed. From now on, the new method will be referred to as *LDAr*, which is an abbreviation for *linear discriminant analysis for regression*.

Consider a set of input/target pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ as described in Section 3. In this regression setting, we are to find a set of features f_i's $(= \boldsymbol{w}_i^T\boldsymbol{x})$, which are linear transformations of $\boldsymbol{x}$ such that they contain much information about the target variable y. In doing so, we focus on Fisher's criterion (17) and modify it appropriately to fit in the regression problems.

Firstly, we incorporate the classification problems in this regression setting by regarding the class label as a discrete variable y. Then, the between-scatter S_b and within-scatter S_w matrices in LDA (Section 2.5) can be rewritten up to a constant scaling as

$$S_b' = \frac{1}{n_b} \sum_{(i,j) \in A_b} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$$

$$S_w' = \frac{1}{n_w} \sum_{(i,j) \in A_w} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \tag{21}$$

where A_b and A_w are the sets of index pairs with different class labels and the same labels respectively:

$$A_b = \{(i,j) | y_i \neq y_j, i<j\} = \{(i,j) | y_i - y_j \neq 0, i<j\}$$

$$A_w = \{(i,j) | y_i = y_j, i<j\} = \{(i,j) | y_i - y_j = 0, i<j\} \tag{22}$$

and n_b and n_w are the number of elements in A_b and A_w respectively. Note that $n_b + n_w = \binom{n}{2} = n(n-1)/2$.

Unlike classification problems which have discrete target variable, or classes, in regression problems it is difficult to define between-class scatter and within-class scatter matrices because target variable is continuous. However, by regarding y in the classification problem as a continuous variable and introducing a *soft class* instead of the *hard class*, we can modify (22) to fit in the

regression setting. The modified set of index pairs A_{br} and A_{wr} are defined as follows:

$$A_{br} = \{(i,j)|\ |y_i - y_j| \geq \tau, i < j\}$$

$$A_{wr} = \{(i,j)|\ |y_i - y_j| < \tau, i < j\} \quad (23)$$

Here, τ is a threshold separating an index pair into A_{br} or A_{wr} and $n_w = |A_{wr}|$ and $n_b = |A_{br}|$. Note that if $\tau = 0$, the sets of index pairs become identical to that used in LDA for classification problems (hard class).

In this modification, the simple idea that the samples with small differences in the target values can be considered as belonging to the same class and the ones with large differences should be considered as belonging to the different classes is used. Therefore, the sets A_{wr} and A_{br} can be regarded as the sets of *close*- and *far*- pairs respectively.

By using A_{wr} and A_{br} and incorporating a weight for each pair, the modified within-class and between-class scatter matrices for LDAr can be given as follows:

$$S_{wr} = \frac{1}{n_w} \sum_{(i,j) \in A_{wr}} f(y_i - y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$$

$$S_{br} = \frac{1}{n_b} \sum_{(i,j) \in A_{br}} f(y_i - y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \quad (24)$$

Here, the function $f(\cdot)$ is a weight function which takes on a positive value. The discussion on $f(\cdot)$ will follow later.

Using this modified scatter matrices, Fisher's criterion (17) can be rewritten for regression problems as

$$W = \underset{W}{\operatorname{argmax}} \frac{|W^T S_{br} W|}{|W^T S_{wr} W|} \quad (25)$$

Eqs. (23)–(25) constitute the LDAr problem and as before, maximizing the above Fisher's criterion (25) is equivalent to solving the following generalized eigenvalue problem:

$$S_{br}\boldsymbol{w}_k = \lambda_k S_{wr}\boldsymbol{w}_k, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \quad (26)$$

which is again equivalent to the following eigenvalue decomposition problem:

$$S_{wr}^{-1} S_{br}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \quad (27)$$

where $\boldsymbol{w}_1$ is the most important component, $\boldsymbol{w}_2$ is the second and so on. Note that S_{br} and S_{wr} are positive semi-definite and all the eigenvalues λ_i's are non-negative.

In modifying LDA for regression problems, we could have segmented the given dataset into several virtual classes based on the target values with fixed boundaries and could have applied conventional LDA for classification problems. Although this method is simpler, the results can be highly dependent on the segmenting boundaries and the number of virtual classes. In addition, this approach cannot take the levels of similarity among different classes into account. Therefore, in LDAr, a soft boundary which differs from sample to sample is adopted by using τ .

Note that the threshold parameter τ plays an important role in setting the boundary. If τ is small, n_w becomes small while n_b becomes large and vise versa. In the limit, if $\tau = 0$, $S_{wr} = 0$ and S_{br} becomes identical to S_{yx} in WPCA. The threshold τ can be represented as a multiple of the standard deviation σ_y of target variable y such that $\tau = \alpha\sigma_y$. Typical range for α is 0.1–1.0.

Although the weight function $f(\cdot)$ can be set as a constant, i.e., $f(x) = 1$, it is reasonable to force it to take on different values for different inputs. Because $|y_i - y_j| = \tau$ sets a boundary whether the pair (i,j) should belong to A_{wr} or A_{br}, the effect of (i,j)-pair which is near this boundary can be reduced by setting $f(x) \simeq 0$ for $|x| \simeq \tau$. Typical examples of $f(\cdot)$ fulfilling this requirement are $f(x) = ||x| - \tau|$ and $f(x) = \sqrt{||x| - \tau|}$.

As LDA, LDAr is not invariant to the transformation of input features and susceptible to the scaling of any input features. Therefore, it is desirable to preprocess the given dataset by the sphering technique presented in the previous subsection.

As in WPCA, the computational complexity of LDAr can be decomposed into two parts. The first part is related to obtaining the scatter matrices shown in (24) and it is proportional to the square of the number of examples and the square of the input dimension, i.e., $C_1 = O(n^2 d^2)$. The second part is related to solving the eigenvalue decomposition problem in (26) and it is typically proportional to the cubic of the input dimension, i.e., $C_2 = O(d^3)$.

Comparing this to the complexity of LDA, because the second part is common to LDA and LDAr, we can see that LDAr is more computationally complex than LDA which requires $O(nd^2)$ operations in obtaining the scatter matrices. However, as stated in the previous subsection, for large n, a subset of samples can be selected in computing the scatter matrices to reduce the computational complexity.

3.2.2. Coping with small sample size (SSS) problem

In many real world problems, the dimension of the input space d can be so large compared to the number of available samples n that the scatter matrix S_{wr} or S_{br} in LDAr becomes singular. In this case, especially when S_{wr} is singular, taking inverse in (27) is impossible and alternative method should be devised. The same problem also arises due to an unappropriate setting of τ such that n_w becomes so small.

This kind of problems are well known as the small sample size (SSS) problem for LDA and there has been much effort to solve this problem for LDA (see [24]). Note that the SSS is also a problem for WPCA and other conventional methods such as SIR and PHD presented in the previous section especially when the dimension of the input space is too large.

Preprocessing by PCA-like transform: The first approach to resolve the SSS problem is to reduce the dimension of the input space from d to the order of n by removing the space which does not contribute to the objective function. Looking at Eq. (25), we can see that the common null space of S_{br} and S_{wr} does not contribute to the objective function and by removing this we can alleviate the SSS problem. Because S_{br} and S_{wr} are positive semi-definite, the common null space of S_{br} and S_{wr} can be removed by preprocessing the given data using PCA-like technique as follows:

1. Solve the eigenvalue problem and find W_0:
$$S_a W_0 = W_0 \Lambda_0 \quad \text{subject to } W_0^T W_0 = I \quad (28)$$
where $S_a = aS_{wr} + (1-a)S_{br}$ for any $0 < a < 1$.
2. Transform the original data by multiplying the eigenvectors $\{\boldsymbol{w}_{0i}\}_{i=1}^{l}$ whose corresponding eigenvalues are not zero.

Note that if we set $a = 2n_w/n(n-1)$, S_a becomes covariance of $\boldsymbol{x}$, i.e., $S_a = S_x = (1/n)\sum_{i=1}^{n}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$ and this process becomes exactly identical to the sphering process which serves both to suppress the effect of the underlying imbalance among the original input features and to resolve SSS problem. With this technique, we can reduce the dimension of the input space from d to $l \leq n$.

Regularization: The second approach to resolve SSS problem is normally called as a 'regularization' method. For LDA, it is well known that the null space of the within covariance matrix S_w is very important because it makes the objective function to infinity. However, because the sphering operation is not possible for zero eigenvalues, in some implementations of LDA, the null space is removed and the optimal weight vectors are searched for only in

the range space of S_w. In other implementation [25], a user has to decide whether to find optimal direction in the range space of S_w or in the null space. In this case, if the range space is selected, the sphering operation is performed on it, while the normalized weight is used if the null space is selected. Also in [25], the weighted average of the two projections are used in the classification step without justification.

The same problem also applies to LDAr in utilizing the null space of S_{wr}. And it can be solved by the following regularization methods. They try to avoid singularity of S_{wr} by adding a small quantities to the diagonal of S_{wr} as in [26]:

$$S'_{wr} = (1-\mu)S_{wr} + \frac{\mu}{d}\mathrm{tr}[S_{wr}]I_d \tag{29}$$

or as used in RDA [27]:

$$S'_{wr} = S_{wr} + \gamma I_d \tag{30}$$

Here small scalars μ and γ are regularization parameters, tr[·] is a trace of a matrix and I_d denotes d dimensional identity matrix.

These regularization methods have the effect of decreasing the large eigenvalues and increasing the small ones, thus counteracting the biased estimation of the eigenvalues. However, if d is very large, this approach is very heavy because we have to deal with $d \times d$ matrix operations.

Small variant of the above approach is used in R-LDA [24]. In this method, instead of adding a small diagonal matrix to within-covariance scatter matrix, it is added to the eigenvalue matrix and the modified eigenvalues are used in the sphering process. Unlike the conventional regularization methods, this method only focuses on the small eigenvalues and can be carried out faster.

Note that all the regularization techniques presented so far can be utilized for LDAr.

3.3. Examples

To show the effectiveness of the proposed methods, two simple examples are shown in the following.

3.3.1. Example 1: linear target

Consider we have two independent input features x_1 and x_2 which have normal distribution with zero mean and variance of 1. In addition, suppose the target output variable y has the following linear relationship with the input $\boldsymbol{x}$:

$$y = 2x_1 + x_2. \tag{31}$$

For this problem, the optimal feature is $f = 2x_1 + x_2$ which corresponds to the optimal weight vector $\boldsymbol{w}^* = [2,1]^T$.

In Fig. 1, we have plotted 1000 samples. Its contour map which connects the points that have the same y value is also drawn in slanted lines. For this empirical data, we have applied SIR, PHD, WPCA and LDAr.

Considering that the area between the neighboring slanted lines can be considered as a slice in SIR, there will be significant differences in the mean values $\overline{\boldsymbol{x}}_l$ ($l = 1,\ldots,L$) of each slice and we can expect the SIR will work well for this problem. As expected, SIR resulted in $\boldsymbol{w} = [0.89,0.45]^T$ which is very close to the optimal value $\boldsymbol{w}^*$. The number of slices was set to $L = 10$ in this case.

Regarding PHD, because y is linear to $\boldsymbol{x}$, all the elements in the Hessian matrix of this problem are zeros and we can expect the PHD will fail to this problem. As a matter of fact, for the empirical data shown in Fig. 1, PHD produced $\boldsymbol{w} = [0.88,-0.51]$ which is far from $\boldsymbol{w}^*$.

When we applied WPCA to this problem with weight function $g(x) = \sqrt{|x|}$, we could get $\boldsymbol{w} = [0.90,0.44]^T$ which is very close to the optimal value $\boldsymbol{w}^*$. Considering the similarity between S_{yx} in (20) and S_{yxx} in (11), the result may look odd.

The reason WPCA succeeds while PHD fails to this problem lies in the form of the weight function. In PHD, the weight function is just the deviation from the target mean $\overline{y}$. Therefore, the points in the lower left part in Fig. 1 will have negative weights ($y_i - \overline{y} < 0$) and the other points which are located in the upper right part will correspond to the positive weights ($y_j - \overline{y} > 0$). As a result, contributions of any two points of symmetry about the center cancel out each other in the formation of S_{yxx} and the eigenvalues of S_{yxx} will be very small resulting in the failure of PHD.

On the other hand, in WPCA, because there will be no negative weight ($g(\cdot) \geq 0$), instead of cancelling out, in computing S_{yx}, the contribution of any pair of points of symmetry about the center adds up and the eigenvalues of S_{yx} become large enough to reflect the gradient direction.

For this example, LDAr is also applied with weight function $f(x) = \sqrt{||x| - \tau|}$ and $\alpha = 0.3$. LDAr resulted in $\boldsymbol{w} = [0.89,0.45]^T$ which is very close to the optimal weight. Note that in LDAr, the scatter matrices are all positive semi-definite.

3.3.2. Example 2: quadratic target

As in *Example* 1, suppose we have two independent input features x_1 and x_2 which have normal distribution with zero mean and variance of 1. However, in this case, suppose that the target output variable y has the following quadratic relationship with the input $\boldsymbol{x}$:

$$y = 4(x_1 - 2x_2)^2 + (2x_1 + x_2)^2 \tag{32}$$

As can be seen from the equation, for a fixed y, (x_1, x_2) constitutes an ellipsoid whose major axis is in the direction of (2,1) and the minor axis is in $(-1,2)$.

If we are to extract only one feature among the set of linear combinations of input variables x_1 and x_2, the major axis is the best projection which corresponds to a feature $f = x_1 - 2x_2$, i.e., $\boldsymbol{w}^* = [1,-2]^T$.

For this problem, we have generated 1000 samples as shown in Fig. 2 and applied SIR, PHD, WPCA and LDAr. In the figure, the ellipses are the contour map which connects the points that have the same y value.

As was expected, SIR does not work out for this problem because all the mean values of different slices are near zero and random direction which are highly dependent on a specific data will be chosen. For example, for the empirical data shown in Fig. 2, SIR with $L = 10$ extracted the first weight vector $\boldsymbol{w} = [-0.84,0.52]^T$ which is far from the optimal value $\boldsymbol{w}^* = [1,-2]^T$.

Unlike SIR, PHD works well for this problem because y is quadratic to $\boldsymbol{x}$ and the principal Hessian directions are easily calculated. Calculating the Hessian matrix, it becomes $H = \left[\begin{smallmatrix} 16 & -12 \\ -12 & 34 \end{smallmatrix}\right]$ and the principle Hessian direction is $[1, -2]^T$ as expected. For the empirical data shown in Fig. 2, PHD algorithm resulted in $\boldsymbol{w} = [0.44,-0.90]^T$ which is very close to the optimal value.

WPCA also works well for this example. Consider a random sample point $\boldsymbol{x}_a$ and two other samples $\boldsymbol{x}_b$ and $\boldsymbol{x}_c$ located on the same distance from $\boldsymbol{x}_a$ but in the directions of major axis and minor axis respectively. In this case, the contribution of $\boldsymbol{x}_a - \boldsymbol{x}_c$ to the S_{yx} in (20) is much more than that of $\boldsymbol{x}_a - \boldsymbol{x}_b$ because the y value in $\boldsymbol{x}_c$ will be much more different from that in $\boldsymbol{x}_a$ than the one in $\boldsymbol{x}_b$ from that in $\boldsymbol{x}_a$. Averaging out these effects for all the samples, the eigenvector corresponding to the maximum eigenvalue of S_{yx} will be the direction of minor axis and it will be chosen as the best feature. For the empirical data shown in Fig. 2, we applied WPCA with weight function $g(x) = \sqrt{|x|}$ and could get $\boldsymbol{w} = [0.44,-0.90]^T$ as in PHD.

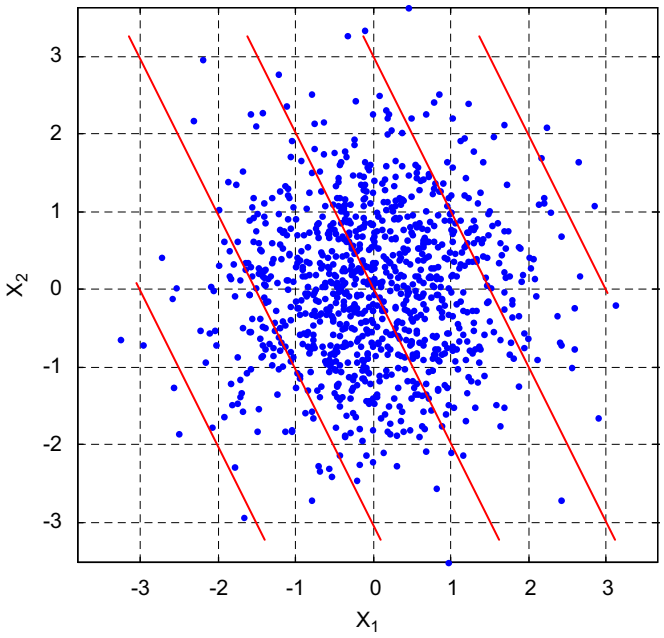

Fig. 1. Example 1: 1000 random points drawn from $N(0, I_2)$. The slanted lines are the contour map which connects the points that have the same y value.

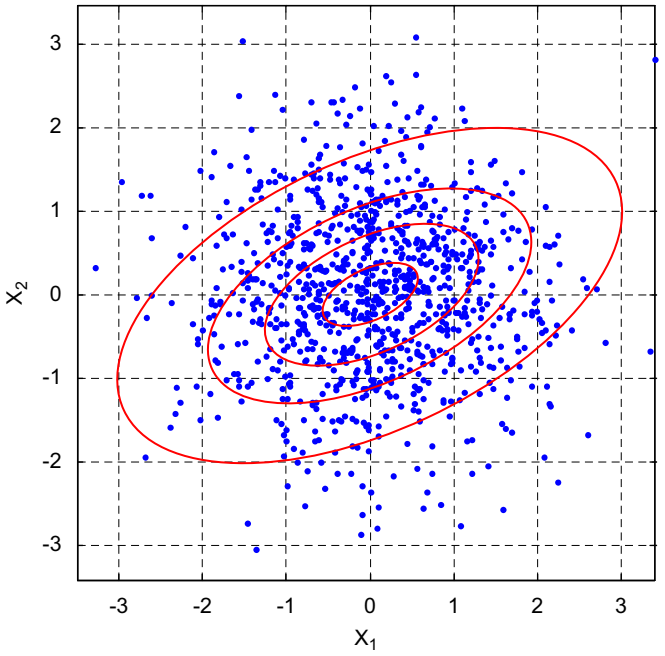

Fig. 2. Example 2: 1000 random points drawn from $N(0, I_2)$. The ellipses are the contour map which connects the points that have the same y value.

For this example, LDAr is also applied with weight function $f(x)=\sqrt{||x|-\tau|}$ and $\alpha=0.3$. LDAr resulted in $\boldsymbol{w}=[0.44,-0.90]^T$ which is the same value as in PHD and WPCA.

3.4. Sensitivity to parameters

In this part, we test the sensitivity of the proposed algorithms to model parameters and the sub-sampling scheme is checked to confirm that for large number of samples, sub-sampling can be used to speed up the process without degrading the performance. *Examples* 1 and 2 in the previous subsection are used as datasets. All the tests were performed on Pentium III processor with MATLAB.

3.4.1. WPCA

The only undetermined parameter in WPCA is the weight function $g(\cdot)$. In Section 3.1, we recommend the readers to use $g(x)=|x|$ or $g(x)=\sqrt{|x|}$. For *Examples* 1 and 2, in addition to these two weight functions, $g(x)=x^2$ was also tested and the angles between the optimal weight $\boldsymbol{w}^*$ and the weight $\boldsymbol{w}$ obtained by WPCA were measured in degree. Table 1 shows the angles. In both the examples, we can see that the performance of WPCA is rather good and does not much depend on the choice of weight function.

For these examples, we also tested the effect of subsampling and show the result in Table 2. We randomly selected 20, 50, 100, 200 and 400 samples from 1000 original data for both examples 20 times and show the average angles between $\boldsymbol{w}_{WPCA}$ and $\boldsymbol{w}^*$ and standard deviations. The average elapsed times are also provided in the table. For all the tests, the weight function $g(x)=\sqrt{|x|}$ was used.

For both examples, as the number of samples decreases, the deviation of $\boldsymbol{w}_{WPCA}$ from the optimal weight $\boldsymbol{w}^*$ increases, while the elapsed time decreases much. If we permit 5° error from the optimal weight, over 50 samples are acceptable for *Ex.* 1 while more than 200 samples are needed for *Ex.* 2. This shows that the easier problem *Ex.* 1 needs less samples than the more difficult problem *Ex.* 2.

3.4.2. LDAr

In LDAr, there are two user determined parameters. The first one is the weight function $f(\cdot)$ and the second is the threshold parameter τ. Table 3 shows various experimental results obtained by varying both the parameters simultaneously. In Table 3, weight functions $f(x)=1$, $f(x)=\sqrt{||x|-\tau|}$ and $f(x)=||x|-\tau|$ was tested for $\tau=\alpha\sigma_y$ with $\alpha\in\{0.1,0.2,\ldots,1.0\}$. In the table, we can see that in all the cases, the performance is good and does not depend much on the parameters.

We also tested the effect of subsampling in LDAr as we did for WPCA. In Table 4, out of 1000 original data, 20, 50, 100, 200 and 400 samples were randomly selected 20 times for both examples and the average angles between $\boldsymbol{w}_{LDAr}$ and $\boldsymbol{w}^*$ and standard deviations are presented. The average elapsed times are also provided in the table. For all the tests, the weight function $f(x)=\sqrt{||x|-\tau|}$ was used with $\tau=0.3\sigma_y$.

From the table, we can see that as the number of samples decreases, the elapsed time decreases much while the performance degrades a bit. If we permit 5° deviation from the optimal weight $\boldsymbol{w}^*$, even 20 samples are enough for *Ex.* 1 while at least 100 samples are needed for *Ex.* 2. Comparing the results with those in Table 2, we can see that the performance of LDAr is better than that of WPCA, especially when the number of samples is small.

4. Experimental results

In this section, the proposed algorithms are applied to several regression problems and the performances of the proposed algorithms are compared to those of other conventional methods.

As regression systems, the weighted 5 *nearest neighborhood* (5NN) regression [28] as well as the *Gaussian process regression* (GPR) [16] were used.

In the weighted version of the 5 nearest neighborhood regression, the estimation of the target variable $\hat{t}(\boldsymbol{z})$ with input variables $\boldsymbol{z}$ is obtained as follows:

$$\hat{t}(\boldsymbol{z})=\frac{1}{\sum_{i\in B(\boldsymbol{z})}q(\boldsymbol{z},\boldsymbol{z}_i)}\sum_{i\in B(\boldsymbol{z})}q(\boldsymbol{z},\boldsymbol{z}_i)t_i \tag{33}$$

Here, $B(\boldsymbol{z})$ is the set of indices of 5 nearest neighbors of $\boldsymbol{z}$ in the training set and $q(\boldsymbol{z},\boldsymbol{z}_i)$ is a weight function which was set $q(\boldsymbol{z},\boldsymbol{z}_i)=1/(1+\sqrt{\|\boldsymbol{z}-\boldsymbol{z}_i\|})$.

The basic prediction model of GPR is

$$\hat{t}(\boldsymbol{z})=\boldsymbol{k}_*^T(K+\sigma_n^2)^{-1}\boldsymbol{t} \tag{34}$$

Here, $\boldsymbol{k}_*$ denote the n-dimensional vector of covariances between the test point $\boldsymbol{z}$ and n training points, the kernel matrix K is an $n\times n$ matrix whose (i,j) element is the covariance between the i-th training point and the j-th training point, σ_n^2 is the noise variance, and $\boldsymbol{t}$ is the n-dimensional target vector of the training points.

In our experiments, independent covariance function (i.e., white noise) was added to squared exponential covariance function with adaptive relevance determination (ARD) to give the composite covariance function [16].

Table 1
Angle between $\boldsymbol{w}_{WPCA}$ and the optimal weight $\boldsymbol{w}^*$ (deg).

	$g(x)=\sqrt{x}$	$g(x)=\|x\|$	$g(x)=x^2$
Ex. 1	0.48	0.63	0.88
Ex. 2	1.20	1.10	0.86

For all the experiments, the weight function $g(x)=\sqrt{|x|}$ was used for WPCA. For LDAr, $f(x)=\sqrt{\|x|-\tau|}$ and α was set to 0.3. In SIR, the number of slices was set to be $L=15$.

4.1. Artificial problems

4.1.1. Linear case

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. Tenfold cross-validation was applied to report the performances of various feature extraction methods such as PCA, MLR, GPL, SIR, PHD, WPCA and LDAr in Table 5. As regression systems, weighted 5 nearest neighborhood regressor in (33) and GPR are used. The performances are the root mean square (rms) errors on the test data. The numbers in the parentheses are the standard deviations.

From the table, we can see that when the number of extracted features is one, all the feature extraction methods except PCA, GPL and PHD perform well and resulted almost the same rms around 0.16 for 5NN and 0.11 for GPR. GPL was better than PCA and PHD, but it resulted in poorer performance than the other methods.

Table 2
Average angle between $\boldsymbol{w}_{WPCA}$ and the optimal weight $\boldsymbol{w}^*$ (deg) and elapsed time (s) for various numbers of samples (average of 20 random tests).

	No. samples	20	50	100	200	400	1000
Ex. 1	Avg. angle (deg)	11.37 (11.19)	4.27 (3.49)	2.44 (1.65)	1.44 (1.33)	0.66 (0.66)	0.48 (0.00)
	Elap. time (s)	0.01	0.02	0.09	0.35	1.42	8.87
Ex. 2	Avg. angle (deg)	24.36 (15.43)	12.54 (11.60)	7.61 (4.46)	4.10 (3.19)	2.31 (1.72)	1.20 (0.00)
	Elap. time (s)	0.00	0.02	0.08	0.35	1.42	8.87

The numbers in the parentheses are the standard deviations.

Table 3
Angle between $\boldsymbol{w}_{LDAr}$ and the optimal weight $\boldsymbol{w}^*$ (deg).

	α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Ex. 1	$f(x)=1$	0.01	0.01	0.04	0.08	0.08	0.09	0.10	0.14	0.14	0.17
	$f(x)=\sqrt{x}$	0.01	0.01	0.02	0.05	0.07	0.08	0.10	0.12	0.14	0.16
	$f(x)=x$	0.01	0.01	0.01	0.03	0.06	0.07	0.09	0.11	0.12	0.14
Ex. 2	$f(x)=1$	1.83	1.75	1.65	1.72	1.73	1.51	1.65	1.97	2.03	1.86
	$f(x)=\sqrt{x}$	1.73	1.70	1.64	1.63	1.62	1.54	1.56	1.63	1.66	1.62
	$f(x)=x$	1.59	1.60	1.57	1.53	1.51	1.47	1.44	1.44	1.45	1.44

Table 4
Average angle between $\boldsymbol{w}_{LDAr}$ and the optimal weight $\boldsymbol{w}^*$ (deg) and elapsed time (s) for various numbers of samples (average of 20 random tests).

	No. samples	20	50	100	200	400	1000
Ex. 1	Avg. angle (deg)	0.97 (0.84)	0.28 (0.17)	0.14 (0.10)	0.10 (0.09)	0.03 (0.03)	0.02 (0.00)
	Elap. time (s)	0.01	0.02	0.09	0.36	1.45	9.07
Ex. 2	Avg. angle (deg)	19.37 (12.60)	10.31 (9.37)	4.63 (2.87)	3.32 (2.78)	2.44 (1.34)	1.64 (0.00)
	Elap. time (s)	0.01	0.02	0.09	0.37	1.45	9.08

The numbers in the parentheses are the standard deviations.

Note that the MLR is the optimal in the least square sense, but the LDAr performed slightly better than MLR when 5NN was used as a regressor. This is due to the fact that the reported rms error is on the test data. Actually, on the training data, MLR performed the best but the performance differences between that of MLR and those of SIR, WPCA and LDAr were not significant. The reason PHD performs poor can be explained by the fact that there is no Hessian direction because the target value is linear to the input features as in *Example* 1. As the number of extracted features increased, the performances of all the other methods except PCA

Table 5
Performance for the simple linear dataset (rms error).

No. of features		1	2	3	4	5
Original	5NN	–	–	–	–	1.09 (0.07)
	GPR	–	–	–	–	**0.10 (0.01)**
PCA	5NN	3.03 (0.70)	2.46 (0.63)	2.31 (0.69)	1.66 (0.71)	1.09 (0.07)
	GPR	2.74 (0.29)	2.60 (0.24)	2.57 (0.23)	2.55(0.28)	**0.10 (0.01)**
MLR	5NN	0.16 (0.08)	–	–	–	–
	GPR	**0.10(0.01)**	–	–	–	–
GPL	5NN	0.36 (0.03)	–	–	–	–
	GPR	0.32 (0.05)	–	–	–	–
SIR	5NN	0.16 (0.07)	0.43 (0.07)	0.69 (0.06)	0.90 (0.05)	1.11 (0.04)
	GPR	0.11 (0.01)	0.11 (0.01)	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**
PHD	5NN	3.01 (0.87)	2.67 (0.80)	2.05 (0.62)	1.69 (0.57)	1.11 (0.04)
	GPR	2.63 (0.34)	2.55 (0.33)	2.14 (0.58)	1.10(0.87)	**0.10 (0.01)**
WPCA	5NN	0.18 (0.06)	0.44 (0.07)	0.70 (0.06)	0.92 (0.05)	1.11(0.04)
	GPR	0.11 (0.01)	0.11 (0.01)	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**
LDAr	5NN	**0.15 (0.08)**	**0.17 (0.07)**	**0.18 (0.07)**	**0.20 (0.06)**	**0.20 (0.06)**
	GPR	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**

Averages of 10-fold CV. Numbers in the parentheses are the standard deviations.

Table 6
Performance for the nonlinear dataset (rms error).

No. of features		1	2	3	4	5
Original	5NN	–	–	–	–	0.46 (0.02)
	GPR	–	–	–	–	**0.09 (0.01)**
PCA	5NN	0.77 (0.02)	0.76 (0.05)	0.61 (0.16)	0.51 (0.14)	0.46 (0.02)
	GPR	0.70 (0.01)	0.70 (0.01)	0.68 (0.03)	0.60 (0.08)	0.13 (0.02)
MLR	5NN	0.52 (0.07)	–	–	–	–
	GPR	0.53(0.05)	–	–	–	–
GPL	5NN	0.54 (0.06)	–	–	–	–
	GPR	0.54(0.05)	–	–	–	–
SIR	5NN	0.56 (0.09)	0.50 (0.08)	0.48 (0.09)	0.44 (0.06)	0.46 (0.01)
	GPR	0.57 (0.05)	0.56 (0.05)	0.45 (0.08)	0.37 (0.12)	**0.09 (0.01)**
PHD	5NN	0.72 (0.14)	0.66 (0.19)	0.59 (0.20)	0.54 (0.15)	0.46 (0.01)
	GPR	0.69 (0.02)	0.67 (0.02)	0.63 (0.04)	0.50 (0.15)	0.16 (0.04)
WPCA	5NN	0.48 (0.07)	0.48 (0.08)	0.45 (0.08)	0.43 (0.05)	0.46 (0.01)
	GPR	0.58 (0.05)	0.53 (0.06)	0.48 (0.07)	0.46 (0.08)	0.10 (0.02)
LDAr	5NN	**0.47 (0.10)**	**0.44 (0.11)**	**0.37 (0.07)**	**0.38 (0.03)**	**0.44 (0.02)**
	GPR	**0.48 (0.06)**	**0.47 (0.06)**	**0.43 (0.08)**	**0.32 (0.11)**	0.10 (0.01)

Averages of 10-fold CV. Numbers in the parentheses are the standard deviations.

Table 7
Performance for the Housing dataset (rms error).

No. of features	1	3	5	7	9	11	13
Original	–	–	–	–	–	–	4.02 (0.48)
PCA	7.98 (0.82)	4.44 (0.63)	4.10 (0.55)	4.03 (0.50)	3.98 (0.57)	3.90 (0.50)	4.02 (0.48)
MLR	**4.04 (0.50)**	–	–	–	–	–	–
GPL	6.82 (1.24)	–	–	–	–	–	–
SIR	4.26 (0.56)	4.15 (0.48)	3.66 (0.53)	3.77 (0.58)	3.98 (0.67)	4.01 (0.60)	4.17 (0.66)
PHD	8.25 (0.81)	5.16 (0.77)	4.57 (0.39)	4.32 (0.72)	4.23 (0.61)	4.19 (0.60)	4.17 (0.66)
WPCA	4.68 (0.51)	4.18 (0.66)	3.89 (0.59)	3.78 (0.70)	4.06 (0.60)	4.08 (0.59)	4.17 (0.66)
LDAr	4.19 (0.64)	**3.98 (0.61)**	**3.60 (0.73)**	**3.55 (0.60)**	**3.48 (0.67)**	**3.49 (0.66)**	**3.52 (0.67)**

Averages of 10 experiments. Numbers in the parentheses are the standard deviations.

Table 8
Performance for the orange juice dataset (rms error).

No. of features	1	3	5	7	9	11	13	700
Original	–	–	–	–	–	–	–	8.92
PCA	9.89	9.38	9.11	9.10	8.92	8.91	8.91	–
MLR	7.46	–	–	–	–	–	–	–
GPL	9.04	–	–	–	–	–	–	–
SIR	9.32	9.38	9.15	8.91	8.92	8.93	8.92	–
PHD	10.38	9.20	9.36	9.05	9.20	8.93	8.92	–
WPCA	9.61	9.38	9.09	9.09	8.91	8.91	8.91	–
LDAr	**6.39**	**6.83**	**6.85**	**6.52**	**6.15**	**6.41**	**6.54**	–

and PHD become worse while those of PCA and PHD become better when 5NN was used. On the other hand, when GPR was used, the performances were rather a constant and did not depend on the number of extracted features. This is due to the effect of ARD in GPR which estimates the real covariance matrix while in 5NN Euclidian distance (i.e., identity covariance matrix) is used.

From the table, we can infer that all the feature extraction methods except PCA, GPL and PHD extracted a near best feature as the first feature and the added features from the second to the fifth do not contribute to the enhancement of the regression performance. On the other hand, PCA and PHD fail to extract a good feature as the first one and the added information in the additional features contributes to the performance enhancement of the regression system.

Note that when 5NN is used, the performance of LDAr does not degrade much compared to those of SIR and WPCA as the number of extracted features increases and regardless of the number of extracted features, LDAr performs the best. This shows that the LDAr is better fitted to the nearest neighbor regressor.

4.1.2. Nonlinear case

Suppose we have five independent input features $x_1 \sim x_5$ which have normal distribution with zero mean and variance of 1. Furthermore, 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. Tenfold cross-validation was applied to this dataset and the rms errors of various feature extraction methods on the test data are reported in Table 6. As regression systems, weighted 5 nearest neighborhood regressor in (33) and GPR are used. The numbers in the parentheses are the standard deviation.

As can be seen from the table, when only one feature is extracted, LDAr and WPCA outperform other methods for 5NN. Regarding GPR, when the number of extracted features is one, LDAr was the best. As the number of extracted features is increased, the WPCA and SIR show the same performance trends. Note that the rms error of LDAr is the smallest for most of the cases as in the previous problem.

4.2. Real world datasets

4.2.1. Housing—Boston

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

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 7.

As in the previous experiments, weighted 5 nearest neighborhood regressor is used as a regression system. In the table, the numbers in the parentheses are the standard deviation of 10 experiments.

From the table, we can see that the LDAr is better than other methods, except for the case when only one feature is extracted where the MLR performed the best. This is somewhat expected because MLR is optimal in the sense of least squared error. However, the performance difference is not much even for the case when the number of extracted features is one. For this problem, SIR is slightly better than WPCA for all the numbers of extracted features.

4.2.2. Orange juice

Orange juice dataset was obtained from the UCL machine learning database [30], which 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 features. The target is a continuous variable which corresponds to the level of saccharose.

As can be seen, this problem is a typical example of SSS problem whose input dimension d ($=700$) is much larger than the number of training examples n ($=150$). To resolve this SSS problem, for all the feature extraction methods except PCA, we have preprocessed the dataset with PCA and reduced the dimension of input space into 149 ($=n-1$). In addition, for LDAr, we have applied the regularization technique in (30) with $\gamma = 0.01$ after applying PCA.

Table 8 shows the performances of various feature extraction methods on the test dataset. Weighted 5 nearest neighborhood regressor is used as a regression system.

For this problem, the performance enhancement of WPCA over PCA was not significant. On the other hand, LDAr performed far better than the other methods regardless of the number of extracted features. The best performance was reported by LDAr when nine features were extracted.

Table 9
Performance for the SARCOS robot arm dataset (rms error).

Target	1	2	3	4	5	6	7
Original	9.79 (21)	5.92 (21)	3.49 (21)	4.51 (21)	0.37 (21)	0.70 (21)	0.86 (21)
PCA	10.21 (10)	6.21 (10)	3.65 (10)	5.29 (10)	0.39 (10)	0.73 (10)	1.01 (10)
MLR	6.18 (1)	5.23 (1)	3.36 (1)	3.42 (1)	0.40 (1)	0.94 (1)	0.72 (1)
GPL	15.28 (1)	8.65 (1)	5.14 (1)	6.58 (1)	0.59 (1)	1.20 (1)	1.29 (1)
SIR	6.55 (1)	5.01 (3)	3.22 (3)	3.50 (3)	0.37 (5)	0.69 (10)	0.71 (3)
PHD	10.20 (10)	6.33 (10)	4.13 (10)	4.55 (10)	0.40 (10)	0.81 (10)	0.84 (10)
WPCA	8.07 (5)	5.54 (5)	3.56 (5)	3.59 (5)	0.39 (10)	0.70 (10)	0.74 (3)
LDAr	**5.79 (10)**	**4.44 (10)**	**2.73 (10)**	**2.78 (10)**	**0.33 (10)**	**0.66 (10)**	**0.58 (10)**

The numbers in the parentheses are the best number of extracted features.

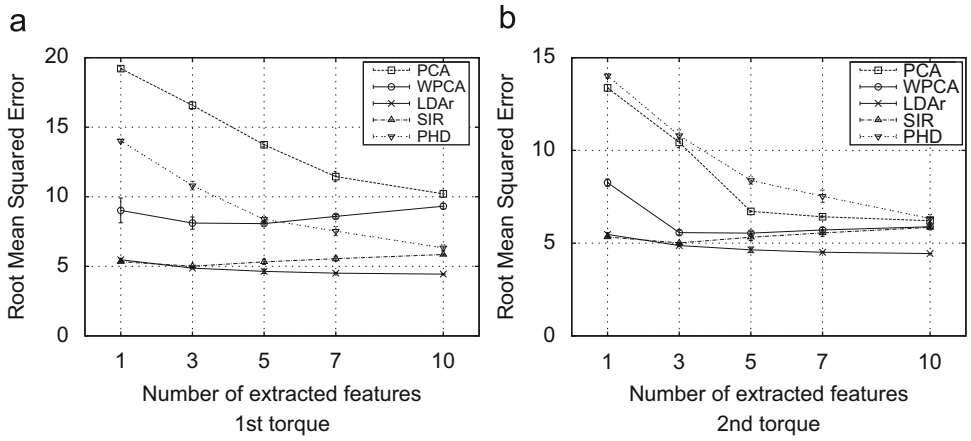

Fig. 3. RMS error for the first two target variables of SARCOS robot arm dataset.

4.2.3. SARCOS robot arm

The dataset is to learn the inverse dynamics of a seven degrees-of-freedom SARCOS anthropomorphic robot arm. It consists of 21 input features (7 joint positions, 7 velocities, 7 joint accelerations) and 7 output variables (the corresponding 7 joint torques). The dataset has previously been used to study regression algorithms [16,31]. There are 48,933 input–output pairs in the dataset, of which 44,484 were used as a training set and the remaining 4,449 were used as a test set in [16,31].

In our experiment, we test various feature extraction methods for each of the seven output variables separately. To reduce the computational complexity, we randomly selected 1000 examples from the 44,484 training set 20 times and report the performances on the 4449 test data in Table 9 and Fig. 3. As a regression system, weighted 5 nearest neighborhood regressor was used in all the tests. In the figure, due to space limitation, only the rms errors of the first two target variables with PCA, SIR, PHD, WPCA and LDAr are shown. For each feature extraction methods, 1, 3, 5, 7 and 10 features were used for the regression. The standard deviations were also drawn for each of the points, however in most cases, they are very small and difficult to be distinguished in the figure.

In both Fig. 3(a) and (b), when the number of extracted features is one, the performances of SIR and LDAr were better than those of other methods. However, as the number of features increases, the RMS error of LDAr drops slowly while that of SIR increases, especially from the case of five extracted features. These trends also hold for other target variables not shown in the figure. The performance of WPCA was better than PHD in many cases but sometimes it performed poorer than PHD. On the other hand, WPCA consistently outperformed PCA in all the cases.

In the table, we show the best performances of various feature extraction methods. The best number of extracted features is also indicated in the table. For PCA, SIR, PHD, WPCA and LDAr the 1, 3, 5, 7 and 10 extracted features were used for the weighted 5NN regressor and the best performance was shown. Note that MLR and GPL can extract only one feature. From the table, we can see that the best performance of LDAr was better than those of other methods for all the seven target variables. MLR and SIR performed relatively well and PCA, PHD and GPL showed poor performances.

5. Conclusions

In this paper, we have proposed two new methods for linear feature extraction for regression problems. The first one is WPCA which is closely related to the conventional PCA but unlike PCA, it incorporates target information in the optimization process and try to find a set of linear transforms that maximizes the distances between points with large differences in target values. The second one is a regressional version of LDA which is very popular for classification problems.

We have applied the proposed methods to several regression problems and compared the performance with the conventional feature extraction methods.

The two examples in Section 3 show the advantage of WPCA and LDAr compared to the conventional methods such as SIR and PHD. Both methods show good performance on the two examples while PHD fails to the problem where the target is a linear combination of input features and SIR fails to the problem where the target is a quadratic function to the input variables.

On the real world problems shown in Section 4, although the performance of WPCA was better than that of PCA, it shows little or no performance enhancement over other conventional methods such as SIR or PHD. On the other hand, for all the problems, LDAr outperforms the other methods. This shows the ability of LDAr to merge the samples with similar target values in

the transformed space helps improving the performance of the regression system.

The computational complexity of both methods is proportional to the square of the number of the examples but this can be reduced to a manageable quantity for a large dataset if appropriate sample selection is utilized.

Acknowledgments

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-313-D00942).

This work was partially supported by the Ajou University under Research Grant 20083770.

References

[1] P. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice Hall, London, 1982.
[2] K. Cios, W. Pedrycz, R. Swiniarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic Publishers, 1998 (Chapter 9).
[3] I. Joliffe, Principal Component Analysis, Springer-Verlag, 1986.
[4] A. Bell, T. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Computation 7 (6) (1995) 1129–1159.
[5] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed., Academic Press, 1990.
[6] D. Cai, X. He, J. Han, Spectral regression: a unified subspace learning framework for content-based image retrieval, in: ACM Multimedia, 2007.
[7] T. Okada, S. Tomita, An optimal orthonormal system for discriminant analysis, Pattern Recognition 18 (2) (1985) 139–144.
[8] M. Loog, R. Duin, Linear dimensionality reduction via a heteroscedastic extension of lda: The Chernoff criterion, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2004) 732–739.
[9] D. Zhou, X. Yang, Face recognition using direct-weighted lda, in: Lecture Notes in Computer Science, vol. 3157, 2004, pp. 760–768.
[10] D. Tao, X. Li, X. Wu, S. Maybank, Geometric mean for subspace selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 260–274.
[11] N. Kwak, C.-H. Choi, Feature extraction based on ica for binary classification problems, IEEE Transactions on Knowledge and Data Engineering 15 (6) (2003) 1374–1388.
[12] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighborhood components analysis, NIPS. URL: 〈http://www.cs.utoronto.ca/~roweis/papers/ncanips.pdf〉.
[13] D. Tao, X. Li, W. Hu, S. Maybank, X. Wu, Supervised tensor learning, Knowledge and Information Systems 13 (1) (2007) 1–42.
[14] D. Tao, X. Li, X. Wu, S. Maybank, General tensor discriminant analysis and Gabor features for gait recognition, IEEE Transactions on Pattern Recognition and Machine Intelligence 29 (10) (2007) 1700–1715.
[15] X. Li, S. Lin, S. Yan, D. Xu, Discriminant locally linear embedding with high order tensor data, IEEE Transactions on Systems, Man and Cybernetics—Part B 38 (2) (2008) 342–352.
[16] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, first ed, The MIT Press, 2006.
[17] S. Weisberg, Applied Linear Regression, second ed., John Wiley, New York, 1985, p. 324 (Chapter 3).
[18] M. Loog, Supervised Dimensionality Reduction and Contextual Pattern Recognition in Medical Image Processing, Ponsen & Looijen, Wageningen, The Netherlands, 2004 (Chapter 3).
[19] K.C. Li, Sliced inverse regression for dimension reduction (with discussion), Journal of the American Statistical Association 86 (1991) 316–342.
[20] K.C. Li, On principal Hessian directions for data visualization and dimension reduction: another application of Stein's lemma, Journal of the American Statistical Association 87 (1992) 1025–1039.
[21] N. Kwak, C. Kim, Dimensionality reduction based on ica for regression problems, in: Proceedings of the International Conference on Artificial Neural Networks, 2006, pp. 1–10.
[22] C. Stein, Estimation of the mean of a multivariate normal distribution, The Annals of Statistics 9 (1981) 1135–1151.
[23] J. Ye, Least squares linear discriminant analysis, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 1087–1093.
[24] 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.
[25] 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 Transactions on Pattern Analysis and Machine Intelligence 27 (2) (2005) 230–244.
[26] X.S. Zhou, T.S. Huang, Small sample learning during multimedia retrieval using biasmap, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, Hawaii, USA, 2001, pp. 11–17.
[27] J. Friedman, Regularized discriminant analysis, Journal of American Statistics Association 84 (1989) 165–175.
[28] E. Narayada, On estimating regression, Theory of Probability and its Applications 9 (1964) 141–142.
[29] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉, 1998.
[30] M. Meurens, Orange juice data 〈http://www.dice.ucl.ac.be/mlg/DataBases/ORANGE_JUICE/〉.
[31] S. Vijayakumar, A. D'Souza, S. Schaal, Incremental online learning in high dimensions, Neural Computation 17 (12) (2005) 2602–2634.

Nojun Kwak received his B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, in 1997, 1999 and 2003 respectively. During his Ph.D. study, he visited Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign one year from 2001 to 2002. From 2003 to 2006, he worked for Samsung Electronics. In 2006, he joined Seoul National University as a BK21 assistant professor. Currently, he is an assistant professor at Ajou University, Suwon, Korea. His research interests include pattern recognition, neural networks, machine learning, data mining, image processing, and their applications.

Jung-Won Lee received her Ph.D. in Computer Science and Engineering from Ewha Womans University, Seoul, Korea, in 2003. She was a researcher of LG Electronics and was in internship in IBM Almaden Research Center, USA. Currently, she is an assistant professor of the Department of Electrical and Computer Engineering of Ajou University, Korea. Her areas of research include compiler, service-oriented architecture and ubiquitous computing.