Pattern Recognition ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Kernel discriminant analysis for regression problems

Nojun Kwak*

Division of Electrical and Computer Engineering, Ajou University, Suwon, South Korea

ARTICLE INFO

Article history:
Received 17 May 2011
Received in revised form
3 November 2011
Accepted 6 November 2011

Keywords:
Regression
Feature extraction
Dimensionality reduction
Discriminant analysis
Kernel trick
KDAr

ABSTRACT

In this paper, we propose a nonlinear feature extraction method for regression problems to reduce the dimensionality of the input space. Previously, a feature extraction method LDAr, a regressional version of the linear discriminant analysis, was proposed. In this paper, LDAr is generalized to a nonlinear discriminant analysis by using the so-called kernel trick. The basic idea is to map the input space into a high-dimensional feature space where the variables are nonlinear transformations of input variables. Then we try to maximize the ratio of distances of samples with large differences in the target value and those with small differences in the target value in the feature space. It is well known that the distribution of face images, under a perceivable variation in translation, rotation, and scaling, is highly nonlinear and the face alignment problem is a complex regression problem. We have applied the proposed method to various regression problems including face alignment problems and achieved better performances than those of conventional linear feature extraction methods.

1. Introduction

In the statistics, machine learning and pattern recognition societies, regression is one of the classical problems, which tries to estimate a functional relationship between a set of sampling points taken from a input space and target values. With classification problems, regression problems are categorized as the supervised learning where a dataset consists of pairs of observations which have input objects, called as the *input variables*, and desired outputs, called as the *target variables*. On the other hand, in unsupervised learning only input variables are given to investigate the intrinsic data structure [1].

In many real world applications of learning process, such as biomedical data analysis and image processing, the large number of input variables may cause the so-called *curse of dimensionality* where overfitting easily appears and the supervised learning may be ill-posed [1]. In addition, irrelevant or redundant input variables tend to complicate the learning process, thereby resulting in a poor generalization performance [2,3]. For these reasons, it is desirable to reduce the number of input variables through dimensionality reduction techniques such as feature extraction that can improve the overall performance of the learning process [4]. Dimensionality reduction is quite desirable not only in the aspect of reducing the number of required data, but also in terms of data storage and computational complexity. It also finds applications in data visualization in unsupervised learning.

Traditionally, linear feature extraction methods such as principle component analysis (PCA) [5], independent component analysis (ICA) [6], and linear discriminant analysis (LDA) [7] have been extensively studied and successfully applied to various problems such as face recognition, image retrieval and so on [8–12]. 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, another unsupervised learning method, also leaves much room for improvement to be used for supervised learning problems.

Unlike PCA and ICA, linear discriminant analysis (LDA) [7] was originally developed for supervised learning, especially for classification problems, to find the optimal linear discriminating projections. 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 $C-1$ features, where C is the number of classes [13–15]. 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 [16].

Compared to classification problems, relatively little attention has been taken on the feature extraction for the regression problems in the machine learning society. On the other hand, in statistics, several algorithms have been developed as dimensionality reduction techniques for regression problems among which classical multivariate linear regression (MLR) [17] can be the

* Tel.: +82 31 219 2480; fax: +82 31 212 9531.
E-mail address: nojunk@ajou.ac.kr

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.11.006

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 have 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. Rasmussen et al. introduced a regressor called *Gaussian process regression* (GPR) [21] and its linear version *Gaussian process with linear regression model* (GPL) can be regarded as a linear feature extraction method [22].

In our previous work, a couple of feature extraction algorithms have been introduced for regression problems [23,22]. The first one is the extension of ICA-FX to regression problems [23] which produced relatively good performance. However, because ICA based methods are basically 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 utilizes higher order statistics which requires a lot of training samples in achieving reliable estimation of the distributions. Our second feature extraction method for regression problems is LDAr [22] which is a generalization of LDA to regression problems. Although LDA is widely used for classification problems, it cannot be directly applied for regression problems and LDAr was proposed for 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. Because LDAr involves in singular value decomposition, it is relatively faster than iterative methods such as ICA-FX.

Although the aforementioned linear subspace methods finds a compact representation of the original data when the data form a linear subspace, the distribution of some data such as face images, under a perceivable variation in viewpoint, illumination or facial expression, is highly nonlinear and complex. It is therefore reasonable that linear subspace methods for feature extraction fail to provide reliable and robust solutions to those with nonlinear variations. A number of nonlinear methods have been developed to tackle these shortcomings of the linear subspace methods and the two most attracting and popular methods are manifold learning techniques [24–29] and kernel-based approaches [30–35].

In the manifold learning techniques, the data are assumed to lie on or near a low dimensional manifold. Finding this inherent low-dimensional nonlinear embedding hidden in the original data space is the motivation of the manifold learning. In doing so, they view the local neighborhood of nonlinear manifold as linear manifold whose intrinsic spatial geometric structure is preserved by minimizing the local scatter, thus they can well preserve the adjacency similarity structure among data points [36].

Unlike manifold learning, the basic idea of kernel-based techniques is to implicitly map the observed data into potentially much higher dimensional feature space by using the kernel trick and to find a linear subspace of the feature space [30]. A good introduction of the kernel trick can be found in [37,30]. In [31], the kernel PCA (KPCA) which is an application of the kernel trick to PCA was introduced showing good performance. In [32,33], kernel fisher discriminant (KFD) and generalized discriminant analysis (GDA) were presented. Both KFD and GDA can be viewed as an extension of LDA to feature space using the kernel trick and it was shown that KFD is equivalent to applying LDA after KPCA [38].

In this paper, we extend the recently proposed LDAr [22] to nonlinear feature space using the kernel trick and show the superiority of the proposed algorithm. Because it is a generalization of linear discriminant analysis for regression problems in the feature space by using the kernel trick, we refer the algorithm as the kernel discriminant analysis for regression (KDAr).

This paper is organized as follows. In the next section, we will briefly review LDA and LDAr. In Section 3, the algorithm KDAr is derived by extending LDAr using the kernel trick. Discussion on the properties of KDAr as well as the relationship of KDAr with Laplacian eigenmap [26] is made in Section 4. Experimental results are provided in Section 5 followed by conclusions in Section 6.

2. Review of LDAr

In classification problems, we are given a dataset consisting of n input and target pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i-th input vector, and $y_i \in \{1,\ldots,C\}$ denotes the corresponding class label. Here, C is the number of classes. On the other hand, in the regression problems, the difference is that the target variables are continuous such that $\boldsymbol{y}_i \in \mathbb{R}^t$, where t is the dimension of the target vector which is typically 1.[1]

2.1. LDA

In LDA, the optimal projection vector $\boldsymbol{v} \in \mathbb{R}^d$ is searched for to maximize the following Fisher's criterion $J(\boldsymbol{v})$, which is defined as the ratio of the between-covariance matrix $S_b = (1/n)\sum_{c=1}^C n_c(\overline{\boldsymbol{x}}_c - \overline{\boldsymbol{x}})(\overline{\boldsymbol{x}}_c - \overline{\boldsymbol{x}})^T$ and the within-covariance matrix $S_w = (1/n)\sum_{c=1}^C \sum_{i \in \{j|y_j=c\}} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}_c)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_c)^T$:

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T S_b \boldsymbol{v}}{\boldsymbol{v}^T S_w \boldsymbol{v}}. \tag{1}$$

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, $\overline{\boldsymbol{x}}_c = (1/n_c)\sum_{i \in \{j|y_j=c\}} \boldsymbol{x}_i$ is the mean of the samples belonging to the class c. By successively finding m such $\boldsymbol{v}$, we can constitute the projection matrix $V \in \mathbb{R}^{d \times m}$ whose k-th column is the k-th projection vector denoted as $\boldsymbol{v}_k$. Here, m denotes the number of projection vectors to be found.

The optimization problem in (1) is equivalent to the following generalized eigenvalue decomposition (GED) problem,

$$S_b \boldsymbol{v}_k = \lambda_k S_w \boldsymbol{v}_k, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m, \tag{2}$$

where λ_k, $k \in \{1,\ldots,m\}$ is the k-th largest eigenvalue and $\boldsymbol{v}_k$ is the corresponding eigenvector.

2.2. LDAr

In LDAr, LDA is extended to regression problems [22]. Unlike classification problems which have discrete classes, in regression problems it is difficult to define between-class scatter and within-class scatter matrices because target variable y_i is continuous. This problem is resolved by introducing the notion of *soft class*, which is summarized as '*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*'. In contrast, the class information in classification problems can be viewed as the *hard class*. The followings are the modified within-class and between-class scatter matrices for LDAr:

$$S_{wr} = \frac{1}{n_w}\sum_{(i,j) \in A_w} f(y_i, y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T, \tag{3}$$

$$S_{br} = \frac{1}{n_b}\sum_{(i,j) \in A_b} f(y_i, y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T. \tag{4}$$

[1] 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.

Here, A_w and A_b are the membership sets of *close*- and *far*-sample pairs respectively which are defined by the radius ϵ as

$$A_w = \{(i,j) : |y_i - y_j| \le \epsilon,\ i,j \in \{1,\ldots,n\},\ i \ne j\},$$

$$A_b = \{(i,j) : |y_i - y_j| > \epsilon,\ i,j \in \{1,\ldots,n\},\ i \ne j\}, \tag{5}$$

where the variables n_w and n_b are the size of the set A_w and A_b respectively, i.e., $n_w = |A_w|$ and $n_b = |A_b|$, where $|\cdot|$ denotes the cardinality of a set. The function $f(\cdot,\cdot)$ is a non-negative weight function.

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

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T S_{br} \boldsymbol{v}}{\boldsymbol{v}^T S_{wr} \boldsymbol{v}}. \tag{6}$$

As stated earlier, maximizing the above Fisher's criterion is equivalent to solving the GED problem

$$S_{br}\boldsymbol{v}_k = \lambda_k S_{wr}\boldsymbol{v}_k, \quad \lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_m, \tag{7}$$

which is again equivalent to the following eigenvalue decomposition (ED) problem:

$$S_{wr}^{-1} S_{br}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k, \quad \lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_m, \tag{8}$$

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

Note that the threshold parameter ϵ plays an important role in setting the boundary. If ϵ is small, n_w becomes small while n_b becomes large and vise versa. In the limit, if $\epsilon = 0$, LDAr becomes identical to LDA with n classes which utilizes a *hard boundary*. The threshold ϵ can be represented as a multiple of the standard deviation σ_y of target variable y such that $\epsilon = \alpha\sigma_y$. In [22], $\alpha \in [0.1, 1.0]$ was recommended.

Three versions of weight function $f(\cdot,\cdot)$ were used in [22], i.e., $f(a,b) = 1$, $f(a,b) = ||a-b|-\epsilon|$, and $f(a,b) = \sqrt{||a-b|-\epsilon|}$ respectively.

3. Kernel discriminant analysis for regression problems (KDAr)

The idea of KDAr is to extend LDAr to a nonlinear version by using the so-called kernel trick [30]. Assume that we have training data consisting of n input and target pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Also assume a nonlinear mapping $\phi(\cdot)$ that maps a point in a d-dimensional input space into f-dimensional feature space, i.e.,

$$\phi : \mathbb{R}^d \to \mathbb{R}^f. \tag{9}$$

Here, the dimension of the feature space f can either be finite or infinite. Suppose for the moment, that the mapping is centered, i.e., $\sum_{i=1}^n \phi(\boldsymbol{x}_i) = 0$. This assumption will be removed in Section 3.3.

In the following subsection, the modified Fisher's criterion which was used in LDAr is extended to a high dimensional feature space.

3.1. Fisher's criterion

By replacing $\boldsymbol{x}$ in (3) with $\phi(\boldsymbol{x})$, we obtain a new within covariance matrix in the feature space as follows:

$$\begin{aligned}
S_{wr}^{\Phi} &= \frac{1}{n_w}\sum_{(i,j)\in A_{wr}} f(y_i,y_j)[\phi(\boldsymbol{x}_i)-\phi(\boldsymbol{x}_j)][\phi(\boldsymbol{x}_i)-\phi(\boldsymbol{x}_j)]^T \\
&= \frac{1}{n_w}\sum_{i=1}^n\sum_{j=1}^n [\phi(\boldsymbol{x}_i)-\phi(\boldsymbol{x}_j)] w_{ij} [\phi(\boldsymbol{x}_i)^T-\phi(\boldsymbol{x}_j)^T] \\
&= \frac{2}{n_w}\left[\sum_{i=1}^n\sum_{j=1}^n \phi(\boldsymbol{x}_i) w_{ij}\phi(\boldsymbol{x}_i)^T - \sum_{i=1}^n\sum_{j=1}^n \phi(\boldsymbol{x}_i) w_{ij}\phi(\boldsymbol{x}_j)^T\right] \\
&= \frac{2}{n_w}\left[\sum_{i=1}^n \phi(\boldsymbol{x}_i)\left(\sum_{j=1}^n w_{ij}\right)\phi(\boldsymbol{x}_i)^T - \sum_{i=1}^n\sum_{j=1}^n \phi(\boldsymbol{x}_i) w_{ij}\phi(\boldsymbol{x}_j)^T\right] \\
&= \frac{2}{n_w}\phi(X)(D_w - W_w)\phi(X)^T \\
&= \frac{2}{n_w}\phi(X) L_w \phi(X)^T,
\end{aligned} \tag{10}$$

where $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ is an input matrix, $\phi(X) = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)] \in \mathbb{R}^{f\times n}$ is a feature matrix, $W_w = [w_{ij}] \in \mathbb{R}^{n\times n}$ is a symmetric matrix whose (i,j) element is

$$w_{ij} = \begin{cases} f(y_i,y_j) & \text{if } (i,j) \in A_{wr}, \\ 0 & \text{otherwise}, \end{cases}$$

$D_w = diag(d_1^w, \ldots, d_n^w)$ is a diagonal matrix with its i-th diagonal element being $d_i^w = \sum_{j=1}^n w_{ij}$, and $L_w \triangleq D_w - W_w$ is a Laplacian matrix whose row and column sums equal zero.

Likewise, a new between class covariance matrix in the feature space becomes

$$S_{br}^{\Phi} = \frac{2}{n_b}\phi(X)(D_b - W_b)\phi(X)^T = \frac{2}{n_b}\phi(X) L_b \phi(X)^T, \tag{11}$$

where $W_b = [b_{ij}] \in \mathbb{R}^{n\times n}$ is a symmetric matrix whose (i,j) element is

$$b_{ij} = \begin{cases} f(y_i,y_j) & \text{if } (i,j) \in A_{br}, \\ 0 & \text{otherwise}, \end{cases}$$

$D_b = diag(d_1^b, \ldots, d_n^b)$ is a diagonal matrix with its i-th diagonal element being $d_i^b = \sum_{j=1}^n b_{ij}$, and $L_b \triangleq D_b - W_b$.

The matrices W_w and W_b can be regarded as edge matrices that represent the *local similarity* and the *local dissimilarity* of the samples, respectively.[2]

By replacing S_{wr} and S_{br} in (6) with S_{wr}^{Φ} and S_{br}^{Φ} respectively, we obtain the Fisher's criterion in the feature space as follows:

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T S_{br}^{\Phi} \boldsymbol{v}}{\boldsymbol{v}^T S_{wr}^{\Phi} \boldsymbol{v}}. \tag{12}$$

Here, $\boldsymbol{v} \in \mathbb{R}^f$ is a projection vector in the feature space.

However, direct calculation of $\boldsymbol{v}$ by solving the corresponding GED problem of (12) is difficult because the dimension of $\boldsymbol{v}$ is not known and furthermore it could be infinite. To resolve this problem, an alternative way of using the *kernel trick* to obtain a projection of a sample in the feature space is derived in the next subsection.

3.2. Projection in the feature space

We know that the projection vector $\boldsymbol{v}$ is a linear combination of the training samples in the feature space, i.e.,

$$\boldsymbol{v} = \sum_{i=1}^n \alpha_i \phi(\boldsymbol{x}_i) = \phi(X)\boldsymbol{\alpha} \tag{13}$$

for some $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T \in \mathbb{R}^n$.

Considering that the projection of a sample $\boldsymbol{x}$ in the feature space is obtained by the inner product of the projection vector $\boldsymbol{v}$ and the sample $\phi(\boldsymbol{x})$, the projection of the entire training data is obtained by

$$\boldsymbol{v}^T\phi(X) = \boldsymbol{\alpha}^T\phi(X)^T\phi(X) = \boldsymbol{\alpha}^T K, \tag{14}$$

[2] Note that W_w can be treated the same way as the edge matrix for the Laplacian-eigenmap [26] except that the element w_{ij} is determined by the target value.

where $K \triangleq \phi(X)^T\phi(X)$ is a centerized kernel matrix whose (i,j) element is $k(\boldsymbol{x}_i,\boldsymbol{x}_j)=\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$. Then it is easy to show that the numerator and denominator of (12) are

$$\boldsymbol{v}^T S_{br}^{\Phi}\boldsymbol{v} = \frac{2}{n_b}\boldsymbol{\alpha}^T K(D_b-W_b)K\boldsymbol{\alpha},$$

$$\boldsymbol{v}^T S_{wr}^{\Phi}\boldsymbol{v} = \frac{2}{n_w}\boldsymbol{\alpha}^T K(D_w-W_w)K\boldsymbol{\alpha}, \tag{15}$$

respectively, because K is symmetric.

Let $S_B^K = K(D_b-W_b)K$ and $S_W^K = K(D_w-W_w)K$. Then, the modified Fisher's criterion (12) becomes a function of $\boldsymbol{\alpha}$ as follows:

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T S_B^K\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T S_W^K\boldsymbol{\alpha}}. \tag{16}$$

The optimal $\boldsymbol{\alpha}$'s can be obtained by solving the following GED problem:

$$S_B^K\boldsymbol{\alpha}_k = \lambda_k S_W^K\boldsymbol{\alpha}_k, \quad \lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_m. \tag{17}$$

After the computation of $\boldsymbol{\alpha}_k$, $k=1,\ldots,m$, an m-dimensional nonlinear feature vectors $Z \in \mathbb{R}^{m\times n}$ of the training data is obtained by the inner product of the matrix $A \triangleq [\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m] \in \mathbb{R}^{n\times m}$ and the kernel matrix K, i.e., $Z=A^TK$.

Now let us consider how an arbitrary sample in the input space is mapped to the feature space. When a new sample $\boldsymbol{x}\in\mathbb{R}^d$ is presented, it is firstly mapped to the feature space by $\phi(\boldsymbol{x})$ and then projected in the feature space by the projection vector $\boldsymbol{v}$, i.e.,

$$\boldsymbol{x} \to \phi(\boldsymbol{x}) \to \boldsymbol{v}^T\phi(\boldsymbol{x}). \tag{18}$$

Since $\boldsymbol{v}=\phi(X)\boldsymbol{\alpha}$, it becomes

$$\boldsymbol{v}^T\phi(\boldsymbol{x}) = \boldsymbol{\alpha}^T\phi(X)^T\phi(\boldsymbol{x}) = \boldsymbol{\alpha}^T[k(\boldsymbol{x},\boldsymbol{x}_1),\ldots,k(\boldsymbol{x},\boldsymbol{x}_n)]^T = \boldsymbol{\alpha}^T k(\boldsymbol{x}), \tag{19}$$

where $k(\boldsymbol{x}) = [k(\boldsymbol{x},\boldsymbol{x}_1),\ldots,k(\boldsymbol{x},\boldsymbol{x}_n)]^T \in \mathbb{R}^n$. Aggregating m such projections, a nonlinear feature vector $\boldsymbol{z}\in\mathbb{R}^m$ is obtained by $\boldsymbol{z}=A^Tk(\boldsymbol{x})$.

Note that in the computation of $\boldsymbol{\alpha}$'s, the exact form of the nonlinear mapping $\phi(\cdot)$ is not needed. Instead, $\phi(\cdot)$ always appears in the form of a kernel function $k(\cdot,\cdot)$ which is defined as the inner product of $\phi(\cdot)$ to itself. Therefore, we only need to define an appropriate kernel function $k(\cdot,\cdot)$ to obtain $\boldsymbol{\alpha}$ and consequently the projection of a sample in the feature space $\boldsymbol{v}^T\phi(\boldsymbol{x})(=\boldsymbol{\alpha}^Tk(\boldsymbol{x}))$. Note also that once the kernel function $k(\cdot,\cdot)$ is defined, the exact form of the mapping $\phi(\cdot)$ cannot be derived from it in most cases.

3.3. Centerization

Until now, we have assumed that the nonlinear mapping is centered, i.e., $\sum_{i=1}^n \phi(\boldsymbol{x}_i)=0$. However, for an arbitrary kernel function $k(\cdot,\cdot)$, it is not guaranteed that the samples in the feature spaces $\phi(X)$ have zero mean. Therefore, the centerization process which is described below is essential in KDAr.

To distinguish from the centerized version of the mapping $\phi(\cdot)$, we denote the uncenterized mapping as $\varphi(\cdot)$. Likewise, the uncenterized kernel is denoted as $\kappa(\cdot,\cdot)$ in contrast to the centerized kernel $k(\cdot,\cdot)$. Furthermore, it is assumed that $\kappa(\boldsymbol{x},\boldsymbol{y}) = \varphi(\boldsymbol{x})^T\varphi(\boldsymbol{y})$.

Because the projection $\boldsymbol{v}$ in the feature space should be performed on the centered data, $\varphi(X)$ is shifted to $\phi(X)$ by subtracting its mean $\overline{\varphi}=(1/n)\sum_{i=1}^n\varphi(\boldsymbol{x}_i)=(1/n)\varphi(X)\boldsymbol{e}_n$, i.e., $\phi(X)=\varphi(X)-\overline{\varphi}\boldsymbol{e}_n^T=\varphi(X)(I_n-(1/n)\boldsymbol{e}_n\boldsymbol{e}_n^T)$, where $\boldsymbol{e}_n=[1,\ldots,1]^T\in\mathbb{R}^n$.

If we define $\mathbf{1}_n \triangleq (1/n)\boldsymbol{e}_n\boldsymbol{e}_n^T$, i.e, all the elements of $\mathbf{1}_n$ being $1/n$, then

$$\phi(X) = \varphi(X)(I_n-\mathbf{1}_n). \tag{20}$$

Consider that the kernel functions and the nonlinear mappings have the relationship $\kappa(x,y)=\varphi(x)^T\varphi(y)$ and $k(x,y)=\phi(x)^T\phi(y)$, respectively. Then, the kernel matrices $K=[k(\boldsymbol{x}_i,\boldsymbol{x}_j)]$ and $\mathcal{K}=[\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)]$ have their relationship as follows:

$$\begin{aligned} K &= \phi(X)^T\phi(X) \\ &= [\varphi(X)^T(I_n-\mathbf{1}_n)]^T[\varphi(X)^T(I_n-\mathbf{1}_n)] \\ &= (I_n-\mathbf{1}_n)\mathcal{K}(I_n-\mathbf{1}_n). \end{aligned} \tag{21}$$

Likewise for an arbitrary sample $\boldsymbol{x}$, the centerization can be obtained by

$$\begin{aligned} k(\boldsymbol{x}) &= \phi(X)^T\phi(\boldsymbol{x}) \\ &= (I_n-\mathbf{1}_n)\varphi(X)^T\left[\varphi(\boldsymbol{x})-\frac{1}{n}\varphi(X)\boldsymbol{e}_n\right] \\ &= (I_n-\mathbf{1}_n)\left[\varphi(X)^T\varphi(\boldsymbol{x})-\frac{1}{n}\varphi(X)^T\varphi(X)\boldsymbol{e}_n\right] \\ &= (I_n-\mathbf{1}_n)\left[\kappa(\boldsymbol{x})-\frac{1}{n}\mathcal{K}\boldsymbol{e}_n\right], \end{aligned} \tag{22}$$

where $\kappa(\boldsymbol{x}) = [\kappa(\boldsymbol{x},\boldsymbol{x}_1),\ldots,\kappa(\boldsymbol{x},\boldsymbol{x}_n)]^T\in\mathbb{R}^n$.

3.4. Algorithm: KDAr

Summarizing the previous subsections, the overall KDAr algorithm is as follows:

- Training:
  1. Generate an uncentered kernel matrix $\mathcal{K}\in\mathbb{R}^{n\times n}$ from the training samples.
  2. Center the kernel matrix by $K=\mathcal{K}-\mathbf{1}_n\mathcal{K}-\mathcal{K}\mathbf{1}_n+\mathbf{1}_n\mathcal{K}\mathbf{1}_n$.
  3. Generate the edge matrices W_w,W_b and the corresponding diagonal matrices D_w,D_b by using the weight function $f(y_i,y_j)$. Set $L_w=D_w-W_w$ and $L_b=D_b-W_b$.
  4. Set $S_B^K=KL_bK$ and $S_W^K=KL_wK$.
  5. Apply GED for (S_B^K,S_W^K) to find the eigenvector matrix $A=[\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m]$ whose k-th column, $\boldsymbol{\alpha}_k$, is the eigenvector corresponding to the k-th largest eigenvalue.
  6. Obtain a nonlinear feature matrix Z of the training data by $Z=A^TK$.
- Test:
  1. For a test sample $\boldsymbol{x}$, generate an uncentered kernel vector $\kappa(\boldsymbol{x})\in\mathbb{R}^n$.
  2. Center the kernel vector by $k(\boldsymbol{x})=(I_n-\mathbf{1}_n)[\kappa(\boldsymbol{x})-(1/n)\mathcal{K}\boldsymbol{e}_n]$.
  3. Obtain a nonlinear feature vector $\boldsymbol{z}$ of the test sample by $\boldsymbol{z}=A^Tk(\boldsymbol{x})$.

In KDAr, the kernel function $\kappa(\cdot,\cdot)$ plays an important role and the essential property of the kernel function is that it should be decomposed into an inner product of a mapping $\varphi(\cdot)$ to itself, i.e., $\kappa(\boldsymbol{x},\boldsymbol{y})=\varphi(\boldsymbol{x})^T\varphi(\boldsymbol{y})$. However, it is obvious that not all the functions meet this property. To be a proper kernel function, a function should meet the so-called *Mercer's condition* [30] and the two most popular kernels are the (*inhomogeneous*) *polynomial kernel* $\kappa(\boldsymbol{x},\boldsymbol{y})=(\boldsymbol{x}^T\boldsymbol{y}+c)^d$ and the *Gaussian RBF kernel* $\kappa(\boldsymbol{x},\boldsymbol{y})=\exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma)$ [30] in which c, d, and σ are the kernel parameters.

In the training of KDAr, the most time consuming step is Step 5 where the GED problem should be solved. Because the matrices S_B^K and S_W^K are $\mathbb{R}^{n\times n}$, the computational complexity of KDAr is normally $O(n^3)$. From this, we can see that the computational complexity of KDAr does not depend much on the dimension of the input space d. On the other hand, the computational complexity of linear methods such as LDA and LDAr highly depends on d. In the next section, more detailed discussion on KDAr is made.

4. Discussion on KDAr

In this section, we investigate the properties of KDAr by solving the solution of (16) in an alternative way and the relationship between KDAr and Laplacian Eigenmap [26] is made. In addition, different methods of choosing neighborhood of a sample such as ϵ-neighborhood and k-nearest neighborhood are discussed.

4.1. An alternative solution

The solution of the optimization problem (16) can be obtained in an alternative way. Let $\boldsymbol{\beta} \triangleq K\boldsymbol{\alpha}$. Then the optimization problem (16) becomes

$$J(\boldsymbol{\beta}) = \frac{\boldsymbol{\beta}^T L_b \boldsymbol{\beta}}{\boldsymbol{\beta}^T L_w \boldsymbol{\beta}} \tag{23}$$

and the corresponding GED problem becomes

$$L_b\boldsymbol{\beta} = \lambda L_w \boldsymbol{\beta}. \tag{24}$$

If we take m eigenvectors $B \triangleq [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m] \in \mathbb{R}^{n\times m}$ corresponding to the m largest eigenvalue of (24), it is easy to show that the nonlinear projection Z of the training data X is $Z = B^T$. It is important to note that the projection Z is independent of the kernel matrix K. Whatever the kernel function $\kappa(\cdot,\cdot)$ may be, the training samples are mapped to fixed positions in the feature space. Furthermore, the projection does not depend on the input space either, which only depends on the choice of the edge matrices W_w and W_b.

Before we further check this property more closely by a simple toy example, let us focus on the solution of the GED problem. In (24), both L_w and L_b are Laplacian matrices and it is easy to show that all the Laplacian matrices have a trivial eigenvector $\boldsymbol{\beta}_0 = \boldsymbol{e}_n$ corresponding to the eigenvalue of 0. With $\boldsymbol{\beta} = \boldsymbol{\beta}_0$, the Fisher's criterion $J(\boldsymbol{\beta})$ is of the form $\frac{0}{0}$ and λ in (24) can take any value, which makes the problem ill-posed. Therefore, it is necessary to remove this trivial solution from the eigenvector matrix B.

One way of doing this is to avoid 0 from the denominator of $J(\boldsymbol{\beta})$ as follows. Let $r(<n)$ be the rank of L_w.[3] Then, by the eigenvalue decomposition, it becomes $L_w = U\Lambda U^T$, where $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{n\times r}$ and $\Lambda = diag(\lambda_1, \ldots, \lambda_r) \in \mathbb{R}^{r\times r}$. Let $U' = [\boldsymbol{u}_1/\sqrt{\lambda_1}, \ldots, \boldsymbol{u}_r/\sqrt{\lambda_r}]$ and let us introduce a new variable $\boldsymbol{\beta}'$ such that $\boldsymbol{\beta} = U'\boldsymbol{\beta}'$. Then $U'^T L_w U' = I_r$ and maximizing (23) is equivalent to maximizing

$$J(\boldsymbol{\beta}') = \frac{\boldsymbol{\beta}'^T U'^T L_b U' \boldsymbol{\beta}'}{\boldsymbol{\beta}'^T \boldsymbol{\beta}'},$$

which is again equivalent to maximizing

$$J(\boldsymbol{\beta}') = \boldsymbol{\beta}'^T L_b' \boldsymbol{\beta}' \quad \text{subject to } \|\boldsymbol{\beta}'\| = 1, \tag{25}$$

where $L_b' = U'^T L_b U'$. The solution $\boldsymbol{\beta}'$ of the above optimization problem is just the eigenvector corresponding to the largest eigenvalue of L_b'. By this technique, we can successfully avoid the ill-posed condition of $\frac{0}{0}$.

Revisiting the original solution $\boldsymbol{\alpha}$, the rank of S_W^K does not exceed that of L_w because $S_W^K = KL_wK$. Furthermore, $\boldsymbol{\alpha}_0 = K^{-1}\boldsymbol{e}_n$ becomes a trivial generalized eigenvector for (S_B^K, S_W^K). To remove $\boldsymbol{\alpha}_0$ from the eigenvector matrix A, the same technique can also be applied in solving the GED problem for (S_B^K, S_W^K).

Although the kernel function does not have any influence on the projection of the training data, it plays its role for the projection of the unseen test data $\boldsymbol{x}$. Because $\boldsymbol{\beta} = K\boldsymbol{\alpha}$, it becomes $\boldsymbol{\alpha} = K^{-1}\boldsymbol{\beta}$, if K is nonsingular. Therefore, replacing $\boldsymbol{\alpha}$ with $K^{-1}\boldsymbol{\beta}$ in (19), the projection of the test data can be obtained as $\boldsymbol{z} = B^T K^{-1} k(\boldsymbol{x})$. However, this alternative approach of solving $\boldsymbol{\beta}$ instead of $\boldsymbol{\alpha}$ is not preferable in practice because taking the inverse of K is a time consuming job especially when the number of training samples is large.

[3] In most cases, r will be $r = n-1$.

4.2. Different type of membership sets

Now, let us more closely look at the property of KDAr that the nonlinear projection Z of the training data X only depends on the choice of the edge matrices W_w and W_b by using a toy example shown below. From now on, we assume that the data $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ is sorted according to their target values y's in ascending order, i.e., if $i<j$ then $y_i \le y_j$ for all $i,j \in \{1, \ldots, n\}$. In addition, to simplify the problem, let the ϵ-neighborhood membership sets (5) are modified as the k-nearest neighborhood type membership sets as follows:

$$A_w = \{(i,j) : |i-j| \le \tau,\ i,j \in \{1, \ldots, n\}\},$$

$$A_b = \{(i,j) : |i-j| > \tau,\ i,j \in \{1, \ldots, n\}\} \tag{26}$$

and let the weight function be constant, i.e., $f(a,b) = 1$.

Now consider five training samples $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5]$ are given. If we set $\tau = 1$ in (26), then the edge matrices become

$$W_w = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad W_b = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix},$$

and the corresponding Laplacian matrices become

$$L_w = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \quad L_b = \begin{bmatrix} 3 & 0 & -1 & -1 & -1 \\ 0 & 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & -1 & -1 & 0 & 3 \end{bmatrix}.$$

Solving the GED problem (25), the largest eigenvalue is $\lambda_1 = 12.09$ and the corresponding eigenvector becomes $\boldsymbol{\beta}_1 = [-0.97, -0.60, 0.00, 0.60, 0.97]^T$. This result shows that regardless of the input data X, the nonlinear mapping maps the training sample $\boldsymbol{x}_1$ with smallest target value to -0.97, $\boldsymbol{x}_2$, with the second smallest target value to -0.60 and so on. Note that as the target value increases, the projection value also increases. The four eigenvectors $\boldsymbol{\beta}_i, i = 1, \ldots, 4$ are plotted in Fig. 1 with their corresponding eigenvalues. In the figure, we can see that as the index i of the eigenvector increases, the frequency of $\boldsymbol{\beta}_i$ becomes higher.

4.3. Relationship with Laplacian eigenmap

From now on, let us further focus on the optimization function (23). As in the previous toy example, if we use a constant weight function $f(a,b) = 1$, it becomes $L_w + L_b = nI_n - \mathbf{1}_n$ because $W_w + W_b = \mathbf{1}_n$ and $D_w + D_b = nI_n$. From this, (23) becomes

$$J(\boldsymbol{\beta}) = \frac{\boldsymbol{\beta}^T (nI_n - \mathbf{1}_n - L_w)\boldsymbol{\beta}}{\boldsymbol{\beta}^T L_w \boldsymbol{\beta}} = \frac{n - (\boldsymbol{e}_n^T\boldsymbol{\beta})^2}{\boldsymbol{\beta}^T L_w \boldsymbol{\beta}} - 1 \tag{27}$$

if $\|\boldsymbol{\beta}\| = 1$.

Considering that $\boldsymbol{e}_n$ is a trivial eigenvector of L_w, all the other eigenvectors of L_w can be made to be orthogonal to $\boldsymbol{e}_n$. Therefore, $\boldsymbol{e}_n^T\boldsymbol{\beta} = 0$ if $\boldsymbol{\beta}$ is a non-trivial eigenvector of L_w. Then, we can clearly see that maximizing $J(\boldsymbol{\beta})$ in (27) is equivalent to minimizing the denominator $\boldsymbol{\beta}^T L_w \boldsymbol{\beta}$ of (27) excluding the trivial solution of

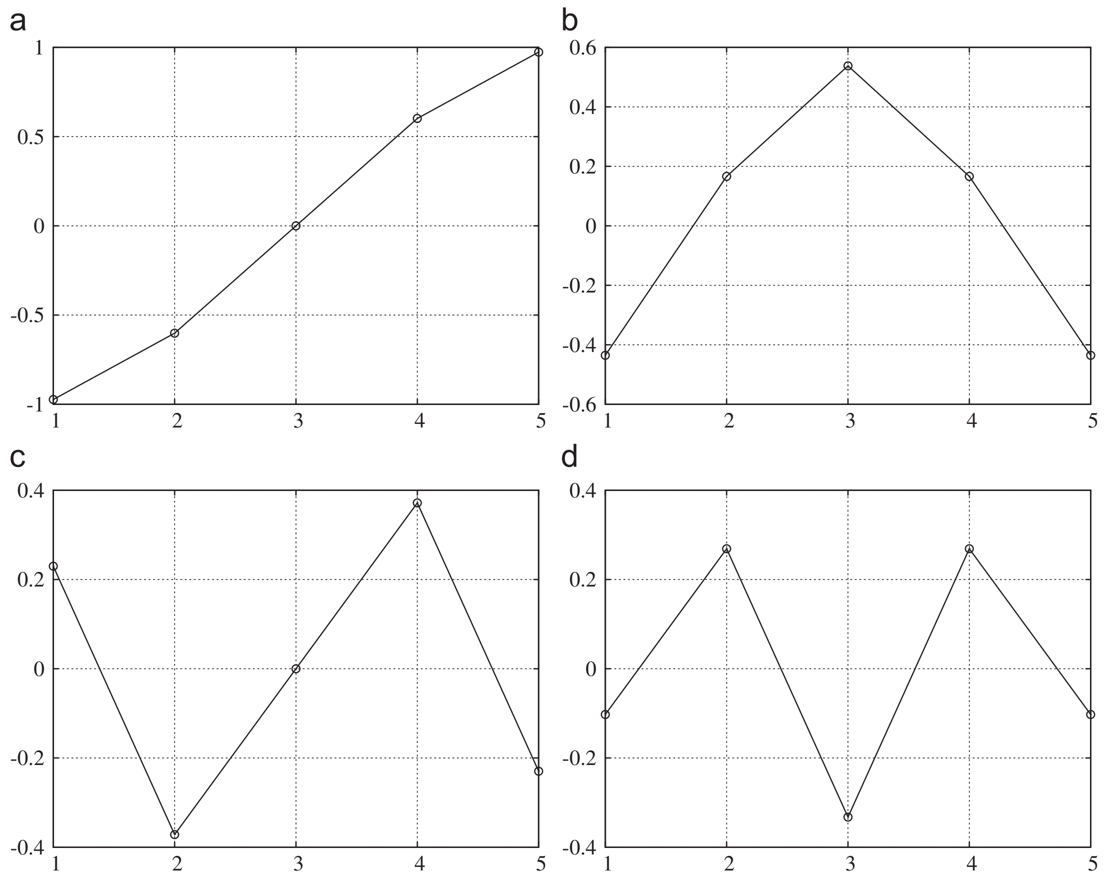

Fig. 1. Projection of the training samples ($B=[\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_4]$) for the toy problem. (a) $\boldsymbol{\beta}_1$ ($\lambda_1=12.09$), (b) $\boldsymbol{\beta}_2$ ($\lambda_2=2.62$), (c) $\boldsymbol{\beta}_3$ ($\lambda_3=0.91$), (d) $\boldsymbol{\beta}_4$ ($\lambda_4=0.38$).

$\boldsymbol{\beta}_0=\boldsymbol{e}_n$, i.e.,

$$\boldsymbol{\beta}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\boldsymbol{\beta}^T L_w \boldsymbol{\beta} \quad \text{subject to } \|\boldsymbol{\beta}\|=1,\ \boldsymbol{\beta} \not\propto \boldsymbol{e}_n.$$

This is an ED problem of L_w and the formulation is almost the same as the Laplacian eigenmap which tries to solve the following GED [26]:

$$\boldsymbol{\beta}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\boldsymbol{\beta}^T L_w \boldsymbol{\beta} \quad \text{subject to } \boldsymbol{\beta}^T D_w \boldsymbol{\beta}=1, \boldsymbol{\beta} \not\propto \boldsymbol{e}_n.$$

Note that especially when the number of training samples n is big, D_w approaches I_n, in which case, the form of the above Laplacian eigenmap problem becomes exactly the same as that of KDAr. The main difference between Laplacian eigenmap and KDAr lies in that the Laplacian matrix L_w is generated based on the target value y in KDAr, while it is from the neighborhood information of the input data X in Laplacian eigenmap. Again, note that although L_w is independent of the neighborhood information of the input data X, X play its role through the kernel matrix K by $\boldsymbol{\alpha}=K^{-1}\boldsymbol{\beta}$ and consequently in the projection of the unseen test data $\boldsymbol{x}$.

4.4. Choice of the edge matrices

In KDAr, the edge matrices W_w and W_b which are from the membership sets A_w and A_b play a key role. Therefore, the choice of membership sets are very important in the success of KDAr. Previously in this paper, two versions of membership sets have been introduced, i.e., ϵ-neighborhood (5) and k-nearest neighborhood (26). In this subsection, the conditions to be good membership sets are discussed.

Let each sample of the training data $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^n$ be denoted as a node of a size n adjacency graph G, which is indexed by an integer $i \in \{1,\ldots,n\}$. In addition, assume that a pair of nodes (i,j) are linked if the pair is an element of A_w. A graph is said to be *connected* in the sense of a topological space, if there is a path from any node to any other node in the graph [36] and in our case, the connectivity of G is absolutely determined by the edge matrix W_w. Remind that we assume that the data are sorted in the ascending order of the target value.

Now, consider G can be divided into two non-empty connected sub-graph $G_1=\{1,\ldots,k\}$ and $G_2=\{k+1,\ldots,n\}$ where all the nodes in G_1 have no link with the nodes in G_2. In this case, the similarity edge matrix W_w can be divided into two parts

$$W_w=\begin{bmatrix} W_{w1} & 0 \\ 0 & W_{w2} \end{bmatrix},$$

where $W_{w1} \in \mathbb{R}^{k\times k}$ and $W_{w2} \in \mathbb{R}^{(n-k)\times(n-k)}$. Consequently, the corresponding Laplacian matrix L_w can also be divided into two parts

$$L_w=\begin{bmatrix} L_{w1} & 0 \\ 0 & L_{w2} \end{bmatrix}.$$

Note that like L_w, the sub-matrices L_{w1} and L_{w2} are also Laplacian matrices. Therefore, L_{w1} and L_{w2} have their corresponding trivial eigenvectors of $\boldsymbol{e}_k$ and $\boldsymbol{e}_{n-k}$, respectively. Combining these two, the rank of L_w will be at most $n-2$ and the two orthogonal eigenvectors corresponding to zero eigenvalue are $\boldsymbol{\beta}_{0,1}=[\boldsymbol{e}_k^T,\boldsymbol{0}_{n-k}^T]^T$ and $\boldsymbol{\beta}_{0,2}=[\boldsymbol{0}_k^T,\boldsymbol{e}_{n-k}^T]^T$, where $\boldsymbol{0}_i$ is the i-dimensional zero vector. Any combination $\boldsymbol{\beta}^0=c_1\boldsymbol{\beta}_{0,1}+c_2\boldsymbol{\beta}_{0,2}$ ($c_1,c_2 \in \mathbb{R}$) of these two orthogonal eigenvectors also makes $L_w\boldsymbol{\beta}^0=0$. However, $\boldsymbol{\beta}^0(\not\propto \boldsymbol{e}_n)$ does not make $L_b\boldsymbol{\beta}^0=0$ in general.

Table 1
RMS errors for the simple linear dataset. Averages of 10-fold CV. Numbers in the parentheses are the standard deviations.

No. of features (m)	1	2	3	4	5
(a) 5NN					
Original	–	–	–	–	1.09 (0.07)
PCA	3.03 (0.70)	2.46 (0.63)	2.31 (0.69)	1.66 (0.71)	1.09 (0.07)
MLR	0.16 (0.08)	–	–	–	–
GPL	0.36 (0.03)	–	–	–	–
SIR	0.16 (0.07)	0.43 (0.07)	0.69 (0.06)	0.90 (0.05)	1.11 (0.04)
PHD	3.01 (0.87)	2.67 (0.80)	2.05 (0.62)	1.69 (0.57)	1.11 (0.04)
WPCA	0.18 (0.06)	0.44 (0.07)	0.70 (0.06)	0.92 (0.05)	1.11 (0.04)
LDAr	**0.15 (0.08)**	0.17 (0.07)	0.18 (0.07)	0.20 (0.06)	0.20 (0.06)
KDAr ($\sigma=10^3$)	0.16 (0.04)	**0.16 (0.04)**	**0.16 (0.04)**	**0.16 (0.04)**	**0.16 (0.04)**
(b) SVR					
Original	–	–	–	–	0.14 (0.02)
PCA	2.75 (0.18)	2.59 (0.24)	2.57 (0.24)	2.58 (0.29)	0.14 (0.02)
MLR	**0.11 (0.01)**	–	–	–	–
GPL	0.32 (0.05)	–	–	–	–
SIR	**0.11 (0.01)**	0.12 (0.01)	0.13 (0.02)	0.13 (0.02)	0.14 (0.02)
PHD	2.63 (0.34)	2.55 (0.32)	2.15 (0.57)	1.10 (0.86)	0.14 (0.02)
WPCA	0.12 (0.01)	0.12 (0.01)	0.13 (0.01)	0.13 (0.01)	0.14 (0.02)
LDAr	0.12 (0.01)	0.13 (0.02)	0.12 (0.01)	0.13 (0.02)	0.14 (0.02)
KDAr ($\sigma=10^3$)	**0.11 (0.01)**	**0.11 (0.01)**	**0.11 (0.01)**	**0.11 (0.01)**	**0.11 (0.01)**
(c) GPR					
Original	–	–	–	–	**0.10 (0.01)**
PCA	2.74 (0.29)	2.59 (0.24)	2.57 (0.23)	2.55 (0.28)	**0.10 (0.01)**
MLR	**0.10 (0.01)**	–	–	–	–
GPL	0.32 (0.05)	–	–	–	–
SIR	0.11 (0.01)	0.11 (0.01)	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**
PHD	2.63 (0.34)	2.55 (0.33)	2.14 (0.58)	1.10 (0.87)	**0.10 (0.01)**
WPCA	0.11 (0.01)	0.11 (0.01)	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**
LDAr	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**
KDAr ($\sigma=10^3$)	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**	**0.10 (0.01)**

Therefore, with $\boldsymbol{\beta}=\boldsymbol{\beta}^0$, the generalized eigenvalue λ in (24) will be positive infinite and $\boldsymbol{\beta}^0$ becomes the optimal solution of (23). However, considering that input data X are projected to $\boldsymbol{\beta}^T$, we can see that $\boldsymbol{\beta}^0$ is not a good choice. With $\boldsymbol{\beta}^0$ all the data in G_1, i.e., $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k\}$, are projected to the same point and likewise, the ones in G_2 $(\{\boldsymbol{x}_{k+1},\ldots,\boldsymbol{x}_n\})$ are also projected to a single point.

Although this problem can be resolved by the technique described in Section 4.1 which avoids 0 from the denominator of $J(\boldsymbol{\beta})$, there remains another problem. Remind that finding $\boldsymbol{\beta}$ is equivalent to finding the eigenvector of L_w corresponding to the smallest non-trivial eigenvalue, if the constant weight function $f(a,b)=1$ is used. If we assume L_w can be separated into two disjoint sub-matrices $L_{w1}\in\mathbb{R}^{k\times k}$ and $L_{w2}\in\mathbb{R}^{(n-k)\times(n-k)}$, then the eigenvectors of L_w are also divided into two sets $\{\boldsymbol{\beta}:\boldsymbol{\beta}=[\boldsymbol{v}^T,\boldsymbol{0}_{n-k}^T]^T\}$ and $\{\boldsymbol{\beta}:\boldsymbol{\beta}=[\boldsymbol{0}_k^T,\boldsymbol{u}^T]^T\}$, where $\boldsymbol{v}$'s and $\boldsymbol{u}$'s are the eigenvectors of L_{w1} and L_{w2} respectively. In this case, any eigenvector contains a group of zeros, which indicates that many training samples are projected to a single point 0.

From the above discussion, it is desirable that L_w is not separated into sub-matrices which is possible by making a connected graph G. If we use the ϵ-neighborhood membership sets (5), then the connectivity depends on the difference of the target values of two consecutive nodes i and $i+1$. That is, if $y_{i+1}-y_i>\tau$, the two consecutive nodes are not linked, making the adjacency graph G disconnected. Therefore, to ensure G be connected, for all the $i\in\{1,\ldots,n-1\}$, $y_{i+1}-y_i$ should not exceed ϵ. As a consequence, in case of large variation of $y_{i+1}-y_i$ values, the performance of ϵ-neighborhood membership sets (5) is expected to be poor.

On the other hand, if we adopt k-nearest neighborhood type membership sets (26), then for all the $\tau>0$, the connectivity of G is guaranteed. Therefore, in all the experiments of the next section, we use (26) for the membership sets.

5. Experimental results

In this section, the proposed algorithm KDAr is applied to several regression problems and the performance of KDAr is compared with those of conventional linear feature extraction methods such as PCA, MLR, GPL [21], SIR, PHD, WPCA [22], LDAr. Note that among these, PCA can be categorized as unsupervised feature extraction methods, while the others are classified as supervised feature extraction methods which utilizes the target information.

Throughout the paper, the following experimental settings were used:

- *Regressor*: As in [22], the weighted *five nearest neighborhood* (5NN) regression [39] was used as a regression system. This regressor was chosen because of its simplicity. For some experiments, support vector regressor (SVR) [30,40] and Gaussian process regressor (GPR) [21] were also used for comparison.
  In the 5NN regressor, the estimation of the target variable $\hat{t}(\boldsymbol{z})$ with input variables $\boldsymbol{z}$ is obtained as follows:

$$\hat{t}(\boldsymbol{z})=\frac{\sum_{i\in N(\boldsymbol{z})}q(\boldsymbol{z},\boldsymbol{z}_i)t_i}{\sum_{i\in N(\boldsymbol{z})}q(\boldsymbol{z},\boldsymbol{z}_i)}. \tag{28}$$

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

Table 2
RMS errors for the nonlinear dataset. Averages of 10-fold CV. Numbers in the parentheses are the standard deviations.

No. of features (m)	1	2	3	4	5
(a) 5NN					
Original	–	–	–	–	0.46 (0.02)
SVR ($\sigma=1$)	–	–	–	–	0.27 (0.02)
PCA	0.77 (0.02)	0.76 (0.05)	0.61 (0.16)	0.51 (0.14)	0.46 (0.02)
MLR	0.52 (0.07)	–	–	–	–
GPL	0.54 (0.06)	–	–	–	–
SIR	0.56 (0.09)	0.50 (0.08)	0.48 (0.09)	0.44 (0.06)	0.46 (0.01)
PHD	0.72 (0.14)	0.66 (0.19)	0.59 (0.20)	0.54 (0.15)	0.46 (0.01)
WPCA	0.48 (0.07)	0.48 (0.08)	0.45 (0.08)	0.43 (0.05)	0.46 (0.01)
LDAr	0.47 (0.10)	0.44 (0.11)	0.37 (0.07)	0.38 (0.03)	0.44 (0.02)
KDAr ($\sigma=5$)	**0.24 (0.04)**	**0.24 (0.04)**	**0.24 (0.04)**	**0.23 (0.04)**	**0.23 (0.03)**
(b) SVR					
Original	–	–	–	–	0.24 (0.02)
PCA	0.70 (0.02)	0.80 (0.11)	1.11 (0.27)	1.39 (0.31)	0.24 (0.02)
MLR	0.55 (0.06)	–	–	–	–
GPL	0.64 (0.07)	–	–	–	–
SIR	0.50 (0.07)	0.50 (0.07)	0.78 (0.20)	0.86 (0.31)	0.24 (0.02)
PHD	0.71 (0.03)	0.82 (0.14)	1.04 (0.16)	1.18 (0.39)	0.24 (0.02)
WPCA	0.60 (0.06)	0.64 (0.14)	0.69 (0.08)	1.18 (0.25)	0.24 (0.02)
LDAr	0.59 (0.05)	0.64 (0.08)	0.70 (0.20)	0.88 (0.19)	0.25 (0.02)
KDAr ($\sigma=5$)	**0.22 (0.03)**	**0.21 (0.02)**	**0.21 (0.02)**	**0.21 (0.03)**	**0.20 (0.02)**
(c) GPR					
Original	–	–	–	–	**0.10 (0.01)**
PCA	0.70 (0.01)	0.70 (0.01)	0.68 (0.03)	0.57 (0.07)	0.13 (0.02)
MLR	0.54 (0.05)	–	–	–	–
GPL	0.54 (0.05)	–	–	–	–
SIR	0.48 (0.06)	0.46 (0.06)	0.45 (0.09)	0.38 (0.12)	**0.10 (0.01)**
PHD	0.69 (0.02)	0.67 (0.02)	0.65 (0.03)	0.53 (0.15)	0.16 (0.03)
WPCA	0.58 (0.05)	0.53 (0.06)	0.49 (0.07)	0.48 (0.08)	**0.10 (0.02)**
LDAr	0.58 (0.04)	0.56 (0.05)	0.42 (0.09)	0.33 (0.11)	**0.10 (0.01)**
KDAr ($\sigma=5$)	**0.18 (0.02)**	**0.16 (0.02)**	**0.16 (0.02)**	**0.15 (0.02)**	**0.10 (0.01)**

- *Edge matrices*: For KDAr, the (i,j)-th elements of the similarity edge matrix W_w and the dissimilarity edge matrix W_b were set to

$$w_{ij} = \begin{cases} \tau-|i-j| & \text{if } |i-j| < \tau, \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_{ij} = \begin{cases} \min(|i-j|-\tau,\tau) & \text{if } |i-j| \geq \tau, \\ 0 & \text{otherwise,} \end{cases}$$

respectively where τ was set to $n/10$.
- *Kernel function*: As a kernel function, Gaussian RBF kernel of the form $\kappa(\boldsymbol{x},\boldsymbol{y}) = \exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/d\sigma)$ is used with various values of σ. Here, in the denominator of the exponential, the dimensionality of input space d is multiplied as a normalizing factor. The σ that resulted the best performance on the test data is reported in this paper.

All the experimental settings of the conventional methods are the same as in [22].

5.1. Artificial problems

5.1.1. Linear case

Suppose we have five independent input features x_1–x_5 which have Gaussian 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 and the performance of KDAr was compared with those of conventional feature extraction methods. As a regressor, not only 5NN regressor described above, but also SVR and GPR were used. Ten-fold cross-validation was applied and root mean square (rms) errors on the test data with various numbers of extracted features ($m=1,\ldots,5$) are shown in Table 1. The numbers in the parentheses are the standard deviations. For KDAr, the optimal σ value is also reported in the table. Note that MLR and GPL can extract only one feature.

From the table, regardless of the type of regressor, we can see that when the number of extracted features is 1 ($m=1$), all the feature extraction methods except PCA, GPL and PHD performed well and resulted almost the same rms error. GPL was better than PCA and PHD, but it resulted in poorer performance than the other methods. Because this problem is linear and the distribution is Gaussian, MLR is optimal in the least square sense and other linear methods such as SIR, WPCA and LDAr performed almost the same as MLR when $m=1$. Regardless of the type of regressor, KDAr which extracts nonlinear features performed equally well for this linear problem.

When 5NN regressor was used, as the number of extracted features increases, the performances of other conventional linear feature extraction methods become worse, while the rms error of KDAr remains unchanged. On the other hand, when other nonlinear regressors such as SVR and GPR were used, the performances were almost constant regardless of the number of extracted features. The reason why 5NN regressor performs poorer with additional features is that the additional features act as noise in calculating the Euclidian distance in (28) while the nonlinear regressors, SVR and GPR, efficiently reduce the effect of the additional features.

Table 3
RMS errors for the Housing dataset with 5NN. Averages of 10 experiments. Numbers in the parentheses are the standard deviations.

No. of features (m)	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)
KDAr ($\sigma = 10^3$)	**3.10 (0.42)**	**2.73 (0.51)**	**2.65 (0.44)**	**2.77 (0.43)**	**2.81 (0.43)**	**2.85 (0.46)**	**2.84 (0.47)**

5.1.2. Nonlinear case

Suppose we have five independent input features x_1–x_5 which have Gaussian 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. Ten-fold cross-validation was applied to this dataset and the rms errors of various feature extraction methods on the test data are reported in Table 2. As well as 5NN regressor, SVR and GPR were also used. As in Table 1, the numbers in the parentheses are the standard deviation and the optimal value of σ is reported for KDAr.

As can be seen from the table, for this nonlinear problem, when 5NN was used as a regressor, KDAr performed far better than the other linear methods because it utilizes nonlinear feature space. Note that the performance of KDAr is almost constant regardless of the number of extracted features m in Table 2(a). When SVR was used, in Table 2(b), we can obtain a slightly improved performance of KDAr compared to 5NN case. In this case also, KDAr slightly outperformed other methods. In Table 2(c), when GPR was used, the rms errors of MLR, SIR, WPCA, LDAr and KDAr were almost the same regardless of the number of extracted features.

The reason for this phenomenon can be conjectured as follows. The conventional feature extractors have a limited power of extracting good features because they only try to find linear combinations of input variables. Therefore, simple regressors such as 5NN cannot achieve good performance, while more complex and contrived regressors such as SVR and GPR may achieve relatively good results on these linear feature extractors. On the other hand, because KDAr is a nonlinear feature extractor for regression problems, its output features are good for any regressor, whether it is simple or complex. From this, we can conclude that the performance of KDAr is rather consistent regardless of the regressor used.

5.2. Real world datasets

5.2.1. Housing—Boston

In this section, we have applied the proposed feature extraction methods to the *Housing (Boston)* dataset in UCI Machine Learning Repository [41].

The dataset contains 13 input features, 12 continuous and 1 binary, and 1 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. Weighted 5NN regressor is used as a regression system. In the table, the numbers in the parentheses are the standard deviation of 10 experiments.

Table 4
One tailed t-test for the Housing dataset.

No. of features (m)	1	3	5	7	9	11	13
T-value	4.55	4.97	3.52	3.34	2.66	2.52	2.63
d.o.f.	17	17	15	16	15	16	16
$T_{99\%}$	2.57	2.57	2.60	2.58	2.60	2.58	2.58
$T_{95\%}$	1.74	1.74	1.75	1.75	1.75	1.75	1.75
Accepted (99%)	H_A	H_A	H_A	H_A	H_A	H_0	H_A
Accepted (95%)	H_A	H_A	H_A	H_A	H_A	H_A	H_A

From the table, we can see that the KDAr is better than all the other methods especially when the number of extracted features m is small. For $m=1$, the difference of rms errors between KDAr and LDAr is more than 1.0, while it decreases to around 0.7 when $m=13$. The minimum rms error 2.65 was obtained when $m=5$.

To show the statistical significance of the experimental results, we have performed one tailed Welch's t-test [42] in Table 3. The null (H_0) and the alternative (H_A) hypotheses for this statistical test are as follows:

- H_0: For a fixed number of extracted features m, the performances of KDAr and the best linear feature extraction method are the same.
- H_A: For a fixed number of extracted features m, KDAr outperforms all the other linear feature extraction methods.

The computed T-value, degree of freedom (d.o.f.) and the corresponding target T values are shown in Table 4. For each m, if the T-value is greater than $T_{99\%}$ ($T_{95\%}$), the null hypothesis H_0 is rejected with 99% (95%) of confidence, thus alternative hypothesis H_A is adopted.

In the table, when the confidence level is 95%, for all the numbers of extracted features m, the null hypothesis was rejected, thus the alternative hypothesis was accepted. If we raise the confidence level to 99%, for all the numbers of extracted features except for the case of $m=11$, the null hypothesis was rejected. From this, we can conclude that the KDAr outperforms other linear feature extraction methods for Housing dataset.

5.2.2. Year prediction of million song dataset

This data, a subset of the Million Song Dataset (MSD) [43], can be found in the UCI Machine Learning Repository [41]. This dataset consists of total 515,345 songs, among which the first 463,715 examples are training data while the other 51,630 examples are test data. In the dataset, there are 90 input attributes and the target value is the year of the song, ranging from 1922 to 2011.

Because the number of training data is so huge, for computational efficiency, we randomly selected 2000 training samples 20

Table 5
RMS errors for the year prediction MSD dataset with 5NN. The numbers in the parentheses are the standard deviations of 20 experiments.

No. of features (m)	1	3	5	10	15
PCA	11.80 (0.13)	11.49 (0.15)	11.41 (0.13)	11.16 (0.17)	11.01 (0.13)
MLR	10.44 (0.21)	–	–	–	–
GPL	11.25 (0.12)	–	–	–	–
SIR	10.44 (0.12)	10.39 (0.14)	10.33 (0.18)	10.29 (0.15)	10.28 (0.14)
PHD	11.81 (0.16)	11.73 (0.16)	11.55 (0.17)	11.33 (0.21)	11.17 (0.21)
WPCA	11.72 (0.26)	11.02 (0.41)	10.74 (0.27)	10.50 (0.12)	10.49 (0.15)
LDAr	11.37 (0.46)	10.74 (0.28)	10.60 (0.09)	10.51 (0.09)	10.49 (0.12)
KDAr	**10.23 (0.34)**	**10.03 (0.10)**	**10.01 (0.10)**	**9.96 (0.14)**	**9.95 (0.11)**

Table 6
One tailed t-test for the year prediction MSD dataset.

No. of features (m)	1	3	5	10	15
T-value	1.84	6.43	4.91	5.09	5.86
d.o.f.	11	16	14	18	17
$T_{99\%}$	2.72	2.58	2.62	2.56	2.57
$T_{95\%}$	1.77	1.75	1.76	1.73	1.74
Accepted (99%)	H_0	H_A	H_A	H_A	H_A
Accepted (95%)	H_A	H_A	H_A	H_A	H_A

Table 7
RMS errors for the orange juice dataset with 5NN.

No. of features (m)	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	–
KDAr ($\sigma = 2 \times 10^4$)	**5.45**	**6.23**	**6.11**	**6.29**	6.31	**6.36**	**6.41**	–

times to extract features with various methods and report the average rms errors in Table 5. The numbers in the parentheses are the standard deviations of 20 experiments. In the table, we can see that for all m, KDAr outperforms other conventional feature extraction methods.

To show the statistical significance of this result, we also performed one tailed Welch's t-test [42] in Table 5. The null (H_0) and the alternative (H_A) hypotheses for this statistical test are the same as the ones in the previous experiments on Housing dataset. Table 6 shows that regardless of the number of extracted features m, H_A is accepted with 95% of confidence. This is also true for 99% confidence except when $m=1$ where H_0 is adopted. This results show that KDAr is better than other conventional feature extraction methods for this dataset.

5.2.3. *Orange juice*

Orange juice dataset was obtained from the UCL machine learning database [44], 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 small sample size (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 linear feature extraction methods except PCA, we have preprocessed the dataset with PCA and reduced the dimension of input space into $149(=n-1)$. On the other hand, unlike the conventional linear feature extraction methods, the preprocessing step is not applied to KDAr because KDAr does not suffer from SSS problem.

Table 7 shows the performances of various feature extraction methods on the test dataset. For this problem, the best rms error 5.45 was obtained by KDAr when $m=1$. As m increases, the rms error of KDAr increases up to 6.41 when $m=13$. Note that the performance is somewhat unstable and LDAr performed better than KDAr when $m=9$. The reason can be attributed to the fact that 5NN regressor performs unstably when the number of training examples is small. In fact for $\sigma = 2 \times 10^3$, the rms error of KDAr was 6.07 which is smaller than that of LDAr. However, for this value of σ rms error was 5.95 when $m=1$.

5.2.4. *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 (seven joint positions, seven joint velocities, seven joint accelerations) and seven output variables (the corresponding seven joint torques). The dataset has previously been used to study regression algorithms [21,45]. There are 48,933 input–output pairs in the dataset, of which 44,484 were used as a training set and the remaining 4449 were used as a test set in [21,45].

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 8 and Fig. 2. In the figure, due to space limitation, only the rms errors of the first two and last two target variables with PCA, SIR, PHD, WPCA, LDAr, and KDAr 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. We tested various values of σ of Gaussian RBF kernel for KDAr and reported best performance in the figure. Fig. 2(a)–(d) is obtained with $\sigma = 10^4, 5 \times 10^3, 2 \times 10^2$, and 2×10^2 respectively.

In both Fig. 2(a) and (b), the performance of KDAr was better than any other feature extraction methods. Note that among linear feature extraction methods, LDAr performs best but KDAr outperforms LDAr by at least 10%. This phenomenon can also be seen in Table 8 where we show the best performances of various feature extraction methods. The best number of extracted features are also indicated. In the table, KDAr outperforms LDAr by 27.3% on average. For PCA, SIR, PHD, WPCA, LDAr and KDAr, we extracted 1, 3, 5, 7 and 10 features. Note that MLR and GPL can

Table 8
RMS errors for the SARCOS Robot Arm dataset with 5NN. The numbers in the parentheses are the best number of extracted features.

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)
KDAr	**5.07** (7)	**3.36** (3)	**1.89** (7)	**1.62** (5)	**0.25** (5)	**0.45** (5)	**0.43** (5)
σ	10^4	5×10^3	2×10^3	5×10^3	2×10^2	2×10^2	2×10^2

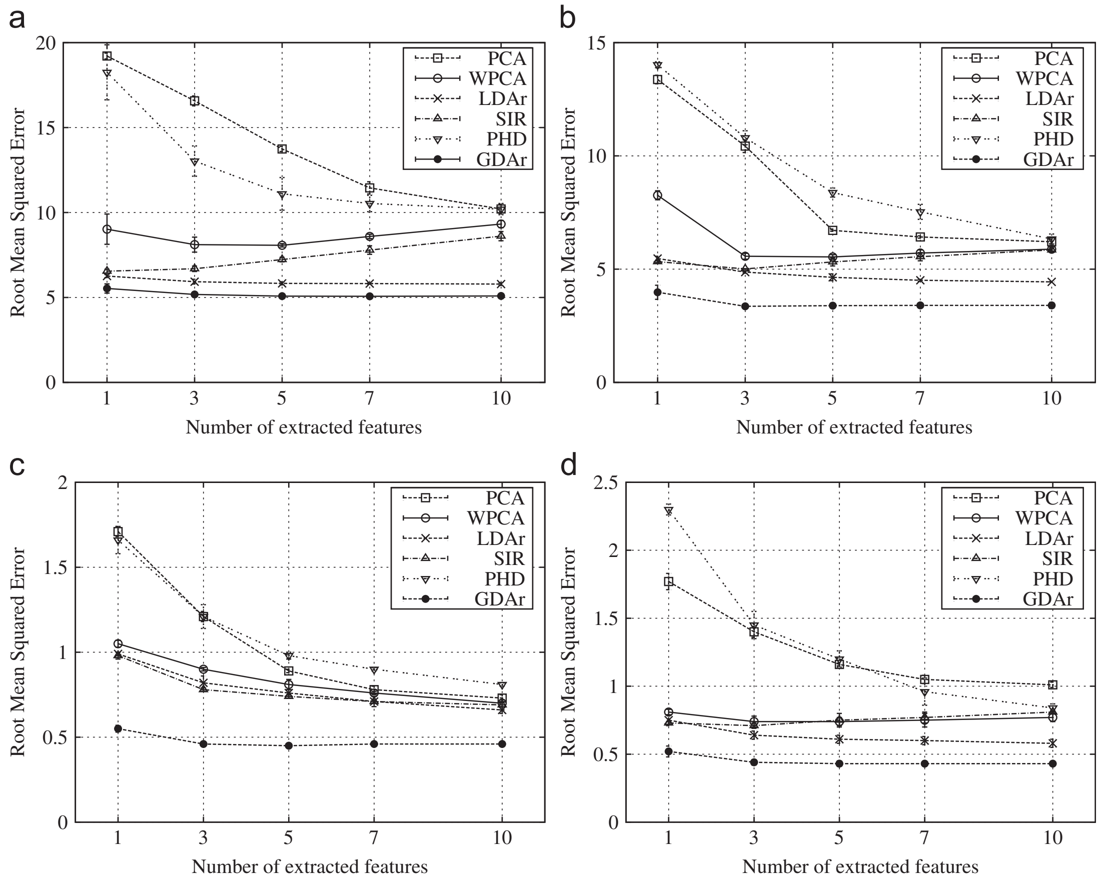

Fig. 2. RMS error for the four target variables of SARCOS Robot Arm dataset. (a) 1st torque, (b) 2nd torque, (c) 6th torque, (d) 7th torque.

extract only one feature. From the table, we can see that the best performance of KDAr was better than other feature extraction methods for all the seven target variables. Comparing LDAr and KDAr, we can see that the kernel trick is effective in performance enhancement of regression for this problem.

5.3. *Face alignment*

5.3.1. *Yale database—rotation*

In this section, we perform experimental studies on finding the rotation angle of Yale database [9] to evaluate the performance of the KDAr. In [9], the authors report two types of databases: a closely cropped set and a full face set. In this paper, the full face set whose size is 100×80 pixels was used. The Yale database consists of 165 images which contains 11 images per each of 15 individuals. Original images are in gray color. In our experiment, 55 images of the first five individuals were rotated to generate training data while those of the other 11 individuals were used for test data.

If we rotate a rectangular face image to obtain a new rectangular image of the same size, the corner pixels cannot be filled as shown in Fig. 3. With this information, the rotation angle can easily be recovered. Therefore, in our experiment, instead of rectangular image, we used circularly cropped face images as follows. Each image was firstly cropped circularly based on a fixed center point of (57, 40) with a radius of 30 pixels, then rotated with 10 random angles uniformly distributed from -30° to $+30^\circ$. As a result, 550 $(5\times11\times10)$ training images and 1100 $(10\times11\times10)$ test images were obtained. Fig. 4 shows examples of circularly cropped and randomly rotated images of one original image.

In this problem, each pixel constitutes one input variable while the target variable is the rotation angle in degree. Using 550 training images, we extracted various numbers of features with different feature extraction methods and estimated the rotation angles of the 1100 test data. Table 9 shows the rms errors on test data with various feature extraction methods. Both 5NN and GPR were used as a regressor and the reported rms errors are the smaller of the two.

In the table, rms error was smallest when KDAr was used as a feature extraction method except the case for $m=1$. Although the performance of KDAr was worse than MLR and LDAr for $m=1$ when $\sigma=200$, for different values of σ KDAr performed better than LDAr and MLR even when $m=1$. In the table, the best performance was obtained with KDAr when $m=40$. Comparing the performance of GPR and 5NN, when $m=1$, GPR performed better than 5NN except for MLR, LDAr and KDAr. For other values of m, 5NN outperformed GPR in all the experiments.

5.3.2. Yale database—scaling

In this subsection, each of the Yale face images was scaled 10 times by a random factor from 100% up to 150% and then cropped to the original 100×80 size as shown in Fig. 5. As in the rotation experiment in the previous subsection, the first 55 images of five individuals were used to create the training images, while the remaining 110 images were used for testing. Therefore, the number of training and test images are 550 and 1100 respectively.

In this problem, the target variable is the scaling factor in percent. As a regressor, both 5NN and GPR were used and the smaller rms errors of the two regressors on the test data are reported in Table 10.

In the table, we can see that KDAr outperforms other linear feature extraction methods for all the number of extracted features m. Note that LDAr performs rather bad for this problem but its nonlinear extension KDAr performs better than other conventional feature extraction methods. For this problem, when the number of extracted features is one, GPR performed better than 5NN but as the number of features increases, 5NN becomes better except for the LDAr case.

6. Conclusion

In this paper, a new feature extraction method KDAr for regression problem has been proposed. It is an extension of LDAr, a linear discriminant analysis for regression problems, to a non-linear version by using the so-called kernel trick. The basic idea is to map the input space to a high-dimensional feature space in which variables are nonlinearly related to the input space and then try to maximize the ratio of distances of samples with large differences in the target value and those with small differences in the target value. In the derivation of KDAr, we have investigated the relationship of KDAr with Laplacian eigenmap and gave a guideline for edge matrices. In addition, the properties of KDAr were also investigated.

We have applied the proposed method to several regression problems including artificial and real-world problems as well as facial alignment problems and compared the performance of KDAr with those of the conventional linear feature extraction methods. The experimental results show that the proposed KDAr outperforms the conventional feature extraction methods in most cases and can be used as a dimensionality reduction method for regression problems.

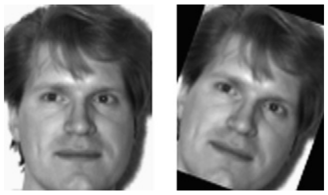
Fig. 3. Original image (left) and 20° rotated image (right) of Yale database. The corner pixels that cannot be filled are painted black.

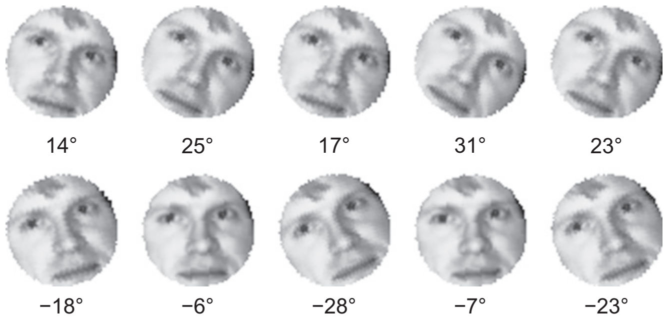

Fig. 4. Rotated images of a circularly cropped Yale database with their corresponding rotation angles.

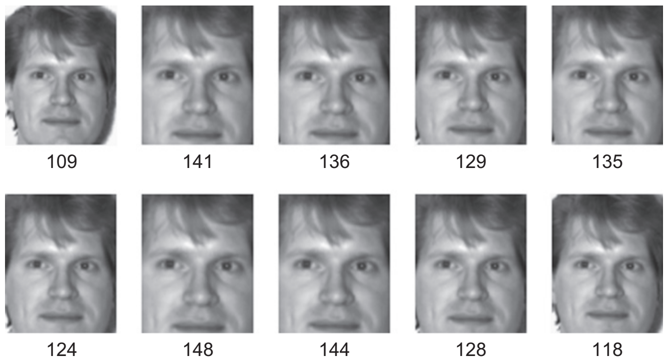

Fig. 5. Scaled images of Yale database with their corresponding scaling factor in percent.

Table 9
Performance for the Yale rotation dataset (rms errors of test data in °). Both 5NN and GPR were used and the smaller rms errors of the two is reported. 'N' and 'G' in the parenthesis denote 5NN and GPR respectively.

No. of features (m)	1	10	20	30	40	50
PCA	19.92 (G)	4.45 (N)	3.95 (N)	3.58 (N)	3.46 (N)	3.37 (N)
MLR	**4.20 (N)**	–	–	–	–	–
GPL	7.30 (G)	–	–	–	–	–
SIR	6.92 (G)	3.69 (N)	3.74 (N)	3.60 (N)	3.58 (N)	3.49 (N)
PHD	15.71 (G)	4.50 (N)	4.09 (N)	3.71 (N)	3.82 (N)	3.79 (N)
WPCA	19.13 (G)	4.35 (N)	3.93 (N)	3.57 (N)	3.48 (N)	3.40 (N)
LDAr	5.11 (N)	1.82 (N)	4.02 (N)	3.90 (N)	6.85 (N)	7.25 (N)
KDAr ($\sigma=2\times 10^2$)	5.71 (N)	**1.55 (N)**	**1.28 (N)**	**1.19 (N)**	**1.17 (N)**	**1.19 (N)**

Table 10
Performance for the Yale scaling dataset (rms errors of test data in %). Both 5NN and GPR were used and the smaller rms errors of the two is reported. 'N' and 'G' in the parenthesis denote 5NN and GPR respectively.

No. of features (m)	1	10	20	30	40	50
PCA	16.03 (G)	8.32 (N)	7.77 (N)	7.76 (N)	7.45 (N)	7.36 (N)
MLR	11.02 (N)	–	–	–	–	–
GPL	7.82 (G,N)	–	–	–	–	–
SIR	7.98 (G)	7.12 (N)	7.53 (N)	7.51 (N)	7.53 (N)	7.60 (N)
PHD	10.43 (G)	8.95 (N)	7.40 (N)	7.52 (N)	7.32 (N)	7.28 (N)
WPCA	16.01 (G)	8.23 (N)	7.75 (N)	7.75 (N)	7.45 (N)	7.31 (N)
LDAr	10.75 (N)	14.22 (G)	14.22 (G)	14.22 (G)	14.22 (G)	14.22 (G)
KDAr ($\sigma = 1.5 \times 10^1$)	**7.60 (G)**	**6.80 (N)**	**6.83 (N)**	**6.84 (N)**	**6.86 (N)**	**6.82 (N)**

Acknowledgments

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KPF-2011-0005324) and Ajou University under Research Grant 20094720.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., Wiley, 2001.
[2] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice Hall, London, 1982.
[3] V. Cherkassky, F. Mulier, Learning from Data: Concept, Theory, and Methods, John Wiley & Sons, 1998.
[4] H. Liu, H. Motoda (Eds.), Feature Extraction, Construction and Selection: A Data Mining Perspective, Springer, 1998.
[5] I.T. Joliffe, Principal Component Analysis, Springer-Verlag, 1986.
[6] A.J. Bell, T.J. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Computation 7 (6) (1995) 1129–1159.
[7] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd ed., Academic Press, 1990.
[8] M. Turk, A. Pentland, Face recognition using eigenfaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1991, pp. 586–591.
[9] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (7) (1997) 711–720.
[10] G. Donato, M.S. Bartlett, J.C. Hager, P. Ekman, T.J. Sejnowski, Classifying facial actions, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (10) (1999) 974–989.
[11] X. Wang, X. Tang, A unified framework for subspace face recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 40–51.
[12] D. Xu, S. Yan, D. Tao, S. Lin, H.J. Zhang, Marginal fisher analysis and its variants for human gait recognition and content-based image retrieval, IEEE Transactions on Image Processing 16 (11) (2007) 2811–2812.
[13] T. Okada, S. Tomita, An optimal orthonormal system for discriminant analysis, Pattern Recognition 18 (2) (1985) 139–144.
[14] M. Loog, R.P.W. 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.
[15] M. Li, B.Z. Yuan, 2d-LDA: a statistical linear discriminant analysis for image matrix, Pattern Recognition Letters 26 (5) (2005) 527–532.
[16] 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.
[17] S. Weisberg, Applied Linear Regression, 2nd 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] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, 1st ed., The MIT Press, 2006.
[22] N. Kwak, J.-W. Lee, Feature extraction based on subspace methods for regression problems, Neurocomputing 73 (June) (2010) 1740–1751.
[23] N. Kwak, C. Kim, H. Kim, Dimensionality reduction based on ICA for regression problems, Neurocomputing 71 (August) (2008) 2596–2603.
[24] J.B. Tenembaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (December) (2000) 2319–2323.
[25] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (December) (2000) 2323–2326.
[26] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (6) (2003) 1373–1396.
[27] L. Cayton, Algorithms for Manifold Learning, Technical Report, UCSD, 2005.
[28] X. He, P. Niyogi, Locality preserving projections, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, MA, 2004.
[29] H. Abdi, Encyclopedia of Measurement and Statistics, Thousand Oaks, 2007, pp. 598–605.
[30] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2002.
[31] B. Schölkopf, A.J. Somla, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.
[32] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, K.R. Müller, Fisher discriminant analysis with kernels. in: Proceedings of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing, vol. IX, 1999, pp. 41–48.
[33] G. Baudat, R. Anouar, Generalized discriminant analysis using a kernel approach, Neural Computation 12 (10) (2000) 2385–2404.
[34] R. Rosipal, M. Girolami, L.J. Trejo, Kernel PCA for feature extraction and de-noising in non-linear regression, Neural Computing & Applications 10 (2001) 231–243.
[35] P. Jain, B. Kulis, I. Dhillon, Inductive regularized learning of kernel function, in: Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS 2010), December, 2010.
[36] S. Yan, D. Xu, B. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 40–51.
[37] P. Vincent, Y. Bengio, Kernel matching pursuit, Machine Learning 48 (2002) 169–191.
[38] J. Yang, A.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.
[39] E.A. Narayada, On estimating regression, Theory of Probability and Its Applications 9 (1964) 141–142.
[40] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, SVM and kernel methods Matlab toolbox, Perception Systemes et Information, INSA de Rouen, Rouen, France, 2005.
[41] D.J. Newman, S. Hettich, C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, 1998, 〈http://www.ics.uci.edu/∼mlearn/MLRepository.html〉.
[42] B.L. Welch, Biometrika 34 (1–2) (1947) 28–35.
[43] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, P. Lamere, The million song dataset, in: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.
[44] M. Meurens, Orange Juice Data, 〈http://www.dice.ucl.ac.be/mlg/DataBases/ORANGE_JUICE/〉.
[45] S. Vijayakumar, A. D'Souza, S. Schaal, Incremental online learning in high dimensions, Neural Computation 17 (12) (2005) 2602–2634.

Nojun Kwak received the BS, MS, and PhD degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, in 1997, 1999 and 2003 respectively. From 2003 to 2006, he worked for Samsung Electronics. In 2006, he joined Seoul National University as a BK21 assistant professor. Since 2007, he has been with Ajou University, Korea where he works as an associate professor. His research interests include pattern recognition, machine learning, computer vision, data mining, image processing, and their applications.