Sliced Coordinate Analysis for Effective Dimension Reduction and Nonlinear Extensions

Zhihua Zhang[*], Dit-Yan Yeung[†], James T. Kwok[‡], and Edward Y. Chang[§]

May 1, 2007

Abstract

Sliced inverse regression (SIR) is an important method for reducing the dimensionality of input variables. Its goal is to estimate the effective dimension reduction directions. In classification settings, SIR is closely related to Fisher discriminant analysis. Motivated by reproducing kernel theory, we propose a notion of nonlinear effective dimension reduction and develop a nonlinear extension of SIR called kernel SIR (KSIR). Both SIR and KSIR are based on principal component analysis. Alternatively, based on principal coordinate analysis, we propose the dual versions of SIR and KSIR, which we refer to as sliced coordinate analysis (SCA) and kernel sliced coordinate analysis (KSCA), respectively. In the classification setting, we also call them discriminant coordinate analysis and kernel discriminant coordinate analysis. The computational complexities of SIR and KSIR rely on the dimensionality of the input vector and the number of input vectors, respectively, while those of SCA and KSCA both rely on the number of slices in the output. Thus, SCA and KSCA are very efficient dimension reduction methods.

1. Introduction

The notion of *effective dimension reduction* (e.d.r.) (Li, 1991) plays a central role in dimension reduction under a regression model. The desire behind this notion is that one can reduce the dimensionality of input variables without losing any information that is essential for predicting the corresponding output. Li (1991) developed a notable *sliced inverse regression* (SIR) method for estimating the e.d.r. space. Unlike principal component regression, which first applies principal component analysis (PCA) (Jolliffe, 2002) on the input variables and then models the relationship between the first few principal components and the output, SIR uses the idea of inverse regression. Roughly speaking, it reduces the dimensionality of an input vector by regressing the input vector against the corresponding output to form an e.d.r. space, and then projects an input vector onto this space. Based on the inverse regression, many other methods have been proposed to estimate the e.d.r. space, such as sliced average-variance estimate (SAVE) (Cook and Weisberg, 1991) and principal

∗. Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA.

†. Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

‡. Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China.

§. Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA, USA.

Hessian direction (PHD) (Li, 1992, Cook, 1998). Methods based on the e.d.r. space have also been extended to the classification problem (Cook and Lee, 1999, Cook and Yin, 2001). In fact, except for a scaling factor, SIR is equivalent to Fisher discriminant analysis (FDA), which seeks to find a linear transformation by maximizing the ratio of the between-class scatter to the within-class scatter (Mardia et al., 1979). For this reason, we will use the terms SIR and FDA interchangeably in this paper to refer to essentially the same method.

SIR estimates the e.d.r. directions by solving a generalized eigenvalue problem (Golub and Loan, 1996) that involves the between-slice covariance matrix and the sample covariance matrix of the input vectors. Thus, its computational complexity depends on the dimensionality of the input space. To solve the generalized eigenvalue problem, SIR requires the sample covariance matrix to be nonsingular. This can become problematic when the dimensionality is high. On the one hand, the computational cost of SIR becomes high. On the other hand, the sample covariance matrix is likely to be singular. For example, if the number of input vectors is less than the dimensionality of the input space, the covariance matrix is singular. As a result, the generalized eigenvalue problem for standard SIR becomes intractable. However, thanks to the equivalence between SIR and FDA, we can resort to the existing approaches developed for FDA. For example, the regularization approach (Hastie et al., 2001) is commonly used. Recently, Howland et al. (2003) applied the generalized singular value decomposition (GSVD) method (Paige and Saunders, 1981) to solve the generalized eigenvalue problem so that the non-singularity requirement on the sample covariance matrix is no longer necessary.

In this paper, we propose a new approach to e.d.r. under the inverse regression scheme. Instead of estimating the e.d.r. directions, our basic idea is to directly estimate the coordinates of the projections of the input vectors in the e.d.r. space. Accordingly, we develop a new method called *sliced coordinate analysis* (SCA). Specifically, we first calculate the projection coordinates of the means within each slice on the e.d.r. space by applying principal coordinate analysis (PCO) (Gower, 1966, Mardia et al., 1979) on the distance matrix between the means. Utilizing these coordinates, we then interpolate the projection of an input vector onto the e.d.r. space. Since SCA is derived from the notion of e.d.r., it inherits the theoretical framework developed for SIR. It is worth noting that SCA is similar to the analysis of distance proposed by Gower and Krzanowski (1999), whose aim is to estimate the coordinates of a set of observations when only a distance function between any two such observations is available. The main computational cost of SCA comes from the eigen-decomposition of the between-slice distance matrix. Since the size of the between-slice distance matrix is equal to the number of slices, which is typically far less than the number of input vectors, our proposed SCA is very efficient, especially in the case that the data set is high-dimensional. Moreover, SCA does not explicitly use the sample covariance matrix. Therefore, it does not matter whether the sample covariance matrix is singular or not.

Both SIR and SCA rely on the assumption of linearity of the data at hand. A sufficient condition for satisfying this assumption is that the data follow some elliptically symmetric distributions, e.g., the normal distribution. In recent years, kernel methods (Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004) have been increasingly popular for data and information processing due to their benefits from conceptual simplicity and theoretical potentiality. Kernel methods work by nonlinearly mapping vectors in the input space to a higher-dimensional feature space, and then implementing traditional linear algorithms

(Duda et al., 2001) in the feature space. They are attractive since the vectors in the feature space are more likely to be linearly separable than those in the input space. Moreover, kernel methods can alleviate the linearity assumption of the original input vectors. Motivated by these, we present a notion of *nonlinear effective dimension reduction*. Subsequently, we develop nonlinear extensions of SIR and SCA, which are referred to as kernel SIR (KSIR) and kernel SCA (KSCA). In order to contrast with FDA and kernel FDA (KFDA), we also refer to SCA and KSCA as *discriminant coordinate analysis* (DCA) and *kernel discriminant coordinate analysis* (KDCA) in the classification setting.

In the existing literature on the kernel extension of FDA, many different approaches have been developed. For example, Baudat and Anouar (2000) and others (Mika et al., 2000, Roth and Steinhage, 2000) extended FDA to KFDA. Recently, Park and Park (2005) proposed a GSVD-based KFDA method. Although there exists an equivalence between FDA and SIR, we present a simple derivation of KSIR using GSVD. From the classification point of view, KSIR and KSCA are able to extract the most discriminating features in the feature space. This is equivalent to extracting the most discriminating nonlinear features in the original input space because KSIR and KSCA utilize high-order statistics of the input space. The computational complexity of GSVD-based KSIR is dependent on the sum of the number of input vectors and the number of slices, while the complexity of KSCA is dependent on the number of slices only. Thus, if the number of input vectors is too large, KSIR becomes computationally infeasible but KSCA is still efficient. There also exist other kernel dimension reduction methods, such as kernel PCA (KPCA) (Schölkopf et al., 1998). KPCA is based on an unsupervised scheme, and its computational complexity is dependent on the number of input vectors. Thus, KPCA becomes computationally expensive as the number of input vectors increases.

The rest of this paper is organized as follows. In Section 2, we present a brief discussion on e.d.r. and the SIR algorithm. In Section 3, we give the detailed procedure of implementing the SCA algorithm. In Section 4, we propose the notion of nonlinear e.d.r. and then derive the kernel SIR and kernel SCA algorithms. In Section 5, we illustrate the applications of SCA and KSCA to classification based on some real-world datasets. The last section gives some concluding remarks.

2. Effective Dimension Reduction and Sliced Inverse Regression

Consider the regression model

$$y = f(\boldsymbol{\eta}_1'\mathbf{x}, \boldsymbol{\eta}_2'\mathbf{x}, \ldots, \boldsymbol{\eta}_q'\mathbf{x}, \epsilon), \tag{1}$$

where $\mathbf{x}$ is a p-dimensional input vector, y is a univariate output variable, $\boldsymbol{\eta}$'s are unknown p-dimensional vectors, ϵ is independent of $\mathbf{x}$ but its distribution is unknown, and f is an arbitrary unknown function. The $(\bullet)'$ is used to denote vector or matrix transpose. We refer to any linear combination of $\boldsymbol{\eta}$'s as an effective dimension reduction (e.d.r.) direction, and the linear space spanned by $\boldsymbol{\eta}$'s as the e.d.r. space. Based on these, Li (1991) presented the following Theorem:

Theorem 1 *Under the regression model (1) and the linear design condition, the centered inverse regression curve* $E(\mathbf{x}|y) - E(\mathbf{x})$ *is contained in the linear space spanned by* $\boldsymbol{\Sigma}_t\boldsymbol{\eta}_j$ $(j = 1, \ldots, q)$*, where* $\boldsymbol{\Sigma}_t$ *is the covariance matrix of* $\mathbf{x}$*.*

Here, the so-called linear design condition says that, for any $\boldsymbol{\beta} \in \mathbb{R}^p$, the conditional expectation $E(\boldsymbol{\beta}'\mathbf{x}|\boldsymbol{\eta}_1'\mathbf{x}, \ldots, \boldsymbol{\eta}_q'\mathbf{x})$ is linear in $\boldsymbol{\eta}_1'\mathbf{x}, \ldots, \boldsymbol{\eta}_q'\mathbf{x}$. This condition is satisfied when the distribution of $\mathbf{x}$ is elliptically symmetric, e.g., the normal distribution. We now use $(\boldsymbol{\eta}_1'\mathbf{x}, \ldots, \boldsymbol{\eta}_q'\mathbf{x})'$ as a new feature vector for $\mathbf{x}$. When q is small, we may achieve the goal of reducing the dimensionality of $\mathbf{x}$ from p to q. Given the data points $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, n)$, the SIR algorithm seeks to estimate $\boldsymbol{\eta}$ via the procedure as given below in Algorithm 1.

Algorithm 1 SIR algorithm

1: **procedure** SIR($\{\mathbf{x}_i, y_i\}_{i=1}^n$, m, $\mathbf{x}$)
2: Divide equally the range of y_i's into m slices, $I_1, \ldots, I_m$. Let n_c be the cardinality of I_c.
3: Calculate the sample mean $\mathbf{u} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$, and each sliced mean $\mathbf{u}_c = \frac{1}{n_c}\sum_{y_i \in I_c} \mathbf{x}_i$ for $c = 1, \ldots, m$.
4: Calculate $\hat{\boldsymbol{\Sigma}}_t = \frac{1}{n}\sum_{i=1}^n (\mathbf{x}_i - \mathbf{u})(\mathbf{x}_i - \mathbf{u})'$, and $\hat{\boldsymbol{\Sigma}}_b = \frac{1}{n}\sum_{c=1}^m n_c(\mathbf{u}_c - \mathbf{u})(\mathbf{u}_c - \mathbf{u})'$.
5: Solve the generalized eigenvalue problem as

$$\hat{\boldsymbol{\Sigma}}_b \boldsymbol{\mu}_i = \lambda_i \hat{\boldsymbol{\Sigma}}_t \boldsymbol{\mu}_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0 \tag{2}$$

and refer to $\boldsymbol{\mu}_i$ as the ith SIR direction.
6: Project $\mathbf{x}$ along the SIR directions to form a q-dimensional new vector $\mathbf{a} = (\boldsymbol{\mu}_1'(\mathbf{x} - \mathbf{u}), \ldots, \boldsymbol{\mu}_q'(\mathbf{x} - \mathbf{u}))'$ with $q \leq \min\{p, m-1\}$.
7: Return $\mathbf{a}$ as the low-dimensional representation of $\mathbf{x}$.
8: **end procedure**

Although the SIR algorithm was originally designed for the regression problem, the inverse scheme behind it has also been applied to the classification problem (Cook and Yin, 2001). Alternatively, we consider a classification problem with J classes. In this case, y is the class label for $\mathbf{x}$ and it only takes one value from $\{1, 2, \ldots, J\}$. Furthermore, we let $m = J$, $\mathbf{u}_c$ be the mean of the cth class, $\hat{\boldsymbol{\Sigma}}_b$ be the between-class covariance matrix, and $\hat{\boldsymbol{\Sigma}}_w$ be the within-class covariance matrix. Then FDA solves the following generalized eigenvalue problem:

$$\hat{\boldsymbol{\Sigma}}_b \mathbf{v} = \gamma \hat{\boldsymbol{\Sigma}}_w \mathbf{v}. \tag{3}$$

Since $\hat{\boldsymbol{\Sigma}}_t = \hat{\boldsymbol{\Sigma}}_b + \hat{\boldsymbol{\Sigma}}_w$, we can rewrite (3) as

$$\hat{\boldsymbol{\Sigma}}_b \mathbf{v} = \frac{\gamma}{1+\gamma} \hat{\boldsymbol{\Sigma}}_t \mathbf{v}. \tag{4}$$

By Eqns. (2) and (4), the SIR variates are the same as the canonical variates except for a scaling factor. In other words, SIR is equivalent to FDA. It is also well-known that FDA relies on the assumption of normality of the input vectors.

3. Sliced Coordinate Analysis

Given the regression model (1), we let $\{\mathbf{b}_1, \ldots, \mathbf{b}_q\}$ be an orthonormal basis of the space spanned by the $\boldsymbol{\Sigma}_t \boldsymbol{\eta}_j$'s. This implies that $\mathbf{b}_j$'s form a q-dimensional e.d.r. space. We start

with approximating each input vector $\mathbf{x}$ by its projection onto this e.d.r. space. That is,

$$\mathbf{x} \approx \sum_{j=1}^{q} a_j \mathbf{b}_j + \mathbf{u}, \tag{5}$$

where the weights a_j form a vector $\mathbf{a} = (a_1, \ldots, a_q)'$ that describes the contribution of each vector in the basis for representing $\mathbf{x}$. The weight vector and weight space are just the feature vector and feature space that we want to obtain. To avoid possible confusion with the same terms used in the kernel literature (Shawe-Taylor and Cristianini, 2004), we still refer to them as weight vector and weight space here. This procedure can also be called *feature transformation*.

SIR is an efficient algorithm for estimating the weight vector. Essentially, SIR first estimates the bases $\mathbf{b}_j$'s using $\mathbf{u}_c$'s and then calculates the weight vector $\mathbf{a}$ for input $\mathbf{x}$. Specifically, from (5), we have

$$a_j = \mathbf{b}_j'(\mathbf{x} - \mathbf{u}), \quad j = 1, \ldots, q.$$

For the mean $\mathbf{u}_c$ of the input within the cth slice, it follows from (5) that

$$\mathbf{u}_c = \sum_{j=1}^{q} w_{cj} \mathbf{b}_j + \mathbf{u}, \quad c = 1, \ldots, m, \tag{6}$$

where $\mathbf{w}_c = (w_{c1}, \ldots, w_{cq})'$ is the weight vector associated with $\mathbf{u}_c$. Based on (6), SIR attempts to perform PCA on the *covariance matrix* for $\mathbf{u}_c$'s to estimate $\mathbf{b}_j$'s.

In this section, we introduce an alternative to computing weight vectors through performing PCO on the *distance matrix* for $\mathbf{u}_c$'s. We call this algorithm *sliced coordinate analysis* (SCA). Unlike SIR, SCA directly estimates the weight vector $\mathbf{w}_c$ associated with $\mathbf{u}_c$ and then calculates the weight vector $\mathbf{a}$ associated with any input $\mathbf{x}$. Computationally, it is similar to the analysis of distance proposed by Gower and Krzanowski (1999), whose aim is to estimate the coordinates of a set of observations when only a distance function between any two such observations is available. In the following, we first obtain the weight vectors for the means $\mathbf{u}_c$'s (Section 3.1) and then that of any input $\mathbf{x}$ (Section 3.2).

3.1 Representation of Means in Weight Space

Let $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]'$ $(m \times p)$ and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]'$ $(m \times q)$. From (6), we have

$$\|\mathbf{u}_c - \mathbf{u}_h\|^2 = \|\mathbf{w}_c - \mathbf{w}_h\|^2, \quad c, h = 1, \ldots, m.$$

As a result, we can obtain an $m \times q$ matrix $\mathbf{D}$ with d_{ch}^2, the squared distance between $\mathbf{w}_c$ and $\mathbf{w}_h$, as the (c, h)th entry. It can be seen readily that

$$d_{ch}^2 = \mathbf{w}_c'\mathbf{w}_c + \mathbf{w}_h'\mathbf{w}_h - 2\mathbf{w}_c'\mathbf{w}_h.$$

This can be expressed in matrix form as:

$$\mathbf{D} = \mathbf{z}\mathbf{1}_m' + \mathbf{1}_m\mathbf{z}' - 2\mathbf{W}\mathbf{W}', \tag{7}$$

where $\mathbf{z}$ is an $m\times 1$ vector containing the diagonal elements of $\mathbf{WW}'$ and $\mathbf{1}_m$ is the $m\times 1$ vector of ones. Here and later, we denote $\mathbf{H}_w = \mathbf{I}_m - \frac{\mathbf{1}_m\mathbf{n}'}{n}$ with $\mathbf{I}_m$ being the $m\times m$ identity matrix and $\mathbf{n} = (n_1, \ldots, n_m)'$. Pre- and post-multiplying (7) by $\mathbf{H}_w$, we have

$$\begin{aligned} -\frac{1}{2}\mathbf{H}_w\mathbf{D}\mathbf{H}_w{}' &= -\frac{1}{2}\mathbf{H}_w\mathbf{z}\mathbf{1}_m'\mathbf{H}_w{}' - \frac{1}{2}\mathbf{H}_w\mathbf{1}_m\mathbf{z}'\mathbf{H}_w' + \mathbf{H}_w\mathbf{W}\mathbf{W}'\mathbf{H}_w' \\ &= \mathbf{H}_w\mathbf{W}\mathbf{W}'\mathbf{H}_w' = \mathbf{W}\mathbf{W}'. \end{aligned} \tag{8}$$

The first two terms on the right-hand side of the first line are zero because $\mathbf{H}_w\mathbf{1}_m = \mathbf{0}$ and $\mathbf{1}_m'\mathbf{H}_w' = \mathbf{0}$, and the second line can be obtained since we assume that the origin of the axes in the weight space is at the weighted centroid of the m weight vectors.

Recall that d_{ch}^2 is also the squared distance between $\mathbf{u}_c$ and $\mathbf{u}_h$. Hence we have

$$d_{ch}^2 = \mathbf{u}_c'\mathbf{u}_c + \mathbf{u}_h'\mathbf{u}_h - 2\mathbf{u}_c'\mathbf{u}_h. \tag{9}$$

Thus, in matrix form,

$$-\frac{1}{2}\mathbf{H}_w\mathbf{D}\mathbf{H}_w' = \mathbf{H}_w\mathbf{U}\mathbf{U}'\mathbf{H}_w' \triangleq \mathbf{\Psi}. \tag{10}$$

Hence, $\mathbf{W}$ can be obtained by performing an eigen-decomposition on either $-\frac{1}{2}\mathbf{H}_w\mathbf{D}\mathbf{H}_w'$ or $\mathbf{H}_w\mathbf{U}\mathbf{U}'\mathbf{H}_w'$, say $\mathbf{\Psi} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}' = (\mathbf{Q}\mathbf{\Lambda}^{1/2})(\mathbf{Q}\mathbf{\Lambda}^{1/2})'$. Recall that the rank of $\mathbf{\Psi}$ is $q \le m$. Thus, we can express matrices $\mathbf{Q}$ and $\mathbf{\Lambda}$ as $\mathbf{Q} = [\mathbf{Q}_1, \mathbf{Q}_2]$ and

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are $m\times q$ and $m\times(m-q)$ matrices, respectively, and $\mathbf{\Lambda}_1$ is a $q\times q$ diagonal matrix with positive elements. Consequently, we can write $\mathbf{\Psi} = \mathbf{Q}_1\mathbf{\Lambda}_1\mathbf{Q}_1'$. Moreover, except for an orthonormal matrix, we have

$$\mathbf{W} = \mathbf{Q}_1{\mathbf{\Lambda}_1}^{1/2}. \tag{11}$$

and hence $(\mathbf{W}'\mathbf{W})^{-1} = \mathbf{\Lambda}_1^{-1}$.

3.2 Interpolation of Inputs in Weight Space

We now consider the problem of calculating the disposition of the associated weight vector $\mathbf{a}$ for an input vector $\mathbf{x}$. Having obtained the weight vectors of the means, the weight vector $\mathbf{a}$ can be added to the diagram by using the technique developed by Gower (1968). Specifically, let $\mathbf{d}_0$ be the m-dimensional vector whose elements are the squared distances from $\mathbf{w}_c$ to the origin of the axes in the weight space, and let $\mathbf{d}$ be the m-dimensional vector whose elements are the squared distances from weight vector $\mathbf{a}$ to each of the weight vectors $\mathbf{w}_c$'s. Then, from (Gower, 1968), the weight vector $\mathbf{a}$ is given by

$$\mathbf{a} = -\frac{1}{2}(\mathbf{W}'\mathbf{W})^{-1}\mathbf{W}'\mathbf{H}_w(\mathbf{d} - \mathbf{d}_0) = -\frac{1}{2}\mathbf{\Lambda}_1^{-1}\mathbf{W}'\mathbf{H}_w(\mathbf{d} - \mathbf{d}_0). \tag{12}$$

Our current problem is to compute $\mathbf{d} = (d_1, \ldots, d_m)'$ and $\mathbf{d}_0 = (d_{01}, \ldots, d_{0m})'$. From (5) and (6), we have

$$\|\mathbf{x} - \mathbf{u}_c\|^2 \approx \sum_{j=1}^{q}(a_j - w_{cj})^2 = \|\mathbf{a} - \mathbf{w}_c\|^2 = d_c,$$

which motivates us to approximate d_c by $\|\mathbf{x} - \mathbf{u}_c\|^2$ for $c = 1, \ldots, m$. Thus, $\mathbf{d}$ is the m-dimensional vector whose cth element d_c is the squared distance from the input $\mathbf{x}$ to the mean $\mathbf{u}_c$. Consequently, we can obtain $\mathbf{d}_0$ and $\mathbf{d}$ as

$$d_{0c} = \|\mathbf{w}_c\|^2 \quad \text{and} \quad d_c = \|\mathbf{x} - \mathbf{u}_c\|^2, \quad c = 1, \ldots, m. \tag{13}$$

The SCA algorithm is summarized in Algorithm 2. We can see that the main computational cost of SCA comes from the eigen-decomposition of $\mathbf{\Psi}$, which is of size $m \times m$. Thus, the computational cost is low. Moreover, the computational procedure is stable, even when $\mathbf{\Psi}$ is singular. It is worth noting that the issue of determining the number of slices has been addressed in the context of SIR and PHD (Schott, 1994, Cook and Yin, 2001). These discussions are also relevant to SCA. In general, it is reasonable for the user to specify the number of slices to be between 10 to 20 for a data set with $n = 300$ observations. In the classification scenario, we also refer to SCA as DCA and set the number of slices as the number of classes. If the number of classes is too small, we can employ multiple sub-means for each class and then apply our algorithm on these sub-means separately. In the following experiments, we concentrate our attention on classification problems where the number of slices is specified as the number of classes.

SCA is based on the notion of effective dimension reduction and the inverse regression setting. Similar to the dual relationship between PCA and PCO, there also exists such a relationship between SIR and SCA. Thus, Theorem 1 also justifies our methods as well as SIR. Moreover, Vempala and Wang (2002) have recently proven that in the expectation, if having m classes, the subspace spanned by the top m singular vectors of the observation matrix is equivalent to the subspace spanned by the m mean vectors.

Algorithm 2 SCA algorithm

1: **procedure** SCA($\{\mathbf{x}_i, y_i\}_{i=1}^n$, m, $\mathbf{x}$)
2: Divide equally the range of y_i's into m slices, $I_1, \ldots, I_m$. Let n_c be the cardinality of I_c.
3: Calculate each sliced mean $\mathbf{u}_c = \frac{1}{n_c} \sum_{y_i \in I_c} \mathbf{x}_i$ for $c = 1, \ldots, m$, and $\mathbf{\Psi} = \mathbf{H}_w \mathbf{U}\mathbf{U}'\mathbf{H}_w'$.
4: Perform eigen-decomposition on $\mathbf{\Psi}$ as $\mathbf{\Psi} = \mathbf{Q}_1 \mathbf{\Lambda}_1 \mathbf{Q}_1'$ and let $\mathbf{W} = \mathbf{Q}_1 \mathbf{\Lambda}_1^{1/2}$.
5: Calculate $\mathbf{d}_0$ and $\mathbf{d}$ from (13), and then $\mathbf{a}$ from (12) for given $\mathbf{x}$.
6: Return $\mathbf{a}$ as the low-dimensional representation of $\mathbf{x}$.
7: **end procedure**

4. Nonlinear Effective Dimension Reduction

Kernel methods (Shawe-Taylor and Cristianini, 2004) work in a feature space $\mathcal{F}$, which is related to the original input space $\mathcal{I} \subset \mathbb{R}^p$ by a mapping,

$$\boldsymbol{\varphi} : \mathcal{I} \to \mathcal{F}.$$

That is, $\boldsymbol{\varphi}$ is a vector-valued function which gives a vector $\boldsymbol{\varphi}(\mathbf{s})$, called a *feature vector*, corresponding to an input $\mathbf{s} \in \mathcal{I}$. In many kernel methods, we are usually given only a Mercer kernel or reproducing kernel $K : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ such that $K(\mathbf{s}, \mathbf{t}) = \boldsymbol{\varphi}(\mathbf{s})'\boldsymbol{\varphi}(\mathbf{t})$ for

$\mathbf{s}, \mathbf{t} \in \mathcal{I}$. The mapping $\boldsymbol{\varphi}(\cdot)$ itself is typically not given explicitly. Rather, there exist only inner products between feature vectors in $\mathcal{F}$. In order to implement a kernel method without referring to $\boldsymbol{\varphi}(\cdot)$ explicitly, one resorts to the so-called *kernel trick*.

Let $L_2(\mathcal{I})$ be the square integrable Hilbert space of functions whose elements are functions defined on $\mathcal{I}$. It is a well-known result that if K is a reproducing kernel for the Hilbert space $L_2(\mathcal{I})$, then $\{K(\cdot, \mathbf{t})\}$ spans $L_2(\mathcal{I})$. Here $K(\cdot, \mathbf{t})$ represents a function that is defined on $\mathcal{I}$ with values at $\mathbf{s} \in \mathcal{I}$ equal to $K(\mathbf{s}, \mathbf{t})$. There are some common kernel functions:

(a) Linear kernel: $K\big(\mathbf{s}, \mathbf{t}\big) = \mathbf{s}'\mathbf{t}$,

(b) Gaussian kernel: $K\big(\mathbf{s}, \mathbf{t}\big) = \exp(-\sum_{j=1}^{p}(s_j - t_j)^2/\beta_j\big)$ with $\beta_j > 0$,

(c) Laplacian kernel: $K\big(\mathbf{s}, \mathbf{t}\big) = \exp(-\sum_{j=1}^{p}|s_j - t_j|/\beta_j\big)$ with $\beta_j > 0$,

(d) Polynomial kernel: $K(\mathbf{s}, \mathbf{t}) = (\mathbf{s}'\mathbf{t} + 1)^k$ of degree k.

Motivated by the idea behind kernel methods, we consider the following regression model instead of that given in (1):

$$y = f(\tilde{\boldsymbol{\eta}}_1'\tilde{\mathbf{x}}, \tilde{\boldsymbol{\eta}}_2'\tilde{\mathbf{x}}, \ldots, \tilde{\boldsymbol{\eta}}_q'\tilde{\mathbf{x}}, \epsilon), \tag{14}$$

where $\tilde{\mathbf{x}}$ is the shorthand for $\boldsymbol{\varphi}(\mathbf{x})$ and $\tilde{\boldsymbol{\eta}}$'s are vectors of the same dimension as $\tilde{\mathbf{x}}$. In the sequel, we use the tilde notation $\tilde{}$ to denote configurations in the feature space. Thus, for our input data $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathcal{I}$, the corresponding feature vectors in the feature space are denoted as $\tilde{\mathcal{X}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n\} \subset \mathcal{F}$. Since there exists a nonlinear mapping between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, we call any linear combination of $\tilde{\boldsymbol{\eta}}$'s a nonlinear effective dimension reduction (e.d.r.) direction, and the space spanned by $\boldsymbol{\eta}$'s a *nonlinear e.d.r. space*. For model (14), the linear design condition is currently required to hold on $\tilde{\mathbf{x}}$. Thus, it is not necessary to satisfy the condition for $\mathbf{x}$. We now perform SIR and SCA over $\{(\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_n, y_n)\}$ giving rise to their nonlinear extensions. We refer to these extensions as kernel SIR (KSIR) and kernel SCA (KSCA), respectively.

4.1 Kernel Sliced Inverse Regression

KSIR seeks to solve the following generalized eigenvalue problem:

$$\tilde{\mathbf{C}}_b\tilde{\boldsymbol{\mu}} = \tilde{\lambda}\tilde{\mathbf{C}}_t\tilde{\boldsymbol{\mu}}, \tag{15}$$

where $\tilde{\mathbf{C}}_t$ and $\tilde{\mathbf{C}}_b$ are the total covariance matrix and the between-slice covariance matrix in $\mathcal{F}$, respectively, i.e.,

$$\begin{aligned}
\tilde{\mathbf{C}}_t &= \frac{1}{n}\sum_{i=1}^{n}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{u}})(\tilde{\mathbf{x}}_i - \tilde{\mathbf{u}})', \\
\tilde{\mathbf{C}}_b &= \frac{1}{n}\sum_{c=1}^{m} n_c(\tilde{\mathbf{u}}_c - \tilde{\mathbf{u}})(\tilde{\mathbf{u}}_c - \tilde{\mathbf{u}})',
\end{aligned}$$

with $\tilde{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{u}}_c = \frac{1}{n_c}\sum_{y_i \in I_c}\tilde{\mathbf{x}}_i$. Let $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]'$ and $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_m]'$. Then we have the kernel matrix $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}'$. To solve (15), we now resort to the kernel

trick to find an equivalent problem that works on $\mathbf{K}$ without involving $\tilde{\mathbf{X}}$. Notice that since there exists an equivalence relationship between SIR and FDA, we can immediately make use of existing methods in the KFDA literature to derive a KSIR method. However, the KFDA method in (Mika et al., 2000) was developed for two-class problems only. The more general method, called generalized discriminant analysis (GDA) (Baudat and Anouar, 2000), requires that the kernel matrix be nonsingular. Unfortunately, centering in the feature space will violate this requirement. Park and Park (2005) argued that this breaks down the theoretical justification for devising GDA and thus proposed the generalized SVD (GSVD) method (Paige and Saunders, 1981) to avoid this requirement for non-singularity. In this paper, along the same line as in Park and Park (2005), we present a simple formulation of KSIR.

Let $\mathbf{G}$ be an $n\times m$ indicator matrix with $g_{ic}=1$ if input $\mathbf{x}_i$ is in slice c and $g_{ic}=0$ otherwise. Denote $\mathbf{N}=\mathrm{diag}(n_1,n_2,\ldots,n_m)$, $\mathbf{n}=(n_1,n_2,\ldots,n_m)'$, $\sqrt{\mathbf{N}}=\mathrm{diag}(\sqrt{n}_1,\sqrt{n}_2,\ldots,\sqrt{n}_m)$, $\sqrt{\mathbf{n}}=(\sqrt{n}_1,\sqrt{n}_2,\ldots,\sqrt{n}_m)'$ and $\mathbf{H}_n=\mathbf{I}_n-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n'$. It thus follows that $\mathbf{1}_n'\mathbf{G}=\mathbf{1}_m'\mathbf{N}=\mathbf{n}'$, $\mathbf{G}\mathbf{1}_m=\mathbf{1}_n$, $\mathbf{1}_m'\mathbf{n}=n$, $\mathbf{G}'\mathbf{G}=\mathbf{N}$, $\mathbf{N}^{-1}\mathbf{n}=\mathbf{1}_m$ and

$$\tilde{\mathbf{U}}=\mathbf{N}^{-1}\mathbf{G}'\tilde{\mathbf{X}}. \tag{16}$$

We rewrite $\tilde{\mathbf{C}}_t$ and $\tilde{\mathbf{C}}_b$ as

$$\tilde{\mathbf{C}}_t=\frac{1}{n}\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{H}_n\tilde{\mathbf{X}}=\frac{1}{n}\tilde{\mathbf{X}}'\mathbf{H}_n\tilde{\mathbf{X}}$$

and

$$\begin{aligned}\tilde{\mathbf{C}}_b &= \frac{1}{n}\tilde{\mathbf{U}}'\left[\sqrt{\mathbf{N}}-\frac{1}{n}\mathbf{n}\sqrt{\mathbf{n}}'\right]\left[\sqrt{\mathbf{N}}-\frac{1}{n}\sqrt{\mathbf{n}}\mathbf{n}'\right]\tilde{\mathbf{U}}\\ &= \frac{1}{n}\tilde{\mathbf{X}}'\mathbf{G}\mathbf{N}^{-1}\left[\sqrt{\mathbf{N}}-\frac{1}{n}\mathbf{n}\sqrt{\mathbf{n}}'\right]\left[\sqrt{\mathbf{N}}-\frac{1}{n}\sqrt{\mathbf{n}}\mathbf{n}'\right]\mathbf{N}^{-1}\mathbf{G}'\tilde{\mathbf{X}}\\ &= \frac{1}{n}\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'\mathbf{H}_n\tilde{\mathbf{X}}.\end{aligned}$$

Here we use the fact that $\mathbf{G}\mathbf{N}^{-1}\left[\sqrt{\mathbf{N}}-\frac{1}{n}\mathbf{n}\sqrt{\mathbf{n}}'\right]=\mathbf{G}\sqrt{\mathbf{N}}^{-1}-\frac{1}{n}\mathbf{1}_n\sqrt{\mathbf{n}}'$, $\mathbf{H}_n\mathbf{G}\sqrt{\mathbf{N}}^{-1}=\mathbf{G}\sqrt{\mathbf{N}}^{-1}-\frac{1}{n}\mathbf{1}_n\sqrt{\mathbf{n}}'$ and $\sqrt{\mathbf{N}}^{-1}\sqrt{\mathbf{N}}^{-1}=\mathbf{N}^{-1}$. Now, we can reformulate the problem (15) as

$$\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'\mathbf{H}_n\tilde{\mathbf{X}}\tilde{\boldsymbol{\mu}}=\tilde{\lambda}\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{H}_n\tilde{\mathbf{X}}\tilde{\boldsymbol{\mu}}. \tag{17}$$

On the other hand, since the eigenvectors are in the space spanned by $\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_n$ (refer to (Mika et al., 2000, Park and Park, 2005) for more detailed explanation), we express $\tilde{\boldsymbol{\mu}}$ as

$$\tilde{\boldsymbol{\mu}}=\sum_{i=1}^{n}\beta_i(\tilde{\mathbf{x}}_i-\tilde{\mathbf{u}})=\tilde{\mathbf{X}}'\mathbf{H}_n\boldsymbol{\beta}, \tag{18}$$

where $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_n)'$ is an $n\times 1$ coefficient vector. Hence, (17) is equivalent to

$$\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'\mathbf{H}_n\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}_n\boldsymbol{\beta}=\tilde{\lambda}\tilde{\mathbf{X}}'\mathbf{H}_n\mathbf{H}_n\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}_n\boldsymbol{\beta}.$$

Pre-multiplying the equation by $\mathbf{H}_n\tilde{\mathbf{X}}$, we have a new generalized eigenvalue problem

$$\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'\mathbf{H}_n\mathbf{K}\mathbf{H}_n\boldsymbol{\beta} = \tilde{\lambda}\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{H}_n\mathbf{K}\mathbf{H}_n\boldsymbol{\beta}, \tag{19}$$

which involves the kernel matrix $\mathbf{K}$ rather than $\tilde{\mathbf{X}}$. Moreover, given a new input vector $\mathbf{x}$, we can compute the projection of its feature vector $\tilde{\mathbf{x}}$ onto $\tilde{\boldsymbol{\mu}}$ through

$$(\tilde{\mathbf{x}} - \tilde{\mathbf{u}})'\tilde{\boldsymbol{\mu}} = \Big(\tilde{\mathbf{x}} - \frac{1}{n}\tilde{\mathbf{X}}'\mathbf{1}_n\Big)'\tilde{\mathbf{X}}'\mathbf{H}_n\boldsymbol{\beta} = \Big(\mathbf{k}_\mathbf{x} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\Big)'\mathbf{H}_n\boldsymbol{\beta}, \tag{20}$$

where $\mathbf{k}_\mathbf{x} = \big(K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_n)\big)'$. This shows that the kernel trick can be used for KSIR. Our current concern is to solve the problem (19). Although $\mathbf{K}$ is assumed to be non-singular, $\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{H}_n\mathbf{K}\mathbf{H}_n$ is positive semi-definite but not positive definite because the centering matrix $\mathbf{H}_n$ is singular. In fact, the rank of $\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{H}_n\mathbf{K}\mathbf{H}_n$ is not larger than $n-1$ because the rank of $\mathbf{H}_n$ is $n-1$. In this case, the method devised in (Baudat and Anouar, 2000) cannot be used for the problem (19). Alternatively, we resort to GSVD to solve this problem, with the detailed procedure given in Algorithm 3. Detailed derivation for the implementation of GSVD can be found in (Howland et al., 2003). Since the rank of $\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'$ is $m-1$, the rank of $\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{G}\mathbf{N}^{-1}\mathbf{G}'\mathbf{H}_n\mathbf{K}\mathbf{H}_n$ is not larger than $m-1$. This implies that we can at most obtain the $q = m-1$ e.d.r. directions, giving q $\boldsymbol{\beta}$'s. For our problem given in (19), running GSVD requires the complete orthogonal decomposition of matrix $\mathbf{Z} = [\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{G}\sqrt{\mathbf{N}}^{-1}, \mathbf{H}_n\mathbf{K}\mathbf{H}_n]'$, which is of size $(n+m)\times n$. Thus, when n is large, the computational cost is expected to be expensive.

Algorithm 3 GSVD-based KSIR algorithm

1: **procedure** KSIR($\{\mathbf{x}_i, y_i\}_{i=1}^n$, m, $\mathbf{x}$, "kernel function")
2: Divide equally the range of y_i's into m slices, $I_1,\ldots,I_m$, and assign the indicator matrix $\mathbf{G}$ $(n\times m)$. Let n_c be the cardinality of I_c and $\mathbf{N} = \mathsf{diag}(n_1,\ldots,n_m)$.
3: Calculate $\mathbf{K} = [K(\mathbf{x}_i,\mathbf{x}_j)]_{i,j=1}^n$ and $\mathbf{k}_\mathbf{x} = \big(K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_n)\big)'$.
4: Calculate $\mathbf{Z} = [\mathbf{H}_n\mathbf{K}\mathbf{H}_n\mathbf{G}\sqrt{\mathbf{N}}^{-1}, \mathbf{H}_n\mathbf{K}\mathbf{H}_n]'$ $((n+m)\times n)$.
5: Compute the orthogonal-triangular decomposition of $\mathbf{Z}$, which is

$$\mathbf{P}'\mathbf{Z}\mathbf{Q} = \begin{array}{c} \\ t \\ n+m-t \end{array}\!\!\begin{array}{c} \begin{array}{cc} t & n-t \end{array} \\ \begin{pmatrix} \mathbf{R} & 0 \\ 0 & 0 \end{pmatrix} \end{array} \quad \text{with } \mathbf{R} = [r_{ij}] \text{ and } |r_{11}| \geq |r_{22}| \geq |r_{tt}| > 0.$$

6: Perform SVD of $\mathbf{P}(1:m, 1:t)$ as $\mathbf{P}(1:m,1:t) = \mathbf{E}\mathbf{S}\mathbf{V}'$.
7: Compute $\mathbf{B} = \mathbf{Q}\begin{pmatrix} \mathbf{R}^{-1}\mathbf{V} & 0 \\ 0 & \mathbf{I}_{n-t} \end{pmatrix}$ and set $\mathbf{F} = \mathbf{B}(:, 1:m-1)$.
8: Return $\tilde{\mathbf{a}} = \mathbf{F}'\mathbf{H}_n\big(\mathbf{k}_\mathbf{x} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\big)$ as the low-dimensional representation of $\mathbf{x}$.
9: **end procedure**

4.2 Kernel Sliced Coordinate Analysis

In Section 3, SCA was developed over the input space. Similar to KSIR, we propose KSCA in this subsection. As in other kernel methods, the idea is to first map the input space into a feature space and then apply SCA in this feature space.

Let $\tilde{\mathbf{W}}$ be the weight matrix associated with $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_m]'$ and assume that the column of $\tilde{\mathbf{W}}$ is centered. It is straightforward to extend (12) in Section 3 to the feature space. That is, we can calculate the associated weight vector $\tilde{\mathbf{a}}$ of $\mathbf{x}$ by

$$\tilde{\mathbf{a}} = -\frac{1}{2}(\tilde{\mathbf{W}}'\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}'\mathbf{H}_w(\tilde{\mathbf{d}} - \tilde{\mathbf{d}}_0). \tag{21}$$

Here $\tilde{\mathbf{d}} - \tilde{\mathbf{d}}_0 = (\tilde{d}_1 - \tilde{d}_{01}, \ldots, \tilde{d}_m - \tilde{d}_{0m})'$ where $\tilde{d}_c$ is approximated by the squared distance between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}_c$, and $\tilde{d}_{0c}$ is the squared distance between the origin of the axes in the weight space and $\tilde{\mathbf{w}}_c$, for $c = 1, \ldots, m$.

In order to calculate $\tilde{\mathbf{a}}$, we seek to calculate $\tilde{\mathbf{W}}$, $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{d}}_0$ first. First, similar to (8) and (10), we have

$$\tilde{\mathbf{W}}\tilde{\mathbf{W}}' = \mathbf{H}_w\tilde{\mathbf{U}}\tilde{\mathbf{U}}'\mathbf{H}_w'.$$

It follows from (16) that

$$\tilde{\mathbf{U}}\tilde{\mathbf{U}}' = \mathbf{N}^{-1}\mathbf{G}'\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{G}\mathbf{N}^{-1} = \mathbf{N}^{-1}\mathbf{G}'\mathbf{K}\mathbf{G}\mathbf{N}^{-1},$$

which leads to

$$\tilde{\mathbf{W}}\tilde{\mathbf{W}}' = \mathbf{H}_w\mathbf{N}^{-1}\mathbf{G}'\mathbf{K}\mathbf{G}\mathbf{N}^{-1}\mathbf{H}_w'.$$

Second, with the kernel trick, for $c = 1, \ldots, m$, we have

$$\begin{aligned}
\tilde{d}_c &= \|\tilde{\mathbf{x}} - \tilde{\mathbf{u}}_c\|^2 = \tilde{\mathbf{x}}'\tilde{\mathbf{x}} - 2\tilde{\mathbf{x}}'\tilde{\mathbf{u}}_c + \tilde{\mathbf{u}}_c'\tilde{\mathbf{u}}_c \\
&= \tilde{\mathbf{x}}'\tilde{\mathbf{x}} - \frac{2}{n_c}\sum_{y_i \in I_c}\tilde{\mathbf{x}}'\tilde{\mathbf{x}}_i + \frac{1}{n_c^2}\sum_{y_i, y_j \in I_c}\tilde{\mathbf{x}_j}'\tilde{\mathbf{x}}_i \\
&= K(\mathbf{x}, \mathbf{x}) - \frac{2}{n_c}\sum_{y_i \in I_c}K(\mathbf{x}, \mathbf{x}_j) + \frac{1}{n_c^2}\sum_{y_i, y_j \in I_c}K(\mathbf{x}_i, \mathbf{x}_j),
\end{aligned}$$

and hence,

$$\tilde{d}_c - \tilde{d}_{0c} = \tilde{d}_c - \tilde{\mathbf{w}}_c'\tilde{\mathbf{w}}_c, \quad c = 1, \ldots, m. \tag{22}$$

We now summarize the KSCA procedure in Algorithm 4. We can see that $\tilde{\mathbf{W}}$, $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{d}}_0$ are computed by using $\mathbf{K}$ instead of $\tilde{\mathbf{X}}$. Moreover, in order to obtain $\tilde{\mathbf{W}}$, we are only required to perform SVD on the $m \times m$ matrix $\mathbf{H}_w\mathbf{N}^{-1}\mathbf{G}'\mathbf{K}\mathbf{G}\mathbf{N}^{-1}\mathbf{H}_w'$. The computational complexity is $O(m^3)$. However, the complexity of KSIR (or KFDA) is larger than $O(n^2(m + 2n/3))$ because it needs to perform QR decomposition on the $(n+m)\times n$ matrix $\mathbf{Z}$.

It is worth noting that there exists a one-to-one relationship between an observation in the original input space $\mathbb{R}^p$ and a weight vector in the weight space $\mathbb{R}^q$. Accordingly, the weight vector $\mathbf{a}_i$ (or $\tilde{\mathbf{a}}_i$) may then be used as a new feature for $\mathbf{x}_i$. More specifically, we form a new set of training data $\{\mathbf{a}_i, \mathbf{y}_i\}_{i=1}^n$ or $\{\tilde{\mathbf{a}}_i, \mathbf{y}_i\}_{i=1}^n$. In the regression setting, the new training set is subsequently used to train any suitable regression model. In the classification setting, one commonly uses a nearest mean classifier to assign a label to $\mathbf{x}$, namely,

$$y = \arg\min_j\{\|\mathbf{a} - \boldsymbol{\omega}_j\|, j = 1, \ldots, m\}$$

where $\boldsymbol{\omega}_j$ is the jth column of $\mathbf{A}'\mathbf{G}\mathbf{N}^{-1}$ with $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]'$. Alternatively, we may also use $\{\mathbf{a}_i, \mathbf{y}_i\}_{i=1}^n$ or $\{\tilde{\mathbf{a}}_i, \mathbf{y}_i\}_{i=1}^n$ to train other kernel-based classifiers such as a support vector machine (SVM). Figure 1 illustrates the whole procedure for classification purpose.

Algorithm 4 KSCA algorithm

procedure KSCA($\{\mathbf{x}_i, y_i\}_{i=1}^n$, m, $\mathbf{x}$, "kernel function")
  Divide equally the range of y_i's into m slices, $I_1, \ldots, I_m$, and assign the indicator matrix $\mathbf{G}$ $(n \times m)$. Let n_c be the cardinality of I_c and $\mathbf{N} = \mathsf{diag}(n_1, \ldots, n_m)$.
  Calculate $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n$, $\mathbf{k}_\mathbf{x} = \big(K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)\big)'$ and $\tilde{\mathbf{\Psi}} = \mathbf{H}_w \mathbf{N}^{-1} \mathbf{G}' \mathbf{K} \mathbf{G} \mathbf{N}^{-1} \mathbf{H}_w'$.
  Perform eigen-decomposition on $\tilde{\mathbf{\Psi}}$ as $\tilde{\mathbf{\Psi}} = \tilde{\mathbf{Q}}_1 \tilde{\mathbf{\Lambda}}_1 \tilde{\mathbf{Q}}_1'$ and let $\tilde{\mathbf{W}} = \tilde{\mathbf{Q}}_1 \tilde{\mathbf{\Lambda}}_1^{1/2}$.
  Compute $\tilde{\mathbf{d}} - \tilde{\mathbf{d}}_0$ from (22), and then $\tilde{\mathbf{a}}$ from (21) for given $\mathbf{x}$.
  Return $\tilde{\mathbf{a}}$ as the low-dimensional representation of $\mathbf{x}$.
end procedure

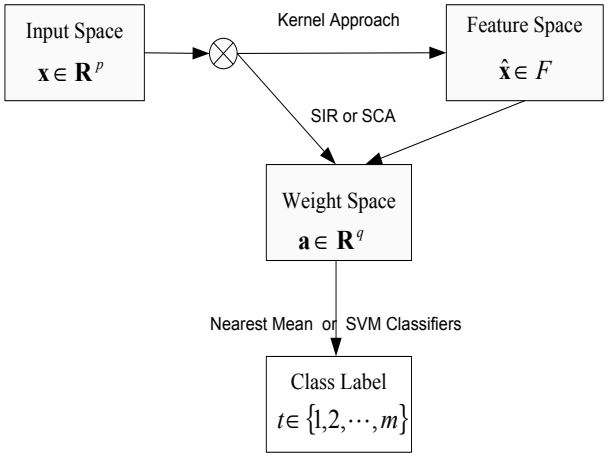

Figure 1: Schematic diagram of using weight vectors in classification.

Table 1: Summary of the Datasets: n—the size of the training set; k—the size of the test set; p—the dimension of the input vector; m—the number of slices (or classes); q—the dimensionality after reduction.

Datasets	n	k	p	m	q
AT&T	200	200	112×92	40	39
Yale face	90	75	128×128	15	14
2K-image	1328	569	144	14	13
USPS	4649	4649	256	10	9

5. Experiments

In this section, we illustrate the applications of SCA and KSCA for classification, and compare them with PCA and KPCA as well as SIR and KSIR. For the kernel methods, we adopt the Gaussian RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\beta^2}\big)$ and the Laplacian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\sum_{l=1}^{p} \frac{|x_{il} - x_{jl}|}{\beta}\big)$ where β is taken as the product of a positive coefficient ϵ and the average distance between slice means in the training data. We find that if the value of ϵ is taken from the interval $[0.5, 1.5]$, there is little influence on the algorithms. In the following experiments, we set $\beta = 0.9$ and m, the number of slices, as the number of classes. Table 1 gives a summary of the datasets that will be used.

5.1 Application to Face Recognition

Using the publicly available AT&T and Yale face image data sets, we compared SCA (KSCA) with PCA (KPCA) and FDA (KFDA). We first used these methods for feature transformation to generate a set of low-dimensional weight vectors. After that, the weight vectors, acting as new feature vectors, were given to both a nearest mean (NM) classifier and an SVM for training and testing. The AT&T data set contains 400 images of 40 subjects, with variations mainly due to the scale and pose of the subjects. Each image consists of 112×92 pixels, i.e., $p = 112\times92$. The Yale data set contains 165 images of 15 subjects, with variations mainly due to facial expression and lighting. Each image consists of 128×128 pixels, i.e., $p = 128\times128$. Each subject is considered as a class (or slice). Now we let the number of classes equal to the number of slices, i.e., $m = 40$ for the AT&T data set and $m = 15$ for the Yale data set. Notice that the SIR and PCA methods were not directly performed, because p is too high. Instead, we regard PCA and FDA as special cases of KPCA and KFDA, respectively, with the linear kernel. In this case, PCA and FDA are called Eigenface (Turk and Pentland, 1991) and Fisherface (Belhumeur et al., 1997), respectively, in the face recognition literature.

We randomly split the images for each subject into two subsets, one for training and the other for testing. The classification accuracies, based on NM and SVM, were estimated from 50 random splits. For the AT&T data set, 200 of the 400 images were used for training and the remaining 200 for testing. For the Yale data set, 90 of the 165 images were used for training and the remaining 75 for testing. The split was randomly repeated 50 times and the classification accuracies were then averaged. All the experiments were performed

Table 2: Recognition results for the AT&T database using nearest mean classifier.

Method	Linear Kernel		Gaussian Kernel	
	Accuracy (%)	CPU time (s)	Accuracy (%)	CPU time (s)
KSCA	**90.90** (±2.36)	**0.0107**	92.96 (±2.00)	**0.0140**
KFDA	88.64 (±2.80)	0.1324	**93.38** (±2.69)	0.1502
KPCA	89.39 (±2.61)	0.1088	85.65 (±2.68)	0.4676

Table 3: Recognition results for the Yale database using nearest mean classifier.

Method	Linear Kernel		Gaussian Kernel	
	Accuracy (%)	CPU time (s)	Accuracy (%)	CPU time (s)
KSCA	82.91 (±3.25)	**0.0032**	85.56 (±3.80)	**0.0040**
KFDA	**94.75** (±2.45)	0.0168	**95.79** (±3.65)	0.0206
KPCA	78.13 (±3.44)	0.0158	78.49 (±3.75)	0.0187

in Matlab on a Pentium 4 PC with 2.66GHz CPU and 1.50GB of RAM. We used the SVM-light (`http://www.kernel-machines.org/`) package with one-per-class (OPC) ensemble strategy for SVM and set the parameter $C = 1000$.

The computational complexities of both (K)PCA and (K)FDA are $O(n^3)$, i.e., $O(200^3)$ for AT&T and $O(90^3)$ for Yale. On the other hand, the complexity of (K)SCA is $O(m^3)$, i.e., $O(40^3)$ for AT&T and $O(15^3)$ for Yale. Therefore, (K)SCA is more efficient than (K)PCA and (K)SIR. Tables 2 and 3 show the CPU time of different dimension reduction (DR) methods for the AT&T and Yale data sets. Since these methods all take the same amount of time to compute the kernel matrix, we do not include the time of computing the kernel matrix in our results. After obtaining the new features with (K)PCA, (K)FDA and (K)SCA, we used NM and SVM for the classification target. When a DR method with the linear kernel was used, we performed a Gaussian-kernel SVM. Otherwise, we performed a linear-kernel SVM because the Gaussian kernel was already used in KPCA, KFDA and KSCA. At the same time, we implemented a Gaussian-kernel SVM on the original face data sets for baseline comparison. Tables 2, 3 and 4 list the classification accuracies and the corresponding standard deviations. From the results, the KFDA classifier often achieves the lowest recognition error rate. However, it takes a long time. The KSCA classifier ranks second in terms of the error rate, but it requires much less processing time. KSCA and KFDA utilize the class label information during training, whereas KPCA does not. This is the main reason why KSCA and KFDA outperform KPCA.

5.2 Application to Image Classification

We applied our methods to two relatively large image data sets: *2K-image data set* and *USPS data set*. The 2K-image data set was collected from the Corel Image CDs. This data set contains 2K, or exactly 1897, representative images from fourteen categories (m=14): *architecture, bears, clouds, elephants, fabrics, fireworks, flowers, food, landscape, people, textures, tigers, tools* and *waves*. Each image is represented by a vector of 144 dimensions

Table 4: Classification accuracies for both the AT&T and Yale data sets. "RBF" ("LIN") means that the Gaussian (linear) kernel is used in DR or SVM.

DR method + Classifier	AT&T	Yale
Original Data + RBF-SVM	96.27 (±1.78)	94.49 (±2.42)
LIN-KPCA + RBF-SVM	96.07 (±1.71)	81.42 (±3.90)
LIN-KFDA + RBF-SVM	88.03 (±2.78)	**95.29** (±2.50)
LIN-KSCA + RBF-SVM	**96.50** (±1.38)	86.31 (±3.11)
RBF-KPCA + LIN-SVM	93.27 (±1.84)	83.87 (±4.15)
RBF-KFDA + LIN-SVM	93.45 (±2.08)	**96.27** (±2.70)
RBF-KSCA + LIN-SVM	**95.25** (±)1.54	87.24 (±3.55)

including color, texture, and shape features (Tong and Chang, 2001). The experimental results were evaluated over 30 random splits of the data set, with 70% for training and 30% for testing. The USPS data set contains 9298 handwritten digits from 0 to 9. Each digit consists of 16×16 pixels. We treat each digit as a class. In this case, $m = 10$ and $p = 256$. The experimental results were also evaluated over 30 random splits of the data set, with 50% for training and 50% for testing.

We first performed PCA, FDA (SIR) and DCA (SCA) to reduce the dimensionality of the image from 144 (256) to 13 (9) due to $q = m-1$. Second, we applied the NM and SVM classifiers to the reduced images. In SVM, we used the Laplacian kernel for the 2K image data and the Gaussian kernel for the USPS data. Tables 5 and 6 show the CPU time of running these methods for dimension reduction, and the classification accuracies for NM and SVM. The computational time that DCA needs is the lowest. For the 2K image data, we used GSVD to solve the generalized eigenvalue problem (2) because the sample covariance matrix $\hat{\mathbf{\Sigma}}_t$ is singular. The FDA-based SVM classifier achieves the highest classification accuracy, while the performance of the SCA-based SVM is comparatively better. We also applied KPCA, KFDA and KDCA to USPS with the Laplacian kernel for the 2K image data and the RBF kernel for the USPS data. For the USPS data set, since n is too large, we instead ran these methods in Matlab on an 8 × Sun Microsystems Ultra-SPARC III 900MHz CPU, each with 8MB E-Cache and 8GB RAM. It took 6.1982×10^3 and 1.0632×10^4 seconds to run KFDA and KPCA, respectively, one time. However, it only took about 4 seconds to implement KDCA one time.

6. Conclusion

In this paper, we proposed the sliced coordinate analysis method and its kernel version to reduce the dimension of the input vector in regression and classification problems. For many image and video applications, FDA and kernel FDA are computationally infeasible if the size of the image set is large and/or the resolution of the image is high. However, our proposed SCA and KSCA methods can still proceed because the number of classes is typically much smaller than the size of the image set or the resolution of the image. For unsupervised learning problems, we can first cluster the data into several classes so that our

Table 5: Experimental results for the 2K image data set. "LAP-SVM" means that the Laplacian kernel is used in SVM.

Method	CPU time (s)	Accuracy (%)	
		NM	LAP-SVM
PCA	0.6813	57.78 (±0.43)	63.87 (±0.30)
FDA	4.0062	**71.80** (±0.81)	**72.82** (±0.35)
DCA	**0.0344**	62.89 (±0.42)	67.32 (±0.32)

Table 6: Experimental results for the USPS data set. "RBF-SVM" means that the RBF kernel is used in SVM.

Method	CPU time (s)	Accuracy (%)	
		NM	RBF-SVM
PCA	8.2901	78.50 (±0.43)	91.41 (±0.30)
FDA	22.5109	**89.90** (±0.81)	92.02 (±0.35)
DCA	**0.2276**	84.08 (±0.42)	**92.64** (±0.32)

methods can still work well. Therefore, we expect our methods to have many applications in machine learning and pattern recognition, especially for kernel methods applied to data sets of large sizes.

Acknowledgments

This research has been supported by two Competitive Earmarked Research Grants (CERG), HKUST6174/04E and HKUST6195/02E, from the Research Grants Council of the Hong Kong Special Administrative Region, China.

References

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12:2385–2404, 2000.

P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. PAMI*, 19(7):711–720, 1997.

R. D. Cook. Principal Hessian directions revisited. *Journal of the American Statistical Association*, 93:84–94, 1998.

R. D. Cook and H. Lee. Dimension reduction in regression with a binary response. *Journal of the American Statistical Association*, 94:1187– 1200, 1999.

Table 7: Experimental results for the 2K image data set using the kernel methods.

Method	CPU time (s)	Accuracy (%)	
		NM	LIN-SVM
KPCA	87.8531	60.06 (±1.62)	71.06 (±1.77)
KFDA	41.4323	**84.55** (±0.99)	**86.02** (±1.08)
KDCA	**1.3141**	70.56 (±1.68)	77.48 (±1.52)

R. D. Cook and Weisberg. Discussion of Li (1991). *Journal of the American Statistical Association*, 86:328–332, 1991.

R. D. Cook and X. Yin. Dimension reduction and visualization in discriminant aanalysis. *Australian & New Zealand Journal of Statistics*, 43:147–199, 2001.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, second edition, 2001.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.

J. C. Gower. Some distance properties of latent root and vector methods used in multivariate data analysis. *Biometrika*, 53:315–328, 1966.

J. C. Gower. Adding a point to vector diagrams in multivariate analysis. *Biometrika*, 55: 582–585, 1968.

J. C. Gower and W. J. Krzanowski. Analysis of distance for structured multivariate data and extensions to multivariate analysis of variance. *Journal of the Royal Statistial Society Series C*, 48:505–519, 1999.

T. Hastie, R. Tishiran, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

P. Howland, M. Jeon, and H. Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

I.T. Jolliffe. *Principal component analysis*. Springer, New York, second edition edition, 2002.

K. C. Li. Sliced inverse regression for dimension reduction (with discussions). *Journal of the American Statistical Association*, 86:316–342, 1991.

K. C. Li. On principal Hessian direction for data visualization and dimension reduction: another application of Stein's lemma. *Journal of the American Statistical Association*, 87:1025–1039, 1992.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, New York, 1979.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K. R. Müller. Invariant feature extraction and classification in kernel space. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 526–532, 2000.

C. C. Paige and M. A. Saunders. Towards a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, 18(3):398–405, 1981.

C. H. Park and H. Park. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 27(1):87–102, 2005.

V. Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 568–574, 2000.

B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, 2002.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

J. R. Schott. Determining the dimensionality in sliced inverse regression. *Journal of the American Statistical Association*, 89:141–148, 1994.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of ACM International Conference on Multimedia*, pages 107–118, 2001.

M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1): 71–86, 1991.

S. Vempala and G. Wang. A spectral algorithm for learning mixtures of distributions. In *Proceedings of the 43rd IEEE Foundations of Computer Science*, 2002.