Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 5927306, 18 pages
http://dx.doi.org/10.1155/2016/5927306

Research Article

A Multikernel-Like Learning Algorithm Based on Data Probability Distribution

Guo Niu,[1] Zhengming Ma,[1] and Shuyu Liu[2]

[1]*School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China*
[2]*Public Experimental Teaching Center, Sun Yat-sen University, Guangzhou 510006, China*

Correspondence should be addressed to Shuyu Liu; lsy@mail.sysu.edu.cn

Received 22 February 2016; Accepted 9 May 2016

Academic Editor: Marco Perez-Cisneros

In the machine learning based on kernel tricks, people often put one variable of a kernel function on the given samples to produce the basic functions of a solution space of learning problem. If the collection of the given samples deviates from the data distribution, the solution space spanned by these basic functions will also deviate from the real solution space of learning problem. In this paper a multikernel-like learning algorithm based on data probability distribution (MKDPD) is proposed, in which the parameters of a kernel function are locally adjusted according to the data probability distribution, and thus produces different kernel functions. These different kernel functions will generate different Reproducing Kernel Hilbert Spaces (RKHS). The direct sum of the subspaces of these RKHS constitutes the solution space of learning problem. Furthermore, based on the proposed MKDPD algorithm, a new algorithm for labeling new coming data is proposed, in which the basic functions are retrained according to the new coming data, while the coefficients of the retrained basic functions remained unchanged to label the new coming data. The experimental results presented in this paper show the effectiveness of the proposed algorithms.

1. Introduction

(a) Data Spaces and Data Distributions. Let X represent the data and Ω the data space. In mathematics, the data X can be regarded as a random variable/vector/matrix defined on the data space Ω. There will be different kinds of data on a data space. For example, the data space Ω can be the one consisting of all images of 512×512 pixels, while the data X_f represents all face images of 512×512 pixels and the data X_l represents all landscape images of 512×512 pixels; both of them are defined on the data space Ω but subject to different probability distributions.

If the data is regarded as a random variable, the samples of data can be then regarded as the concrete realization of the random variable. In machine learning, the samples of data can be exploited to estimate the probability distribution of data (the probabilistic modeling of data). There are a lots of researches on the probabilistic modeling of data [1–3].

(b) Classification/Labels and Classifiers/Label Functions. The classification of the data may be different when the data are used in different applications. For example, let the data be the face images. In the application of identification recognition, the face images of the same person are all grouped into the same class, even though the expressions and postures of these face images may be different. However, in the application of expression recognition, the face images of the same expression are all grouped into the same class, even though these face images belong to different persons.

The classifier of data is a machine indicating the class to which a data point belongs. The classifiers of data are trained with the data samples [4, 5]. The classes of data are also called the labels of data and the classifiers of data are also called the label functions of data. In this paper, we adopt the terminology of labels and label functions of data.

(c) Kernel Tricks in Machine Learning. The applications of kernel tricks in machine learning can be roughly divided into two categories: the transformation of data spaces and the construction of label functions. In the transformation of data spaces, the kernel functions are used to transform data spaces into other spaces where the data can be linearly separated.

The famous Kernel PCA [6] and kernel Fisher discriminant (KFD) [7] belong to this category. In the construction of label functions, the kernel functions are used to serve as the basic functions of label functions. The famous manifold regularization learning [8, 9] belongs to this category. In this paper we address the problems involved in the construction of label functions.

In the construction of label functions, the label functions are expressed as $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x, x_i)$, where $k(x, v)$ represents a kernel function, $\{x_1 \cdots x_b\}$ the labeled samples, and $\{x_{b+1} \cdots x_{b+u}\}$ the unlabeled samples. The coefficients $\vec{\alpha} = [\alpha_1 \cdots \alpha_{b+u}]$ can be derived from solving the following learning problem:

$$\begin{aligned}\vec{\alpha}^* &= \arg\min_{f \in H_X} \sum_{i=1}^{b} V(y_i, f(x_i)) + \kappa \|f\|_{H_X}^2 \\ &= \arg\min_{\vec{\alpha} \in R^{b+u}} \sum_{i=1}^{b} V\left(y_i, \sum_{j=1}^{b+u} \alpha_i k(x_i, x_j)\right) + \kappa \vec{\alpha}^T K_X \vec{\alpha}.\end{aligned} \tag{1}$$

In the above equation, $H_X = \text{span}\{k(x, x_i) \mid i = 1, \ldots, b + u\}$ represents the solution space of the learning problem; $\{y_1 \cdots y_b\}$ represents the labels of the labeled samples $\{x_1 \cdots x_b\}$;

$$K_X = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_{b+u}) \\ \vdots & \ddots & \vdots \\ k(x_{b+u}, x_1) & \cdots & k(x_{b+u}, x_{b+u}) \end{bmatrix} \tag{2}$$

represents the kernel matrix; and $V(y_i, f(x_i))$ represents the cost function. We want to find a label function $f(x)$ which will make the cost as small as possible.

There are two kinds of properties about the data. The first kind of properties is the natural properties of data. The probability distributions of data are the examples of natural properties of data. The second kind of properties is the semantic properties of data. The data labels are the examples of semantic properties of data. It is clear that, in the framework of learning problem shown in (1), the semantic properties hidden in the data labels have been fully utilized, while the natural properties hidden in data probability distributions seem not to be deeply exploited. At present, the usual way to learn more information other than the data labels is to add various regularization terms to the cost function. For example, in the manifold regularization learning [8–11], a manifold regularization term is added to the cost function:

$$f^* = \arg\min_{f \in H_X} \sum_{i=1}^{b} V(y_i, f(x_i)) + \kappa \|f\|_{H_X}^2 + \lambda \|f\|_M^2, \tag{3}$$

where $\|f\|_M^2 = \vec{f}_X^T L_X \vec{f}_X$ is the so-called manifold regularization term, in which $\vec{f}_X = [f(x_1) \cdots f(x_{b+u})]$ and L_X is the Laplacian matrix reflecting the adjacency relations of data samples [12].

However, the addition of too many regularization terms to the cost function will make the learning problem complicated and difficult to solve. In this paper, rather than adding regularization terms, an alternative way to exploit the data probability distribution in the learning problem is proposed. For the convenience of description, let us denote the kernel function as $k(x, z \mid \theta)$, where θ represents the parameter of the kernel function. In the proposed algorithm, the parameters of the basic functions $k(x, x_i \mid \theta_i)$ are adjusted based on the data distribution sample-by-sample; that is, $\theta_i = \theta(p(x_i))$, where $p(x)$ is the probability distribution of data, $i = 1, \ldots, b+u$. These basic functions are then used to span the solution space of the learning problem: $H_D = \text{span}\{k(x, x_i \mid \theta_i) \mid i = 1, \ldots, b+u\}$. It is clear that the probability distribution of data is integrated into the basic functions.

According to the theory of kernel functions, if $\theta_i \neq \theta_j$, then $k(x, z \mid \theta_i)$ and $k(x, z \mid \theta_j)$ are two different kernel functions and will generate two different RKHS. Now let H_i denote the RKHS generated by the kernel function $k(x, z \mid \theta_i)$, $i = 1, \ldots, b + u$; then, theoretically speaking, the solution space H_D can be regarded as a subspace of the direct sum space $H_1 \oplus \cdots \oplus H_{b+u}$. Therefore, the proposed algorithm can be regarded as a kind of multikernel learning algorithm, but quite different from the commonly used multikernel learning algorithm [13–16]. Therefore, we call the proposed algorithm the multikernel-like learning algorithm based on data probability distribution, referred to as MKDPD algorithm.

How to label the new coming data (the out-of-samples) x is the key topic in machine learning [12, 17, 18]. There are two extreme algorithms. One algorithm uses the original label function f_{old} to label the new coming data x; that is, the labels of the new coming data x are given by $f_{\text{old}}(x)$. This algorithm is best in efficiency, but worst in accuracy. Another algorithm regards the new coming data x as the unlabeled samples and mixes the new coming data x with the original samples to retrain a new label function f_{new}. The labels of the new coming data x are then given by $f_{\text{new}}(x)$. This latter algorithm is best in accuracy, but worst in efficiency. Various algorithms for labeling new coming data are the trade-off between these two extreme algorithms. In the proposed MKDPD algorithm, there are two learning processes. In the first learning process, the basic functions of label function are trained. In the second learning process, the weights of basic functions in the label function are solved. Accordingly, a new algorithm for labeling the new coming data is proposed. In the proposed algorithm, the new coming data will be exploited in the first learning process to retrain the basic functions, but the weights of basic functions remained unchanged and combined with the retrained basic functions to label new coming data. The proposed labeling algorithm achieves a better trade-off between the computational efficiency and accuracy.

The rest of the paper is arranged as follows: in Section 2, the literatures related to our work are reviewed briefly. In Section 3, the main theories of kernel functions and RKHS are introduced. In Section 4, the MKDPD algorithm is proposed. In Section 5, an MKDPD-based algorithm for labeling new coming data is proposed. In Section 6, the experimental results on toy and real-world data are presented to show the performance of the MKDPD algorithms. In Section 7, some conclusions are presented for reference.

2. Related Works

Learning from the given data is the main process to machine learning. So how to fully make use of the given samples is the key to the successful learning. In general, supervised learning has sufficient labeled samples and these kinds of algorithms are suitable for classification problems, such as the representative linear discriminant analysis (LDA) [19] and KFD [7]. In practice, a large number of samples are unlabeled, only a small number of samples are labeled. In this case, supervised learning algorithms do not effectively make use of the information hidden in unlabeled samples. To tackle the issue, semisupervised learning [18] is proposed and a wide range of semisupervised learning algorithms have been proposed and widely applied in many areas of machine learning.

In recent years, the study of semisupervised learning is not limited to the simple introduction of unlabeled data. Many researchers pay attention to exploit the intrinsic geometry of data and introduce the kernel learning to semisupervised methods. For example, manifold regularization learning proposed by Belkin et al. [8] exploits the underlying data structure by adding a manifold regular term to a general-purpose learner. And following it, a serious of algorithms are proposed. Sindhwani et al. [20] proposed a linear MR (LMR) algorithm, in which a global linear mapping between the samples and their labels is constructed for labeling novel samples. Inspired by Gaussian fields and harmonic functions (GFHF) [21], local and global consistency (LGC) [22], and LMR [20], Nie et al. [23] extended LMR algorithm to flexible manifold embedding algorithm (FMA). FMA relaxes the hard linear constraint in LMR by adding a flexible regression residue. Geng et al. [10] proposed the ensemble manifold regularization (EMR) to deal with the aforementioned problems by learning an optimal graph Laplacian based on a set of given candidate graph Laplacian. With the spare assumption, Fan et al. [24] replaced the manifold regularizer with a sparse regularizer under the MR framework. Luo et al. [11] applied MR framework to solve the problem of multilabel image classification by learning a discriminative subspace.

Introducing kernel trick to semisupervised learning methods is an important progress in machine learning. The kernel function [17] is either used to map input sample into a high dimensional kernel space for learning problem nonlinearization, or used to span a RKHS for the learning of label function. Take the MR learning as example, the label function (classifier function) in MR is a linear combination of single kernel function on labeled and unlabeled samples and the performance of MR algorithms strongly depends on this label function.

The theory of RKHS plays an important role in kernel methods and RKHS has found a wide range of applications such as minimum variance unbiased estimation of regression coefficients, least squares estimation of random variables, detection of signals in Gaussian noise, problems in optimal approximation [25]. Some RKHS-based learning algorithms appearing recently find applications to online learning with the problem of classification or regression [26–28], while others find applications to the classification of hyperspectral images [29, 30]. Gurram and Kwon [29] achieved the weights for SVM separating hyperplanes by combining both local spectral and spatial information. Gu et al. [30] introduced the conception of Multiple-Kernel Hilbert Space (MKHS) to analyze spectral unmixing problems, and the resultant algorithm performs well in solving nonlinear problems.

In theory, RKHS can be generated with some specific functions called kernel functions such as Gaussian, Laplacian, and polynomial [31]. Modifying kernel functions is a way of improving the performance of kernel methods; for example, Wu and Amari [32] extended conformal transformation of kernel functions to improve the performance of Support Vector Machine classifiers, Gurram and Kwon [33] defined a new inner product to warp the RKHS structure to reflect the intrinsic geometry of the given samples, and the literatures [33, 34] obtained the best kernel parameters by calculating the derivatives of objective functions. The application of multiple kernels is hot topic in kernel methods and multikernel learning (MKL) [35] is a successful method, which enhances the interpretability of the classifier with a combination of base kernels and improves the performances of kernel methods. MKL algorithms have been widely investigated [13–16, 35–37] and the reviews of MKL algorithms can be found in [13, 14]. MKL offers a feasible scheme to ensemble multiple kernels, but high computational cost raised by optimization procedure is a bad limitation when it is used to process large-scale data and a number of kernels. Therefore, one main research direction in MKL is how to effectively solve the MKL problem. Lots of MKL algorithms have been proposed; for example, SimpleMKL [36] proposed by Rakotomamonjy et al. is one of the state-of-the-art algorithms used to solve MKL problem which is addressed by a simple subgradient descent method. However, the MKL task is still challenging because it must on one hand learn an optimal combination of multiple kernels and determine the optimal classifier in each iteration and on the other hand make sure that the two optimization procedures are feasible.

3. RKHS and Its Application to Machine Learning

3.1. RKHS. In machine learning RKHS are often used as the solution spaces of learning problems. The definition of RKHS is as follows.

Definition 1. Let $H = (S(\Omega), \langle\cdot,\cdot\rangle)$ be a Hilbert space; if there is a function $k : \Omega \times \Omega \to R$, such that

(a) $\forall v \in \Omega, k(x, v) = k_v(x) \in S(\Omega)$;

(b) $\forall f \in S(\Omega), f(x) = \langle f, k_x\rangle = \langle f(\cdot), k(\cdot, x)\rangle$,

then, H is said to be a Reproducing Kernel Hilbert Space (RKHS) and the function $k(x, z)$ is called the reproducing kernel of H.

Note that in $H = (S(\Omega), \langle\cdot,\cdot\rangle)$, Ω is a data space, $S(\Omega)$ is a linear space of functions defined on the data space Ω, and $\langle\cdot,\cdot\rangle$ is an inner product defined on $S(\Omega)$. According to the theory of RKHS, RKHS can be generated from a kernel function. The kernel function is defined as follows.

Definition 2. Let $k : \Omega \times \Omega \to R$ be a function such that

(a) symmetric: $\forall x, v \in \Omega, k(x, v) = k(v, x)$;

(b) positive definite: for all the positive integer n and all data $v_1, \ldots, v_n \in \Omega$, the following matrix $K \in R^{n\times n}$ is positive definite:

$$\begin{bmatrix} k(v_1, v_1) & \cdots & k(v_1, v_n) \\ \vdots & \ddots & \vdots \\ k(v_n, v_1) & \cdots & k(v_n, v_n) \end{bmatrix}; \tag{4}$$

then the function $k(x, v)$ is called a kernel function.

A kernel function $k(x, v)$ can be used to generate a RKHS such that the kernel function is the reproducing kernel of RKHS. The generating procedure is as follows.

First, a linear space can be generated from the kernel function $k(x, v)$,

$$\begin{aligned} S_k(\Omega) &= \operatorname{span}\{k_v(x) \mid k_v(x) = k(x, v),\ v \in \Omega\} \\ &= \left\{ f(x) \mid f(x) = \sum_{i=1}^{n} \alpha_i k(x, v_i),\ \alpha_i \in R,\ v_i \in \Omega,\ i \right. \\ &\quad \left. = 1, \ldots, n,\ n \in N \right\}, \end{aligned} \tag{5}$$

where N is the set of all positive integers.

Second, an inner product $\langle \cdot, \cdot \rangle_k$ is defined on $S_k(\Omega)$; that is, for all $f, g \in S_k(\Omega)$, since $f(x) = \sum_{i=1}^{n} \alpha_i k(x, v_i)$ and $g(x) = \sum_{j=1}^{n} \beta_j k(x, u_j)$, the inner product of f and g is then defined as

$$\langle f, g \rangle_k = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \beta_j k(v_i, u_j). \tag{6}$$

It is easy to prove that the definition of $\langle \cdot, \cdot \rangle_k$ meets the requirements of the inner product and, therefore, $H_k = (S_k(\Omega), \langle \cdot, \cdot \rangle_k)$ is an inner product space.

It is worth noting that for all $f \in S_k(\Omega)$, since $f(x) = \sum_{i=1}^{n} \alpha_i k(x, v_i)$

$$\begin{aligned} \langle f(\cdot), k(\cdot, x) \rangle_k &= \left\langle \sum_{i=1}^{n} \alpha_i k(\cdot, v_i), k(\cdot, x) \right\rangle_k \\ &= \sum_{i=1}^{n} \alpha_i \langle k(\cdot, v_i), k(\cdot, x) \rangle_k \\ &= \sum_{i=1}^{n} \alpha_i k(x, v_i) = f(x). \end{aligned} \tag{7}$$

That is to say, the functions in the inner space H_k can be reproduced with the kernel function $k(x, v)$.

Third, the inner space H_k can be completed if it is not completed. The completion of H_k, denoted by $\overline{H}_k$, is then a RKHS and the kernel function $k(x, v)$ is the reproducing kernel of $\overline{H}_k$. By the way, it can be seen from the completion that the inner space H_k is dense in the RKHS $\overline{H}_k$.

3.2. Solution Spaces of Machine Learning Problems. In practice, it is impossible to take the space H_k as the solution space of learning problem because it is infinite-dimensional. It is reasonable to require that the solution space be both finite-dimensional and sample-dependent. Thus, for the given samples $\{x_1, \ldots, x_b, x_{b+1}, \ldots, x_{b+u}\}$, a linear space can be generated as follows:

$$\begin{aligned} S_X(\Omega) &= \operatorname{span}\{k(x, x_i) \mid i = 1, \ldots, b+u\} \\ &= \left\{ \sum_{i=1}^{b+u} \alpha_i k(x, x_i) \mid \alpha_i \in R,\ i = 1, \ldots, b+u \right\}. \end{aligned} \tag{8}$$

It is clear that $S_X(\Omega)$ is both finite-dimensional and sample-dependent. Furthermore, $S_X(\Omega)$ is exactly a subspace of $S_k(\Omega)$ and therefore $H_X = (S_X(\Omega), \langle \cdot, \cdot \rangle_k)$ is a subspace of H_k. Since H_X is finite-dimensional, then it is complete; that is, H_X is a Hilbert space. However, H_X is no longer a RKHS.

For all functions $f, g \in S_X(\Omega)$, since $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x, x_i)$ and $g(x) = \sum_{i=1}^{b+u} \beta_i k(x, x_i)$, then, according to (6), we have

$$\langle f, g \rangle_k = \sum_{i=1}^{b+u} \sum_{j=1}^{b+u} \alpha_i \beta_j k(x_i, x_j) = \vec{\alpha}^T K_X \vec{\beta}, \tag{9}$$

where $\vec{\alpha} = [\alpha_1 \cdots \alpha_{b+u}]^T$, $\vec{\beta} = [\beta_1 \cdots \beta_{b+u}]^T$, and

$$K_X = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_{b+u}) \\ \vdots & \ddots & \vdots \\ k(x_{b+u}, x_1) & \cdots & k(x_{b+u}, x_{b+u}) \end{bmatrix}. \tag{10}$$

Since the matrix K_X is symmetric and positive definite, the inner product on $S_X(\Omega)$ can be defined by itself, not necessarily inherited from $\langle \cdot, \cdot \rangle_k$. In fact, for all $f, g \in S_X(\Omega)$, the inner product can be defined as

$$\langle f, g \rangle_X = \vec{\alpha}^T K_X \vec{\beta}. \tag{11}$$

It can be easily proven that the definition of $\langle \cdot, \cdot \rangle_X$ meets the requirements of inner product and therefore $H_X = (S_X(\Omega), \langle \cdot, \cdot \rangle_X)$ is an inner space. Again, since $S_X(\Omega)$ is finite-dimensional, H_X is complete. In machine learning, it is the space H_X that is taken as the solution space of learning problem.

4. A Multikernel-Like Learning Algorithm Based on Data Probability Distribution MKDPD

4.1. Motivation. As shown in (5), the space of label functions is as follows:

$$S_X(\Omega) = \operatorname{span}\{k(x, x_i) \mid i = 1, \ldots, b+u\}. \tag{12}$$

This means that the functions $\{k(x, x_i) \mid i = 1, \ldots, b+u\}$ play the role of basic functions of $S_X(\Omega)$. Obviously, these basic functions are only dependent on the locations of given

samples and seem too simple to adapt to various probability distributions of data. Take Gaussian kernel function $k(u,v) = \exp^{-\|u-v\|^2/2\sigma^2}$ as an example, the basic functions $\{k(x,x_i) \mid i = 1,\ldots,b+u\}$ generated from Gaussian kernel function are identical with each other, only different in the locations of data space. A basic function can be derived from another basic function only by translation in the data space Ω. In fact, if $i \neq j$, then

$$\begin{aligned} k_i\left(x - x_j + x_i\right) &= \exp^{-\|x-x_j+x_i-x_i\|^2/2\sigma^2} = k\left(x, x_j\right) \\ &= k_j\left(x\right). \end{aligned} \tag{13}$$

Furthermore, since $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x,x_i)$, then for all $x \in \Omega$ with $p(x) \neq 0$, $f(x)$ should give the label of x. This means that $\sup p(p) \subseteq \cup_{i=1}^{b+u} \sup p(k(\cdot,x_i))$, where $\sup p(p)$ is the support of $p(x)$ and $\sup p(k(\cdot,x_i))$ is the support of $k(x,x_i)$; that is,

$$\begin{aligned} \sup p\left(p\right) &= \left\{x \mid x \in \Omega,\ p\left(x\right) \neq 0\right\}, \\ \sup p\left(k\left(\cdot,x_i\right)\right) &= \left\{x \mid x \in \Omega,\ k\left(x,x_i\right) \neq 0\right\}. \end{aligned} \tag{14}$$

If the relation $\sup p(p) \subseteq \cup_{i=1}^{b+u} \sup p(k(\cdot,x_i))$ is not true, there would be $x \in \Omega$ such that $p(x) \neq 0$, but $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x,x_i) = 0$; that is, $f(x)$ cannot give the label of x.

However, the union $\cup_{i=1}^{b+u} \sup p(k(\cdot,x_i))$ is dependent on the locations of the given data samples $\{x_1 \cdots x_{b+u}\}$, not dependent on the data probability distribution $p(x)$. In practice, kernel functions are often compactly supported and the data are not evenly distributed over the data space. In these cases, label function $f(x)$ will be overfitted in the areas where too many data samples are collected, or underfitted in the areas where there are too few data samples collected, or not fitted at all in the area where the union $\cup_{i=1}^{b+u} \sup p(k(\cdot,x_i))$ fails to cover.

Based on the above considerations, a learning algorithm based on the data probability distribution is proposed in this paper. In the proposed algorithm, the union is not only dependent on the locations of the given samples, but also dependent on the data probability distribution.

4.2. Construction of Solution Spaces. For the convenience of description, let $k(x,v \mid \theta)$ denote the kernel function, where θ represents the parameter of the kernel function. Thus, for the given data samples $\{x_1 \cdots x_{b+u}\}$ and data probability distribution $p(x)$, the basic functions of solution space generated from the kernel function $k(x,v \mid \theta)$ are expressed as $k(x,x_i \mid \theta_i)$, where $\theta_i = \theta_{p(i)}$, $i = 1 \cdots b+u$.

With these basic functions, we can span a linear space $S_D(\Omega)$ as follows:

$$\begin{aligned} S_D\left(\Omega\right) &= \operatorname{span}\left\{k\left(x,x_i \mid \theta_i\right) \mid i = 1,\ldots,b+u\right\} \\ &= \left\{\sum_{i=1}^{b+u} \alpha_i k\left(x,x_i \mid \theta_i\right) \mid i = 1,\ldots,b+u\right\}. \end{aligned} \tag{15}$$

It is clear that $S_D(\Omega)$ is a finite-dimensional linear space. Further, in order to define an inner product on $S_D(\Omega)$, we need to define a symmetric and positive definite matrix first:

$$M_D = K_D^T K_D + \rho I, \tag{16}$$

where ρ, $I \in R^{(b+u)\times(b+u)}$ is a unit matrix and

$$K_D = \begin{bmatrix} k\left(x_1,x_1 \mid \theta_1\right) & \cdots & k\left(x_1,x_{b+u} \mid \theta_{b+u}\right) \\ \vdots & \ddots & \vdots \\ k\left(x_{b+u},x_1 \mid \theta_1\right) & \cdots & k\left(x_{b+u},x_{b+u} \mid \theta_{b+u}\right) \end{bmatrix}. \tag{17}$$

Note that, since $k(x_i,x_j \mid \theta_j) \neq k(x_j,x_i \mid \theta_i)$, $i \neq j$, K_D is not symmetric and positive definite. However, M_D is symmetric and definite positive and can be used to define an inner product $\langle\cdot,\cdot\rangle_D$ on $S_D(\Omega)$: for all $f,g \in S_D(\Omega)$, since $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x,x_i \mid \theta_i)$ and $g(x) = \sum_{j=1}^{b+u} \beta_j k(x,x_j \mid \theta_j)$, then

$$\left\langle f,g\right\rangle_D = \vec{\alpha}^T M_D \vec{\beta}, \tag{18}$$

where $\vec{\alpha} = [\alpha_1 \cdots \alpha_{b+u}]^T$, $\vec{\beta} = [\beta_1 \cdots \beta_{b+u}]^T$.

It can be easily proven that $\langle\cdot,\cdot\rangle_D$ meets the requirements of inner product and therefore $H_D = (S_D(\Omega), \langle\cdot,\cdot\rangle_D)$ is an inner product space. Furthermore, since $S_D(\Omega)$ is finite-dimensional, H_D is then complete; that is, H_D is a Hilbert space. However, it is worth noting that H_D is neither a RKHS, nor a subspace of H_k. Recall that although H_X is not RKHS, H_X is a subspace of H_k.

In the proposed algorithm, H_D is taken as the solution space of learning problem:

$$f^* = \arg\min_{f\in H_D} \sum_{i=1}^{b} V\left(y_i, f\left(x_i\right)\right) + \kappa\left\|f\right\|_{H_D}^2 + \lambda\left\|f\right\|_M^2. \tag{19}$$

Below we explain the rationality of the definition $\langle\cdot,\cdot\rangle_D$:

(1) If $\langle\cdot,\cdot\rangle_D$ is an inner product of $S_D(\Omega)$, according to the linearity of inner product, for all $f,g \in S_D(\Omega)$, we have

$$\begin{aligned} \left\langle f,g\right\rangle &= \left\langle \sum_{i=1}^{b+u} \alpha_i k\left(\cdot,x_i \mid \theta_i\right), \sum_{j=1}^{b+u} \beta_j k\left(\cdot,x_j \mid \theta_j\right) \right\rangle \\ &= \sum_{i=1}^{b+u}\sum_{j=1}^{b+u} \alpha_i\beta_j \left\langle k\left(\cdot,x_i \mid \theta_i\right), k\left(\cdot,x_j \mid \theta_j\right)\right\rangle. \end{aligned} \tag{20}$$

Usually, the inner product of functional space is often defined as the integral of product of functions; therefore we have

$$\begin{aligned} &\left\langle k\left(\cdot,x_i \mid \theta_i\right), k\left(\cdot,x_j \mid \theta_j\right)\right\rangle \\ &\quad= \int_\Omega k\left(x,x_i \mid \theta_i\right) k\left(x,x_j \mid \theta_j\right) dx \\ &\quad\approx \sum_{h=1}^{b+u} k\left(x_h,x_i \mid \theta_i\right) k\left(x_h,x_j \mid \theta_j\right). \end{aligned} \tag{21}$$

Substituting (21) into (20) gives

$$\begin{aligned}\langle f, g\rangle &\approx \sum_{i=1}^{b+u}\sum_{j=1}^{b+u}\sum_{h=1}^{b+u}\alpha_i\beta_j k\left(x_h, x_i \mid \theta_i\right) k_j\left(x_h, x_j \mid \theta_j\right) \\ &= \vec{\alpha}^T K_D^T K_D \vec{\beta}.\end{aligned} \tag{22}$$

However the matrix $K_D^T K_D$ is only positive semidefinite and cannot be used to define an inner product. This problem can be easily solved by adding a regularization term ρI, where $I \in R^{(b+u)\times(b+u)}$ is the unit matrix and ρ is the regularization parameter: $M_D = K_D^T K_D + \rho I$.

The matrix M_D is now symmetric and positive definite and can be used to define an inner product on $S_D(\Omega)$:

$$\begin{aligned}\langle f, g\rangle_D &= \vec{\alpha}^T M_D \vec{\beta} = \vec{\alpha}^T K_D^T K_D \vec{\beta} + \rho \vec{\alpha}^T \vec{\beta} \\ &\approx \vec{\alpha}^T K_D^T K_D \vec{\beta} = \langle f, g\rangle.\end{aligned} \tag{23}$$

The regularization parameter ρ can also alleviate the ill-condition of the matrix $K_D^T K_D$ and reduce the error stemming from the substitution of integral with summation in (29).

(2) If the data probability distribution $p(x)$ is uniform, that is, $p(x)$ is constant on the support sup $p(p)$, then $\theta_i = \theta(p(x_i))$, $i = 1, \ldots, b+u$. In this case, we have

$$\left\langle k\left(\cdot, x_i \mid \theta_i\right), k\left(\cdot, x_j \mid \theta_j\right)\right\rangle = k\left(x_i, x_j \mid \theta\right), \tag{24}$$

$$\langle f, g\rangle = \langle f, g\rangle_X. \tag{25}$$

Combining (23) and (25) will give the following result:

$$\langle f, g\rangle_D \approx \langle f, g\rangle = \langle f, g\rangle_X. \tag{26}$$

This means that if the parameter θ of the kernel function $k(x, x_i \mid \theta)$ does not adjuste sample by sample, then $H_D = H_X$.

4.3. Analytic Solutions to Learning Problems

4.3.1. Analytic Solutions to Two-Class Learning Problems. In the proposed algorithm, the Hilbert space H_D is taken as the solution space of learning problems. Then for all $f \in H_D$, $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x, x_i \mid \theta_i)$, we have

$$\vec{f}_D = \left[f\left(x_1\right) \cdots f\left(x_{b+u}\right)\right]^T = K_D \vec{\alpha}, \tag{27}$$

$\|f\|_D^2 = \vec{\alpha}^T M \vec{\alpha}$, $\|f\|_M^2 = \vec{\alpha}^T K_D^T L_X K_D \vec{\alpha}$. Based on the above results, the problem shown in (19) can be simplified as follows:

$$\begin{aligned}\vec{\alpha}^* = \arg\min_{\vec{\alpha}\in R^{b+u}} \Bigg\{ &\sum_{i=1}^{b} V\left(y_i, \sum_{i=1}^{b+u}\alpha_i k\left(x_i, x_j \mid \theta_j\right)\right) \\ &+ \vec{\alpha}^T\left(\kappa M_D + \lambda K_D^T L_X K_D\right)\vec{\alpha}\Bigg\}.\end{aligned} \tag{28}$$

Furthermore, if the cost function $V(y, f(x))$ is set to be the square error function, that is, $V(y, f(x)) = (y - f(x))^2$, we have

$$\begin{aligned}\vec{\alpha}^* &= \arg\min_{\vec{\alpha}\in R^{b+u}} \vec{y}^T\vec{y} - 2\vec{y}^T S K_D \vec{\alpha} \\ &\quad + \vec{\alpha}^T\left(K_D^T S^T S K_D + \kappa M_D + \lambda K_D^T L_X K_D\right)\vec{\alpha} \\ &= \arg\min_{\vec{\alpha}\in R^{b+u}} \vec{y}^T\vec{y} - 2\vec{b}_D^T\vec{\alpha} + \vec{\alpha}^T A_D \vec{\alpha} = A_D^{-1}\vec{b}_D,\end{aligned} \tag{29}$$

where S is the selection matrix, $\vec{b}_D = K_D^T S^T \vec{y}$, and $A_D = K_D^T S^T S K_D + \kappa M_D + \lambda K_D^T L_X K_D$. Note that the matrix A_D is a symmetric and positive definite matrix.

4.3.2. Analytic Solutions to Multiclass Learning Problems. In principle, the deduction shown above is also suitable for the multiclass learning problems. In fact, for the data sample x_i, its label y_i can take different values to indicate the different classes to which the data sample x_i belongs. However, in practice, the different values of y_i may be too close to facilitate the optimization calculation. Therefore, for the multiclass problems, we adopt another way to indicate the data labels.

For the data sample x_i, let its label $\vec{y}_i$ be a C-dimensional vector, where C is the number of classes. If the data sample x_i belongs to the cth class, then the cth component of $\vec{y}_i$ is set to be 1 while the other components of $\vec{y}_i$ are set to zero, where $c = 1, \ldots, C$. Furthermore, a label function $f_c(x)$ is set to describe the probability that the data x belongs to the cth class. Based on these notations, the multiclass problem can be expressed as follows:

$$\begin{aligned}f^* = \arg\min_f &\sum_{i=1}^{b}\left\|\vec{y}_i - \vec{g}_i\right\|^2 + \kappa\sum_{c=1}^{C}\left\|\vec{f}_c\right\|_{H_D}^2 \\ &+ \lambda\sum_{c=1}^{C}\vec{f}_c{}^T L_X \vec{f}_c,\end{aligned} \tag{30}$$

where

$$\begin{aligned}f &= \left(f_1, \ldots, f_C\right), \\ \vec{g}_i &= \begin{bmatrix} f_1\left(x_i\right) \\ \vdots \\ f_C\left(x_i\right)\end{bmatrix}, \\ \vec{f}_c &= \begin{bmatrix} f_c\left(x_1\right) \\ \vdots \\ f_c\left(x_l\right)\end{bmatrix}, \\ &\qquad i = 1, \ldots, l.\end{aligned} \tag{31}$$

We first calculate the term $\sum_{c=1}^{C}\|\vec{f}_c\|_{H_D}^2$. Since $f_c(x) \in S_D(\Omega)$, then $f_c(x) = \sum_{i=1}^{b+u}\alpha_i^c k(x, x_i \mid \theta_i)$ and

$$\sum_{c=1}^{C}\left\|\vec{f}_c\right\|_{H_D}^2 = \sum_{c=1}^{C}\vec{\alpha}_c^T M_D \vec{\alpha}_c = \mathrm{Tr}\left(A^T M_D A\right), \tag{32}$$

where $\vec{\alpha}_c = [\alpha_1^c, \ldots, \alpha_{b+u}^c]^T$ and $A = [\vec{\alpha}_1, \ldots, \vec{\alpha}_C]$.

Secondly, we calculate the term $\sum_{c=1}^{C} \vec{f}_c^{\,T} L_X \vec{f}_c$. Since $\vec{f}_c = K_D \vec{\alpha}_c$ then

$$\sum_{c=1}^{C} \vec{f}_c^{\,T} L_X \vec{f}_c = \operatorname{trace}\left(A^T K_D^T L_X K_D A\right). \tag{33}$$

Thirdly, we calculate the term $\sum_{i=1}^{b} \|\vec{y}_i - \vec{g}_i\|^2$. Let $Y = [\vec{y}_1, \ldots, \vec{y}_b]$, $G = [\vec{g}_1, \ldots, \vec{g}_b]$, and $F = [\vec{f}_1, \ldots, \vec{f}_C] = K_D A$; then

$$\begin{aligned} \sum_{i=1}^{b} \left\|\vec{y}_i - \vec{g}_i\right\|^2 &= \|Y - G\|^2 \\ &= \operatorname{trace}\left(Y^T Y - 2YSK_D A + A^T K_D^T S^T S K_D A\right). \end{aligned} \tag{34}$$

Again, the matrix S is a selection matrix such that $G = F^T S^T$.

At last, substituting (32), (33), and (34) into (30) will give the following result:

$$\begin{aligned} A^* &= \arg\min_A \operatorname{trace}\left(YY^T - 2YSK_D A \right. \\ &\quad \left. + A^T\left(K_D^T S^T K_D + \kappa M_D + \lambda K_D^T L K_D\right) A\right) \\ &= \left(K_D^T\left(S^T S + \lambda L + \kappa I\right) K_D + \rho I\right)^{-1} K_D^T S^T Y^T. \end{aligned} \tag{35}$$

4.4. The Framework of Multikernel-Like Learning Algorithms. In the algorithm proposed in this paper, the label function $f(x)$ is set to be $f(x) = \sum_{i=1}^{b+u} \alpha_i k(x, x_i \mid \theta_i)$, where the parameter θ_i is adjusted according to the data probability distribution $p(x)$ on the data sample x_i. In general, the data probability distribution is not uniform and therefore the parameter θ_i will be different sample by sample. As a result, the functions $k(x, v \mid \theta_1), \ldots, k(x, v \mid \theta_{b+u})$ are different kernel functions and will produce different RKHS. In this sense, the algorithm proposed in this paper can be regarded as a kind of multikernel learning algorithms, but quite different from the commonly used multikernel learning algorithm.

In the commonly used multikernel learning algorithms, the multikernel function is a linear combination of multiple basic kernel functions: $k_{\mathrm{MK}}(x, v) = \sum_{j=1}^{M} \beta_j k_j(x, v)$, where the functions $k_1, \ldots, k_M$ are called basic kernel functions, while the function $k_{\mathrm{MK}}(x, v)$ is called the multikernel function. Since the basic kernel functions are symmetric and positive definite, it can be easily proven that the multikernel function is also symmetric and positive definite. Therefore the label function $f_{\mathrm{MK}}(x)$ based on the multikernel function can be expressed as

$$\begin{aligned} f_{\mathrm{MK}}(x) &= \sum_{i=1}^{b+u} \alpha_i k_{\mathrm{MK}}(x, x_i) = \sum_{i=1}^{b+u} \alpha_i \sum_{j=1}^{M} \beta_j k_j(x, x_i) \\ &= \sum_{i=1}^{b+u} \sum_{j=1}^{M} \alpha_i \beta_j k_j(x, x_i), \end{aligned} \tag{36}$$

where the coefficients $\alpha_1, \ldots, \alpha_{b+u}$ and $\beta_1, \ldots, \beta_M$ are determined through machine learning.

If we follow the ideas of the commonly used multikernel learning algorithms and regard the functions $k(x, v \mid \theta_1), \ldots, k(x, v \mid \theta_{b+u})$ as the basic kernel functions, then the multikernel function becomes $k_{\mathrm{MK}}(x, v) = \sum_{j=1}^{M} \beta_j k(x, v \mid \theta_j)$. Thus, according to (36), the label function $f_{\mathrm{MK}}(x)$ based on the multikernel function becomes

$$\begin{aligned} f_{\mathrm{MK}}(x) &= \sum_{i=1}^{b+u} \alpha_i k_{\mathrm{MK}}(x, x_i) = \sum_{i=1}^{b+u} \alpha_i \sum_{j=1}^{M} \beta_j k\left(x, x_i \mid \theta_j\right) \\ &\neq \sum_{i=1}^{b+u} \alpha_i k\left(x, x_i \mid \theta_i\right) = f(x). \end{aligned} \tag{37}$$

It can be seen from (37) that, no matter how to adjust the coefficients $\beta_1, \ldots, \beta_M$, it is impossible to make $f_{\mathrm{MK}}(x) \neq f(x)$. From the perspective of solution spaces, in the solution space of $f_{\mathrm{MK}}(x)$, there are $b + u$ functions $k(x, x_i \mid \theta_1), \ldots, k(x, x_i \mid \theta_{b+u})$ around the data sample x_i, while in the solution space of $f(x)$, there is only one function $k(x, x_i \mid \theta_i)$ around the data sample x_i. Therefore the algorithm proposed in this paper is quite different from the commonly used multikernel learning algorithm.

Nevertheless, the algorithm proposed in this paper still belongs to the realm of multikernel learning. As stated above, the functions $k(x, x_i \mid \theta_1), \ldots, k(x, x_i \mid \theta_{b+u})$ are different kernel functions and can produce different RKHS: $H_1, \ldots, H_{b+u}$. Now let $V_i = \{\alpha k(x, x_i \mid \theta_i) \mid \alpha \in R\}$, as stated in Section 3.1; V_i is then a 1-dimensional subspace of H_i. Furthermore, the direct sum of these subspaces turns out to be

$$V = V_1 \oplus \cdots \oplus V_{b+u} = \left\{ \sum_{i=1}^{b+u} \alpha_i\left(x, x_i \mid \theta_i\right) \mid \alpha_i \in R \right\}. \tag{38}$$

Obviously the direct sum of these subspaces is the solution space H_D.

Due to the fact that our algorithm is different from the commonly used multikernel learning algorithm, but still involved in multiple kernel functions, our algorithm is called multikernel-like algorithm.

5. An MKDPD-Based Algorithm for Labeling New Coming Data

5.1. Problems. How to label new coming data $\{x_1^{\mathrm{new}}, \ldots, x_n^{\mathrm{new}}\}$ has been a hot topic in machine learning, where $n \geq 1$ represents the number of new coming data. Generally speaking, there are two extreme methods for labeling new coming data: relearning methods and unlearning methods.

The relearning methods regard the new coming data as unlabeled data samples and mixe them with the original data samples to form new data samples: $\{x_1, \ldots, x_{b+u}, x_{b+u+1}, \ldots, x_{b+u+n}\}$, where $x_{b+u+j} = x_j^{\mathrm{new}}$, $j = 1, \ldots, n$. Based on these new data samples, the methods relearn the new coefficients $\{\alpha_1^{\mathrm{new}}, \ldots, \alpha_{b+u+n}^{\mathrm{new}}\}$ of the label function f. The label of the new coming data x_j^{new} is then given by $f_{\mathrm{MR}}^{\mathrm{re}}(x_j^{\mathrm{new}}) = \sum_{i=1}^{b+u+n} \alpha_i^{\mathrm{new}} k(x_j^{\mathrm{new}}, x_i)$.

The unlearning methods make use of the original coefficients $\{\alpha_1^{\text{old}}, \ldots, \alpha_{b+u}^{\text{old}}\}$ of label function f to label new coming data: $f_{\text{MR}}^{\text{un}}(x_j^{\text{new}}) = \sum_{i=1}^{b+u} \alpha_i^{\text{old}} k(x_j^{\text{new}}, x_i)$.

Obviously, in terms of accuracy, the relearning methods perform best, whiles the unlearning methods perform worst. In terms of efficiency, the unlearning methods perform best, while the relearning methods perform worst. For years the researchers have been hovering between these two extreme methods and try to find the trade-offs between accuracy and efficiency.

5.2. A MKDPD-Based Algorithm for Labeling New Coming Data. As stated in Section 4, there are two times of learning in the MKDPD algorithm. In the first time of learning, the MKDPD algorithm has to adjust the parameters of kernel functions according to data probability distribution. In the second time of learning, the MKDPD algorithm has to determine the coefficients of label functions. Therefore, in the framework of the MKDPD algorithm, there are at least three ways to label new coming data:

(1) The MKDPD-based relearning method: $f_{\text{MKDPD}}^{\text{re}}(x_j^{\text{new}}) = \sum_{i=1}^{b+u+n} \alpha_i^{\text{new}} k(x_j^{\text{new}}, x_i \mid \theta_i^{\text{new}})$, $j = 1, \ldots, n$. Obviously, the MKDPD-based relearning method can achieve the best accuracy but perform worst in efficiency because the MKDPD-based relearning method has to calculate both the parameters θ^{new} and coefficients α^{new}.

(2) The MKDPD-based unlearning method: $f_{\text{MKDPD}}^{\text{un}}(x_j^{\text{new}}) = \sum_{i=1}^{b+u} \alpha_i^{\text{old}} k(x_j^{\text{new}}, x_i \mid \theta_i^{\text{old}})$. Obviously, the MKDPD-based unlearning method can achieve the best efficiency but perform worst in accuracy because the unlearning method does not calculate the parameters θ^{new} and coefficients α^{new} either.

(3) The MKDPD-based semilearning method: $f_{\text{MKDPD}}^{\text{semi}}(x_j^{\text{new}}) = \sum_{i=1}^{b+u} \alpha_i^{\text{old}} k(x_j^{\text{new}}, x_i \mid \theta_i^{\text{new}})$. The MKDPD-based semilearning method regards the new coming data as unlabeled data samples and mixes them with the original data samples to retrain the new parameters θ^{new} of kernel functions. However, the coefficients α^{old} remained unchanged and combined with the retrained kernel functions to label the new coming data. The MKDPD-based semilearning method takes full advantage of two times of learning in the MKDPD algorithm and achieves a better trade-off between computational accuracy and efficiency.

5.3. Error and Efficiency Analysis

5.3.1. Experimental Data and Experimental Settings. We test our algorithm in the framework of manifold regularization and therefore the experimental data are downloaded from the website of manifold regularization (http://manifold.cs.uchicago.edu/manifold_regularization/manifold.html). There are a total of 400 sets of data collected from two half-moons, 200 sets of data from one half-moon, and another 200 sets of data from another half-moon.

We randomly take 1 set of datum as labeled sample and 99 sets of data as unlabeled samples from each half-moon. The remaining 200 sets of data are taken as new coming data for labeling.

In order to alleviate the effect of random sampling on the objectivity of the experimental results, the random sampling has been done for N times and each random sampling will produce an experimental result. The average of N experimental results is taken as the end result. N is set to be 30, 50, 70, and 90, respectively (Table 1).

Table 1: Average error rates (%) of four algorithms on testing samples with different number of tests.

Algorithm	Train times			
	$N = 30$	$N = 50$	$N = 70$	$N = 90$
$f_{\text{MR}}^{\text{un}}$	0.8500	0.8800	0.7286	1.0167
$f_{\text{MKDPD}}^{\text{un}}$	0.2333	0.2700	0.2786	0.2722
$f_{\text{MKDPD}}^{\text{semi}}$	0.1667	0.2200	0.19297	0.2000
$f_{\text{MKDPD}}^{\text{re}}$	0	0	0	0.0333

5.3.2. Error Analysis. Table 1 lists the error rates of various algorithms for labeling new coming data. Not surprisingly, the order of error rates is $f_{\text{MKDPD}}^{\text{re}} \leq f_{\text{MKDPD}}^{\text{semi}} \leq f_{\text{MKDPD}}^{\text{un}}$. This order coincides with the amount of information exploited by these algorithms from the new coming data.

In addition, the error rate of $f_{\text{MKDPD}}^{\text{un}}$ is smaller than that of $f_{\text{MR}}^{\text{un}}$, where $f_{\text{MKDPD}}^{\text{un}} = \sum_{i=1}^{b+u} \alpha_i^{\text{old}} k(x, x_i \mid \theta_i^{\text{old}})$ and $f_{\text{MR}}^{\text{un}} = \sum_{i=1}^{b+u} \alpha_i^{\text{old}} k(x, x_i)$. It can be seen from the formulae of $f_{\text{MKDPD}}^{\text{un}}$ and $f_{\text{MR}}^{\text{un}}$ that the basic functions $k(x, x_i \mid \theta_i^{\text{old}})$ of $f_{\text{MKDPD}}^{\text{un}}$ exploit not only the locations of samples, but also the probabilities of data on the samples, while the basic functions $k(x, x_i)$ of $f_{\text{MR}}^{\text{un}}$ exploit only the locations of samples.

Figure 1(a) shows the error rates of various algorithms change along with the number of new coming data. Again, the error rate of $f_{\text{MKDPD}}^{\text{semi}}$ is between those of $f_{\text{MKDPD}}^{\text{un}}$ and $f_{\text{MKDPD}}^{\text{re}}$.

5.3.3. Efficiency Analysis. Figure 1(b) shows the runtime of various algorithms in labeling the new coming data. It can be seen from Figure 1(b) that the runtime of $f_{\text{MKDPD}}^{\text{re}}$ increases exponentially along with the number of new coming data, while the runtime of $f_{\text{MKDPD}}^{\text{un}}$, $f_{\text{MKDPD}}^{\text{semi}}$, and $f_{\text{MR}}^{\text{un}}$ almost remains unchanged.

Since both $f_{\text{MKDPD}}^{\text{un}}$ and $f_{\text{MR}}^{\text{un}}$ make use of the original parameters θ^{old} and α^{old} to label the new coming data, the runtime of $f_{\text{MKDPD}}^{\text{un}}$ and $f_{\text{MR}}^{\text{un}}$ will not change no matter how many new coming data are coming. However, in the proposed algorithm $f_{\text{MKDPD}}^{\text{semi}}$, although the parameters θ are retrained from θ^{old} to θ^{new} according to the new coming data, the runtime of $f_{\text{MKDPD}}^{\text{semi}}$ almost remains unchanged along with the number of new coming data. The means that $f_{\text{MKDPD}}^{\text{semi}}$ achieves a certain amount of accuracy without increasing its runtime.

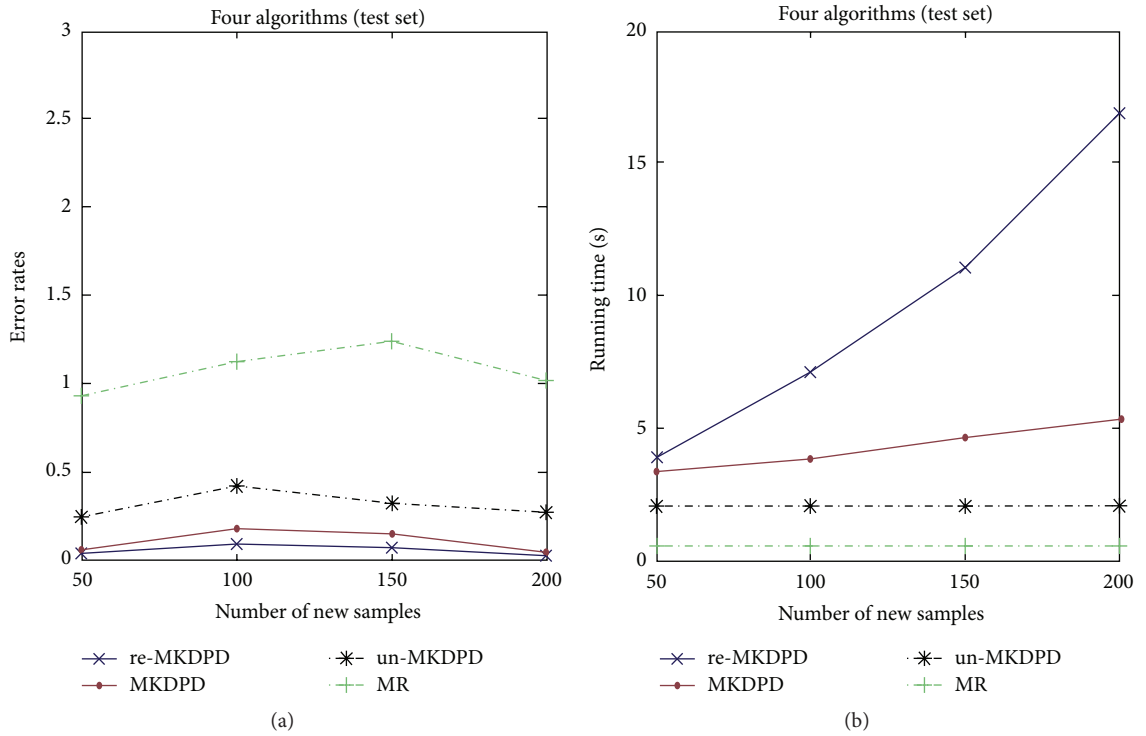

Figure 1: (a) shows the error rates of the four algorithms ($f^{\text{un}}_{\text{MR}}$, $f^{\text{un}}_{\text{MKDPD}}$, $f^{\text{semi}}_{\text{MKDPD}}$, and $f^{\text{re}}_{\text{MKDPD}}$) on the testing set, and the number of new coming samples is 50, 100, 150, and 200, respectively. (b) reports the computation time of the four algorithms with the increasing of the new coming sample (from 50 to 200).

6. Experiments

6.1. Adjustment of Parameters of Kernel Functions. In the proposed MKDPD algorithm, the basic functions of label function $f(x)$ are set to be $k(x, x_i \mid \theta_i)$, where $\theta_i = \theta(p(x_i))$, $i = 1, \ldots, b+u$. The schemes of how to adjust the parameters θ_i are open. People can adopt various schemes according to their specific applications. No matter how to adjust the parameters θ_i, the structures of analytic solutions shown in (19) will not change in the framework of the proposed MKDPD algorithm. In the experiments presented in this paper, the scheme of adjusting the parameters is based on the following considerations;

(1) The parameters θ_i should be adjusted so as to make $\sup p(p) \subseteq \cup_{i=1}^{b+u} \sup p(k(\cdot, x_i \mid \theta_i))$. In this way, for all $x \in \Omega$ with $p(x) \neq 0$, the label function $f(x)$ can give the label of x.

(2) If the value of $p(x_i)$ is large, the number of samples gathering in the neighborhood of x_i will be large too because they are more likely to be collected. In order to prevent data overfitting in the area, it is reasonable to adjust the parameter θ_i to reduce the scope of the support $\sup p(k(\cdot, x_i \mid \theta_i))$. Conversely, if the value of $p(x_i)$ is small, the number of samples will be small too because they are more unlikely to be collected. In order to prevent data underfitting in the area, it is reasonable to adjust the parameter to expand the scope of the support $\sup p(k(\cdot, x_i \mid \theta_i))$. This means that the probability $p(x_i)$ is inversely proportional to the scope of the support.

(3) In the following experiments, we adopt Gaussian kernel functions $k(x, x_i \mid \theta_i) = \exp^{-\|x-x_i\|^2/2\theta_i^2}$. The Gaussian kernel function can be regarded as a compactly support function, the sample x_i is its center, and $3\theta_i$ is often regarded as its effective radius. Therefore, the parameter θ_i is proportional to the scope of the support $\sup p(k(\cdot, x_i \mid \theta_i))$, or inversely proportional to the probability $p(x_i)$; that is, $\theta_i = \eta_i / p(x_i)$, where η_i is an adjustable parameter. In our experiments, the probability $p(x_i)$ of sample x_i is set to $p(x_i) = p_i(x_i) / \sum_i p_i(x_i)$, where $p_i(x_i) = \varepsilon(x_i)/(b+u)$ and $\varepsilon(x_i)$ is the number of neighbors of x_i.

6.2. Experiments on Synthetic Dataset (Two Moons Dataset). The synthetic dataset is the Two Moons Dataset, which has already been used in the experiments in Section 5.

The Two Moons dataset contains 400 sets of data non-evenly collected from two half-moons, where 200 sets of data are collected from one half-moon and the other 200 sets of data are collected from the other half-moon. We randomly take 100 sets of data from each half-moon as samples, where 1 sample is labeled and the other samples are unlabelled. The remaining 200 sets of data in the Two Moons dataset are taken as the test data (i.e., new coming data in Section 5).

Figure 2 shows the experimental results of the proposed MKDPD algorithm and MR algorithm. In Figures 2(a) and

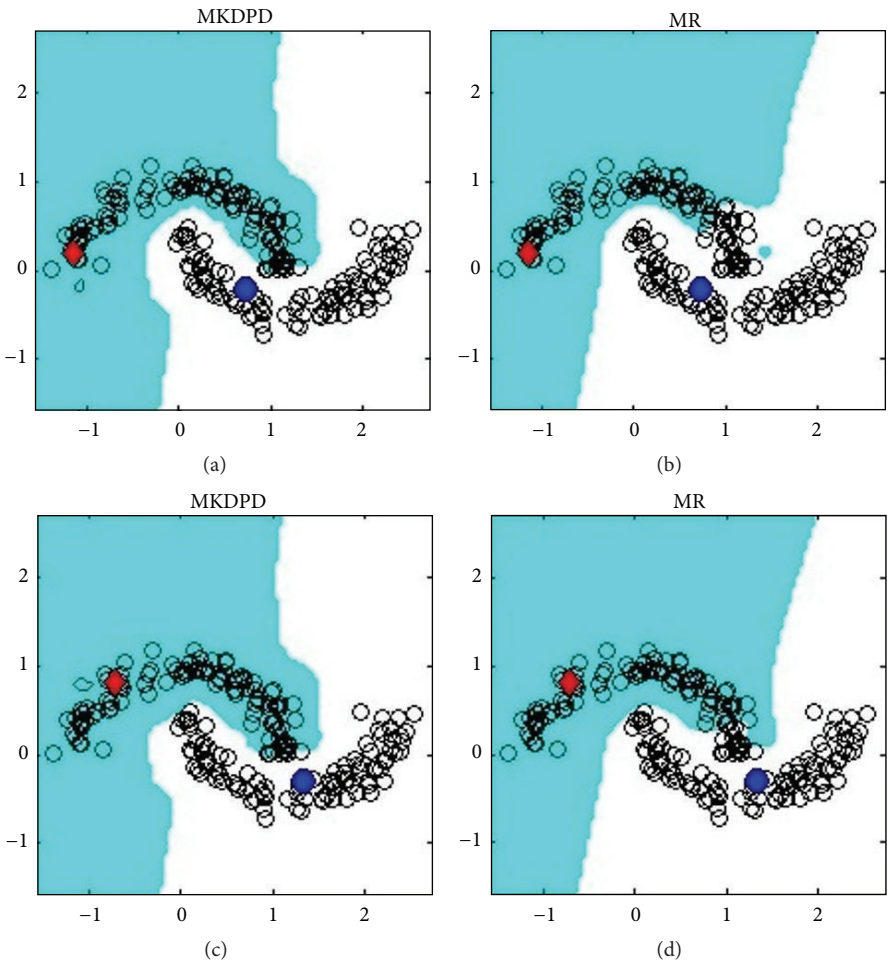

Figure 2: Two Moons Dataset: separating planes of MKDPD (see (a) and (c)) and MR (see (b) and (d)); the red and blue color points are labeled samples; the rest of the points are unlabeled samples.

2(b), although the labeled sample in the upper half-moon is badly located, the proposed MKDPD algorithm can still separate the upper half-moon from the lower half-moon, while MR algorithm fails at the one corner of the upper half-moon. In Figures 2(c) and 2(d), the labeled sample in the upper half-moon is located much near the center of the upper half-moon and accordingly the performance of MR algorithm becomes much better. However, the proposed MKDPD algorithm still outperforms MR algorithm in this circumstance.

The labeled samples are randomly taken for N times and each time will produce an experimental result. The average of experimental results is listed in Table 2. It can be seen from Table 2 that the proposed MKDPD algorithm outperforms MR algorithm under all circumstances.

There are two regularization terms in the proposed MKDPD algorithm and MR algorithm: manifold regularization $\lambda\|f\|_M^2$ and function regularization $\kappa\|f\|_H^2$, where $H = H_D$ in MKDPD or $H = H_X$ in MR. We keep λ unchanged, while letting κ change from 0 to 0.2 and comparing their performances (see Figure 3). As shown in Figure 3, the error rates of the proposed MKDPD algorithm and MR algorithm decrease where the parameter κ decreases, but the error rate of the proposed MKDPD decreases much faster than that of MR algorithm.

Table 2: Average error rates (%) of MKDPD and MR on unlabeled samples of two moons set with different number of tests.

Algorithm	Train times			
	$N = 30$	$N = 50$	$N = 70$	$N = 90$
MKDPD	0	0	0	0.0112
MR	0.8586	0.9293	0.7720	1.0325

6.3. Recognition of Handwritten Digits

6.3.1. USPS and MNIST Datasets. USPS (United States Postal) Dataset (http://www.cs.nyu.edu/~roweis/data.html) is a very popular dataset of handwritten digits, which contains 10 handwritten digits from "0" to "9"; each digit has 1100 images and each image is sized as 16×16 and can be converted into a 256-dimensional vector. We take the first 400 images of

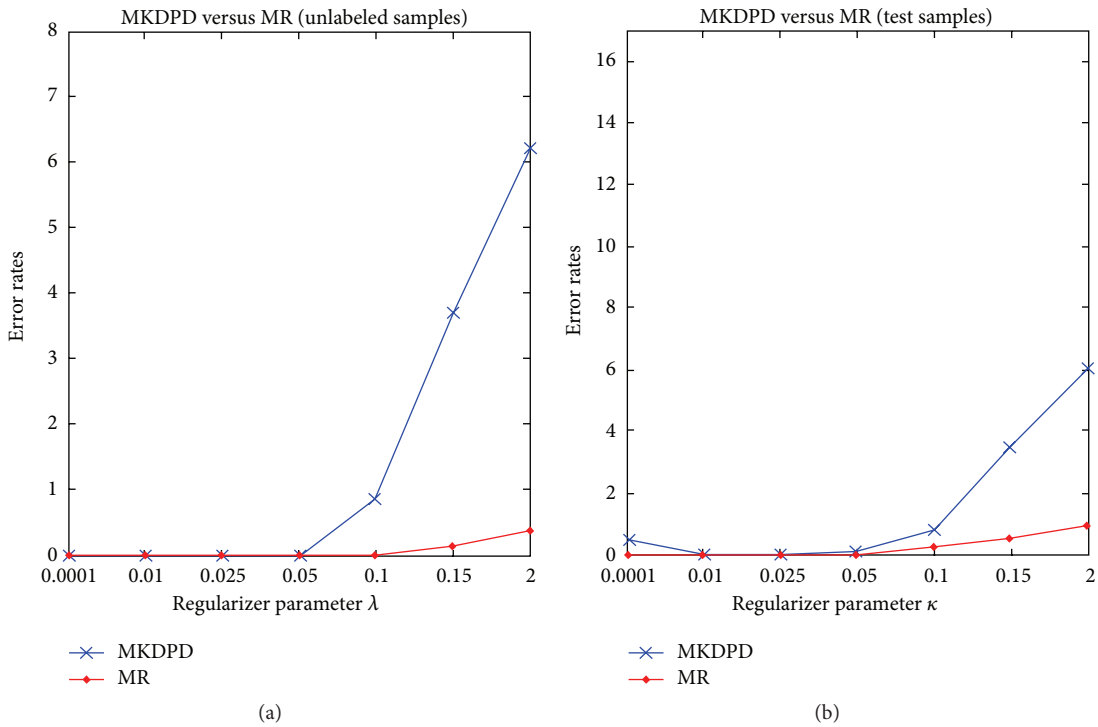

Figure 3: Error rates (%) of MKDPD and MR on unlabeled and testing samples of two moons set with the regular parameter κ varied from 0 to 0.2.

each digit as the samples and the remaining images as the test data (new coming data).

The MNIST (http://yann.lecun.com/exdb/mnist/) is another popular handwritten digits dataset, which contains a training set of 60000 images and a test set of 10000 images. Each image in MNIST is of size 28×28 and can be converted into a 784-dimensional vector. We select 400 sets of data from the training set as the samples for each digit and all data in the test set as the test data (new coming data).

6.3.2. The Two-Class Experiments. We randomly select two different digits from the ten digits to construct a binary classification problem, and then there is a total of 45 classification problems. For each binary classification problem, a sample is taken as the labeled sample for each digit and the remaining samples are taken as the unlabeled samples. To avoid the randomness, the labeled samples are randomly selected for ten times and each time will produce an experimental result. The average of ten experimental results is presented as the final experiment result.

The experimental results of the proposed MKDPD algorithm and MR algorithm are presented in Figure 4. In Figures 4(a), 4(b), 4(c), and 4(d), the x-axis represents the 45 binary classification problems, and the y-axis represents the error rates. The error rates on the unlabeled samples are shown in Figures 4(a) and 4(c), and the error rates on the test data are shown in Figures 4(b) and 4(d). As can be seen, the error rates of the proposed MKDPD algorithm are lower than those of the MR algorithm. Furthermore, the averages of the results of 45 binary classifications are listed in Table 3. It can be seen from Table 3 that the proposed MKDPD algorithm outperforms the MR algorithm.

Table 3: Average error rates (%) of the two-class experiments on USPS, MNIST, and ISOLET datasets.

Dataset	Algorithm	Unlabeled samples	Testing samples
USPS	MKDPD	**2.5494**	**2.4723**
	MR	3.1541	3.5774
MNIST	MKDPD	**5.4213**	**6.0654**
	MR	6.8557	8.1182
ISOLET	MKDPD	**15.4067**	**17.7037**
	MR	17.6039	24.9947

In Figures 4(e), 4(f), 4(g), and 4(h), the x-axis represents the error rates of labeling the unlabeled samples and the y-axis represents the error rates of labeling the test data. For a good learning algorithm, its error rates on the unlabeled samples and on the test data should be close to each other; that is, the scatter points in Figures 4(e), 4(f), 4(g), and 4(h) should be close to the diagonal line. Again, in this respect, the proposed MKDPD algorithm performs the MR algorithm better.

6.3.3. The Multiclass Experiments. In the multiclass experiments, there are 10 classes and each class corresponds to a digit. The labeled samples of each digit are randomly selected from its samples for 10 times and each time will produce an experimental result. The average of 10 experimental results is taken as the final result and listed in the first and second

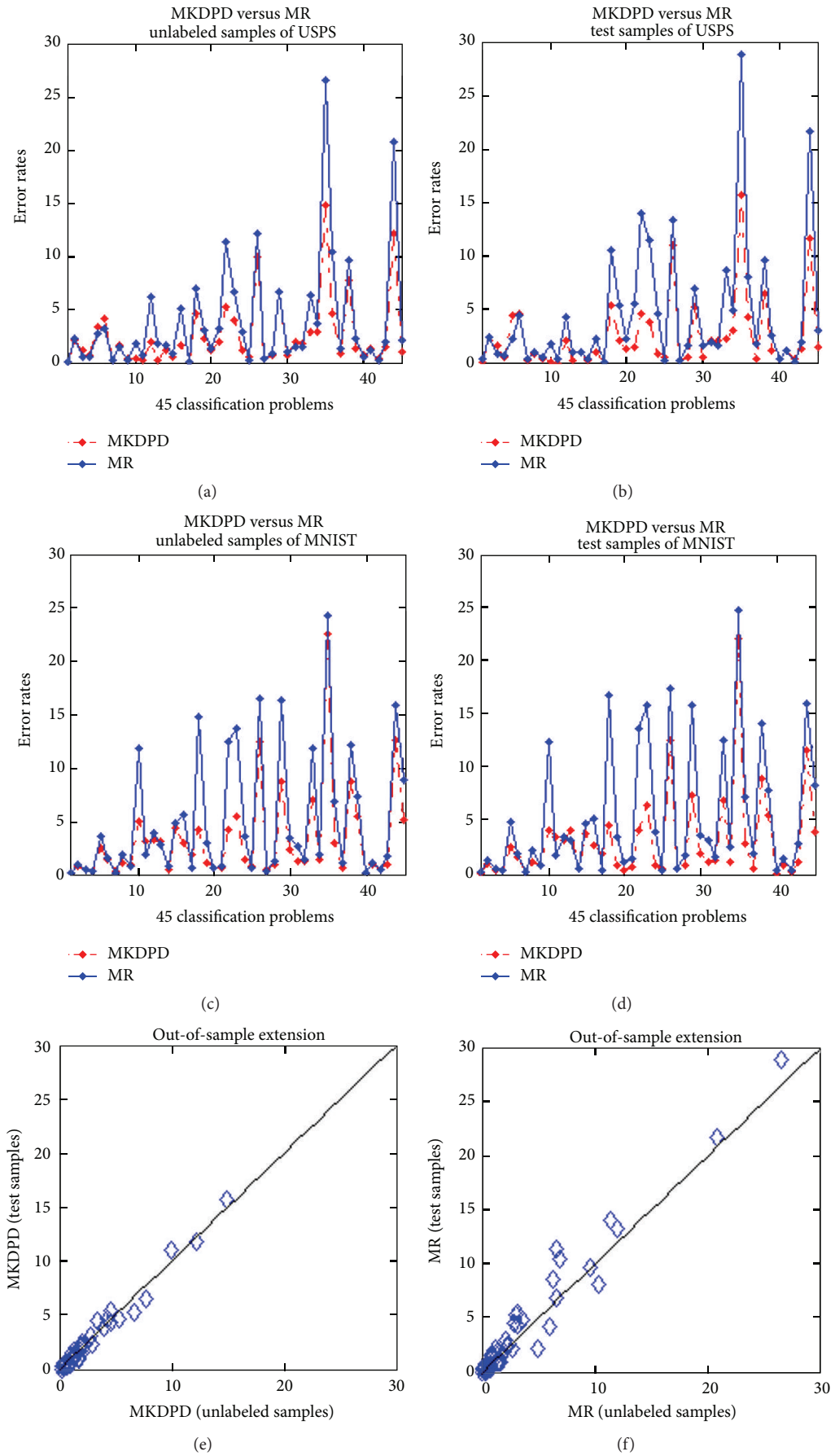

Figure 4: Continued.

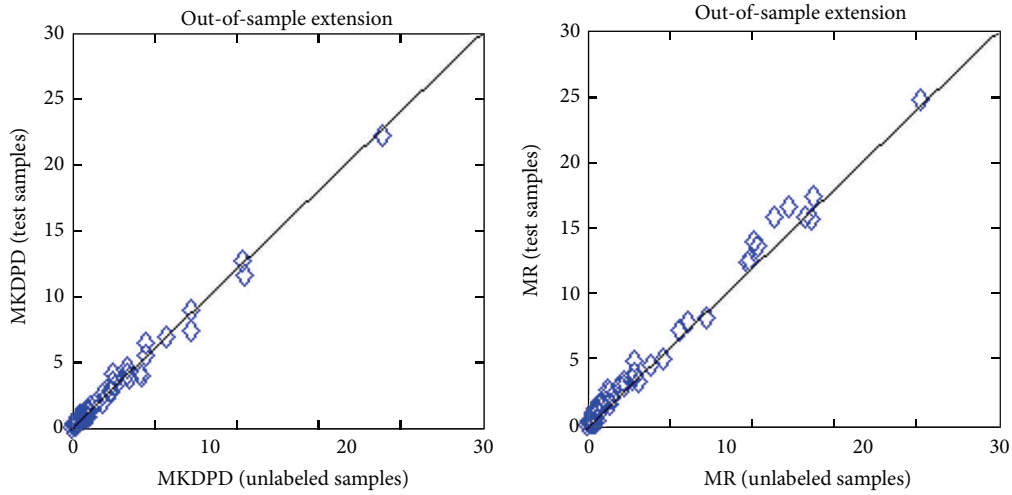

Figure 4: Two-class problem experiments: in (a, b, c, and d), (a) and (b) report the error rates (%) of MKDPD and MR on unlabeled and test samples of the USPS set, respectively, and (a) and (b) report the error rates (%) of MKDPD and MR on unlabeled and test samples of the MNIST set, respectively, where the y-axis is the 45 binary classification problems; in (e, f, g, and h), we report the scatter plots of error rate on unlabeled samples versus test samples. (e) and (g) are the scatter plots of error rates of MKDPD on USPS and MNIST, respectively, and (f) and (h) are the scatter plots of error rates of MR on USPS and MNIST, respectively.

column of Table 5. The number of labeled samples is set to be 1, 3, and 5, respectively. It can be seen from Table 5 that the proposed MKDPD algorithm outperforms the MR algorithm.

6.4. Recognition of Spoken Letters

6.4.1. ISOLET Dataset. ISOLET is a dataset of spoken letters and can be downloaded from UCI machine learning repository. ISOLET contains the utterances of 150 speakers who spoke 26 English letters twice, and then each speaker has 52 utterances. In the experiment, two subsets of ISOLET, whose names are ISOLET1 and ISOLET5 respectively, are directly download from (http://manifold.cs.uchicago.edu/manifold_regularization/manifold.html). Each subset contains the utterances of 30 speakers. We take the data in ISOLET1 as the samples and the data in ISOLET5 as the test data (new coming data).

6.4.2. The Two-Class Experiments. In the two-class experiments, the utterances of the first 13 English letters are classified as one class and the utterances of the last 13 English letters are classified as another class. We take the 52 utterances of one speaker from ISOLET1 as the labels samples and the utterances of the other speakers in ISOLET1 as the unlabeled samples. Since there are 30 speakers in ISOLET1, we can construct 30 two-class experiments this way and the 30 experimental results are presented in Figure 5. As can be seen from Figure 5, the proposed MKDPD algorithm outperforms the MR algorithm. The averages of 30 experimental results experiments are listed in Table 4, which also shows that the proposed MKDPD performs better than the MR algorithm.

6.4.3. The Multiclass Experiments. In the multiclass experiments, the utterances of each English letter are classified as a class, and then there are 26 classes. We randomly select N speakers from ISOLET1 and take their utterances as the labeled samples. The utterances of the other speakers in ISOLET1 are then taken as the unlabeled samples. In order to alleviate the effect of randomness, the selection of N speakers is performed for 10 times and each time will produce an experimental result. The averages of 10 experimental results are taken as the final results and listed in the third column of Table 5. It can be observed from Table 5 that the proposed MKDPD algorithm achieves about 1%~3% improvements over the MR algorithm.

6.5. Face Recognition

6.5.1. YALE-B and CMU-PIE Datasets. YALE-B (http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html) is a dataset of face images which consists of the face images of 10 persons; each person is photographed under 9 poses and 64 illuminations. In the experiment, we select the face images of 8 persons of 64 illuminations as the experimental dataset. Each image is cropped and resized to an image of 32×32 pixels. For each person, 50% of his face images are taken as the samples, while his other face images are taken as the test data (new coming data). CMU-PIE [38] is another dataset of face images which consists of more than 40000 images of 68 persons; each person is photographed under 4 expressions, 24 Illuminations, and 13 poses. In the experiment, we select the face images of 8 persons of 3 poses and 24 illuminations as the experimental dataset. Each image is cropped and resized to an image of 32×32 pixels. Again, for each person, 50% of

Table 4: Average error rates (%) of the two-class experiments on Yale-B and CMU-PIE datasets with different number of labeled samples.

Dataset	Algorithm	3 labeled samples		5 labeled samples		7 labeled samples	
		Unlabeled	Test	Unlabeled	Test	Unlabeled	Test
Yale-B	MKDPD	**35.2279**	**31.4955**	**30.4664**	**25.1786**	**27.3871**	**20.7087**
	MR	36.7658	32.7455	31.7899	26.8136	28.8743	22.7087
CMU-PIE	MKDPD	**23.9889**	**20.4167**	**13.6962**	**10.1042**	**8.5904**	**5.5754**
	MR	26.0776	22.9712	15.9315	12.3611	10.0515	6.8452

Table 5: Error rates (%) of the multiclass problem experiments on five datasets: USPS, MNIST, ISOLET, Yale-B, and CMU-PIE, where the value in bracket is the number of labeled sample in each class.

Dataset	Algorithm	Unlabeled (labeled samples)			Test (labeled samples)		
USPS	MKDPD	**19.8241 (1)**	**12.7056 (3)**	**10.3949 (5)**	**18.1100 (1)**	**10.9359 (3)**	**8.5658 (5)**
	MR	23.5377 (1)	14.5279 (3)	11.2359 (5)	21.2428 (1)	13.3242 (3)	10.2856 (5)
MNIST	MKDPD	**31.8342 (1)**	**20.5939 (3)**	**16.0872 (5)**	**38.5650 (1)**	**27.2950 (3)**	**23.0300 (5)**
	MR	35.1206 (1)	21.2741 (3)	17.9026 (5)	44.6050 (1)	31.3800 (3)	26.7300 (5)
ISOLET	MKDPD	**18.5743 (2)**	**12.7060 (4)**	**10.3704 (6)**	**29.9469 (2)**	**24.4297 (4)**	**21.7241 (6)**
	MR	20.6300 (2)	15.1305 (4)	13.0057 (6)	30.1790 (2)	25.4111 (4)	22.4801 (6)
Yale-B	MKDPD	**36.4440 (3)**	**26.6991 (5)**	**18.4750 (7)**	**35.3516 (3)**	**26.8203 (5)**	**17.2070 (7)**
	MR	45.5172 (3)	36.9676 (5)	31.3500 (7)	41.9531 (3)	36.1406 (5)	25.5078 (7)
CMU-PIE	MKDPD	**23.9773 (3)**	**11.5591 (5)**	**5.6034 (7)**	**20.1968 (3)**	**9.3056 (5)**	**4.1551 (7)**
	MR	25.2904 (3)	12.2715 (5)	6.3793 (7)	21.0532 (3)	11.0069 (5)	5.7292 (7)

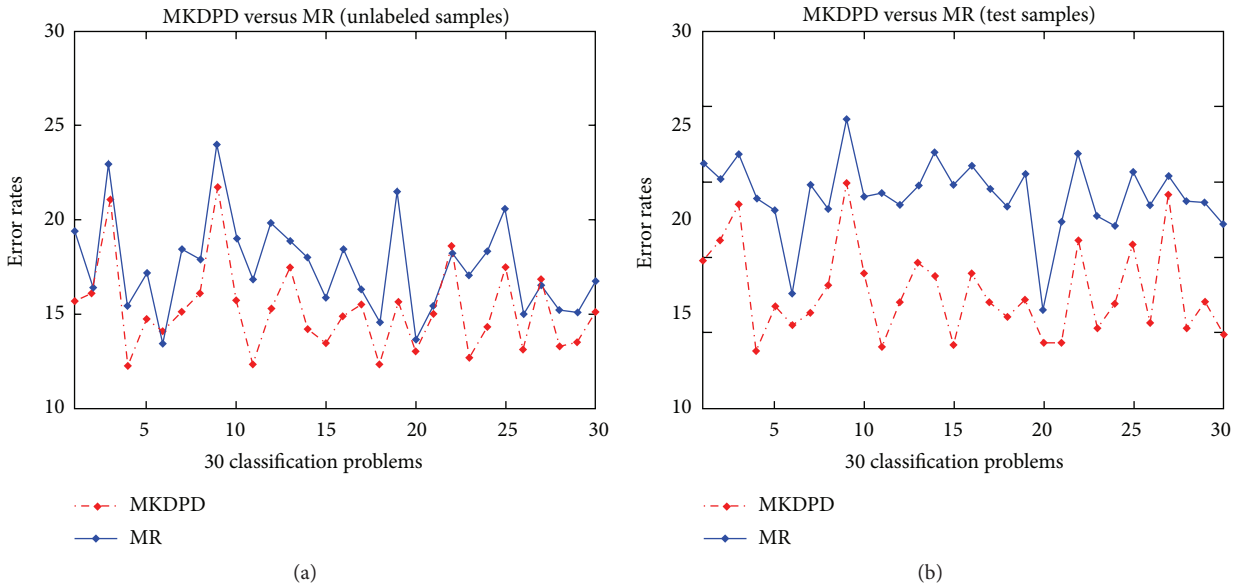

Figure 5: Two-class problem experiments: (a) is the error rates (%) of MKDPD and MR on unlabeled samples of ISOLET set, where the y-axis is the 30 binary classification problems; (b) is the error rates of MKDPD and MR on test samples of ISOLET set.

his face images are taken as the samples, while his other face images are taken as the test data (new coming data).

6.5.2. The Two-Class Experiments. We select the face images of two persons to construct a binary classification problem and then there are a total of 28 binary classification problems. For each binary classification problem, N samples of each person are taken as the labeled samples and the remaining samples are taken as the unlabeled samples. To avoid the randomness, the N samples are randomly taken for 10 times and each time will produce an experimental result. The average of 10 experimental results is presented as the final results (see Figures 6 and 7), where the number of labeled samples is set to be $N = 3, 5$, and 7, respectively. The average of 28 binary classification results is listed in Table 4. It can be seen that, compared with the MR algorithm, the proposed MKDPD algorithm achieves about 1%~5% improvements.

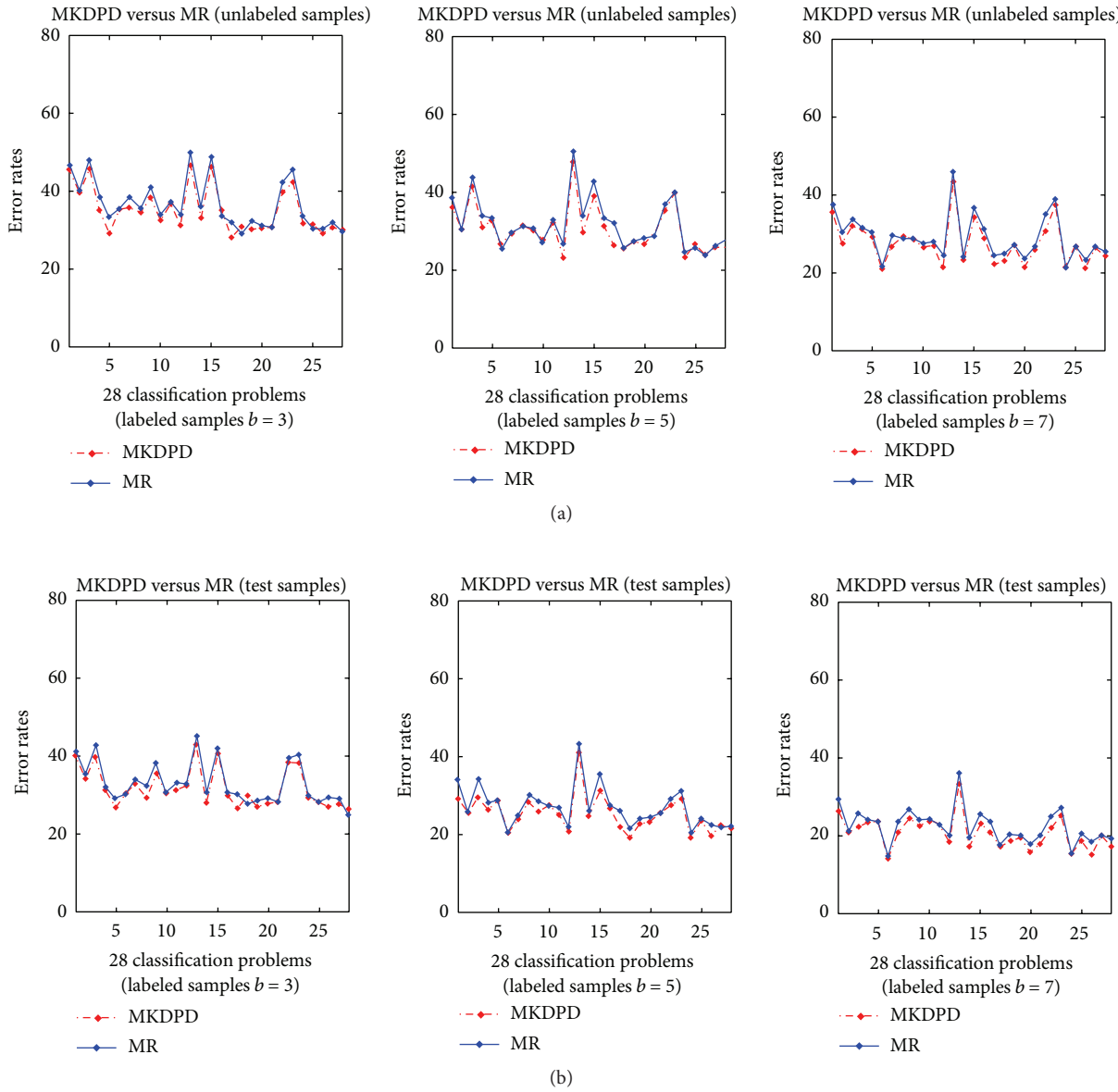

Figure 6: Two-class problem experiments: (a) is the error rates (%) of MKDPD and MR on unlabeled samples of the Yale-B set; (b) is the error rates (%) of MKDPD and MR on test samples of the Yale-B set. And the y-axis of each figure is the 28 binary classification problems and the value in bracket is the number of labeled samples in each class.

6.5.3. The Multiclass Experiments. In the multiclassification experiments, we take the face images of a person as one class and then there are 8 classes. N samples of each person are taken as the labeled samples and the remaining samples as the unlabeled samples. Again, the N labeled samples are randomly taken for 10 times and each time will produce an experimental result. The average of 10 experimental results is presented as the final result and listed in the last four columns of Table 5, where the number of labeled samples is set to be $N = 3, 5$, and 7, respectively. As can be seen, the proposed MKDPD algorithm achieves about 1%~12% improvements to the MR algorithm.

7. Conclusion

In machine learning, one variable of a kernel function is often anchored on each given sample and thus derives a number of basic functions of the label function. The weights of basic functions in the label function are then trained by exploiting the labels of labeled samples. The basic functions derived this way are the same in shape, only different in the positions of data space. Obviously, these basic functions seem too simple to adapt to the changes of data distribution. For example, if the given samples are distributed unevenly, then in the area where there are too many samples are given, there

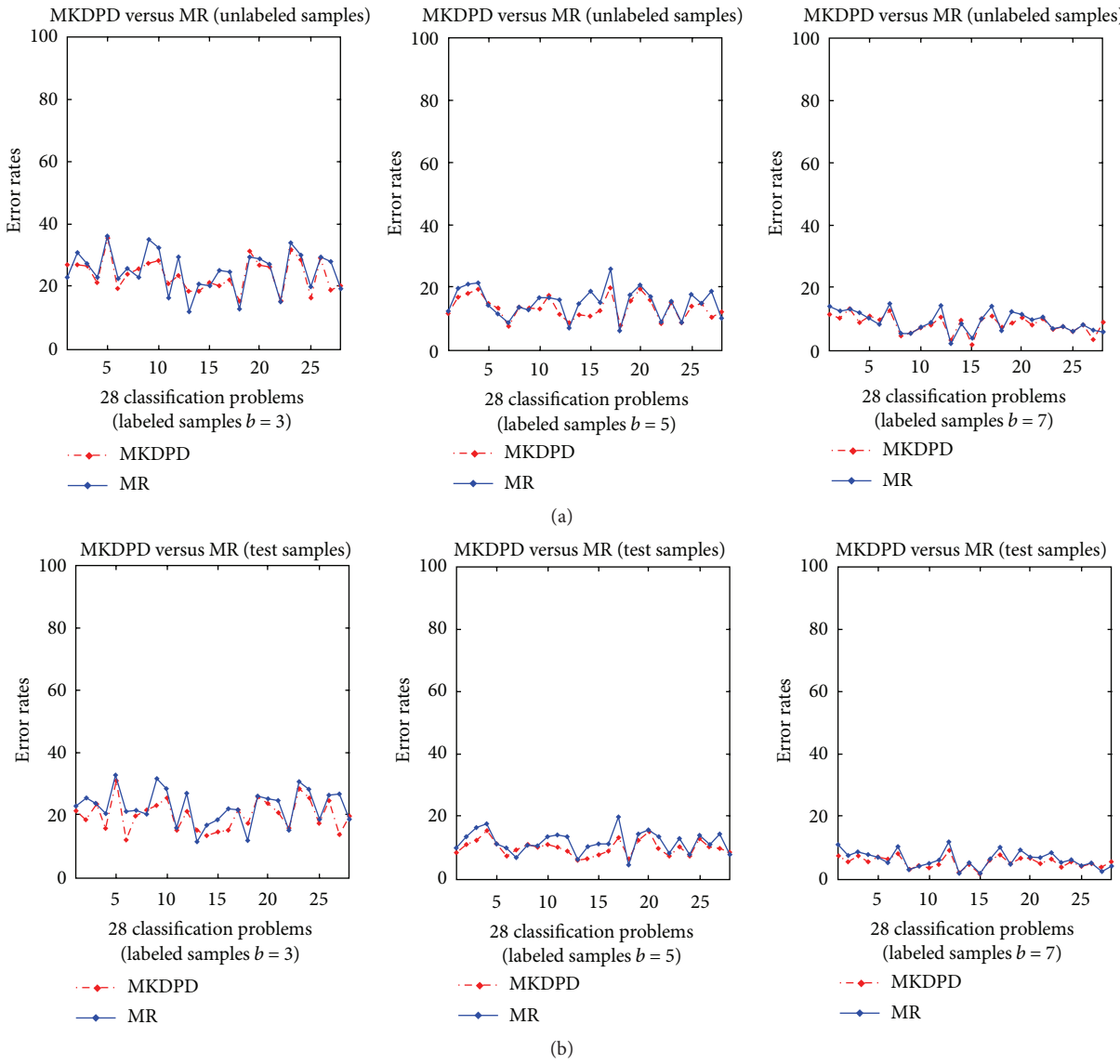

Figure 7: Two-class problem experiments: (a) is the error rates (%) of MKDPD and MR on unlabeled samples of the CMU-PIE set; (b) is the error rates (%) of MKDPD and MR on test samples of the CMU-PIE set. And the y-axis of each figure is the 28 binary classification problems and the value in bracket is the number of labeled sample in each class.

will be too many basic functions located in this area and maybe overlapped too much, while in the area where there are too few samples given, there will be too few basic functions located in this region and maybe overlapped too little or not overlapped at all.

In the MKDPD algorithm proposed in this paper, we adjust the basic functions according to the probabilities of data on the given samples. If the probability of data on a sample is large, then the number of samples in the vicinity of the sample will be large too and we can reduce the support of the basic function located on the sample accordingly to avoid overlapping too much with other basic functions. Likewise, if the probability of data on a sample is small, the number of samples in the vicinity of the sample will be small too and we can expand the support of the basic function located on the sample accordingly to avoid overlapping too little with other basic functions. The experimental results justify the proposed MKDPD algorithm.

From the perspective of the applications of data classification, the aim of machine learning is to label the new coming data. Usually, there are three methods: unlearning, relearning, and semilearning. In the MKDPD algorithm proposed in this paper, there are two learning processes: learning the basic functions and learning the weights of basic functions. In this paper we propose a semilearning method based on the MKDPD algorithm. The proposed semilearning method regards the new coming data as unlabeled samples and mixes them with the original samples to relearn the basic functions, but still the original weights to combine the new basic functions to label the new coming data.

The proposed MKDPD-based method for labeling new coming data achieves a better trade-off between the computational accuracy and efficiency.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported in part by the Guangdong Provincial Science and Technology Major Projects of China under Grant 67000-42020009.

References

[1] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[2] S. Kay, "Model-based probability density function estimation," *IEEE Signal Processing Letters*, vol. 5, no. 12, pp. 318–320, 1998.

[3] A. G. Bors and N. Nasios, "Kernel bandwidth estimation in methods based on probability density function modelling," in *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*, Tampa, Fla, USA, December 2008.

[4] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[5] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, New York, NY, USA, 2004.

[6] B. Scholkopf, A. Smola, and K. R. Mller, "Nonlinear component analysis as a kernel eigenvalue problem," *Journal of Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[7] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, "Fisher discriminant analysis with kernels," in *Proceedings of the IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX*, pp. 41–48, Madison, Wis, USA, August 1999.

[8] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[9] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, 2011.

[10] B. Geng, D. Tao, C. Xu, L. J. Yang, and X.-S. Hua, "Ensemble manifold regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1227–1233, 2012.

[11] Y. Luo, D. C. Tao, B. Geng, C. Xu, and S. J. Maybank, "Manifold regularized multitask learning for semi-supervised multilabel image classification," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 523–536, 2013.

[12] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[13] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1354–1369, 2014.

[14] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[15] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, "Multikernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 265–277, 2014.

[16] Y.-L. Xu, D.-R. Chen, H.-X. Li, and L. Liu, "Least square regularized regression in sum space," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 635–646, 2013.

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

[18] X. J. Zhu, "Semi-supervised learning literature survey," Computer Sciences TR 1530, 2008.

[19] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[20] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: from transductive to semi-supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, pp. 825–832, August 2005.

[21] X. J. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, Washington, DC, USA, 2003.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16*, 2004.

[23] F. P. Nie, D. Xu, I. W. Tsang, and C. S. Zhang, "Flexible manifold embedding: a framework for semi-supervised and unsupervised dimension reduction," *IEEE Transactions on Image Processing*, vol. 19, no. 7, pp. 1921–1932, 2010.

[24] M. Y. Fan, N. N. Gu, H. Qiao, and B. Zhang, "Sparse regularization for semi-supervised classification," *Pattern Recognition*, vol. 44, no. 8, pp. 1777–1784, 2011.

[25] J.-W. Xu, A. R. C. Paiva, I. Park, and J. C. Principe, "A reproducing kernel Hilbert space framework for information-theoretic learning," *IEEE Transactions on Signal Processing*, vol. 56, no. 12, pp. 5891–5902, 2008.

[26] S. Liwicki, S. Zafeiriou, G. Tzimiropoulos, and M. Pantic, "Efficient online subspace learning with an indefinite kernel for visual tracking and recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 10, pp. 1624–1636, 2012.

[27] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Adaptive multiregression in reproducing kernel hilbert spaces: the multiaccess MIMO channel case," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 260–276, 2012.

[28] G. Q. Li, C. Y. Wen, Z. G. Li, A. M. Zhang, F. Yang, and K. Z. Mao, "Model-based online learning with kernels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 3, pp. 356–369, 2013.

[29] P. Gurram and H. Kwon, "Contextual SVM using Hilbert space embedding for hyperspectral classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 5, pp. 1031–1035, 2013.

[30] Y. Gu, S. Wang, and X. Jia, "Spectral unmixing in multiple-kernel hilbert space for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 7, pp. 3968–3981, 2013.

[31] B. Scholkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, Mass, USA, 2001.

[32] S. Wu and S.-I. Amari, "Conformal transformation of kernel functions: a data-dependent way to improve support vector machine classifiers," *Neural Processing Letters*, vol. 15, no. 1, pp. 59–67, 2002.

[33] P. Gurram and H. Kwon, "Sparse kernel-based ensemble learning with fully optimized kernel parameters for hyperspectral classification problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 787–802, 2013.

[34] J. Huang, P. C. Yuen, W.-S. Chen, and J. H. Lai, "Choosing parameters of kernel subspace LDA for recognition of face images under pose and illumination variations," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 37, no. 4, pp. 847–862, 2007.

[35] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[36] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[37] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multiple kernel sparse representations for supervised and unsupervised learning," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2905–2915, 2014.

[38] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003.