Neural Comput & Applic (2011) 20:1087–1096
DOI 10.1007/s00521-011-0534-5

ORIGINAL ARTICLE

Accelerating the kernel-method-based feature extraction procedure from the viewpoint of numerical approximation

Yong Xu · David Zhang

Received: 8 December 2009 / Accepted: 27 January 2011 / Published online: 22 February 2011

Abstract The kernel method suffers from the following problem: the computational efficiency of the feature extraction procedure is inversely proportional to the size of the training sample set. In this paper, from a novel viewpoint, we propose a very simple and mathematically tractable method to produce the computationally efficient kernel-method-based feature extraction procedure. We first address the issue that how to make the feature extraction result of the reformulated kernel method well approximate that of the naïve kernel method. We identify these training samples that statistically contribute much to the feature extraction results and exploit them to reformulate the kernel method to produce the computationally efficient kernel-method-based feature extraction procedure. Indeed, the proposed method has the following basic idea: when one training sample has little effect on the feature extraction result and statistically has the high correlation with regard to all the training samples, the feature extraction term associated with this training sample can be removed from the feature extraction procedure. The proposed method has the following advantages: First, it proposes, for the first time, to improve the kernel method through formal and reasonable evaluation on the feature extraction term. Second, the proposed method improves the kernel method at a low extra cost and thus has a much more computationally efficient training phase than most of the previous improvements to the kernel method. The experimental comparison shows that the proposed method performs well in classification problems. This paper also intuitively shows the geometrical relation between the identified training samples and other training samples.

Keywords Pattern recognition · Kernel methods · Feature extraction · Kernel minimum squared error · Kernel PCA

Y. Xu (✉)
Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, 518055 Shenzhen, Guangdong, People's Republic of China
e-mail: laterfall2@yahoo.com.cn

D. Zhang
Biometrics Research Centre, Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong

1 Introduction

In the field of pattern recognition, the kernel method, which can deal with nonlinearly separable problems well, has attracted much attention. The kernel method can implicitly map the original nonlinearly separable samples into a new higher dimensional space i.e. feature space, where the samples from different classes may become linearly separable and a linear classifier can accurately classify them [1, 2]. The reason why the kernel method can do so is that the mapping function is nonlinear, which is able to convert nonlinearly separable problems into linearly separable problems. For the same reason, the kernel method is referred to as a nonlinear feature extraction method. Up to now, many kernel methods, e.g. kernel regression, kernel principal component analysis, and kernel Fisher discriminant analysis have been developed [3–15].

It is clear that the kernel-method-based feature extraction is superior to conventional nonlinear feature extraction. As we know, a conventional nonlinear feature extraction method should first transform samples into a new space by an explicit nonlinear mapping and then perform feature extraction in the new space. However, because of the use of the kernel function, kernel methods indeed do not explicitly

perform a nonlinear mapping. As a result, the kernel methods are usually more computationally efficient than the conventional nonlinear methods.

The kernel method also suffers from the following problem: when it extracts features from a sample, it should calculate all the kernel functions regarding this sample and all the training samples. Consequently, the computational efficiency of the feature extraction procedure is inversely proportional to the size of the training sample set. On the other hand, a large-scale training set is helpful for a method to achieve a good performance. In real-world applications, people also exploit some scheme to enlarge the training set with the purpose of the proposed method being able to fit more data. It is clear that if the size of the training set is very large, the computation of the kernel method will become very inefficient and even impractical [11]. Indeed, this might hinder the kernel method from being applicable to real-world applications. In this sense, it is significant to improve the kernel method for obtaining a fast feature extraction procedure.

A variety of improvements to kernel methods have been proposed for the purpose of speeding the feature extraction efficiency. These improvements are proposed from different viewpoints. For example, Xu et al. [9, 11, 14, 16] constructed computationally efficient kernel-method-based feature extraction procedures based on the nature of different kernel methods. They proposed the improvements to kernel discriminant analysis (KLDA), kernel minimum squared error (KMSE), and kernel PCA (KPCA) with the following idea: the improvements to the kernel methods should produce the possible optimal values of the corresponding objective functions of naïve kernel methods. For instance, when reformulating KLDA, they required that the reformulated KLDA model should be the candidate model that has the largest Fisher criterion value. M. E. Tipping also exploited the nature of the principal component analysis methodology to develop sparse KPCA [17]. Researchers also proposed to construct efficient feature extraction procedures for kernel methods from the viewpoint of vector approximation. For example, Scholkopf et al. [10] proposed that the discriminant vector of the improved kernel method should have the minimum numerical deviation with respect to the genuine discriminant vector corresponding to the naïve kernel method. Other methods to improve the kernel method include the leave-one-out cross-validation method [18] and prototype reduction schemes [19]. So far, improvements to the several typical kernel methods, e.g. improvements to support vector machine [20], to KPCA [17], to KFD [11, 21], and to KMSE [14, 16] are all available. However, most of these improvements have the following common characteristic: their training phases have a high computational cost, being much more computationally inefficient than the training phases of the naïve kernel methods. This characteristic will reduce the applicability in real-world applications of these improvements. In this paper, we also show the locations in the original space and the location in the feature space of the identified training samples that contribute much to the feature extraction results, which help us intuitively see the geometrical relation between the identified training samples and other training samples.

In this paper, we propose a new numerical approximation viewpoint and reformulate the kernel method for obtaining a computationally efficient feature extraction procedure from this viewpoint. The proposed method is simple, intuitive and computationally efficient in both the training phase and the feature extraction phase. Our basic ideas are as follows: first, suppose that a linear combination of some of the training samples in the feature space can be used to replace the linear combination of all the training samples to represent the genuine discriminant vector. Second, if the feature extraction results obtained using the constructed approximation discriminant vector are similar to the feature extraction results obtained using the genuine discriminant vector, we say that the constructed approximation discriminant vector is good. This is because classification is directly based on the feature extraction result, and similar feature extraction result can almost produce the same classification decision. Using the above-mentioned ideas, we develop the algorithm that evaluates the contribution, to the feature extraction result, of each element of the training sample set. Actually, as shown later, the contribution, to the feature extraction result of samples, of the element of the training set is reflected by the product of the corresponding kernel function and the component of the discriminant vector. The algorithm identifies the training samples that contribute much to the feature extraction results and exploit a linear combination of them to express the discriminant vector in the feature space. Since the corresponding feature extraction procedure needs to calculate only as many kernel functions as there are the identified training samples, which are much fewer than the total training samples, the feature extraction procedure will extract features more efficiently than the original feature extraction procedure based on the naïve kernel method. It also should be pointed out that the reformulated kernel method has a lower structure complexity, so it might generalize better than the naïve kernel method. In addition, we also illustrate, for the first time, the locations in the original space and in the feature space of the identified training samples. This is very useful for us to understand the geometrical relation between the identified training samples and other training samples.

The remainder of this paper is organized as follows: In Sect. 2, we present the idea to improve the kernel method, the scheme to evaluate the contribution, to the feature extraction results of samples, of the element of the training

sample set and the rationale of the proposed scheme. In Sect. 3, we present how to reformulate KMSE and KPCA as well as the main steps of the corresponding feature extraction and classification procedure. In Sect. 4, we show the experimental results on the benchmark datasets. Finally, we offer our brief conclusion in Sect. 5.

2 The idea, scheme, and analysis of the reformulated kernel method

2.1 The idea to improve the kernel method

In this subsection, we formally present our idea to improve the kernel method. For simplicity, in this paper, we assume that there are only two classes. We focus on only the kernel method based on the reproducing kernel theory. Typical examples of this kind of kernel methods include KPCA, KLDA, and KMSE et al. Each of these kernel methods correspond to a discriminant vector w in the feature space. According to the reproducing kernel theory, the discriminant vector can be expressed as a linear combination of all the (training) samples in the feature space as follows:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i). \tag{1}$$

For most of kernel methods, the feature extraction result of sample $\phi(x)$ in the feature space is indeed the projection onto w of $\phi(x)$. As a result, the feature extraction result of $\phi(x)$ can be formally expressed as

$$y(x) = w^T \phi(x) = \left(\sum_{i=1}^{n} \alpha_i \phi(x_i) \right)^T \phi(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x), \tag{2}$$

where $k(x_i, x) = \varphi(x_i)^T \varphi(x)$ is the defined kernel function. $k(x_i, x)$ is also referred to as the kernel function with regard to x and x_i. The use of the kernel function allows us not to explicitly perform the nonlinear mapping ϕ for any sample. It is clear that the feature extraction result of a sample is one linear combination of all the kernel functions with regard to this sample and all the training samples.

We also note that in the feature space of KMSE, the discriminant vector w includes the bias w_0, i.e. $w = w_0 + \sum_{i=1}^{n} \alpha_i \phi(x_i)$. As a result, the feature extraction result associated with KMSE is formally little different from (2) and can be described as follows

$$y(x) = w_0 + \sum_{i=1}^{n} \alpha_i k(x_i, x), \tag{3}$$

From (2) and (3), we know that for sample x, if the product of the kernel function $k(x_i, x)$ and the corresponding coefficient α_i has a large absolute value, then the i-th training sample x_i actually contributes much to the feature extraction result of x. Once this product always has a large absolute value for a sample set $S = \{x\}$ rather than a sole sample, then we can say that from the viewpoint of statistics, the i-th training sample x_i always contributes much to the feature extraction result of an arbitrary sample. On the contrary, if the product of the kernel function $k(x_j, x)$, $x \in S$ and the corresponding coefficient α_j always has a small absolute value, then the j-th training sample x_j actually contributes little to the feature extraction result. As a result, if we remove the term $\alpha_j k(x_j, x)$, which corresponds to the j-th training sample, from (2) and (3), the feature extraction result of x will change only little. Assume that we can identify all of these kinds of training samples, then we can remove all the terms corresponding to these training samples. This will bring only a little change to the feature extraction result. As a result, we can exploit $y(x) = \sum_{i=1}^{r} \beta_i k(x'_i, x)$ or $y(x) = w'_0 + \sum_{i=1}^{r} \beta_i k(x'_i, x)$ to extract feature from sample x. We will show how to determine β_i and w'_0 in Sect. 3. Since $r < n$, we can extract features more computationally efficiently.

2.2 The scheme to evaluate the feature extraction term

We devise the scheme to evaluate the feature extraction term as follows. Hereafter, let S be identical to the training set, i.e. $S = \{x_p, p = 1, 2, \ldots, n\}$. For the p-th element x_p of S, the main part (or the whole part) of its feature extraction result is $\sum_{i=1}^{n} \alpha_i k(x_i, x_p)$. As mentioned in Sect. 2.1, the value of $|\alpha_i| k(x_i, x_p)$ indeed represents the contribution of the i-th training sample to the feature extraction result of x_p. We also refer to α_i as discriminant component corresponding to the i-th training sample. We can evaluate the sum of the contribution of the i-th training sample to the feature extraction result of all the elements of S by $f_i = \sum_{p=1}^{n} |\alpha_i| k(x_i, x_p)$. We also calculate the variance $\hat{f}_i$ of $|\alpha_i| k(x_i, x_p)$ as follows: $\hat{f}_i = \frac{1}{n} \sum_{p=1}^{n} \left(f_{ip} - \bar{f}_i \right)^2$, where $\bar{f}_i$ represents the mean of $f_{ip} = |\alpha_i| k(x_i, x_p)$, $p = 1, 2, \ldots, n$. We define $F_i = f_i + tt * \hat{f}_i$. If $j = \arg\min_i F_i$, then from the viewpoint of statistics, we can consider that the j-th training sample contributes the least to the feature extraction result of all the elements. The coefficient tt allows the effect of the variance of f_{ip} to be partially considered. Indeed, a small F_i not only means that the contribution to the feature extraction result of the i-th training sample is small but also implies that the contribution also appears to be stable. As a result, if we remove the term $\alpha_j k(x_j, x_p)$ from the feature extraction result of each x_p, $p = 1, 2, \ldots, n$, this removal will cause only a little change to the feature extraction result. $\alpha_i k(x_i, x_p)$ is referred to as

the i-th feature extraction term of sample x_p. Since there are a number of feature extraction terms whose F have small values, we can identify all these terms and remove them from the feature extraction result of each x_p at one time. Then, we can reformulate the kernel method, solve the solution, and exploit it to perform computationally efficient feature extraction.

2.3 More analysis on the algorithm framework

In this subsection, we attempt to show the underlying rationale of the scheme presented in Sect. 2.2 from another viewpoint. According to the definition of the kernel function, we have $\sum_{p=1}^{n} |\alpha_i| k(x_i, x_p) = |\alpha_i| \sum_{p=1}^{n} \phi^T(x_i)\phi(x_p) = |\alpha_i| \cdot \sum_{p=1}^{n} \|\phi(x_p)\| . \|\phi(x_i)\| \cos\theta_{ip}$ where θ_{ip} stands for the angle between the sample vectors $\phi(x_i)$ and $\phi(x_p)$. We can say that the lower the value of $\cos\theta_{ip}$, the higher the correlation between $\phi(x_i)$ and $\phi(x_p)$. Thus, the value of $\cos\theta_{ip}$ also partially indicates the correlation between different sample vectors. From this, we know that the term removing rule shown in Sect. 2.2 indeed takes into account both the correlation between different sample vectors and the value of the coefficient α_i. While the term removing rule inclines to remove the feature extraction term whose discriminant component has a small absolute value, it is also probably that this rule removes the feature extraction term in which the corresponding training sample has high correlation with regard to all the other training samples. Thus, from this viewpoint, the term removing rule is also reasonable.

Furthermore, if the kernel function is set to the Gaussian function, then we have $\sum_{p=1}^{n} \|\phi(x_p)\| \cdot \|\phi(x_i)\| = \sqrt{k(x_p, x_p)k(x_i, x_i)} = 1$. As a result, $\sum_{p=1}^{n} |\alpha_i| k(x_i, x_p) = |\alpha_i| \cdot \sum_{p=1}^{n} \cos\theta_{ip}$. From this, we can conclude that when the Gaussian kernel function is adopted, the scheme proposed in Sect. 2.2 removes the contribution, to the feature extraction results, of the training samples with small discriminant component and with high statistical correlation with respect to other training samples.

We use Figs. 1 and 2 to describe the structure of the training phase of the kernel method and the reformulated kernel method, respectively. Figure 1 shows that all of $x_1, x_2, \ldots, x_n$ serve as the inputs of the training phase of the kernel method. The training phase exploits the desired outputs of the inputs to determine the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$. We can regard that the second layer of Fig. 1 plays a similar role of one hidden layer of the neural network. Figure 2 shows that the training phase of the reformulated kernel method has a similar structure as the kernel method except that its second layer has fewer nodes than the second layer of Fig. 1. Thus, we can say that the reformulated kernel method has a lower structure complexity, which implies that it is possible for the reformulated kernel method to generalize better than the kernel method.

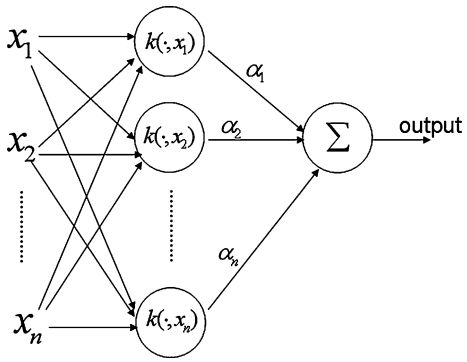

Fig. 1 The illustration of the kernel method

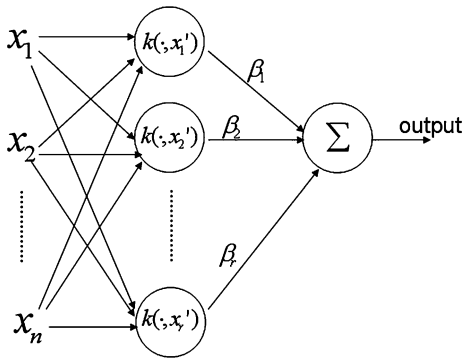

Fig. 2 The illustration of the reformulated kernel method

3 To reformulate KMSE and KPCA

3.1 Description of KMSE

KMSE is established on the basis of the following equation:

$$KA + \Xi = Y, \tag{4}$$

where

$$K = \begin{bmatrix} 1 & k(x_1, x_1) & \cdots & k(x_1, x_n) \\ 1 & k(x_2, x_1) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & & \vdots \\ 1 & k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}.$$

$\Xi = [e_1 \quad \ldots \quad e_n]^T$ stands for the error vector. $Y = [y_1 \quad y_2 \quad \ldots \quad y_n]^T$ represents the known output vector of samples. For two-class classification problems, Y represents the known class labels of all training

samples. For example, if the i-th training sample is from the first class, we can set y_i to 1; otherwise, we set y_i to -1. As presented in [16], A can be solved using the following equation:

$$A = (K^T K + \gamma I)^{-1} K^T Y. \tag{5}$$

where γ is a small positive constant and I is the identity matrix.

3.2 Improvement to KMSE

The main steps to implement the improvement to KMSE and the corresponding feature extraction and classification procedure are as follows:

Step 1 We solve A using (5).

Step 2 We evaluate the contribution, to the feature extraction results, of each training sample using the scheme described in Sect. 2.2. Then, we order the training samples in terms of their contributions to the feature extraction results and identify the first $n - r$ training samples that contributes little to the feature extraction results and denote them by $x_1^n, x_2^n, x_3^n, \ldots, x_{n-r}^n$, respectively. Since each of the kernel vectors $[k(x_1, x_i^n) \quad k(x_2, x_i^n) \quad \ldots \quad k(x_n, x_i^n)]^T$ corresponding to x_i^n, $i = 1, 2, \ldots, n - r$ must be one column of K, we identify these columns and remove them from K and denote the renewed K by K^1. In other words, K^1 is defined as $(K^1)_{ij} = k(x_i, x'_j)$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, r$. x'_i, $i = 1, 2, \ldots, r$ belong to the set $\{x'_i\} = \{x_1, x_2, \ldots, x_n\} - \{x_1^n, x_2^n, x_{n-r}^n\}$. x_1^n, x_2^n, x_{n-r}^n are the identified $n - r$ training samples that contribute little to the feature extraction results.

Step 3 Since the new KMSE system has the equation $K^1 A^1 + \Xi^1 = Y$, we solve A^1 using the equation $A^1 = (K^{1T} K^1)^{-1} K^{1T} Y$. We perform feature extraction for an arbitrary sample x using $y(x) = w'_0 + \sum_{i=1}^{r} \beta_i k(x'_i, x)$, where $A^1 = [w'_0 \quad \beta_1 \quad \beta_2 \quad \ldots \quad \beta_r]^T$.

Step 4 We extract the feature from an arbitrary sample x using $y(x) = w'_0 + \sum_{i=1}^{r} \beta_i k(x'_i, x)$.

Step 5 We exploit the feature extraction result to classify samples as follows. If $y(x) > 0$, we classify it into the first class; otherwise, we classify it into the second class.

3.3 Description of KPCA

In this subsection, we briefly describe KPCA. KPCA solves the m eigenvectors corresponding to the first m largest eigenvalues of the Gram matrix P defined as $(P)_{ij} = k(x_i, x_j)$ and takes the discriminant vectors of the feature space corresponding to these eigenvectors as transform axes to transform a sample into an m dimensional feature space. The feature extraction results of sample x can be formulated by [22]

$$Z_2 = \left[\sum_{j=1}^{N} \alpha_j^{(1)} k(x_j, x) \Big/ \sqrt{\lambda_1^{\alpha}} \quad \sum_{j=1}^{N} \alpha_j^{(2)} k(x_j, x) \Big/ \sqrt{\lambda_2^{\alpha}} \quad \ldots \quad \sum_{j=1}^{N} \alpha_j^{(m)} k(x_j, x) \Big/ \sqrt{\lambda_m^{\alpha}} \right]^T \tag{6}$$

where $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(m)}$ are respectively the m eigenvectors corresponding to the first m largest eigenvalues $\lambda_1^{\alpha}, \lambda_2^{\alpha}, \ldots, \lambda_m^{\alpha}$ of P. Note that $\alpha_j^{(i)}$ denotes the j-th component of the vector $\alpha^{(i)}$.

According to the nature of the PCA methodology, the first component of the feature extraction result can capture the data component that varies most. As a result, we regard that the first component of the feature extraction result is the most important among all the m components. We thus exploit only the first component to evaluate the contribution of a training sample to the feature extraction result. Since the first component is produced by the first transform axis, we also say that we exploit only the first transform axis to evaluate the "contribution".

3.4 Improvement to KPCA

In this subsection, we present the main steps to implement the improvement to KPCA and the corresponding feature extraction and classification procedure.

Step 1 We solve the eigenvector corresponding to the largest eigenvalue of the Gram matrix P of KPCA.

Step 2 We evaluate the contribution, to the feature extraction results, of each training sample using the scheme described in Sect. 2.2. Then, we order the training samples in terms of their contributions to the feature extraction results and identify the first $n - r$ training samples that contributes little to the feature extraction results and denote them by $x_1^n, x_2^n, x_3^n, \ldots, x_{n-r}^n$, respectively. Since each of the kernel vectors $[k(x_1, x_i^n) \; k(x_2, x_i^n) \ldots k(x_n, x_i^n)]^T$ corresponding to x_i^n, $i = 1, 2, \ldots, n - r$ must be one column of K, we identify these columns and remove them from K and denote the renewed K by K_1. In other words, $(K_1)_{ij} = k(x_i, x'_j)$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, r$. $\{x'_j\} = \{x_1, x_2, \ldots, x_n\} - \{x_1^n, x_2^n, \ldots, x_{n-r}^n\}$.

Step 3 We solve the m eigenvector corresponding to the first m largest eigenvalues of the following equation:

$$K_1(K_1)^T\beta = \lambda_i^{\beta} K_2 \beta, \tag{7}$$

where K_2 is defined as $(K_2)_{ij} = k(x_i', x_j')$, $i,j = 1,2,\ldots,r$ [22]. We perform feature extraction for sample x using

$$Z(x) = \left[\sum_{j=1}^{r}\beta_j^{(1)}k(x_j',x) \Big/ \sqrt{\lambda_1^{\beta}} \sum_{j=1}^{r}\beta_j^{(2)}k(x_j',x) \Big/ \sqrt{\lambda_2^{\beta}} \ \ldots \ \sum_{j=1}^{r}\beta_j^{(m)}k(x_j',x) \Big/ \sqrt{\lambda_m^{\beta}}\right]^T.$$

λ_i^{β}, $i = 1,2,\ldots,m$ are the first m largest eigenvalues of K^1, respectively, and $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(m)}$ respectively denote the corresponding m eigenvectors of (7). $\beta^{(i)} = [\beta_1^{(i)} \ \beta_2^{(i)} \ \ldots \ \beta_r^{(i)}]^T$, $i = 1,2,\ldots,m$. x_i', $i = 1,2,\ldots,r$ still belongs to the set $\{x_i'\} = \{x_1, x_2, \ldots, x_n\} - \{x_1^n, x_2^n, x_{n-r}^n\}$.

Step 4 We exploit the feature extraction result $Z(x)$ and the nearest neighbor classifier to classify samples.

3.5 Characteristics of the proposed method

The proposed method has the following characteristics. First, it provides an intuitive and very mathematically tractable means to produce computationally efficient kernel-method-based feature extraction procedure. Starting from the viewpoint of numerical approximation, the proposed method can find the training samples that statistically contribute much to the feature extraction results and exploit them to reformulate KMSE or KPCA. Second, the proposed method also has a computationally efficient training phase. Indeed, the main computational burden of its training phase is to solve the equations of naïve KMSE (or naïve KPCA) and the reformulated KMSE (or KPCA) model. Since naïve KMSE or naïve KPCA also should solve one equation, the extra cost that the proposed method needs is just from solving the equation of the reformulated KMSE (or KPCA) model. On the contrary, other improvements to KPCA or KMSE such as the ones proposed in [16, 22] usually improve KPCA or KMSE at a very high extra computational cost.

4 Experimental results

We used several benchmark datasets [16] to test the proposed method and other methods. Every dataset except for "Splice" includes 100 partitions each consisting of one training sample subset and one test sample subset. We adopted the Gaussian kernel in the form of $k(x,y) = \exp(-\|x-y\|^2/\sigma)$. When testing the methods using each dataset, we took the first training subset as the training set and took all the testing subsets as the testing set. Because every test subset obtained an error rate, we figured out the mean of the error rates of all the testing subsets of one dataset and showed it by the table.

4.1 Experiments on the reformulated KPCA

This subsection presents the experiments on KMSE. The kernel function parameter σ was set to the squared norm of the covariance matrix of the training samples. Tables 1 and 2 show the means of the classification error rates of our KPCA method and the naïve KPCA method, respectively. Table 3 shows the mean of the classification error rat of the KPCA proposed in [22]. In these tables, the first column shows the value of r. r varies from 10 to 44. In Table 2, r denotes the dimension of the features obtained using naïve KPCA. In Table 1, r represents both the dimension of the features obtained using our KPCA and the number of the identified training samples that contribute much to the feature extraction results. In Table 3, the different rows show similarly except that the used method is the KPCA proposed in [22]. For each dataset shown in one table, the second row to the last row respectively show the variation with r of the mean of the classification error rates of the corresponding KPCA method. These tables show that averagely the classification error rate obtained using our KPCA method is not higher than the error rate obtained using the KPCA method in [22] and is only little higher than the error rate obtained using naive KPCA. Moreover, as shown in early, our KPCA method has a much more computationally efficient training phase than the KPCA method in [22].

Figures 3 and 4 use the magenta circles to respectively show the location in the original space and the location in

Table 1 The mean of the classification error rat of our KPCA method

r	Splice	Thyroid	Diabetes	Heart	Banana	German
10	0.2484	0.0099	0.1160	0.1049	0.1355	0.1153
12	0.2417	0.0217	0.1274	0.1106	0.1367	0.1077
14	0.2336	0.0224	0.1215	0.0819	0.1384	0.0964
16	0.2288	0.0281	0.1178	0.0853	0.1384	0.0991
18	0.2313	0.0316	0.1267	0.0891	0.1368	0.1048
20	0.2307	0.0225	0.1344	0.0872	0.1356	0.1055
22	0.2266	0.0145	0.1222	0.0836	0.1363	0.1047
24	0.2273	0.0235	0.1210	0.0777	0.1400	0.1106
26	0.2311	0.0193	0.1240	0.0736	0.1465	0.1116
28	0.2266	0.0224	0.1240	0.0907	0.1379	0.1043
30	0.2181	0.0224	0.1233	0.0937	0.1352	0.1085
32	0.2177	0.0377	0.1241	0.0851	0.1325	0.0997
34	0.1994	0.0329	0.1238	0.0965	0.1361	0.1075
36	0.2051	0.0331	0.1252	0.0840	0.1352	0.1044
38	0.1937	0.0289	0.1205	0.0976	0.1369	0.1009
40	0.1928	0.0195	0.1229	0.0973	0.1354	0.0990
42	0.1925	0.0195	0.1210	0.1052	0.1356	0.1003
44	0.1944	0.0268	0.1195	0.1056	0.1344	0.1004

Table 2 The mean of the classification error rat of naïve KPCA

r	Splice	Thyroid	Diabetes	Heart	Banana	German
10	0.2168	0.0099	0.1077	0.0828	0.1371	0.0988
12	0.2153	0.0099	0.1077	0.0870	0.1380	0.1051
14	0.2237	0.0099	0.1120	0.0910	0.1367	0.0962
16	0.2078	0.0099	0.1171	0.0735	0.1373	0.0960
18	0.2035	0.0099	0.1167	0.0735	0.1380	0.0932
20	0.2062	0.0099	0.1171	0.0735	0.1378	0.0947
22	0.2060	0.0099	0.1199	0.0769	0.1382	0.0975
24	0.2031	0.0099	0.1132	0.0803	0.1380	0.0950
26	0.2048	0.0099	0.1140	0.0803	0.1380	0.0961
28	0.2033	0.0099	0.1175	0.0803	0.1380	0.0908
30	0.1984	0.0099	0.1169	0.0803	0.1380	0.0934
32	0.2032	0.0099	0.1160	0.0803	0.1380	0.0945
34	0.2025	0.0099	0.1169	0.0803	0.1380	0.0908
36	0.1969	0.0099	0.1172	0.0803	0.1380	0.0946
38	0.1898	0.0099	0.1141	0.0803	0.1380	0.0975
40	0.1908	0.0099	0.1130	0.0726	0.1380	0.0970
42	0.1894	0.0099	0.1130	0.0726	0.1380	0.0959
44	0.1899	0.0099	0.1171	0.0726	0.1380	0.0968

Table 3 The mean of the classification error rat of the KPCA proposed in [22]

r	Splice	Thyroid	Diabetes	Heart	Banana	German
10	0.2625	0.0147	0.1133	0.1051	0.1369	0.1071
12	0.2560	0.0223	0.1085	0.0966	0.1365	0.0964
14	0.2435	0.0229	0.1009	0.0986	0.1355	0.0996
16	0.2532	0.0195	0.1108	0.1018	0.1372	0.0954
18	0.2472	0.0193	0.1178	0.1030	0.1368	0.0934
20	0.2371	0.0195	0.1077	0.1009	0.1348	0.0993
22	0.2335	0.0241	0.1046	0.0927	0.1351	0.1048
24	0.2394	0.0193	0.1133	0.0905	0.1370	0.1000
26	0.2182	0.0188	0.1070	0.0919	0.1344	0.1038
28	0.2127	0.0228	0.1108	0.0865	0.1317	0.1036
30	0.2114	0.0228	0.1179	0.0942	0.1350	0.0983
32	0.2122	0.0269	0.1221	0.0940	0.1339	0.1036
34	0.2086	0.0291	0.1130	0.0974	0.1340	0.1092
36	0.2059	0.0265	0.1097	0.1010	0.1357	0.1046
38	0.2154	0.0228	0.1133	0.1050	0.1372	0.1081
40	0.2178	0.0285	0.1132	0.0915	0.1380	0.1086
42	0.2134	0.0269	0.1198	0.0909	0.1367	0.1085
44	0.2126	0.0269	0.1224	0.0838	0.1405	0.1046

the feature space of the identified training samples that contribute much to the feature extraction results. They intuitively show that since the mapping function corresponding to the kernel trick is a nonlinear function, in the original space and in the feature space, samples might have

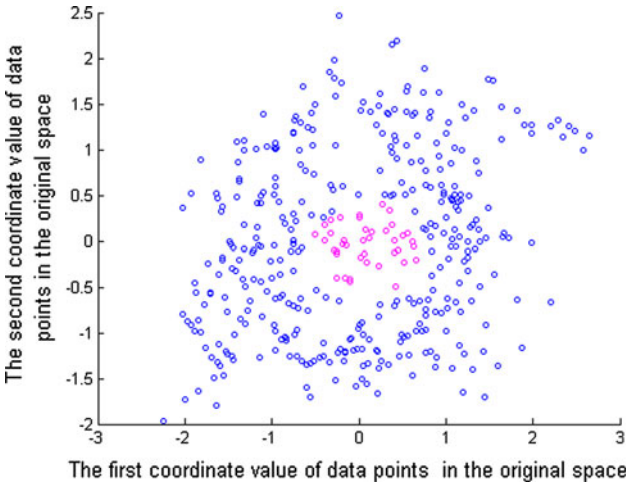

Fig. 3 Location in the original space of the 40 identified training samples that contribute much to the feature extraction results. *Magenta circles* denote the identified training samples, and *blue circles* denote other training samples. The samples are all from the first training subset of the dataset "Banana"

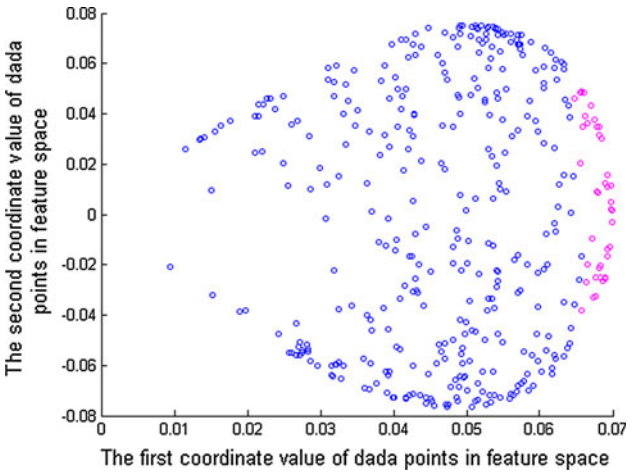

Fig. 4 Location in the feature space of the 40 identified training samples that contribute much to the feature extraction results. *Magenta circles* denote the identified training samples, and *blue circles* denote other training samples. The samples are all from the first training subset of the dataset "Banana". Only the first two components of the features obtained using our KPCA method are shown in this figure

entirely different locations and distribution. Due to the same reason, it is feasible for us to identify the samples that contribute much to the feature extraction results through analysis on the feature space, whereas it is not a right way to identify these samples through the analysis on the original space. Indeed, in this paper, we also exploit the analysis on the feature extraction results in the feature space to obtain the reformulated KMSE method and the reformulated KPCA method. Figure 5 shows the location

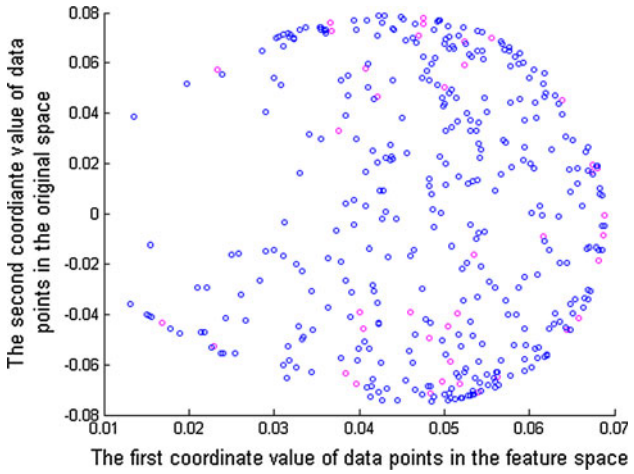

Fig. 5 Location in the feature space of the 40 identified training samples that were used to construct the KPCA method in [22]. *Magenta circles* denote the identified training samples, and *blue circles* denote other training samples. The samples are all from the first training subset of the dataset "Banana". Only the first two components of the features obtained using our KPCA method are shown in this figure

in the feature space of the 40 identified training samples that were used to construct the KPCA method in [22]. The samples in Figs. 3, 4, and 5 are all from the first training subset of the dataset "Banana". The difference between Figs. 4 and 5 also indicate that different improvements to KPCA may use entirely different samples to construct the improved KPCA method. On the other hand, these different improvements incline to obtain similar classification performance as shown in Tables 1 and 3.

4.2 Experiments on the reformulated KMSE

This subsection describes the experiments on KMSE. The kernel function parameter σ was set as follows: First, we calculated the mean of all the training samples. Then, we calculated the squared norm of the result of each training sample subtracted from the mean and set σ to the mean of these squared norms. Table 4 shows the mean of the classification error rat of the naive KMSE method on different datasets. Tables 5 and 6 show the mean of the classification error rates of our KMSE method and the KMSE method proposed in [16], respectively. In Tables 5 and 6, the first column shows the ratio of r to N, the total number of the training samples. From these tables, we see that while r/N increases, the classification errors of both our KMSE method and the KMSE method proposed in [14] decreases. In addition, our KMSE method classifies more efficiently than the KMSE method proposed in [14]. If r/N is not too small, the classification error rate obtained using our KMSE method will be close to the error rate

Table 4 The mean of the classification error rat of the naive KMSE method

Cancer	Splice	Thyroid	Diabetes	Heart	Solar	Banana	German
0.1825	0.0909	0.135	0.1821	0.1043	0.3243	0.1103	0.1512

Table 5 The mean of the classification error rat of our KMSE method

r/N	Cancer	Splice ($tt = 0.1$)	Thyroid	Diabetes	Heart	Banana	German
0.03	0.2462	0.2179	0.0799	0.2371	0.1761	0.1333	0.2339
0.04	0.2504	0.1871	0.0656	0.2385	0.1651	0.1161	0.2276
0.05	0.2600	0.1788	0.0713	0.2391	0.1718	0.1136	0.2152
0.06	0.2596	0.1658	0.0712	0.2348	0.1710	0.1119	0.2187
0.07	0.2597	0.1564	0.0505	0.2302	0.1681	0.1121	0.2207
0.08	0.2555	0.1453	0.0460	0.2295	0.1668	0.1125	0.2282
0.09	0.2400	0.1390	0.0409	0.2226	0.1729	0.1117	0.2215
0.10	0.2566	0.1357	0.0361	0.2235	0.1622	0.1115	0.2153
0.11	0.2447	0.1342	0.0361	0.2297	0.1661	0.1117	0.2171
0.12	0.2339	0.1221	0.0233	0.2305	0.1758	0.1110	0.2184
0.13	0.2126	0.1191	0.0185	0.2225	0.1574	0.1111	0.2129
0.14	0.2079	0.1157	0.0185	0.2237	0.1560	0.1119	0.2064
0.15	0.2087	0.1166	0.0185	0.2195	0.1639	0.1111	0.2040
0.16	0.2301	0.1152	0.0185	0.2170	0.1439	0.1111	0.1977
0.17	0.2319	0.1129	0.0185	0.2154	0.1490	0.1113	0.2008
0.18	0.2323	0.1089	0.0135	0.2074	0.1513	0.1115	0.1917
0.19	0.2200	0.1113	0.0135	0.2065	0.1481	0.1115	0.1940
0.20	0.1766	0.1076	0.0135	0.2058	0.1586	0.1107	0.1868
0.21	0.1723	0.1084	0.0135	0.2084	0.1448	0.1103	0.1891
0.22	0.1723	0.1101	0.0135	0.1968	0.1347	0.1100	0.1713
0.23	0.1648	0.1100	0.0135	0.1947	0.1288	0.1102	0.1709
0.24	0.1694	0.1063	0.0135	0.1931	0.1318	0.1096	0.1710

For all the datasets except "Splice", tt was set to 1

obtained using the naïve KMSE method. For example, when $r/N = 0.23$, our KMSE method obtained the means of the classification error rates, 0.1648, 0.0135, 0.1947, 0.1102 for datasets "Cancer", "Thyroid", "Diabetes" and "Banana", respectively. The means of the classification error rates that the naïve KMSE method obtained for these datasets are 0.1825, 0.135, 0.1821, and 0.1103, respectively.

5 Conclusion

The method proposed in this paper is able to obtain computationally efficient feature extraction procedure for the kernel method. The method solves the efficient feature extraction problem from the viewpoint of numerical

Table 6 The mean of the classification error rat of the KMSE method proposed in [14]

r/*N*	Cancer	Splice	Thyroid	Diabetes	Heart	Banana	German
0.03	0.2648	0.2402	0.0752	0.2385	0.1732	0.1313	0.2394
0.04	0.2656	0.2092	0.0752	0.2328	0.1644	0.1284	0.2221
0.05	0.2617	0.1734	0.0623	0.2272	0.1528	0.1176	0.2288
0.06	0.2617	0.1590	0.0623	0.2258	0.1688	0.1150	0.2223
0.07	0.2578	0.1528	0.0513	0.2217	0.1682	0.1134	0.2151
0.08	0.2478	0.1498	0.0460	0.2196	0.1761	0.1131	0.2128
0.09	0.2391	0.1429	0.0460	0.2213	0.1761	0.1140	0.2140
0.10	0.2158	0.1371	0.0460	0.2194	0.1624	0.1119	0.2153
0.11	0.2121	0.1299	0.0460	0.2164	0.1662	0.1121	0.2145
0.12	0.2113	0.1274	0.0409	0.2159	0.1636	0.1125	0.2232
0.13	0.2008	0.1283	0.0420	0.2195	0.1591	0.1108	0.2113
0.14	0.1930	0.1254	0.0328	0.2195	0.1594	0.1110	0.2115
0.15	0.1849	0.1252	0.0328	0.2104	0.1691	0.1119	0.2029
0.16	0.1751	0.1235	0.0183	0.2111	0.1655	0.1121	0.2087
0.17	0.1751	0.1184	0.0183	0.2064	0.1573	0.1119	0.2100
0.18	0.1858	0.1222	0.0183	0.2107	0.1575	0.1117	0.1995
0.19	0.1826	0.1203	0.0183	0.2079	0.1645	0.1115	0.2051
0.20	0.1718	0.1210	0.0183	0.2009	0.1609	0.1117	0.1965
0.21	0.1929	0.1166	0.0183	0.2053	0.1452	0.1119	0.1874
0.22	0.1853	0.1161	0.0135	0.2042	0.1402	0.1121	0.1853
0.23	0.1847	0.1136	0.0135	0.2024	0.1375	0.1126	0.1827
0.24	0.1800	0.1148	0.0135	0.2027	0.1363	0.1126	0.1782

approximation. The method identifies the terms that contribute little to the feature extraction results and improves the feature extraction efficiency by removing the corresponding feature extraction terms from the feature extraction procedure. This method has the following rationale: when one training sample corresponds to a discriminant component with a small absolute value and statistically has high correlation with regard to all the other training samples, the feature extraction term associated with this training sample can be removed from the feature extraction procedure. The proposed method also has an efficient training phase. The experimental results also show that the proposed method can obtain a good classification performance. The figures on the locations in the original space and the location in the feature space of the identified training samples that contribute much to the feature extraction results is very helpful to intuitively show the geometrical relation between the identified training samples and other training samples.

Acknowledgments This article is partly supported by Program for New Century Excellent Talents in University (NCET-08-0156), National Nature Science Committee of China under grant Nos. 61071179, 90820306, 60902099, and 61001037, the Fundamental Research Funds for the Central Universities (HIT.NSRIF. 2009130), 863 Program Project under Grant No. 2007AA01Z195 and the CERG fund from the HKSAR Government and the central fund from Hong Kong Polytechnic University.

References

1. Muller KR, Mika S, Ratsch G et al (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Netw 12(2):181–201
2. Vapnik VN (1995) The nature of statistical learning theory. Springer, Berlin
3. Xu J, Zhang X, Li Y (2001) Kernel MSE algorithm: a unified framework for KFD, LS-SVM and KRR. In: Proceedings of the international joint conference on neural networks (IJCNN-2001), Washington, DC, pp 1486–1491
4. Muller K-R, Mika S, Rätsch G, Tsuda K, Schölkopf B (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Netw 12(1):181–201
5. Girolami M (2002) Mercer kernel based clustering in feature space. IEEE Trans Neural Netw 13(4):669–688
6. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge
7. Yang J, Frangi AF, Yang J-Y, Zhang D, Jin Z (2005) KPCA Plus LDA: A complete kernel fisher discriminant framework for feature extraction and recognition. IEEE Trans Pattern Anal Mach Intell 27(2):230–244
8. Schölkopf B, Smola A (2002) Learning with kernels. MIT Press, Cambridge
9. Xu Y, Yang JY, Lu JF, Yu DJ (2004) An efficient renovation on kernel Fisher discriminant analysis and face recognition experiments. Pattern Recogn 37:2091–2094
10. Scholkopf B, Mika S, Burges C, Knirsch P, Muller KR, Ratsch G, Smola A (1999) Input space versus feature space in kernel-based methods. IEEE Trans Neural Netw 10(5):1000–1017
11. Xu Y, Yang JY, Yang J (2004) A reformative Fisher discriminant analysis. Pattern Recogn 37:1299–1302
12. Schölkopf B, Smola A, Muller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319
13. Mika S, Rätsch G, Weston J, Schölkopf B, Müller K-R (1999) Fisher discriminant analysis with kernels. In: Neural networks for signal processing IX. IEEE, New York, pp 41–48
14. Xu Y, Yang JY, Lu JF (2005) An efficient kernel-based nonlinear regression method for two-class classification. Proceedings of the fourth international conference on machine learning and cybernetics, Guangzhou, China, 18–21 August 2005, pp 4442–4445
15. Steve B, Kian L (2002) Nonlinear Fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. Neural Netw 15(1):263–270
16. Xu Y, Zhang D, Jin Z, Li M, Yang J-Y (2006) A fast kernel-based nonlinear discriminant analysis for multi-class problems. Pattern Recogn 39(6):1026–1033
17. Tipping ME (2000) Sparse kernel principal component analysis. In: Leen TK, Dietterich TG, Tresp V (eds) NIPS 2000: Neural information processing systems. MIT Press, Cambridge, pp 633–639
18. Cawley GC, Talbot NLC (2003) Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. Pattern Recogn 36(11):2585–2592
19. Kim S-W, Oommen BJ (2008) On using prototype reduction schemes to optimize kernel-based fisher discriminant analysis. IEEE Trans Syst Man Cybern Part B Cybern 38(2):564–570

20. Burges CJC, Scholkopf B (1997) Improving the accuracy and speed of support vector learning machines. In: Mozer M, Jordan M, Petsche T (eds) Advances in neural information processing systems, vol 9. MIT Press, Cambridge, pp 375–381
21. Mika S, Ratsch G, Müller K-R (2001) A mathematical programming approach to the kernel Fisher algorithm. Adv Neural Inf Process Syst 13:591–597
22. Xu Y, Zhang D, Song F et al (2007) A method for speeding up feature extraction based on KPCA. Neurocomputing 70: 1056–1061