Front. Electr. Electron. Eng. China 2011, 6(1): 43–55
DOI 10.1007/s11460-011-0129-z

RESEARCH ARTICLE

Jian YANG

Kernel feature extraction methods observed from the viewpoint of generating-kernels

Abstract This paper introduces an idea of generating a kernel from an arbitrary function by embedding the training samples into the function. Based on this idea, we present two nonlinear feature extraction methods: generating kernel principal component analysis (GKPCA) and generating kernel Fisher discriminant (GKFD). These two methods are shown to be equivalent to the function-mapping-space PCA (FMS-PCA) and the function-mapping-space linear discriminant analysis (FMS-LDA) methods, respectively. This equivalence reveals that the generating kernel is actually determined by the corresponding function map. From the generating kernel point of view, we can classify the current kernel Fisher discriminant (KFD) algorithms into two categories: KPCA + LDA based algorithms and straightforward KFD (SKFD) algorithms. The KPCA + LDA based algorithms directly work on the given kernel and are not suitable for non-kernel functions, while the SKFD algorithms essentially work on the generating kernel from a given symmetric function and are therefore suitable for non-kernels as well as kernels. Finally, we outline the tensor-based feature extraction methods and discuss ways of extending tensor-based methods to their generating kernel versions.

Keywords kernel methods, feature extraction, principal component analysis (PCA), Fisher linear discriminant analysis (FLD or LDA), tensor-based methods

1 Introduction

Over the past ten years, kernel-based learning machines, represented by support vector machines (SVMs) [1], kernel principal component analysis (KPCA), and kernel Fisher discriminant analysis (KFD), have aroused considerable interest in the fields of pattern recognition and machine learning [2]. KPCA was originally developed by Schölkopf in 1998 [3], while KFD was first proposed by Mika in 1999 [4,5]. Subsequent research saw the development of a series of kernel-based methods [6–27]. Typically, Baudat [6] and Roth [7] presented a multiclass KFD version based on the generalized discriminant analysis. Yang [16,17] suggested a two-phase KFD framework, i.e., KPCA plus Fisher linear discriminant analysis (LDA). Bach and Jordan [22] developed a kernel version of the independent component analysis (ICA).

In kernel-based methods, the "kernel" usually refers to a positive semi-definite function (or Mercer kernel) [11,12]. That is, the associated Gram matrix of the function is positive semi-definite for any finite subset of the input space. Two most popular positive semi-definite kernels are the polynomial kernel and the Gaussian kernel. Some techniques have been given regarding how to construct a new kernel function from the known kernels [11,12], features [26], or wavelet functions [27]. In general, we need to prove that the constructed function is a kernel before applying it with an existing kernel-based method.

In practice, however, some functions (not necessarily kernels) have been used and demonstrated to be effective in a number of cases. For example, the sigmoid "kernel" does not actually define a positive semi-definite Gram matrix, hence are not a real kernel by definition. Nevertheless, the sigmoid "kernel" has been used in support vector machines [11] and successfully applied to handwritten digit recognition [28]. A fuzzy sigmoid "kernel" based support vector classifier was demonstrated effective on ten databases selected from the UCI machine learning repository [29]. Recently, the fractional power polynomial model, which is not necessarily a positive semi-definite function, have been applied with a kernel Fisher analysis method to face recognition and achieved a very good performance on the face recognition grand challenge (FRGC) Experiments 1 and 4 [20]. The Gaus-

Received July 16, 2010; accepted November 23, 2010

Jian YANG (✉)
School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China
E-mail: csjyang@mail.njust.edu.cn

sian kernel function has been modified into a new function by replacing the Euclidean distance measure with a Chi square distance measure [30] and applied with a KFD algorithm in Ref. [19]. The modified Gaussian function is not necessarily a kernel, but it can take advantage of the local binary pattern (LBP) [30] features and turns out to be more powerful than the Gaussian kernel on the FRGC Experiment 1. Why can these kinds of functions (not necessarily kernels) be directly applied in some existing kernel methods, e.g. KFD? This paper will address this problem in detail.

In this paper, we seek to build a bridge from functions to kernels. We suggest a way of generating a kernel from a given arbitrary function and a set of training samples. The resulting kernel is thus called a (function) generating kernel. The generating kernel idea provides us more flexibility to construct a kernel for recognition purposes. For example, we can first design a function that can take advantage of the prior knowledge of the problem domain (just as that was done in Ref. [19]), and then based on this function, we can generate a kernel more suitable for solving the problem. In theory, the generating kernel can be applied to any kernel-based methods. For a given function, through the process of generating kernel and existing kernel-based methods, we can easily derive the function-based learning algorithms (i.e., generating-kernel based methods), as illustrated in Fig. 1.

This paper focuses on two generating-kernel based nonlinear feature extraction methods: generating kernel principal component analysis (GKPCA) and generating kernel Fisher discriminant (GKFD). To gain more insight into the two methods, we further reveal their connections to the function-mapping-space principal component analysis (FMS-PCA) method and the function-mapping-space Fisher linear discriminant analysis (FMS-LDA) method. The basic idea of FMS-PCA and FMS-LDA was presented in Refs. [24,25], where the function map is called function replacement operator. We show that GKPCA is equivalent to FMS-PCA and GKFD is equivalent to FMS-LDA. This equivalence reveals that the generating kernel is actually determined by the corresponding function map.

This paper also analyzes the current KFD algorithms from the generating kernel point of view and divides these algorithms into two categories: KPCA + LDA based algorithms [6,16,17] and Straightforward KFD (SKFD) algorithms [4,19]. SKFD is shown to be equivalent to GKFD when the given function is symmetric. This equivalence reveals that the SKFD algorithms are essentially generating kernel based methods. This is the underlying reason why the SKFD algorithms are suitable for non-kernel functions.

In literature, an idea most relevant to ours was presented in the generalized support vector machines (GSVMs) paper [31]. In the paper, two SVM variants, quadratic programming SVM and linear programming SVM, were proposed based on general "kernels". In the formulation of the quadratic programming SVM, if the matrix $\boldsymbol{H}$ in the convex quadratic function was set to be an identity matrix, the optimization problem that involves the quadratic programming SVM involves is always a solvable convex quadratic program for any "kernel" (e.g., non-mercer kernel). This can be viewed as an application of generating kernel idea to SVMs. The GSVM paper addresses the problem of how to generalize an SVM to suit for a general "kernel", whereas this paper goes one step further to address the problem of how to build a Mercer kernel from a given function. More importantly, we apply the generating kernel idea to kernel-based feature extraction methods and reveal something meaningful underlying these methods.

The remainder of the paper is organized as follows. Section 2 addresses the basic idea of generating kernels from functions. Section 3 presents GKPCA method and reveals its equivalent relationship with FMS-PCA. In Sect. 4, we re-examine the current KFD algorithms from the generating kernel point of view and divide these algorithms into two categories: original kernel-based methods and generating kernel-based methods. In Sect. 5, we outline the tensor-based methods and discuss ways of extending tensor-based methods to their generating kernel versions. Brief conclusions are finally offered in Sect. 6.

2 Generating kernels from functions

This section first overviews the basic concept of kernels and then suggests a way to construct a kernel from a given arbitrary function.

2.1 Basic concept of kernels

When the data are nonlinearly separable in the input space, we may consider to nonlinearly map the data into a feature space, where the data becomes linearly separable. This makes it possible to directly use some linear feature extraction or classification methods in the feature space. In practice, however, it is still difficult to do so because the high (or even infinite) dimensionality of

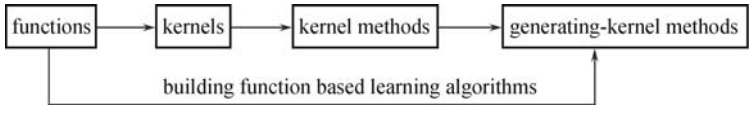

Fig. 1 Process of constructing function-based learning algorithms

feature map space makes the computation of inner products very time-consuming (or impossible). Fortunately, kernel techniques can be introduced to tackle this problem. The inner products in the feature space can be computed efficiently using a direct function of data in the input space. The explicit mapping process is not required at all. A function that performs this direct computation is known as a kernel function, which is formally defined as follows [11,12].

Definition 1 (kernel function) A kernel k is a function that for all $\boldsymbol{x}, \boldsymbol{z} \in \Xi$ satisfies

$$k(\boldsymbol{x}, \boldsymbol{z}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z}), \tag{1}$$

where Φ is a mapping from the input space Ξ to the feature space Φ, i.e.,

$$\Phi : \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}) \in \Phi. \tag{2}$$

Definition 2 (Gram matrix) Given a finite dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\} \subset \Xi$ and a function $g : \Xi \times \Xi \mapsto P$, the $M \times M$ matrix $\boldsymbol{G}$ with elements $\boldsymbol{G}_{ij} = g(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called the Gram matrix of the function g with respect to the dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$.

The Gram matrix of kernel function k with respect to $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ is often referred to as the kernel matrix and denoted by $\boldsymbol{K}$ with elements $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. It is easy to prove that any Gram matrix of a kernel function (kernel matrix) is positive semi-definite [11,12].

The Gram matrix is an important concept in kernel-based methods. It provides an alternative tool to characterize kernel. Since the definition of kernel is associated with a mapping Φ, it is inconvenient to directly use this definition to judge if a function is a kernel because the mapping Φ is generally implicit. We want a method that characterizes kernels without knowing the explicit mapping. Actually, the positive semi-definite property of the Gram matrix forms a basis for this characterization.

Theorem 1 (Characterization of Kernels) [12] Let Ξ be the input space. A function $k : \Xi \times \Xi \mapsto P$ (or X) is kernel if and only if for any $M \in N$ and any finite dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\} \subset \Xi$, the associated Gram matrix is positive semi-definite.

Theorem 1 tells us that if a function satisfies positive semi-definite property (i.e., for any finite subset of the input space, the associated Gram matrix is positive semi-definite), it must be a kernel. That is, in this case, there must exist a feature map Φ from the input space Ξ to the feature space Φ (a Hilbert space) such that $k(\boldsymbol{x}, \boldsymbol{z}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z})$ for all $\boldsymbol{x}, \boldsymbol{z} \in \Xi$.

2.2 Generating kernels from functions

As we know, some demonstrated-to-be-effective functions, such as the sigmoid "kernel" and fractional power polynomial model:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh(a(\boldsymbol{x} \cdot \boldsymbol{z}) + b), \quad a > 0, \quad b < 0, \tag{3}$$

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x} \cdot \boldsymbol{z})^r, \quad \text{where } r \text{ is a fraction}, \tag{4}$$

are not necessarily kernels because their associated Gram matrices are not positive semi-definite even for a given set of training data. In the following, we will discuss how to build a kernel from a given arbitrary function. Note that the given function is not necessarily a kernel. In other words, its associated Gram matrix might not be positive semi-definite.

For a given set of training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\} \subset \Xi$, and a function g, we can construct the following function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}) g(\boldsymbol{x}_l, \boldsymbol{z}), \quad \boldsymbol{x}, \boldsymbol{z} \in \Xi. \tag{5}$$

This function has a desirable property:

Proposition 1 The function defined in Eq. (5) is a kernel.

Proof For an arbitrary finite dataset $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_q\} \subset \Xi$, the associated Gram matrix of the function k is

$$\begin{aligned} \boldsymbol{K} &= [k(\boldsymbol{z}_i, \boldsymbol{z}_j)]_{q \times q} \\ &= \left[\sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{z}_i) g(\boldsymbol{x}_l, \boldsymbol{z}_j) \right]_{q \times q} \\ &= \boldsymbol{G}_{M \times q}^{\mathrm{T}} \boldsymbol{G}_{M \times q}, \end{aligned} \tag{6}$$

where $\boldsymbol{G}_{M \times q} = [g(\boldsymbol{x}_l, \boldsymbol{z}_i)]_{M \times q}$. Since $\boldsymbol{G}_{M \times q}^{\mathrm{T}} \boldsymbol{G}_{M \times q}$ is always positive semi-definite, the Gram matrix of the function k, $\boldsymbol{K}$, is positive semi-definite. The function defined in Eq. (5) is therefore a kernel from Theorem 1.

Proposition 1 indicates that for a given set of training samples, we can generate a kernel using Eq. (5) from a given arbitrary function. This kernel is thus called (function) generating kernel in this paper. Obviously, the generating kernel is with respect to a given set of training samples. A function may generate different kernels given different training sample sets. It is the embedding of training samples (into a function) that turns a function into a kernel. The generating kernel, therefore, takes advantage of the information of training samples.

Actually, we can understand the generating kernel from the kernel-definition point of view. To this end, let us construct the following map for a given function g and a set of training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$:

$$\Psi : \boldsymbol{x} \mapsto [g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})]^{\mathrm{T}}. \tag{7}$$

This map is called the function map[1)] with respect to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, which maps a sample point $\boldsymbol{x}$ in the input space into its image in the M-dimensional function-mapping space. This function map can determine the following kernel

$$k(\boldsymbol{x}, \boldsymbol{z}) = \Psi(\boldsymbol{x}) \cdot \Psi(\boldsymbol{z}) = \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}) g(\boldsymbol{x}_l, \boldsymbol{z}), \quad (8)$$

which is exactly the generating kernel defined in Eq. (5). This indicates that the generating kernel in Eq. (5) is actually determined by the function map in Eq. (7).

In theory, the (function) generating kernel can be applied to any kernel-based learning algorithms. In the following sections, we will focus on the examination of the two popular feature extraction methods, KPCA and KFD, using the generating kernel idea.

3 GKPCA

In this section, we first outline KPCA method and then present GKPCA method which is suitable for function (not necessarily a kernel) based learning. To provide more insight into GKPCA method, we further reveal the equivalent relationship of GKPCA and FMS-PCA.

3.1 Outline of KPCA

Given a set of M training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ in $\mathbb{R}^n$, the covariance operator on the feature space Φ can be constructed by

$$\boldsymbol{S}_{\mathrm{t}}^{\Phi} = \sum_{j=1}^{M} \left[\Phi(x_j) - \boldsymbol{m}_0^{\Phi}\right] \left[\Phi(x_j) - \boldsymbol{m}_0^{\Phi}\right]^{\mathrm{T}}, \quad (9)$$

where $\boldsymbol{m}_0^{\Phi} = \dfrac{1}{M} \sum\limits_{j=1}^{M} \Phi(\boldsymbol{x}_j)$, and Φ is a map into feature space Φ which is determined by a kernel k. In a finite-dimensional Hilbert space, the operator $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ is generally called the covariance matrix.

Let

$$\boldsymbol{X} = [\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \ldots, \Phi(\boldsymbol{x}_M)]$$

and

$$\boldsymbol{Q} = \left[\Phi(\boldsymbol{x}_1) - \boldsymbol{m}_0^{\Phi}, \Phi(\boldsymbol{x}_2) - \boldsymbol{m}_0^{\Phi}, \ldots, \Phi(\boldsymbol{x}_M) - \boldsymbol{m}_0^{\Phi}\right].$$

It is easy to show that

$$\boldsymbol{Q} = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{D} = \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{D}), \quad (10)$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{D} = (1/M)_{M \times M}$. The operator $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ can thus be rewritten as

$$\boldsymbol{S}_{\mathrm{t}}^{\Phi} = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{D})[(\boldsymbol{I} - \boldsymbol{D})\boldsymbol{X}]^{\mathrm{T}}. \quad (11)$$

The eigenvector $\boldsymbol{\beta}$ of $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ can be linearly expanded by [3]

$$\boldsymbol{\beta} = \sum_{i=1}^{M} a_i \left[\Phi(\boldsymbol{x}_i) - \boldsymbol{m}_0^{\Phi}\right] = \boldsymbol{Q}\boldsymbol{\alpha}, \quad (12)$$

where $\boldsymbol{\alpha} = [a_1, a_2, \ldots, a_M]^{\mathrm{T}}$.

To obtain the expansion coefficients, let us construct the $M \times M$ Gram matrix $\boldsymbol{K} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ with elements $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and define a centralized Gram matrix $\boldsymbol{K}$ as follows

$$\begin{aligned} \tilde{\boldsymbol{K}} &= \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{D})\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}(\boldsymbol{I} - \boldsymbol{D}) \\ &= (\boldsymbol{I} - \boldsymbol{D})\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{D}). \end{aligned} \quad (13)$$

It is easy to show that the following proposition holds.

Proposition 2 The centralized Gram matrix $\tilde{\boldsymbol{K}}$ is semi-positive definite if and only if the Gram matrix $\boldsymbol{K}$ is semi-positive definite.

For a given kernel function k, its corresponding Gram matrix $\boldsymbol{K}$ is semi-positive definite; thus, $\tilde{\boldsymbol{K}}$ is semi-positive definite from Proposition 2. Calculate the orthonormal eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ of $\tilde{\boldsymbol{K}}$ corresponding to the m largest positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. The orthonormal eigenvectors $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m$ of $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ corresponding to the m largest positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ are

$$\boldsymbol{\beta}_j = \frac{1}{\sqrt{\lambda_j}} \boldsymbol{Q}\boldsymbol{v}_j, \quad j = 1, 2, \ldots, m, \quad (14)$$

where the coefficients in the vector $\frac{1}{\sqrt{\lambda_j}}\boldsymbol{v}_j$ correspond to a set of expansion coefficients in Eq. (12).

After the projection of the mapped sample $\Phi(\boldsymbol{x})$ onto the eigenvector system $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m$, we can obtain KPCA-transformed feature vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$ by

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m)^{\mathrm{T}} \left[\Phi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Phi}\right] \\ &= \Lambda^{-\frac{1}{2}} \boldsymbol{V}^{\mathrm{T}} \boldsymbol{Q}^{\mathrm{T}} \left[\Phi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Phi}\right] \\ &= \Lambda^{-\frac{1}{2}} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{D}) (\boldsymbol{K}_x - \boldsymbol{K}\boldsymbol{D}_1), \end{aligned} \quad (15)$$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$, $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$, $\boldsymbol{K}_x = [k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_M, \boldsymbol{x})]^{\mathrm{T}}$, and $\boldsymbol{D}_1 = (1/M)_{M \times 1}$.

3.2 GKPCA

KPCA method has the limitation that it is only suitable for kernel. If the function k is not a kernel, it may give rise to a negative definite Gram matrix $\boldsymbol{K}$ and thus the

1) This kind of map is called function replacement operator in Refs. [24,25], or called empirical kernel map in Ref. [11] provided that the given function is a kernel.

centralized Gram matrix $\tilde{\boldsymbol{K}}$ might have negative eigenvalues. These negative eigenvalues make it difficult to determine what the principal components are and difficult to normalize the eigenvectors in the feature space using Eq. (14). To overcome the limitation of KPCA, we will develop a "function" PCA method using the (function) generating kernel idea discussed in Sect. 2.2. This method is thus called GKPCA, which is theoretically suitable for any function, no matter whether k is a kernel or not.

Given a function g and a set of M training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, we can generate a kernel $k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}) g(\boldsymbol{x}_l, \boldsymbol{z})$. Let the Gram matrix of the function g with respect to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ be $\boldsymbol{G}$. Then, the Gram matrix of the generating kernel k is $\boldsymbol{K} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$ and the centralized Gram matrix is

$$\begin{aligned} \tilde{\boldsymbol{K}} &= (\boldsymbol{I} - \boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D}) \\ &= \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right]^{\mathrm{T}} \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right]. \end{aligned} \tag{16}$$

Note that, here, $\tilde{\boldsymbol{K}}$ is always positive semi-definite, no matter whether g is a kernel or not. Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ be the orthonormal eigenvectors of $\tilde{\boldsymbol{K}}$ corresponding to the m largest positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. The orthonormal eigenvectors $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m$ of $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ corresponding to the m largest positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ are thus

$$\boldsymbol{\beta}_j = \frac{1}{\sqrt{\lambda_j}} \boldsymbol{Q}\, \boldsymbol{v}_j, \quad j = 1, 2, \ldots m, \tag{17}$$

where $\boldsymbol{Q} = \left[\Phi(\boldsymbol{x}_1) - \boldsymbol{m}_0^{\Phi}, \Phi(\boldsymbol{x}_2) - \boldsymbol{m}_0^{\Phi}, \ldots, \Phi(\boldsymbol{x}_M) - \boldsymbol{m}_0^{\Phi}\right]$, and Φ is the map determined by the generating kernel k.

For a given sample $\boldsymbol{x}$, we can obtain its GKPCA-transformed feature vector $\boldsymbol{y}$ by

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m)^{\mathrm{T}} \left[\Phi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Phi}\right] \\ &= \Lambda^{-\frac{1}{2}} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{D}) \left(\boldsymbol{K}_x - \boldsymbol{K}\boldsymbol{D}_1\right), \end{aligned} \tag{18}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$, $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$, $\boldsymbol{K} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$, $\boldsymbol{D}_1 = (1/M)_{M\times 1}$ and

$$\begin{aligned} \boldsymbol{K}_x &= \left[k(\boldsymbol{x}_1, \boldsymbol{x}),\ k(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}} \\ &= \left[\sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}_1)\, g(\boldsymbol{x}_l, \boldsymbol{x}),\ \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}_2)\, g(\boldsymbol{x}_l, \boldsymbol{x}), \ldots, \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}_M)\, g(\boldsymbol{x}_l, \boldsymbol{x})\right]^{\mathrm{T}} \\ &= \boldsymbol{G}^{\mathrm{T}} \left[g(\boldsymbol{x}_1, \boldsymbol{x}),\ g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}. \end{aligned} \tag{19}$$

Equation (18) is therefore rewritten as

$$\boldsymbol{y} = \Lambda^{-\frac{1}{2}} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{D}) \boldsymbol{G}^{\mathrm{T}} \left(\boldsymbol{G}_x - \boldsymbol{G}\boldsymbol{D}_1\right), \tag{20}$$

where $\boldsymbol{G}_x = \left[g(\boldsymbol{x}_1, \boldsymbol{x}),\ g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}$.

GKPCA algorithm is summarized as follows:

Algorithm 1 (GKPCA algorithm)

Step 1 Based on the given function g and a set of M training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, construct the Gram matrix $\boldsymbol{G}$ of the function g.

Step 2 Let $\tilde{\boldsymbol{K}} = \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right]^{\mathrm{T}} \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right]$, where $\boldsymbol{D} = (1/M)_{M\times M}$. Calculate the orthonormal eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ of $\tilde{\boldsymbol{K}}$ corresponding to the m largest positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. Let $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$.

Step 3 Calculate $\boldsymbol{G}_x = \left[g(\boldsymbol{x}_1, \boldsymbol{x}),\ g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}$ for a given sample $\boldsymbol{x}$. Perform the GKPCA transformation of $\boldsymbol{x}$ by $\boldsymbol{y} = \Lambda^{-\frac{1}{2}} \boldsymbol{V}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{D}) \boldsymbol{G}^{\mathrm{T}} \left(\boldsymbol{G}_x - \boldsymbol{G}\boldsymbol{D}_1\right)$, where $\boldsymbol{D}_1 = (1/M)_{M\times 1}$.

3.3 Further understanding of GKPCA: an equivalent implementation

To gain more insight into the GKPCA algorithm, we will introduce an FMS-PCA method and show that GKPCA is equivalent to FMS-PCA. From this equivalence, we know that FMS-PCA is actually a kernel-based feature extraction method.

3.3.1 FMS-PCA

For a given function g and a set of training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, let us construct the following function map [24,25]

$$\Psi:\ \boldsymbol{x} \mapsto \left[g(\boldsymbol{x}_1, \boldsymbol{x}),\ g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}. \tag{21}$$

This map maps the training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ into $\Psi(\boldsymbol{x}_1), \Psi(\boldsymbol{x}_2), \ldots, \Psi(\boldsymbol{x}_M)$, which are actually the column vectors of the Gram matrix $\boldsymbol{G}$ corresponding to function g with respect to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, i.e.,

$$\boldsymbol{G} = \left[\Psi(\boldsymbol{x}_1), \Psi(\boldsymbol{x}_2), \ldots, \Psi(\boldsymbol{x}_M)\right]. \tag{22}$$

Let us perform PCA based on the mapped training samples $\Psi(\boldsymbol{x}_1), \Psi(\boldsymbol{x}_2), \ldots, \Psi(\boldsymbol{x}_M)$. The covariance matrix is constructed as follows

$$\begin{aligned} \boldsymbol{S}_{\mathrm{t}}^{\Psi} &= \sum_{j=1}^{M} \left(\Psi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Psi}\right) \left(\Psi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Psi}\right)^{\mathrm{T}} \\ &= \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right] \left[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\right]^{\mathrm{T}}, \end{aligned} \tag{23}$$

where $\boldsymbol{m}_0^{\Psi} = \frac{1}{M}\sum_{j=1}^{M}\Psi(\boldsymbol{x}_j) = \boldsymbol{G}\boldsymbol{D}_1$. It is obvious that $\boldsymbol{S}_{\mathrm{t}}^{\Psi}$ is an $M \times M$ positive semi-definite matrix.

Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ be the orthonormal eigenvectors of $\boldsymbol{S}_{\mathrm{t}}^{\Psi}$ corresponding to the m largest positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. For a given sample $\boldsymbol{x}$, we can obtain its FMS-PCA transformed feature vector $\boldsymbol{y}$ by

$$\begin{aligned}\boldsymbol{y} &= (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)^{\mathrm{T}}\left[\Psi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Psi}\right] \\ &= \boldsymbol{U}^{\mathrm{T}}\left(\Psi(\boldsymbol{x}) - \boldsymbol{G}\boldsymbol{D}_1\right),\end{aligned} \tag{24}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m]$ and $\Psi(\boldsymbol{x}) = [g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})]^{\mathrm{T}}$.

3.3.2 Equivalence of GKPCA and FMS-PCA

To prove the Equivalence of GKPCA and FMS-PCA, let us first introduce a corollary that is directly derived from the singular value decomposition (SVD) theorem [32] as follows:

Corollary 1 For an arbitrary matrix $\boldsymbol{A}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, and $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ have the same positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$, and the eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ corresponding to positive eigenvalues and the eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ satisfy

$$\boldsymbol{u}_j = \frac{1}{\sqrt{\lambda_j}}\boldsymbol{A}\boldsymbol{v}_j. \tag{25}$$

Let $\boldsymbol{A} = \boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})$. From Corollary 1, we know that $\boldsymbol{S}_{\mathrm{t}}^{\Psi} = [\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})]\,[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})]^{\mathrm{T}}$ in Eq. (23) and $\tilde{\boldsymbol{K}} = [\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})]^{\mathrm{T}}\,[\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})]$ in Eq. (16) have the same positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$, and their eigenvectors satisfy

$$\boldsymbol{u}_j = \frac{1}{\sqrt{\lambda_j}}\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\boldsymbol{v}_j, \quad j = 1, 2, \ldots, m. \tag{26}$$

Equation (26) can be rewritten in the following matrix

$$\boldsymbol{U} = \boldsymbol{G}(\boldsymbol{I} - \boldsymbol{D})\boldsymbol{V}\Lambda^{-\frac{1}{2}}. \tag{27}$$

Therefore, FMS-PCA transformation in Eq. (24) can be expressed by

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{U}^{\mathrm{T}}\left(\Psi(\boldsymbol{x}) - \boldsymbol{G}\boldsymbol{D}_1\right) \\ &= \Lambda^{-\frac{1}{2}}\boldsymbol{V}^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}\left(\Psi(\boldsymbol{x}) - \boldsymbol{G}\boldsymbol{D}_1\right).\end{aligned} \tag{28}$$

Since $\Psi(\boldsymbol{x}) = \boldsymbol{G}_x = [g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})]^{\mathrm{T}}$, we can see that FMS-PCA transformation in Eq. (24) is the same as GKPCA transformation in Eq. (20).

So far, we have proven that GKPCA is equivalent to FMS-PCA by rigorous derivation. Actually, we can understand this equivalence from the kernel definition point of view. Since the generating kernel in Eq. (5) is determined by the function map in Eq. (21), we can conclude that FMS-PCA is the method that directly works in a function map reduced feature space, while GKPCA is the corresponding method that works in the input space by virtue of a kernel determined by the function map. These two methods are essentially the same. Since FMS-PCA seems more straightforward and simple, we can implement GKPCA algorithm using the procedure of FMS-PCA method.

4 Observing KFD from a generating kernel point of view

This section first outlines the basic KFD model and summarizes two categories of KFD algorithms: KPCA + LDA based algorithms and SKFD algorithms. We then present GKFD method and FMS-LDA method. Finally, we reveal the equivalence relationships of the three methods SKFD, GKFD and FMS-LDA. From this equivalence, we know that SKFD algorithms are essentially generating kernel-based methods. Further, we divide the kernel-related feature extraction methods into two categories: original kernel-based methods and generating kernel-based methods.

4.1 An overview of KFD

4.1.1 KFD model

Suppose there are L known pattern classes. The between-class scatter operator $\boldsymbol{S}_{\mathrm{b}}^{\Phi}$ and the total scatter operator $\boldsymbol{S}_{\mathrm{t}}^{\Phi}$ in the feature space Φ are defined below:

$$\begin{aligned}\boldsymbol{S}_{\mathrm{b}}^{\Phi} &= \sum_{i=1}^{L} M_i\left(\boldsymbol{m}_i^{\Phi} - \boldsymbol{m}_0^{\Phi}\right)\left(\boldsymbol{m}_i^{\Phi} - \boldsymbol{m}_0^{\Phi}\right)^{\mathrm{T}} \\ &= \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^{\mathrm{T}} \\ &= \boldsymbol{X}\left(\boldsymbol{I} - \boldsymbol{D}\right)\boldsymbol{B}\left(\boldsymbol{I} - \boldsymbol{D}\right)\boldsymbol{X}^{\mathrm{T}},\end{aligned} \tag{29}$$

$$\begin{aligned}\boldsymbol{S}_{\mathrm{t}}^{\Phi} &= \sum_{j=1}^{M}\left(\Phi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Phi}\right)\left(\Phi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Phi}\right)^{\mathrm{T}} \\ &= \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} \\ &= \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{D})(\boldsymbol{I} - \boldsymbol{D})\boldsymbol{X}^{\mathrm{T}},\end{aligned} \tag{30}$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ is a set of M training samples in the input space, M_i is the number of training samples of class i and satisfies $\sum_{i=1}^{c} M_i = M$, $\boldsymbol{m}_i^{\Phi}$ is the mean vector of the mapped training samples of class i, $\boldsymbol{m}_0^{\Phi}$ is the mean vector across all mapped training samples, $\boldsymbol{X} = [\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2), \cdots, \Phi(\boldsymbol{x}_M)]$, $\boldsymbol{Q} = \left[\Phi(\boldsymbol{x}_1) - \boldsymbol{m}_0^{\Phi}, \Phi(\boldsymbol{x}_2) - \boldsymbol{m}_0^{\Phi}, \ldots, \Phi(\boldsymbol{x}_M) - \boldsymbol{m}_0^{\Phi}\right] =$

$\boldsymbol{X}(\boldsymbol{I}-\boldsymbol{D})$, and the matrix $\boldsymbol{B}$ is defined as [6]

$$\begin{aligned}&\boldsymbol{B}=\mathrm{diag}\left(\boldsymbol{B}_1,\boldsymbol{B}_2,\ldots,\boldsymbol{B}_L\right),\\&\boldsymbol{B}_i=\left(1/M_i\right)_{M_i\times M_i},\ \ i=1,2,\ldots,L.\end{aligned}\tag{31}$$

KFD seeks a set of optimal discriminant vectors by maximizing the following Fisher criterion in the feature space:

$$J^{\Phi}(\boldsymbol{\varphi})=\frac{\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{\Phi}\boldsymbol{\varphi}}{\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{\Phi}\boldsymbol{\varphi}},\ \ \boldsymbol{\varphi}\neq\boldsymbol{0}.\tag{32}$$

The optimal discriminant vectors with respect to the Fisher criterion are actually the eigenvectors of the generalized equation $\boldsymbol{S}_{\mathrm{b}}^{\Phi}\boldsymbol{\varphi}=\lambda\boldsymbol{S}_{\mathrm{t}}^{\Phi}\boldsymbol{\varphi}$. Since each eigenvector can be expressed by a linear combination of the observations in the feature space, we have

$$\boldsymbol{\varphi}=\sum_{i=1}^{M}a_i\Phi\left(\boldsymbol{x}_i\right)=\boldsymbol{X}\boldsymbol{\alpha},\quad \boldsymbol{\alpha}=\left(a_1,a_2,\ldots,a_M\right)^{\mathrm{T}}.\tag{33}$$

Substituting Eq. (33) into Eq. (32), the Fisher criterion becomes

$$\begin{aligned}\boldsymbol{J}(\boldsymbol{\alpha})&=\frac{\boldsymbol{\alpha}^{\mathrm{T}}\left[\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{B}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{K}\right]\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\mathrm{T}}\left[\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{K}\right]\boldsymbol{\alpha}}\\&=\frac{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{\alpha}},\end{aligned}\tag{34}$$

where $\boldsymbol{K}$ is the Gram matrix of kernel function k with respect to $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_M$, $\boldsymbol{S}_{\mathrm{b}}^{K}=\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{B}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{K}$, and $\boldsymbol{S}_{\mathrm{t}}^{K}=\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{K}$. The remaining problem is how to find a solution $\boldsymbol{\alpha}$ to maximize the criterion in Eq. (34).

4.1.2 KFD solutions

We always encounter the ill-posed problem when trying to maximize $J(\boldsymbol{\alpha})$ in Eq. (34) because we use the M training samples to evaluate an $M\times M$ matrix $\boldsymbol{S}_{\mathrm{t}}^{K}=\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{K}$. This matrix is always singular since its rank is at most M–1. To deal with this problem, two regularization techniques are widely used. In the regularization technique I, a scalar matrix $\varepsilon\boldsymbol{I}$ is added to the original matrix $\boldsymbol{S}_{\mathrm{t}}^{K}$ such that $\boldsymbol{S}_{\mathrm{t}}^{K}+\varepsilon\boldsymbol{I}$ becomes nonsingular, and then $\varepsilon\boldsymbol{I}$ is used to replace $\boldsymbol{S}_{\mathrm{t}}^{K}$ in Eq. (34) for the maximum value computation. The regularization technique II first employs the eigenvalue decomposition technique to remove zero or small eigenvalues of $\boldsymbol{S}_{\mathrm{t}}^{K}$ and then maximizes the resulting criterion in the principal eigenvector spanned space. This kind of regularization technique is exploited in the famous Fisherfaces method [33–36]. For convenience, we will uniformly use the regularization technique II to address the singularity problem in this paper.

No matter what regularization technique is used, the current KFD algorithms can be divided into two categories if the null space information [17] is not taken into account. In the first category of KFD algorithms, we need to perform QR decomposition of the Gram matrix $\boldsymbol{K}$ in matrices $\boldsymbol{S}_{\mathrm{b}}^{K}$ and $\boldsymbol{S}_{\mathrm{t}}^{K}$ as $\boldsymbol{K}=\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}}$ and then based on this decomposition to develop the KFD algorithm (GDA) [6]. It has been proven that GDA algorithm can be equivalently implemented in two steps: KPCA + LDA [16,17]. This category of KFD algorithms is thus called KPCA + LDA framework in this paper. Obviously, the implementation of this kind of algorithms depends on the positive semi-definiteness of the Gram matrix $\boldsymbol{K}$, since $\boldsymbol{K}$ cannot be decomposed as $\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}}$ if it is not positive semi-definite, and KPCA also requires a positive semi-definite Gram matrix. As a result, if the given function is not kernel, the KPCA + LDA based algorithms cannot be directly used.

In the second category of KFD algorithms, we would rather solve the optimization problem directly based on $\boldsymbol{S}_{\mathrm{b}}^{K}$ and $\boldsymbol{S}_{\mathrm{t}}^{K}$ than decompose the involved $\boldsymbol{K}$ in advance [4,19,20]. This category of algorithms is thus called SKFD in this paper. It is easy to show that $\boldsymbol{S}_{\mathrm{b}}^{K}$ and $\boldsymbol{S}_{\mathrm{t}}^{K}$ are always positive semi-definite, as long as $\boldsymbol{K}$ is a symmetric matrix (not necessarily positive semi-definite). It seems that SKFD algorithms are independent of the positive semi-definiteness of the Gram matrix $\boldsymbol{K}$. In this section, we will provide theoretical justifications for why SKFD algorithms can be applied to non-kernel functions. To this end, let us present a typical implementation of SKFD by virtue of the regularization technique II as follows. For convenience of further discussion, we use a general symmetric function, which can be a kernel or not, in the following formulation.

4.1.3 Implementation of SKFD

Let g be a symmetric function and $\boldsymbol{G}$ be the associated Gram matrix. This means that $\boldsymbol{G}=\boldsymbol{G}^{\mathrm{T}}$. To maximize the criterion in Eq. (34) using the regularization technique II, we first calculate the orthonormal eigenvectors $\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots,\boldsymbol{u}_m$ of $\boldsymbol{S}_{\mathrm{t}}^{K}=\boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}$ corresponding to the m largest eigenvalues $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_m$, where m is chosen as a proper value smaller than the real rank of $\boldsymbol{S}_{\mathrm{t}}^{K}$. Then, we try to find the optimal solution $\boldsymbol{\alpha}$ from the space spanned by $\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots,\boldsymbol{u}_m$. Thus, $\boldsymbol{\alpha}$ can be expressed as

$$\boldsymbol{\alpha}=\boldsymbol{U}\boldsymbol{\eta},\tag{35}$$

where $\boldsymbol{U}=[\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots,\boldsymbol{u}_m]$, $\boldsymbol{\eta}\in\mathbb{R}^m$.

Plugging Eq. (35) into Eq. (34), we have

$$\begin{aligned}J(\boldsymbol{\alpha})&=\frac{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{\alpha}}\\&=\frac{\boldsymbol{\eta}^{\mathrm{T}}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{U}\right)\boldsymbol{\eta}}{\boldsymbol{\eta}^{\mathrm{T}}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{U}\right)\boldsymbol{\eta}}\\&\triangleq J(\eta).\end{aligned}\tag{36}$$

It is easy to show that $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{U}$ and $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{U}$ are both $m \times m$ positive semi-definite matrices (note that $\boldsymbol{S}_{\mathrm{b}}^{K} = \boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{B}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}$ is positive semi-definite given a symmetric function g). Therefore, $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d$ is a generalized Rayleigh quotient [32] in $\mathbb{R}^m$. Its stationary points $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d$ are generalized eigenvectors of $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{U}$ and $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{U}$ corresponding to d largest eigenvalues. As a result, we obtain a set of optimal solutions, $\boldsymbol{\alpha}_1 = \boldsymbol{U}\boldsymbol{\eta}_1, \boldsymbol{\alpha}_2 = \boldsymbol{U}\boldsymbol{\eta}_2, \ldots, \boldsymbol{\alpha}_d = \boldsymbol{U}\boldsymbol{\eta}_d$, which maximize the criterion in Eq. (34). The optimal discriminant vectors with respect to the Fisher criterion in Eq. (32) are thereby

$$\boldsymbol{\varphi}_j = \boldsymbol{X}\boldsymbol{\alpha}_j = \boldsymbol{X}\boldsymbol{U}\boldsymbol{\eta}_j, \quad j = 1, 2, \ldots, d. \tag{37}$$

The SKFD transformation of a given sample $\boldsymbol{x}$ is

$$\begin{aligned}\boldsymbol{y} &= (\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_d)^{\mathrm{T}} \left[\Phi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Phi}\right] \\ &= (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \ldots, \boldsymbol{\eta}_d)^{\mathrm{T}} \boldsymbol{U}^{\mathrm{T}} \left[\boldsymbol{G}_x - \boldsymbol{G}\boldsymbol{D}_1\right],\end{aligned} \tag{38}$$

where $\boldsymbol{G}_x = \left[g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}$.

4.2 GKFD

In this section, we will use the KPCA + LDA [16,17] framework to develop GKFD.

As discussed before, for a given function g and a set of M training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, we can generate a kernel $k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{M} g(\boldsymbol{x}_l, \boldsymbol{x}) g(\boldsymbol{x}_l, \boldsymbol{z})$. Applying this generating kernel to the KPCA + LDA framework, we can build a two-step GKFD algorithm: GKPCA + LDA. Specifically, for a given sample $\boldsymbol{x}$, in the first step, we perform the GKPCA transformation:

$$\boldsymbol{y} = \Lambda^{-\frac{1}{2}}\boldsymbol{V}^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}\left(\boldsymbol{G}_x - \boldsymbol{G}\boldsymbol{D}_1\right), \tag{39}$$

where $\boldsymbol{G}_x = \left[g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}$.

In the second step, we perform LDA in the GKPCA-transformed space. Let $\boldsymbol{S}_{\mathrm{b}}$ and $\boldsymbol{S}_{\mathrm{w}}$ be the between-class and within-class scatter matrices in the GKPCA-transformed space. Calculate the generalized eigenvectors $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d$ of $\boldsymbol{S}_{\mathrm{b}}$ and $\boldsymbol{S}_{\mathrm{w}}$ corresponding to the d largest eigenvalues. The LDA transformation is

$$\boldsymbol{z} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d)^{\mathrm{T}}\boldsymbol{y}. \tag{40}$$

Combining Eq. (39) and Eq. (40), we obtain GKFD transformation

$$\boldsymbol{z} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d)^{\mathrm{T}}\Lambda^{-\frac{1}{2}}\boldsymbol{V}^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}\left(\boldsymbol{G}_x - \boldsymbol{G}\boldsymbol{D}_1\right). \tag{41}$$

4.3 FMS-LDA

Similar to the derivation of FMS-PCA, we can develop FMS-LDA method. After the function mapping given in Eq. (21), the training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ are mapped into $\Psi(\boldsymbol{x}_1), \Psi(\boldsymbol{x}_2), \ldots, \Psi(\boldsymbol{x}_M)$, which are actually the column vectors of the Gram matrix $\boldsymbol{G}$ corresponding to function g with respect to $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$.

Let us perform LDA based on the mapped training samples $\Psi(\boldsymbol{x}_1), \Psi(\boldsymbol{x}_2), \ldots, \Psi(\boldsymbol{x}_M)$. The between-class scatter and total scatter matrices are constructed below

$$\begin{aligned}\boldsymbol{S}_{\mathrm{b}}^{\Psi} &= \sum_{i=1}^{L} M_i \left(\boldsymbol{m}_i^{\Psi} - \boldsymbol{m}_0^{\Psi}\right)\left(\boldsymbol{m}_i^{\Psi} - \boldsymbol{m}_0^{\Psi}\right)^{\mathrm{T}} \\ &= \boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{B}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}},\end{aligned} \tag{42}$$

$$\begin{aligned}\boldsymbol{S}_{\mathrm{t}}^{\Psi} &= \sum_{j=1}^{M} \left(\Psi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Psi}\right)\left(\Psi(\boldsymbol{x}_j) - \boldsymbol{m}_0^{\Psi}\right)^{\mathrm{T}} \\ &= \boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}},\end{aligned} \tag{43}$$

where $\boldsymbol{m}_i^{\Psi}$ is the mean vector of the mapped training samples of class i. It is obvious that $\boldsymbol{S}_{\mathrm{b}}^{\Psi}$ and $\boldsymbol{S}_{\mathrm{t}}^{\Psi}$ are both $M \times M$ positive semi-definite matrices.

The Fisher criterion in the function-mapping-space is

$$J^{\Psi}(\boldsymbol{\varphi}) = \frac{\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{\Psi}\boldsymbol{\varphi}}{\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{\Psi}\boldsymbol{\varphi}}\boldsymbol{\varphi} \neq \boldsymbol{0}. \tag{44}$$

Since $\boldsymbol{S}_{\mathrm{t}}^{\psi}$ is always singular, we can apply the two-step PCA + LDA strategy [33–36] to maximize the above criterion. Based on the matrix $\boldsymbol{S}_{\mathrm{t}}^{\Psi}$, to perform PCA is actually FMS-PCA method presented in Sect. 3.3. Therefore, the two-step FMS-LDA can be implemented as follows:

In the first step, perform FMS-PCA transformation for a given sample $\boldsymbol{x}$:

$$\begin{aligned}\boldsymbol{y} &= (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)^{\mathrm{T}} \left[\Psi(\boldsymbol{x}) - \boldsymbol{m}_0^{\Psi}\right] \\ &= \boldsymbol{U}^{\mathrm{T}}\left(\Psi(\boldsymbol{x}) - \boldsymbol{G}\boldsymbol{D}_1\right),\end{aligned} \tag{45}$$

where $\Psi(\boldsymbol{x}) = \left[g(\boldsymbol{x}_1, \boldsymbol{x}), g(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, g(\boldsymbol{x}_M, \boldsymbol{x})\right]^{\mathrm{T}}$, $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m]$ and $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ are the orthonormal eigenvectors of $\boldsymbol{S}_{\mathrm{t}}^{\Psi}$ corresponding to the m largest positive eigenvalues.

In the second step, perform LDA in the FMS-PCA transformed space. Let $\boldsymbol{S}_{\mathrm{b}}$ and $\boldsymbol{S}_{\mathrm{t}}$ be the between-class and within-class scatter matrices in the FMS-PCA transformed space. It is easy to show that

$$\boldsymbol{S}_{\mathrm{b}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{\Psi}\boldsymbol{U} \quad \text{and} \quad \boldsymbol{S}_{\mathrm{t}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{\Psi}\boldsymbol{U}. \tag{46}$$

Calculate the generalized eigenvectors $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d$ of $\boldsymbol{S}_{\mathrm{b}}$ and $\boldsymbol{S}_{\mathrm{t}}$ corresponding to the d largest eigenvalues. The LDA transformation is

$$\boldsymbol{z} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d)^{\mathrm{T}}\boldsymbol{y}. \tag{47}$$

Combining Eq. (45) and Eq. (47), we obtain the FMS-LDA transformation

$$\boldsymbol{z} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \cdots, \boldsymbol{\eta}_d)^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}}\left(\Psi(\boldsymbol{x}) - \boldsymbol{G}\boldsymbol{D}_1\right). \tag{48}$$

4.4 Equivalence relationships of the foregoing three methods

Let us first consider the relationship between GKFD and FMS-LDA methods. Since GKPCA and FMS-PCA are equivalent (i.e., both methods produce the same features for a given sample), the two-step GKFD algorithm (GKPCA + LDA) is equivalent to the two-step FMS-LDA method (FMS-PCA + LDA). We can view that FMS-LDA is the method directly working in a function map reduced feature space, while GKFD is the corresponding method working in the input space by virtue of a kernel determined by the function map.

Now, let us further examine the connections between SKFD and FMS-LDA methods. Suppose that the given function g is a symmetric function. In the SKFD method, the two matrices in Eq. (34) are $\boldsymbol{S}_{\mathrm{b}}^{K}=\boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{B}(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}$ and $\boldsymbol{S}_{\mathrm{t}}^{K}=\boldsymbol{G}(\boldsymbol{I}-\boldsymbol{D})(\boldsymbol{I}-\boldsymbol{D})\boldsymbol{G}^{\mathrm{T}}$, which are the same as the between-class scatter and total scatter matrices $\boldsymbol{S}_{\mathrm{b}}^{\psi}$ and $\boldsymbol{S}_{\mathrm{t}}^{\psi}$ in the FMS-LDA method. As a result, the two matrices in Eq. (36), $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{b}}^{K}\boldsymbol{U}$ and $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{S}_{\mathrm{t}}^{K}\boldsymbol{U}$, in SKFD method are exactly the between-class and within-class scatter matrices $\boldsymbol{S}_{\mathrm{b}}$ and $\boldsymbol{S}_{\mathrm{t}}$ in the FMS-PCA transformed space in the FMS-LDA method. The two methods therefore have the same transformation matrix $\boldsymbol{U}(\boldsymbol{\eta}_1,\boldsymbol{\eta}_2,\ldots,\boldsymbol{\eta}_d)$. By comparing the SKFD transformation in Eq. (38) and the FMS-LDA transformation in Eq. (48), we can conclude that SKFD and FMS-LDA are equivalent when the given function is symmetric.

Since SKFD is equivalent to FMS-LDA as long as the given function is symmetric and FMS-LDA is equivalent to GKFD, we can derive that SKFD is equivalent to GKFD when the given function is symmetric. These equivalence relationships are illustrated in Fig. 2.

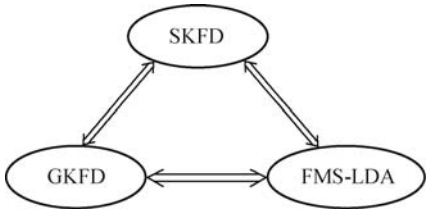

Fig. 2 Illustration of the equivalence relationships of the three methods. Note that the equivalence between SKFD and GKFD, and that between SKFD and FMS-LDA are under the condition that the given function is a symmetric function

4.5 Further discussions: two kinds of kernel-related feature extraction methods

From the equivalence of SKFD and GKFD, we can understand the SKFD algorithms from the generating kernel point of view. The SKFD algorithms are essentially generating kernel-based Fisher discriminant methods. Specifically, for a given symmetric function g, the SKFD algorithms actually use the generating kernel $k(\boldsymbol{x},\boldsymbol{z})=\sum_{l=1}^{M}g(\boldsymbol{x}_l,\boldsymbol{x})g(\boldsymbol{x}_l,\boldsymbol{z})$ rather than the original function g, no matter whether g is a kernel or not. This is the underlying reason why SKFD algorithms are suitable for non-kernel functions.

In summary, the kernel-related methods can be divided into two categories: original kernel-based methods and the generating kernel-based methods. The former category includes KPCA [3], GDA [6] and KPCA + LDA [16,17], which directly work on the given kernel and are not suitable for non-kernel functions. The later category includes GKPCA, SKFD [4,19,20] and GKFD, which work on the generating kernel from a given function and thus are suitable for non-kernels as well as kernels. These two categories of kernel methods are concluded and listed in Table 1.

5 Tensor-based feature extraction methods and their generating kernel extensions

The classical feature extraction methods, such as PCA [37] and LDA [34], are all vector (1D array) based methods. As these kinds of methods are applied to image feature extraction, image matrices need to be transformed into vectors in advance. The resulting image vectors usually lead to a high-dimensional image vector space, where it is difficult to evaluate the covariance matrix accurately due to its large size and the relatively small number of training samples. In addition, one generally encounters the rank-deficiency problem of the within-class scatter matrix that makes LDA intractable. To alleviate this difficulty, we can appeal to the matrix (2D array) based representation instead of the vector-based representation.

To the best of our knowledge, the primary idea of using matrix-based representation can be traced to K. Liu's work [38], where the matrix-based generalized Fisher criterion function was introduced. In the light of this idea, the two-dimensional PCA method (2DPCA) [39] was proposed to extend the concept of PCA. In contrast to PCA, the two-dimensional PCA method (2DPCA) is a more efficient technique for dealing with 2D images (matrices), as 2DPCA works on matrices (2D arrays) rather than on vectors (1D arrays). Therefore,

Table 1 Two categories of kernel-related methods

categories	unsupervised	supervised
original kernel-based methods	KPCA [3]	GDA [6], KPCA + LDA [16,17]
generating kernel-based methods	GKPCA, FMS-PCA	GKFD, SKFD [4,19,20], FMS-LDA

2DPCA does not transform an image into a vector, but rather, it constructs an image covariance matrix directly from the original image matrices. In contrast to the covariance matrix of PCA, the size of the image covariance matrix of 2DPCA is much smaller. For example, if the image size is 100×100, the image covariance matrix of 2DPCA is still 100×100, regardless of the training sample size. As a result, 2DPCA has a remarkable computational advantage over PCA. Recent research reveals the popularity of the 2DPCA method in pattern recognition in general and face recognition in particular [40–44]. In addition, following the idea of 2DPCA, researchers have extended the classical feature extraction methods like LDA and canonical correlation analysis (CCA), and the state-of-the-art methods like ICA [45] and local preserving projection (LPP) [46], to their two-dimensional (or matrix-based) versions, respectively [47–56]. Since a matrix can be viewed as the second-order tensor, the matrix-based representation methods have been further generalized to high-order tensor based methods [57–61].

The tensor-based feature extraction methods have been kernelized. For example, Kong et al. [43] developed a kernel version of 2DPCA based on the fact that 2DPCA can be equivalently implemented by performing PCA on rows of all training image matrices. Kernel 2DPCA (K2DPCA) is done in the following way: perform KPCA based on all rows of all training image samples by taking each row as an individual sample. Of course, we can apply the generating kernel idea to K2DPCA and obtain a generating kernel 2DPCA. This idea can be further extended to high-order tensor based feature extraction methods. However, the (generating) kernel versions of tensor methods are generally very time-consuming, because the training sample size becomes extremely large as each row of a sample matrix is viewed as an individual sample. The large training sample size leads to a large-scaled, intractable Gram matrix. To alleviate this problem, Sun et al. [62] presented a fast approximate algorithm for calculating the eigenvectors of the Gram matrix by dividing it into a block matrix with smaller sized sub-matrices. Despite this, the computational complexity of K2DPCA is still much larger than that of the standard KPCA. We know that 2DPCA (or its variants [42–44,63]) is much faster than the standard PCA. The existing kernelized version of 2DPCA loses its computational advantage as compared to KPCA. The underlying reason is that we still apply the classical vector-based kernels to kernelize the tensor (matrix)-based methods. To develop the truly meaningful kernelized tensor-based feature extraction methods, we need to build the concept of the tensor-based kernels instead of the vector-based kernels. This poses a new topic for further exploration in the future.

6 Conclusions

This paper presents an idea of generating a kernel from a given arbitrary function by embedding the training samples into the function. The generating kernel is therefore with respect to a set of training samples and takes advantage of the information of the training samples. Based on the generating kernel idea, we develop two nonlinear feature extraction methods: GKPCA and GKFD. To gain more insight into these two methods, we give their equivalent versions: FMS-PCA method and FMS-LDA method, and show that FMS-PCA (FMS-LDA) is the method that directly works in a function map reduced feature space, while GKPCA (GKFD) is the corresponding method that works in the input space by virtue of a kernel determined by the function map.

This paper divides the current KFD algorithms into two categories: KPCA + LDA based algorithms and SKFD algorithms. SKFD is shown to be equivalent to GKFD as long as the given function is symmetric. This equivalence reveals that the SKFD algorithms are essentially generating kernel-based methods. Specifically, for a given symmetric function g, SKFD algorithms actually use the generating kernel in Eq. (5) rather than the original function g, no matter whether g is a kernel or not. This is the underlying reason why the SKFD algorithms are suitable for non-kernel functions. From the generating kernel point of view, we re-examine the two categories of KFD algorithms and realize that KPCA + LDA based algorithms directly work on the given kernel and are not suitable for non-kernel functions, whereas SKFD algorithms essentially work on the generating kernel from a given function and are therefore suitable for non-kernels as well as kernels.

This paper finally outlines the tensor-based feature extraction methods which are generalizations of vector-based methods. In contrast to vector-based methods such as PCA and LDA, tensor-based methods use the form of sample as it exists for representation rather than converting it into vector. Therefore, tensor-based methods are computationally more efficient than vector-based methods. The current tensor-based methods have been kernelized, and their generating kernel versions can be further obtained. However, the existing kernelized tensor-based methods lose the computational advantages. The underlying reason is that the vector-based kernels are applied to kernelize the tensor-based methods. We need to build the concept of the tensor-based kernels based on which the truly meaningful kernelized tensor-based feature extraction methods can be developed in the future.

Acknowledgements The author would like to thank Prof. Lei Xu for his constructive comments and suggestions. This work was

partially supported by the Program for New Century Excellent Talents in University of China, the NUST Outstanding Scholar Supporting Program, and the National Natural Science Foundation of China (Grant No. 60973098).

References

1. Vapnik V. The Nature of Statistical Learning Theory. New York: Springer, 1995
2. Müller K R, Mika S, Rätsch G, Tsuda K, Schölkopf B. An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks, 2001, 12(2): 181–201
3. Schölkopf B, Smola A, Muller K R. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 1998, 10(5): 1299–1319
4. Mika S, Rätsch G, Weston J, Schölkopf B, Müller K R. Fisher discriminant analysis with kernels. In: Proceedings of IEEE International Workshop on Neural Networks for Signal Processing IX. 1999, 41–48
5. Mika S, Rätsch G, Schölkopf B, Smola A, Weston J, Müller K R. Invariant feature extraction and classification in kernel spaces. Advances in Neural Information Processing Systems, 1999, 12: 526–532
6. Baudat G, Anouar F. Generalized discriminant analysis using a kernel approach. Neural Computation, 2000, 12(10): 2385–2404
7. Roth V, Steinhage V. Nonlinear discriminant analysis using kernel functions. In: Solla S A, Leen T K, Mueller K R, eds. Advances in Neural Information Processing Systems. 2000, 12: 568–574
8. Mika S, Rätsch G, Weston J, Schölkopf B, Smola A, Müller K R. Constructing descriptive and discriminative non-linear features: Rayleigh coefficients in kernel feature spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(5): 623–628
9. Yang M H. Kernel Eigenfaces vs. kernel Fisherfaces: face recognition using kernel methods. In: Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition. 2002, 215–220
10. Lu J, Plataniotis K N, Venetsanopoulos A N. Face recognition using kernel direct discriminant analysis algorithms. IEEE Transactions on Neural Networks, 2003, 14(1): 117–126
11. Schölkopf B, Smola A. Learning with Kernels. Cambridge: MIT Press, 2002
12. Shawe-Taylor J, Cristianini N. Kernel Methods for Pattern Analysis. Cambridge: Cambridge University Press, 2004
13. Xu J, Zhang X, Li Y. Kernel MSE algorithm: a unified framework for KFD, LS-SVM, and KRR. In: Proceedings of the International Joint Conference on Neural Networks. 2001, 1486–1491
14. Billings S A, Lee K L. Nonlinear fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. Neural Networks, 2002, 15(2): 263–270
15. Zheng W, Zhao L, Zou C. Foley-Sammon optimal discriminant vectors using kernel approach. IEEE Transactions on Neural Networks, 2005, 16(1): 1–9
16. Yang J, Jin Z, Yang J Y, Zhang D, Frangi A F. Essence of kernel Fisher discriminant: KPCA plus LDA. Pattern Recognition, 2004, 37(10): 2097–2100
17. Yang J, Frangi A F, Yang J Y, Zhang D, Jin Z. KPCA plus LDA: a complete kernel Fisher discriminant framework for feature extraction and recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(2): 230–244
18. Xu Y, Zhang D, Jin Z, Li M, Yang J Y. A fast kernel-based nonlinear discriminant analysis for multi-class problems. Pattern Recognition, 2006, 39(6): 1026–1033
19. Zhao J, Wang H, Ren H, Kee S C. LBP discriminant analysis for face verification. In: Proceeding of the 2005 IEEE Conference on Computer Vision and Pattern Recognition. 2005, 3: 167
20. Liu C. Capitalize on dimensionality increasing techniques for improving face recognition grand challenge performance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(5): 725–737
21. Cevikalp H, Neamtu M, Wilkes M, Barkana A. Discriminative common vector method with kernels. IEEE Transactions on Neural Networks, 2006, 17(6): 1550–1565
22. Bach F R, Jordan M I. Kernel independent component analysis. Journal of Machine Learning Research, 2002, 3(1): 1–48
23. Yang J, Gao X, Zhang D, Yang J Y. Kernel ICA: an alternative formulation and its application to face recognition. Pattern Recognition, 2005, 38(10): 1784–1787
24. Ma J. Function replacement vs. kernel trick. Neurocomputing, 2003, 50: 479–483
25. Ma J, Theiler J, Perkins S. Two realizations of a general feature extraction framework. Pattern Recognition, 2004, 37(5): 875–887
26. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C. Text classification using string kernels. Journal of Machine Learning Research, 2002, 2(2): 419–444
27. Chen W S, Yuen P C, Huang J, Lai J. Wavelet kernel construction for kernel discriminant analysis on face recognition. In: Proceeding of the 2006 Conference on Computer Vision and Pattern Recognition Workshop. 2006, 47–52
28. Schölkopf B. Support vector learning. Dissertation for the Doctoral Degree. Berlin: Berlin Technical University, 1997
29. Camps-Valls G, Martin-Guerrero J, Rojo-Alvarez J, Soria-Olivas E. Fuzzy sigmoid kernel for support vector classifiers. Neurocomputing, 2004, 62: 501–506
30. Ahonen T, Hadid A, Pietikäinen M. Face description with local binary patterns: application to face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(12): 2037–2041
31. Mangasarian L. Generalized support vector machines. In: Smola A J, Bartlett P, Schokopf B, Schuurmans D, eds. Advances in Large Margin Classifiers. 2000, 135–146
32. Golub G H, Van Loan C F. Matrix Computations. 3rd ed. Baltimore: The Johns Hopkins University Press, 1996
33. Swets D L, Weng J. Using discriminant eigenfeatures for image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996, 18(8): 831–836
34. Belhumeur P N, Hespanha J P, Kriengman D J. Eigenfaces vs. Fisherfaces: Recognition using class specific linear pro-

jection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(7): 711–720
35. Yang J, Yang J Y. Why can LDA be performed in PCA transformed space? Pattern Recognition, 2003, 36(2): 563–566
36. Liu C J, Wechsler H. Robust coding schemes for indexing and retrieval from large face databases. IEEE Transactions on Image Processing, 2000, 9(1): 132–137
37. Turk M, Pentland A. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 1991, 3(1): 71–86
38. Liu K, Cheng Y Q, Yang J Y. Algebraic feature extraction for image recognition based on an optimal discriminant criterion. Pattern Recognition, 1993, 26(6): 903–911
39. Yang J, Zhang D, Frangi A F, Yang J Y. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(1): 131–137
40. Visani M, Garcia C, Laurent C. Comparing robustness of two-dimensional PCA and eigenfaces for face recognition. Lecture Notes in Computer Science, 2004, 3212: 717–724
41. Zuo W M, Zhang D. K. Wang K. An assembled matrix distance metric for 2DPCA-based image recognition. Pattern Recognition Letters, 2006, 27(3): 210–216
42. Zhang D Q, Zhou Z H. (2D)(2)PCA: Two-directional two-dimensional PCA for efficient face representation and recognition. Neurocomputing, 2005, 69(1-3): 224–231
43. Kong H, Wang L, Teoh E K, Li X, Wang J G, Venkateswarlu R. Generalized 2D principal component analysis for face image representation and recognition. Neural Networks, 2005, 18(5-6): 585–594
44. Ye J. Generalized low rank approximations of matrices. Machine Learning, 2005, 61(1-3): 167–191
45. Hyvärinen A, Oja E. Independent component analysis: algorithms and applications. Neural Networks, 2000, 13(4-5): 411–430
46. He X, Yan S, Hu Y, Niyogi P, Zhang H J. Face recognition using Laplacianfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(3): 328–340
47. Ye J, Janardan R, Li Q. Two-dimensional linear discriminant analysis. Advances in Neural Information Processing Systems, 2004, 17: 1569–1576
48. Li M, Yuan B. 2D-LDA: a novel statistical linear discriminant analysis for image matrix. Pattern Recognition Letters, 2005, 26(5): 527–532
49. Xiong H, Swamy M, Ahmad M. Two-dimensional FLD for face recognition. Pattern Recognition, 2005, 38(7): 1121–1124
50. Yang J, Zhang D, Yong X, Yang J. Two-dimensional linear discriminant transform for face recognition. Pattern Recognition, 2005, 38(7): 1125–1129
51. Zuo W, Zhang D, Yang J, Wang K. BDPCA plus LDA: A novel fast feature extraction technique for face recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2006, 36(4): 946–953
52. Zou C, Sun N, Ji Z, Zhao L. 2DCCA: a novel method for small sample size face recognition. In: Proceedings of IEEE Workshop on Applications of Computer Vision. 2007, 43
53. Lee S H, Choi S. Two-dimensional canonical correlation analysis. IEEE Signal Processing Letters, 2007, 14(10): 735–738
54. Gao Q, Zhang L, Zhang D, Xu H. Independent components extraction from image matrix. Pattern Recognition Letters, 2010, 31(3): 171–178
55. Chen S B, Zhao H F, Kong M, Luo B. 2d-lpp: a two-dimensional extension of locality preserving projections. Neurocomputing, 2007, 70(4-6): 912–921
56. Hu D W, Feng G Y, Zhou Z T. Two-dimensional locality preserving projections (2DLPP) with its application to palmprint recognition. Pattern Recognition, 2007, 40(1): 339–342
57. Xu D, Yan S, Zhang L, Lin S, Zhang H J, Huang T S. Reconstruction and recognition of tensor-based objects with concurrent subspaces analysis. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(1): 36–47
58. Wang H, Ahuja N. A tensor approximation approach to dimensionality reduction. Journal of Computer Vision, 2008, 76(3): 217–229
59. Tao D, Li X, Wu X, Maybank S J. General tensor discriminant analysis and gabor features for gait recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(10): 1700–1715
60. Lu H, Plataniotis K N, Venetsanopoulos A N. MPCA: multilinear principal component analysis of tensor objects. IEEE Transactions on Neural Networks, 2008, 19(1): 18–39
61. Zhang L, Gao Q, Zhang D. Directional independent component analysis with tensor representation. In: Proceedings of Computer Vision and Pattern Recognition. 2008, 1–7
62. Sun N, Wang H, Ji Z, Zou C, Zhao L. An efficient algorithm for kernel two-dimensional principal component analysis. Neural Computing & Applications, 2008, 17(1): 59–64
63. Liu J, Chen S, Zhou Z H, Tan X. Generalized low-rank approximations of matrices revisited. IEEE Transactions on Neural Networks, 2010, 21(4): 621–632

Jian YANG received the BS degree in mathematics from the Xuzhou Normal University in 1995. He received the MS degree in applied mathematics from the Changsha Railway University in 1998 and the PhD degree from the Nanjing University of Science and Technology (NUST), on the subject of pattern recognition and intelligence systems in 2002. In 2003, he was a postdoctoral researcher at the University of Zaragoza, and in the same year, he was awarded the RyC program Research Fellowship sponsored by the Spanish Ministry of Science and Technology. From 2004 to 2006, he was a Postdoctoral Fellow at Biometrics Centre of Hong Kong Polytechnic University. From 2006 to 2007, he was a Postdoctoral Fellow at Department of Computer Science of New Jersey Institute of Technology. Now, he is a professor in the School of Computer Science

and Technology of NUST. He is the author of more than 50 scientific papers in pattern recognition and computer vision. His journal papers have been cited more than 1000 times in the ISI Web of Science, and 2000 times in the Web of Scholar Google. His research interests include pattern recognition, computer vision and machine learning. Currently, he is an associate editor of Pattern Recognition Letters and Neurocomputing, respectively.