PERGAMON

Available at www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 875–887

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

# Two realizations of a general feature extraction framework

Junshui Ma*, James Theiler, Simon Perkins

*Los Alamos National Laboratory, Los Alamos, NM 87544, USA*

Received 6 November 2002; accepted 7 October 2003

**Abstract**

A general feature extraction framework is proposed as an extension of conventional linear discriminant analysis. Two nonlinear feature extraction algorithms based on this framework are investigated. The first is a kernel function feature extraction (KFFE) algorithm. A disturbance term is introduced to regularize the algorithm. Moreover, it is revealed that some existing nonlinear feature extraction algorithms are the special cases of this KFFE algorithm. The second feature extraction algorithm, mean–STD[1]–norm feature extraction algorithm, is also derived from the framework. Experiments based on both synthetic and real data are presented to demonstrate the performance of both feature extraction algorithms.
© 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

*Keywords:* Nonlinear feature extraction; Multi-class feature extraction; Class separability; Regularization; Discriminant analysis; Kernel functions; Classification

## 1. Introduction

Feature extraction (FE) techniques are widely employed in many areas such as pattern recognition, machine learning, and data mining. They can be classified into wrapper- and filter-type FE algorithms. The filter-type FE algorithms can be further loosely classified into two categories [1]: (a) FE algorithms for generating *representative features*; and (b) FE algorithms for generating *discriminant features* or *classification-oriented features*. As well as providing a more parsimonious description of the data, the use of a small number of features also permits a faster computation of the classifier, and by eliminating irrelevant or redundant features, can also produce more robust and accurate classification.

Most of the algorithms in the first category try to extract features with a goal of maximally retaining the energy in the original patterns, and are usually derived from signal processing techniques [2,3], or based on some knowledge-based models [4,5]. Typical examples of this class of algorithms are the principal component analysis (PCA) [6,7], and its kernel version of kernel PCA [8].

In the past 10 years, increasing attention has been raised to the FE algorithms belonging to the second category. Numerous algorithms of this type have been developed, among them are the discriminant component analysis algorithm [9,10], a FE algorithm based on decision boundaries [11], and the local discriminant basis algorithm [12]. However, these examples generally explore only linear discriminants. The breakthrough on the direction of general nonlinear discriminant FE algorithms was made by Mika et al. [13] with the kernel fisher discriminant (KFD) algorithm. However, the KFD algorithm can only handle 2-class problems, because it was derived from the 2-class Fisher's discriminant analysis. Thereafter, quite a few improvements and variants of the KFD algorithm which can cope with multiple class problems came into being by following the general idea proposed in KFD [14–17]. These algorithms were all derived using the *kernel trick* [18], and thus require expensive computation when the training set is not small. Mika et al. later on proposed an alternative formulation of the original KFD algorithm, which suggested a promising direction to reduce the computation requirement of the original KFD algorithm

[1] STD: Standard Deviation.

* Corresponding author. Ohio Supercomputer Center, 1224 Kinnear Rd., Columbus, OH 43212, USA. Tel.: +1-614-688-4893; fax: +1-614-292-7168.

*E-mail address:* junshui@osc.edu (J. Ma).

0031-3203/$30.00 © 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2003.10.010

[19,20]. However, it can only deal with 2-class problems. In order to simplify our following presentation, the original KFD algorithm, as well as all these algorithms that were devised following the original KFD algorithm, is generally called KFD-like nonlinear FE algorithms in this paper.

The results presented in this paper are a set of nonlinear FE algorithms to generate classification-oriented features. The main contributions of this paper are listed as follows:

(1) A FE framework for general multi-class problems is proposed as an extension of the linear discriminant analysis (LDA). This framework provides a systematic avenue to easily devise new nonlinear FE algorithms, which is demonstrated in this paper by deriving two new FE algorithms based upon this framework.
(2) The first new FE algorithm, called the kernel function feature extraction (KFFE) algorithm, is proposed under this FE framework. The relationship between the KFFE algorithm and the existing KFD-like algorithms is discussed. We discover that many existing KFD-like algorithms are special cases of the proposed KFFE algorithm. Moreover, the KFFE algorithm suggests a new direction to reduce the computation burden bothering most of the KFD-like algorithms when dealing with large data set.
(3) The second new FE algorithm, called the mean–STD–norm feature extraction (MSNFE) algorithm, is also derived based upon the FE framework. This algorithm is experimentally demonstrated to have high performance for time-series type of data.

In order to clarify the notation and facilitate subsequent derivation, the LDA is presented in Section 2. The general FE framework is introduced in Section 3. In Sections 4 and 5 two concrete nonlinear FE algorithms are developed based on the framework. A set of experiments are implemented and presented in Section 6 both to justify some discussion in Section 4 and to demonstrate the performance of proposed two FE algorithms. The famous *No Free Lunch Theorem* [21] suggests that no FE algorithms can universally improve the class separability. Therefore, we do not only present the results that demonstrate the outstanding performance of our proposed algorithms. Instead, in this paper we select three fairly different problems, and demonstrate the different algorithmic performance for all these problems. We hope this practice can help readers figure out when they *should* and/or *should not* consider the proposed algorithms in this paper.

## 2. Linear discriminant analysis (LDA)

We first define some notations that are frequently used in this paper.

- $\mathbf{X}_l$: the $l$th $n$-feature pattern in a available pattern set,
- $\mathbf{X}_l^{(i)}$: the $l$th $n$-feature pattern among all patterns from the $i$th class,
- $\mathbf{I}_N$: a $N \times N$ identity matrix,
- $\mathbf{1}_{N_1 \times N_2}$: a $N_1 \times N_2$ matrix, whose elements are all one. When $N_1 = N_2$, it can be simplified as $\mathbf{1}_{N_1}$.

In order to simplify our notation system, in the following derivation we do not explicitly distinguish a random variable from its realization, as well as a random parameter from its estimation, in representation. Its real meaning is self-telling in the context.

The foundation of the LDA is the definition of a group of scatter-matrices, among which within-class matrix and between-class matrix are two related to our subsequent presentation.

If we denote the mean of all the patterns from class $i$, or $\omega_i$, as $\mathbf{M}_i^{\mathbf{X}}$, or $\mathbf{M}_i^{\mathbf{X}} = E\{\mathbf{X} \mid \omega_i\}$, the within-class scatter matrix, $\mathbf{S}_w^{\mathbf{X}}$, can thus be represented as [1]

$$\mathbf{S}_w^{\mathbf{X}} = \sum_{i=1}^{L} P_i E\{(\mathbf{X} - \mathbf{M}_i^{\mathbf{X}})(\mathbf{X} - \mathbf{M}_i^{\mathbf{X}})^{\mathrm{T}} \mid \omega_i\}$$

$$= \sum_{i=1}^{L} P_i \mathbf{\Sigma_i}, \tag{1}$$

where $L$ is the number of classes, $P_i$ is the prior probability of class $i$, or $\omega_i$. The within-class matrix captures the spread of patterns around their individual class means.

If we replace $P_i$ with the pattern frequency of class $i$, and replace $E\{\cdot\}$ with the pattern average, we can obtain the estimate of $\mathbf{S}_w^{\mathbf{X}}$ directly from a set of patterns

$$\mathbf{S}_w^{\mathbf{X}} = \sum_{i=1}^{L} \frac{N_i}{N} \left[ \frac{1}{N_i} \sum_{k=1}^{N_i} (\mathbf{X}_k^{(i)} - \mathbf{M}_i^{\mathbf{X}})(\mathbf{X}_k^{(i)} - \mathbf{M}_i^{\mathbf{X}})^{\mathrm{T}} \right]$$

$$= \frac{1}{N} \mathbf{\Theta_X} \mathbf{W} \mathbf{\Theta_X}^{\mathrm{T}}, \tag{2}$$

where

$$\mathbf{W} = \begin{bmatrix} \mathbf{I}_{N_1} - \frac{1}{N_1}\mathbf{1}_{N_1} & & \\ & \ddots & \\ & & \mathbf{I}_{N_L} - \frac{1}{N_L}\mathbf{1}_{N_L} \end{bmatrix},$$

$N_i$ is the number of patterns from class $i$ in the available pattern set, $N$ is the total number of patterns from all classes, and $\mathbf{\Theta_X}$ is a set of available patterns with the patterns from the same class grouped together. That is, $\mathbf{\Theta_X}$ can be represented as

$$\mathbf{\Theta}_X = [\mathbf{\Theta}_X^{(1)}\ \mathbf{\Theta}_X^{(2)}\ \cdots\ \mathbf{\Theta}_X^{(L)}] = [\mathbf{X}_1\ \mathbf{X}_2\ \cdots\ \mathbf{X}_N],$$

where $\mathbf{\Theta}_X^{(i)} = [\mathbf{X}_1^{(i)}\ \mathbf{X}_2^{(i)}\ \cdots\ \mathbf{X}_{N_i}^{(i)}]$. In this paper, we call $\mathbf{\Theta_X}$ the *training set*. Note that $(\mathbf{I}_{N_i} - \frac{1}{N_i}\mathbf{1}_{N_i}) = (\mathbf{I}_{N_i} - \frac{1}{N_i}\mathbf{1}_{N_i})(\mathbf{I}_{N_i} - \frac{1}{N_i}\mathbf{1}_{N_i})$ when deriving Eq. (2).

The between-class scatter matrix, $\mathbf{S}_b^{\mathbf{X}}$, can be represented as [1]

$$\mathbf{S}_b^{\mathbf{X}} = \sum_{i=1}^{L} P_i(\mathbf{M}_i^{\mathbf{X}} - \mathbf{M}_0^{\mathbf{X}})(\mathbf{M}_i^{\mathbf{X}} - \mathbf{M}_0^{\mathbf{X}})^{\mathrm{T}}, \tag{3}$$

where $\mathbf{M}_0^{\mathbf{X}} = E\{\mathbf{X}\} = \sum_{i=1}^{L} P_i\mathbf{M}_i^{\mathbf{X}}$ denotes the mean of the mixture distribution of all the classes. This between-class matrix captures the spread of the mean of each class around the mean of all the classes.

The estimate of $\mathbf{S}_b^{\mathbf{X}}$ can be obtained similarly to that of $\mathbf{S}_w^{\mathbf{X}}$, and can be expressed as

$$\mathbf{S}_b^{\mathbf{X}} = \sum_{i=1}^{L} \frac{N_i}{N}(\mathbf{M}_i^{\mathbf{X}} - \mathbf{M}_0^{\mathbf{X}})(\mathbf{M}_i^{\mathbf{X}} - \mathbf{M}_0^{\mathbf{X}})^{\mathrm{T}} = \frac{1}{N}\mathbf{\Theta}_{\mathbf{X}}\mathbf{B}\mathbf{\Theta}_{\mathbf{X}}^{\mathrm{T}}, \tag{4}$$

where

$$\mathbf{B} = \mathbf{G}\begin{bmatrix} \mathbf{N}_1 & & \\ & \ddots & \\ & & \mathbf{N}_L \end{bmatrix}\mathbf{G}^{\mathrm{T}} \quad \text{and}$$

$$\mathbf{G} = \begin{bmatrix} \frac{1}{N_1}\mathbf{1}_{N_1\times 1} & & \\ & \ddots & \\ & & \frac{1}{N_L}\mathbf{1}_{N_L\times 1} \end{bmatrix} - \frac{1}{N}\mathbf{1}_{N\times L}.$$

Reorganizing the definition of within-class matrix $\mathbf{S}_w^{\mathbf{X}}$ and between-class matrix $\mathbf{S}_b^{\mathbf{X}}$ in the forms of Eqs. (2) and (4), respectively, are very critical in our subsequent derivation, which will become clearer in our following presentation.

Intuitively, it is easy to reason that a bigger between-class scatter matrix and/or a smaller within-class scatter matrix imply higher class separability, which in turn suggests that the involved feature set is better. This intuition can be quantitatively represented as trying to maximize the criterion defined in

$$J = tr(\mathbf{S}_w^{-1}\mathbf{S}_b), \tag{5}$$

where $tr\{\bullet\}$ denotes the trace operation of a matrix. This separability criterion $J$ is attractive because of its simplicity and its coordinate-independent property—it is invariant under any nonsingular linear transformation [1].

As is well known, any linear FE algorithm can be represented as

$$\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{X}, \tag{6}$$

where $\mathbf{X}$ is a original $n$-feature pattern, and is thus a $1 \times n$ vector; $\mathbf{R}$ is a new pattern formed by $m$ extracted new features, and is thus a $1 \times m$ vector; $\mathbf{A}$ is the $n \times m$ matrix that characterizes the linear FE algorithm.

From the definitions of within-class matrix in Eq. (1), and between-class matrix in Eq. (3), it is straightforward to derive the relationship between the scatter matrix of the original patterns and that of the new patterns. The relationship can be represented as

$$\mathbf{S}_w^{\mathbf{R}} = \mathbf{A}^{\mathrm{T}}\mathbf{S}_w^{\mathbf{X}}\mathbf{A}, \tag{7a}$$

$$\mathbf{S}_b^{\mathbf{R}} = \mathbf{A}^{\mathrm{T}}\mathbf{S}_b^{\mathbf{X}}\mathbf{A}. \tag{7b}$$

The LDA can thus be defined as, given a separability criterion $J$, find a best $n \times m$ transform matrix $\mathbf{A}$, which can be used to generate the $m$-feature new pattern $\mathbf{R}$ from the $n$-feature original pattern $\mathbf{X}$. That is

$$\begin{aligned}\mathbf{A} &= \arg\max_{\mathbf{A}} J(\mathbf{A}) \\ &= \arg\max_{\mathbf{A}} tr\{(\mathbf{S}_w^{\mathbf{R}})^{-1}\mathbf{S}_b^{\mathbf{R}}\} \\ &= \arg\max_{\mathbf{A}} tr\{(\mathbf{A}^{\mathrm{T}}\mathbf{S}_w^{\mathbf{X}}\mathbf{A})^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{S}_b^{\mathbf{X}}\mathbf{A})\}. \end{aligned} \tag{8}$$

It can be proved that the best transform matrix $\mathbf{A}$ is formed by the $m$ eigenvectors corresponding to the $m$ largest eigenvalues of the matrix $(\mathbf{S}_w^{\mathbf{X}})^{-1}\mathbf{S}_b^{\mathbf{X}}$ [1]. That is

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_m], \tag{9}$$

where $((\mathbf{S}_w^{\mathbf{X}})^{-1}\mathbf{S}_b^{\mathbf{X}})\mathbf{a}_i = \lambda_i\mathbf{a}_i$, $i = 1, \ldots, n$, $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$, and $m \leqslant n$.

## 3. A general feature extraction framework

A simple analysis of the LDA can expose at least its two critical drawbacks. First, it can only exploit linear class separability in the underlying problem; and second, linear FE algorithms based on the criterion $J$ cannot robustly deal with the situation where the means of different classes are very close to each other.

These limitations of LDA motivate us to introduce a more general form of FE framework, which is obtained using an operation called *function replacement* [22]:

$$\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{F}(\mathbf{X}), \tag{10}$$

where $\mathbf{F}(\mathbf{X})$ is a functional vector, and can be represented as

$$\mathbf{F}(\mathbf{X}) = [F_1(\mathbf{X})\ F_2(\mathbf{X})\ \cdots\ F_k(\mathbf{X})]^{\mathrm{T}}.$$

If we define $\mathbf{S}_w^{\mathbf{F}(\mathbf{X})}$ and $\mathbf{S}_b^{\mathbf{F}(\mathbf{X})}$ as the within-class and between-class matrices, respectively, of the data in the $\mathbf{F}(\mathbf{X})$ space, then based on Eqs. (7)–(8), we know that the best transform matrix $\mathbf{A}$ in Eq. (10) is formed by the $m$ eigenvectors corresponding to the $m$ largest eigenvalues of the matrix $(\mathbf{S}_w^{\mathbf{F}(\mathbf{X})})^{-1}\mathbf{S}_b^{\mathbf{F}(\mathbf{X})}$. That is

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_m], \tag{11}$$

where $((\mathbf{S}_w^{\mathbf{F}(\mathbf{X})})^{-1}\mathbf{S}_b^{\mathbf{F}(\mathbf{X})})\mathbf{a}_i = \lambda_i\mathbf{a}_i$, $i = 1, \ldots, k$, $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_k$, and $m \leqslant k$.

If we define the element function $F_i(\mathbf{X})$ of Eq. (10) as nonlinear functions of the original pattern $\mathbf{X}$, Eq. (10) can be employed to represent almost any nonlinear FE algorithm. On the other hand, the original linear formulation (6) is a special case of Eq. (10). Therefore, Eqs. (10) and (11) jointly represent a general framework of feature extraction. In the remaining part of this paper, we mostly focus on the cases where $F_i(\mathbf{X})$ is a nonlinear function.

Generally speaking, nonlinear FE algorithms have more flexibility to explore the class separability embedded in a problem than corresponding linear algorithms, and thus potentially have higher performance. However, optimization of a nonlinear problem is usually not straightforward. Compared with Eq. (6), Eq. (10) can be obtained by simply replacing $\mathbf{X}$ in Eq. (6) with the nonlinear functional vector $\mathbf{F}(\mathbf{X})$ in Eq. (10). This function replacement operation seems trivial at first glance. However, it incorporates two immediate advantages in Eq. (10) over Eq. (6).

(1) It introduces a nonlinear relationship between $\mathbf{X}$ and $\mathbf{Y}$, while retains the theoretic tractability of Eq. (6). That is, the nonlinearity is achieved by the choice of a nonlinear functional vector $\mathbf{F}(\mathbf{X})$, while the tractability is maintained by only finding the transform matrix $\mathbf{A}$ in Eq. (10). This FE framework provides a mechanism to separate the nonlinearity of a FE algorithm from its optimization procedure.
(2) The dimension of $\mathbf{F}(\mathbf{X})$, $k$, is a variable, and can be chosen according to the underlying problem. It thus relieves us from the restriction of the dimension of input vector $\mathbf{X}$.

These two advantages will become clearer in our subsequent presentation.

Now, the only problem we are facing is how to define the functional vector $\mathbf{F}(\mathbf{X})$. From Eq. (10), we can see that the final FE functions are the weighted linear combination of those element functions in the functional vector $\mathbf{F}(\mathbf{X})$. That is, the resultant FE functions must be in a space spanned by the element functions in the selected functional vector $\mathbf{F}(\mathbf{X})$. Therefore, the choice of $\mathbf{F}(\mathbf{X})$ will critically determine the final performance of the resultant FE functions.

Intuitively, there are two directions to choose $\mathbf{F}(\mathbf{X})$. One direction is problem-dependent. That is, we can choose element function $F_i(\mathbf{X})$ based on prior knowledge of the problem. For example, we can ask an expert in that area to propose some element functions as finding the energy of $\mathbf{X}$, or the frequency where $\mathbf{X}$ reaches its spectral peak, etc. The second direction of choosing $\mathbf{F}(\mathbf{X})$ is, in contrast, problem-independent. As suggested in Ref. [23], we can define $\mathbf{F}(\mathbf{X})$ in a way that will apply to all kinds of problems. For example, we can choose the elements of $\mathbf{F}(\mathbf{X})$ as a basis of a functional space, in which the resultant FE functions will reside. The two algorithms presented in the subsequent two sections are examples in this problem-independent direction.

## 4. Kernel function feature extraction (KFFE)

Theories in the area of reproducing kernels Hilbert space and support vector machines inspired us to consider a set of kernel functions to form the functional vector $\mathbf{F}(\mathbf{X})$ [8,13,15]. That is,

$$\begin{aligned}\mathbf{F}(\mathbf{X}) &= \mathbf{K}(\bullet,\mathbf{X})\\ &= [K(\mathbf{X}_1,\mathbf{X}) \quad K(\mathbf{X}_2,\mathbf{X}) \ \cdots \ K(\mathbf{X}_k,\mathbf{X})]^{\mathrm{T}},\end{aligned} \tag{12}$$

where $\mathbf{X}_i \in \boldsymbol{\Omega}_{\mathbf{X}}$, and $\boldsymbol{\Omega}_{\mathbf{X}}$ is a set of available patterns. In this paper, we call $\boldsymbol{\Omega}_{\mathbf{X}}$ the *reference set*. Note that there is no requirement that this reference set $\boldsymbol{\Omega}_{\mathbf{X}}$ should coincide with the training set, $\boldsymbol{\Theta}_{\mathbf{X}}$, of the underlying problem, although, in order to make full use of all available patterns, we usually choose $\boldsymbol{\Omega}_{\mathbf{X}}$ to be the same as $\boldsymbol{\Theta}_{\mathbf{X}}$, or to be a subset of $\boldsymbol{\Theta}_{\mathbf{X}}$. In fact, when the size of training set $\boldsymbol{\Theta}_{\mathbf{X}}$ is very large, defining $\boldsymbol{\Omega}_{\mathbf{X}}$ as only a subset of $\boldsymbol{\Theta}_{\mathbf{X}}$ will significantly reduce the subsequent computation. From this perspective, differentiating the concept of $\boldsymbol{\Omega}_{\mathbf{X}}$ and $\boldsymbol{\Theta}_{\mathbf{X}}$ provides a different perspective, or potentially a different solution, of the study presented in Ref. [24].

Theoretically speaking, many functions can be used as the kernel function $K(\mathbf{X},\mathbf{Y})$. Among them, two typical choices, which are widely used in machine learning area [8], are listed as follows:

(i) The radial basis functions (RBFs), such as

$$K(\mathbf{X},\mathbf{Y}) = \exp\{-\gamma\|\mathbf{X}-\mathbf{Y}\|^2\}, \tag{13}$$

where $\gamma$ is a positive real value.

(ii) The polynomial functions:

$$K(\mathbf{X},\mathbf{Y}) = (\langle\mathbf{X},\mathbf{Y}\rangle + 1)^d = (\mathbf{X}^{\mathrm{T}}\mathbf{Y}+1)^d, \tag{14}$$

where $d$ is the degree of the polynomial function.

Note that it is not required that the kernel functions in Eq. (12) must meet the *Mercer Theorem*, although both Eqs. (13) and (14) do [8].

This choice of $\mathbf{F}(\mathbf{X})$ however incurs a special situation in the property of $\mathbf{F}(\mathbf{X})$. Because the functional vector $\mathbf{K}(\bullet,\mathbf{X})$ is based upon a set of available patterns, or the *reference set* $\boldsymbol{\Omega}_{\mathbf{X}}$, $\mathbf{K}(\bullet,\mathbf{X})$ will vary with the different choice of $\boldsymbol{\Omega}_{\mathbf{X}}$. In order to roughly capture the variance of $\mathbf{K}(\bullet,\mathbf{X})$, we introduce a disturbance term into Eq. (12), and thus obtain a new realization of $\mathbf{F}(\mathbf{X})$:

$$\mathbf{F}(\mathbf{X}) = \mathbf{K}(\bullet,\mathbf{X}) + \varepsilon, \tag{15}$$

where $\varepsilon = [\varepsilon_1 \ \varepsilon_2 \ \cdots \ \varepsilon_k]^{\mathrm{T}}$, where $\varepsilon_i$, $i=1,\ldots,k$, are independent random variables with zero mean and a constant variance of $\tau$. That is,

$$E\{\varepsilon_i\} = 0, \qquad E\{\varepsilon_i\varepsilon_j\} = \begin{cases}\tau, & i=j,\\ 0, & i\neq j.\end{cases}$$

By substituting Eq. (15) into Eqs. (1) and (3), we obtained the within-class and between-class matrices regarding $\mathbf{F}(\mathbf{X})$

as follows:

$$\mathbf{S}_w^{\mathbf{F(X)}} = \mathbf{S}_w^{\mathbf{K(\bullet,X)}} + \tau\mathbf{I}, \tag{16a}$$

$$\mathbf{S}_b^{\mathbf{F(X)}} = \mathbf{S}_b^{\mathbf{K(\bullet,X)}}, \tag{16b}$$

where $\mathbf{S}_w^{\mathbf{K(\bullet,X)}}$ and $\mathbf{S}_b^{\mathbf{K(\bullet,X)}}$ are the within-class and between-class matrices regarding $\mathbf{K(\bullet,X)}$, respectively.

More light can be shed to the effects of adding the disturbance term by substituting Eq. (16) into Eq. (8),

$$J(\mathbf{A}) = tr\{(\mathbf{A}^{\mathrm{T}}\mathbf{S}_w^{\mathbf{K(\bullet,X)}}\mathbf{A} + \tau\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}(\mathbf{A}^{\mathrm{T}}\mathbf{S}_b^{\mathbf{K(\bullet,X)}}\mathbf{A})\}. \tag{17}$$

Eq. (17) presents a relationship between adding a disturbance term and adding a regularization term. That is, adding the disturbance term is equivalent to adding a regularization term of the transform matrix **A**, which makes the role of the parameter $\tau$ in KFFE algorithm clearer. That is, if $\tau$ is too big, the resultant FE functions tend to under fit the available patterns, while, if $\tau$ is too small, the resultant FE functions are more likely to over fit the available patterns, and result in poor generalization. Therefore, the value of $\tau$ has the function of adjusting the generalization property of the KFFE algorithm. This mechanism is very useful for this kind of learning-from-training-pattern algorithms. Because $\tau$ is directly related to the generalization property of the whole algorithm, we call $\tau$ the *regularization coefficient*. In Section 5, an experiment is designed and implemented to further demonstrate the effect of $\tau$ in avoiding overfitting.

By substituting Eq. (16) into Eq. (11), we can finally obtain a general-purpose nonlinear FE algorithm, Kernel-function-based feature extraction method (KFFE). The steps to implement this algorithm can thus be summarized as follows:

(1) Form a kernel matrix **K** from a training set $\mathbf{\Theta}_X$ according to Eq. (12). That is, if $\mathbf{\Theta}_X$ is represented as

$$\mathbf{\Theta}_X = [\mathbf{X}_1\ \mathbf{X}_2\ \cdots\ \mathbf{X}_N],$$

the kernel matrix **K** can be represented as:

$$\mathbf{K} = [\mathbf{K}(\bullet,\mathbf{X}_1)\ \mathbf{K}(\bullet,\mathbf{X}_2)\ \cdots\ \mathbf{K}(\bullet,\mathbf{X}_N)]. \tag{18}$$

(2) Form the within-class matrix $\mathbf{S}_w^{\mathbf{K(\bullet,X)}}$ and between-class matrix $\mathbf{S}_b^{\mathbf{K(\bullet,X)}}$ regarding $\mathbf{K(\bullet,X)}$ by substituting Eq. (18) into Eqs. (2) and (4), respectively.

(3) Calculate matrix

$$\mathbf{C} = (\mathbf{S}_w^{\mathbf{K(\bullet,X)}} + \tau\mathbf{I})^{-1}\mathbf{S}_b^{\mathbf{K(\bullet,X)}}. \tag{19}$$

(4) Eigen-decompose the matrix **C**, and build the transform matrix **A**:

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_m],$$

where $\mathbf{C}\mathbf{a}_i = \lambda_i\mathbf{a}_i,\ i = 1,\ldots,k,\ \lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_k$, and $m \leqslant k$.

(5) By substituting matrix **A** obtained in Eq. (4) into Eq. (10), the final FE function for a given pattern **X** is obtained as

$$\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{K}(\bullet,\mathbf{X}) = \begin{bmatrix} \sum_{i=1}^{k} \alpha_{i,1}K(\mathbf{X}_i,\mathbf{X}) \\ \sum_{i=1}^{k} \alpha_{i,2}K(\mathbf{X}_i,\mathbf{X}) \\ \vdots \\ \sum_{i=1}^{k} \alpha_{i,m}K(\mathbf{X}_i,\mathbf{X}) \end{bmatrix} = \begin{bmatrix} H_1(\mathbf{X}) \\ H_2(\mathbf{X}) \\ \vdots \\ H_m(\mathbf{X}) \end{bmatrix}, \tag{20}$$

where $H_i(\mathbf{X})$ is called the $i$th *feature extraction function*, where $i = 1,\ldots,m$.

We note that the form of FE functions in Eq. (20) is similar to that of the KFD-like algorithms proposed in Refs. [13,15–17]. However, all of the KFD-like algorithms were derived through applying the "*kernel trick*" [18] to some linear algorithms, while our KFFE algorithm is obtained simply by replacing the original pattern with a functional vector **F(X)**, and is thus much more straightforward. Our simply derivation conveys an insight into these KFD-like algorithms. That is, we can obtain similar form of FE functions without even implicitly mapping the patterns from the input-space into a huge dimensional feature-space using the "*kernel trick*", which provides a different perspective to understand the impact of the feature-space on the final performance of the KFD-like algorithms.

From our derivation we can also see that the KFFE algorithm does not require the selected kernel functions to meet the Mercer's Theorem [8], while all of the KFD-like algorithms do. This implies a wider choice of kernel functions in KFFE algorithm. Furthermore, the derivation of our KFFE algorithm demonstrates that it is not necessary for the reference set $\mathbf{\Omega_X}$ to be the same as the training set $\mathbf{\Theta_X}$. In contrast, almost all of the KFD-like algorithms require $\mathbf{\Omega_X}$ to be the same as $\mathbf{\Theta_X}$, which is an intrinsic consequence of deriving the KFD-like algorithms through "kernelizing" linear algorithms. From these two perspectives, most of the KFD-like algorithms can therefore be considered as the special cases of the KFFE algorithm.

However, that unnecessary requirement of equating $\mathbf{\Omega_X}$ with $\mathbf{\Theta_X}$ introduces at least two difficult situations in most of the KFD-like algorithms. The first difficulty arises, when the training set $\mathbf{\Theta_X}$becomes very large; then the number of element functions in **F(X)** will be very large according to Eq. (12), which in turn will make the resultant FE algorithm very computationally expensive. The second difficulty happens when trying to take the inverse of $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$, which is required in some of the KFD-like algorithms. However, if the number of patterns in the training set is $N$, and the number of classes involved is $L$, the rank of $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ is at most $N-L$ [15]. However, $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ is an $N\times N$ matrix when the algorithms are directly derived via kernelizing the linear algorithms, which makes the direct inverse of $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ impossible. Currently, the method that most of the KFD-like algorithms employed to deal with the singularity of $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ is to replace $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ with $(\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}+\tau\mathbf{I})$ [13,15,17]. However, none of these algorithms provide a satisfactory explanation for the physical meaning of $\tau$, and instead simply suggest that introducing $\tau\mathbf{I}$ is helpful in stabilizing, or regularizing, the algorithm. In contrast, in our KFFE algorithm, $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ is a $k\times k$ matrix, where $k$ is the size of the reference set $\mathbf{\Omega}_\mathbf{X}$, and can be chosen by us. Therefore, direct inversion of $\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}$ becomes possible in our KFFE. Moreover, although we finally also adopt the inverse of $(\mathbf{S}_w^{\mathbf{K}(\bullet,\mathbf{X})}+\tau\mathbf{I})$ in our implementation, the physical meaning of the parameter $\tau$ becomes relatively clearer during our derivation, as well as the role that it plays in the whole algorithm.

Another issue worth mentioning relates to the between-class matrix $\mathbf{S}_b^{\mathbf{K}(\bullet,\mathbf{X})}$. From its definitions (3) and (4), we know that the rank of $\mathbf{S}_b^{\mathbf{K}(\bullet,\mathbf{X})}$ will be no more than $L-1$, where $L$ is the number of classes involved. Therefore, from Eq. (19) we know that our KFFE algorithm can only maximally extract $L-1$ meaningful features, which will limit its flexibility when the number of classes, $L$, is small. For example, for a 2-class problem, the KFFE currently can only extract one meaningful feature. One approach for addressing this limitation was discussed in our previous study [25], and that result can be readily incorporated into the KFFE algorithm by introducing a data-set-dependent structural matrix to the $\mathbf{F}(\mathbf{X})$ in Eq. (12), which can also be considered as a new choice of the $\mathbf{F}(\mathbf{X})$ in the framework.

Finally, we end this section with a summary of the parameters required by the KFFE algorithm:

(1) Type of kernel function, as well as the parameter(s) associated with it, such as the $\gamma$ associated with RBF kernel in Eq. (13) and the degree $d$ associated with the polynomial kernel in Eq. (14).
(2) Regularization coefficient $\tau$.
(3) Number of new features to extraction $m$.
(4) Choice of the reference set $\mathbf{\Omega}_\mathbf{X}$.

As for $m$, we generally choose the maximally extractable number of features, which is $L-1$, where $L$ is the number of classes involved. As for the choice of reference set $\mathbf{\Omega}_\mathbf{X}$, when the training data set is not too large, we tend to set the whole training set as the reference set; when the training set is very large, how to select an optimal reference set from the training set is still an open topic. The ideas proposed in [19,20,24] could be promising directions, although the algorithm in Refs. [19,20] only targets 2-class problems. How to directly find an optimal set of kernel parameters and the regularization coefficient $\tau$, given a particular problem, is unfortunately also an open topic. Cross-validation-based parameter selection is generally used in real-world practice.

## 5. Mean–STD–norm feature extraction (MSNFE)

In this section, a new FE algorithm is proposed by employing a different way to choose the elements of $\mathbf{F}(\mathbf{X})$ in Eq. (10).

With the time-series-type data in mind and inspired by the wavelet analysis theory, we speculate that, if we can extract characteristic of a pattern $\mathbf{X}$ at its different scale levels, a linear combination of all these characteristics is promising for generating a set of high-performance features of the pattern $\mathbf{X}$. This idea exactly fits the general FE framework. That is, as long as we can define a way to extract the characteristics of $\mathbf{X}$ at different levels, and use them as the elements of $\mathbf{F}(\mathbf{X})$ in Eq. (10), the general FE framework can readily find a set of new features, which are the linear combination of the obtained characteristics. The new features can maximize the class separability.

In order to implement the above idea, we need to define both what kind of characteristics we want to extract and how to define the scale levels of a pattern $\mathbf{X}$. Given an $l$-element pattern $\mathbf{X}=[x_1\ x_2\ \cdots\ x_l]^\mathrm{T}$, its characteristics that we want to extract are defined by three basic operations:

(1) mean:

$$M(\mathbf{X})=\frac{1}{l}\sum_{i=1}^{l}x_i, \tag{21a}$$

(2) standard deviation (STD):

$$S(\mathbf{X})=\sqrt{\frac{1}{L-1}\left[\sum_{i=1}^{l}x_i^2-\left(\sum_{i=1}^{l}x_i\right)^2\right]}, \tag{21b}$$

(3) norm:

$$N(\mathbf{X})=\|\mathbf{X}\|=\sqrt{\sum_{i=1}^{l}x_i^2}. \tag{21c}$$

Scale levels of the pattern $\mathbf{X}$ are defined in Fig. 1. That is, we iteratively evenly split the pattern $\mathbf{X}$ into many segments in a way following a binary tree structure, which is illustrated in Fig. 1. Note that Fig. 1 only plots two levels of the splitting. In fact, the number of levels that this splitting can move down is only limited by the number of features in $\mathbf{X}$.

We can thus define a group of functions by applying the three operations defined in Eq. (21) to each of the segments in Fig. 1, and use this group of functions as the element

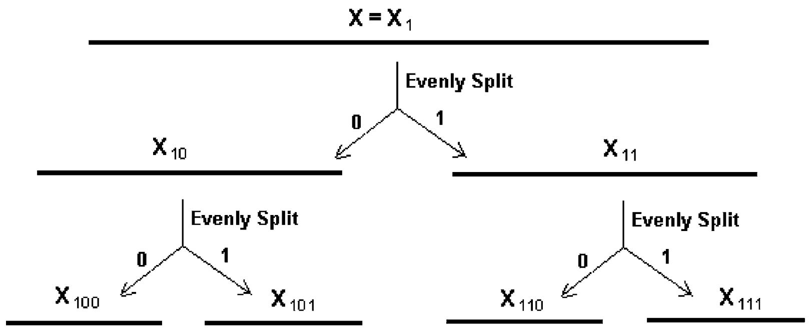

Fig. 1. Illustration of iteratively evenly splitting pattern **X** into many segments.

functions of $\mathbf{F}(\mathbf{X})$ in Eq. (10). That is,

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} F_1(\mathbf{X}) \\ F_2(\mathbf{X}) \\ F_3(\mathbf{X}) \\ F_4(\mathbf{X}) \\ \vdots \\ F_{k-1}(\mathbf{X}) \\ F_k(\mathbf{X}) \end{bmatrix} = \begin{bmatrix} M(\mathbf{X}_1) \\ S(\mathbf{X}_1) \\ N(\mathbf{X}_1) \\ M(\mathbf{X}_{10}) \\ \vdots \\ S(\mathbf{X}_{1\cdots1}) \\ N(\mathbf{X}_{1\cdots1}) \end{bmatrix}. \qquad (22)$$

Because the three basic operations are *mean*, *standard deviation*, and *norm*, and segments were obtained by splitting a pattern **X** in a binary tree structure, the group of functions in Eq. (22) is called as *MSNTree functions*. The feature extraction algorithm obtained by replacing Eq. (22) into the FE framework (10) is named as *mean–STD–norm feature extraction* (*MSNFE*) *algorithm*. Because two operations defined in Eq. (21) are nonlinear operations, the MSNFE algorithm is also a nonlinear FE algorithm with regard to **X**.

Note that the procedure to build up the MSNTree functions is inspired mainly from experience and intuition, instead of rigorous theoretically analysis. Neither the selection of the three basic operations nor the way to split **X** is the only proven optimal choice. One of our major purposes to propose MSNFE is to demonstrate the utility of the general FE framework, and to illustrate a different approach to define the $\mathbf{F}(\mathbf{X})$ under framework (10). In fact, it can be predicted that the MSNFE algorithm performs well mainly for time-series-type problems according to the way it constructs the MSNTree functions. This prediction is confirmed by the experimental results presented in Section 6.

The MSNFE algorithm only requires one parameter, which is the number of levels to split a given pattern **X**. The maximal possible value of this parameter is determined by the number of features in **X** through the relationship in (23):

$$LN_{max} = \lfloor \log_2(n/2) \rfloor, \qquad (23)$$

where $LN_{max}$ is the maximal number of levels that the splitting of **X** can move down, and $n$ is the number of features in **X**. Basically, Eq. (23) guarantees that each segment of **X** at least has two features inside, which ensures the three basic operations on that segment nontrivial.

According to the discussion in Section 4, the regularization coefficient $\tau$ plays an important role in preventing the FE algorithms from overfitting the training set. Therefore, the regularization coefficient $\tau$ is also introduced in the MSNFE algorithm. Surely, the $\tau$ used in MSNFE algorithm losses the physical meaning when it was introduced in KFFE algorithm, and it is used here only for the purpose of algorithmic robustness.

Similar to the KFFE algorithm, the MSNFE algorithm can be summarized as follows:

(1) Split each pattern $\mathbf{X}_i$ in the training set $\mathbf{\Theta}_X = [\mathbf{X}_1\ \mathbf{X}_2\ \cdots\ \mathbf{X}_N]$ in the way demonstrated in Fig. 1. Construct the functional vector $\mathbf{F}(\mathbf{X}_i)$ from each pattern $\mathbf{X}_i$ according to Eq. (22), and form the matrix $\mathbf{F}(\mathbf{\Theta}_X)$ as

$$\mathbf{F}(\mathbf{\Theta}_X) = [\mathbf{F}(\mathbf{X}_1)\ \mathbf{F}(\mathbf{X}_2)\ \cdots\ \mathbf{F}(\mathbf{X}_N)].$$

(2) Form the within-class matrix $\mathbf{S}_w^{\mathbf{F}(\mathbf{X})}$ and between-class matrix $\mathbf{S}_b^{\mathbf{F}(\mathbf{X})}$ regarding $\mathbf{F}(\mathbf{X}_i)$ by substituting $\mathbf{F}(\mathbf{X}_i)$ into Eqs. (2) and (4), respectively.

(3) Calculate the matrix $\mathbf{C} = (\mathbf{S}_w^{\mathbf{F}(\mathbf{X})} + \tau\mathbf{I})^{-1}\mathbf{S}_b^{\mathbf{F}(\mathbf{X})}$.

(4) Eigendecompose the matrix **C**, and build the transform matrix **A**:

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_m],$$

where $\mathbf{C}\mathbf{a}_i = \lambda_i \mathbf{a}_i$, $i = 1, \ldots, k$, $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_k$, and $m \leqslant k$.

(5) By substituting matrix **A** obtained in Eq. (4) into Eq. (10), the final FE function for a given pattern **X**

is obtained as

$$\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \sum_{i=1}^{k} \alpha_{i,1} F_i(\mathbf{X}) \\ \sum_{i=1}^{k} \alpha_{i,2} F_i(\mathbf{X}) \\ \vdots \\ \sum_{i=1}^{k} \alpha_{i,m} F_i(\mathbf{X}) \end{bmatrix} = \begin{bmatrix} H_1(\mathbf{X}) \\ H_2(\mathbf{X}) \\ \vdots \\ H_m(\mathbf{X}) \end{bmatrix}, \tag{20}$$

where $H_i(\mathbf{X})$ is called the $i$th *feature extraction function*, where $i = 1, \ldots, m$.

## 6. Experiments

*Experiment* 1: *Demonstrating the effect of* $\tau$ *in avoiding overfitting in KFFE algorithm*. The experimental data set is a group of 2-feature patterns from two classes, which is plotted in Fig. 2. The random process to generate the patterns is represented in Eq. (24):

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} r\cos(\phi) \\ r\sin(\phi) \end{bmatrix}, \tag{24}$$

where $\phi$ is a random variable with a uniform distribution between $[-\pi, \pi]$. For class 1, $r$ is a zero-mean Gaussian random variable with a standard deviation (STD) of 0.5, while, for class 2, $r$ is a Gaussian random variable with a mean of 0.75 and a STD of 0.5.

We set $m = 1$, and used a RBF kernel with $\gamma = 1$. When we set $\tau = 0.01$, the resultant FE function $H_1(\mathbf{X})$ is plotted in Fig. 3. It is clearly shown that, although this FE function is good enough to generate new feature samples to distinguish the patterns in training set, the unnecessary warp suggests an overfitting over the current training set.

After we set $\tau = 5$, the resultant FE function $H_1(\mathbf{X})$ is plotted in Fig. 4. We can see that the shape of the FE function becomes more regular, and is basically in accordance with the ideal feature extraction function directly derived from the generation random process defined in Eq. (21). This experiment confirms our discussion in Section 4 that bigger $\tau$ is helpful in avoiding overfitting.

*Experiment* 2: *Demonstrating the performance of the KFFE algorithm given a synthetic multi-class data set*. The original data set is a 3-class problem, formed by a group of 2-feature patterns centering around 6 different locations, which is plotted in Fig. 5(a). From the plot, we can see that this is not a linearly separable problem. We set $m=2$, $\tau=0.1$, and used a RBF kernel with $\gamma = 1$. The distribution of the new patterns, $\mathbf{R}$, formed from 2 extracted features is shown in Fig. 5(b). We can see that the class distribution of the new patterns becomes generally linearly separable.

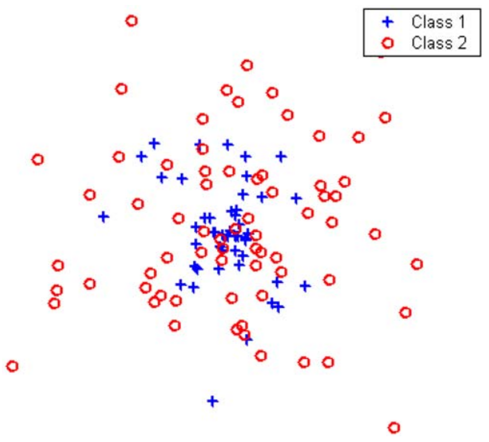

Fig. 2. Original distribution of a 2-class problem composed by 2-feature patterns.

The two FE functions, $H_1(\mathbf{X})$ and $H_2(\mathbf{X})$, employed to extract the new patterns are shown in Figs. 5(c)–(f).

*Experiment* 3: *Demonstrating the performance of KFFE and MSNFE algorithm on real-world data sets*. Some facts of the selected data sets are listed in Table 1. Note that the last column in Table 1 shows the number of new features that will be extracted from the original patterns using both the KFFE and MSNFE algorithms. Basically, they are chosen as $L - 1$, where $L$ is the number of classes involved. As for the reason why they are chosen in this way, please refer to the last two paragraphs in Section 4, and we will not repeat it here.

The data set IRIS and Glass are from the UCI Machine Learning Repository. The data set FORTÉ is a group time-series data collected from a satellite named FORTÉ [26]. It is composed of 143 samples from seven different types of lightning, and each of the samples is a 3180-element time series. The relatively small number of available samples, the considerable number of classes involved, and the huge number of features in each sample altogether make it a difficult classification problem.

Although our goal is to examine the performance of the FE algorithms, it is difficult to isolate the role of the classifiers used when we employ the classification rate as the performance measurement. In order to avoid being misled by a special classifier, two quite different classifiers, support vector machine (SVM) classifiers and $k$-nearest-neighbor (kNN) classifiers, are employed. Note that each FE algorithm requires a set of parameters to be pre-determined, such as the kernel function in KFFE, the number of levels in MSNFE, and the regularization coefficient $\tau$ in both algorithms. Ten-fold cross-validation method is employed to find the optimal parameter set for each problem. Once the optimal set of parameters is identified, the 10-fold cross-validation result under this parameter set is reported as the estimation of the final classification rate.

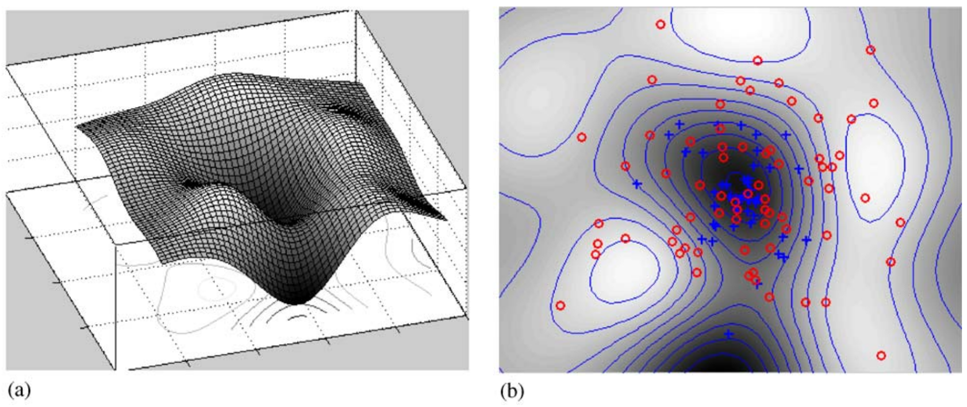
Fig. 3. (a) 3-D plot of the first FE function, $H_1(\mathbf{X})$ when $\tau = 0.01$. (b) Gray-scale representation of the $H_1(\mathbf{X})$ in (a), overlapped with original patterns.

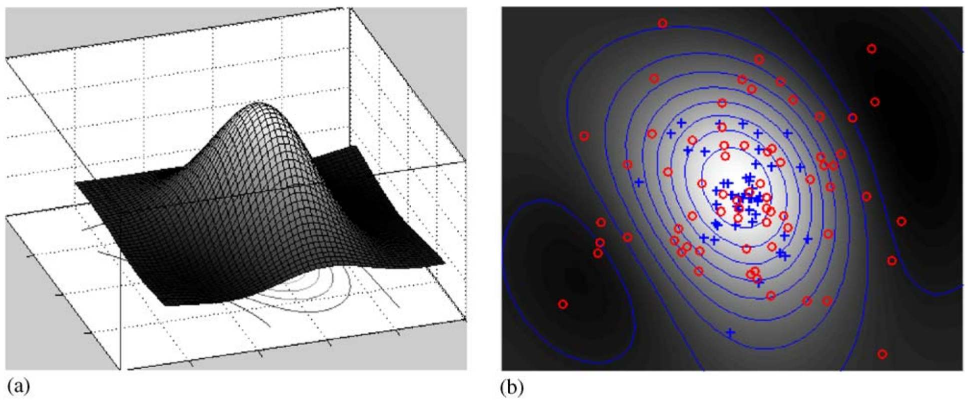
Fig. 4. (a) 3-D plot of the first FE function, $H_1(\mathbf{X})$ when $\tau = 1$. (b) Gray-scale representation of the $H_1(\mathbf{X})$ in (a), overlapped with original patterns.

(1) *Experimental results obtained using KFFE algorithm*: Nonlinear SVM classifiers are applied to the original patterns, while linear SVM classifiers are applied to the extracted new patterns. This is for a "fair" comparison between the FE extraction capability embedded in nonlinear SVM classifiers and the KFFE algorithms. In order to demonstrate that our KFFE algorithm differentiates the reference set $\mathbf{\Omega_X}$ from the training set $\mathbf{\Theta_X}$, besides presenting the results when $\mathbf{\Omega_X}$ is defined the same as the training set $\mathbf{\Theta_X}$, we also presented the results when the reference set $\mathbf{\Omega_X}$ is formed by just picking up every other pattern in the training set $\mathbf{\Theta_X}$. That is, $\mathbf{\Omega_X}$ is composed of only half of the patterns in $\mathbf{\Theta_X}$.

Comparing the last two columns in Table 2, we can see that the results obtained when the reference set $\mathbf{\Omega_X}$ is only half of the training set $\mathbf{\Theta_X}$ are fairly comparable with those obtained when $\mathbf{\Omega_X}$ and $\mathbf{\Theta_X}$ are the same. When $\mathbf{\Omega_X}$ is half of $\mathbf{\Theta_X}$, the size of matrix $\mathbf{C}$ in Eq. (19) is only one forth of that of the $\mathbf{C}$ when $\mathbf{\Omega_X} = \mathbf{\Theta_X}$. The computation required can thus be scaled down with an even bigger factor. This result validates that separating $\mathbf{\Omega_X}$ from $\mathbf{\Theta_X}$ in the KFFE algorithm is a reasonable direction to reduce computation. Also, the smaller reference set $\mathbf{\Omega_X}$ is currently obtained simply by picking every other pattern in $\mathbf{\Theta_X}$. It is highly possible that better performance can be achieved if we employ a method to construct the $\mathbf{\Omega_X}$ from $\mathbf{\Theta_X}$ optimally, or sub-optimally. Note that a different approach to reduce the computation of the original KFD algorithms for 2-class problems is presented in Refs. [19,20]. Constructing a sub-optimal $\mathbf{\Omega_X}$ for KFFE algorithm using the idea in [19,20] is a promising future topic.

(2) *Experimental results obtained using MSNFE algorithm*: The experimental results obtained using MSNFE algorithm is presented in Table 3. Again, comparable, even better, classification performance was achieved based on a much smaller number of features extracted using MSNFE.

In additions, compared with the results in Table 2, Table 3 illustrates two more interesting points. First, both FE

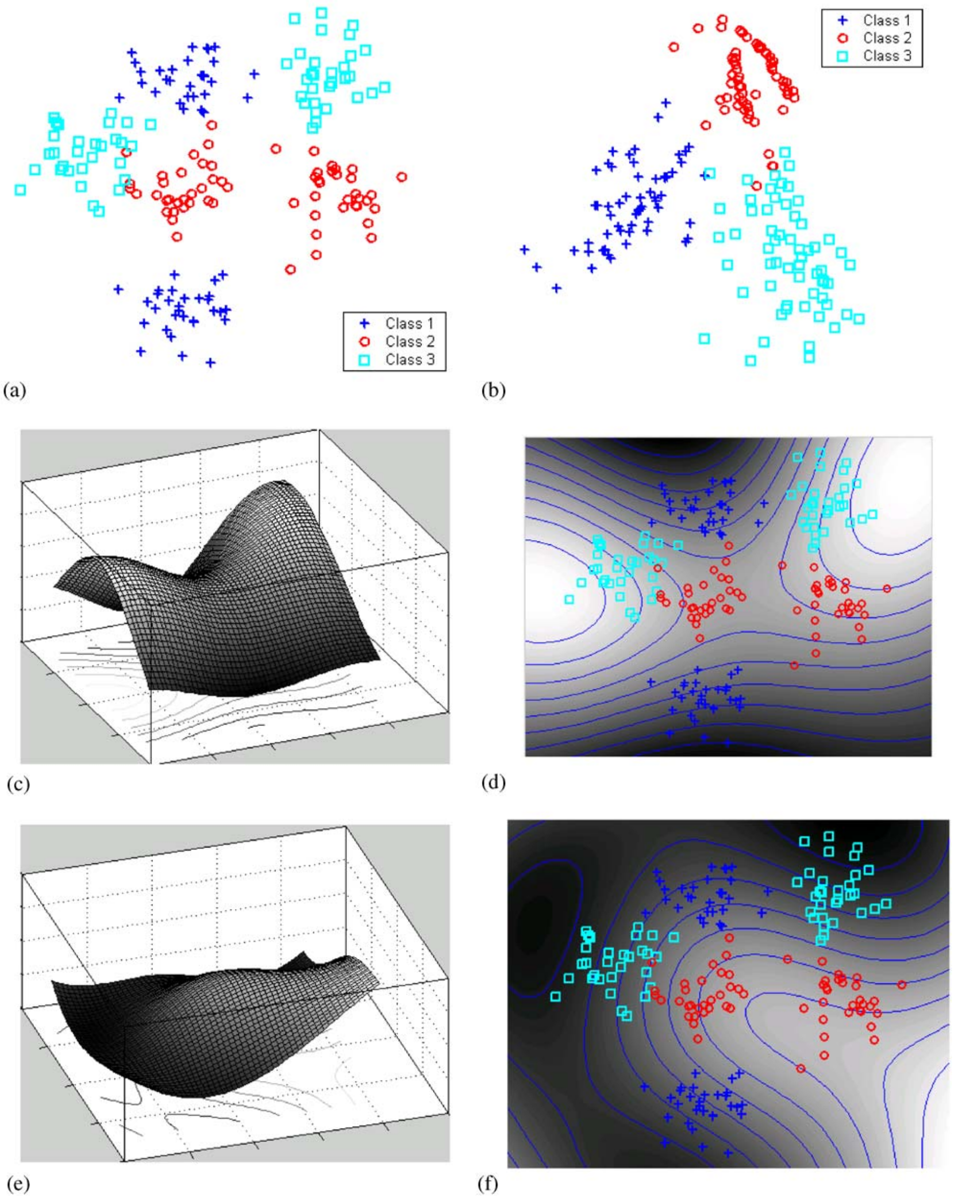

Fig. 5. (a) Original distribution of a 3-class problem composed by 2-feature patterns. (b) Resultant distribution of new patterns formed from 2 extracted features. (c) 3-D plot of the first FE function, $H_1(\mathbf{X})$. (d) Gray-scale representation of the $H_1(\mathbf{X})$ in (c), overlapped with original patterns. (e) 3-D plot of the second FE function, $H_2(\mathbf{X})$. (f) Gray-scale representation of the $H_2(\mathbf{X})$ in (e), overlapped with original patterns.

algorithms, after reducing the number of features, improve the classification rate for the FORTÉ Data, degrade it in the UCI Glass, and maintain it for the UCI IRIS. This observation suggests that how significantly FE algorithms can affect the overall performance in an application is heavily related to the property of the underlying problems. It is easy to reason that, if there is not too much redundancy among the features in the original patterns, FE algorithms that forcefully reduce the number of features in the new patterns will corrupt the class separability. Second, significant performance improvement is achieved using the MSNFE algorithm for FORTÉ Data. This observation confirms our prediction in Section 5 that the MSNFE will work well for time-series-type problem due to the way the MSNTree functions are constructed. Meanwhile, considerable performance corruption is also observed using this algorithm for UCI Glass, which again coincides with the idea presented in the *No Free Lunch Theorem* [21].

Table 1
Experimental data sets

| | No. class | No. available patterns | No. original features in each original pattern | No. extracted features in each new pattern |
|---|---|---|---|---|
| FORTÉ | 7 | 143 | 3180 | 6 |
| UCI IRIS | 3 | 150 | 4 | 2 |
| UCI glass | 6 | 214 | 13 | 5 |

Table 2

| CV rate (%) | Nonlinear SVM with original patterns | Linear SVM with new patterns ($\mathbf{\Omega_X} = \mathbf{\Theta_X}$) | Linear SVM with new patterns ($\mathbf{\Omega_X} =$ half $\mathbf{\Theta_X}$) |
|---|---|---|---|
| (a) *Results obtained using KFFE algorithm and SVM classifiers* | | | |
| FORTÉ data | 68.62 | 70.43 | 69.95 |
| UCI IRIS | 98.67 | 98.67 | 96.00 |
| UCI glass | 73.81 | 73.31 | 72.45 |

| CV rate (%) | kNN with original patterns | kNN with new patterns ($\mathbf{\Omega_X} = \mathbf{\Theta_X}$) | kNN with new patterns ($\mathbf{\Omega_X} =$ half $\mathbf{\Theta_X}$) |
|---|---|---|---|
| (b) *Results obtained using KFFE algorithm and kNN classifiers* | | | |
| FORTÉ data | 69.14 | 73.38 | 73.43 |
| UCI IRIS | 97.33 | 98.00 | 98.00 |
| UCI glass | 75.67 | 74.24 | 73.90 |

Table 3

| CV rate (%) | Nonlinear SVM with original patterns | Linear SVM with new patterns |
|---|---|---|
| (a) *Results obtained using MSNFE algorithm and SVM classifiers* | | |
| FORTÉ data | 68.62 | 76.19 |
| UCI IRIS | 98.67 | 98.67 |
| UCI glass | 73.81 | 65.30 |

| CV rate (%) | kNN with original patterns | kNN with new patterns |
|---|---|---|
| (b) *Results obtained using MSNFE algorithm and kNN classifiers* | | |
| FORTÉ data | 69.14 | 78.48 |
| UCI IRIS | 97.33 | 98.00 |
| UCI glass | 75.67 | 72.88 |

In sum, the benefits of the proposed algorithms can be summarized as reducing the number of features while maintaining, or even improving, the class separability. Less number of features, or lower pattern dimension, has many positive impacts on the downstream processing, such as a wider choice of downstream processing algorithms, less demand in memory, and shorter processing time. However, given a class-separability criterion in Eq. (5), existing linear FE algorithms generally reduce the number of features at the cost of class separability [1,11]. In contract, the experimental results obtained using KFFE and MSNFE algorithms, which are presented in Tables 2 and 3 respectively, illustrate that the patterns obtained from both algorithms can achieve comparable class separability with much smaller number of features. In some cases, they can even outperform the original patterns. Meanwhile, Table 4 shows the computational advantages at the classification stage for the feature reduction that we have done in these experiments.

Table 4
Computation time for classifiers is reduced when using a smaller number of features

| Data set | | SVM classifier (in s) | KNN classifier (in s) |
|---|---|---|---|
| FORTÉ data | w/o Feature Extraction | 9.467 | 5.146 |
| | w/ Feature Extraction | 0.092 | 0.232 |
| UCI IRIS | w/o Feature Extraction | 0.054 | 0.162 |
| | w/ Feature Extraction | 0.027 | 0.162 |
| UCI glass | w/o Feature Extraction | 0.160 | 0.275 |
| | w/ Feature Extraction | 0.084 | 0.300 |

Reported time does not include time to perform the feature extraction.

## 7. Conclusions

Feature extraction (FE) methods have been widely employed in the statistics literature. Fukunaga [1] provides an excellent overview in the context of supervised classification; see also Ref. [23] which describes the problem from a machine learning and data mining point of view. Gordon [27] discusses the issue primarily from the point of view of unsupervised classification, and makes some cogent remarks on the confounding effects of including too many features. Fowlkes et al. [28], in particular, study the problem of feature selection for clustering.

In this paper, a general FE framework for multi-class problems is proposed by extending the linear discriminant analysis. The significance of introducing this FE framework is that it provides a mechanism to separate the nonlinearity of a FE algorithm from its optimization procedure. Based on this framework, two concrete FE algorithms are readily developed. The first algorithm, kernel function feature extraction (KFFE) algorithm, is derived based a set of kernel functions. Discussion on the KFFE algorithm reveals that some existing KFD-like algorithms are its special cases. Moreover, the KFFE algorithm proposes a new direction to reduce the computation required by the existing KFD-like algorithms. The second algorithm, MSNFE algorithm, is obtained by introducing into the FE framework a different set of functions, the MSNTree functions. Experimental results suggest that the MSNFE algorithm is promising in boosting the performance of the time-series-type problems. Three fairly different problems are specially chosen to demonstrate the different performance of the proposed algorithms, which hopefully help readers have an easier judgment on when our algorithms should and should not be considered in their applications.

The research presented in this paper also invokes a few topics, which are still open, such as (a) developing new algorithms based on different choices of $\mathbf{F}(\mathbf{X})$ in (10); (b) developing the framework by choosing different class-separability criterion from the criterion $J$ in Eq. (5). Meanwhile, some open topics regarding the KFFE algorithm include (a) how to directly determine an optimal set of algorithm parameters; (b) when the reference set $\mathbf{\Omega_X}$ is a subset of the training set $\mathbf{\Theta_X}$, how to define an optimal subset $\mathbf{\Omega_X}$ from $\mathbf{\Theta_X}$; (c) how to refine the KFFE algorithm to make it more computationally efficient.

## Acknowledgements

We want to thank Sean Davis and David Suszcnsky for their help in preparing the FORTÉ Data for us. Some of the ideas were first shaped when Junshui Ma was under the supervision of Professor Stanley C. Ahalt in the Department of Electrical Engineering, Ohio State University. This work is supported by the NASA project NRA-00-01-AISR-088 and by the Los Alamos Laboratory Directed Research and Development (LDRD) program.

## References

[1] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd Edition, Academic Press, Boston, 1990.

[2] F.D. Garber, N.F. Chamberlain, O. Snorrason, Time-domain and frequency-domain feature selection for reliable radar target identification, Proceedings of the IEEE 1988 National Radar Conference, Ann Arbor, MI, April 20–21, 1988, pp. 79–84.

[3] J. Li, P. Stoica, Efficient mixed-spectrum estimation with applications to target feature extraction, IEEE Trans. Signal Process. 44 (1996) 281–295.

[4] J. Ma, X. Du, S. Ahalt, 2D HRR radar data modeling and processing, Multidimensional Systems Signal Process. (special issue), Radar Signal Process. Appl. 14 (2003) 25–48.

[5] L.C. Potter, R.L. Moses, Attributed scattering centers for SAR ATR, IEEE Trans. Image Process. Special Issue Automat. Target Recogn. 6 (1) (1997) 79–91.

[6] T.Y. Young, The reliability of linear feature extractors, IEEE Trans. Comput. 20 (1971) 967–971.

[7] H.L. VanTrees, Detection, Estimation, and Modulation Theory: Part I, Wiley, New York, 1968.

[8] B. Schölkopf, C. Burges, A. Smola, Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, 1999.

[9] W. Zhao, Discriminant component analysis for face recognition, Proceedings of the International Conference on Pattern Recognition, Barcelona, 2000.

[10] W. Zhao, A. Krishnaswamy, R. Chellappa, D.L. Swets, J. Weng, Discriminant analysis of principal components for face

recognition, Face Recognition: from Theory to Applications, 4th Edition, Springer, New York, 1998, pp. 73–85.

[11] C. Lee, D.A. Landgrebe, Feature extraction based on decision boundaries, IEEE Trans. Pattern Anal. Mach. Intell. 10 (4) (1993) 388–400.

[12] N. Saito, R. Coifman, Local discriminant bases, Technical Report, Department of Mathematics, Yale University, 1994.

[13] S. Mika, G. Ratsch, J. Weston, Fisher discriminant analysis with kernels, Proceedings of IEEE International Workshop on Neural Networks for Signal Processing, Madison, Wisconsin, August 1999, pp. 41–48.

[14] A. Ruiz, P.E. Lopez-de-Teruel, Nonlinear kernel-based statistical pattern analysis, IEEE Trans. Neural Networks 12 (1) (2001) 16–32.

[15] J. Ma, Feature study of high range resolution based automatic target recognition: analysis and extraction, Ph. D. Dissertation, the Ohio State University, May 2001.

[16] G. Baudat, F. Anouar, Generalized discriminant analysis using a kernel approach, Neural Comput. 12 (10) (2000) 2385–2404.

[17] V. Roth, V. Steinhage, Nonlinear discriminant analysis using kernel functions, in: S.A. Solla, T.K. Leen, K.-R. Muller (Eds.), Advances in Neural Information Processing Systems, Vol. 12, MIT Press, Cambridge, 1999, pp. 568–574.

[18] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowledge Discovery 2 (2) (1998) 121–167.

[19] K. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Networks 12 (2) (2001) 181–201.

[20] S. Mika, G. Rätsch, K. Müller, A mathematical programming approach to the Kernel Fish algorithm, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, Vol. 13, MIT Press, Cambridge, 2001, pp. 591–597.

[21] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evolut. Comput. 1 (1) (1997) 67–82.

[22] J. Ma, Function replacement vs. Kernel trick, Neurocomputing 50 (2003) 479–483.

[23] T. Hastie, R. Tibshirani, J. Fridman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, New York, 2001.

[24] D. Achlioptas, F. McSherry, B. Schölkopf, Sampling Techniques for Kernel Methods, in: T.G. Dietterich, S. Becher, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, Cambridge, MA, 2002.

[25] J. Ma, S. Perkins, J. Theiler, S. Ahalt, Modified Kernel-base nonlinear feature extraction, Proceedings of International Conference on Machine Learning and Applications, Las Vegas, Nevada, USA, June 2002, pp. 127–132.

[26] D.M. Suszcynsky, M.W. Kirkland, A. Jacobson, R. Franz, S. Knox, J.L. Guillen, J. Green, FORTE observations of simultaneous VHF and optical emissions from lightning: basic phenomenology, J. Geophys. Res.—Atmos. 105 (2000) 2191–2201.

[27] A.D. Gordon, Classification, Chapman & Hall, London, 1999.

[28] E.B. Fowlkes, R. Gnanadesikan, J.R. Kettenring, Variable selection in clustering, J. Classification 5 (1988) 205–228.

**About the author**—JUNSHUI MA received his Ph.D. degree in Electrical Engineering from the Ohio State University, Columbus, USA, in June 2001. He then became a post-doctor in Los Alamos National Lab, New Mexico, USA. In April 2003 he joined Aureon Biosciences Corp in New York as a machine learning researcher. His major research interests include machine learning, data mine, and signal/image processing.