Pattern Recognition 46 (2013) 1532–1555

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Three-fold structured classifier design based on matrix pattern

Zhe Wang [a,*], Changming Zhu [a], Daqi Gao [a], Songcan Chen [b,**]

[a] Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, PR China
[b] Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, PR China

ARTICLE INFO

Article history:
Received 1 March 2012
Received in revised form
10 November 2012
Accepted 6 December 2012
Available online 14 December 2012

Keywords:
Vector pattern
Matrix pattern
Global structure
Local structure
Classifier design
Pattern recognition

ABSTRACT

The traditional vectorized classifier is supposed to incorporate the class structural information but ignore the individual structure of single pattern. In contrast, the matrixized classifier is supposed to consider both the class and the individual structures, and thus gets a superior performance to the vectorized classifier. In this paper, we explore one middle granularity named the cluster between the class and individual, and introduce the cluster structure that means the structure within each class into the matrixized classifier design. Doing so can simultaneously utilize the class, the cluster, and the individual structures in the way that is from global to point. Therefore, the proposed classifier design here owns the three-fold structural information, and can bring the classification performance to an improving trend. In practice, we adopt the Modification of Ho–Kashyap algorithm with Squared approximation of the misclassification errors (MHKS) as the learning paradigm and develop a Three-fold Structured MHKS named TSMHKS. The advantage of the three-fold structural learning framework is considering different close degrees between samples so as to improve the performance. The experimental results demonstrate the feasibility and effectiveness of the TSMHKS. Furthermore, we discuss the theoretical and experimental generalization bound of the proposed algorithm.

1. Introduction

In statistical machine learning, the pattern to be dealt is generally represented by the vector, which can be taken as one point in a d-dimensional space [1–3]. The vector representation is supposed to bring a convenience since the vector is expediently dealt in mathematics. But if a pattern such as the image has its own structure, the vectorization would make the image lose some structural information such as spatial relationships between pixels [4]. Especially for large images, the vector representation would cause a high dimensional feature space, and thus increases the computational complexity and the space of storage. In order to solve the problems induced by the vector representation, a matrix representation strategy was developed [5–7]. The matrix representation learning was established in feature extraction. Yang et al. [7] first proposed two-dimensional principal component analysis (2DPCA) which extracted features directly from image matrices. 2DPCA was validated to have a superior performance to PCA in favor of both image classification and the reduction of computational complexity and the storage space. Li and Yuan [8] proposed two-dimensional linear discriminant analysis (2DLDA) which was also based on image matrices and could achieve a better recognition accuracy to linear discriminant analysis (LDA). Chen et al. [9] developed a more general matrixized feature strategy called MatPCA and MatFLDA. Compared with both PCA and LDA, the MatPCA and MatFLDA reshaped a given pattern into a new matrix one before extracting features. In this way, due to that the newly formed matrix pattern retained all the original components, the original information generally seems not to be lost. Further, some new implicit structural or contextual information could likely be additionally introduced, which was validated in the experiments [9]. Moreover, in order to guide a best way for describing a given texture such as matrix image, Nanni et al. [13] conducted different approaches. They found that the fusion approach with the uniform local quinary pattern (LQP) and the rotation invariant local quinary pattern could obtain a method that performed well. In this fusion approach, both the bin selection based on variance and the neighborhood preserving embedding (NPE) feature transform were applied.

Some researchers found that in the matrixized feature extractors such as 2DPCA and 2DLDA, the matrix representation is just applied in feature extraction, but the subsequent classifier design still resorts to the traditional vector-based technique. In other words, the operated pattern of the classifier itself is still of vector representation which has the following discriminant function:

$$g(x) = \tilde{w}^T x + w_0 \tag{1}$$

where $\tilde{w} \in \mathbb{R}^d$ is a weight vector, $w_0 \in \mathbb{R}$ is a bias. Therefore, we had proposed a matrixized classifier learning framework [10]

* Corresponding author. Tel.: +86 021 64253471.
** Corresponding author.
E-mail addresses: wangzhe@ecust.edu.cn (Z. Wang), 020100084@mail.ecust.edu.cn (C. Zhu), gaodaqi@ecust.edu.cn (D. Gao), s.chen@nuaa.edu.cn (S. Chen).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.12.003

which has the following discriminant function:

$$g(A) = u^T A\tilde{v} + v_0 \quad (2)$$

where $u \in \mathbb{R}^{d_1}, \tilde{v} \in \mathbb{R}^{d_2}$ are the two weight vectors respectively, and $v_0 \in \mathbb{R}$ is a bias. The differences between the matrixized classifier with (2) [10] and the vectorized one with (1) are: (a) the matrix pattern with the size of $d_1 \times d_2$ replaces the $d_1 d_2 \times 1$ vector one; (b) the two weight vectors in (2) respectively acting on the two sides of the matrix pattern A replaces the original single weight vector in (1); (c) the discriminant function (2) is the (1) imposed with the *Kronecker* product decomposability constraint, which was demonstrated in [10]. Further, a so-called fully matrixized approach was developed in [11,12], which means that both the feature extraction and the classifier design are directly based on matrix pattern. Their experiments [10] demonstrated that the fully matrixized approach indeed brings a significant advantage in terms of both classification and computational performance.

On the other hand, it can be found that the motivation of the matrixized learning above is based on the pattern itself. It means that both the matrixized feature extraction and classifier expect to get some structural information through the matrixization of pattern. Therefore, the matrixized learning can be supposed to consider the individual structural information, i.e. the information of the individual matrix pattern. In contrast, the vectorization of pattern might decrease the attention for the structure of the individual pattern. But it is fortunate that the classical linear or non-linear classifier design based on the vectorized pattern can still get some class discriminating information through some constraints in the learning model. We take the Modification of Ho–Kashyap algorithm with Squared approximation of the misclassification errors (MHKS) [14] for an example. MHKS is a vectorized linear classifier. It can integrate the class structure such as the discriminating information through making the distance between all the patterns belonging to different classes as far as possible. Since the matrixized classifiers such as MatMHKS [11] are derived from those vectorized one with class structure, the matrixized ones inherit the characteristic of class structure. Thus the matrixized classifier can be viewed as the one with both class and individual structure.

However, it can be found that in many real-world cases, patterns belonging to the same class would also have different structures. In other words, it might be coarse to just consider the discriminating information between different classes. Meanwhile, the individual structure induced by the matrixized methods just reflects the characteristic of the single pattern itself, and cannot describe the structural information of the patterns which are in the same class. Thus neither the vectorized nor matrixized classifiers discussed above can integrate the structure within each class. In order to solve the problem, in this paper we introduce the structure within each class into the matrixized classifier design. Compared with the discriminating structure between different classes, the structure within each class can be viewed more local. Compared with the individual structure of single pattern, the structure within each class can be viewed more global. For convenience, we call the discriminating structure between different classes, the structure within each class, and the individual structure of single pattern as the class, the cluster, and the individual structures respectively. Namely, in the design of the classifier, the class structure means the discriminating information between different classes. The cluster structure means the class itself information, i.e. the relationship between samples in the same class. The individual structure means the information of single pattern itself. Consequently, through introducing the structure within each class into the matrixized classifier, we can develop a new classifier design that owns the three-fold structural information and brings classification performance to an improving trend.

In practice, we adopt the vectorized and matrixized Modification of Ho–Kashyap algorithm with Squared approximation of the misclassification errors, i.e. MHKS [14] and MatMHKS [11] as the learning paradigms since (1) both of them fall into the regularization framework; (2) both employ a modification of the gradient descent with a heuristic update-rule and thus it is simple to achieve the optimization for their objective functions. Consequently, through capturing the underlying structures within each class with some unsupervised or supervised clustering techniques, we can obtain multiple clusters to enclose the patterns of each class, then introduce the multiple clusters into the matrixized classifier MatMHKS, and finally develop a three-fold structural classifier named TSMHKS. To demonstrate the feasibility of the proposed TSMHKS, we discuss its empirical generalization risk bound. The experimental results here validate the effectiveness of the TSMHKS. The advantage of the proposed learning strategy is to not only inherit both the class discriminating information and the individual structural information, but also to integrate the information from different clusters, where the samples in the same cluster are distributing as tightly as possible.

The rest of this paper are organized as follows. Section 2 reviews the related work of the structured classifiers and the family of different Ho–Kashyap algorithms. Section 3 gives the description of the proposed algorithm TSMHKS. In Section 4, the experimental results on some synthetical data sets and real-world benchmark data sets have shown the feasibility and effectiveness of the TSMHKS. Finally, conclusions are given in Section 5.

2. Related work

This section first reviews the work about the structured classifiers. Then since this paper takes the improved Ho–Kashyap algorithm [3] as the working paradigm, we here introduce the family of different Ho–Kashyap algorithms as well.

2.1. Structured classifiers

The structural information is important for designing a good classifier in machine learning. Recently, many techniques have been proposed to introduce as much structural information in data as possible so as to improve the generalization performance of classifiers. These classifiers introduced with structural information can be called structured classifiers. Support vector machine (SVM) [2,15] is one of the state-of-the-art classifiers and aims to find a hyperplane that can separate two classes of data with a maximal margin. Since SVM is a nice model, the structural techniques based on SVM were well discussed in [16,18–21]. To well describe the structure of the pattern, some researchers gave the definition of the structural granularity that can characterize the whole data [16]. According to the size of the granularity, there are four forms, i.e. global granularity, class granularity, cluster granularity and point (individual) granularity. This section gives a brief introduction about the structured classifiers in terms of different structural granularities.

Ellipsoidal kernel machine (EKM) [17] is a structural classifier based on global granularity. EKM exploits the ellipsoidal structure of the data through estimating the minimum volume bounding ellipsoid. In the linear case of EKM, the classifier with a given margin is shown to shatter fewer points than the previous estimated one. The non-linear of EKM is kernelized through scaling the general Hilbert spaces. The computational complexity of EKM is in polynomial time. The optimization of EKM is achieved through Semi-Definite Programming (SDP).

Both minimax probability machine (MPM) [18] and maxi-min margin machine (M^4) [19] are the classical instances of structural classifiers based on class granularity. To maximize the probability of correct classification of future data points, Lanckriet et al. [18] proposed MPM. Under all the possible choices of the class-conditional densities with a given mean and covariance matrix, MPM minimizes the worst probability of misclassification of future data points. For the decision boundary, the above model is translated into a convex second order cone optimization problem. The mini-max problem can be interpreted geometrically as minimizing the maximum of the Mahalanobis distances between two classes. Lanckriet et al. [18] further addressed the issue of the robustness with respect to the estimation errors through a simple modification of input data. MPM is supposed to consider data only either locally or globally. In contrast, M^4 is constructed based on both local and global views of data. Furthermore, the optimization of M^4 can be cast as a sequential conic programming problem, which can be solved more efficiently. M^4 provides a clear geometrical interpretation and presents a theoretical justification.

Both structured large margin machine (SLMM) [20] and structural regularized support vector machine (SRSVM) [16] are based on cluster granularity. SLMM is sensitive to the structure of the data distribution. It incorporates the merits of structured learning such as radial basis function networks and Gaussian mixture models. SLMM is derived from the concepts of structured degrees and bases on an analysis of the existing structured and unstructured learning models. Through using the Wards agglomerative hierarchical clustering technique to extract the underlying data structure, SLMM formulates the training processing as a sequential second order cone programming. The advantages of SLMM are validated in terms of accuracy, scalability, extensibility, and noise tolerance. M^4 is demonstrated to be a special case of SLMM. SRSVM [16] is located at the cross of the cluster granularity and aims to integrate the compactness within classes with the separability between classes. SRSVM is demonstrated to not only have a lower computational complexity but also hold the sparsity of the solution. Laplacian support vector machine (LapSVM) [21] is based on point granularity. LapSVM falls into the semi-supervised learning framework that incorporates both labeled and unlabeled data into a general purpose learner. LapSVM uses the properties of reproducing Hilbert space to set up a new representing theory that provides a theoretical basis for the proposed algorithm. In this paper, through introducing the structure of each class into the matrixized classifier discussed in Section 1, we develop a classifier design that owns the three-fold including class, cluster, and individual structural information.

2.2. Family of different Ho–Kashyap (HK) algorithms

Since the proposed method is an improved HK algorithm, this section gives the description on the series of the related Ho–Kashyap algorithms including the original HK [3], the Modification of HK algorithm with Squared approximation of the misclassification errors (MHKS) [14], and the matrixized MHKS (MatMHKS) [11].

2.2.1. HK

The HK algorithm is known as a simple linear classifier with a fast gradient descent optimization. Suppose that there is a binary-class classification problem with N samples (x_i,φ_i), $i=1\ldots N$, where $x_i\in\mathbb{R}^d$ and its class label $\varphi_i\in\{+1,-1\}$. If the binary classification problem is linearly separable, the decision function of the HK classifier should satisfy the following formulation:

$$g(x_i)=\tilde{w}^T x_i+w_0\begin{cases}>0 & \text{if } \varphi_i=+1\\ <0 & \text{if } \varphi_i=-1\end{cases} \tag{3}$$

where $\tilde{w}\in\mathbb{R}^d$ is a weight vector, and $w_0\in\mathbb{R}$ is a bias. Then the formulation (3) can be rewritten as the matrix form

$$Yw>0_{N\times 1} \tag{4}$$

where $Y=[y_1,\ldots,y_N]^T$, $y_i=\varphi_i[x_i^T,1]^T$, and $w=[\tilde{w}^T,w_0]\in\mathbb{R}^{d+1}$ are the augmented weight vector. To get the solution for the w, we give the criterion function of the HK algorithm as follows:

$$\min L(w,b)=\|Yw-b\|_2^2 \tag{5}$$

where $b\geq 0_{N\times 1}\in\mathbb{R}^N$ is the vector with each element greater than or equal to zero. To optimize (5), the HK algorithm adopts the modified gradient descent technique, and at each iteration k takes the derivative of the L with respect to w and b respectively as follows:

$$\frac{\partial L}{\partial w}=2Y^T(Yw-b)$$
$$\frac{\partial L}{\partial b}=-2(Yw-b) \tag{6}$$

At each iteration k, the augmented weight vector w_k is derived as

$$w_k=Y^{\dagger}b_k \tag{7}$$

where $Y^{\dagger}$ is the pseudo-inverse of Y, and the vector b_k is requested to satisfy $b_k\geq 0_{N\times 1}$, which is achieved through starting with $b_0\geq 0_{N\times 1}$ and preventing any of its components from decreasing. Thus the HK rule for the b can be given as follows:

$$\begin{cases}b_1\geq 0_{N\times 1}\\ b_{k+1}=b_k+\rho(e_k+|e_k|)\end{cases} \tag{8}$$

where the error vector $e_k=Yw_k-b_k$, and the learning rate $0<\rho<1$. In practice, we define a criterion $\|b_{k+1}-b_k\|_2\leq\xi$ for the termination.

2.2.2. MHKS

The HK algorithm is supposed to be sensitive to outliers [14]. In order to solve this problem, Leski proposed a modified HK algorithm named MHKS [14]. MHKS bases on the least square rule and maximizes the separating margin [22–24]. Differently from the separating hyperplane (4), MHKS replaces the $0_{N\times 1}$ with $1_{N\times 1}$ as follows:

$$Yw\geq 1_{N\times 1} \tag{9}$$

Consequently, the criterion function of MHKS is changed as

$$\min_{w\in\mathbb{R}^{d+1},b\geq 0} L(w,b)=(Yw-1_{N\times 1}-b)^T(Yw-1_{N\times 1}-b)+C\tilde{w}^T\tilde{w} \tag{10}$$

where $C\geq 0$ is the regularized hyperparameter that adjusts the tradeoff between the model complexity and the training error. The procedure of MHKS is almost the same as that of the HK classifier. The difference of MHKS from HK is that the argument weight vector w_k in MHKS model becomes

$$w_k=(Y^TY+C\tilde{I})^{-1}Y^T(b_k+1_{N\times 1}) \tag{11}$$

where $\tilde{I}$ is an identity matrix with the last element on the main diagonal set to zero. Both the iteration for b_k and the decision function in MHKS are the same as those in HK.

2.2.3. MatMHKS

Since the vector representation for patterns fails in some image-based learning, some matrix-based methods were proposed in feature extraction [25–27] and classifier design [11]. MatMHKS is a matrixized classifier and can directly classify

patterns represented with matrix. In the matrix case, suppose that there is a binary-class classification problem with N matrix patterns (A_i,φ_i), $i=1\ldots N$, where $A_i\in\mathbb{R}^{d_1\times d_2}$ and its class label $\varphi_i\in\{+1,-1\}$. The decision function of MatMHKS for the binary classification problem is given as

$$g(A_i)=u^TA_i\tilde{v}+v_0\begin{cases}>0 & \text{if } \varphi_i=+1\\ <0 & \text{if } \varphi_i=-1\end{cases} \tag{12}$$

where $u\in\mathbb{R}^{d_1}$ and $\tilde{v}\in\mathbb{R}^{d_2}$ are the weight vectors. The corresponding optimization function of MatMHKS is defined as follows:

$$\min_{u\in\mathbb{R}^{d_1},\tilde{v}\in\mathbb{R}^{d_2},v_0,b\geq 0} L(u,\tilde{v},v_0,b)=\sum_{i=1}^{N}(\varphi_i(u^TA_i\tilde{v}+v_0)-1-b_i)^2 +C(u^TS_1u+\tilde{v}^TS_2\tilde{v}) \tag{13}$$

where $S_1=d_1I_{d_1\times d_1}, S_2=d_2I_{d_2\times d_2}$ are two regularized matrices corresponding to the u and $\tilde{v}$ respectively, and the regularized parameter C ($C\in\mathbb{R}, C\geq 0$) controls the generalization ability of the classifier by making a tradeoff between the complexity of the classifier and the errors of training. The vectors u, $\tilde{v}$ and the bias v_0 are obtained by the gradient of the formulation (13) with respect to u, $\tilde{v}$ and v_0, respectively. The iteration for b_k is the same as that in MHKS. Here, we discuss the relationship between MHKS and MatMHKS in the terms of the pattern expression. We have given the following theorem as follows [10].

Theorem 1. *Let $A\in R^{m\times n}$, $B\in R^{n\times p}$ and $C\in R^{p\times q}$, then*

$$vec(ABC)=(C^T\otimes A)vec(B) \tag{14}$$

where $vec(X)$ denotes an operator that vectorizes the matrix X into the corresponding vector. For example, let $X=(x_{ij})\in R^{p\times q}$ and $x_i=(x_{1i},\ldots,x_{pi})^T$ be the ith column of X, and thus $vec(X)=(x_1^T,\ldots,x_i^T,\ldots,x_q^T)^T$ is a vector with $p\times q$ dimensionality. $\otimes$ denotes Kronecker product operation. According to Theorem 1, we can transform the discriminant function of MatMHKS (12) into the following form:

$$\begin{aligned}g(A_i)&=u^TA_i\tilde{v}+v_0=(\tilde{v}^T\otimes u^T)vec(A_i)+v_0\\&=(\tilde{v}\otimes u)^Tvec(A_i)+v_0\end{aligned} \tag{15}$$

Further compared with the discriminant function (3) of MHKS, it can be found that both MatMHKS and MHKS have the same form of the discriminant functions. $\tilde{v}\otimes u$ in (15) of MatMHKS plays the same role as the weight vector $\tilde{w}$ in MHKS. For the reason that the weight vector $\tilde{w}$ of MHKS does not always satisfy decomposability of the Kronecker product, we can find that the solution space for $\tilde{w}$ is including that of $u,\tilde{v}$. Moreover, we make $\tilde{v}\otimes u$, $t_0=v_0$, $t=[\tilde{t}^T,t_0]^T$, $Y=[y_1,y_2,\ldots,y_N]^T$, $y_i=\varphi_i[vec(A_i),1]^T$, $i=1\ldots N$, and then the criterion function (13) of MatMHKS can be reformulated as follows:

$$\min_{t\in R^{d+1},b\geq 0,t=\tilde{v}\otimes u} L(t,b)=(Yt-1_{N\times 1}-b)^T(Yt-1_{N\times 1}-b) +C(u^TS_1u+\tilde{v}^TS_2\tilde{v}) \tag{16}$$

Compared with the (10) of MHKS, we can find that the first terms of their right-handed sides have the same form, and the second terms are both the regularization terms. But the difference between MatMHKS and MHKS is that t in (16) should satisfy a decomposability constraint of the Kronecker product, but w in (10) does not have the constraint. Thus MatMHKS can be viewed as the MHKS imposed with Kronecker product decomposability constraint. It means that if we search for its optimal weight vector on the space of the objective function (16), MatMHKS is guided by some prior information such as the structural or local contextual information which is reflected in the representation of Kronecker production of the u and $\tilde{v}$, which is the reason why MatMHKS outperforms MHKS in terms of classification performance. It can also be found that MatMHKS can avoid overtraining due to both the introduction of structural information and the reduction of dimensionality for patterns, i.e. from the matrix pattern A with the dimensionality of $d_1\times d_2$ to Av or u^{TA} with the corresponding dimensionality of d_1 or d_2.

3. Three-fold structured matrixized classifier design

This section gives the description for the architecture of the proposed TSMHKS. The architecture of TSMHKS is made up of the two parts. The first one is to construct the cluster structure of each class through some unsupervised or supervised methods. The second one is to introduce the obtained cluster information into our previous matrixized classifier learning framework.

3.1. Detection of data cluster

There are a lot classical clustering methods such as the K-means clustering [28], the Agglomerative Hierarchical Clustering (AHC) [29] and fuzzy-based clustering [30]. The proposed TSMHKS adopts the K-means and AHC as the detection for the cluster structure.

3.1.1. K-means clustering

Hartigan et al. [28] gave a K-means clustering method to deal with unsupervised data. K-means clustering aims to partition N samples into k clusters, i.e. $S=\{S_1,S_2,\ldots,S_k\}$ in which each sample belongs to the cluster center with the nearest mean. In practice, given a set of samples $x_1,x_2,\ldots,x_n$, K-means clustering achieves the above aim through minimizing the within-cluster sum of squares as follows:

$$\arg\min\sum_{i=1}^{k}\sum_{x_j\in S_i}\|x_j-\mu_i\|^2 \tag{17}$$

where μ_i is the mean of points in S_i. It can be found that the initialization of the k is sensitive for K-means clustering. If the k is too small, some samples with different structures would be included in the same cluster. If the k is too large, the size of the clusters would be excessive, which might cause a large computation.

3.1.2. Agglomerative Hierarchical Clustering (AHC)

As a hierarchical clustering, the AHC [29] can effectively solve the selection for the size of clusters. Thus we here adopt the AHC as the method to grasp the structure of the given data. In AHC, one can initialize each point as a cluster and then calculate the distance between every two clusters. While more than one cluster remain, one should find the closest pair of clusters and merge the two clusters. Meanwhile, one should also update the distance between each pair of clusters. The distance between every two clusters is measured by Ward's linkage [31]. Ward's linkage between the clusters A and B is given as

$$W(A,B)=\frac{|A|\cdot|B|\|\mu_A-\mu_B\|^2}{|A|+|B|} \tag{18}$$

where μ_A and μ_B are the means of the clusters A, B, respectively. If the clusters A and B are merged into a new cluster D, Ward's linkage $W(E,D)$ between the E and D can be conveniently derived

from $W(A,E)$, $W(B,E)$, and $W(A,B)$ through the following equation [20]:

$$W(E,D)=\frac{(|A|+|E|)W(A,E)+(|B|+|E|)W(B,E)-|E|W(A,B)}{|A|+|B|+|E|} \tag{19}$$

In the initialization, Ward's linkage between the two patterns x_i and x_j is computed as

$$W(x_i,x_j)=\frac{\|x_i-x_j\|^2}{2} \tag{20}$$

Ward's linkage between the merged clusters increases as the size of clusters decreases. In our method, we determine the size of clusters by selecting the size corresponding to the knee point. The knee point is the point of the maximum curvature on the curve which shows the relationship between the size of clusters and the merge distance as Salvador and Chan [32] do.

3.2. Architecture of three-fold structured classifier (TSMHKS)

Suppose that there is a set of samples $T_{vec}=\{(x_1,\varphi_1),\ldots,(x_N,\varphi_N)\}$, where N is the sample number, $x_p\in\mathbb{R}^d$, and its corresponding class label $\varphi_p\in\{+1,-1\}$. The vector set T_{vec} can be matrixized into its corresponding matrix set $T_{mat}=\{(A_1,\varphi_1),\ldots,(A_N,\varphi_N)\}$ through some matrixized techniques [11], where $A_p\in\mathbb{R}^{d_1\times d_2}, d=d_1d_2$. The objective function of MatMHKS is given as Eq. (13). The first term of the right-handed side in Eq. (13) is to minimize the number of misclassified samples and to separate the binary-class samples as much as possible. This term considers the class label information and is viewed to use the class structure. The second term of the right-handed side in Eq. (13) is a regularization term which makes a tradeoff between the complexity of the classifier and the training errors. Meanwhile this term uses the individual structure of samples through the two weight vectors. Therefore, the objective function (13) of MatMHKS shows the two-fold structures with both the class and the individual aspects.

In this paper we introduce the cluster structure of each class into the matrixized classifier learning and develop a novel classifier design with three-fold structures. We first adopt the above discussed clustering techniques and give a regularization term including the cluster information as follows:

$$R_d=\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}u^T(A_{ij}-\overline{A_i})\tilde{v}\tilde{v}^T(A_{ij}-\overline{A_i})^Tu, \tag{21}$$

where A_{ij} is the jth sample in the ith cluster, $\overline{A_i}$ is the mean of the ith cluster, n_d is the size of the total clusters, and the n_i is the size of the samples belonging to the ith cluster. The advantage of introducing the R_d is given as follows. It is well-known that in LDA, the within-class scatter S_i of the ith class is defined by w_{iw}^{TS}, where $S_i=\sum_{x\in D_i}(x-m_i)(x-m_i)^T$, m_i is the mean of the ith class, D_i is the set of samples in the ith class, x is the sample of the ith class, and w is the transform vector. In LDA, we should make the S_i as small as possible. It means that the samples belonging to the same class should be fixed as tightly as possible, which has the same meaning as the R_d here. In the matrix case, we make the samples in one cluster as tight as possible, which is achieving through minimizing the R_d. R_d can keep the samples in the same cluster as tight as possible.

Further it should be stated that although the above discussed clustering methods such as the AHC is designed to deal with the vector sample set T_{vec}, we can still make the AHC work through A_p in the T_{mat} corresponds to x_p in the T_{vec}. The reason for doing so is that in the cluster granularity, we can regard the sample whatever it is vector or matrix as one point, and need not consider the represented form of the sample itself. Through adding the R_d into the objective function (13), we can get the following objective function:

$$\begin{aligned}\min J(u,\tilde{v},v_0,b_p)=&\frac{1}{2}\sum_{p=1}^{N}(\varphi_p(u^TA_p\tilde{v}+v_0)-1-b_p)^2\\&+\frac{C}{2}(u^TS_1u+\tilde{v}^TS_2\tilde{v})+\frac{\lambda}{2}R_d\end{aligned} \tag{22}$$

where both C and λ are the regularized parameters. The first term of the function (22) reflects the class structural information to minimize the classification error. The second one reflects the individual structural information. The third one reflects the cluster structural information through minimizing the close degrees of each cluster.

In order to optimize the objective function (22), we set $Y=[y_1,\ldots,y_N]^T$, $y_p=\varphi_p[u^TA_p,1]^T$, $p=1,\ldots,N$, $v=[\tilde{v}^T,v_0]^T$, and rewrite the function (22) as the following matrix form:

$$\begin{aligned}\min J(u,v,b_p)=&\frac{1}{2}(Yv-1-b)^T(Yv-1-b)\\&+\frac{C}{2}(u^TS_1u+v^T\tilde{S}_2v)+\frac{\lambda}{2}R_d\end{aligned} \tag{23}$$

where $1, b\in\mathbb{R}^N$ and $\tilde{S}_2=\left(\begin{smallmatrix}S_2 & 0\\ 0 & 0\end{smallmatrix}\right)$. From the functions (22) and (23), we cannot directly get the closed-form optimal weights. Instead we adopt the gradient descent technique to iteratively seek them. The gradients of the objective functions (22) and (23) with respect to u, v and b are given as follows:

$$\frac{\partial J}{\partial u}=uS_1+C\sum_{p=1}^{N}\varphi_pA_p\tilde{v}(\varphi_p(u^TA_p\tilde{v}+v_0)-1-b_p)+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}Pu \tag{24}$$

$$\frac{\partial J}{\partial v}=v\tilde{S}_2+CY^T(Yv-1-b)+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}\tilde{G}v \tag{25}$$

$$\frac{\partial J}{\partial b}=-C(Yv-1-b) \tag{26}$$

where

$$\tilde{G}=\begin{pmatrix}G & 0\\ 0 & 0\end{pmatrix},\quad G=(A_{ij}-\overline{A_i})^Tuu^T(A_{ij}-\overline{A_i}),\quad P=(A_{ij}-\overline{A_i})\tilde{v}\tilde{v}^T(A_{ij}-\overline{A_i})^T \tag{27}$$

Then by setting $\partial J/\partial u=0$ and $\partial J/\partial v=0$, we can get

$$u=\left[S_1+C\sum_{p=1}^{N}A_p\tilde{v}\tilde{v}^TA_p^T+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}P\right]^{-1}\left[C\sum_{p=1}^{N}\varphi_pA_p\tilde{v}(1+b_p-\varphi_pv_0)\right] \tag{28}$$

$$v=\left(\tilde{S}_2+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}\tilde{G}+CYY^T\right)^{-1}(CY^T(1+b)) \tag{29}$$

From Eqs. (28) and (29), it can be found that the weight vectors u and v are all determined by each other and the other b which is the margin vector whose components determine the distance of the corresponding sample to the separation hyperplane [16]. Moreover, since both u and v are also mutually dependent, we iteratively seek the u and v by initializing the b, which is similar to MatMHKS [11]. The design procedure for the proposed TSMHKS is shown in Table 1. In Table 1, the k denotes the iteration index, and the equation $b(k+1)=b(k)+\rho(e^{(k)}+|e(k)|)$ prevents the reduction of all the components of the b. From the procedure of the TSMHKS, it can be found that if the parameter λ is set to be zero, the TSMHKS is degenerated to MatMHKS. Further if d_1 and u are set to be 1 and omitting the iteration for the u, MatMHKS is degenerated to MHKS [14]. Moreover, if the

Table 1
Algorithm: TSMHKS.

Input: $S=\{(A_1,\varphi_1),\ldots,(A_N,\varphi_N)\}$;
Output: the weight vectors u, $\tilde{v}$, and the bias v_0;

1. Fix $C\geq 0$, $\lambda\geq 0$, $0<\rho<1$; initialize $b(1)\geq 0$ and $u(1)$, $v(1)$, set the iteration index $k=1$;
2. Let $Y=[y_1,y_2,\ldots,y_N]^T$, where $y_p=\varphi_p[u(k)^TA_p,1]^T, p=1,\ldots,N$;
3. Let $G(k)=(A_{ij}-\overline{A_i})^Tu(k)u(k)^T(A_{ij}-\overline{A_i})$,
   $P(k)=(A_{ij}-\overline{A_i})\tilde{v}(k)\tilde{v}(k)^T(A_{ij}-\overline{A_i})^T$,
   If $k=1$, then $v=v(1)$;
   else $v(k)=(\tilde{S}_2+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}\tilde{G}(k-1)+CYY^T)^{-1}(CY^T(1+b(k-1)))$, then $v=v(k)$;
4. Let $e(k)=Yv(k)-1-b(k)$;
6. Let $b(k+1)=b(k)+\rho(e(k)+|e(k)|)$;
7. If $\|b(k+1)-b(k)\|>\xi$, then $k=k+1$, go to Step 8; else stop;
8. Let $u(k+1)=[S_1+C\sum_{p=1}^{N}A_p\tilde{v}(k)\tilde{v}(k)^TA_p^T+\lambda\sum_{i=1}^{n_d}\sum_{j=1}^{n_i}P(k)]^{-1}[C\sum_{p=1}^{N}\varphi_pA_p\tilde{v}(k)(1+b_p(k)-\varphi_pv_0(k))]$,
   $k=k+1$, go to Step 2.

regularize parameter C is set to be 0, MHKS is degenerated to the original HK algorithm. Therefore, all the MatMHKS, MHKS, and HK can be viewed as one special instances of the TSMHKS. Finally, the discriminant function of the TSMHKS can be given as follows:

$$g(A)=u^TA\tilde{v}+v_0\begin{cases}>0, & A\in class+1\\ <0, & A\in class-1\end{cases} \tag{30}$$

4. Experiments

This section is validating the effectiveness of the proposed TSMHKS. Firstly, we give the experimental setting for all the implemented algorithms. Secondly, we compare and discuss the discriminant boundaries of TSMHKS with MHKS and MatMHKS on the synthetic data sets. Thirdly, we further analyze TSMHKS on some UCI Benchmark data sets [36] in terms of: (1) the size of clusters; (2) the matrix size of patterns; (3) the regularized parameters C and λ; (4) the convergence; (5) the generalization risk; (6) the classification and computational complexity, and (7) the comparison between TSMHKS and the related ensemble techniques. Finally, we also show the analysis for the performance of TSMHKS with the varying size of the image data.

4.1. Experimental setting

In our experiments, we have carried out the proposed TSMHKS, the vectorized and matrixized modifications of Ho–Kashyap algorithm named by MHKS and MatMHKS, the state-of-the-art SVM with linear, polynomial and RBF kernels. In the implementation of TSMHKS, the vector b is initialized by 10^{-6}. The learning rate ρ is set to 0.99. Both the regularized parameters C and λ are chosen from the same set $\{10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3}\}$. The termination variable ξ is fixed to 10^{-4}. In practice, we add another termination condition by a maximal size of the iteration $maxIter=500$ so as to avoid the over-fitting loop. In the experimental processing, we optimize the initialized weight vectors u and $\tilde{v}$ from the set $\{0.1,0.2,0.3,\ldots,0.9,1\}$ and find that their corresponding results are not sensitive to the initialization for the u or $\tilde{v}$. In both MHKS and MatMHKS, the initializations for the parameters b, ρ, C, ξ, u, and $\tilde{v}$ are given the same setting as those in TSMHKS. For the implemented SVM, the used kernels are the linear kernel $ker(x_i,x_j)=x_ix_j^T$, polynomial kernel $ker(x_i,x_j)=(x_ix_j^T+1)^d$ and the radial basis function (RBF) kernel $ker(x_i,x_j)=\exp(-\|x_i-x_j\|_2^2/\sigma^2)$. The kernel parameter d is set to 2. The kernel parameter σ is chosen from the same set like C and λ, i.e. $\{10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3}\}$. Here, we report SVM with the best result of the three kinds of kernels including the linear, polynomial, and RBF ones. All the computations are performed on Intel Core 2 processors with 2.66 GHz, 4 G RAM DDR3, Microsoft Windows XP, and MATLAB environment.

Table 2
The information description for the two groups of synthetical data denoted with ToyA and ToyB.

Class	Mean	Covariance matrix
ToyA-class I	[2.0, 0.1]	[1.8, 0.0; 0.0, 8.0]
	[9.1, 40]	[1.3, 0.0; 0.0, 9.0]
	[20, −0.1]	[1.7, 0.0; 0.0, 7.2]
ToyA-class II	[9.9, −40]	[1.3, 0.0; 0.0, 8.3]
	[2.0, −80]	[1.0, 0.0; 0.0, 7.8]
	[20, −81]	[1.3, 0.0; 0.0, 7.2]
ToyB-class I	[6.0, 20]	[4.3, 0.0; 0.0, 5.0]
	[5.0, 0.1]	[4.8, 0.0; 0.0, 5.0]
	[7.0, −20]	[4.7, 0.0; 0.0, 4.2]
ToyB-class II	[20, 22]	[4.3, 0.0; 0.0, 4.3]
	[21, −1.2]	[5.0, 0.0; 0.0, 4.8]
	[21, −21]	[4.3, 0.0; 0.0, 4.2]

4.2. Synthetical data

In this subsection, we compare TSMHKS with MatMHKS and MHKS on the synthetic data. MHKS is viewed to integrate the class structure through making the distances between all the patterns belonging to different classes as far as possible. MatMHKS is viewed to inherit the information of both class and individual structures. TSMHKS is viewed to integrate the cluster structure besides the class and individual structures. In order to explicitly validate the characteristic of TSMHKS, we generate two groups of synthetical data with different cluster structures. The first group denoted as ToyA is made up of two classes ('+' vs. '○'). Each class consists of three clusters which appear as a line distribution. Each cluster is generated under the Gaussian distribution, whose mean and covariance matrix are shown in Table 2. Here we sort each cluster into the two non-overlapping parts including the training set with the S samples and the testing set with the S samples, where S is set to 25, 50, 100 respectively. Fig. 1(a), (c), and (e) shows the discriminant boundaries of TSMHKS, MatMHKS, and MHKS in the testing set of ToyA with different S. The second group of the used synthetical data denoted as ToyB is generated in the similar way as ToyA. But the difference is that the three clusters of each class in ToyB appear as a triangle distribution. Fig. 1(b), (d), and (f) shows the discriminant boundaries of TSMHKS, MatMHKS, and MHKS in the testing set of ToyB with different S. From this figure, it can be found that TSMHKS

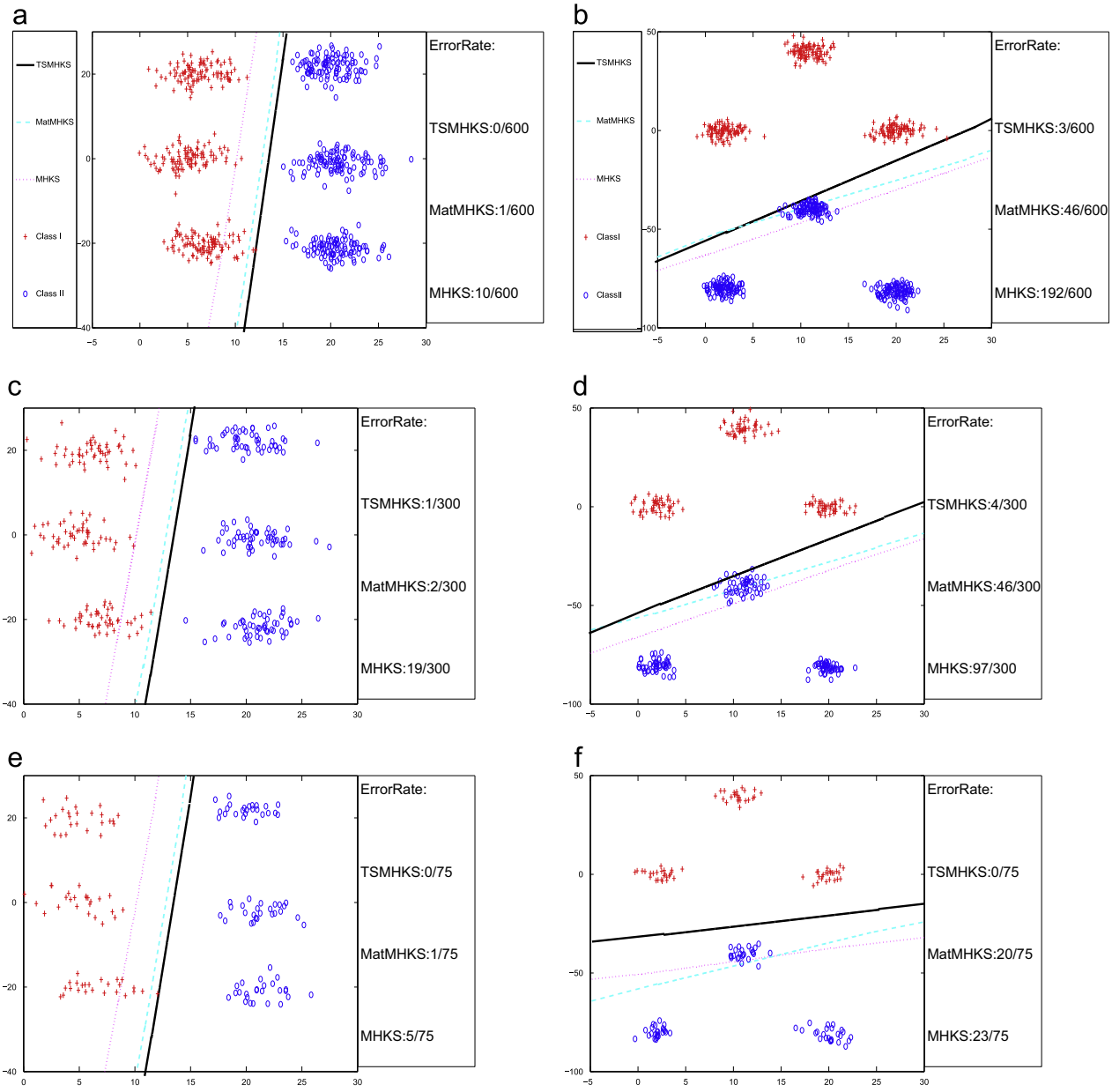

Fig. 1. The discriminant boundaries of TSMHKS, MatMHKS, and MHKS in the two groups of the synthetical data: (a), (c) and (e) give the discriminant boundaries in the testing set of ToyA; (b), (d) and (f) give the discriminant boundaries in the testing set of ToyB.

with different S's can have a more accurate discriminant boundary than both MatMHKS and MHKS. The advantage brought by TSMHKS can be attributed to the introduced information of the clusters. Further, we also give the error rates of TSMHKS, MatMHKS, and MHKS for the testing cases in Fig. 1. It shows that TSMHKS has a superior accuracy to both MatMHKS and MHKS.

Moreover, it can be found that although the used synthetical data has the dimensionality with two and cannot be matrixized into different sizes, MatMHKS still can have the two weights and the corresponding two regularization matrices as shown in (13). MHKS just owns only one weight and one regularization matrix $\tilde{w}^T\tilde{w}$. Therefore MatMHKS might be guided by some prior information which is reflected in the representation of Kronecker production of the two weights u and $\tilde{v}$, which induces that MatMHKS outperforms MHKS as shown in Fig. 1.

4.3. UCI data

In order to investigate the feasibility and the effectiveness of TSMHKS, we evaluate its performance in terms of the following seven aspects on some real-world data sets which are obtained from UCI repository of machine learning databases [36]. Table 3 gives the description for the used data sets, where the data sets Australian Card, Breast-Cancer-Wisconsin, German Data, House-Votes-84, Liver-disorders, Pima-Indians-Diabetes, and Post-Operative are denoted as AuC, BCW, GeD, HOV, Liver, PID, and POS for short respectively. The one-against-one classification strategy [33] is adopted for multi-class problems here. All the performances including the average classification accuracy and training time of the implemented algorithms here are reported through the strategy named Monte Carlo cross validation (MCCV) [34], i.e. randomly splitting the data set into the two parts

including the training and validation sets and repeating the procedure for T times. In our experiments, T is set to 10.

4.3.1. Influence of the size of clusters

As we state above, TSMHKS is made up of the two parts. The first one is constructing the cluster information in each class through one certain clustering technique. Here we adopt K-means clustering and AHC to generate the clusters for each class. Thus we discuss the influence of the size of the clusters generated by both K-means and AHC over the performance of the data sets from Table 3. Here the TSMHKS with K-means is named TSMHKS$_K$, and the TSMHKS with AHC is named TSMHKS$_A$. For the reason that the K-means method is sensitive for the choice of the initialized value of the k. If the k is too small, some samples with different structure would be included in the same cluster. If the k is too large, the size of the clusters would be excessive, which might cause a large computation. Moreover if we set the k too large, we have to put some strong-relative samples into different clusters, which would cause a poor result. In this case with a large k. Table 9 gives the average results of TSMHKS$_K$ and TSMHKS$_A$ with the varying size of the clusters in terms of classification accuracy and training time. Since AHC can optimize the size k of the clusters by the knee point [32], we give its corresponding result with the optimal size in Table 9.

From this table, it can be found that: (1) the TSMHKS$_A$ considers only the optimal size for k but the TSMHKS$_K$ should consider multiple cases so as to choose a better result. The result of TSMHKS$_K$ might not be the best one since we cannot give all the possible sizes for k. In TSMHKS$_K$, it might happen that if the k is small, some samples which belong to different classes would be enclosed into the same cluster and if the k is large, we would have some isolate samples. (2) TSMHKS$_A$ has a better performance than TSMHKS$_K$ in most of the cases since the adopted AHC can give the optimal k but the K-means one cannot. (3) Although for each k TSMHKS$_A$ has a comparable training time to TSMHKS$_K$, the K-means one in TSMHKS$_K$ should consider multiple cases with different k's. Thus it can be supposed that TSMHKS$_A$ takes less training time than TSMHKS$_K$. Taking AuC for example, TSMHKS$_A$ takes a 52.36 s for training while TSMHKS$_K$ uses 167.90 s in total so as to get the optimal recognition.

Table 3
The description for the used data sets.

Data set	# of features	# of class	# of training instances	# of testing instances
Australian Card (AuC)	14	2	460	230
Audiology	70	24	203	23
Bands	39	2	486	54
Breast-Cancer-Wisconsin (BCW)	9	2	466	233
Crx	15	2	620	69
German Data (GeD)	24	2	666	334
Glass	9	6	143	71
Heart	13	2	180	90
House-Votes-84 (HOV)	16	3	390	44
Ionosphere	33	2	234	117
Iris	3	3	75	75
Letter (UCI)	16	3	16 000	4000
Liver-disorders (Liver)	6	2	230	115
Mushroom	22	2	7311	813
Pendigits	16	10	7494	3498
Pima-Indians-Diabetes (PID)	8	2	512	256
Post-Operative (POS)	8	3	81	9
Satellite Image	36	10	4435	2000
Shuttle	9	7	43 500	14 500
Sonar	60	2	104	104
Thyroid	21	3	3772	3428
Waveform	21	3	2501	2499
Waveform-noise	40	3	2501	2499
Wine	13	3	118	60
Vowel	10	11	528	462

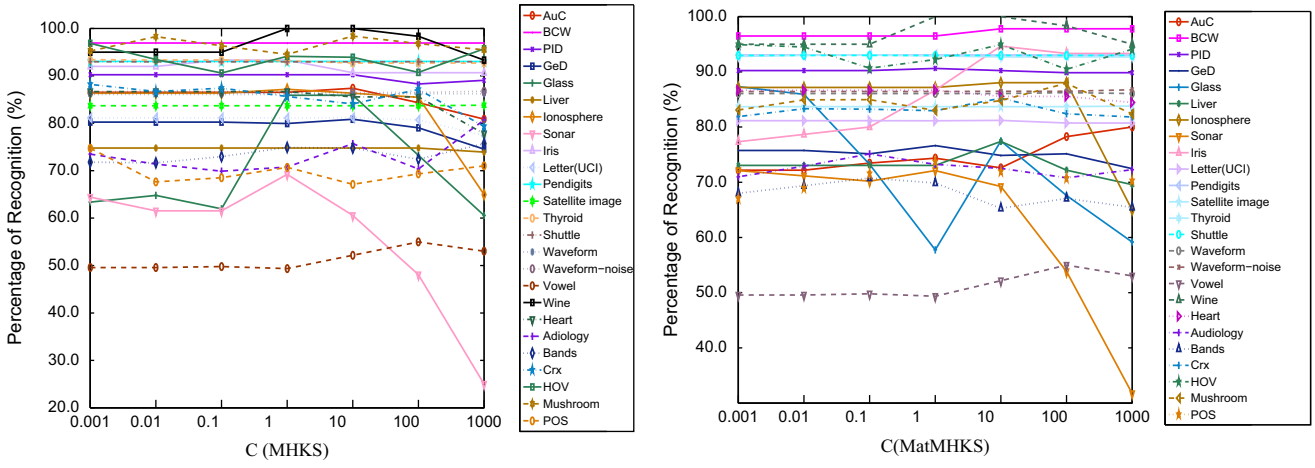

Fig. 2. Classification accuracies (%) of MHKS and MatMHKS with the varying C on the used data sets.

4.3.2. Influence of the size of matrix

Our previous work [11,10] has shown that for the same data set, different matrix sizes of one pattern would lead to different classification results for the matrixized classifiers. Since TSMHKS also falls into the matrixized classifier learning, we here discuss how the role of the matrix size plays. To generate different matrix sizes for each data set in Table 3, we adopt the similar matrixization technique in the literature [10,35]. The adopted matrixization way for each pattern x is reshaping without overlapping among the components of the x. It is that the original x is partitioned into multiple equal size sub-vectors which are then arranged column-by-column into the new matrix. For the same pattern, we can generate different matrix sizes by using different sub-vectors. Table 10 shows the classification performance of the matrixized classifiers TSMHKS including TSMHKS_A and TSMHKS_K, and MatMHKS on the data sets with different matrix sizes.

From Table 10, we can observe that compared with MatMHKS, TSMHKS has a better performance on almost all the used data sets. Since the difference between TSMHKS and MatMHKS is that TSMHKS is integrated by the cluster structure information but MatMHKS is not. Thus the improved performance of TSMHKS can be attributed to the introduced cluster information. It can be also found that TSMHKS_A takes the first or second place in terms of the classification. Further considering the large computational complexity for TSMHKS_K, we would adopt the AHC as the technique for generating the cluster information in the following experiments. Moreover, we can also conclude that to get the better result of TSMHKS, we should choose enough kinds of the matrix sizes of pattern and select the best one from different matrix sizes.

4.3.3. Influence of the regularized parameters C and λ

It is known that the regularized parameters C and λ play a balance role between the empirical and generalization risk. Thus both C and λ can influence the performance of the related algorithms. Here, TSMHKS, MatMHKS, and MHKS all adopt these parameters. The C is used in both MatMHKS and MHKS. TSMHKS uses both C and λ. To clearly show the role of the C and λ, we give the average classification accuracies of both MHKS and MatMHKS with the varying C on the used data sets in Fig. 2. From this figure, it can be found that when the C is initialized by 1 or 10, the best performance might be got on the used most data sets. For MatMHKS, the Glass attains the best recognition when the C is set

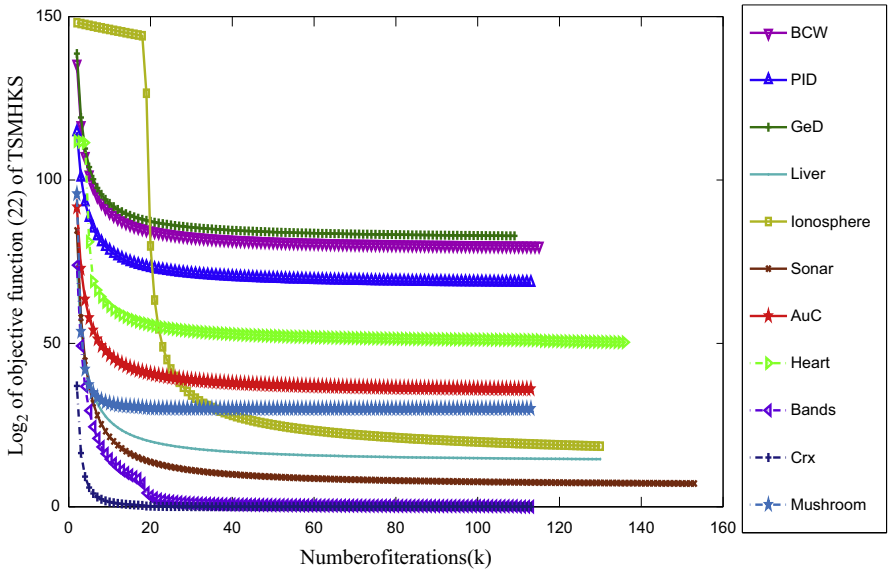

Fig. 3. The logarithm value of the objective function (22) of TSMHKS changes with the iteration number on the binary-class data sets including AuC, Bands, BCW, Crx, Heart, MUS, PID, GeD, Liver, Ionosphere, and Sonar respectively.

Table 4
Algorithm: computing the generalization bound.

Input: $S=\{(x_1,\varphi_1),\ldots,(x_N,\varphi_N)\}$.
Output: The generalization bound $R(h)$.

1. Fix $\theta=0.05$; Compute the maximum distance from the mean of samples to one sample in the training set and set the maximum distance to be B.
2. Compute the kernel matrix of training set and get its eigenvalues: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.
3. Let M be the number of eigenvalues, and n is the number of training samples; compute the $c(n,j)$ by Eq. (37), and get the largest one to be the c of the norm of weight w; compute the $\gamma(m)$ and $k(l)$ by Eqs. (35) and (36), respectively.
4. Set a certain margin γ between $8\gamma(m)<\gamma\leq 1$; Set $\gamma=\frac{8\gamma(m)+1}{2}$.
5. Give a hypothesis class $F(c)_B$; Start to train a classifier, and compute the number of the misclassified samples under the present classifier and set the number to be m.
6. Use Eq. (32) to compute the $R_s^{\gamma}(h)$.
7. Compute the result of $A_j=R_s^{\gamma}(h)+\sqrt{(m\ln 2+\ln(\lceil c\rceil/\theta\gamma)\lceil 8B/\gamma\rceil)/(2n)}$ under present classifier j.
8. Compute all the A_i (for $i=1,2,\ldots,L$), where L denote the Lth classifier by the hypothesis class $F(c)_B$.
9. Sort the result of A_i (for $i=1,2,\ldots,L$) and according to the degree of the confidence θ and Eq. (34), compute the generalization bounds $R(h)$.

by 0.001, but it attains the worst result with C initialized by 1. For Wine, the best performance for both MHKS and MatMHKS is got when C is initialized by 1 or 10. Thus, it can be concluded that the C plays an important role onto the performance of both MHKS and MatMHKS. The C should be optimized and here we adopt the MCCV technique [34] stated above. For TSMHKS, there are two used regularized parameters C and λ. Therefore, we give the classification accuracies of the varying λ with some certain fixed C so as to show the roles of both C and λ. Fig. 7 gives their corresponding results, where the regularized parameters C and λ are selected from the same set $\{10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3}\}$. From this figure, it can be found that both C and λ have a great influence on the classification performance. It is known that for the learning framework of TSMHKS, the function (22) has three terms. The first one uses the class discriminant information to minimize the classification error. The second one reflects the individual structural information. The third one reflects the cluster structural information through minimizing the close degrees of each cluster. The C and λ decide the importance of the regularization terms $(u^{T}S_1u+\tilde{v}^{T}S_2\tilde{v})$ and R_d, respectively. The large the parameter is, the importance the corresponding regularization term is.

4.3.4. Convergence analysis for TSMHKS

In this section, we give an experimental discussion about the convergence of TSMHKS. We adopt an empirical justification as used in [37] so as to demonstrate that TSMHKS can converge in the limited iterations. Fig. 3 shows that the logarithm value of the objective function (22) of TSMHKS changes with the iteration number on the binary-class data sets including AuC, Bands, BCW, Crx, Heart, MUS, PID, GeD, Liver, Ionosphere, and Sonar respectively. From this figure, it can be found that the target (22) value on the used data sets can quickly converge to stable values, where less than 40 iterations is usually enough to achieve convergence.

4.3.5. Generalization risk analysis for TSMHKS

It is known that the analysis of the generalization risk bound is important for interpreting the performance behavior of a learning algorithm. Here, we give the discussion of the generalization risk bound for TSMHKS, MatMHKS, and MHKS through the empirical covering number in terms of the distribution of the eigenvalues of the kernel matrix [38].

Supposing that the pattern $(x,\varphi)\in Z$ follows a certain distribution $P(x,\varphi)$, the generalization risk of a hypothesis $h\in F$ can be given by [39]

$$R(h)=\sum_{(x,\varphi)\in Z}\delta(\varphi h(x)\le 0)P(x,\varphi) \tag{31}$$

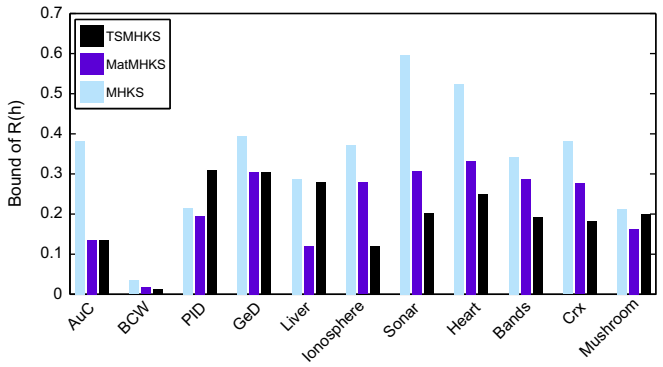

Fig. 4. Empirical bounds of the generalization risks for TSMHKS, MatMHKS, and MHKS on the binary-class data sets including AuC, BCW, PID, GeD, Liver, Ionosphere, Sonar, Heart, Bands, Crx, and Mushroom.

If the training set $\{x_i,\varphi_i\}_{i=1}^{n}$ distributed from Z is given, the empirical margin risk for a certain margin γ can be defined by the rate of the samples with [39]

$$\varphi_i h(x_i)<\gamma : R_s^{\gamma}(h)=(1/n)\sum_{i=1}^{n}\delta(\varphi_i h(x_i)<\gamma) \tag{32}$$

Then the following derivation gives an upper bound for the generalization risk of one learning algorithm [38,39]. Let $\lambda_1\ge\lambda_2\ge\cdots\ge\lambda_n$ be the eigenvalues of the kernel matrix derived

Table 5
Classification accuracy (%) comparison between TSMHKS, MatMHKS, MHKS, and SVM (The best accuracy for each data set is denoted by bold and the second one by italic.).

Data set	TSMHKS	MatMHKS	MHKS	SVM
AuC	**89.57**	*87.39*	80.00	84.78
Audiology	**74.95**	73.40	72.73	73.12
Bands	**81.61**	67.96	73.33	70.72
BCW	**98.71**	97.85	97.00	*98.71*
Crx	**86.46**	82.57	85.93	85.57
GeD	79.04	76.65	**80.84**	*79.34*
Glass	**94.37**	*87.32*	85.92	74.65
Heart	*86.67*	86.67	86.67	**88.90**
HOV	**93.30**	93.25	93.13	91.26
Ionosphere	**93.16**	88.03	87.18	*92.31*
Iris	**97.33**	94.67	93.33	*97.33*
Letter	**94.68**	81.20	81.20	*94.65*
Liver	**81.74**	*77.39*	74.78	75.65
Mushroom	94.71	85.19	95.57	**99.96**
Pendigits	*93.05*	93.05	93.05	**96.91**
PID	**92.58**	*90.63*	90.23	77.73
POS	**78.09**	71.11	70.00	77.12
Satellite image	**83.70**	*83.70*	83.70	83.65
Shuttle	*92.97*	92.97	**92.98**	85.95
Sonar	**82.69**	72.12	69.23	*76.92*
Thyroid	**93.60**	93.03	*93.41*	92.13
Vowel	**57.35**	*54.98*	54.98	49.59
Waveform	**86.27**	*86.23*	86.23	80.91
Waveform-noise	**86.75**	*86.47*	86.43	86.37
Wine	**100.00**	**100.00**	**100.00**	96.67

Table 6
The average training time (in seconds) comparison between TSMHKS, MatMHKS, MHKS, and SVM.

Data set	TSMHKS	MatMHKS	MHKS	SVM
AuC	52.36	19.90	23.32	7.16
Audiology	43.69	21.21	13.91	0.13
Bands	44.63	28.12	17.58	0.07
BCW	32.78	18.91	13.26	7.75
Crx	88.61	58.32	49.12	0.31
GeD	57.30	35.54	33.60	17.46
Glass	70.18	37.85	41.19	2.43
Heart	9.71	9.79	9.34	3.20
HOV	97.79	41.93	21.01	0.05
Ionosphere	356.48	9.87	15.81	1.55
Iris	15.00	8.60	8.26	0.30
Letter	20 202.01	21 219.37	26 252.85	24 388.83
Liver	39.78	9.94	8.19	1.68
Mushroom	216.47	126.02	58.01	3.01
Pendigits	5763.78	5732.23	4012.34	84 822.63
PID	31.70	23.09	17.43	8.84
POS	97.79	8.78	2.47	0.91
Satellite image	1435.12	1483.45	2303.76	18 123.36
Shuttle	6853.00	6821.21	6796.60	16 066.39
Sonar	23.74	2.91	10.42	0.58
Thyroid	247.42	237.47	575.82	13 253.17
Vowel	304.65	332.18	358.52	1691.56
Waveform	176.60	203.63	277.73	4014.23
Waveform-noise	320.69	328.19	897.68	3812.33
Wine	14.16	14.11	16.33	46.30

from the training samples. For the hypothesis class

$$F(c)_B = \{\langle w,x\rangle + b : \|w\| \le c, |b| \le B\} \tag{33}$$

we have the following inequality holding simultaneously for all $\gamma, 8\gamma(m) < \gamma \le 1$:

$$P_{s\in Z^n}(\exists h \in F(c)_B : R(h) \ge R_s^r(h) + \sqrt{(m\ln 2 + \ln(\lceil c\rceil/\theta\gamma)\lceil 8B/\gamma\rceil)/(2n)}) \le \theta \tag{34}$$

where

$$\gamma(m) = \min_{j\in\{1,2,\ldots,m-1\}} 6\times 2^{(1-j)/k(2^{j-1})}(\lambda_1\ldots\lambda_{k(2^{j-1})})^{1/2k(2^{j-1})}c(m,j) \tag{35}$$

$$k(l) = \min\{k\in\{1,\ldots,n\} : \lambda_{k+1} \le (\lambda_1\ldots\lambda_k/l^2)^{1/k}\} \tag{36}$$

$$c(m,j) = \min(1, 1.86\sqrt{\log_2(n/(m-j)+1)/(m-j)}) \tag{37}$$

Table 7
Classification accuracy (%) comparison between TSMHKS and the different ensemble techniques including "AHC+SVDD", "K_{means}+SVDD", "K_{means}+Adaboost", "K_{means}+Bagging", "K_{means}+Voting", "AHC+Adaboost", "AHC+Bagging", and "AHC+Voting", which can be shortly denoted as "A+S", "K+S", "K+A", "K+B", "K+V", "A+A", "A+B", and "A+V" respectively. (The best accuracy for each data set is denoted by bold and the second one by italic.)

Data set	TSMHKS	A+S	K+S	K+A	K+B	K+V	A+A	A+B	A+V
AuC	89.57	60.49	60.43	78.65	70.96	79.83	**90.07**	86.37	*90.07*
Audiology	74.95	74.95	73.63	71.06	71.01	71.72	**76.91**	73.76	*76.21*
Bands	*81.61*	**81.64**	81.11	68.54	68.26	68.39	70.92	70.26	70.74
BCW	98.71	98.84	98.67	98.68	95.70	98.71	**100.00**	95.31	*99.03*
Crx	86.46	86.46	85.77	82.60	82.42	83.11	**87.45**	85.38	*87.21*
GeD	79.04	77.36	76.65	73.35	70.96	74.80	**79.52**	79.26	*79.26*
Glass	94.37	95.60	95.21	88.57	85.18	92.96	**95.75**	94.51	*95.65*
Heart	86.67	*87.80*	86.67	80.64	69.12	86.77	**88.13**	87.13	87.13
HOV	92.10	72.25	72.10	90.36	89.48	90.16	*93.02*	92.96	**93.30**
Ionosphere	93.16	89.63	88.89	88.48	77.27	95.46	92.64	**93.92**	*93.92*
Iris	97.33	97.67	97.67	96.32	84.00	97.33	**99.38**	87.14	*98.32*
Letter	94.68	**96.23**	95.10	70.47	68.98	72.22	*95.13*	95.13	95.13
Liver	81.74	61.02	60.87	60.87	58.26	63.21	**84.07**	79.73	*81.81*
Mushroom	**94.71**	86.81	85.75	*94.71*	91.09	82.62	85.42	84.86	85.30
Pendigits	93.05	*95.06*	93.86	93.02	93.01	93.08	**95.59**	92.51	93.27
PID	92.58	77.88	77.34	73.23	68.75	78.13	**93.43**	92.68	*93.03*
POS	*78.09*	**78.09**	76.67	68.74	68.73	68.80	71.08	71.00	71.70
Satellite image	83.70	**84.33**	83.83	84.55	80.75	82.82	*83.92*	83.92	83.92
Shuttle	92.97	95.20	93.70	92.94	92.96	93.01	**95.55**	95.17	*95.32*
Sonar	82.69	85.17	83.65	84.62	58.65	70.87	**87.72**	83.11	*85.51*
Thyroid	93.60	*95.35*	93.66	93.00	92.59	93.23	**95.88**	94.27	95.13
Vowel	57.35	**59.24**	*58.25*	45.45	38.31	51.30	57.88	52.51	57.88
Waveform	86.27	**87.03**	*87.02*	85.71	85.74	85.71	86.91	86.91	86.91
Waveform-noise	86.75	88.28	87.26	86.75	86.43	86.12	**88.81**	86.04	*88.32*
Wine	**100.00**	*100.00*	100.00	96.59	90.91	100.00	100.00	100.00	100.00

Table 8
The average training time (in seconds) comparison between TSMHKS and the different ensemble techniques including "AHC+SVDD", "K_{means}+SVDD", "K_{means}+Adaboost", "K_{means}+Bagging", "K_{means}+Voting", "AHC+Adaboost", "AHC+Bagging", and "AHC+Voting", which can be shortly denoted as "A+S", "K+S", "K+A", "K+B", "K+V", "A+A", "A+B", and "A+V" respectively.

Data set	TSMHKS	A+S	K+S	K+A	K+B	K+V	A+A	A+B	A+V
AuC	52.36	47.47	43.35	562.82	399.45	545.22	1078.98	546.72	997.34
Audiology	43.69	42.64	36.98	675.00	371.44	399.90	733.45	718.07	439.43
Bands	44.63	43.81	36.40	503.83	274.06	702.85	830.24	547.19	1319.41
BCW	32.78	31.27	30.70	386.21	280.98	563.39	433.00	508.03	935.81
Crx	88.61	87.17	84.19	1572.20	744.04	1117.83	2830.56	1405.64	2064.87
GeD	57.30	52.41	44.80	660.96	464.67	811.46	939.56	788.58	1539.24
Glass	70.18	65.45	61.76	1018.79	612.78	877.02	1147.09	1152.27	1181.30
Heart	9.71	8.84	8.83	188.46	59.47	96.48	199.65	83.34	191.69
HOV	97.79	92.01	74.43	1425.88	850.47	982.39	2835.13	1027.92	1742.94
Ionosphere	356.48	322.27	270.50	6700.47	2186.21	4871.91	12 216.95	3770.60	4983.93
Iris	15.00	13.56	11.09	177.94	98.15	121.66	312.02	181.10	164.23
Letter	20 202.01	18 421.99	15 770.63	27 2416.80	143 392.92	203 767.58	489 995.80	193 247.50	374 510.61
Liver	39.78	39.52	34.41	488.48	237.32	581.13	543.78	428.85	1161.60
Mushroom	216.47	200.43	167.36	3733.04	1383.47	2243.03	6693.92	1701.81	2835.20
Pendigits	5763.78	5248.22	4211.92	108 307.35	48 653.00	59 441.97	157 476.53	92 748.26	101 041.93
PID	31.70	29.88	25.10	520.13	273.78	380.48	567.75	326.82	711.84
POS	97.79	97.36	93.27	1548.38	570.01	1462.25	1648.30	629.61	1978.77
Satellite image	1435.12	1418.49	1301.81	24 021.18	10 920.90	14 533.72	32 327.88	15 880.05	25 395.98
Shuttle	6853.00	6544.09	6020.96	110 312.30	64 127.49	117 515.71	114 515.18	92 426.66	134 171.67
Sonar	23.74	23.11	19.35	270.51	151.89	335.79	482.53	187.33	662.14
Thyroid	247.42	229.41	202.65	2998.18	2118.62	3824.79	4115.80	3044.55	3844.40
Vowel	304.65	287.64	261.44	5195.19	1729.95	4066.91	7374.99	2619.15	6589.91
Waveform	176.60	171.62	156.36	3395.88	1166.83	2485.25	5838.91	1177.66	4653.92
Waveform-noise	320.69	311.16	295.04	6091.58	2558.01	4126.66	9456.96	2947.02	5001.18
Wine	14.16	14.07	11.83	243.99	77.45	213.29	383.88	127.81	424.81

Table 9
The performance comparison between $TSMHKS_A$ and $TSMHKS_K$ in terms of the average classification accuracy (%) and training time (seconds), where the best accuracy for each data set is denoted by bold and the second one by italic.

Data set	Size of cluster		Percentage of recognition		Training time	
	$TSMHKS_K$	$TSMHKS_A$	$TSMHKS_K$	$TSMHKS_A$	$TSMHKS_K$	$TSMHKS_A$
AuC	15	45	78.65	**85.65**	25.47	52.36
	18		79.52		27.93	
	21		70.96		35.34	
	24		*78.83*		33.11	
	27		79.83		46.05	
Audiology	24	67	69.88	**75.41**	12.76	41.36
	48		72.01		19.21	
	72		*73.14*		39.21	
Bands	2	12	67.21	**73.03**	21.11	44.63
	3		66.28		25.47	
	4		*70.03*		29.31	
	5		65.56		32.98	
	6		68.38		38.03	
BCW	2	34	**98.71**	98.25	33.69	32.78
	3		*98.71*		37.57	
	4		98.68		40.38	
	5		98.70		49.70	
	6		95.7		65.13	
Crx	2	19	79.97	**88.31**	48.11	88.61
	3		83.96		52.12	
	4		85.40		58.31	
	5		84.67		61.36	
	6		*85.43*		79.31	
GeD	2	55	72.16	**78.44**	31.17	57.30
	3		*73.35*		40.84	
	4		71.86		53.76	
	5		71.86		61.22	
	6		70.96		79.47	
Glass	6	35	85.18	**92.96**	48.42	70.18
	7		*88.73*		60.29	
	8		85.18		72.03	
	9		88.57		72.05	
	10		88.68		84.18	
	11		85.18		103.52	
	12		88.72		130.80	
	13		85.18		141.78	
	14		88.56		159.00	
	15		88.70		169.01	
	16		88.75		188.06	
	17		88.73		225.62	
	18		88.21		245.54	
Heart	2	19	*86.67*	**87.78**	8.92	9.71
	3		71.11		8.95	
	4		62.22		8.93	
	5		61.11		8.90	
	6		60.00		8.90	
HOV	3	10	85.09	**98.64**	49.12	97.79
	4		89.84		51.10	
	5		90.25		53.29	
	6		93.65		57.50	
	7		*93.70*		59.87	
	8		93.00		62.78	
	9		90.52		69.35	
Ionosphere	2	28	*88.89*	**92.31**	237.01	356.48
	3		83.95		301.60	
	4		78.02		337.13	
	5		74.09		363.42	
	6		74.07		363.42	
Iris	3	10	96.11	**97.33**	14.49	15.00
	4		96.32		14.82	
	5		87.31		13.06	
	6		89.23		14.18	
	7		*97.33*		13.83	
	8		97.33		15.91	
	9		84.00		13.25	

Table 9 (*continued*)

Data set	Size of cluster		Percentage of recognition		Training time	
	TSMHKS$_K$	TSMHKS$_A$	TSMHKS$_K$	TSMHKS$_A$	TSMHKS$_K$	TSMHKS$_A$
Letter	26	1567	*94.55*	**94.68**	20 214.45	20202.01
	35		94.52		20 214.52	
	44		94.51		20 214.55	
	53		93.55		20 214.54	
	62		94.32		20 214.63	
	70		94.11		20 214.56	
	78		94.52		20 214.53	
Liver	2	40	58.26	**81.74**	20.28	39.78
	3		58.26		35.59	
	4		57.39		34.62	
	5		*60.87*		46.17	
	6		60.87		59.86	
Mushroom	2	42	95.23	87.77	87.31	213.11
	3		95.13		93.21	
	4		**96.48**		118.12	
	5		92.87		131.89	
	6		*95.56*		179.32	
Pendigits	10	367	93.01	93.05	5775.44	5763.78
	13		93.02		5775.51	
	17		**93.08**		5775.54	
	20		93.02		5775.53	
	23		*93.08*		5775.62	
	27		93.05		5775.55	
	30		93.02		5775.48	
PID	2	67	75.38	**91.41**	34.47	31.70
	3		75.82		40.05	
	4		75.83		41.84	
	5		75.85		44.94	
	6		*87.50*		51.33	
POS	3	21	69.64	**74.50**	29.03	97.79
	4		69.94		31.09	
	5		69.61		33.11	
	6		69.93		41.21	
	7		*71.06*		42.32	
	8		67.92		51.11	
	9		65.92		57.10	
Satellite image	6	137	*83.70*	**83.70**	1468.95	1435.12
	7		83.65		1465.35	
	8		83.45		1465.13	
	9		83.47		1459.86	
	10		83.65		1459.41	
	11		83.45		1461.46	
	12		83.45		1472.14	
	13		83.49		1459.65	
	14		83.12		1506.95	
	15		82.32		1462.41	
	16		82.98		1459.44	
	17		83.32		1465.49	
	18		82.20		1461.60	
Shuttle	7	5488	93.00	92.97	6886.95	6853.00
	9		*93.01*		6883.35	
	10		92.94		6883.13	
	11		92.96		6877.26	
	13		**93.01**		6877.41	
	15		92.96		6879.46	
	16		92.90		6890.14	
	18		92.90		6877.65	
	20		92.83		6924.95	
	21		92.81		6877.82	
Sonar	2	15	*78.85*	**81.73**	7.37	23.74
	3		78.82		14.72	
	4		78.83		39.20	
	5		78.55		53.84	
	6		78.18		54.22	
Thyroid	3	1770	*93.23*	**93.37**	243.95	247.21
	4		93.16		240.35	
	5		93.00		236.02	
	6		92.59		234.30	
	7		93.20		234.41	

Table 9 (*continued*)

Data set	Size of cluster		Percentage of recognition		Training time	
	TSMHKS$_K$	TSMHKS$_A$	TSMHKS$_K$	TSMHKS$_A$	TSMHKS$_K$	TSMHKS$_A$
Vowel	11	70	*55.19*	**57.58**	343.06	304.65
	18		55.19		361.96	
	22		54.98		349.92	
	28		55.19		343.78	
	33		54.98		343.57	
Waveform	3	166	**86.47**	86.27	173.95	176.60
	4		86.43		170.35	
	5		86.15		170.13	
	6		86.15		164.26	
	7		*86.43*		164.41	
	8		86.15		166.46	
	9		85.74		177.14	
	10		85.71		164.65	
	11		85.38		211.95	
	12		85.32		164.82	
Waveform-noise	3	100	**86.75**	86.47	343.95	320.69
	4		*86.71*		340.35	
	5		86.43		340.13	
	6		86.43		334.26	
	7		86.71		334.41	
	8		86.43		336.46	
	9		86.12		347.14	
	10		86.32		334.65	
	11		85.12		381.95	
	12		85.67		334.82	
Wine	3	12	*100.00*	**100.00**	16.06	14.16
	4		100.00		34.96	
	5		98.33		66.92	
	6		100.00		16.78	
	7		98.33		15.77	

Since TSMHKS, MatMHKS, and MHKS all belong to the linear algorithms, the kernel matrix is got through introducing the linear kernel. But in Eq. (33), its hypothesis falls into the vectorized function. The matrixized TSMHKS and MatMHKS cannot be directly applied into the above equation. Therefore, we should revise the decision function (30) for TSMHKS and MatMHKS through the technique in the literature [10]. As showed in [10], we let $A \in R^{m\times n}$, $B \in R^{n\times p}$, and $C \in R^{p\times q}$, then

$$vec(ABC) = (C^T \otimes A)vec(B) \tag{38}$$

where $vec(X)$ denotes an operator that vectorizes the matrix X into the corresponding vector. For example, let $X = (x_{ij}) \in R^{p\times q}$ and $x_i = (x_{1i}, \ldots, x_{pi})^T$ be the ith column of X, and thus $vec(X) = (x_1^T, \ldots, x_i^T, \ldots, x_q^T)^T$ is a vector with $p \times q$ dimensionality. $\otimes$ denotes the *Kronecker* product operation. Through this conversion, we can convert the matrixized decision function to the vectorized one.

By the above generalization risk analysis [38], we give the algorithm that gives how to compute the generalization bound for one data set in Table 4. Here we select 11 binary-class data sets shown in Table 3 to estimate the generalization bounds for the TSMHKS, MatMHKS, and MHKS. The corresponding results are shown in Fig. 4. From this figure, it can be found that the bounds of TSMHKS are smaller than those of both MHKS and MatMHKS on the seven data sets AuC, Bands, Crx, BCW, GeD, Ionosphere, and Sonar. However, due to more free regularization parameters involved than the other algorithms and the insufficient samples characterizing the data manifold structure, TSMHKS yields an unstable capability and its bounds achieve even the largest ones on PID. Moreover, since MatMHKS is the MHKS imposed with Kronecker product decomposability constraint and guided by some prior information which is reflected in the representation of Kronecker production of the u and $\tilde{v}$, MatMHKS can outperform MHKS in terms of classification performance MHKS especially for Sonar, which accords with the generalization risk analysis.

4.3.6. Performance comparison between TSMHKS, MatMHKS, MHKS, and SVM

In this section, we give a comparison between TSMHKS, MatMHKS, MHKS, and SVM in terms of classification and training time. Here we give the best results of SVM from the used three kinds of kernels including the linear, polynomial, and RBF kernels. We report the average classification accuracies and training time of the TSMHKS, MatMHKS, MHKS, and SVM through the MCCV [34] in Tables 5 and 6, where we give the best accuracy for each data set denoted by bold and the second one by italic. From the two tables, it can be found that: (1) the proposed TSMHKS attains a superior classification accuracy to the other three methods in most used data sets due to its more consideration of data distribution within and between classes, though TSMHKS is a linear algorithm; (2) the TSMHKS has a comparable quantity to the other algorithms in terms of training time cost, though it would need time to implement the cluster processing.

4.3.7. Performance comparison between TSMHKS and ensemble algorithms

In this subsection, we give the performance comparison between TSMHKS and the designed two kinds of ensemble strategies. The first one is showed as follows. Instead of adding a regularization term for the cluster information, we give an ensemble of classifiers, where each classifier is trained by using only one cluster of a given class. Thus we use some patterns of one certain class to train a certain classifier. Support vector data description (SVDD) [40] can construct a hypersphere which can enclose as many target objects as possible, which can minimize

Table 10
Classification accuracy (%) comparison between $TSMHKS_A$, $TSMHKS_K$, and MATMHKS. (The best accuracy for each data set is denoted by bold and the second one by italic. The matrix size of each data set is written in the bracket.)

Data set	Matrix size	$TSMHKS_A$	$TSMHKS_K$	MatMHKS
AuC	(2 × 7)	**89.57**	58.26	80.00
	(7 × 2)	*85.22*	57.83	74.35
Audiology	(2 × 35)	*75.66*	74.09	70.99
	(5 × 14)	75.53	72.01	72.90
	(14 × 5)	**76.41**	73.88	75.12
	(35 × 2)	75.08	72.62	73.24
Bands	(3 × 13)	68.40	68.68	68.00
	(13 × 3)	**72.64**	70.22	*70.77*
BCW	(3 × 3)	**98.71**	*98.25*	97.81
Crx	(3 × 5)	**88.49**	85.16	83.32
	(5 × 3)	*88.40*	85.15	81.82
GeD	(2 × 12)	71.56	71.56	71.56
	(3 × 8)	**79.04**	73.35	*76.65*
	(4 × 6)	72.46	72.16	72.46
	(6 × 4)	74.25	71.86	73.95
	(8 × 3)	71.86	71.86	71.86
	(12 × 2)	70.96	71.86	70.96
Glass	(3 × 3)	**94.37**	*88.73*	87.32
Heart	(1 × 13)	**86.67**	*86.67*	86.67
	(13 × 1)	64.44	72.22	62.22
HOV	(2 × 8)	**99.60**	93.73	94.92
	(4 × 4)	94.07	93.00	90.65
	(8 × 2)	*95.02*	93.09	92.24
Ionosphere	(3 × 11)	*90.60*	86.32	88.03
	(11 × 3)	**97.44**	88.89	87.18
Iris	(2 × 2)	*94.67*	**96.00**	94.67
Letter	(1 × 16)	94.68	94.55	81.20
	(2 × 8)	89.70	89.33	63.73
	(4 × 4)	83.50	83.13	39.05
	(8 × 2)	67.30	66.93	20.88
	(16 × 1)	54.43	54.05	8.88
Liver	(2 × 3)	77.39	77.39	75.65
	(3 × 2)	80.00	80.00	77.39
Mushroom	(2 × 11)	**98.83**	94.01	87.58
	(11 × 2)	85.48	*94.81*	87.74
Pendigits	(1 × 16)	**93.05**	*92.11*	93.05
	(2 × 8)	88.22	93.08	85.82
	(4 × 4)	78.53	91.21	76.24
	(8 × 2)	53.89	92.31	44.97
	(16 × 1)	35.53	87.12	23.30
PID	(2 × 4)	92.58	**99.22**	90.63
	(4 × 2)	92.19	*97.66*	90.23
POS	(2 × 4)	**74.50**	71.63	71.93
	(4 × 2)	*73.43*	67.92	70.73
Satellite image	(1 × 36)	**83.70**	*83.70*	83.70
	(2 × 18)	79.50	78.12	74.35
	(3 × 12)	81.15	77.11	81.30
	(4 × 9)	46.20	78.31	43.75
	(6 × 6)	75.00	80.12	66.00
	(9 × 4)	79.05	81.11	80.10
	(12 × 3)	45.90	79.31	43.70
	(18 × 2)	70.10	67.32	62.25
	(36 × 1)	45.70	78.12	43.00
Shuttle	(1 × 9)	*92.97*	**93.01**	92.97
	(3 × 3)	85.95	87.11	85.22
	(9 × 1)	80.05	89.32	79.26
Sonar	(2 × 30)	79.81	61.54	70.19
	(3 × 20)	**82.69**	62.50	72.12
	(4 × 15)	*81.73*	65.38	72.12
	(5 × 12)	77.88	67.31	66.35
	(6 × 10)	80.77	65.38	68.27
	(10 × 6)	76.92	73.08	67.31
	(12 × 5)	77.88	78.85	68.27
	(15 × 4)	74.04	73.08	64.42

Table 10 (*continued*)

Data set	Matrix size	TSMHKS$_A$	TSMHKS$_K$	MatMHKS
	(20 × 3)	77.88	71.15	65.38
	(30 × 2)	64.42	61.54	53.85
Thyroid	(3 × 7)	**93.23**	*93.23*	93.03
	(7 × 3)	92.71	93.03	92.71
Vowel	(1 × 10)	*54.98*	47.84	54.98
	(2 × 5)	50.87	**55.19**	41.34
	(5 × 2)	43.94	55.19	34.42
	(10 × 1)	37.66	47.40	28.14
Waveform	(1 × 21)	**86.27**	76.65	86.23
	(3 × 7)	84.91	*86.27*	81.43
	(7 × 3)	69.23	76.95	64.99
	(21 × 1)	44.38	85.95	39.42
Waveform-noise	(1 × 40)	*86.47*	77.74	86.47
	(2 × 20)	85.23	**86.75**	74.23
	(4 × 10)	85.11	77.74	80.03
	(5 × 8)	80.55	85.13	80.35
	(8 × 5)	74.03	80.55	57.18
	(10 × 4)	81.91	81.94	71.07
	(20 × 2)	50.10	74.33	41.34
	(40 × 1)	50.18	65.32	42.10
Wine	(1 × 13)	**100.00**	*100.00*	100.00
	(13 × 1)	76.67	96.67	76.67

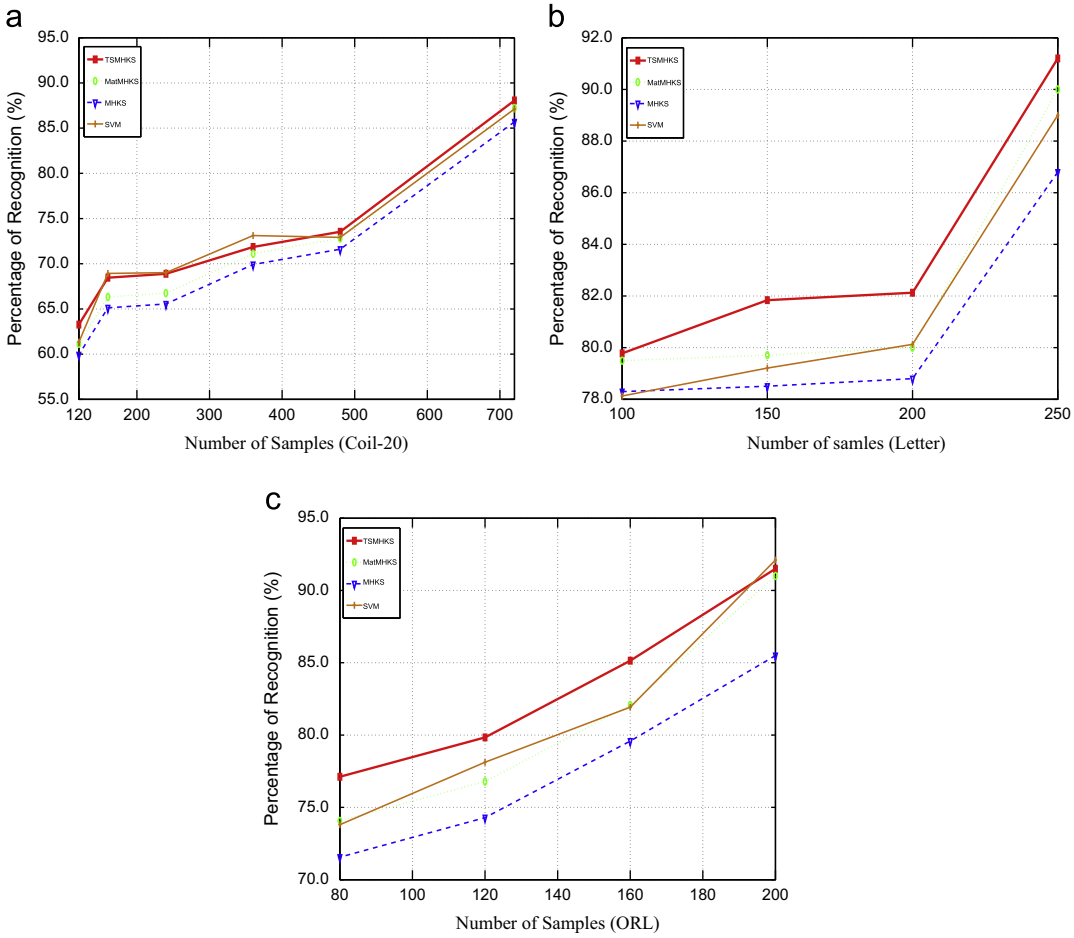

Fig. 5. Classification accuracy (%) comparison of TSMHKS, MatMHKS, MHKS, and SVM with the varying number of the training sets on Coil-20, Letter, and ORL.

the probability of accepting the non-target data named outlier objects. Thus the first ensemble strategy adopts SVDD as the base classifier that trains on each generated cluster. Further each cluster can induce one hypersphere. For the test processing, one given test pattern should be classified by multiple SVDDs. We firstly compute the distances between the test pattern and

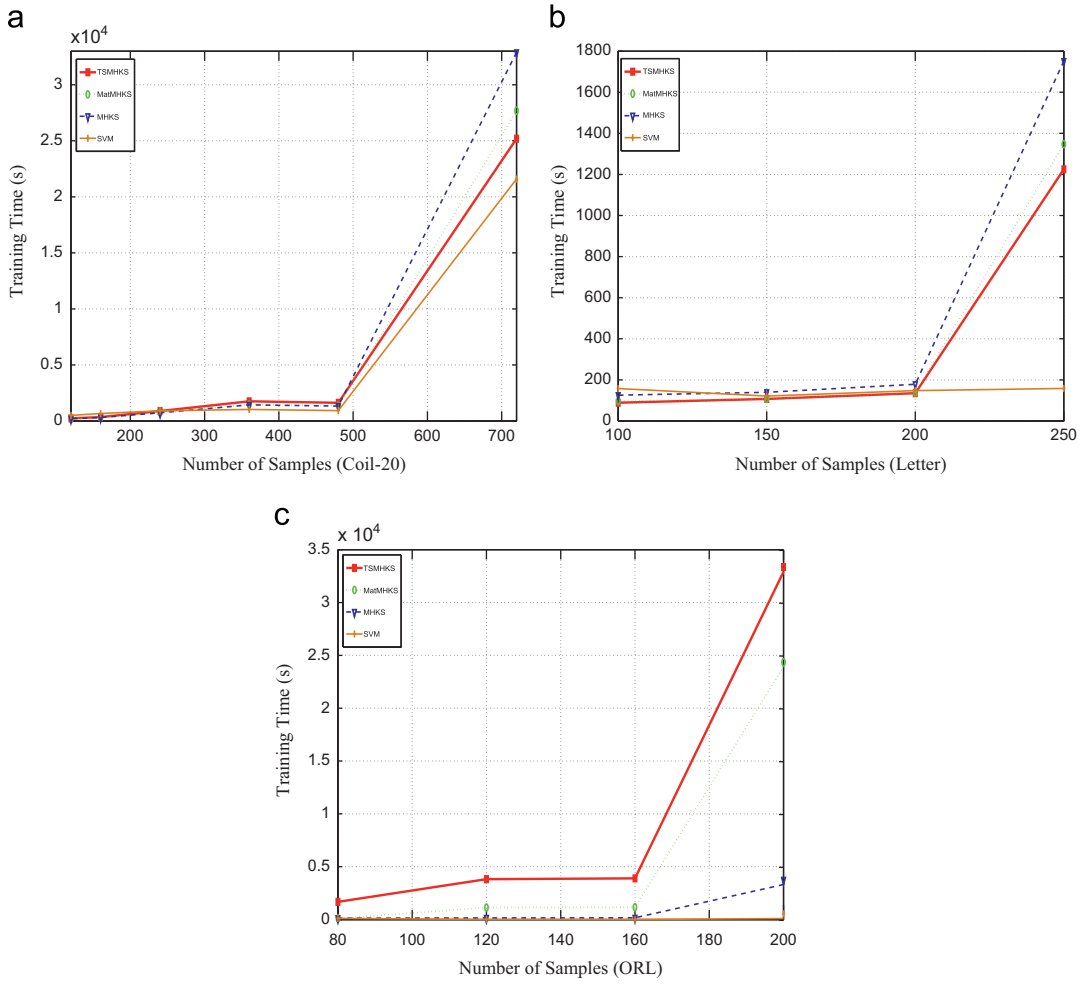

Fig. 6. The average training time (in seconds) comparison of TSMHKS, MatMHKS, MHKS, and SVM with the varying number of the training sets on Coil-20, Letter, and ORL.

multiple hypersphere centers of SVDDs. Then the test pattern has the same class label as the patterns which are in the nearest hypersphere. Since K-means clustering and AHC for generating the clusters is firstly implemented and SVDD follows, the first ensemble strategy in our experiments can be denoted as "K_{means}+SVDD" and "AHC+SVDD".

The second ensemble strategy is combining multiple TSMHKSs with different sets of parameters. In detail, both the regularized parameters C and λ are chosen from the set $\{10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3}\}$. The initialized weight vectors u and v are from the set $\{0.1,0.2,0.3,\ldots,0.9,1\}$. The pattern itself also has different kinds of matrix representations. In the experimental processing, we combine the classifiers generated from the above different parameters. In terms of the ensemble technique, we use three ways including the Adaboost, Bagging, and Voting. In the experiments, we firstly carry out the step including K-means clustering and AHC for generating the clusters. Then we adopt different ensemble techniques including Adaboost, Bagging, and Voting for multiple TSMHKSs. Thus the second ensemble strategy has the following different implementing ways including "K_{means}+Adaboost", "K_{means}+Bagging", "K_{means}+Voting", "AHC+Adaboost", "AHC+Bagging", and "AHC+Voting".

Tables 7 and 8 give the classification and computation time comparison between TSMHKS and the designed two ensemble strategies. In the tables, we use "K" to replace "K_{means}", "A" to "AHC", "S" to "SVDD", "A" to "Adaboost", "B" to "Bagging", and "V" to "Voting". From Table 7, it can be found that the first and second ensemble strategies have some advantages here due to the reduced risk of overfitting. The second ensemble strategy could work well on almost all the data sets. Consequently, its computation time increases but the risk of overfitting is reduced. Furthermore, it can also be found that Adaboost and Voting can bring better performance than Bagging. On the other hand, the second ensemble strategy would take more computation time, which is shown in Table 8.

4.4. Image data

The above experiments are carried out on the synthetical and some UCI data sets which are represented by vector. In the vector case, we should firstly reshape these data into matrices in different size so as to fit with TSMHKS and MatMHKS. In this section, we give a discussion for the proposed algorithm on the matrix data, i.e. images including COIL-20,[2] Letter,[3] and ORL.[4] COIL-20 is a database of gray-scale images of 20 objects. The objects are placed on a motorized turntable against a black background. The turntable is rotated through 360° to vary the object poses with respect to a fixed camera. Images of the objects are taken at pose intervals of 5°, which corresponds to 72 images

[2] http://www.cs.columbia.edu/CAVE/coil-20.html.
[3] http://sun16.cecs.missouri.edu/pgader/CECS477/NNdigits.zip.
[4] http://www.cam-orl.co.uk.

per object. For our experiments, we have resized each of the original 1440-dimensionality images down to 3232 pixels per image. Then, we use the size of the training with 36, 24, 18, 12, 8, and 6 and the rest as testing in each class in each run. Letter data set contains 10 digits from 0 to 9, each of which provides 50 different images with the size 24×18. Here we randomly choose 20%, 30%, 40%, and 50% per digit for training and the remaining for testing in each run. ORL data set has images from 40 persons, each

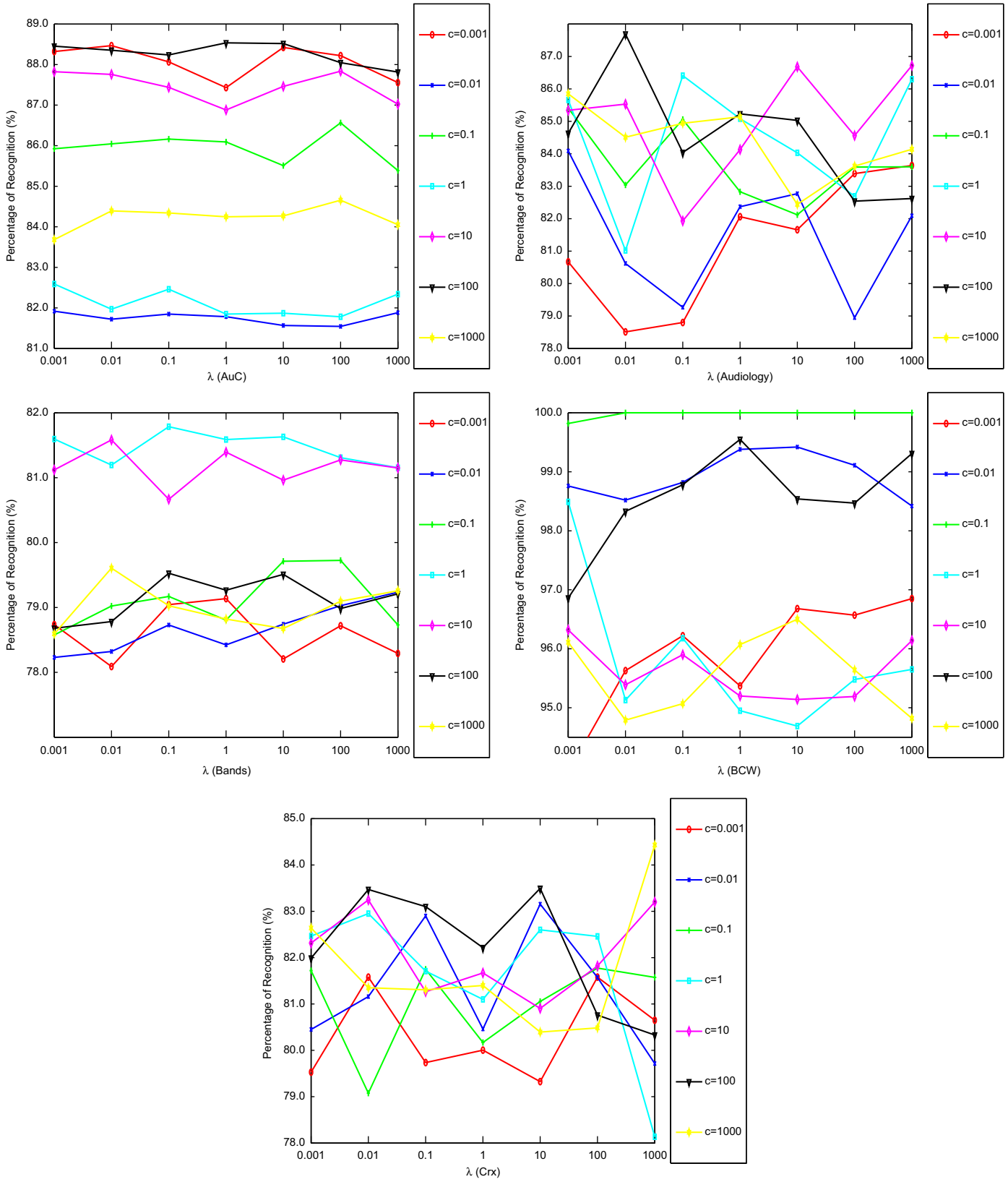

Fig. 7. Classification accuracies (%) of TSMHKS with the varying λ at certain fixed C on the used data sets.

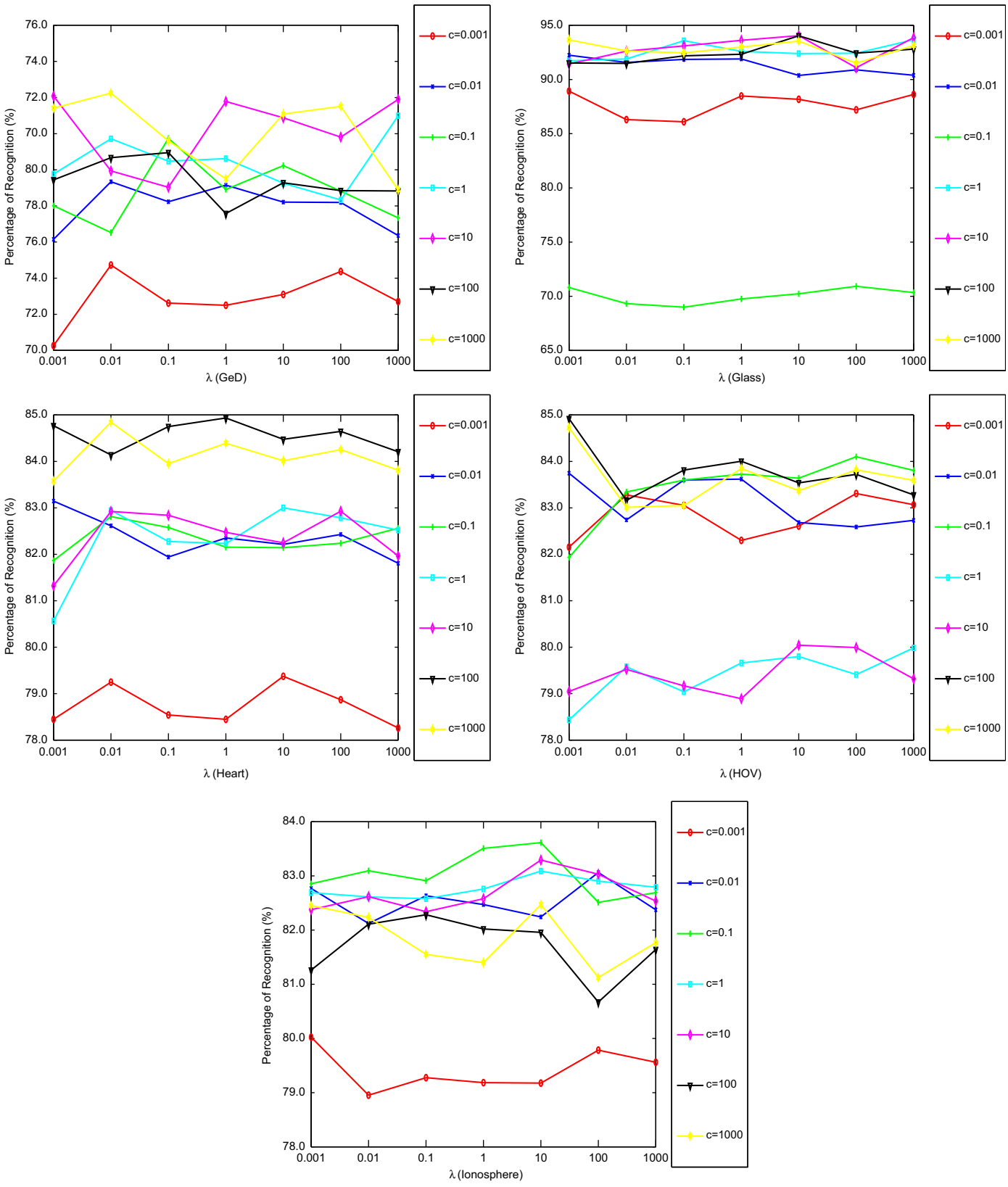

Fig. 7. (*continued*)

of which provides 10 different images. The main challenge on this data set is of pose and expression variations. For ORL, we employ the first five, four, three, and two images per person for training and the rest for testing in each run.

For the use images, we report the average classification accuracy and training time of TSMHKS, MatMHKS, MHKS, and SVM with the above setting. Fig. 5 gives the classification accuracy of TSMHKS, MatMHKS, MHKS, and SVM with the varying number of

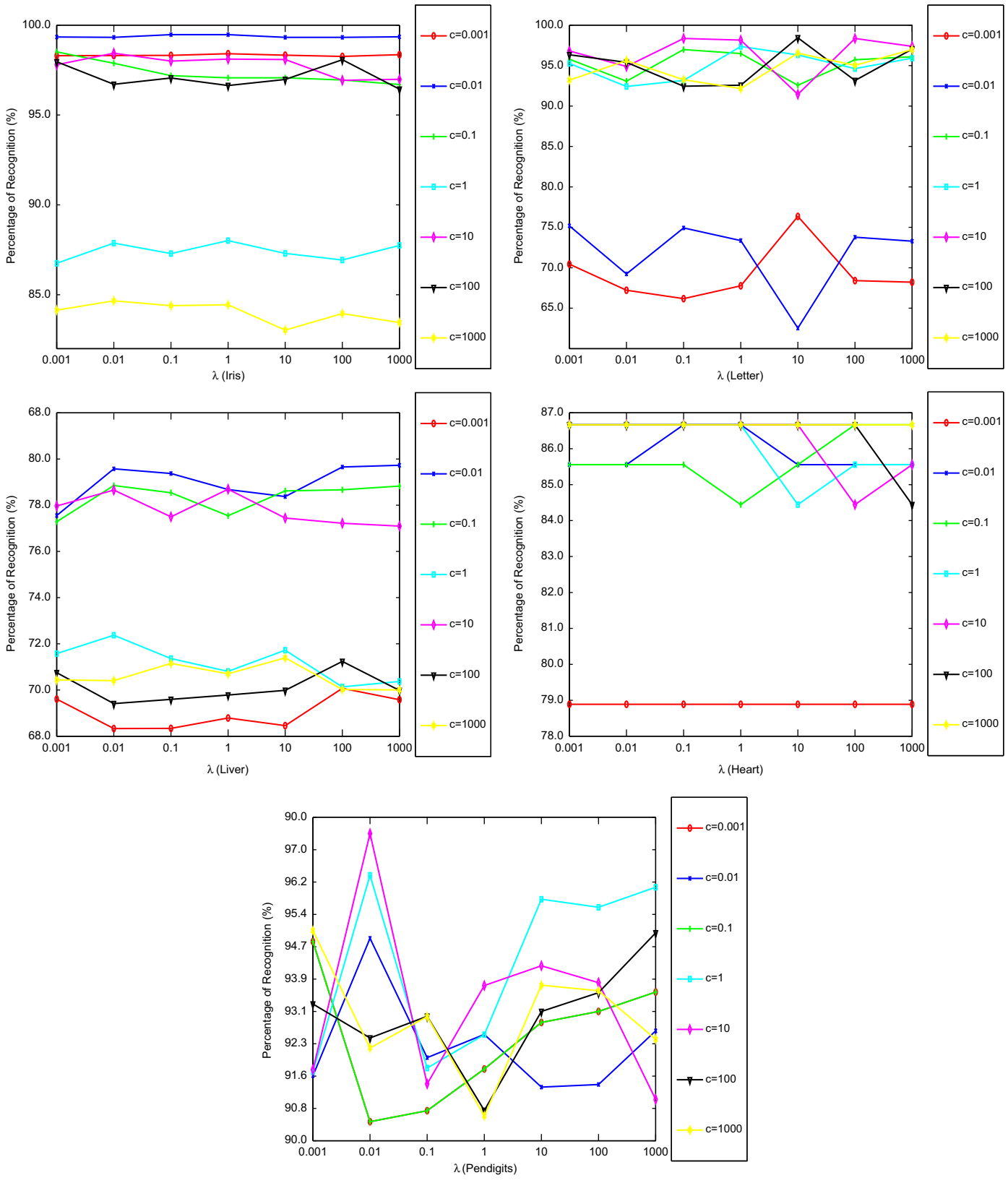

Fig. 7. (*continued*)

the training sets. Fig. 6 gives the average training time with the varying number of the training sets. From the figures, it can be found that: (a) TSMHKS has the best classification accuracy on the used three image data sets; (b) the training cost of the compared four algorithms are comparable; (c) TSMHKS has a significant classification superior to the other algorithms on the images with the small training set; (d) both the classification accuracy and training cost of the compared algorithms increase with the increasing size of training sets. Moreover, it can also be found that the matrixized classifier can well deal with the image data for the

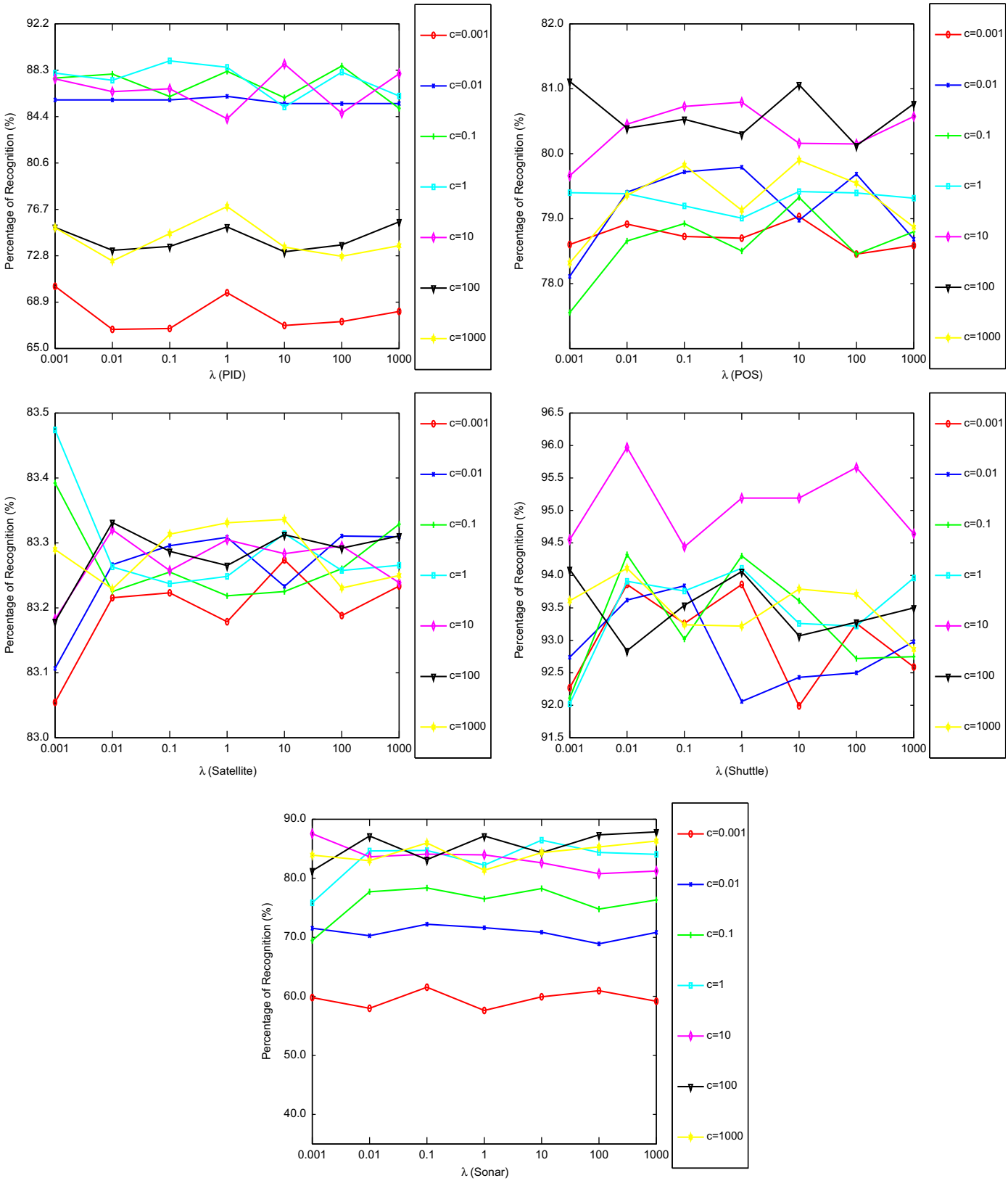

Fig. 7. (*continued*)

reason that the image data is a matrix pattern and the weight vectors u, $\tilde{v}$ can carry with prior information such as the structural or local contextual information which is shown by the representation of Kronecker production of the weight vectors u and $\tilde{v}$.

5. Conclusions

In this paper, we give the data information including the individual, cluster, and class structures. We explore one middle granularity

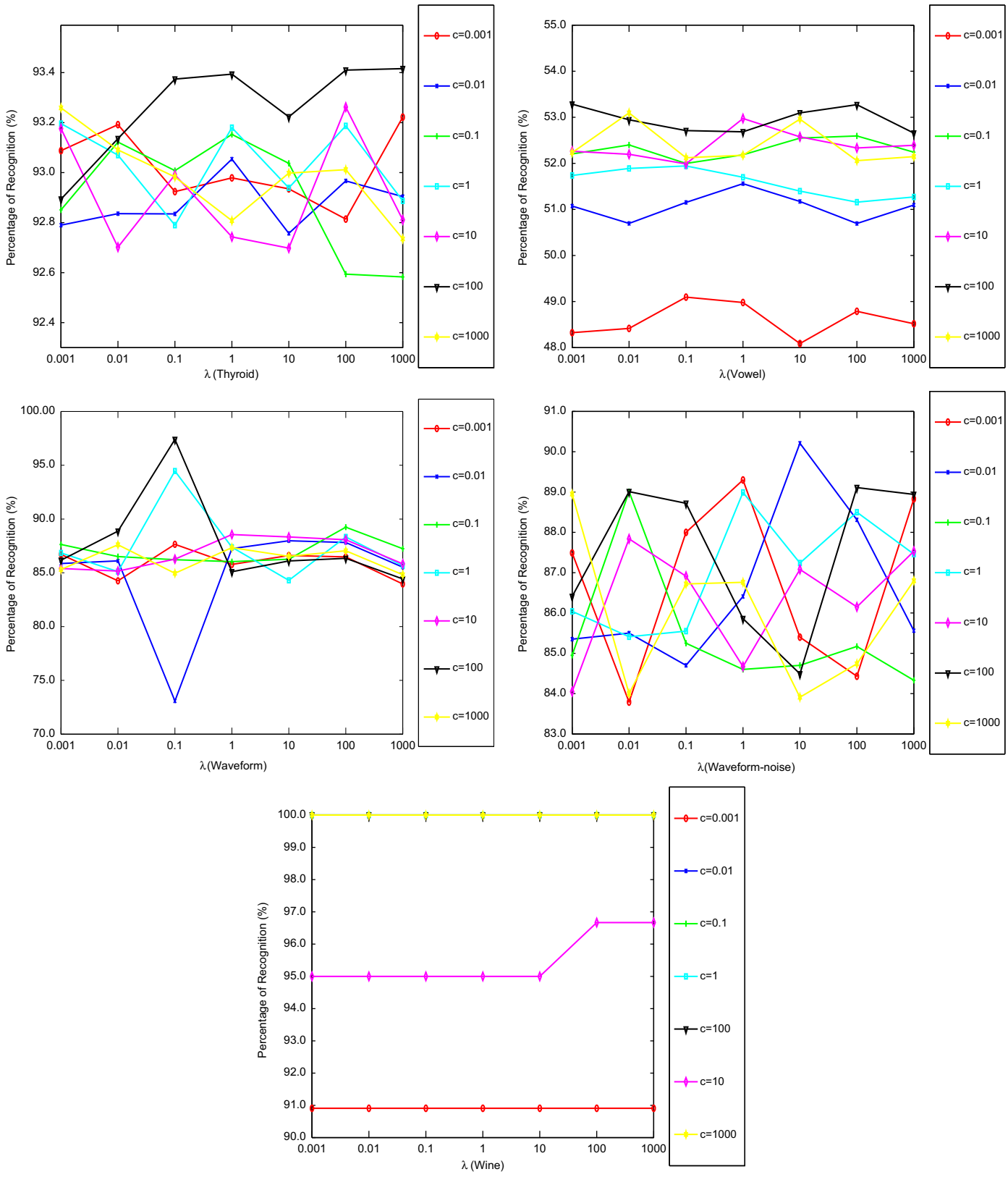

Fig. 7. (*continued*)

named the cluster between the class and individual, and introduce the cluster structure that means the structure within each class into the matrixized classifier design. In doing so, we can simultaneously utilize the class, the cluster, and the individual structures in the way that is from global to point. Therefore, the proposed classifier named TSMHKS has a three-fold structural information. For analyzing the feasibility and effectiveness of TSMHKS, we carry out TSMHKS and the related algorithms including MatMHKS, MHKS, and SVM on the

synthetic and some real-world data. We give the discussion for TSMHKS in terms of: (1) the size of the clusters; (2) the matrix size of pattern; (3) the regularized parameters; (4) the convergence; (5) the generalization risk; (6) the classification and computational complexity; and (7) the relationship with the ensemble techniques. According to the above discussion, it could be concluded that the proposed three-fold structural learning strategy has a superior classification performance and can lead to a low empirical generation risk bound.

Conflict of interest

None declared.

Acknowledgment

The authors would like to thank Natural Science Foundations of China under Grant nos. 61272198, 60903091, 21176077, and 61170151, the Specialized Research Fund for the Doctoral Program of Higher Education under Grant nos. 20090074120003, and the Fundamental Research Funds for the Central Universities for support.

References

[1] D. Beymer, T. Poggio, Image representations for visual learning, Science 272 (1996) 1905–1909.
[2] V.N. Vapnik, The Nature of Statistical Learning Theory, second edition, Springer Verlag, Berlin, 2000.
[3] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second edition, Wiley, New York, 2001.
[4] H. Wang, N. Ahuja, Rank-R approximation of tensors using image-as-matrix representation, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 346–353.
[5] H. Kong, L. Wang, E.K. Teoh, X. Li, J.G. Wang, R. Venkateswarlu, Generalized 2D principal component analysis for face image representation and recognition, Neural Networks 18 (5–6) (2005) 585–594.
[6] K. Liu, Y.Q. Cheng, J.Y. Yang, Algebraic feature extraction for image recognition based on an optimal discriminant criterion, Pattern Recognition 26 (6) (1993) 903–911.
[7] J. Yang, D. Zhang, A.F. Frangi, J. Yang, Two-dimensional PCA: a new approach to appearance-based face representation and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (1) (2004) 131–137.
[8] M. Li, B. Yuan, 2D-LDA: a statistical linear discriminant analysis for image matrix, Pattern Recognition Letters 26 (5) (2005) 527–532.
[9] S.C. Chen, Y.L. Zhu, D.Q. Zhang, J.Y. Yang, Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA, Pattern Recognition Letters 26 (8) (2005) 1157–1167.
[10] Z. Wang, S.C. Chen, J. Liu, D.Q. Zhang, Pattern representation in feature extraction and classifier design: matrix versus vector, IEEE Transactions on Neural Networks 19 (5) (2008) 758–769.
[11] S.C. Chen, Z. Wang, Y.J. Tian, Matrix-pattern-oriented Ho–Kashyap classifier with regularization learning, Pattern Recognition 40 (5) (2007) 1533–1543.
[12] Z. Wang, S.C. Chen, New least squares support vector machines based on matrix patterns, Neural Processing Letters 26 (1) (2007) 41–56.
[13] L. Nanni, A. Lumini, S. Brahnam, Survey on LBP based texture descriptors for image classification, Expert Systems with Applications 39 (3) (2012) 3634–3641.
[14] J. Leski, Ho–Kashyap classifier with generalization control, Pattern Recognition Letters 24 (14) (2003) 2281–2290.
[15] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Transactions on Neural Networks 12 (2) (2001) 181–201.
[16] H. Xue, S.C. Chen, Q. Yang, Structural regularized support vector machine: a framework for structural large margin classifier, IEEE Transactions on Neural Networks 22 (4) (2011) 573–587.
[17] P.K. Shivaswamy, T. Jebara, Ellipsoidal kernel machines, in: Proceedings of the 12th International Workshop on Artificial Intelligence and Statistics, 2007.
[18] G.R.G. Lanckriet, L.E. Ghaoui, C. Bhattacharyya, M.I. Jordan, A robust minimax approach to classification, Journal of Machine Learning Research 3 (2003) 555–582.
[19] K. Huang, H. Yang, I. King, M.R. Lyu, Learning large margin classifiers locally and globally, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004, p. 51.
[20] D.S. Yeung, D. Wang, W.W.Y. Ng, E.C.C. Tsang, X. Wang, Structured large margin machines: sensitive to data distributions, Machine Learning 68 (2) (2007) 171–200.
[21] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, Journal of Machine Learning Research 7 (48) (2006) 2399–2434.
[22] C. Cortes, V. Vapnik, Support vector machine, Machine Learning 20 (3) (1995) 273–297.
[23] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, Advances in Computational Mathematics 13 (1) (2000) 1–50.
[24] P. Zhang, J. Peng, SVM vs regularized least squares classification, in: Proceedings of the 17th International Conference on Pattern Recognition, vol. 1, 2004, pp. 176–179.
[25] L. Wang, X. Wang, J. Feng, On image matrix based feature extraction algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (1) (2006) 194–197.
[26] Q. Gao, L. Zhang, D. Zhang, J. Yang, Comments on "on image matrix based feature extraction algorithms", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (5) (2007) 1373–1374.
[27] T. Zhang, B. Fang, Y.Y. Tang, Z. Shang, B. Xu, Generalized discriminant analysis: a matrix exponential approach, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 40 (1) (2010) 186–197.
[28] J.A. Hartigan, M.A. Wong, Algorithm AS 136: a k-means clustering algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics) 28 (1) (1979) 100–108.
[29] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., 1988.
[30] L.A. Zadeh, Fuzzy sets, Information and Control 8 (3) (1965) 338–353.
[31] J.H. Ward Jr., Hierarchical grouping to optimize an objective function, American Statistical Association 58 (301) (1963) 236–244.
[32] S. Salvador, P. Chan, Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms, in: 16th IEEE International Conference on Tools with Artificial Intelligence, 2004, pp. 576–584.
[33] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2) (2002) 415–425.
[34] Q.S. Xu, Y.Z. Liang, Monte Carlo cross validation, Chemometrics and Intelligent Laboratory Systems 56 (1) (2001) 1–11.
[35] Z. Wang, S.C. Chen, D.Q. Gao, A novel multi-view learning developed from single-view patterns, Pattern Recognition 44 (10) (2011) 2395–2413.
[36] A. Frank, A. Asuncion, UCI machine learning repository (〈http://archive.ics.uci.edu/ml〉), University of California, Irvine, School of Information and Computer Sciences, 2010.
[37] J.P. Ye, Generalized low rank approximations of matrices, Machine Learning 61 (1) (2005) 167–191.
[38] B. Schölkopf, J. Shawe-Taylor, A.J. Smola, B.S. Gmd, E.J. Smola, R.C. Williamson, Generalization Bounds via Eigenvalues of the Gram Matrix, Technical Report 99-035, NeuroColt, 1999.
[39] H. Kashima, S. Oyama, Y. Yamanishi, K. Tsuda, On pairwise kernels: an efficient alternative and generalization analysis, in: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2009, pp. 1030–1037.
[40] D.M.J. Tax, R.P.W. Duin, Support vector domain description, Pattern Recognition Letters 20 (1999) 1191–1199.

Zhe Wang received the B.Sc. and Ph.D. degrees in Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2003 and 2008, respectively. He is now an Associate Professor in Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His research interests include feature extraction, kernel-based methods, image processing, and pattern recognition. At present, he has several papers with the first author published on some international journals including IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Neural Networks, Pattern Recognition, etc.

Changming Zhu received the B.Sc. degree in Department of Computer Science and Engineering, East China University of Science and Technology, China, 2010. Currently he is a graduate student at the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His research interests focus on neural computing and pattern recognition.

Daqi Gao received the Ph.D. degree from Zhejiang University, China, in 1996. Currently, he is a Professor in East China University of Science and Technology. He is a member of the International Neural Network Society. He has published over 50 scientific papers. His research interests are pattern recognition, neural networks, and machine olfactory.

Songcan Chen received his B.S. degree in mathematics from Hangzhou University (now merged into Zhejiang University) in 1983. In 1985, he completed his M.S. degree in computer applications at Shanghai Jiaotong University and then worked at Nanjing University of Aeronautics and Astronautics in January 1986. There he received a Ph.D. degree in communication and information systems in 1997. Since 1998, as a full-time professor, he has been with the Department of Computer Science and Engineering at Nanjing University of Aeronautics and Astronautics. His research interests include pattern recognition, machine learning and neural computing. In these fields, he has authored or coauthored over 130 scientific peer-reviewed papers.