Volume 40, Number 5
May 2007
ISSN 0031-3203
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect

Pattern Recognition 40 (2007) 1533 – 1543

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Matrix-pattern-oriented Ho–Kashyap classifier with regularization learning

Songcan Chen*, Zhe Wang, Yongjun Tian

Department of Computer Science & Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China

Received 14 April 2006; received in revised form 27 July 2006; accepted 5 September 2006

Abstract

Existing classifier designs generally base on vector pattern, hence, when a non-vector pattern such as a face image as the input to the classifier, it has to be first concatenated to a vector. In this paper, we, instead, explore using a set of given matrix patterns to design a classifier. For this, first we represent a pattern in matrix form and recast existing vector-based classifiers to their corresponding matrixized versions and then optimize their parameters. Concretely, considering its similar principle to the support vector machines of maximizing the separation margin and superior generalization performance, the modified HK algorithm (MHKS) is chosen and then a matrix-based MHKS classifier (MatMHKS) is developed. Experimental results on ORL, Letters and UCI data sets show that MatMHKS is more powerful in generalization than MHKS. This paper focuses on: (1) purely exploring the classification performance discrepancy between matrix- and vector-pattern representations; more importantly, (2) developing a new classifier design directly for matrix pattern.
© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Linear classifier; Matrix pattern; Vector pattern; Modified Ho–Kashyap with squared approximation of the misclassification errors (MHKS); Regularization; Pattern recognition

1. Introduction

Classifier design plays an important role in pattern recognition. There are many classifiers reported, including statistical [1], linear discrimination [2], k-nearest neighbor [2], neural network [2], kernel method [3] and so on. However, linear classifiers are still of special interest due to their simplicity, ease of analysis and implementation, and easy expansibility to nonlinear classifiers [2,4].

In tradition, the classifier design almost based on vector pattern, i.e., before applying them, any non-vector pattern such as an image should be firstly vectorized into a vector pattern by some technique like concatenation. But there is neither research to explore how classification performance is changed after such a vectorization, nor a classifier is designed directly based on non-vector (e.g., matrix) pattern. And Ugly Duckling Theorem [2] indicated that it cannot be said that one pattern representation is always better than another, so it is not always right that we design classifier based on vector pattern.

On the other hand, there is indeed a novel two-dimensional (2D) method, which can directly operate on non-vector pattern such as an image. Especially in CVPR'2005, there are several papers [5–9] about such 2D method. The 2D method mainly focuses on feature extraction and there are two typical categories of extracting features directly from original 2D image matrix: one belongs to the transforming feature methods such as 2D fast Fourier or discrete cosine transformations (FFT or DCT), 2D wavelet transformation and so on. They extract so-needed features via transforming original domains (such as spatial domain) to another domain (such as frequency domain). The other is an algebraic method typically including two-dimensional principal component analysis (2DPCA) [10] and 2D linear discriminant analysis (2DLDA) [11,12]. Yang et al. [10] proposed 2DPCA as an extension to PCA to extract features directly from the two-dimension face images and then so-obtained features are in turn concatenated to a vector pattern for classifier design. Their experiments on several well-known benchmark face

* Corresponding author. Tel.: +86 25 84892452; fax: +86 25 84498069.
E-mail address: s.chen@nuaa.edu.cn (S. Chen).

data sets showed that this classification method designed on top of the reshaped vector from 2D feature extractor is better than the counterpart based on the features extracted by classical PCA in favor of both image classification and reduction of computational complexity in feature extraction step. Ye et al. [11] and Li et al. [12] also, respectively, proposed 2DLDA based on images, which overcomes the singularity of the within-class scatter matrix in one-dimensional LDA incurred by small sample size problem and achieves competitive recognition accuracy on face identification. But all the above 2D methods are only used to deal with 2D image patterns in themselves. Chen et al. [13] went further and developed a more general method, called MatPCA and MatFLDA, to extract features directly based on matrix patterns reshaped from an original one-dimensional, even original 2D pattern. Compared with the conventional methods such as PCA and LDA operating on vector patterns, Chen et al.'s method firstly matrixizes a one-dimensional or 2D original pattern into a corresponding matrix pattern before extracting features. In doing so, the information, intuitively, should not be lost due to that such newly formed matrix pattern still retains all its feature components, more likely, some new implicit structural or contextual information can additionally be introduced though cannot be explicitly known or expressed.

However, as mentioned before, these 2D methods are only applied to feature extraction phase, and the subsequent classifier design still resorts to the traditional vector-based technique, i.e., the operating pattern of the classifier itself is still of vector representation rather than matrix representation. At the same time, as we have known, although the classifier based on the 2D feature extractor achieves better performance compared with that based on the 1D extractor, no further investigation is made which factors, the 2D strategy itself or just only dimension reduction, contributes to the performance gain. In order to explore reasons behind this, we intentionally sidestep the feature extraction phase and directly design a classifier on top of the existent models directly operating on matrix pattern with aiming to (1) explore whether such a design can really promote the classification performance; (2) develop a new technique for classifier design directly operating on matrix pattern. Without loss of generality, here we confine our investigation to linear classifiers due to their ease of mathematical tractability.

In this paper we mainly concern with two-class problem; but the proposed method is easily generalized to multi-class problem [14]. When designing a linear classifier, first, it is assumed that the discrimination function $g(\boldsymbol{x})$ is a linear function of sample $\boldsymbol{x}$, such that

$$g(\boldsymbol{x}) = \widetilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{x} + w_0, \tag{1}$$

where $\widetilde{\boldsymbol{w}}$ is a weight vector, w_0 is a threshold weight or bias.

From the discrimination function $g(\boldsymbol{x})$ in Eq. (1), it is easily understood that the linear algorithms above are all vector-pattern-oriented. But, such a vectorization will bring three potential problems at least [13]: (1) some implicit structural or local contextual information of image may be lost after vectorization, these information may be useful for classification; (2) as the dimension of weight vector is equal to the dimension of input pattern (e.g. the dimension of an image pattern is $d_1 \times d_2$, after vectorization, the dimension d of input pattern is also equal to $d_1 \times d_2$), then the higher the dimension of input pattern, the more memory required for the weight vector; (3) some researchers [15,16] have also demonstrated that a classifier designed is easily overtrained when the dimensionality of a vector pattern is very high and the sample size is small simultaneously. Intuitively, directly manipulating images in design of a classifier seems simpler and also not to lose too much spatial or local contextual information of the original image. Due to the above possible shortcomings of vector-pattern-oriented linear classifier, another purpose of this paper is to mitigate these shortcomings.

The most important feature of a classifier is its generalization ability which measures classification performance of the designed classifier for previously unseen data in the training phase. In designing a classifier, one useful tool of optimizing its generalization ability is a generalization error bound offered by the Vapnik–Chervonenkis (VC) theory [17]. Leski [18] proposed Ho–Kashyap (HK) classifier with generalization control (MHKS). MHKS adopts similar principle to the support vector machine (SVM) and maximizes the separating margin but without need to solve the quadratic programming (QP) problem, and thus gets better generalization performance than original HK algorithm in their experiments [18]. In this paper, due to both simplicity and ease of mathematical treatment of MHKS, we select it as our matrixizing paradigm and attempt to fuse such a new idea (2D) into the MHKS and consequently develop a matrix-pattern-oriented MHKS (MatMHKS). The major characteristic of MatMHKS is (1) that a matrix pattern with the size of $d_1 \times d_2$ replaces a $d_1 d_2 \times 1$ vector pattern; (2) two weight vectors, respectively, acting on the two sides of the matrix pattern replaces the original single weight vector $\widetilde{\boldsymbol{w}}$ in MHKS, thus, the memory required for the weight vector is reduced from $d_1 \times d_2$ in MHKS to $d_1 + d_2$ in MatMHKS. The experiments on some real benchmark data sets show that MatMHKS is more powerful in classification performance or generalization ability than MHKS.

The rest of this paper is organized as follows: in Section 2, we review the original HK and MHKS algorithms. In Section 3, we will detail our MatMHKS operating directly on matrix patterns. In Section 4, we will analyze the relationship between MatMHKS and MHKS. In Section 5, we present our experimental results on some real data sets and further discussion. Finally, conclusion and future works are drawn in Section 6.

2. Ho–Kashyap and MHKS algorithm

For two-class (ω_1, ω_2) problem, let $Tr^{(N)} = \{(\boldsymbol{x}_1, \varphi_1), (x_2, \varphi_2) \cdots (\boldsymbol{x}_N, \varphi_N)\}$ be a given training data set, where N

is the sample number and each independent pattern $\boldsymbol{x}_i \in R^d$ has a corresponding class label $\varphi_i \in \{+1, -1\}$, when $\boldsymbol{x}_i \in \omega_1$, $\varphi_i = +1$, and when $\boldsymbol{x}_i \in \omega_2$, $\varphi_i = -1$. Through defining the augmented pattern vector $\boldsymbol{y}_i = [\boldsymbol{x}_i^{\mathrm{T}}, 1]^{\mathrm{T}}$, a corresponding augmented weight vector $\boldsymbol{w}=[\widetilde{\boldsymbol{w}}^{\mathrm{T}}, w_0]^{\mathrm{T}} \in R^{d+1}$, then, linear discrimination function for two-class problem, $g(\boldsymbol{y})$, can be described as

$$g(\boldsymbol{y_i}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{y_i} \begin{cases} > 0, & \boldsymbol{y_i} \in \omega_1, \\ < 0, & \boldsymbol{y_i} \in \omega_2. \end{cases} \tag{2}$$

If we multiply class label φ_i by corresponding augmented pattern $\boldsymbol{y}_i$ of the training data set, then (2) can be rewritten in the form as below

$$g(\boldsymbol{y}_i) = \varphi_i \boldsymbol{w}^{\mathrm{T}}\boldsymbol{y}_i > 0, \tag{3}$$

where $i = 1, 2, \ldots, N$. To express simply, let $\boldsymbol{y}_i = \varphi_i \boldsymbol{y}_i$, $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2 \ldots \boldsymbol{y}_N]^{\mathrm{T}}$, then, (3) can be denoted in matrix form

$$\boldsymbol{Yw} > \boldsymbol{0}. \tag{4}$$

The criterion function of HK algorithm is the quadratic loss function as follows:

$$J_s(\boldsymbol{w}, \boldsymbol{b}) = \|\boldsymbol{Yw} - \boldsymbol{b}\|^2, \tag{5}$$

where $\boldsymbol{b}$ is the margin vector, and $\boldsymbol{b} > 0$, the gradients of J_s with respect to $\boldsymbol{w}$ and $\boldsymbol{b}$ are, respectively, given by

$$\nabla_{\boldsymbol{w}} J_s = 2\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{Yw} - \boldsymbol{b}), \tag{6}$$

$$\nabla_{\boldsymbol{b}} J_s = -2(\boldsymbol{Yw} - \boldsymbol{b}). \tag{7}$$

For any value of $\boldsymbol{b}$, we can always take

$$\boldsymbol{w} = \boldsymbol{Y}^{\dagger}\boldsymbol{b}. \tag{8}$$

Thereby, only in one step, we get $\nabla_{\boldsymbol{w}} J_s = \boldsymbol{0}$ and minimize J_s with respect to $\boldsymbol{w}$. Now we are not so free to modify $\boldsymbol{b}$ due to the constraint $\boldsymbol{b} > \boldsymbol{0}$, and we must avoid $\boldsymbol{b}$ converging to zero in the descent procedure. The way to prevent $\boldsymbol{b}$ from converging to zero in HK algorithm is to start with $\boldsymbol{b} > \boldsymbol{0}$ and to refuse reducing any of its components. The detail is that, we firstly set all positive components of $\nabla_b J_s$ to zero, and then try to follow the negative gradient. Thus, we obtain the HK rule for minimizing $J_s(\boldsymbol{w}, \boldsymbol{b})$ from (7) and (8)

$$\begin{cases} \boldsymbol{b}(1) > \boldsymbol{0}, \\ \boldsymbol{b}(k+1) = \boldsymbol{b}(k) + \rho(\boldsymbol{e}(k) + |\boldsymbol{e}(k)|), \end{cases} \tag{9}$$

where $\boldsymbol{e}(k) = \boldsymbol{Yw}(k) - \boldsymbol{b}(k)$, k denotes the iteration index, learning rate $\boldsymbol{\rho} > 0$, HK algorithm can find separable vector in linearly separable case and provide evidence in inseparable case. But this algorithm is sensitive to outliers [18] and cannot guarantee good recognition accuracies. Leski [18] proposed a modified HK algorithm with MHKS to remedy this shortcoming and defined canonical hyperplane as below

$$\boldsymbol{Yw} \geqslant \boldsymbol{1}_{N\times 1}. \tag{10}$$

So, for each class, there is at least one closest point to the corresponding canonical hyperplane, we have $\boldsymbol{w}^{\mathrm{T}}\ \boldsymbol{y}_i = 1$. The margin of separation is defined as the perpendicular distance between the two hyperplanes of both sides, and is equal to $\mu = 2/\|\widetilde{\boldsymbol{w}}\|$ due to the normal vector to the separation hyperplane is $\widetilde{\boldsymbol{w}}/\|\widetilde{\boldsymbol{w}}\|$. This is similar to SVM. The principle of SVM is to find the hyperplane with largest margin, the larger the margin is, the better the classifier designed is. In this case, we can control the Vapnik–Cervonenkis (VC-dimension) [17]: $h_{vc} \leqslant (G^2/\mu^2)+1$, where G is the radius of the smallest ball around the training data. From (10), linear separability is redefined as: if condition (10) is satisfied, then the data are said to be linearly separable. Then, we should seek the weight vector $\boldsymbol{w}$ to satisfy condition (10). To obtain a solution, the above inequality is replaced by the following equality $\boldsymbol{Yw} - \boldsymbol{1}_{N\times 1} = \boldsymbol{b}$, $\boldsymbol{b} = [b_1, b_2, \ldots, b_N]^{\mathrm{T}}$. To find the optimal weight vector $\boldsymbol{w}$, Leski [18] defined a criterion with regularization term as follows:

$$\min_{\boldsymbol{w}\in R^{d+1}, \boldsymbol{b}>0} \quad I(\boldsymbol{w}, \boldsymbol{b}) \triangleq (\boldsymbol{Yw} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{Yw} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b}) + c\widetilde{\boldsymbol{w}}^{\mathrm{T}}\widetilde{\boldsymbol{w}}, \tag{11}$$

where the second term of the right-hand side of (11) is a regularization one, the regularization parameter $c \geqslant 0$, by minimizing criterion (11), we can get the optimal weight vector $\boldsymbol{w}$ by the gradient descent technique:

$$\begin{cases} \boldsymbol{w} = (\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} + c\widetilde{\boldsymbol{I}})^{-1}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{b} + \boldsymbol{1}_{N\times 1}), \\ \boldsymbol{e} = (\boldsymbol{Yw} - \boldsymbol{b} - \boldsymbol{1}_{N\times 1}) = \boldsymbol{0}, \end{cases} \tag{12}$$

where $\widetilde{\boldsymbol{I}}$ is the identity matrix with the last element on the main diagonal set to zero.

3. Matrix-pattern-oriented HK classifier with regularization learning (MatMHKS)

As mentioned above, original linear algorithms are all vector-pattern-oriented and may have several shortcomings for 2D image patterns in themselves. For exploring the performance of a classifier oriented to patterns with different representation forms including 1D and 2D, and expecting to mitigate the shortcomings, in this paper, we develop a matrix-pattern-oriented, two-sided linear classifier based on HK algorithm with the regularization learning. We call it MatMHKS as an extension to MHKS whose discrimination function $g(\boldsymbol{A})(\boldsymbol{A} \in R^{d_1\times d_2})$ has the following form

$$g(\boldsymbol{A}) = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}\widetilde{\boldsymbol{v}} + \mathrm{v}_0, \tag{13}$$

where $\boldsymbol{u}$, $\widetilde{\boldsymbol{v}}$ are the two weight vectors, respectively, applied on left and right side of the matrix pattern $\boldsymbol{A}$ so as to make $g(\boldsymbol{A})$ linear, respectively, in $\boldsymbol{u}$ and $\widetilde{\boldsymbol{v}}$, and thus here is called two-sided linear (discrimination) function, v_0 is a threshold or bias. Naturally, in such a matrix-pattern-oriented case, the dimensions of the weight vectors, $\boldsymbol{u}$ and $\widetilde{\boldsymbol{v}}$, are d_1 and d_2,

Table 1
Memory required and relative reduced rate for the weight vectors of MatMHKS and MHKS (suppose that a vector component takes up one unit space here)

Image size	$d_1 + d_2$ (Matrix)	$d_1 \times d_2$ (Vector)	$(d_1 + d_2)/(d_1 \times d_2)$
28×23	51	644	1:12.6
112×92	204	10304	1:50.5
512×512	1024	262144	1:256

respectively. In original vector-pattern-oriented counterpart, the dimension of the weight vector $\widetilde{\boldsymbol{w}}$ is $d_1 \times d_2$, the ratio of memory required for the weight vectors in the two versions is $(d_1 + d_2)/(d_1 \times d_2)$. Table 1 shows that the higher the dimensionality of a pattern is, the more the memory required in the matrix-pattern-oriented case is reduced. For example, for an image with dimension 512×512, the reduced rate achieves 256, very considerable.

From the discrimination function (13), we can convert a classifier design problem based on matrix pattern to the problem of searching two-sided optimal weight vectors $\boldsymbol{u}$ and $\widetilde{\boldsymbol{v}}$. Now for two-class (ω_1, ω_2) problem in the matrix-pattern-oriented case, let $Tr2D^{(N)}$ denotes training data set, $Tr2D^{(N)} = \{(\boldsymbol{A}_1, \varphi_1), (A_2, \varphi_2), \ldots, (\boldsymbol{A}_N, \varphi_N)\}$, where N is the sample number and each independent pattern $\boldsymbol{A}_i \in R^{d_1 \times d_2}$ has a corresponding class label $\varphi_i \in \{+1, -1\}$ which indicates the assignment to one of the two class ω_1 or ω_2. Two-sided linear discrimination function for two-class problem can be given in terms of

$$g(\boldsymbol{A_i}) = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + \mathrm{v}_0 \begin{cases} > 0, & \boldsymbol{A_i} \in \omega_1, \\ < 0, & \boldsymbol{A_i} \in \omega_2. \end{cases} \tag{14}$$

Similarly, through multiplying φ_i by corresponding matrix pattern $\boldsymbol{A}_i$ of the training data set, then (14) can be reformulated into

$$g(\boldsymbol{A_i}) = \varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + \mathrm{v}_0) > 0 \tag{15}$$

where $i = 1, 2, \ldots, N$. Thus, we can also define a canonical hyperplane in present case as below

$$g(\boldsymbol{A_i}) = \varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + v_0) > 1, \tag{16}$$

where $i = 1, 2, \ldots, N$. Now, we seek the optimal weight vectors to satisfy condition (16) and replace the above inequalities with the following equalities $\varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + v_0) - 1 = b_i$ $(i = 1, 2, \ldots, N)$. Let $\boldsymbol{b} = [b_1, b_2, \ldots, b_N]^{\mathrm{T}}$ be an arbitrary positive vector, $\boldsymbol{b} \geqslant \boldsymbol{0}_{N\times 1}$, and $\boldsymbol{e} = [e_1, e_2, \ldots, e_N]^{\mathrm{T}}$, an error vector with components $e_i = \varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + v_0) - 1 - b_i$ $(i = 1, 2, \ldots, N)$. The pth component, e_p, of the $\boldsymbol{e}$ is a measure of the distance of the pth pattern to the separation hyper-plane (the distance is called margin) and $e_p \geqslant 0$ if its margin is positive, in this case, the pattern $\boldsymbol{A}_p$ is correctly classified and thus e_p can be set to zero by increasing the corresponding b_p. On the other hand, $e_p < 0$ if the margin of the pth pattern is negative, but due to the constraint $b_p > 0$, it is impossible to prevent condition $b_p < 0$ by decreasing b_p to set e_p to zero. Thus, the number of $e_i \geqslant 0$ $(i = 1, 2, \ldots, N)$ in the error vector $\boldsymbol{e}$ denotes the number of the correctly classified samples, and conversely the number of $e_i < 0$ $(i = 1, 2, \ldots, N)$ may reflect the number of the incorrectly classified samples. Through multiplying e_i by -1, the misclassification error canbe characterized in the form

$$\boldsymbol{I}(\boldsymbol{u}, \widetilde{\boldsymbol{v}}, v_0, \boldsymbol{b}) = \sum_{i=1}^{N} \hbar(-\boldsymbol{e}_i), \tag{17}$$

where $\hbar(e_i) = 1$ for $e_i > 0$, and $\hbar(e_i) = 0$ for $e_i \leqslant 0$. Now we can seek the optimal weight vectors by minimizing criterion (17), but this optimization problem is NP-complete due to the criterion is not a convex function, like criterion (11), we define the criterion below instead of (17),

$$\begin{aligned} \min_{\boldsymbol{u}\in R^{d_1}, \widetilde{\boldsymbol{v}}\in R^{d_2}, \boldsymbol{b}>\boldsymbol{0}} \quad & \boldsymbol{I}(\boldsymbol{u}, \widetilde{\boldsymbol{v}}, v_0, \boldsymbol{b}) \\ & \triangleq \sum_{i=1}^{N} [\varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + v_0) - 1 - \boldsymbol{b}_i]^2 \\ & + c(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{S_1}\boldsymbol{u} + \widetilde{\boldsymbol{v}}^{\mathrm{T}}\boldsymbol{S_2}\widetilde{\boldsymbol{v}}), \end{aligned} \tag{18}$$

where $\boldsymbol{S}_1 = d_1\boldsymbol{I}_{d_1\times d_1}$ $\boldsymbol{S_2} = d_2\boldsymbol{I}_{d_2\times d_2}$ are two regularization matrices, respectively, corresponding to the $\boldsymbol{u}$ and $\widetilde{\boldsymbol{v}}$, the regularization parameter $c \geqslant 0$ controls the generalization ability of the classifier designed by making a tradeoff between the complexity of the classifier and the training errors. To express simply, we set $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N]^{\mathrm{T}}$, $\boldsymbol{y}_i = \varphi_i[\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}_i, 1]^{\mathrm{T}}$ $(i = 1, 2, \ldots, N)$, $\boldsymbol{v} = [\widetilde{\boldsymbol{v}}^{\mathrm{T}}, v_0]^{\mathrm{T}}$, thus, (18) can be simplified in matrix form as follows:

$$\begin{aligned} \min_{\boldsymbol{u}\in R^{d_1}, \boldsymbol{v}\in R^{d_2+1}, \boldsymbol{b}>0} \quad I(u, \boldsymbol{v}, \boldsymbol{b}) \triangleq & (\boldsymbol{Y}\boldsymbol{v} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{v} - \boldsymbol{I}_{N\times 1} \\ & - \boldsymbol{b}) + c(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{S_1}\boldsymbol{u} + v^{\mathrm{T}}\widetilde{\boldsymbol{S}}_{\boldsymbol{2}}v), \end{aligned} \tag{19}$$

where $\widetilde{\boldsymbol{S}_2}$ is a matrix with dimensionality of $(d_2+1)\times(d_2+1)$ and $\widetilde{\boldsymbol{S}_2} = \begin{bmatrix} \boldsymbol{S_2} & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix}$. From (18) or (19), we cannot directly find closed-form optimal weights, instead, use the gradient descent technique to iteratively seek them. The gradients of the objective function (18) and (19) with respect to $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{b}$ are

$$\nabla \boldsymbol{I}_u = 2\sum_{i=1}^{N} \varphi_i\boldsymbol{A_i}\widetilde{\boldsymbol{v}}[\varphi_i(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A_i}\widetilde{\boldsymbol{v}} + \boldsymbol{v}_0) - 1 - \boldsymbol{b}_i] + 2c\boldsymbol{S_1}\boldsymbol{u}, \tag{20}$$

$$\nabla \boldsymbol{I_b} = -2(\boldsymbol{Y}\boldsymbol{v} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b}), \tag{21}$$

$$\nabla \boldsymbol{I}_{\boldsymbol{v}} = 2\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{v} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b}) + 2c\widetilde{\boldsymbol{S}}_{\boldsymbol{2}}\boldsymbol{v}. \tag{22}$$

By setting $\nabla \boldsymbol{I}_u = \boldsymbol{0}$ and $\nabla \boldsymbol{I}_{\boldsymbol{v}} = \boldsymbol{0}$, we can get

$$\boldsymbol{v} = (\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} + c\widetilde{\boldsymbol{S}}_{\boldsymbol{2}})^{-1}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{1}_{N\times 1} + \boldsymbol{b}), \tag{23}$$

$$\boldsymbol{u} = \left(\sum_{i=1}^{N} \boldsymbol{A}_{\boldsymbol{i}}\widetilde{\boldsymbol{v}}\widetilde{\boldsymbol{v}}^{\mathrm{T}}\boldsymbol{A}_{\boldsymbol{i}}^{\mathrm{T}} + c\boldsymbol{S}_{\boldsymbol{1}}\right)^{-1} \left(\sum_{i=1}^{N} \varphi_i(1+\boldsymbol{b}_i - \varphi_i v_0)\boldsymbol{A}_{\boldsymbol{i}}\widetilde{\boldsymbol{v}}\right). \tag{24}$$

From Eqs. (23) and (24), it can be seen that the weight vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are all determined by the margin vector $\boldsymbol{b}$ whose components determine the distance of the corresponding sample to the separation hyperplane. Moreover, because $\boldsymbol{u}$ and $\boldsymbol{v}$ are also mutually dependent, hence, we seek the $\boldsymbol{u}$ and $\boldsymbol{v}$ by initializing $\boldsymbol{b}$ and the previous iteration.

The design procedure for MatMHKS can be summarized in the following steps:

(1) Fix $c \geqslant 0$, $0 < \rho < 1$ Initialize $\boldsymbol{b}(1) \geqslant 0$ and $\boldsymbol{u}(1)$, set the iteration index $k = 1$,
(2) $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N]^{\mathrm{T}}$, where $\boldsymbol{y}_i = \varphi_i[\boldsymbol{u}(k)^{\mathrm{T}}\boldsymbol{A}_i, 1]^{\mathrm{T}}$,
(3) $\boldsymbol{v}(k) = (\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} + c\widetilde{\boldsymbol{S}}_{\boldsymbol{2}})^{-1}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{1}_{N\times 1} + \boldsymbol{b}(k))$,
(4) $\boldsymbol{e} = \boldsymbol{Y}\boldsymbol{v} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b}$,
(5) $\boldsymbol{b}(k+1) = \boldsymbol{b}(k) + \boldsymbol{\rho}(\boldsymbol{e}^{(k)} + |\boldsymbol{e}^{(k)}|)$,
(6) if $\|\boldsymbol{b}(k+1) - \boldsymbol{b}(k)\| > \xi$, then $k = k+1$, go to Step (7), else stop,
(7)

$$\boldsymbol{u}(k+1) = \left(\sum_{i=1}^{N} \boldsymbol{A}_{\boldsymbol{i}}\widetilde{\boldsymbol{v}}(k)\widetilde{\boldsymbol{v}}(k)^{\mathrm{T}}\boldsymbol{A}_{\boldsymbol{i}}^{\mathrm{T}} + c\boldsymbol{S}_{\boldsymbol{1}}\right)^{-1} \times \left(\sum_{i=1}^{N} \varphi_i(1+b_i(k) - \varphi_i v_0)\boldsymbol{A}_{\boldsymbol{i}}\widetilde{\boldsymbol{v}}(k)\right) \text{ Go to Step (2),}$$

where k denotes iteration step, $\boldsymbol{\rho} > 0$, and $\boldsymbol{b}(k+1) = \boldsymbol{b}(k)^{+}\rho(\boldsymbol{e}^{(k)} + |\boldsymbol{e}^{(k)}|)$ prevents the reduction of all the components of $\boldsymbol{b}$, ξ is a preset parameter. When $d_1 = 1$, $\boldsymbol{u} = 1$ and omitting step (7), this procedure is degenerated to MHKS [18], and at the same time, when c is set to 0, this procedure is degenerated to original Ho–Kashyap algorithm. So Ho–Kashyap and MHKS algorithm are two special instances of MatMHKS.

Now, we can represent the decision function of the classifier for input pattern $\boldsymbol{A}$

$$\boldsymbol{d}(\boldsymbol{A}) = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}\widetilde{\boldsymbol{v}} + v_0 \begin{cases} \geqslant 0, & \boldsymbol{A} \in \omega_1, \\ < 0, & \boldsymbol{A} \in \omega_2, \end{cases} \tag{25}$$

where the two-sided weight vectors $\boldsymbol{u}$ and $\widetilde{\boldsymbol{v}}$, threshold or bias v_0 are obtained in the process of training.

4. Relationship among MatMHKS and MHKS

In this section, we will further reveal the relationship between MatMHKS and MHKS. Before pointing out their essence, let us first introduce the following theorem:

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

$$vec(ABC) = (C^{\mathrm{T}} \otimes A)vec(B), \tag{26}$$

where $vec(X)$ denotes an operator that vectorizes a matrix X to corresponding a vector, for example, let $\boldsymbol{X} = (\boldsymbol{x}_{ij}) \in \boldsymbol{C}^{p\times q}$ and $\boldsymbol{x}_{\boldsymbol{i}} = (\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \ldots, \boldsymbol{x}_{pi})^{\mathrm{T}}$ *is the ith column of X, thus $vec(X) = (x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \ldots, x_q^{\mathrm{T}})^{\mathrm{T}}$ is a vector with $p \times q$ dimensionality. And "$\otimes$" denotes Kronecker product operation.*

Accordingly, in terms of Theorem 1, the discrimination function of MatMHKS (13) can be transformed into

$$\begin{aligned} \mathrm{g}(\boldsymbol{A}) &= vec(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}\widetilde{\boldsymbol{v}}) + v_0 = (\widetilde{\boldsymbol{v}}^{\mathrm{T}} \otimes \boldsymbol{u}^{\mathrm{T}})vec(\boldsymbol{A}) + v_0 \\ &= (\widetilde{\boldsymbol{v}} \otimes \boldsymbol{u})^{\mathrm{T}}vec(\boldsymbol{A}) + v_0. \end{aligned} \tag{27}$$

Comparing to the discrimination function of MHKS (1), we find that the discrimination functions of MatMHKS and MHKS have the same form now, and $\widetilde{\boldsymbol{v}} \otimes \boldsymbol{u}$ in MatMHKS plays the same role as the weight vector $\widetilde{\boldsymbol{w}}$ in MHKS. It is easy to prove that the solution space for $\widetilde{\boldsymbol{v}} \otimes \boldsymbol{u}$ is contained in that for $\widetilde{\boldsymbol{w}}$, because usually, the weight vector $\widetilde{\boldsymbol{w}}$ of MHKS does not always satisfy decomposability of the Kronecker product.

We can also see this point by comparing the criterion functions of MatMHKS and MHKS. First let $\widetilde{\boldsymbol{t}} = \widetilde{\boldsymbol{v}} \otimes \boldsymbol{u}$, $t_0 = v_0$, $\boldsymbol{t} = [\widetilde{\boldsymbol{t}}^{\mathrm{T}}, \mathrm{t}_0]^{\mathrm{T}}$, $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N]^{\mathrm{T}}$, $\boldsymbol{y}_i = \varphi_i[vec(\boldsymbol{A}_i), 1]^{\mathrm{T}}$ $(i = 1, 2, \ldots, N)$, then criterion function (18) of MatMHKS can be reformulated as below

$$\min_{\substack{\boldsymbol{t} \in R^{d+1}, \boldsymbol{b} > 0 \\ \boldsymbol{t} = \widetilde{\boldsymbol{v}} \otimes \boldsymbol{u}}} I(\boldsymbol{t}, \boldsymbol{b}) \triangleq (\boldsymbol{Y}\boldsymbol{t} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{t} - \boldsymbol{1}_{N\times 1} - \boldsymbol{b}) + c(u^{\mathrm{T}}\boldsymbol{S}_1\boldsymbol{u} + \widetilde{\boldsymbol{v}}^{\mathrm{T}}\boldsymbol{S}_2\widetilde{\boldsymbol{v}}). \tag{28}$$

Comparing to the criterion function of MHKS (11), the first terms of their right-handed sides are the same, and the second parts are regularization terms. The important difference of them is that $\boldsymbol{t}$ in (28) must satisfy a decomposability constraint of the Kronecker product, but $\boldsymbol{w}$ in (11) does not.

Until now, we get that MatMHKS is a MHKS imposed with Kronecker product decomposability constraint, in other words, in searching for its optimal weight vectors on the surface of the objective function (28), 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 $\boldsymbol{u}$ and $\boldsymbol{v}$, it is a point that makes it different from and may outperform in classification performance MHKS. More importantly, it can avoid overtraining due to (1) the aforementioned introduction of structural information such as in ORL and Letter data sets below; and (2) reduction of dimensionality for patterns, i.e., from pattern $\boldsymbol{A}$ with dimension of $d_1 \times d_2$ to $\boldsymbol{A}\boldsymbol{v}$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}$ with corresponding dimension of d_1 or d_2. In next section, we will present several methods of how to use local contextual information.

5. Experiments and discussion

In this section, we will examine the feasibility and practicality of MatMHKS through the experiments on the classification of real-world high-dimensional data. To evaluate the performance of MatMHKS, it was compared, respectively, with MHKS [18] and SVM [17].

5.1. Description of experimental data sets

These experiments are conducted on the benchmark data sets including both those in a matrix representation, including the ORL face[1] database and the Letter text-base,[2] and those in a vector representation from UCI data sets [19]. The details of the data sets are listed in Table 2.

5.2. Initialization and selection of parameters

In all experiments, the parameters involved include the margin $\boldsymbol{b}$, ξ, ρ, $\boldsymbol{u}$ and the regularization parameter c. We set all components of $\boldsymbol{b}(1) = 10^{-6}$, $\xi = 10^{-4}$, $\boldsymbol{\rho} = 0.99$, which are initialized by the same setup as in Ref. [18]. Commonly, $\boldsymbol{u}$ can be initialized with an arbitrary vector, but for simplicity and fairness of comparison among the different algorithms, we definitely initialize $\boldsymbol{u}(1) = \mathbf{1}_{d_1\times 1}$, and the regularization parameter c could be selected by cross-validation (CV), but when the dimensionality of a sample is very high and the size of the data set is too small simultaneously, the CV is not so practicable [20]. And there are several methods [16,21,22] for selecting the regularization parameter, those methods are based on the fixed margin vector. But during the iteration of MatMHKS, the margin is variable, so they can straightforwardly not be used here. We set the regularization parameter c in the range from 0 to 20 (step 1) and carry out MatMHKS on Letter with pattern of 4×108 and water-treatment with pattern of 2×19. The classification accuracies are shown in Fig. 1 from which it is clearly observed that for different data sets, the range of parameter c with best recognition accuracy is different, whereas for the c value in a specified range dependent on single data set such as water-treatment, the classification accuracy is almost unchanged.

In order to select such an appropriate regularization parameter, in this paper, we propose in turn another selection criterion defined as follows:

$$nC(u(c), \widetilde{v}(c)) = e^{\mathrm{T}}e + (u^{\mathrm{T}}S_1u + \widetilde{v}^{\mathrm{T}}S_2\widetilde{v}), \qquad (29)$$

where $u(c)$ and $\widetilde{v}(c)$ denote dependence on the c to be sought. Eq. (29) contains two terms: the first term is training error and the second is the regularization term. The criterion tries to make a trade-off between the training error and generalization. The selection algorithm for the c is summarized below:

(1) Initialize iteration index $i = 1$, $c(i) = 0$, total iteration number m, and selected classifier index $k = 0$, $err(0) = 10^8$,
(2) train a MatMHKS classifier, calculate its modified training error $err(i)$ defined by (29),
(3) if $err(i) < err(k)$, then $k = i$,
(4) if $c(i) = 0$, let $c(i+1) = 2^{-6}$, else $c(i+1) = c(i) \times 2$,
(5) $i = i + 1$, if $i \leqslant m$, go to (2),
(6) The final classifier is the kth classifier trained.

Consequently, this strategy makes the selected parameter able to minimize the criterion $nC(u(c), \widetilde{v}(c))$ and has an obvious advantage: when designing a classifier for M-class problem ($M > 2$), we adopt usually one-against-one or one-against-the rest classification strategy. Such a strategy gives rise to $M(M-1)/2$ or M2-class sub-classifiers in total. Leski [18] used the same regularization parameter for all the two-class sub-classifiers, and searched the best parameter in the range of 0 and 15 (step 0.2) by CV technique to achieve the best performance on the test set. Although restricting the regularization parameters for all the two-class sub-classifiers to be identical is simple for selection, in fact, (1) the same regularization parameter cannot guarantee that all the designed two-class sub-classifiers achieve their separate best performances; (2) even if different best parameters can be selected for sub-classifiers via his method, but due to (a) exhaustive searching in the specified range and (b) dependence of selected parameters on all sub-classifiers, it turns out more time-consuming and grows exponentially with the size of the specified range. Instead in this paper, we select a separate appropriate regularization parameter for each two-class classifier (sub-classifier) by the above selection algorithm to improve the generalization ability of each sub-classifier,

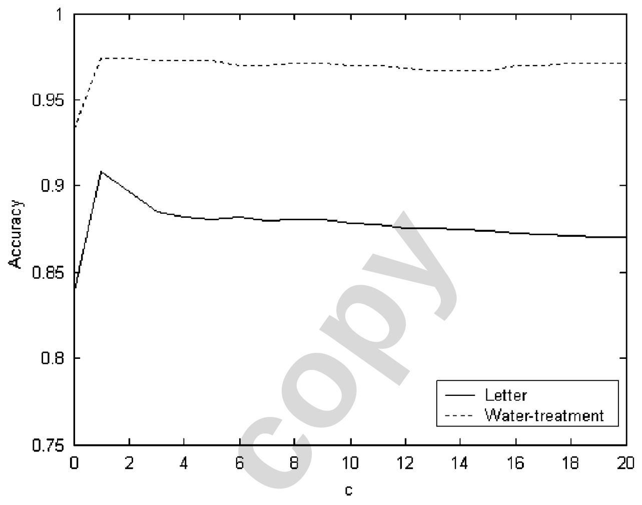

Fig. 1. Classification accuracies with the change of c for MatMHKS on Letter and water data sets.

[1] Available at http://www.cam-orl.co.uk.
[2] Available at http://sun16.cecs.missouri.edu/pgader/CECS477/NNdigits.zip.

Table 2
Detail of the data sets

Data set	Attributes	Classes	Sample number	Remark
ORL	28×23	40	400	10 Faces/class
Letter	24×18	10	500	50 Letters/class
Wine	12	3	178	59, 71 and 48 data, respectively, in class 1, 2, and 3.
Water-treatment	38	2	116	65 and 51 data, respectively, in class 1 and 2.
Sonar	60	2	208	111 and 97 data, respectively, in class 1 and 2.
Musk clean2 (166D)	166	2	6598	1017 and 5581 data, respectively, in class 1 and 2.
Musk clean2 (160D)	160	2	6598	The same data set in Musk clean2 (166D) with the last six dimensions of each pattern being omitted (by us) mainly for generating more matrix patterns.

Table 3
Classification performance (%) comparison on ORL and Letter by the different selecting-regularization-parameter methods

Data sets	Classifiers			
	Selection method in Ref. [18]		Our selection method	
	MHKS	MatMHKS	MHKS	MatMHKS
ORL	90.35 ($c = 0.8$)	90.50 ($c = 11$)	95.50	93.60
Letter	90.44 ($c = 0.1$)	86.39 ($c = 0.62$)	88.88	86.96

Table 4
Classification performance (%) comparison among MHKS, MatMHKS and SVM on UCI data sets

Data sets	Classifiers		
	MHKS	MatMHKS	SVM
Water-treatment	96.97	**98.03** (**2×19**)	97.12
Wine	92.453	90.85 (2×6)	91.89
		93.962 (**3×4**)	
Sonar	69.93	**77.41** (**6×10**)	76.30
		71.76 (5×12)	
		72.685 (4×15)	
		74.352 (3×20)	
		75.09 (2×30)	
Musk-clean 2 (160D)	**88.576**	84.321 (4×40)	82.76
		87.873 (2×80)	
Musk-clean 2 (166D)	**88.476**	87.763 (2×83)	82.73

The best result for each data set is denoted in bold.

thus, the generalization ability of the final classifier. And the parameter selection for each sub-classifier is unrelated so that its time complexity is linear growth only with the size of the specified range. To evaluate the availability of our selection strategy, we compare the parameter selection method in [18] with ours. Both MatMHKS and MHKS algorithms are, respectively, carried out with the two selection methods, and their corresponding classification accuracies on ORL and Letter are listed in Table 3. From the table, we can see that MatMHKS using our selection method can produce better or comparable classification accuracies on both data sets compared with MatMHKS using the selection method in Ref. [18], especially distinct on ORL (achieving 3.1%); while the same phenomenon can be observed for MHKS, its accuracy on ORL is increased by 5.15% but on Letter is slightly decreased. Here, it must point out that due to the different regularization parameters resulted from the different pair-wise sub-classifiers in our selection method, it is difficult for us to give a common best parameter for these sub-classifiers and thus we do not enumerate them in Table 3.

5.3. Experimental results of MHKS and MatMHKS

For multi-class problems, if the samples among classes are too unbalanced, we will use the appropriate strategies. For both Letter and ORL data sets, we take one-against-one classification strategy. For UCI data sets here, one-against-the-rest strategy is used. Each experiment on each data set is independently repeated for 10 times and the final classification result is the average accuracy of the 10 runs.

Table 4 demonstrates all classification accuracies on UCI [19] data sets used here. UCI data sets consist of all vector

Table 5
Classification performance (%) comparison among MHKS, MatMHKS with different reshaping and SVM

Data sets	Classifier				
	MHKS	MatMHKS			SVM
		Strategy1	Strategy2	Random	
ORL	95.5	93.6 (28 × 23) 94.7 (7 × 92) **96.7 (4 × 161)**	93.85 (28 × 23) 96.05 (4 × 161)	77.40 (28 × 23) 93.90 (4 × 161)	94.80
Letter	88.88	86.96 (24 × 18) 89.44 (16 × 27) 92.56 (4 × 108)	90.52 (24 × 18) 90.88 (16 × 27) **93.24 (4 × 108)**	79.68 (24 × 18) 80.72 (16 × 27) 88.52 (4 × 108)	91.72

The best result for each data set is denoted in bold.

patterns, thus in order to tailor them to MatMHKS, we must first transform these vector patterns to corresponding matrix patterns, for example, the 38 dimension vector pattern in water-treatment is transformed to the corresponding matrix pattern with dimension of 2×19 (alternatively 19×2). For these vector data set, generally there are many ways of matrixizing or reshaping them and thus the matrix patterns with different sizes can be formed for the same vector pattern. However, in this paper, our purpose to mainly validate the availability of the proposed MatMHKS, and thus we do not intend to enumerate all the possible matrices reshaped by the original vector. Therefore, for each vector data set, the dimension of the transformed matrices mainly used here are all listed in Table 4. From Tables 3 and 4, we can observe that compared with MHK, (1) for the vector patterns (UCI data sets), MatMHKS improves classification performance on three data sets (water-treatment, Wine, Sonar), especially distinct on Sonar (achieving 7.48%), but produces slightly degenerate performance on Musk-clean 2 (160D)(0.703%) and Musk-clean 2 (166D) (0.613%); (2) for the ORL and Letter image data sets, MatMHKS produces degeneration in classification performance to some degree (1.9% and 1.92%, respectively, on ORL and Letter), which seems counter-intuitive and uneasy to be understood and explained.

5.4. Further experiments on images with different reshaping

In last subsection, although getting better or comparable performance compared with MHKS, MatMHKS is found that the case cannot be hold for image patterns such as in ORL and Letter, inspired by MatMHKS' success on vector-to-matrix transform, in this subsection, we try to reorganize or reshape image patterns themselves into another new matrix patterns to examine the classification performance of MatMHKS. We still focus on ORL and Letter data sets. In reshaping them, without loss of generality we use three different strategies: (1) the first one is to first split each column of a given image (viewed as a matrix) into several equally-sized sub-columns, for example, three sub-columns each column, and then each sub-column becomes one column of a new matrix pattern in some order, for example, from top to bottom and from left to right; (2) the second one is to first partition a given image into several equally sized sub-blocks, such as 2×2, 2×3 or 3×3, then vectorize each such sub-block into a column vector by some technique like concatenation as one column of the new matrix pattern; (3) the third one is to randomly resample the elements of the original image to form a new matrix pattern.

Experimental results of MatMHKS and MHKS on these newly formed patterns of ORL and Letter data sets are listed in Table 5. First of all, we focus on the three different reshaping strategies in MatMHKS. Clearly, it can be found that for MatMHKS, the random strategy yields inferior performance to that with the first and second strategies, while the first strategy leads to the comparable performance to the second one. That this case occurs can attribute to that the random resampling breaks down more likely the spatial relation among elements of the original image than the first and second strategies. In other words, the first and second strategies emphasize the local information of the original image to greater extent. Therefore hereafter, we will mainly discuss MatMHKS using the first and second reshaping strategies.

Further, we find from Table 5 that through reshaping using the first or second strategy, MatMHKS improves classification accuracy distinctly on the two data sets. In such two reshaping strategies, for the same 4×161 reshaped pattern of ORL, the classification accuracies are, respectively, increased by 3.1% and 2.2% compared with non-reshaped case, and similarly, for 4×108 reshaped pattern of Letter, the classification accuracies are respectively increased by 5.6% and 2.72%. Moreover, compared with MHKS, MatMHKS gets better classification performance on both ORL (1.2%) and Letter (4.36%). Therefore, through the matrixization (from vector to matrix) and reshaping (from matrix to matrix) together, MatMHKS can obtain more significant performance gain than MHKS on both original vector patterns and image patterns. It is worthwhile pointing out that different reshaping for given images results in different classification performance of MatMHKS, thus final performance is reshaping-dependent.

5.5. Performance comparison among MatMHKS and SVM

In our experiment, we also compare our MatMHKS with support vector machine (SVM), which is one of the-state-of-the-art classifiers. Due to that both MatMHKS and MHKS both belong to linear classifier, we here also confine SVM to the linear type using the linear kernel for comparison fairness. From Tables 4 and 5, it can be found that MatMHKS with the optimally reshaped matrix pattern has the superior classification performance to SVM on all the data sets used here.

5.6. Further exploration

In this subsection, we will further explore the intuition behind why MatMHKS can get superior performance. In last subsection, we observed that MatMHKS produces better classification performance on almost all data sets used here than MHKS. However, we found that for image data sets, MatMHKS does not achieve the best classification performance on the original matrix pattern, e.g., on the 28×23 matrix pattern of ORL, beyond we expected. On the other hand, the experiments in Refs. [10–13] have shown that using those features extracted by 2D methods can improve generalization performance on ORL face data set of subsequently designed classifier (such as the nearest neighbor classifier). Therefore, we can basically conclude that the performance gains obtained by 2D feature extraction methods can entirely not contribute to the 2D extraction strategy rather than partially to dimensionality reduction for image. In other words, both dimensionality reduction and 2D methodology result in performance promotion *collectively*.

Contrarily, reshaping for the original image, e.g., from 28×23 to a new 4×161 matrix pattern of ORL, MatMHKS results in better classification performance.

Before exploration, we first reformulate the discrimination function of MatMHKS (13) and let $\boldsymbol{B} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{u}$, then (13) can be rewritten as $g(\boldsymbol{B}) = \boldsymbol{B}^{\mathrm{T}}\widetilde{\boldsymbol{v}} + v_0 = \widetilde{\boldsymbol{v}}^{\mathrm{T}}\boldsymbol{B} + v_0$, formally, it is a similar form as that of MHKS in which the $\boldsymbol{B}$ is an input to the MHKS. We decompose $\boldsymbol{B}$ in terms of

$$\boldsymbol{B} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{u} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{d_2}]^{\mathrm{T}}\boldsymbol{u} = [\boldsymbol{a}_1^{\mathrm{T}}\boldsymbol{u}, \boldsymbol{a}_2^{\mathrm{T}}\boldsymbol{u}, \ldots, \boldsymbol{a}_{d_2}^{\mathrm{T}}\boldsymbol{u}]^{\mathrm{T}}, \tag{30}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{d_2}]$ and $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \ldots, a_{id_1}]^{\mathrm{T}}$, $i = 1, 2, \ldots, d_2$, $\boldsymbol{B} = [b_1, b_2, \ldots, b_{d_1}]^{\mathrm{T}}$, $b_j = \boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{u}$, $j = 1, 2, \ldots, d_1$. Thus, each component of the new input $\boldsymbol{B}$ is a linear combination of all the components of each column of original matrix $\boldsymbol{A}$, implying that it integrates global information in each column (coined column-global information) and thus de-emphasizes local information in each column (coined column-local information). If we, instead, reshape the original pattern $\boldsymbol{A}$ with dimension $d_1 \times d_2$ to a new matrix pattern $\boldsymbol{C}$ with dimension $r \times c$, then still let $\boldsymbol{B} = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{u}$, similar to the above analysis, each component of the $\boldsymbol{B}$ is a linear combination of all the components of each column of $\boldsymbol{C}$. Now without loss of generality, it is supposed to use the first reshaping strategy in last subsection, in this case, each column of the $\boldsymbol{C}$ is a sub-column of original pattern $\boldsymbol{A}$, and thus each component of the $\boldsymbol{B}$ is a linear combination of all the components of a sub-column of $\boldsymbol{A}$, implying that it just integrates column-local information rather than column-global information and thus emphasizes local information in each column. Therefore, a possible reason of yielding the above phenomenon in image recognition can be intuitively attributed to such a fact that the reshaping from matrix pattern to another matrix pattern may destroy the whole or global structure in each column of original pattern, but partial or local structure or column-local information can likely be kept and emphasized contrarily, which could be more useful for discrimination [5].

In a word, compared with MHKS, MatMHKS may get guided by *a priori* knowledge of the specific problem but in an implicit way. It is such an incorporation of both matrixization and reshaping to vector and matrix patterns that MatMHKS becomes more flexible and effective for classification.

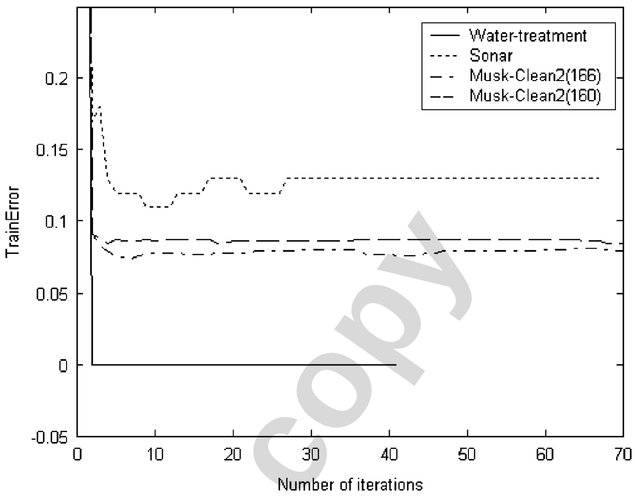

Fig. 2. Training error changes with the iteration number of MatMHKS, respectively, on water-treatment, Sonar, Musk-clean2 (166D) and Musk-clean (160D).

5.7. On convergence of MatMHKS

In this subsection, we give a discussion on the convergence of the proposed MatMHKS algorithm. Here, thanks to some difficulty in theoretical proof, we adopt an empirical means as used in [18,23] to demonstrate that MatMHKS can converge in the limited iterations. Fig. 2 shows the training error changes with the iteration number of MatMHKS, respectively, on water-treatment, Sonar, Musk-clean2 (166D) and Musk-clean (160D). From the figure, it can be found that all the training errors on the four data sets can obviously converge to a certain stable value after the limited time iterations, concretely, 1, 30, 50, 30 iterations respectively for water-treatment, Sonar, Musk-clean2 (166D) and Musk-clean (160D). Consequently, the convergence behavior of the proposed MatMHKS algorithm is demonstrated.

6. Conclusion and future work

Inspired by the idea of 2D feature extractions, in this paper, we proposed a new classifier design method, matrix-pattern-oriented two-sided linear classifier or MatMHKS, extended and developed MHKS algorithm [18]. MatMHKS is particularly suitable to high dimension patterns including image pattern. Compared with MHKS, MatMHKS not only reduces the memory required for the weight vectors but also achieves better generalization performance. And it is worthwhile to point out that the performance gain of MatMHKS is not resulted from popular non-linearization techniques [24] but simply from the matrixization for vector patterns or reshaping for image patterns. This may be another way with low cost to promote classification performance.

Another conclusion we can get is that the vector pattern representation is not always optimal, for example in some case, if the dimension of the pattern is high, the matrix pattern representation may be better. This also directly provides a necessary validation for the Ugly Duckling Theorem [2, Chapter 9].

In addition, our experiment also illustrates that the performance gains obtained by 2D feature extraction methods can entirely not contribute to the 2D extraction strategy rather than partially to dimensionality reduction for image.

In future work, we need to research several problems below: (1) how to matrixize a vector pattern more appropriately to tailor to the inherent but implicit representation of pattern under study makes prior knowledge reflected in the training; (2) extend MatMHKS to a corresponding nonlinear counterpart by such some methods as kernel method [25] although this work is, at present, not so easy as in vector case.

Acknowledgment

The authors thank Natural Science Foundations of China under Grant No. 60473035, and Jiangsu Natural Science Foundation under Grant No. BK2004001 for supports respectively.

References

[1] O. Bousquet, S. Boucheron, G. Lugosi, Introduction to statistical learning theory, Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence, vol. 3176, Springer, Berlin, 2004, pp. 169–207.
[2] R.O. Duda, P.E. Hart, D.G. Stock, Pattern Classification, second ed., Wiley, New York, 2001.
[3] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Networks 12 (2) (2001) 181–202.
[4] W.H. Highleyman, Linear decision functions, with application to pattern recognition, Proc. IRE 50 (1962) 1501–1514.
[5] R. Maree, P. Geurts, J. Piater, L. Wehenkel, Random subwindows for robust image classification, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR05), 2005.
[6] H. Wang, N. Ahuja, Rank-R approximation of tensors: using image-as-matrix representation, IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[7] D. Xu, S. Yan, L. Zhang, H.J. Zhang, Z. Liu, H.Y. Shum, Concurrent subspace analysis, IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[8] H. Kong, L. Wang, E.K. Teoh, J.G. Wang, R. Venkateswarlu, A framework of 2D Fisher discriminant analysis: application to face recognition with small number of training samples, IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[9] S.C. Yan, D. Xu, L. Zhang, B.Y. Zhang, H.J. Zhang, Coupled kernel-based subspace learning, IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[10] J. Yang, D. Zhang, A.F. Frangi, J.Y. Yang, Two-dimensional PCA: a new approach to appearance-based face representation and recognition, IEEE Trans. PAMI 26 (1) (2004) 131–137.
[11] J. Ye, R. Janardan, Q. Li, Two-dimensional linear discriminant analysis, University of Minnesota, NIPS2004.
[12] M. Li, B. Yuan, 2D-LDA: a statistical linear discriminant analysis for image matrix, Pattern Recognition Lett., 2005.
[13] S. Chen, Y. Zhu, D. Zhang, J.-Y. Yang, Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA, Pattern Recognition Lett. 26 (8) (2005) 1157–1167.
[14] F.W. Smith, Design of multicategory pattern classifiers with two-category classifier design procedures, IEEE Trans. Comput. C-18 (6) (1969) 548–551.
[15] L.F. Chen, H.Y.M. Liao, M.T. Ko, J.C. Lin, G.J. Yu, A new LDA-based face recognition system which can solve the small sample size problem, Pattern Recognition 33 (10) (2000) 1713–1726.
[16] K. Hagiwara, Regularization learning, early stopping and biased estimator, Neurocomputing 48 (1–4) (2002) 937–955.
[17] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[18] J. Leski, Ho–Kashyap classifier with generalization control, Pattern Recognition Lett. 24 (14) (2003) 2281–2290.
[19] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, Available from: ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩, 1998.
[20] K. Namwoon, H.J. Kyung, Assessing the integrity of cross-validation: a case for small sample-based research, in: Marketing Working Paper Series, Series No.: MKTG 97.096, Hong Kong: The Hong Kong University of Science & Technology UST Marketing Department, 1997.
[21] M. Sugiyama, H. Qgawa, Option design of regularization term and regularization parameter by subspace information criterion, Neural Networks 15 (2002) 349–361.
[22] C.-T. Leung, T.W.S. Chow, Adaptive regularization parameter selection method for enhancing generalization capability of neural networks, Artif. Intell. 107 (1999) 347–356.
[23] J. Ye, Generalized low rank approximation of matrices, Mach. Learning 61 (1–3) (2005) 167–191.
[24] J. Leski, Kernel Ho–Kashyap classifier with generalization control, Int. J. Appl. Math. Comput. Sci. 14 (1) (2004) 53–61.
[25] B. Scholkopf, A. Smola, K.-R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.

About the Author—SONGCAN CHEN received the B.Sc. degree in mathematics from Hangzhou University (now merged into Zhejiang University) in 1983. In December 1985, he completed the M.Sc. degree in computer applications at Shanghai Jiaotong University and then worked at Nanjing University of Aeronautics & Astronautics (NUAA) in January 1986 as an assistant lecturer. There he received a Ph.D. degree in communication and information systems in 1997. Since 1998, as a full professor, he has been with the Department of Computer Science and Engineering at NUAA. His research interests include pattern recognition, machine learning and neural computing. In these fields, he has authored or coauthored over 70 scientific journal papers.

About the Author—ZHE WANG received a B.Sc. in computer science from Nanjing University of Aeronautics & Astronautics (NUAA), China, in 2003. Currently he is a Ph.D. student at the Department of Computer Science & Engineering, NUAA. His research interests focus on Neural Computing and Pattern Recognition.

About the Author—YONGJUN TIAN received a B.Sc. in Southern Yangze University, China, in 2003. Currently he is a Master student at the Department of Computer Science & Engineering, NUAA. His research interests focus on Neural Computing and Pattern Recognition.