Knowledge-Based Systems 37 (2013) 388–393

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

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

Random projection ensemble learning with multiple empirical kernels

Zhe Wang [a,*], Wenbo Jie [a], Songcan Chen [b], Daqi Gao [a]

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

ARTICLE INFO

Article history:
Received 24 March 2012
Received in revised form 11 July 2012
Accepted 20 August 2012
Available online 12 October 2012

Keywords:
Multiple kernel learning
Empirical mapping
Random projection
Ensemble classifier
Pattern recognition

ABSTRACT

In this paper we propose an effective and efficient random projection ensemble classifier with multiple empirical kernels. For the proposed classifier, we first randomly select a subset from the whole training set and use the subset to construct multiple kernel matrices with different kernels. Then through adopting the eigendecomposition of each kernel matrix, we explicitly map each sample into a feature space and apply the transformed sample into our previous multiple kernel learning framework. Finally, we repeat the above random selection for multiple times and develop a voting ensemble classifier, which is named RPEMEKL. The contributions of the proposed RPEMEKL are: (1) efficiently reducing the computational cost for the eigendecomposition of the kernel matrix due to the smaller size of the kernel matrix; (2) effectively increasing the classification performance due to the diversity generated through different random selections of the subsets; (3) giving an alternative multiple kernel learning from the Empirical Kernel Mapping (EKM) viewpoint, which is different from the traditional Implicit Kernel Mapping (IKM) learning.

1. Introduction

Kernel-based learning is successfully applied in machine learning [10,11,14,15,19,20]. The kernel-based learning transforms the input space into a feature space and works in the feature space, where the transformation is achieved through the mapping $\Phi(x): x \to \mathcal{F}$. In the existing kernel-based learning, there are two kinds of $\Phi(x)$ including implicit and explicit representations denoted as $\Phi^i(x)$ and $\Phi^e(x)$. The implicit mapping $\Phi^i(x)$ called Implicit Kernel Mapping (IKM) [10] is achieved through introducing a kernel function $k(x_i,x_j)$ that can determine the geometrical structure of the mapped data in the feature space, where we need not obtain the form of the $\Phi^i(x)$ but just compute the $k(x_i,x_j) = \Phi^i(x_i) \cdot \Phi^i(x_j)$. In contrast, the $\Phi^e(x)$ called Empirical Kernel Mapping (EKM) [16] should explicitly give all the features of x in the mapped Φ^e-space.

On the other hand, the kernel-based learning can also be sorted into Single Kernel Learning (SKL) and Multiple Kernel Learning (MKL) according to the number of the used kernels in learning processing. SKL selects one optimized kernel from a set of candidates. MKL syncretizes and uses multiple kernels in learning processing. Therefore, in our opinion there are naturally at least four combinations: the SKL with IKM, the SKL with EKM, the MKL with IKM and the MKL with EKM also denoted as MEKL. In traditional kernel-based methods, the SKL with either IKM or EKM is widespread [10,16]. Since MKL can increase performance effectively, the MKL with IKM has recently got much attention [7,13]. In contrast, It is less attracted for the MKL with EKM except our previous work MultiK-MHKS [14].

The MultiK-MHKS explicitly maps the input samples into M feature spaces with given M different kernels. Then, it constructed a term R_{IFSL} called Inter-Function Similarity Loss and introduced R_{IFSL} into a regularization framework. In the MultiK-MHKS, we explicitly mapped each sample $x \in \mathbb{R}^d$ into $\Phi^e(x) \in \mathbb{R}^r$ through the whole training set $S = \{(x_i, \varphi_i)\}_{i=1}^N$ for M times with the given M kernels. With each kernel $k(x_i,\ x_j)$, we generated the kernel matrix $K \in \mathbb{R}^{N\times N}$ with the set S and carried out the eigendecomposition of the K as follows

$$K = Q_{N\times r}\Lambda_{r\times r}Q^T_{r\times N}, \tag{1}$$

where r is the rank of the K. Then through letting $R = Q\Lambda^{1/2}$, each sample x was explicitly mapped through the following form

$$\Phi^e(x)_{r\times 1} = R^{-1}_{r\times N}[k(x,x_1),\dots,k(x,x_N)]^T_{N\times 1}. \tag{2}$$

It can be found that it would take a computational cost with $o(N^3)$ for the Eq. (1). If the N is large, it would be much larger for the computational cost of the eigendecomposition of the K.

To reduce the cost for the eigendecomposition of the K, in this paper we first adopt the strategy of the Random Projection (RP). RP aims to project the original high-dimensional data onto a lower-dimensional space with a random matrix [1–3]. From the

* Corresponding author. Tel.: +86 15000779526.
E-mail addresses: wangzhe@ecust.edu.cn (Z. Wang), pilixiaoxuanfeng@gmail.com (W. Jie), s.chen@nuaa.edu.cn (S. Chen), gaodaqi@ecust.edu.cn (D. Gao).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.08.017

Eq. (2), it is found that the sample x is transformed into a r-dimensional kernel space through the whole training set S with the size N. Here, based on the random characteristic of RP, we randomly select a subset $S' = \{(x_i, \varphi_i)\}_{i=1}^{p}$, $p < N$ from the whole set S and use the S' to construct M kernel matrices $K_l \in \mathbb{R}^{p\times p}$ with M different kernels. Through adopting the eigendecomposition for the K_l instead of the original K, we explicitly map each sample x into $\Phi_l^e(x), l = 1\ldots M$ based on the Eq. (2). Due to changing the transformed size for the x in the Eq. (2), we can reduce the computational cost of the matrix decomposition from $o(N^3)$ to $o(p^3)$. Subsequently, we apply the transformed sample $\Phi_l^e(x), l = 1\ldots M$ into our previous MEKL framework [14].

Secondly, in order to prevent the proposed classifier from reducing classification performance, we further give an ensemble scheme with voting. In detail, we randomly select the subset S' for multiple times and generate multiple $S'_q, q = 1\ldots H$. Subsequently, for each $S'_q, q = 1\ldots H$ we repeat the explicitly mapping (2) and the following learning applied into our previous MEKL [14]. The final ensemble is produced through a voting scheme that is applied to the classification outcomes of all the classifiers generated from $S'_q, q = 1\ldots H$. The whole procedure is named Random Projection Ensemble Learning with Multiple Empirical Kernels denoted RPEMEKL for short. The advantages of the proposed RPEMEKL are highlighted as follows: (1) reducing the computational cost for the eigendecomposition of the kernel matrix from $o(N^3)$ to $o(Hp^3)$; (2) increasing the classification performance due to the diversity generated through different random selections of the S'_qs; (3) giving an alternative multiple kernel learning from the Empirical Kernel Mapping (EKM) viewpoint, which is different from the traditional Implicit Kernel Mapping (IKM) learning.

The rest of this paper is organized as follows. Section 2 gives the architecture of the proposed RPEMEKL. Section 3 demonstrates the feasibility and effectiveness of the proposed RPEMEKL in terms of classification and computation. Finally, both conclusion and future work are given in Section 4.

2. Random Projection Ensemble Learning with Kernels (RPEMEKL)

The proposed RPEMEKL is made up of two parts. The first part is to randomly select a subset S' from the whole set S based on the random characteristic of RP and apply the transformed samples from the S' onto our previous MEKL framework [14]. The second part is to repeat the first part for multiple times based on different random selections for S' and to construct a voting ensemble with different classifiers generated from the first part.

Suppose that there is a training sample set $S = \{(x_i, \varphi_i)\}_{i=1}^{N} \subset \mathcal{X}$, where $x_i \in \mathbb{R}^d$ and its corresponding class label $\varphi_i \in \{+1, -1\}$. In the first part, we randomly select a sample subset $S' = \{(x_i, \varphi_i)\}_{i=1}^{p}$ from the whole set S. Then we adopt the subset S' to generate the kernel matrix $K_l \in \mathbb{R}^{p\times p}$ with one certain kernel $k_l(x_i, x_j)$ and carry out the eigendecomposition of each K_l as follows

$$K_l = Q_l \Lambda_l Q_l^T, \tag{3}$$

where $\Lambda_l \in \mathbb{R}^{r_l\times r_l}$ is a diagonal matrix consisting of the r_l positive eigenvalues of K_l, and $Q_l \in \mathbb{R}^{p\times r_l}$ consists of the corresponding orthonormal eigenvectors. Here we adopt the EKM for each mapping and thus each input sample x can be mapped into $\Phi_l^e(x)$ through the following equation

$$\Phi_l^e(x) = \Lambda_l^{-1/2} Q_l^T [k(x, x_1), \ldots, k(x, x_p)]^T, \tag{4}$$

where $\Phi_l^e(x) \in \mathbb{R}^{r_l}$. The feasibility of doing so can be guaranteed by the characteristic of RP in kernel-based learning, which is that the learning based on the mapping (4) can lead to an approximate separability at one certain margin [3]. Since our previous work MultiK-MHKS [14] makes the p in the (4) with the size of the whole input training samples N and in this paper we adopt a smaller p, the computational cost for the eigendecomposition in the Eq. (3) can decrease from $o(N^3)$ in the MultiK-MHKS to $o(p^3)$ in the proposed method.

Further, the proposed method explicitly maps all the input samples of the set S into the transformed sample set $\{(\Phi_1^e(x_i), \ldots, \Phi_l^e(x_i), \ldots, \Phi_M^e(x_i))\}_{i=1}^{N}$ with the Eq. (4) and M given kernels. Then, we introduced the generated $\{(\Phi_1^e(x_i), \ldots, \Phi_l^e(x_i), \ldots, \Phi_M^e(x_i))\}_{i=1}^{N}$ onto our previous MEKL framework. In detail, we adopt the Modification of Ho-Kashyap algorithm with Squared approximation of the misclassification errors (MHKS) [8] as the base classifier. Therefore for the proposed RPEMEKL, we give the following optimization problem:

$$\min_{\substack{\omega_l \in \mathbb{R}^{r_l+1}, b_l \geqslant 0\\ l=1\ldots M}} L = \sum_{l=1}^{M}\Big[(Y_l\omega_l - 1_{N\times1} - b_l)^T(Y_l\omega_l - 1_{N\times1} - b_l) + c_l\tilde{\omega}_l^T\tilde{\omega}_l\Big] + \lambda\sum_{l=1}^{M}\Big(Y_l\omega_l - \frac{1}{M}\sum_{j=1}^{M}Y_j\omega_j\Big)^T\left(Y_l\omega_l - \frac{1}{M}\sum_{j=1}^{M}Y_j\omega_j\right), \tag{5}$$

where $\omega_l = [\tilde{\omega}_l{}^T, \omega_{0l}]^T$, $\tilde{\omega}_l \in \mathbb{R}^{r}_{l}, \omega_{0l} \in \mathbb{R}$ are the augmented weight vector, the weight vector and the bias respectively; the matrix Y_l is defined as $Y_l = \left[\varphi_1\left(\Phi_l^e(x_1)^T, 1\right); \ldots; \varphi_N\left(\Phi_l^e(x_N)^T, 1\right)\right]$; $1_{N\times1}$ represents the vector of N dimension with all entries equal to 1; $b_l \in \mathbb{R}^{N\times1}$ represents the vector with all entries equal to nonnegative values and the regularized parameters $c_l, \lambda \geqslant 0 \in \mathbb{R}$. In the optimization problem (5), Y_l, ω_l, b_l correspond to one MHKS in one Φ_l^e-space that is determined by the corresponding $\{(\Phi_l^e(x_i), \varphi_i)\}_{i=1}^{N}$. In the right side of the Eq. (5), the first term corresponds to the principle of MHKSs in M views. The second term is to syncretize the M MHKSs, which denotes that the outputs of the samples $\{(\Phi_l^e(x_i), \varphi_i)\}_{i=1}^{N}$ in each Φ_l^e-space onto their corresponding weight vector ω_l are constrained to be maximally close to the average outputs of all the feature spaces.

In the optimizing processing for the (5), we employ a modification of the gradient descent with a heuristic update-rule for each ω_l. Through making the gradient of the L of the (5) with respect to the ω_l be zero, we can obtain

$$\frac{\partial L}{\partial \omega_l} = 0 \Longleftrightarrow \omega_l = \left[\left(1 + \lambda\frac{M-1}{M}\right)Y_l^TY_l + c_l\tilde{I}_l\right]^{-1} Y_l^T\left(b_l + 1_{N\times1} + \lambda\frac{1}{M}\sum_{j=1;j\neq l}^{M}Y_j\omega_j\right), \tag{6}$$

where $\tilde{I}_l$ is a diagonal matrix with full ones except the last element set to zero. In the lth Φ_l^e-space, it can be noted that ω_l depends on b_l from the Eq. (6). Then by differentiating the L in Eq. (5) with respect to b_l and setting the result equal to zero, we can get

$$\frac{\partial L}{\partial b_l} = 0 \Longleftrightarrow e_l = Y_l\omega_l - b_l - 1_{N\times1}. \tag{7}$$

Then, we adopt the iterative algorithm for determining ω_l and b_l similarly to that in [8]. First, with b_l^k representing the vector b_l at the kth iteration, we initialize $b_l^1 \geqslant 0$, then keep $b_l^k \geqslant 0$ at each iteration, and finally obtain

$$\begin{cases} b_l^1 \geqslant 0 \\ b_l^{k+1} = b_l^k + \rho_l(e_l^k + |e_l^k|) \end{cases}, \tag{8}$$

Table 1
Algorithm: RPEMEKL.

```
Input: the training sample set $S = \{(x_i, \varphi_i)\}_{i=1}^{N}$; the $M$ kernels $\{k_l(\cdot,\cdot)\}_{l=1}^{M}$.
For $q = 1 \dots H$,
  1. Randomly select a subset set $S'_q = \{(x_i, \varphi_i)\}_{i=1}^{p}$ from the $S$;
  2. Explicitly map $\{x_i\}_{i=1}^{N}$ into $\left\{\Phi_1^e(x_i), \dots, \Phi_l^e(x_i), \dots, \Phi_M^e(x_i)\right\}_{i=1}^{N}$ by $M$ kernels as shown in the Eq. (4).
  3. For each $\Phi_l^e$, $Y_l = \left[\varphi_1\left(\Phi_l^{e^T}(x_1), 1\right); \dots; \varphi_N\left(\Phi_l^{e^T}(x_N), 1\right)\right], l = 1 \dots M$.
  4. Initialize $\xi > 0, \rho_l > 0, c_l \geqslant 0, b_l^1 \geqslant 0, l = 1 \dots M$ and $\omega_l^1, l = 2 \dots M$; set the iteration index $k = 1$.
  5. $\omega_l^k = \left((1 + \lambda\frac{M-1}{M})Y_l^T Y_l + c_l\tilde{I}\right)^{-1} Y_l^T\left(b_l^k + 1_{N\times1} + \lambda\frac{1}{M}\sum_{j=1;j\neq l}^{m} Y_j\omega_j^k\right), l = 1$.
  6. While $\|L^{k+1} - L^k\|_2 > \xi$
    i. $e_l^k = Y_l\omega_l^k - b_l^k - 1_{N\times1}, l = 1 \dots M$;
    ii. $b_l^{k+1} = b_l^k + \rho_l(e_l^k + |e_l^k|), l = 1 \dots M$;
    iii. $\omega_l^{k+1} = \left((1 + \lambda\frac{M-1}{M})Y_l^T Y_l + c_l\tilde{I}\right)^{-1} Y_l^T\left(b_l^{k+1} + 1_{N\times1} + \lambda\frac{1}{M}\sum_{j=1;j\neq l}^{m} Y_j\omega_j^k\right), l = 1 \dots M$;
    iv. $k = k + 1$;
  end of while
  7. Get a hypothesis $F_q(x_i) = sign\left(\frac{1}{M}\sum_{l=1}^{M}\omega_l^T\left[\Phi_l^{e^T}(x_i), 1\right]^T\right)$;
end of for
Output: the final hypothesis:
  $F_f(x_i) = \arg\max_{\varphi_i\in\{+1,-1\}} \sum_{q:F_q(x_i)=\varphi_i} 1$
```

where at the kth iteration, the error vector of the lth Φ_l^e-space e_l^k is defined as $e_l^k = Y_l\omega_l^k - b_l^k - 1_{N\times1}$, and the learning rate of the lth Φ_l^e-space $\rho_l > 0$. Then ω_l^{k+1} can be given by the Eq. (6). In practice, the termination criterion can be designed as $\|L^{k+1} - L^k\|_2 \leqslant \xi$. For the input sample x_i with its corresponding mapped forms $\left\{\Phi_l^e(x_i)\right\}_{l=1}^{M}$ generated through the Eq. (4), we can give the decision function in one random selection of S' as follows:

$$F(x_i) = sign\left(\frac{1}{M}\sum_{l=1}^{M}\omega_l^T\left[\Phi_l^{e^T}(x_i), 1\right]^T\right), \tag{9}$$

where the $sign(z) = 1$, if $z > 0$; otherwise $sign(z) = -1$.

In our opinion, each random selection of S' would give a possible region in the whole solution space. Different S' could expand the searching region for the final solutions and give a diversity that is required by ensemble learning. Thus, in the second part we randomly select the subset S' for multiple times and generate multiple $S'_q, q = 1 \dots H$. Subsequently, for each $S'_q, q = 1 \dots H$ we repeat the explicitly mapping (4) and the following optimization for the (5). In this case, we can get multiple functions F_q, $q = 1 \dots H$ as shown in the (9). The final ensemble classifier is produced through a voting scheme that is applied to the classification outcomes of all the classifiers generated from $S'_q, q = 1 \dots H$. Therefore, for the input sample x_i with the $\left\{\Phi_l^e(x_i)\right\}_{l=1}^{M}$, we can give the final decision function of the proposed RPEMEKL as follows:

$$F_f(x_i) = \arg\max_{\varphi_i\in\{+1,-1\}} \sum_{q:F_q(x_i)=\varphi_i} 1. \tag{10}$$

The whole procedure of the proposed RPEMEKL is summarized in Table 1.

3. Experiments

3.1. Experimental setting

In this section, we compare the RPEMEKL with its original model MultiK-MHKS and some other state-of-the-art MKL including SVM-2K [4], MKDA(SDP) [6], and the efficient MKL implementations with SILP [12] in terms of classification accuracy and training time so as to validate its effectiveness and efficiency. In our experiment, the used kernels for the proposed RPEMEKL and the MultiK-MHKS are the linear kernel $k(x_i, x_j) = x_i x_j$ and the two RBF kernels $k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$. In practice, we normalize the linear kernel matrix with the following method [11]. Given an unnormalized kernel matrix K, we would like to construct the normalized $\tilde{K}$. Note that $\tilde{K}_{ij} = K_{ij}/\sqrt{K_{ii}K_{jj}}$. Define $k = (1/\sqrt{K_{11}}, \dots, 1/\sqrt{K_{nn}})$. Then, $\tilde{K} = K \cdot (kk^T)$, where $\cdot$ denotes element-wise product. The kernel parameter σ for the first RBF kernel is set to the average value of all the ℓ_2-norm distances $\|x_i - x_j\|_2, i, j = 1, \dots, N$ and the σ for the second one is set to one tenth of the first σ. Since SVM-2K only can deal with two kernels in one learning process, we give the best accuracy corresponding to the optimal combination from the three candidate kernels for all the used data sets. MKDA(SDP) still follows the setting of kernels in the literature [6]. In the RPEMEKL and MultiK-MHKS, the margin vector b_l is initialized to 10^{-6}, the ξ in the termination criterions is fixed to 10^{-3}, the learning ratio ρ is set to 0.99, and the ω_l, $l = 2, \dots, M$ are initialized to one unit vector, respectively. Both the regularization parameters c and λ are optimized from the set $\{2^{-4}, 2^{-3}, \dots, 2^3, 2^4\}$. For the RPEMEKL, we set $H = 3$ and carry out the RPEMEKL with $p = N/10$ and the optimized p, where the p is selected form the set $\{0.1, 0.2, 0.3, 0.4, 0.5\} \times N$. The one-against-one classification strategy is adopted for the multi-class classification problem. The adopted benchmark data are obtained from [5] including: *Arrhythmia* (279 Attributes/16 Classes/452 Samples), *Clean* (166/2/476), *Glass* (10/6/214), *House-votes* (16/2/435), *Ionosphere* (34/2/351), *Iris* (4/3/150), *Letter* (432/10/500), *Pima* (8/2/768), *Wine* (12/3/178), *Segmentation* (19/7/2310), *Page-block* (10/5/5473), *Waveform* (21/3/5000), *Cmc* (9/3/1473), and *Sonar* (60/2/208). All computations are run on *IBM X3650M*$_2$ *Server* with four Intel Xeon 2.00 GHz processors and 6GB ram running Windows Server 2008 R2 Operating System and MATLAB environment.

3.2. Classification performance comparison

In this section, we carry out the RPEMEKL, the MultiK-MHKS, the classical MKL algorithm SVM-2K [4], the state-of-the-art

Table 2
Classification accuracy (%) and p-value comparison between RPEMEKL with the optimized p and $p = 0.1$, MultiK-MHKS, SVM-2K, MKDA(SDP), and MKL [12].

	RPEMEKL	MultiK-MHKS	SVM-2K	MKDA(SDP)	MKL
	Accuracy(optimized p)	Accuracy	Accuracy	Accuracy	Accuracy
	Accuracy($p = 0.1$)	p-value	p-value	p-value	p-value
Cmc	*51.74 ± 1.53*	49.73 ± 1.06*	47.06 ± 1.68*	49.12 ± 1.29*	29.37 ± 9.28*
	50.83 ± 1.44	0.0223	0.0000	0.0032	0.0000
Glass	95.24 ± 2.87	95.43 ± 2.37	*97.90 ± 1.33*	93.69 ± 4.67*	97.71 ± 1.63
	93.52 ± 2.97	0.3217	0.1595	0.0000	0.0618
House-votes	92.17 ± 1.89	91.84 ± 2.12	87.24 ± 1.39*	38.46 ± 0.00*	*94.29 ± 1.43*
	90.60 ± 1.90	0.8733	0.0159	0.0000	0.2588
Ionosphere	*89.60 ± 2.05*	87.60 ± 2.35	83.20 ± 7.66	63.81 ± 0.00*	70.40 ± 1.73*
	87.43 ± 4.22	0.9232	0.9589	0.0000	0.0000
Clean	80.55 ± 2.35	79.16 ± 1.56	55.54 ± 1.85*	56.64 ± 0.00*	89.49 ± 1.81*
	70.59 ± 3.32	0.6264	0.0000	0.0000	0.0000
Iris	96.00 ± 1.54	95.07 ± 2.18*	96.67 ± 1.30*	96.22 ± 2.35*	*98.00 ± 0.94**
	95.20 ± 0.93	0.0289	0.0362	0.0000	0.0023
Letter	82.96 ± 2.30	68.48 ± 3.71	69.60 ± 2.34*	90.20 ± 2.61*	*92.72 ± 0.96**
	79.68 ± 3.40	0.0935	0.0302	0.0000	0.0000
Pima	74.82 ± 1.63	73.85 ± 2.14	65.10 ± 1.75	65.22 ± 0.00*	*76.25 ± 1.26**
	71.77 ± 2.04	0.3794	0.0723	0.0000	0.0017
Page-block	93.45 ± 3.61	91.34 ± 3.34	89.80 ± 0.15*	85.74 ± 1.16*	*94.01 ± 2.17**
	92.99 ± 0.42	0.2374	0.0000	0.0000	0.0000
Arrhythmia	62.61 ± 1.04	57.84 ± 1.69*	55.76 ± 4.22	63.88 ± 4.70*	*64.86 ± 1.69*
	56.85 ± 0.84	0.0000	0.0604	0.0000	0.0532
Wine	95.30 ± 2.69	92.61 ± 3.76*	80.00 ± 4.05*	*96.42 ± 2.88**	95.29 ± 2.57
	75.80 ± 6.34	0.0000	0.0000	0.0000	0.4961
Segmentation	94.70 ± 0.50	94.70 ± 0.69	83.97 ± 1.28*	*97.07 ± 0.61**	92.15 ± 1.94*
	93.19 ± 0.59	0.7519	0.0000	0.0000	0.0085
Sonar	75.44 ± 4.62	76.92 ± 2.39	58.06 ± 4.19*	79.13 ± 4.85*	*85.63 ± 2.39**
	65.92 ± 3.12	0.2162	0.0000	0.0081	0.0000
Waveform	83.44 ± 4.63	85.95 ± 1.67	65.37 ± 1.12*	74.57 ± 3.87*	*86.79 ± 3.67**
	75.30 ± 0.47	0.0551	0.0000	0.0000	0.0336

Note: The best accuracy of each data set is in italic. The p-values are from a t-test comparing each classifier to RPEMEKL.
* Difference from RPEMEKL is significant at 5% significance level, i.e. p-value less than 0.05.

multiple kernel discriminant analysis algorithms MKDA using SDP [6], and the efficient MKL implementations with SILP [12] and discuss their classification performance. For each used data set, the classification accuracies on the validation sets generated by the n-folds MCCV [17] are averaged and reported in Table 2. For n-folds MCCV, we randomly split the data set into two parts including the training and validation sets, and repeat the procedure n times. In our experiments, n is set to 10. In Table 2, the best results of the different methods are in bold. In addition to reporting the average accuracies, we also perform the paired t-test [9] by comparing the RPEMEKL with the other classifiers including the MultiK-MHKS, SVM-2K and MKDA(SDP). The null hypothesis H_0 demonstrates that there is no significant difference between the mean number of samples correctly classified by RPEMEKL and the other classifiers. Under this assumption, the p-value of each test is the probability of a significant difference in correctness values occurring between two testing sets. Thus, the smaller the p-value, the less likely that the observed difference results from identical testing set correctness distributions. The threshold for p-value is set to 0.05 here. From Table 2, we can find that compared with the MultiK-MHKS, the proposed RPEMEKL achieves a clear improvement on most of the used data sets especially for the arrhythmia, ionosphere, letter, and wine. Meanwhile, our RPEMEKL has a competitive accuracy to SVM-2K and MKDA(SDP), which is validated by the p-value reported here. The experimental phenomenon validates the effectiveness of the proposed algorithm.

On the other hand, it can be found that the RPEMEKL is worse than the efficient MKL [12] on some datasets from this table. Here, we give its reason. In RPEMEKL, each input sample x is mapped into $\Phi_l^e(x)$ through the following equation

$$\Phi_l^e(x) = \Lambda_l^{-1/2} Q_l^T [k(x, x_1), \ldots, k(x, x_p)]^T, \tag{11}$$

where p is much smaller than the whole training set size N, which might make some discriminant information lost and thus cause that the experimental results fail to validate the advantage of the proposed method. However, as Section 2 stated, the feasibility of our method can be guaranteed by the characteristic of Random Projection (RP) in kernel-based learning, which is that the learning based on the mapping (11) can lead to an approximate separability at one certain margin [3]. Since our previous work MultiK-MHKS [14] makes the p in the (11) with the size of the whole input training samples N and we adopt a smaller p, the computational cost for the eigendecomposition in the Eq. (12) can decrease from $o(N^3)$ in the MultiK-MHKS to $o(p^3)$ in the proposed method.

$$K_l = Q_l \Lambda_l Q_l^T. \tag{12}$$

Meanwhile, it can be found that the proposed RPEMEKL selects Empirical Kernel Mapping (EKM) [16] instead of the traditional Implicit Kernel Mapping (IKM). As we have demonstrated in the literature [14], we make $B = KQ\Lambda^{-1/2}$. Then the dot product matrix of $\{\Phi^e(x_i)\}_{i=1}^N$ generated by EKM can be calculated as

$$BB^T = KQ\Lambda^{-1/2}\Lambda^{-1/2}Q^T K = K. \tag{13}$$

That is exactly equal to the kernel matrix in the traditional IKM, and thus the mapped samples respectively generated by EKM and IKM have the same geometrical structure. In [16,14], it is shown that comparing EKM with IKM, the former is easier to access and easier to study the adaptability of a kernel to the input space than the latter. That is why we select EKM here. On the whole, the RPEMEKL gives an alternative multiple kernel learning from the EKM viewpoint, which

Table 3
Average training time (in second) comparison between RPEMEKL with the optimized p and $p = 0.1$, MultiK-MHKS, SVM-2K, MKDA(SDP), and MKL [12].

	RPEMEKL	MultiK-MHKS	SVM-2K	MKDA(SDP)	MKL
Cmc	24.09	3572	*0.17*	779.8	15,516
Glass	4.809	14.76	*1.466*	36.69	318.7
House-votes	*0.817*	6.658	0.875	73.28	330.8
Ionosphere	*0.431*	5.580	0.738	44.41	99.72
Iris	0.341	1.360	*0.286*	28.25	63.06
Clean	1.638	20.90	*0.554*	105.6	211.6
Letter	*9.652*	236.9	14.06	230.2	63.06
Pima	*1.793*	11.66	2.878	459.6	2513
Page-block	*82.15*	8983	398.8	41,751	57,266
Arrhythmia	41.88	496.1	*11.63*	100.8	3477
Wine	*0.317*	1.86	0.641	30.37	73.31
Segmentation	680.5	7634	*86.77*	39,218	31,259
Sonar	*0.112*	12.61	0.356	15.95	44.32
Waveform	*45.56*	375.6	145.2	726.9	7558

Note: The shortest time of each data set are in italic.

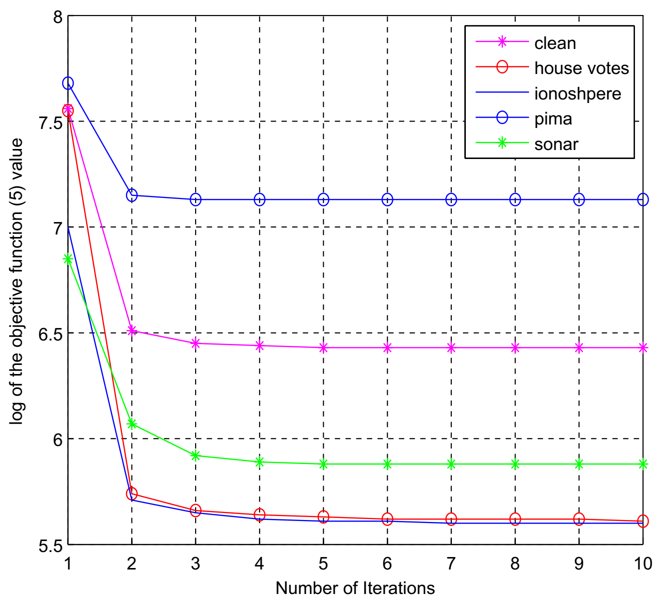

Fig. 1. Convergence of RPEMEKL on the natural logarithm value of the objective function (5) on the binary-class datasets including clean, house-votes, ionosphere, pima, and sonar.

is different from the traditional IKM learning. In future we would make up the lost of some information due to the random selection from the whole sample set as shown in the Eq. (11).

3.3. Running time comparison

In this section, we analyze the RPEMEKL in terms of computation complexity. Since Step 6.iii of RPEMEKL in Table 1 seems to be computational heavy, we first discuss it. The Step 6.iii of RPEMEKL is given as follows

$$\omega_l^{k+1} = \left(\left(1+\lambda\frac{M-1}{M}\right)Y_l^TY_l + c_l\widetilde{I}\right)^{-1} Y_l^T\left(b_l^{k+1} + 1_{N\times 1} + \lambda\frac{1}{M}\sum_{j=1;j\neq l}^{m} Y_j\omega_j^k\right),$$
$$l = 1\ldots M, \tag{14}$$

where $\left((1+\lambda\frac{M-1}{M})Y_l^TY_l + c_l\widetilde{I}\right) \in \mathbb{R}^{(r_l+1)\times(r_l+1)}, Y_l^T \in \mathbb{R}^{(r_l+1)\times N}$, $\left(b_l^{k+1} + 1_{N\times 1} + \lambda\frac{1}{M}\sum_{j=1;j\neq l}^{m} Y_j\omega_j^k\right) \in \mathbb{R}^N$. Since the Eq. (14) needs to get the inverse matrix of $(1+\lambda\frac{M-1}{M})Y_l^TY_l + c_l\widetilde{I}$, we need to decrease its computational complexity for the inverse. Fortunately, the dimensionality of $(1+\lambda\frac{M-1}{M})Y_l^TY_l + c_l\widetilde{I}$ is $(r_l + 1) \times (r_l + 1)$, where r_l is the size of the positive eigenvalues of K_l and determined by the following equation

$$K_l = Q_l\Lambda_lQ_l^T, \tag{15}$$

where $\Lambda_l \in \mathbb{R}^{r_l\times r_l}$ is a diagonal matrix consisting of the r_l positive eigenvalues of K_l, and $Q_l \in \mathbb{R}^{p\times r_l}$ consists of the corresponding orthonormal eigenvectors. Since K_l is generated by the sample subset $S' = \{(x_i, \varphi_i)\}_{i=1}^p$ randomly selected from the whole set S with the size N, p is less than N and $r_l \leqslant p$ is small. Therefore, the computational complexity of the Eq. (14) is smaller.

Then we also give the average training time of the performed algorithms corresponding to the optimal parameters in Table 3. From this table, it can be found that compared with the MultiK-MHKS, RPEMEKL takes a much smaller training time on all the used data sets which attributes to that the RPEMEKL only utilize p samples instead of N ($p < N$) to construct the kernel matrix K and thus the whole decomposition complexity is decreasing from $o(N^3)$ to $o(Hp^3)$. On the other hand, since SVM-2K just adopts two kernel in learning processing, its computational cost is the smallest on most of all the used datasets. However, it should be stated that since SVM-2K can deal with only two kernels in one learning process, it should be implemented for three times for the three candidate kernels and Table 3 just gives one optimal result. We also can find that MKDA(SDP) takes a large time for training since it depends on the complex SDP. Finally, although the RPEMEKL is worse than the efficient MKL [12] on some datasets in terms of classification, RPEMEKL has a superior training time to the MKL [12].

3.4. Convergence analysis

In this section, we discuss the convergence of the RPEMEKL. Here, we adopt an empirical justification as used in [18] to demonstrate that RPEMEKL can converge in the limited iterations. Fig. 1 shows the natural logarithm value of the objective function (5) changes with the iteration number of RPEMEKL respectively on the binary-class datasets including clean, house-votes ionosphere, pima, and sonar. From the figure, it can be found that the optimization objectives (5) on these datasets can obviously converge to stable values, where less than five iterations are usually enough to achieve convergence.

4. Conclusion and future work

In this paper, we develop an ensemble classifier with multiple kernels named RPEMEKL. The characters of the proposed RPEMEKL has two folds. First, the RPEMEKL can efficiently reduce the eigendecomposition of the kernel matrix. The reduction can be guaranteed by the characteristic of RP in kernel-based learning that the learning based on the mapping (4) can lead to an approximate separability at one certain margin [3]. Secondly, the classification performance of the RPEMEKL can be improved by the diversity generated from different random selections of the subsets S_q's. The experimental results here demonstrate the effectiveness and efficiency of the RPEMEKL. On the whole, the proposed RPEMEKL gives an alternative multiple kernel learning from the EKM viewpoint, which is different from the traditional IKM learning. In future we would give a solution to make up the lost of some information due to the random selection from the whole sample set.

Acknowledgments

The authors would like to thank Natural Science Foundations of China under Grant Nos. 60903091, 61170151, 61272198 and

21176077, the Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20090074120003, and the Fundamental Research Funds for the Central Universities for partial support.

References

[1] D. Achlioptas, Database-friendly random projections, Journal of Computer and System Sciences 66 (4) (2003) 671–687.

[2] R. Arriaga, S. Vempala, An algorithmic theory of learning, robust concepts and random projection, Machine Learning 63 (2) (2006) 161–182.

[3] M. Balcan, A. Blum, S. Vempala, Kernels as features: on kernels, margins, and low-dimensional mappings, Machine Learning 65 (2006) 79–94.

[4] J.D.R. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, S. Szedmak, Two view learning: SVM-2k, theory and practice, Advances in Neural Information Processing Systems 18 (2006) 355.

[5] A. Frank, A. Asuncion, UCI Machine Learning Repository, University of California, School of Information and Computer Cciences, Irvine, 2010. <http://archive.ics.uci.edu/ml>

[6] S.J. Kim, A. Magnani, S. Boyd, Optimal kernel selection in kernel fisher discriminant analysis, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 465–472.

[7] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.

[8] J. Łeski, Ho-Kashyap classifier with generalization control, Pattern Recognition Letters 24 (14) (2003) 2281–2290.

[9] Mitchell Tom, Machine Learning, McGraw-Hill, Boston, 1997.

[10] 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) 181–202.

[11] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

[12] S. Sonnenburg, G. Ratsch, C. Schafer, A general and efficient multiple kernel learning algorithm, in: Neural Information Processing Systems, 2005.

[13] S. Sonnenburg, G. Ratsch, C. Schafer, B. Scholkopf, Large scale multiple kernel learning, Journal of Machine Learning Research 7 (2006) 1531–1565.

[14] Z. Wang, S. Chen, T. Sun, MultiK-MHKS: a novel multiple kernel learning algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 348–353.

[15] Y. Wu, C. Chang, Efficient text chunking using linear kernel with masked method, Knowledge-Based Systems 20 (3) (2007) 209–219.

[16] H. Xiong, M.N.S. Swamy, M.O. Ahmad, Optimizing the kernel in the empirical feature space, IEEE Transactions on Neural Networks 16 (2) (2005) 460–474.

[17] Q.S. Xu, Y.Z. Liang, Monte carlo cross validation, Chemometrics and Intelligent Laboratory Systems 56 (2001) 1–11.

[18] J. Ye, Generalized low rank approximation of matrices, Machine Learning 61 (1–3) (2005) 167–191.

[19] M. Zangooei, S. Jalili, PSSP with dynamic weighted kernel fusion based on SVM-PHGS, Knowledge-Based Systems 27 (2012) 424–442.

[20] H. Zhang, J. Lu, Semi-supervised fuzzy clustering: a kernel-based approach, Knowledge-Based Systems 22 (6) (2009) 477–481.