Reduced multiple empirical kernel learning machine

Zhe Wang · MingZhe Lu · Daqi Gao

Received: 24 January 2014 / Revised: 15 July 2014 / Accepted: 15 July 2014 / Published online: 29 July 2014
© Springer Science+Business Media Dordrecht 2014

Abstract Multiple kernel learning (MKL) is demonstrated to be flexible and effective in depicting heterogeneous data sources since MKL can introduce multiple kernels rather than a single fixed kernel into applications. However, MKL would get a high time and space complexity in contrast to single kernel learning, which is not expected in real-world applications. Meanwhile, it is known that the kernel mapping ways of MKL generally have two forms including implicit kernel mapping and empirical kernel mapping (EKM), where the latter is less attracted. In this paper, we focus on the MKL with the EKM, and propose a reduced multiple empirical kernel learning machine named RMEKLM for short. To the best of our knowledge, it is the first to reduce both time and space complexity of the MKL with EKM. Different from the existing MKL, the proposed RMEKLM adopts the Gauss Elimination technique to extract a set of feature vectors, which is validated that doing so does not lose much information of the original feature space. Then RMEKLM adopts the extracted feature vectors to span a reduced orthonormal subspace of the feature space, which is visualized in terms of the geometry structure. It can be demonstrated that the spanned subspace is isomorphic to the original feature space, which means that the dot product of two vectors in the original feature space is equal to that of the two corresponding vectors in the generated orthonormal subspace. More importantly, the proposed RMEKLM brings a simpler computation and meanwhile needs a less storage space, especially in the processing of

testing. Finally, the experimental results show that RMEKLM owns a much efficient and effective performance in terms of both complexity and classification. The contributions of this paper can be given as follows: (1) by mapping the input space into an orthonormal subspace, the geometry of the generated subspace is visualized; (2) this paper first reduces both the time and space complexity of the EKM-based MKL; (3) this paper adopts the Gauss Elimination, one of the on-the-shelf techniques, to generate a basis of the original feature space, which is stable and efficient.

Keywords Multiple kernel learning · Empirical kernel mapping · Reduced kernel learning · Orthonormal basis · Pattern recognition

Introduction

Cognitive neuroscience seeks to clarify the cognitive activity of brain mechanisms. It explores how the human brain calls on all the levels of its components, including molecular, cellular, and the whole brain area, to achieve a variety of cognitive activities. In recent years, the research have made significant progress on several core branches of cognitive neuroscience including cognitive psychology, psychological linguistics, artificial intelligence, artificial neural network and so on. In this paper, we focus on the artificial intelligence and conduct a further research on machine learning. Through our research, we hope to inspire the further connection between machine learning and the other branches of cognitive neuroscience, which is also discussed by some researchers. For example, Wang et al. (2010) applied the principal component analysis (PCA) into the original electroencephalogram (EEG) for

Z. Wang (✉) · M. Lu · D. Gao
Department of Computer Science and Engineering, East China
University of Science and Technology, Shanghai 200237,
People's Republic of China
e-mail: wangzhe@ecust.edu.cn

dimension reduction, which is a novel feature extraction of machine learning. Yu et al. (2011) proposed a new computational model of bottom-up visual attention based on a pulsed PCA transform, which exploited the signs of the PCA coefficients so as to generate a spatial and motional saliency. In the literature Zhang et al. (2013), extracted the psychophysiological measures so as to characterize the operator functional state (OFS) in the automated tasks under a complex form of human-automation interaction, where the fuzzy C-mean (FCM) algorithm was used and tested for its OFS performance. In this paper, we conduct the research from the perspective of multiple kernel learning (MKL).

The classical formulation of MKL induces a linear convex combination of the M candidate kernels as $G = \sum_{l=1}^M \alpha_l K_l$, $\alpha_l \geq 0$, $\sum_{l=1}^M \alpha_l = 1$ into support vector machine (SVM) (Muller et al. 2001). Lanckriet et al. (2004) constructed a convex Quadratically Constrained Quadratic Program (QCQP) for the combinations of multiple kernels $G = \sum_{l=1}^M \alpha_l K_l$. For extending the method of Lanckriet et al. to large scale problems, Bach et al. (2004) took the dual formulation of QCQP as a Second-Order Cone Programming (SOCP) problem. Then Sonnenburg et al. (2006a, b) reconstructed QCQP as a Semi-Infinite Linear Program (SILP) that recycles the standard SVM implementations. Rakotomamonjy et al. (2007, 2008) proposed simple MKL by exploring an adaptive 2-norm regularization formulation. Szafranski et al. (2010), Xu et al. (2010) and Subrahmanya and Shin (2010) constructed the connection between MKL and group-LASSO to model group structure. Most of them were based on implicit kernel mapping (IKM) (Muller et al. 2001). The IKM, denoted as $\Phi^i(x)$, can be achieved by introducing a kernel function $k(x_i, x_j)$, which determines the geometrical structure of the mapped data in feature space, where we do not need to obtain the explicit mapping form, but to compute the $k(x_i, x_j) = \Phi^i(x_i) \cdot \Phi^i(x_j)$ (Muller et al. 2001). In this paper, we focus on another kernel mapping method, which is called empirical kernel mapping (EKM). The EKM, denoted as $\Phi^e(x)$, should explicitly give the feature of x in the mapped Φ^e -space. The classical formulation of EKM is $\Phi^e(x) = \Lambda^{-1/2} Q^T [k(x, x_1), \dots, k(x, x_N)]^T$ (Scholkopf et al. 1999; Xiong et al. 2005), which will be described in Sect. 2.3. Both of $\Phi^i(x)$ and $\Phi^e(x)$ have their own characters. The former is supposed to avoid the curse of dimensionality and keep the linearity property of learning machine in the mapped space at the same time. The latter is supposed to be much easier in processing and analysis the adaptability of a kernel for the input space than the former (Kim et al. 2006; Yan et al. 2010; Farquhar et al. 2005). As we stated above, the MKL with IKM has

recently got much attention (Lanckriet et al. 2004; Bach et al. 2004; Sonnenburg et al. 2006a, b; Rakotomamonjy et al. 2007; Szafranski et al. 2010). In contrast, it is less attracted for the MKL with EKM except our previous work MultiK-MHKS (Wang et al. 2008).

Although those MKL algorithms, which we mentioned above, successfully increase the accuracy, the use of multiple kernels leads to increase the time and space complexity significantly, including our previous work MultiK-MHKS. A real-world application typically requires testing its trained system for numerous instances. Testing a single instance requires smaller amounts of time and space than training the entire system, but the total amount of time and space consumed during the use of the system with numerous tests is critically important in real-world applications, especially when (1) the total storage space is restricted, and (2) the real-time response during testing an instance is required to be sufficiently fast.

There are two main methods to reduce the time and space complexity. One is to pre-process the inputs before training the classifiers, and the other is to optimize the architecture of MKL algorithms. The Nyström approximation method (Williams and Seeger 2001) decreases the computational cost of kernel-based methods from $O(n^3)$ to $O(m^2n)$, where n is the number of training inputs and $0 < m < n$. The approximation matrix $\tilde{G}$ is got by randomly choosing m rows and columns from the Gram kernel matrix G without replacement. And it is not necessary to store the whole Gram kernel matrix G , but only a $m \times n$ portion of it. Wang et al. (2011) applied it into the multi-view learning machine. Moreover, Demir and Alpaydin (2005) developed a utility-based framework to reduce the time complexity. They constructed two novel cost-conscious ensembles; the first one determined a subset of classifiers and the second dynamically selected a single classifier. Both ensembles successfully switched between classifiers according to the accuracy-cost trade-off of an application.

In this paper, we pay attention onto the combination MKL with EKM, and propose a reduced multiple empirical kernel learning machine named RMEKLM for short. As we mentioned above, we take the first method to reduce the time and space complexity, that is pre-processing the EKM-space. It is the first to propose the problem about reducing the time and space complexity for EKM-based MKL algorithm. As we know, in kernel based learning the data is mapped to a kernel feature space ($\mathcal{F}$) of a dimension that corresponds to the number of training data points. In practice, however, the data forms a smaller submanifold in $\mathcal{F}$. The RMEKLM borrows the idea from (Harmeling et al. 2003), which adopts an orthonormal basis to construct a subspace of $\mathcal{F}$. Then by mapping the input vector into the reduced orthonormal space, we can get a reduced vector,

Fig. 1 The picture on the *top side* depicts the input space. By mapping the inputs into the feature space $\mathcal{F}$, we get the picture on the *left-bottom side*. However, the feature vectors in $\mathcal{F}$ constitute an orthonormal subspace of $\mathcal{F}$, which is depicted on the *right-bottom side*

which can reduce the time complexity, especially the testing time complexity. The orthonormality implies that the dot product of two reduced vectors coincides with that of two corresponding input vectors mapped into the original feature space, which is a *remarkable property*. In order to clarify the relationship between the geometry structures of $\mathcal{F}$ and the reduced space intuitively, we draw Fig. 1. In Fig. 1, for simplicity, we suppose that the input space is 1-dimensional. The inputs are non-linear separable. By mapping them into $\mathcal{F}$, which could be 3-dimensional, these feature vectors are linear separable. However, these feature vectors constitute a subspace of $\mathcal{F}$. So by generating an orthonormal basis of the subspace of $\mathcal{F}$, we can get the 2-dimensional subspace, and also these reduced inputs are linear separable. Furthermore, we explored a new way to obtain a group of inputs that later forms an orthonormal basis of $\mathcal{F}$, which is more efficient and stable than the original method in (Harmeling et al. 2003). In practice, rather than storing all the training inputs in the original EKM method, we just need to store a portion of them that later constructs an orthonormal basis of $\mathcal{F}$, which reduces the space cost significantly.

In implementation, we apply RMEKLM to the framework of MultiK-MHKS (Wang et al. 2008), since which is also an EKM-based algorithm. MultiK-MHKS borrowed the characteristic of Canonical Correlation Analysis (CCA)

(Hardoon et al. 2004) and explicitly mapped the input data into multiple feature spaces, where each feature space can be taken as one view of the original input data. Each view is expected to exhibit some geometrical structures of the original data from its own perspective such that all the views can complement for the subsequent learning task. The MultiK-MHKS aimed to make the projection of each view onto their corresponding basis vectors maximally close to the average projection of all views.

The advantages of the proposed RMEKLM are highlighted as follows:

- *Visualized geometry structure*: By mapping the inputs onto an orthonormal basis of the feature space, the well structured subspace is isomorphic to the original feature space. The visualizability of the orthonormal subspace is depicted in Fig. 1.
- *Reduced time and space complexity*: Since we get a subspace of the original feature space, less calculations for kernel functions will be performed, so that the learning process is sped up, especially the testing process. And we only need to store a portion of the training patterns that later construct an orthonormal basis of the feature space.
- *Feature vectors extraction*: Based on Theorem 1 that we will show in Sect. 2.2, we adopt the Gauss

Elimination, one of the on-the-shelf techniques, so as to extract a set of feature vectors of the feature space to generate an orthonormal basis, which is more efficient and stable than the other methods used in Harmeling et al. (2003).

The rest of this paper is organized as follows. Section 2 gives the architecture of the proposed RMEKLM. Section 3 reports on all the experimental results. Finally, conclusions and future work are given in Sect. 4.

Reduced multiple empirical kernel learning machine (RMEKLM)

The proposed RMEKLM inherits the architecture from MultiK-MHKS. But different from it, RMEKLM maps the inputs into an orthonormal subspace of the feature space. However, to construct an orthonormal subspace, RMEKLM needs to generate an orthonormal basis. In this section, firstly, assuming that we have found a group of inputs from the input space that construct an orthonormal basis after mapping them into the feature space, we will specify the architecture of RMEKLM in Sect. 2.1. After that, we will elaborate how to find such group of inputs in Sect. 2.2. Then, comparison between RMEKLM and MultiK-MHKS will be shown in Sect. 2.3 in terms of the time and space complexity.

Architecture of RMEKLM

Suppose that the sample inputs $x_1, \dots, x_N \in \mathfrak{R}'$ are mapped to $\Phi_x^e := [\Phi^e(x_1) \dots \Phi^e(x_N)] \in \mathcal{F}$, where N is the size of the inputs. In addition to the inputs, consider some further points $v_1, \dots, v_r \in \mathfrak{R}'$ from the same inputs, that could generate a basis $\Phi_v^e := [\Phi^e(v_1) \dots \Phi^e(v_r)] \in \mathcal{F}$, which we note by

$$\text{span}(\Phi_x^e) = \text{span}(\Phi_v^e) \text{ and } \text{rank}(\Phi_x^e) = \text{rank}(\Phi_v^e) = r. \quad (1)$$

Moreover, Φ_v^e being a basis implies that the matrix $\Phi_v^{eT} \Phi_v^e$ has full rank and its inverse exists. So, we can define an orthonormal basis

$$\Xi := \Phi_v^e (\Phi_v^{eT} \Phi_v^e)^{-\frac{1}{2}} \quad (2)$$

the column space of which is identical to the column space of Φ_v^e and Φ_x^e . Consequently this basis Ξ enables us to parameterize all inputs that lie in the column space of Φ_x^e by some vectors in $\mathfrak{R}'$. For instance, for two vectors $\Phi_x^e \alpha_{\Phi_x^e}$ and $\Phi_x^e \beta_{\Phi_x^e}$ in the column space of Φ_x^e with $\alpha_{\Phi_x^e}$ and $\beta_{\Phi_x^e}$ in $\mathfrak{R}^N$, there exist α_{Ξ} and β_{Ξ} in $\mathfrak{R}^r$ such that $\Phi_x^e \alpha_{\Phi_x^e} = \Xi \alpha_{\Xi}$ and $\Phi_x^e \beta_{\Phi_x^e} = \Xi \beta_{\Xi}$. The orthonormality implies

$$\alpha_{\Phi_x^e}^T \Phi_x^e \Phi_x^e \beta_{\Phi_x^e} = \alpha_{\Xi}^T \Xi^T \Xi \beta_{\Xi} = \alpha_{\Xi}^T \beta_{\Xi} \quad (3)$$

which states the *remarkable property* that the dot product of two linear combinations of the columns of Φ_x^e in $\mathcal{F}$ coincides with the dot product in $\mathfrak{R}'$. By construction of Ξ , the column space of Φ_x^e is naturally isomorphic (as vector spaces) to $\mathfrak{R}'$. Moreover, this implies that all properties regarding angles and lengths can be taken back and forth between the column space of Φ_x^e and $\mathfrak{R}'$.

Then we show how to map the input data to the right space. The expression

$$(\Phi_v^e \Phi_v^e)_{ij} = \Phi^{eT}(v_i) \Phi^e(v_j) = k(v_i, v_j) \quad (4)$$

with $i, j = 1 \dots r$ can be effectively calculated using the kernel trick and by construction of $v_1, \dots, v_r$, it has full rank and is thus invertible. Similarly we get

$$(\Phi_v^e \Phi_x^e)_{ij} = \Phi^{eT}(v_i) \Phi^e(x_j) = k(v_i, x_j) \\ \text{with } i = 1 \dots r, \quad j = 1, \dots, N.$$

Using both matrices we compute finally the parameter matrix

$$\Psi_r^e(x) := \Xi^T \Phi^e(x) = (\Phi_v^{eT} \Phi_v^e)^{-\frac{1}{2}} \Phi_v^{eT} \Phi^e(x) \\ = (\Phi_v^{eT} \Phi_v^e)^{-\frac{1}{2}} [k(x, v_1), \dots, k(x, v_r)]^T \quad (5)$$

note that $(\Phi_v^{eT} \Phi_v^e)^{-\frac{1}{2}}$ is symmetric.

After getting the parameterized points $\Psi_r^e(x)$ in Eq. (5). We can apply it to the model of MultiK-MHKS (Wang et al. 2008). MultiK-MHKS borrowed an argument from canonical correlation analysis (CCA) (Hardoon et al. 2004) to develop a MKL method based on EKM. It aimed to minimize the objective function

$$R(F) = \sum_{l=1}^M [R_{\text{emp}}(f_l) + c_l R_{\text{reg}}(f_l)] + \lambda R_{\text{IFSL}}(F), \quad (6)$$

where $l = 1 \dots M$, M denotes the number of kernels we used, f_l is the l -th classifier, $R_{\text{emp}}(f_l)$ is the empirical risk term, $R_{\text{reg}}(f_l)$ represents the regularization term that penalizes the roughness or smoothness of f_l , and $c_l \geq 0$ is a regularization parameter that controls the trade-off between $R_{\text{emp}}(f_l)$ and $R_{\text{reg}}(f_l)$, $R_{\text{IFSL}} = \sum_{l=1}^M (f_l - \frac{1}{M} \sum_{j=1}^M f_j)^2$ plays a role of the agreement penalization on the outputs of multiple f_l and $\lambda \geq 0$ is a factor controlling the trade-off between R_{IFSL} and $\sum_{l=1}^M [R_{\text{emp}}(f_l) + c_l R_{\text{reg}}(f_l)]$.

The base classifier f_l takes the Modification of Ho-Kashyap algorithm with Squared approximation of the misclassification errors (MHKS) (Łeński 2003) for each kernel. Given training samples $\{(x_i, \varphi_i)\}_{i=1}^N$, $\varphi_i \in \{-1, +1\}$, $x_i \in \mathfrak{R}'$, the decision function of MHKS

$$f(x_i) = \tilde{\omega}x_i + \omega_0 \quad (7)$$

is obtained by optimizing the following loss function

$$\min_{\omega \in \mathfrak{R}^{r+1}, b \geq 0} I = (Y\omega - \mathbf{1}_{N \times 1} - b_{N \times 1})^T (Y\omega - \mathbf{1}_{N \times 1} - b_{N \times 1}) + c\tilde{\omega}^T \tilde{\omega}, \quad (8)$$

where $\tilde{\omega} \in \mathfrak{R}^r$ are the weight vector and $\omega_0 \in \mathfrak{R}$ is the bias. $\omega = [\tilde{\omega}; \omega_0]$ is the augmented weight vector. The matrix Y is defined as $Y = [\varphi_1(x_1^T, 1); \dots; \varphi_N(x_N^T, 1)]$. $\mathbf{1}_{N \times 1}$ is a vector of N dimension with all elements equal to 1. $b_{N \times 1}$ is a vector with all elements of nonnegative values. $(Y\omega - \mathbf{1}_{N \times 1} - b_{N \times 1})^T (Y\omega - \mathbf{1}_{N \times 1} - b_{N \times 1})$ is the empirical risk term of MHKS. $\tilde{\omega}^T \tilde{\omega}$ is the generalization regularization term and c is the regularization parameter.

Now we introduce MHKS into each feature space as the base classifier. Similarly the given training samples $\{(x_i, \varphi_i)\}_{i=1}^N$, $\varphi_i \in \{-1, +1\}$, $x_i \in \mathfrak{R}^r$ with M kernels, RMEKLM explicitly map $\{x_i\}_{i=1}^N$ into M feature spaces $\{\Psi_1^e(x_i), \dots, \Psi_l^e(x_i), \dots, \Psi_M^e(x_i)\}_{i=1}^N$ by the way of Eq. (5) rather than that of Eq. (18) in MultiK-MHKS. In the l -th feature space, we assign the augmented sample $\Psi_{li}^e = (\Psi_l^e(x_i), 1)^T$ and then let $Y_l = [\varphi_1 \Psi_{l1}^{eT}; \dots; \varphi_N \Psi_{lN}^{eT}]$. For the learning framework Eq. (6) mentioned above, the $R_{emp}(f_l)$ and $R_{reg}(f_l)$ are given as follows:

$$R_{emp}(f_l) = (Y_l \omega_l - \mathbf{1}_{N \times 1} - b_{N \times 1}^l)^T (Y_l \omega_l - \mathbf{1}_{N \times 1} - b_{N \times 1}^l), \quad (9)$$

$$R_{reg}(f_l) = \tilde{\omega}_l^T \tilde{\omega}_l, \quad (10)$$

where $\tilde{\omega}_l \in \mathfrak{R}^{r_l}$, $\omega_{0l} \in \mathfrak{R}$ are the weight vector and the bias in the l -th feature space, and the augmented weight vector is $\omega_l = [\tilde{\omega}_l^T; \omega_{0l}]^T$. Finally, the $R_{IFSL}(F)$ is given as follows:

$$\sum_{i=1}^M \left(Y_l \omega_l - \frac{1}{M} \sum_{j=1}^M Y_j \omega_j \right)^T \left(Y_l \omega_l - \frac{1}{M} \sum_{j=1}^M Y_j \omega_j \right), \quad (11)$$

where M denotes the number of views we use. Minimizing Eq. (11) characterizes that the outputs of each view $\{(\Psi_l^e(x_i), \varphi_i)\}_{i=1}^N$ onto their corresponding weight vector ω_l are constrained to be maximally close to the average outputs of all the views, which is induced from NmCCA (Wang et al. 2008) that is a new formulation for CCA in multi-view case.

Consequently, RMEKLM implements the learning framework Eq. (6) by

$$\min_{\omega_l \in \mathfrak{R}_{l=1 \dots M}^{r_l+1}, b_{N \times 1}^l \geq 0; } L = \sum_{l=1}^M \left[(Y_l \omega_l - \mathbf{1}_{N \times 1} - b_{N \times 1}^l)^T \times (Y_l \omega_l - \mathbf{1}_{N \times 1} - b_{N \times 1}^l) + c_l \tilde{\omega}_l^T \tilde{\omega}_l \right] + \lambda \sum_{i=1}^M \left(Y_l \omega_l - \frac{1}{M} \sum_{j=1}^M Y_j \omega_j \right)^T \times \left(Y_l \omega_l - \frac{1}{M} \sum_{j=1}^M Y_j \omega_j \right), \quad (12)$$

where $Y_l, \omega_l, \tilde{\omega}_l, b_{N \times 1}^l$ correspond to one MHKS in one view. It can be found that the optimization problem Eq. (12) with respect to the single ω_l is convex (Cristianini and Shawe-Taylor 2000) while $\omega_{j \neq l}$ is fixed. Thus, we solve ω_l in a sequence $l = 1 \dots M$ and employ a modification of the gradient descent with a heuristic update-rule for each ω_l . Now, making the gradient of L with respect to ω_l be zero, we can obtain

$$\omega_l = \left[\left(1 + \lambda \frac{M-1}{M} \right) Y_l^T Y_l + c_l \tilde{I} \right]^{-1} Y_l^T \times \left(b_{N \times 1}^l + \mathbf{1}_{N \times 1} + \lambda \frac{1}{M} \sum_{j=1, j \neq l}^M Y_j \omega_j \right), \quad (13)$$

where $\tilde{I}$ is a diagonal matrix with full 1s except the last element set to zero.

In the l -th view, it can be noted that ω_l only depends on $b_{N \times 1}^l$ from Eq. (13). Then, by differentiating L with regard to $b_{N \times 1}^l$ and setting the result equal to zero, we can get

$$e_l = Y_l \omega_l - b_{N \times 1}^l - \mathbf{1}_{N \times 1} = \mathbf{0}. \quad (14)$$

In order to guarantee that the samples are correctly classified in the l -th view, the elements of $b_{N \times 1}^l$ need to be nonnegative. Thus, we employ the iterative algorithm for determining ω_l and $b_{N \times 1}^l$ analogously to Łeski (2003). First, with $b_{N \times 1}^{l,k}$ representing the vector $b_{N \times 1}^l$ at the k -th iteration, we initialize $b_{N \times 1}^{l,1} \geq 0$, then keep $b_{N \times 1}^{l,k} \geq 0$ at each iteration k and, thus, obtain

$$\begin{cases} b_{N \times 1}^{l,1} \geq 0 \\ b_{N \times 1}^{l,k+1} = b_{N \times 1}^{l,k} + \rho_l (e_l^k + |e_l^k|) \end{cases} \quad (15)$$

where at the k -th iteration, the error vector of the l -th view e_l^k is defined as $e_l^k = Y_l \omega_l^k - b_{N \times 1}^{l,k} - \mathbf{1}_{N \times 1}$ and the learning rate of the l -th view $\rho_l > 0$. Then, ω_l^{k+1} can be given by Eq. (13). In practice, the termination criterion can be designed as $\|L^{k+1} - L^k\|_2 \leq \xi$. The designed procedure is summarized in Table 1.

Using the obtained weight vector $\omega_l, l = 1 \dots M$, now we give the decision function of RMEKLM for the input

Table 1 RMEKLM algorithm

Input : $\{(x_i, \varphi_i)\}_{i=1}^N$, $\varphi_i \in \{-1, +1\}$; the m candidate kernels $\{ker_l(x_i, x_j)\}_{l=1}^m$

Output: the weight vectors $\omega_l, l = 1 \dots m$

1. Explicitly map $\{x_i\}_{i=1}^N$ into $\{\Psi_1^e(x_i), \dots, \Psi_l^e(x_i), \dots, \Psi_m^e(x_i)\}_{i=1}^N$ by m kernels as shown in Eq. (5).
 2. For each view, $Y_l = [\varphi_1 \Psi_{l1}^{eT}; \dots; \varphi_N \Psi_{lN}^{eT}]$, $l = 1 \dots m$
 3. Initialize $\xi > 0$, $\rho_l > 0$, $c_l \geq 0$, $b_l^1 \geq 0$, $l = 1 \dots m$ and $\omega_l^1, l = 2 \dots m$; set the iteration index $k = 1$
 4. $\omega_l^k = ((1 + \lambda \frac{m-1}{m}) Y_l^T Y_l + c_l \tilde{I})^{-1} Y_l^T (b_l^k + \mathbf{1}_{N \times 1} + \lambda \frac{1}{m} \sum_{j=1, j \neq l}^m Y_j \omega_j^k)$, $l = 1$
 5. $e_l^k = Y_l \omega_l^k - b_l^k - \mathbf{1}_{N \times 1}$, $l = 1 \dots m$
 6. $b_l^{k+1} = b_l^k + \rho_l (e_l^k + |e_l^k|)$, $l = 1 \dots m$
 7. $\omega_l^{k+1} = ((1 + \lambda \frac{m-1}{m}) Y_l^T Y_l + c_l \tilde{I})^{-1} Y_l^T (b_l^{k+1} + \mathbf{1}_{N \times 1} + \lambda \frac{1}{m} \sum_{j=1, j \neq l}^m Y_j \omega_j^k)$, $l = 1 \dots m$
- if $\|L^{k+1} - L^k\|_2 > \xi$, then $k = k + 1$, go to step 5; else stop

pattern z with its corresponding mapped pattern $\{\Psi_l^e(z)\}_{l=1}^M$:

$$F(z) = \frac{1}{M} \sum_{l=1}^M \omega_l^T [\Psi_l^e(z), 1]^T \begin{cases} > 0, & \text{then } z \in \text{class} + 1 \\ < 0, & \text{then } z \in \text{class} - 1. \end{cases} \quad (16)$$

Generation of orthonormal basis

So far we have assumed to have vectors $v_1, \dots, v_r$ that fulfill Eq. (1) and we presented the beneficial properties of our construction. Due to Eq. (1) and r being finite since it is bounded by N , we can get $v_1, \dots, v_r$ from the original inputs $x_1 \dots x_N$. Here we draw an inference below:

Theorem 1 *The sequence index $r_1 \dots r_r$ of a group of basis vectors in the column space of $\Phi_x^{eT} \Phi_x^e$ is identical to the one in the column space of Φ_x^e .*

Prove Given $A = [\Phi^e(x_1), \dots, \Phi^e(x_N)]$, $\text{rank}(A) = r$, we can get $\Phi_x^{eT} \Phi_x^e = A^T [\Phi^e(x_1), \dots, \Phi^e(x_N)]$. Let $A^T \Phi^e(x_{r_1}), \dots, A^T \Phi^e(x_{r_r})$ be a basis of $\Phi_x^{eT} \Phi_x^e$. We assume that there exists $k_1, \dots, k_r \in \Re$ so that $k_1 \Phi^e(x_{r_1}) + \dots + k_r \Phi^e(x_{r_r}) = 0$. Then, through $k_1 \Phi^e(x_{r_1}) + \dots + k_r \Phi^e(x_{r_r}) = 0$ being pre-multiplied by A^T , we can get $k_1 A^T \Phi^e(x_{r_1}) + k_r A^T \Phi^e(x_{r_r}) = 0$. Because $A^T \Phi^e(x_{r_1}), \dots, A^T \Phi^e(x_{r_r})$ is linearly independent, we know that $k_1 \dots k_r$ are all equal to zero respectively. Thus, the $\Phi^e(x_{r_1}), \dots, \Phi^e(x_{r_r})$ is linearly independent as well. With $\text{rank}(A) = r$, we can draw the conclusion that $[\Phi^e(x_{r_1}), \dots, \Phi^e(x_{r_r})]$ is a basis of Φ_x^e .

Following Theorem 1, we can apply Gaussian Elimination on $\Phi_x^{eT} \Phi_x^e$ to get the sequence index $r_1, \dots, r_r$ and then choose the corresponding $x_{r_1}, \dots, x_{r_r}$ as the vectors $v_1, \dots, v_r$. We use column pivoting algorithm to apply the Gaussian Elimination, whose computational complexity is $\frac{2}{3} O(N^3)$. Since the Random Decrement sometimes loses much information of the feature space for its randomness, and its time complexity is close to $O(n^4)$, which will be

discussed in Sect. 3.4, the Gauss Elimination is more stable and more efficient. The round-off error here we used is fixed to

$$\max(m, n) \text{eps}(\text{class}(K)) \times \|K\|_\infty,$$

where K denotes the kernel matrix, m and n denote the numbers of the rows and columns respectively, $\text{class}(K)$ denotes the type of K (maybe integer, float or double), $\text{eps}(\text{class}(K))$ is the positive distance from $|\text{class}(K)|$ to the next larger in magnitude floating point number of the same precision as $\text{class}(K)$, $\|K\|_\infty$ is the l_∞ -norm of K . We can modify the round-off error manually to get different results. The bigger the round-off error is, the smaller the rank of $v_1, \dots, v_r$ will be and of course the faster the orthonormal basis will be generated. If the round-off error is set too big, much information of the feature space will be lost, which could lead to a bad classification performance.

RMEKLM versus MultiK-MHKS

MultiK-MHKS uses the following empirical kernel mapping to map the inputs into the feature space. Let $\{(x_i, \varphi_i)\}_{i=1}^N$ be a t -dimensional inputs, where $\varphi_i \in \{-1, +1\}$, $i = 1 \dots N$ is the label of x_i , $i = 1 \dots N$. MultiK-MHKS explicitly mapped each sample $x \in \mathbb{R}^t$ into $\Phi^e(x) \in \mathbb{R}^r$. Let K denote the $N \times N$ kernel matrix of rank, say r , that is, $K = [k_{ij}]_{N \times N}$, where $k_{ij} = \Phi^e(x_i)^T \Phi^e(x_j) = k(x_i, x_j)$. Since K is a symmetrical positive-semidefinite matrix, K can be decomposed as

$$K = Q_{N \times r} \Lambda_{r \times r} Q_{r \times N}^T, \quad (17)$$

where Λ is a diagonal matrix containing only the r positive eigenvalues of K in decreasing order, and Q consists of the eigenvectors corresponding to the positive eigenvalues. The map from the input space to an r -dimensional Euclidean space $\Phi_r^e(x) : x \rightarrow \mathbb{R}^r$

$$\Phi_r^e(x) = \Lambda^{-1/2} Q^T [k(x, x_1), \dots, k(x, x_N)]^T. \quad (18)$$

Table 2 Information of UCI data sets

Datasets	Attributes	Classes	Samples
Iris	4	3	150
Wine	13	3	178
Wdbc	30	2	569
Balance	4	3	625
Transfusion	4	2	748
Pima	8	2	768
Mammographic_masses (masses)	6	2	961
CMC	9	3	1,473
Segmentation (segment)	19	7	2,310

is essentially the EKM. This mapping process was implemented for M times with the given M kernels, called multiple empirical kernel learning (MEKL). In implementation, however, the $\Phi_r^e(x)$ is N -dimensional, since during the eigen-decomposition of K , computer cannot get the accurate zero eigenvalue.

In Eq. (18), MultiK-MHKS needs to evaluate the kernel functions of each kernel for $N^2 + N$ times, where N^2 times for evaluating the Gram kernel matrix and N times for evaluating $[k(x, x_1), \dots, k(x, x_N)]$ on the right side of Eq. (18). Besides, the eigen-decomposition of Gram kernel matrix of each kernel needs $O(N^3)$ time complexity, which means the total cost is $O[M \times (N^2 + N + N^3)]$, where M denotes the number of kernels we use. In Eq. (5), however, the input is mapped into an orthonormal subspace of the feature space, which to some extent reduces the computational complexity. RMEKLM needs $N^2 + r$ times for calculating the kernel functions, where N^2 times for evaluating the Gram kernel matrix and r times for evaluating $[k(x, v_1), \dots, k(x, v_r)]$ on the right side of Eq. (5). Rather than evaluating eigendecomposition of each Gram kernel matrix, RMEKLM needs $\frac{2}{3}O(N^3)$ time complexity for extracting the vectors $v_1, \dots, v_r$, which is described in Sect. 2.2. Then the total cost is $O[M \times (N^2 + r + \frac{2}{3}N^3)]$; note that r is much smaller than N . Moreover, from the point of space complexity, MultiK-MHKS needs to store all the training inputs, e.g. $x_1, x_2, \dots, x_N$ in Eq. (18), during both of the training and testing process. On the contrary, RMEKLM needs to store only the r inputs, e.g. $v_1, \dots, v_r$ in Eq. (5), where r is the rank of the subspace, which reduces the space complexity significantly.

Experiments

In this section, we choose MultiK-MHKS, SVM-2K (Farquhar et al. 2005), KMHKS and SVM to compare with RMEKLM for classification performance. Especially, we

pay close attention to RMEKLM and MultiK-MHKS, since both of which are based on EKM. Results of the experiments show that RMEKLM can reduce both the training time and the testing time without losing much classification accuracy, particularly on the large-scale data sets. Benchmark data sets used here are listed in Table 2. Experimental setting, classification and computation performance comparison, performance of two orthonormal basis generation algorithms will be discussed in the following subsections.

Experimental setting

In our experiments, the candidate kernels are linear kernel $kernel(x_i, x_j) = x_i^T x_j$, RBF kernel $kernel(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2})$, where σ is set to the average value of all the l_2 -norm distance $\|x_i - x_j\|_2, i, j = 1 \dots N$ as used in Tsang et al. (2006), and polynomial kernel $kernel(x_i, x_j) = (x_i^T x_j + 1)^d$ where d is set to 2. For RMEKLM and MultiK-MHKS, we use linear kernel, poly kernel and RBF kernel to generate three views, which means the M in Eq. (6) is set to 3, the margin vector b_l is initialized to 10^{-6} , the ξ in the termination criterions is fixed to 10^{-3} , the parameter ρ is set to 0.99, both the regularization parameter c and λ are selected from the set $\{2^{-4}, 2^{-3}, \dots, 2^3, 2^4\}$, and the $\omega_l, l = 2 \dots m$ are initialized to zero vectors, respectively. For SVM-2K, we only use the combination of linear kernel and RBF kernel in our experiment, since we prefer to focus on the time complexity. Moreover, we only choose RBF kernel for the other two single kernel algorithms. The parameter settings for SVM-2K, KMHKS and SVM have been optimized, and we choose the best result of them to compare with RMEKLM. The one-against-one classification strategy (Kreßel 1999) and the majority voting technique are adopted for multi-class problems. For each classification problem, 10 independent runs are performed and their classification accuracies on the test sets are averaged, and in each such run, every used data sets is randomly divided into the two no-overlapping parts with the one for training and the other one for testing, which is known as the Monte Carlo Cross Validation (MCCV) (Xu and Liang 2001).

Computational performance comparison

We compare RMEKLM only with the other two multiple kernel algorithms for training time and testing time. All the computations are performed on Intel(R) Core(TM) i5 CPU @ 3.2 GHz running Linux x86_64 and MATLAB environment. The training time and testing time of RMEKLM as well as those of the compared algorithms with their optimal parameters in 10 runs are recorded in Figs. 2 and 3

Fig. 2 Training time comparison between RMEKLM, MultiK-MHKS and SVM-2K on the UCI data sets listed in Table 2. The graph on the right side is obtained by zooming in the graph on the left side

respectively. In implementation, the training time of RMEKLM includes the time for Gauss Elimination process, computations of mapping the inputs to Ψ^e and training the classifiers.

In Fig. 2, each node represents a data set that list in Table 2. The horizontal axis denotes the scale of the data set, and the vertical axis denotes the time, which amounts in second. From Fig. 2, we can find that RMEKLM runs faster than MultiK-MHKS on most of the data sets. It means through projecting the input space to a lower dimension, RMEKLM can reduce the training time to some extent by less matrix computations in contrast with MultiK-MHKS. RMEKLM works much faster than SVM-2K on most of the datasets except the fifth node, even it combines one more kernel during the training process. The growth rates of the training time of RMEKLM and MultiK-MHKS are much slower than that of SVM-2K. When the scale reaches about 1,200, the gap between RMEKLM and SVM-2K is significantly big. Moreover, we can intuitively find that with the growth of the scale of the inputs, the gap between RMEKLM and MultiK-MHKS is increasing.

In Fig. 3, the meanings of the nodes, the horizontal axis and the vertical axis are the same as those in Fig. 2. From Fig. 3, we can find that RMEKLM could reduce the testing time by mapping the unlabeled samples into the reduced orthonormal space significantly, which means that RMEKLM can offer much quicker response than MultiK-MHKS. Since SVM-2K uses two kinds of kernels to map the unlabeled samples, it works fastest when the scale is large to some degree. Moreover, we can find that when the scale reaches between 800 and 1,200, the growth rate of

RMEKLM is obviously slower than that of MultiK-MHKS. As we stated above, a quick response is important in real-world applications. RMEKLM can classify the unlabeled patterns more efficiently than MultiK-MHKS when the scale of them is growing. On the other hand, experientially speaking, when the time complexity decreases, the space complexity will correspondingly increase. However, in our experiment, RMEKLM needs to store less vectors in the memory and simultaneously works faster than MultiK-MHKS, which is a win-win situation.

Classification comparison

In Table 3, data sets presented in this table are ordered by their scales and the best results are bold. It shows that the classification accuracy of RMEKLM is comparable to that of MultiK-MHKS, which means that the idea, by projecting the input space onto the orthonormal basis of the feature space, is feasible and applicable. Compared with SVM and KMHKS, RMEKLM exhibits clear improvements on Wine, Wdbc, Pima and Segmentation for classification accuracy. It means RMEKLM can offer some needed flexibility and well manipulate the case that involves multiple, heterogeneous data sources, which algorithms based on the single kernel cannot.

Here we focus on the multiple kernel classifiers. From Table 3, we can see that RMEKLM outperforms SVM-2K on all the data sets, especially on Wine, Wdbc, Balance, Pima, CMC and Segmentation, since SVM-2K uses only two kernels to map the patterns. Of course, we can use different combinations of kernels in SVM-2K, and maybe that would provide better results. However, using different combinations means

Fig. 3 Testing time comparison between RMEKLM, MultiK-MHKS and SVM-2K on the UCI data sets listed in Table 2. The graph on the right side is obtained by zooming in the graph on the left side

Table 3 Classification accuracy (%) between RMEKLM, MultiK-MHKS, SVM-2K, KMHKS and SVM on UCI data sets listed in Table 2

Datasets	RMEKLM	Multik-MHKS	SVM-2K	KMHKS	SVM
Iris	98.40 ± 1.23	98.40 ± 1.23	96.27 ± 1.76	97.20 ± 1.60	95.73 ± 2.07
Wine	96.78 ± 1.69	96.78 ± 1.43	73.89 ± 4.48	73.33 ± 2.40	90.00 ± 2.10
Wdbc	95.75 ± 0.88	95.61 ± 0.80	92.18 ± 1.93	92.53 ± 0.98	74.98 ± 5.38
Balance	94.22 ± 0.83	94.22 ± 0.83	88.40 ± 0.89	94.86 ± 0.49	69.71 ± 2.33
Transfusion	77.27 ± 0.64	77.19 ± 0.64	76.20 ± 0.00	76.23 ± 0.43	70.35 ± 6.50
Pima	77.06 ± 1.23	77.11 ± 1.00	73.41 ± 2.47	75.83 ± 1.53	68.46 ± 3.62
Masses	56.65 ± 1.58	56.65 ± 1.58	53.64 ± 0.00	57.55 ± 1.84	52.85 ± 1.10
CMC	55.39 ± 1.12	55.39 ± 1.12	44.58 ± 5.84	55.43 ± 1.29	46.88 ± 1.68
Segment	96.27 ± 0.65	96.30 ± 0.63	88.42 ± 1.05	92.09 ± 0.86	70.85 ± 2.44

that we have to train different optimal kernel parameters, which is time expensive. Moreover, in general, RMEKLM owns a competitive classification performance to MultiK-MHKS, sometimes even better, e.g Wdbc and Transfusion. But the standard deviation of RMEKLM is bigger than that of MultiK-MHKS on most data sets. We think the reason is that the implementation on computer is an approximate value during the Gauss Elimination process, so we lose some information of the original feature space. But for it, RMEKLM well inherits the advantage of the MultiK-MHKS in terms of the classification performance.

Performance of Gauss Elimination and random descending

In this section, we discuss the performance of orthonormal basis generation algorithms. In literature Harmeling et al. (2003), before formulating the algorithm the author defines

the function $rk(n)$ for numbers n by the following process: randomly pick n points $v_1, \dots, v_n$ from the inputs and compute the rank of the corresponding $n \times n$ matrix $\Phi_v^T \Phi_v$. The orthonormal basis is obtained by: (1) start with a large r with $rk(r) < r$. (2) Decrement r by one as long as $rk(r) < r$ holds. As soon as $rk(r) = r$ been obtained, r was found. Choose $v_1, \dots, v_r$ as the vectors that achieve rank r . The problem is that the time complexity of computing the rank of matrix is $O(n^3)$, where n denotes the scale of the inputs. And in order to get the largest rank, we have to do this computation several times, which make the total time complexity close to $O(n^4)$. It is intolerable especially on the large scale data sets. By using the Gauss Elimination, the total time complexity reduces to $\frac{2}{3}O(n^3)$. We can see the significant improvements in Fig. 4. Here, we use all the samples of each data sets to generate an orthonormal basis.

On the other hand, the orthonormal bases generated by Random Descending are unstable because of the

Fig. 4 Time Comparison for generating an orthonormal basis between Gauss Elimination (GE) and Random Descending (RD) of three kernel mappings on UCI data sets listed in Table 2

Table 4 Rank of orthonormal space generated by Gauss Elimination (GE) and Random Descending (RD) on UCI data sets listed in Table 2

Datasets	Linear			Polynomial			RBF		
	Rank	GE	RD	Rank	GE	RD	Rank	GE	RD
Iris	4	4	4	15	15	15	147	147	147
Wine	13	13	13	70	57 ↓	58 ↓	178	178	178
Wdbc	30	29 ↓	28 ↓	85	69 ↓	68 ↓	569	569	569
Balance	4	4	4	15	15	15	624	624	619 ↓
Transfusion	3	3	3	9	5 ↓	8 ↓	502	502	27 ↓
Pima	8	8	8	45	44 ↓	43 ↓	768	768	768
Masses	6	6	6	27	27	19 ↓	689	689	85 ↓
CMC	9	9	9	52	52	46 ↓	1,358	1,358	248 ↓
Segment	15	15	14 ↓	110	81 ↓	90 ↓	2,086	2,086	341 ↓

Rank denotes the rank of the feature space

randomness. Sometimes the rank of the generated basis is smaller than that of the feature space, that is to say $r < \text{rank}(\Phi_x^T \Phi_x)$, which leads to losing some information of the feature space. On the contrary, Gauss Elimination does not have such problem. More detail is shown in Table 4, where the symbol “↓” denotes the rank loss. We can find that Random Descending works worst in the RBF kernel space, because of its high dimension.

Conclusions and future work

In this paper, we propose a reduced multiple empirical kernel learning machine named RMEKLM for short.

RMEKLM explicitly maps the input data points onto an orthonormal basis of the feature space. Compared with MultiK-MHKS, RMEKLM possesses at least two advantages: (1) by mapping the input data points in the way of RMEKLM, the geometrical structure of the feature space is more visualized than that of MultiK-MHKS; (2) in doing so, it reduces the time and space complexity of EKM-based MKL, especially in the process of testing. RMEKLM needs to store only r vectors that can construct an orthonormal basis of the feature space to parameterize the inputs, where r is the rank of the feature space. In addition, by proving Theorem 1, we use the Gaussian Elimination to get the vectors from the inputs to construct an orthonormal basis in the feature space for $\frac{2}{3}O(N^3)$ time complexity, rather than

using the Random Descending method, which is unstable and needs close to $O(N^4)$ time complexity. The experiments results validate the effectiveness and the efficiency of the RMEKLM in terms of the classification performance, the time complexity and the stability and efficiency of the Gauss Elimination. In the future work, we will combine the RMEKLM with the cognitive neuroscience. For example, through sampling and extracting the features of EEG, we can get the new samples of EEG. Then, the RMEKLM can identify the lesions based on the new samples of EEG. Similarly, we can also use the RMEKLM to analyze both medical images and neural information.

Acknowledgments This work was partially supported by Natural Science Foundations of China under Grant Nos.61272198 and 21176077, Innovation Program of Shanghai Municipal Education Commission under Grant No.14ZZ054, the Fundamental Research Funds for the Central Universities, Shanghai Key Laboratory of Intelligent Information Processing of China under Grant No.IIPL-2012-003, and Provincial Key Laboratory for Computer Information Processing Technology of Soochow University.

References

- Bach FR, Lanckriet GR, Jordan MI (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the twenty-first international conference on machine learning. ACM, p 6
- Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, MA
- Demir C, Alpaydin E (2005) Cost-conscious classifier ensembles. *Pattern Recognit Lett* 26:2206–2214
- Farquhar J, Hardoon D, Meng H, Shawe-taylor JS, Szedmak S (2005) Two view learning: Svm-2k, theory and practice. In: Advances in neural information processing systems, pp 355–362
- Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis: an overview with application to learning methods. *Neural Comput* 16:2639–2664
- Harmeling S, Ziehe A, Kawanabe M, Müller K-R (2003) Kernel-based nonlinear blind source separation. *Neural Comput* 15:1089–1124
- Kim S-J, Magnani A, Boyd S (2006) Optimal kernel selection in kernel fisher discriminant analysis. In: Proceedings of the 23rd international conference on machine learning. ACM, pp 465–472
- Kreßel UH-G (1999) Pairwise classification and support vector machines. In: Advances in kernel methods. MIT Press, Cambridge, MA, pp 255–268
- Lanckriet GR, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. *J Mach Learn Res* 5:27–72
- Łęski J (2003) Ho-kashyap classifier with generalization control. *Pattern Recognit Lett* 24:2281–2290
- Muller K-R, Mika S, Ratsch G, Tsuda K, Scholkopf B (2001) An introduction to kernel-based learning algorithms. *Neural Netw IEEE Trans* 12:181–201
- Rakotomamonjy A, Bach F, Canu S, Grandvalet Y (2007) More efficiency in multiple kernel learning. In: Proceedings of the 24th international conference on machine learning. ACM, pp 775–782
- Rakotomamonjy A, Bach F, Canu S, Grandvalet Y et al (2008) SimpleMKL. *J Mach Learn Res* 9:2491–2521
- Scholkopf B, Mika S, Burges CJ, Knirsch P, Muller K-R, Ratsch G, Smola AJ (1999) Input space versus feature space in kernel-based methods. *Neural Netw IEEE Trans* 10:1000–1017
- Sonnenburg S, Rätsch G, Schäfer C (2006) A general and efficient multiple kernel learning algorithm. In: Advances in neural information processing systems 18(2006):1273–1280
- Sonnenburg S, Rätsch G, Schäfer C, Schölkopf B (2006) Large scale multiple kernel learning. *J Mach Learn Res* 7:1531–1565
- Subrahmanya N, Shin YC (2010) Sparse multiple kernel learning for signal processing applications. *Pattern Anal Mach Intell IEEE Trans* 32:788–798
- Szafranski M, Grandvalet Y, Rakotomamonjy A (2010) Composite kernel learning. *Mach Learn* 79:73–103
- Tsang IW, Kocsor A, Kwok JT (2006) Efficient kernel feature extraction for massive data sets. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 724–729
- Wang C, Zou J, Zhang J, Wang M, Wang R (2010) Feature extraction and recognition of epileptiform activity in EEG by combining PCA with ApEn. *Cogn Neurodyn* 4:233–240
- Wang Z, Chen S, Gao D (2011) A novel multi-view classifier based on Nyström approximation. *Expert Syst Appl* 38:11193–11200
- Wang Z, Chen S, Sun T (2008) Multik-MHKS: a novel multiple kernel learning algorithm. *Pattern Anal Mach Intell IEEE Trans* 30:348–353
- Williams C, Seeger M (2001) Using the Nyström method to speed up kernel machines. In: Advances in neural information processing systems 13, Citeseer
- Xiong H, Swamy M, Ahmad MO (2005) Optimizing the kernel in the empirical feature space. *Neural Netw IEEE Trans* 16:460–474
- Xu Q-S, Liang Y-Z (2001) Monte Carlo cross validation. *Chemom Intell Lab Syst* 56:1–11
- Xu Z, Jin R, Yang H, King I, Lyu MR (2010) Simple and efficient multiple kernel learning by group lasso. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 1175–1182
- Yan F, Mikolajczyk K, Barnard M, Cai H, Kittler J (2010) LP norm multiple kernel fisher discriminant analysis for object and image categorisation. In: Computer vision and pattern recognition (CVPR), 2010 IEEE conference on. IEEE, pp 3626–3632
- Yu Y, Wang B, Zhang L (2011) Bottom-up attention: pulsed pca transform and pulsed cosine transform. *Cogn Neurodyn* 5:321–332
- Zhang J-H, Qin P-P, Ratsch J, Wang R-B (2013) Predictive modeling of human operator cognitive state via sparse and robust support vector machines. *Cogn Neurodyn* 7:395–407