



Learning Distance Metric for Support Vector Machine: A Multiple Kernel Learning Approach

WeiQi Zhang¹ · Zifei Yan¹  · Gang Xiao² · Hongzhi Zhang¹ · Wangmeng Zuo¹

Published online: 27 June 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Recent work in distance metric learning has significantly improved the performance in k -nearest neighbor classification. However, the learned metric with these methods cannot adapt to the support vector machines (SVM), which are amongst the most popular classification algorithms using distance metrics to compare samples. In order to investigate the possibility to develop a novel model for joint learning distance metric and kernel classifier, in this paper, we provide a new parameterization scheme for incorporating the squared Mahalanobis distance into the Gaussian RBF kernel, and formulate kernel learning into a generalized multiple kernel learning framework, gearing towards SVM classification. We demonstrate the effectiveness of the proposed algorithm on the UCI machine learning datasets of varying sizes and difficulties and two real-world datasets. Experimental results show that the proposed model achieves competitive classification accuracies and comparable execution time by using spectral projected gradient descent optimizer compared with state-of-the-art methods.

Keywords Metric learning · Multiple kernel learning · Gaussian RBF kernel · Support vector machines

✉ Zifei Yan
cszfyang@gmail.com

WeiQi Zhang
weiqizhanghit@gmail.com

Gang Xiao
amote530@163.com

Hongzhi Zhang
zhanghz0451@gmail.com

Wangmeng Zuo
cswmzuo@gmail.com

¹ Harbin Institute of Technology, Harbin, China

² No. 211 Hospital of PLA, Harbin, China

1 Introduction

Metric learning, aiming at learning an appropriate distance metric that better represents the distance between data points, which pulls the semantically similar ones closer and pushes dissimilar ones farther away, plays an important role in many machine learning and pattern recognition algorithms [4, 13, 19, 21, 38, 50]. Traditional uniformed norms, such as Euclidean distance, tend to ignore the awareness of semantic definition encoded by sample labels in supervised learning, and fail to highlight the discriminative features in varying applications. As a generalized version of Euclidean distance, Mahalanobis distance can be seen as performing a linear projection of the data points firstly and then computing the Euclidean distance in the projected space. Recent years, a surge of innovation on learning the pseudo-metric towards Mahalanobis distance has been raising. Due to the learned distance reflecting the label information, mainstream Mahalanobis distance metric learning methods are usually geared towards k -nearest neighbor (kNN) classifiers, e.g., Large Margin Nearest Neighbor (LMNN) proposed by Weinberger et al. [44], Large Margin Component Analysis (LMCA) proposed by Torresani et al. [37], and Information-Theoretic Metric Learning (ITML) proposed by Davis et al. [7]. Except for methods that are specific to k -nearest neighbors and based on Mahalanobis distance, there are some recent methods that focus on the sparse metric learning, non-linear metric learning, and regularized metric learning [5, 16, 23, 27, 30, 43, 51, 53].

Support Vector Machines (SVMs) are also amongst the most popular classifiers for various pattern recognition problems. They have been widely used not only because of their excellent predictive performance but also because of their generalization ability supported by the solid generalization error bounds defined over maximum margin philosophy. More importantly, the kernel-trick [33] allows SVM to generate higher dimensional non-linear decision boundaries with low computational burden. In order to leverage the progress on metric learning, Nguyen et al. [31] formulated the problem as a quadratic semi-definite programming problem (QSDP) with local neighborhood constraints, which is based on the SVM framework. The proposed method called MLSVM outperforms the raw kNN classifier and metric learning methods based on kNN. Xu et al. [48] investigated the efficacy of three of the most popular Mahalanobis metric learning algorithms, Neighborhood Component Analysis (NCA) [17], LMNN [44] and ITML [7], as pre-processing for SVM training and found that none of them generated metrics particularly suitable for SVM classification. To solve this problem, they incorporated the distance metric learning idea into a single kernel learning framework, and proposed an efficient kernel classifier, Support Vector Metric Learning (SVML). Instead of modeling a particular distance metric, the decomposition of squared Mahalanobis distance, i.e., $\mathbf{M} = \mathbf{L}^T \mathbf{L}$, was learnt for SVM classification with a single kernel learning manner, which incorporated the squared Mahalanobis distance into the RBF kernel function and iteratively optimized \mathbf{L} via off-the-shelf SVM tools.

Many efforts have been devoted to designing kernel learning algorithms. Among them, Multiple Kernel Learning (MKL) has been considered as an effective and efficient way to achieve this goal [2, 3, 6, 24, 25, 32, 35, 40, 46], where the optimal kernel is modeled as a linear combination of a set of basis kernels. MKL based SVM methods have been used to solve kinds of classification tasks in the last decades [20, 36, 45, 52]. Lanckriet et al. [25] formulated MKL as a quadratically constrained quadratic programming problem, using a ℓ_1 -norm constraint to promote sparse combinations. Considering that the model in [25] is non-smooth, Bach et al. [3] proposed a smoothed version and proposed a SMO-like algorithm for solving the problem. Sonnenburg et al. [35] reformulated the problem as a semi-infinite linear program and addressed the problem by iteratively solving a classical SVM problem. Rako-

tomamonjy et al. [32] further developed a SimpleMKL algorithm, and demonstrated that the training time could be further reduced by nearly an order of magnitude on some standard UCI datasets when the number of kernels was large. To further improve the representative ability of combined multiple kernels, non-linear combinations of kernels have been recently considered. Varma et al. [40] developed the Generalized Multiple Kernel Learning (GMKL) formulation, which allowed fairly general kernel parameterization, including both linear and non-linear kernel combinations, together with general regularizations on the kernel parameters. Hierarchical multiple kernel learning [2] learned a linear combination of an exponential number of linear kernels and represented them as a product of sums, classified to a non-linear kernels combination. In [6] Cortes et al. analyze non-linear combination in the case of regression and the kernel ridge regression algorithm, making improvement for regression problem in high dimensions. Besides, to allow for robust kernel mixtures Kloft et al. [24] extended MKL to arbitrary norms, and Xu et al. [46] presented a soft margin perspective for MKL. Both of these two methods achieve an effective and sparse solution for MKL. Except for methods that focus on linear or non-linear mixture of basis kernels, efforts are made to other kinds of approaches to avoid searching for the optimal combination parameters directly. [18] proposed a methods to determine the kernels to be preserved and weighted according to the statistical significance. A radius-margin based MKL algorithm with monotone conjunctive kernels was proposed in [26]. [1] posed the problem of learning the kernel combination as a min-max problem solved by kernelized optimization of the margin distribution (KOMD) algorithm.

In this paper, we suggest a novel method to leverage the progress on multiple kernel learning and promote the classification performance of SVM from the viewpoint of metric learning, motivated by the superiority of SVML to representative metric learning methods such as ITML, NCA and LMNN. We formulate the Mahalanobis distance into the Gaussian RBF kernel, which behaves as the basis kernel inborn with the thinking of metric learning, and incorporate it into a GMKL framework. Benefitted from this formulation, we can adopt GMKL algorithm for joint learning distance metric and kernel classifier. While in the test stage we can utilize the composition of linear transformation and SVM for classification. The complexity of our model is independent with the number of basis kernels, and thus can avoid the computational burden issue of conventional MKL approach. Moreover, radius information is also incorporated as the supplement of considering the within-class distance of input data. Extensive experiments on UCI datasets and real world datasets classification clearly demonstrate the effectiveness of the proposed methods.

This algorithm, which we refer to as Metric Learning-based Multiple Kernel Classifiers (MLMKC), is particularly contributed to two aspects.

- First, by formulating the Mahalanobis distance into the Gaussian RBF kernel, which behaves as the basis kernel implanted within-class minimization via additional regularization inherited from Gaussian kernel, the involved kernel matrix can be obtained by computing the pairwise distance over given pairs of instances in the original feature space, instead of transforming to complicated high dimension feature spaces.
- Second, the learning of the distance metric kernel can be effortlessly incorporated into the Generalized MKL framework, and the flexibility of the selection of kernel size can tremendously reduce the computation overhead of whole MKL framework.

The remainder of this paper is organized as follows. We review the related work in Sect. 2. In Sect. 3, we formulate the Mahalanobis distance into the Gaussian RBF kernel and jointly learn the distance metric via GMKL framework for SVM-based classification. In Sect. 4, we

evaluate the proposed algorithm on several benchmark datasets and two real-world datasets, and finally conclude this work in Sect. 5.

2 Related Work

Consider a binary classification problem in which a training set of n samples is denoted by $\mathcal{X} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i th training sample, $y_i \in \{+1, -1\}$ denotes the class label of \mathbf{x}_i . SVMs learn a hyperplane $H_{\mathbf{w}} : \mathbf{w}^T \mathbf{x} + b = 0$ which maximizes the margin between the two classes. The conventional SVM model can be formulated as:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i, \\ \text{s.t.} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i \end{aligned} \tag{1}$$

where ξ denotes the slack variable.

The performance of kernel based SVM strongly depends on the choice of kernels. A common approach developed to find the appropriate kernel function and parameters for heterogeneous data of varying applications is called MKL, where each kernel encodes a different modality of data. The same technique is suggested by other methods, e.g., dictionary learning [9], hashing based coding [10] and quantization based coding [49] in similarity search. Let $\varphi_p : \mathbb{R}^d \rightarrow \mathcal{H}_p$ be the p th feature mapping, inducing the corresponding basis kernel $K_p(\cdot, \cdot)$ in the Hilbert space \mathcal{H}_p , where $p = 1, \dots, m$. In the MKL framework based on SVM, each sample \mathbf{x} is mapped onto m feature spaces by $\varphi(\mathbf{x}; \boldsymbol{\gamma}) = [\sqrt{\gamma_1}\varphi_1(\mathbf{x}), \dots, \sqrt{\gamma_m}\varphi_m(\mathbf{x})]^T$, where γ_p is the weight of the p th basis kernel. Therefore, the employed kernel can be expressed as a linear or nonlinear combination of the basis kernels, expressed as $K(\cdot, \cdot; \boldsymbol{\gamma}) = \sum_{p=1}^m \gamma_p K_p(\cdot, \cdot)$ or $K(\cdot, \cdot; \boldsymbol{\gamma}) = \prod_{p=1}^m (K_p(\cdot, \cdot))^{\gamma_p}$, the formulation of which depends on the needs of different applications. To seek the optimal combination weight for each basis kernel, most MKL approaches [32,35,47] are suggested to solve the following objective function:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i, \\ \text{s.t.} \quad & y_i(\mathbf{w}^T \varphi(\mathbf{x}_i; \boldsymbol{\gamma}) + b) \geq 1 - \xi_i, \quad \forall i, \\ & \xi_i \geq 0, i = 1, 2, \dots, n, \|\boldsymbol{\gamma}\|_1 = 1, \boldsymbol{\gamma} > 0, \end{aligned} \tag{2}$$

where \mathbf{w} is the normal of the separating hyperplane, b is the bias term, ξ is the slack variable introduced to address the linearly non-separable problem.

Some studies have been given to introduce multiple kernels into metric learning problems. Wang et al. [42] learned a distance metric in the weighted linear combinations of feature spaces by exploring the potential correlations between different kernels, the formulation of which can be seen as learning a projection from the weighted combination of mapped feature space to an embedding space. Rather than projecting to the single embedding space kernelized by the weighted combination of multiple basis feature mappings, McFee et al. [29] defined the embedding as the concatenation of different projections, allowing the algorithm to learn an ensemble of projections tailored to its corresponding domain space and jointly optimized to produce an optimal space. By restricting each projection weight matrix to be diagonal, literature [29] implicitly weighted the contribution of each kernel with respect to corresponding training point in construction of the embedding. It is claimed that the weighted combination kernels embedding formulation is much less flexible than the concatenated

projection embedding formulation, as the former applies the same projection to each feature space. As a weighted version of [29], Lu et al. [28] introduced weights to the unweighted concatenated projection formulation by jointly learning multiple metrics and weights via SVM-based SimpleMKL framework. However, the learned metrics and their corresponding weights are finally exported to a kNN classifier, which is still enduring the deficiency of having to store the entire training set during the test stage. Although the literature did not provide the results of training time, the huge computation burden can be expected.

Under the scenarios that the radius of the smallest data enclosing sphere is no longer fixed, optimizing over both the margin and the radius in an SVM-based problem can be expected to achieve a tighter generalization error bound. However, optimizing over both of them poses several difficulties since the radius is computed in a complex form [15]. Do et al. [12] proposed ϵ -SVM, which in addition to the margin maximization also minimized the within-class distance through the sum of the instance distances from the margin hyperplane. Except providing the interpretation that LMNN can be seen as a set of local related ϵ -SVMs in the quadratic space, their experimental results also demonstrated that ϵ -SVM performed much more pronounced in case of using Gaussian kernel, which could be attributed to its additional regularization on the within-class distance which makes it more appropriate for high dimensional spaces.

Instead of using the ratio, Do et al. [11] proved that the sum of the radius and the inverse of the margin can achieve the same optimal solution under proper parameter choices. Their R-SVM $^+$ $_{\mu}$ is formulated as:

$$\begin{aligned}
 \min_{\mathbf{w}, b, \mu, R_{\mu}, x_0} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \lambda R_{\mu}^2 + C \sum_{i=1}^n \xi_i, \\
 \text{s.t.} \quad & y_i (\langle \mathbf{w}, D_{\sqrt{\mu}} \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \forall i, i = 1, 2, \dots, n, \\
 & \sum_{k=1}^d \mu_k = 1, \mu, \xi \geq 0, \\
 & \|D_{\sqrt{\mu}} \mathbf{x}_i - D_{\sqrt{\mu}} \mathbf{x}_0\| \leq R_{\mu}^2, \quad \forall i,
 \end{aligned} \tag{3}$$

where $\sqrt{\mu}$ is a vector scaling the transformed feature space; $D_{\sqrt{\mu}}$ is a diagonal linear transformation matrix, whose diagonal elements are given by $\sqrt{\mu}$, thus $D_{\sqrt{\mu}} \mathbf{x}$ gives the image of an instance \mathbf{x} in the transformed feature space; R_{μ} denotes the radius of the scaled feature space. Their optimization problem solves SVM from the viewpoint of metric learning, and approximated the radius of the smallest sphere enclosing the data by the maximum pairwise distance over all pairs of instances in the transformed feature space, which may result in very inefficient to solve large-scale problems.

3 Multiple Kernel Classifiers Based on Metric Learning

In this section, by extending Gaussian RBF kernel, we formulate the joint distance metric and kernel classifier learning problem into the Generalized MKL (GMKL) framework. We then present the learning algorithm for solving the proposed model, and radius information is further incorporated to impose the regularization on minimizing the enclosing ball of data in the feature space endowed with the learned kernel.

3.1 Extension of Gaussian RBF Kernel

Gaussian Radial Basis Function (RBF) kernel is a kernel function which has been widely adopted in various kernel methods. Given two samples \mathbf{x} and \mathbf{y} , the Gaussian RBF kernel is defined as

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{\sigma^2}\right) = \exp(-d_\sigma^2(\mathbf{x}, \mathbf{y})), \tag{4}$$

where $d_\sigma^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2/\sigma^2$ can be explained as a scaled version of squared Euclidean distance. Here we further extend Gaussian RBF kernel by replacing the scaled Euclidean distance $d_\sigma(\mathbf{x}, \mathbf{y})$ with the Mahalanobis distance,

$$d_M^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{M}(\mathbf{x} - \mathbf{y}), \tag{5}$$

where the matrix $\mathbf{M} \in \mathcal{R}^{d \times d}$ is semi-positive definite. The resulting extended kernel function is then defined as follows:

$$K_M(\mathbf{x}, \mathbf{y}) = \exp(-d_M^2(\mathbf{x}, \mathbf{y})) = \exp(-(\mathbf{x} - \mathbf{y})^T \mathbf{M}(\mathbf{x} - \mathbf{y})), \tag{6}$$

It is easy to see that $K_M(\mathbf{x}, \mathbf{y})$ is a kernel function and satisfies the Mercer condition [34].

Rather than directly learning \mathbf{M} , Xu et al. [48] parameterized $\mathbf{M} = \mathbf{L}^T \mathbf{L}$, and suggested a gradient descent algorithm with respect to \mathbf{L} , which is incorporated into a single RBF kernel. Inspired by the metric learning with multiple kernel embedding proposed by Lu et al. [28] and Doublet-SVM metric learning methods proposed by Wang et al. [41], we parameterize \mathbf{M} as,

$$\begin{aligned} \mathbf{M} &= \sum_{l=1}^{N_k} \beta_l (\mathbf{x}_{l,1} - \mathbf{x}_{l,2})(\mathbf{x}_{l,1} - \mathbf{x}_{l,2})^T + \lambda_G \mathbf{I} \\ &= \sum_{l=1}^{N_k} \beta_l \mathbf{X}_l + \lambda_G \mathbf{I}, \beta_l \geq 0, \lambda_G > 0 \end{aligned} \tag{7}$$

where $\mathbf{X}_l = (\mathbf{x}_{l,1} - \mathbf{x}_{l,2})(\mathbf{x}_{l,1} - \mathbf{x}_{l,2})^T$, β_l is the weight describing the contribution of the sample pair $(\mathbf{x}_{l,1}, \mathbf{x}_{l,2})$ to make up the matrix \mathbf{M} . When all of $\beta_l = 0$, $K_M(\mathbf{x}, \mathbf{y})$ is identical to a standard Gaussian RBF kernel. With Eq. (6), the kernel can be reformulated as:

$$\begin{aligned} K_M(\mathbf{x}, \mathbf{y}; \boldsymbol{\beta}) &= \exp(-(\mathbf{x} - \mathbf{y})^T \left(\sum_{l=1}^{N_k} \beta_l \mathbf{X}_l + \lambda_G \mathbf{I} \right) (\mathbf{x} - \mathbf{y})) \\ &= K_{\lambda_G} \prod_{l=1}^{N_k} (\exp(-(\mathbf{x} - \mathbf{y})^T \mathbf{X}_l (\mathbf{x} - \mathbf{y})))^{\beta_l} \\ &= K_{\lambda_G} \prod_{l=1}^{N_k} (K_l(\mathbf{x}, \mathbf{y}))^{\beta_l} \end{aligned} \tag{8}$$

When fixing λ_G , the latter denoted by $K_{\lambda_G} = \exp(-\lambda_G (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}))$ can be treated as a constant. By defining the basis kernel $K_l = \exp(-(\mathbf{x} - \mathbf{y})^T \mathbf{X}_l (\mathbf{x} - \mathbf{y}))$, the kernel function in Eq. (8) can also be explained from the multiple kernel perspective. If we take all the sample pairs into account, the number of basis kernels is $N_k = n(n - 1)/2$, which will be too huge for large scale dataset. To address this, one can adopt the following strategies to reduce N_k :

(i) We can refer to the metric learning methods [41] by only using the nearest similar pairs and the nearest dissimilar pairs to construct the set of basis kernels. (ii) After selecting pairs based on (i), some clustering methods (e.g., *k*-means) can be adopted to further reduce the number of pairs for constructing basis kernels.

3.2 Formulation of Metric Learning-Based Multiple Kernel Classifier

Denote by $\varphi_M(\mathbf{x}; \boldsymbol{\beta})$ the feature mapping associated with the kernel function $\mathbf{K}_M(\mathbf{x}, \mathbf{y}; \boldsymbol{\beta})$. Our objective is to learn a function of the form $f(\mathbf{x}) = \mathbf{w}^T \varphi_M(\mathbf{x}; \boldsymbol{\beta}) + b$. Given $\varphi_M(\mathbf{x}; \boldsymbol{\beta})$, one can adopt the SVM solver to learn the global optimal values of (\mathbf{w}, b) from the training data $\{(\mathbf{x}_i, y_i)_{i=1}^n\}$. If we want to jointly learn $\boldsymbol{\beta}$ and (\mathbf{w}, b) , we should consider the MKL framework. Therefore, we adopt the GMKL formulation in [40],

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\beta}} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_i l(y_i, f(\mathbf{x}_i)) + r(\boldsymbol{\beta}) \\ \text{s.t.} \quad & \boldsymbol{\beta} \geq 0. \end{aligned} \tag{9}$$

where both the regularizer $r(\boldsymbol{\beta})$ and the kernel should be differentiable with respect to $\boldsymbol{\beta}$, and $l(\cdot)$ could be some loss functions, e.g., hinge loss and logistic loss.

In order to learn the classifier and the Mahalanobis distance metric jointly, we formulate the problem as:

$$\begin{aligned} \{\mathbf{w}, b, \mathbf{M}, \boldsymbol{\beta}\} = \arg \min_{\mathbf{w}, b, \mathbf{M}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i + \lambda r(\boldsymbol{\beta}), \\ \text{s.t.} \quad & y_i((\mathbf{w}, \varphi_M(\mathbf{x}_i; \boldsymbol{\beta})) + b) \geq 1 - \xi_i, \\ & \boldsymbol{\beta} \geq 0, \xi_i \geq 0, \quad \forall i, \end{aligned} \tag{10}$$

Reformulating above primal as a nested two step optimization, the kernel is learned by optimizing over $\boldsymbol{\beta}$ in the outer loop, while the kernel is fixed and the SVM parameters are learnt in the inner loop. This can be achieved by rewriting the primal as follows:

$$\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}), \text{ s.t. } \boldsymbol{\beta}_l \geq 0, \tag{11}$$

with

$$\begin{aligned} J(\boldsymbol{\beta}) = \max \quad & \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Y} \mathbf{K}_M(\boldsymbol{\beta}) \mathbf{Y} \boldsymbol{\alpha} + \lambda r(\boldsymbol{\beta}), \\ \text{s.t.} \quad & \mathbf{1}^T \mathbf{Y} \boldsymbol{\alpha} = 0, 0 \leq \boldsymbol{\alpha} \leq C, \end{aligned}$$

where \mathbf{Y} is a diagonal matrix with the labels on the diagonal, λ is a tradeoff to balance the regularization part. As to $r(\boldsymbol{\beta})$, its derivative should exist and be continuous. For example, the non-negative ℓ_1 -norm regularization $r(\boldsymbol{\beta}) = \mathbf{1}^T \boldsymbol{\beta}$ could be used for learning sparse solutions, or the ℓ_2 -norm regularization $r(\boldsymbol{\beta}) = (\boldsymbol{\beta} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu})$ if prior knowledge is available. If $\nabla_{\boldsymbol{\beta}} \mathbf{K}_M$ and $\nabla_{\boldsymbol{\beta}} r$ are differentiable functions of $\boldsymbol{\beta}$, we can utilize gradient descent in the outer loop, and $J(\boldsymbol{\beta})$ has derivatives given by

$$\frac{\partial J}{\partial \beta_l} = \lambda \frac{\partial r}{\partial \beta_l} - \frac{1}{2} \boldsymbol{\alpha}^{*T} \frac{\partial \mathbf{Y} \mathbf{K}_M \mathbf{Y}}{\partial \beta_l} \boldsymbol{\alpha}^*, \tag{12}$$

In order to take a gradient step, in the inner loop, all we need to do is to obtain $\boldsymbol{\alpha}^*$, which can be solved by any off-the-shelf SVM optimization package. To solve the non-convex

Algorithm 1: MLMKC with spectral projected gradient descent optimizer.

- Input:** Training samples $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, kernel number N_k , the number of similar sample pairs n_1 and dissimilar sample pairs n_2 .
Output: β, α .
1. For each sample \mathbf{x}_i , find its n_1 nearest similar neighbors $(\mathbf{x}_i, \mathbf{x}_{i,1}^s), \dots, (\mathbf{x}_i, \mathbf{x}_{i,n_1}^s)$, n_2 nearest dissimilar neighbors $(\mathbf{x}_i, \mathbf{x}_{i,1}^d), \dots, (\mathbf{x}_i, \mathbf{x}_{i,n_2}^d)$;
 2. Construct \mathbf{X}_k : compute the difference of these doublets and use k-means find N_k cluster centers $\{\mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_k}, \dots, \mathbf{x}_{c_{N_k}}\}$, $\mathbf{X}_k = \mathbf{x}_{c_k} \mathbf{x}_{c_k}^T$;
 3. $n \leftarrow 0$;
 4. Randomly initialize β^0 from $[0, 1]$;
 5. **repeat**
 6. Compute the kernel matrix \mathbf{K}_M from β^n ;
 7. $\mathbf{K}_M \leftarrow \mathbf{K}_M(\beta^n)$;
 8. Call libSVM to obtain α^* ;
 9. $\alpha^* \leftarrow \text{SolveSVM}_\epsilon(\mathbf{K}_M)$
 10. $\lambda \leftarrow \text{SpectralStepLength}$
 11. $p^n = \beta^n - \max(\beta^n - \lambda \frac{\partial J}{\partial \beta^n}, 0)$;
 12. $s^n \leftarrow \text{Non-Monotone}$
 13. $\epsilon \leftarrow \text{TuneSVMprecision}$
 14. $\beta_k^{n+1} = \beta_k^n - s^n p^n$;
 15. $n \leftarrow n + 1$;
 16. **Until** β and α^* converge
-

formulation results from regularizing β , the spectral projected gradient (SPG) descent optimizer [22] can be adopted to update it. SPG iteratively approximates the objective function with a quadratic model and then optimizes the model at each iteration, which includes spectral step length and spectral projected gradient computation, the non-monotone line search and the SVM solver precision tuning. The proposed MLMKC algorithm is summarized in **Algorithm 1**. From **Algorithm 1**, we can see that the projection operator needs to be applied only once per iteration without the step variable, for which SPG employs a non-monotone line search criterion instead of the Armijo rule. The tolerance of the SVM solver automatically decreases as moving closer to the optimum or the step size becoming too small. These guarantee the robustness to imprecisions in calculating the objective function and the gradient, and reduce the number of SVM evaluations. The duality gap is used as a stable stopping criterion for SPG, or the projected gradient if the duality gap is unavailable.

Therefore, after we obtain the optimal α^* , b^* and β , we can write the SVM decision function as:

$$\begin{aligned}
 f(\mathbf{x}) &= \text{sgn} \left(\sum_{i=1}^n \alpha_i^* y_i \mathbf{K}_M(\mathbf{x}_i, \mathbf{x}; \beta) + b^* \right) \\
 &= \text{sgn} \left(\sum_{i=1}^n \alpha_i^* y_i \left(K_{\lambda_G} \prod_{l=1}^{N_k} (K_l(\mathbf{x}_i, \mathbf{x}))^{\beta_l} \right) + b^* \right).
 \end{aligned}
 \tag{13}$$

3.3 Computational Complexity

As illustrated in **Algorithm 1**, for each training sample, n_1 similar nearest neighbors and n_2 dissimilar nearest neighbors are chosen to construct $n_1 + n_2$ doublets, and in total $N(n_1 + n_2)$ doublets. After computing the difference of these doublets, we use k-means to find N_k cluster centers to construct the distance matrix \mathbf{M} as Eq.(7), of which the computational

complexity is $O(N(n_1 + n_2)d) + O(N_k N(n_1 + n_2)) + O((n_1 + n_2)d^2)$, where d is the dimension of the data feature. In Sect. 4 we will show that we can choose $n_1, n_2, N_k \ll N$ in practice, and thus, the computational cost can be reduced to $O(Nd) + O(N) + O(d^2)$. According to Eq. (8), the computational cost of computing the kernels K_M is $O(N_k N^2 d)$. For solving the alternating subproblem, the LibSVM library for SVM training is utilized, and the computational complexity of SMO-type algorithms is $O(N_k^2 N^2 d)$.

3.4 Incorporating the Radius Information

Although the Gaussian kernel implicitly imposed the regularization on minimizing the within class distance, considering the fact that the generation error of SVM is actually a function of the ratio of radius and margin [39], especially for joint learning of kernels for feature transformation and classifier, the radius information should be valuable to improve the performance of the algorithm. Due to the close relationship with the radius of minimum enclosing ball, instead of incorporating the radius directly, the trace of the total scattering matrix of training data is integrated into the proposed MLMKC framework, with the manner of summation rather than ratio as proposed in [11]. Therefore, the proposed model can be reformulated as

$$\begin{aligned} \{\mathbf{w}, b, \mathbf{M}, \boldsymbol{\beta}\} = \arg \min_{\mathbf{w}, b, \mathbf{M}} & \frac{1}{2} \|\mathbf{w}\|_2^2 + \rho \text{tr}(\mathbf{S}_T^\beta) + C \sum_i \xi_i + \lambda r(\boldsymbol{\beta}) \\ \text{s.t.} & y_i(\langle \mathbf{w}, \varphi_{\mathbf{M}}(\mathbf{x}_i; \boldsymbol{\beta}) \rangle + b) \geq 1 - \xi_i, \\ & \boldsymbol{\beta} \geq 0, \xi_i \geq 0, \quad \forall i, \end{aligned} \tag{14}$$

Thus the corresponding dual problem changes to

$$\begin{aligned} \min_{\boldsymbol{\beta}} & J(\boldsymbol{\beta}), \quad \text{s.t.} \quad \beta_l \geq 0 \\ J(\boldsymbol{\beta}) = \max & \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Y} \mathbf{K}_M(\boldsymbol{\beta}) \mathbf{Y} \boldsymbol{\alpha} + \rho \text{tr}(\mathbf{S}_T^\beta) + \lambda r(\boldsymbol{\beta}), \\ \text{s.t.} & \mathbf{1}^T \mathbf{Y} \boldsymbol{\alpha} = 0, 0 \leq \boldsymbol{\alpha} \leq C \end{aligned} \tag{15}$$

where $\text{tr}(\mathbf{S}_T^\beta)$ denotes the trace of the total scatter matrix in feature space mapped via $\mathbf{K}_M(\boldsymbol{\beta})$, which can be explicitly expressed in the kernel-induced feature space as

$$\text{tr}(\mathbf{S}_T^\beta) = \text{tr}(\mathbf{K}_M(\boldsymbol{\beta})) - \frac{1}{n} \mathbf{1}^T \mathbf{K}_M(\boldsymbol{\beta}) \mathbf{1} = \prod_{l=1}^{N_k} \alpha_l^{\beta_l}, \tag{16}$$

where $\alpha_l \triangleq \text{tr}(\mathbf{K}_l) - (1/n) \mathbf{1}^T \mathbf{K}_l \mathbf{1}$, and $\mathbf{K}_M(\boldsymbol{\beta})$ is the kernel matrix formulated in Eq.(8) which can be obtained by computing the pairwise distance over all pairs or given pairs of instances in the original feature space on training set \mathcal{X} . After incorporating the radius term, the derivatives of $J(\boldsymbol{\beta})$ in **Algorithm 1** becomes

$$\frac{\partial J}{\partial \beta_l} = \lambda \frac{\partial r}{\partial \beta_l} - \frac{1}{2} \boldsymbol{\alpha}^{*T} \frac{\partial \mathbf{Y} \mathbf{K}_M \mathbf{Y}}{\partial \beta_l} \boldsymbol{\alpha}^* + \rho \frac{\partial \text{tr}(\mathbf{S}_T^\beta)}{\partial \beta_l}, \tag{17}$$

4 Experiments and Discussion

In this section, we first discuss the setting for λ_G and N_k in Eq. (8). Then we evaluate the proposed MLMKC algorithm using synthetic datasets with two classes whose distribution is

Table 1 The UCI datasets used in experiments

Dataset	# of samples	Feature dimension	# of class1	# of class2
ACA	690	14	307	383
CMC	962	9	629	333
Mammo	830	5	403	427
Iono	351	34	225	126
Heart	303	13	139	164
Sonar	208	60	97	111
Liver	345	6	145	200
Wpbc	198	33	151	47
Pima	768	8	268	500
Vote	435	16	168	267
Musk	6598	166	5581	1017

Table 2 The real-world datasets used in experiments

Dataset	# of training samples	# of test samples	Feature dimension	PCA dimension	# of classes
COIL20	1140	300	1024	500	20
USPS	7291	2007	256	100	10

nonlinear separable. After that, eleven datasets from UCI Machine Learning repository [14] of varying size, dimensionality and task description are used to evaluate the performance of our method and the state-of-the-art: Australian Credit Approval (ACA), Contraceptive Method Choice (CMC), Mammographic Mass (Mammo), Musk, Ionosphere (Iono), Heart, Sonar, Pima, Vote, Wpbc, and Liver. The dataset statistics are shown in Table 1. We compare the proposed methods with SVM, GMKL and other representative multiple kernel learning methods, i.e., SimpleMKL, RMKL, ℓ_p -MKL, SM1MKL, EasyMKL and RM-GD in terms of classification accuracy and training time, for the reason that we incorporated distance metric into the multiple kernel learning framework. All the experiments on UCI datasets use the following setting: for each dataset, 80% of the data is used for training and the rest is used for test. All data sets have been normalized to have zero mean and unit variance on each feature. Apart from the UCI datasets, we also perform experiments on the large scale multiclass real-world datasets, i.e., handwritten digit set USPS dataset¹ and object classification set COIL20 dataset.² We use the defined training set to train the models using a 1-vs-rest strategy and calculate the classification error rates on the test set. As the dimensions of these two datasets are relatively high, PCA is utilized to reduce the feature dimension. Table 2 summarizes the basic information of the large scale real-world datasets.

We use libSVM³ to solve the SVM dual problem of our method and the parameter C is chosen from $\{0.1, 1, 10, 10^2\}$. The regularization parameter λ and ρ are chosen from $\{1, 10^2, 10^3, 10^4\}$, and the width σ^2 of Gaussian kernel is settled as 1. As MLMKC is not particularly sensitive to the exact choice of λ (i.e., the regularization parameter in Eq. (15))

¹ <https://www-i6.informatik.rwth-aachen.de/~keyzers/usps.html>.

² <http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php>.

³ <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

when $\lambda > 10^2$, and during the experiments we also find that for better results λ and ρ is often set to 10^2 , so we set them all for default. All the experiments are executed in a PC with 4 Intel Core Xeon E3-1230 V2 CPUs (3.3 GHz) and 32GB RAM.

4.1 Parameter Selection

4.1.1 A Reduced Version of MLMKC

Except for several aforementioned standard parameters, the two parameters involved in Eq. (8), i.e., λ_G and N_k , tightly correlate to the classification performance of the proposed model. Here we introduce a reduced version of MLMKC by setting $\lambda_G = 0$, represented as R-MLMKC, the kernel of which is formulated as:

$$\begin{aligned}
 K_{M^-}(\mathbf{x}, \mathbf{y}; \boldsymbol{\beta}) &= \prod_{l=1}^{N_k} \exp(-(\mathbf{x} - \mathbf{y})^T \beta_l \mathbf{X}_l (\mathbf{x} - \mathbf{y})) \\
 &= \prod_{l=1}^{N_k} (\exp(-(\mathbf{x} - \mathbf{y})^T \mathbf{X}_l (\mathbf{x} - \mathbf{y})))^{\beta_l}. \tag{18} \\
 &= \prod_{l=1}^{N_k} (K_l(\mathbf{x}, \mathbf{y}))^{\beta_l}
 \end{aligned}$$

M^- is formulated as $M^- = \sum_{l=1}^{N_k} \beta_l (\mathbf{x}_{l,1} - \mathbf{x}_{l,2})(\mathbf{x}_{l,1} - \mathbf{x}_{l,2})^T = \sum_{l=1}^{N_k} \beta_l \mathbf{X}_l$. We use this formulation to get rid of the consideration on parameter λ_G , and try to verify the effective working range of N_k .

In process of choosing sample pairs to construct the matrix M^- , instead of using all pairs of training samples, for each training sample \mathbf{x}_i , we construct $n_1 + n_2$ doublets $(\mathbf{x}_i, \mathbf{x}_{i,1}^s), \dots, (\mathbf{x}_i, \mathbf{x}_{i,n_1}^s), (\mathbf{x}_i, \mathbf{x}_{i,1}^d), \dots, (\mathbf{x}_i, \mathbf{x}_{i,n_2}^d)$, where $\mathbf{x}_{i,k}^s$ denotes the k th similar nearest neighbor of \mathbf{x}_i , and $\mathbf{x}_{i,k}^d$ denotes the k th dissimilar nearest neighbor of \mathbf{x}_i . After that we compute the difference of these doublets and use k -means to find cluster centers, whose number equals to the predefined kernel size. And then we use the cluster centers to get \mathbf{X}_l . We initialize the R-MLMKC with different numbers of kernel size, ranging from 4 to 40, and set $n_1 = n_2 = 2$. We choose three UCI datasets, whose feature dimensions are of different magnitude, to test the influence on the classification accuracy when varying the size of kernel. The curves of classification accuracy versus kernel size for R-MLMKC are shown in Fig. 1. One can see that, the accuracy tends to be stable when $N_k \geq 20$ on all three datasets. When $\lambda_G > 0$, we choose $4 \leq N_k \leq 20$ in experiment. If the accuracy is not very insensitive to N_k we can choose a small number of kernels instead which can dramatically reduce the computational cost. Experiment on larger number of N_k is needed when choosing the best size of basis kernels for large datasets, and the upper bound is decided by the execution time and memory. In Sect. 4.1.2, we will give a detailed discussion to the selection of λ_G and N_k .

4.1.2 Selection for λ_G and N_k

We first discuss the setting for λ_G . Leveraging the results of the observation on selecting different kernel sizes for R-MLMKC, small kernel size, i.e., N_k , which is the summation of n_1 , the number of similar pairs, and n_2 , the number of dissimilar pairs, can be used to observe the variation on classification performance when changing the model with varying

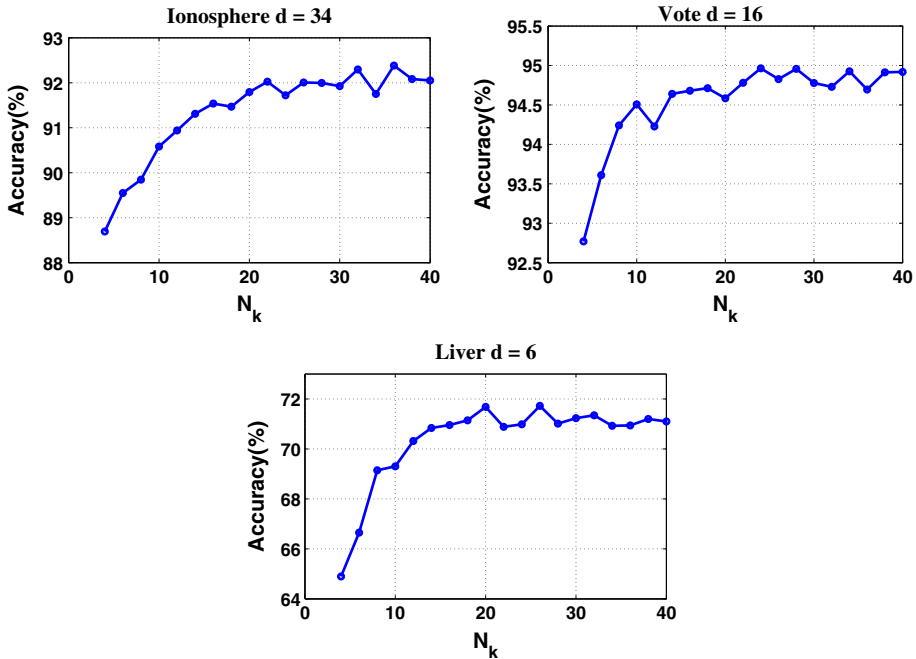


Fig. 1 Classification accuracy using R-MLMKC against different number of basis kernels N_k on three UCI datasets

λ_G . We consider four circumstances of different N_k settings: (1) constructing the matrix \mathbf{M} with equal small number of similar and dissimilar pairs, e.g., $N_k = 4, n_1 = n_2 = 2$; (2) constructing the matrix \mathbf{M} with equal larger number of similar and dissimilar pairs, e.g., $N_k = 8, n_1 = n_2 = 4$; (3) constructing the matrix \mathbf{M} with small number of similar pairs and larger number of dissimilar pairs, e.g., $N_k = 6, n_1 = 2, n_2 = 4$; (4) constructing the matrix \mathbf{M} with larger number of similar pairs and small number of dissimilar pairs, e.g., $N_k = 6, n_1 = 4, n_2 = 2$. Figure 2 shows the classification accuracy against λ_G on different UCI datasets, from left to right are the accuracy curve on Sonar, Heart, Wpbc and Vote datasets, respectively, when N_k using above settings. One can observe that, for all figures the classification accuracy declined after $\lambda_G > 0.02$, and the setting of N_k, n_1 and n_2 seems no effect on the best choice of λ_G , so we choose λ_G within the range of $[0, 0.02]$.

To further validate the effective settings of N_k, n_1 and n_2 , we fixed λ_G to some value within the range of $[0, 0.02]$. Using Wpbc dataset as an example, we get the classification accuracy on each pair of (n_1, n_2) , where $n_1, n_2 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and N_k is the sum of n_1 and n_2 . Figure 3 shows the classification accuracy versus n_1 and n_2 when $\lambda_G = 0.005$, we can find that the model can approach to the highest performance several times with different combination of n_1 and n_2 for different datasets. For example, as shown in Fig. 3b, c, the model can achieve the highest accuracy when $n_2 = 3$ and $n_1 = 2$. From Fig. 3c, we can observe that, when $n_1 = 2$ and $1 \leq n_2 \leq 5$ the model can achieve comparatively stable performance. The classification accuracy surface for other three different datasets are also illustrated in Fig. 4. From Figs. 3 and 4, we can see that generally, the model can achieve comparable performance by choosing small number of n_1 and n_2 . Therefore, we choose $n_1, n_2 \in \{2, 3\}$ to reduce the doublets used for k -means. Besides, it seems choosing significantly different

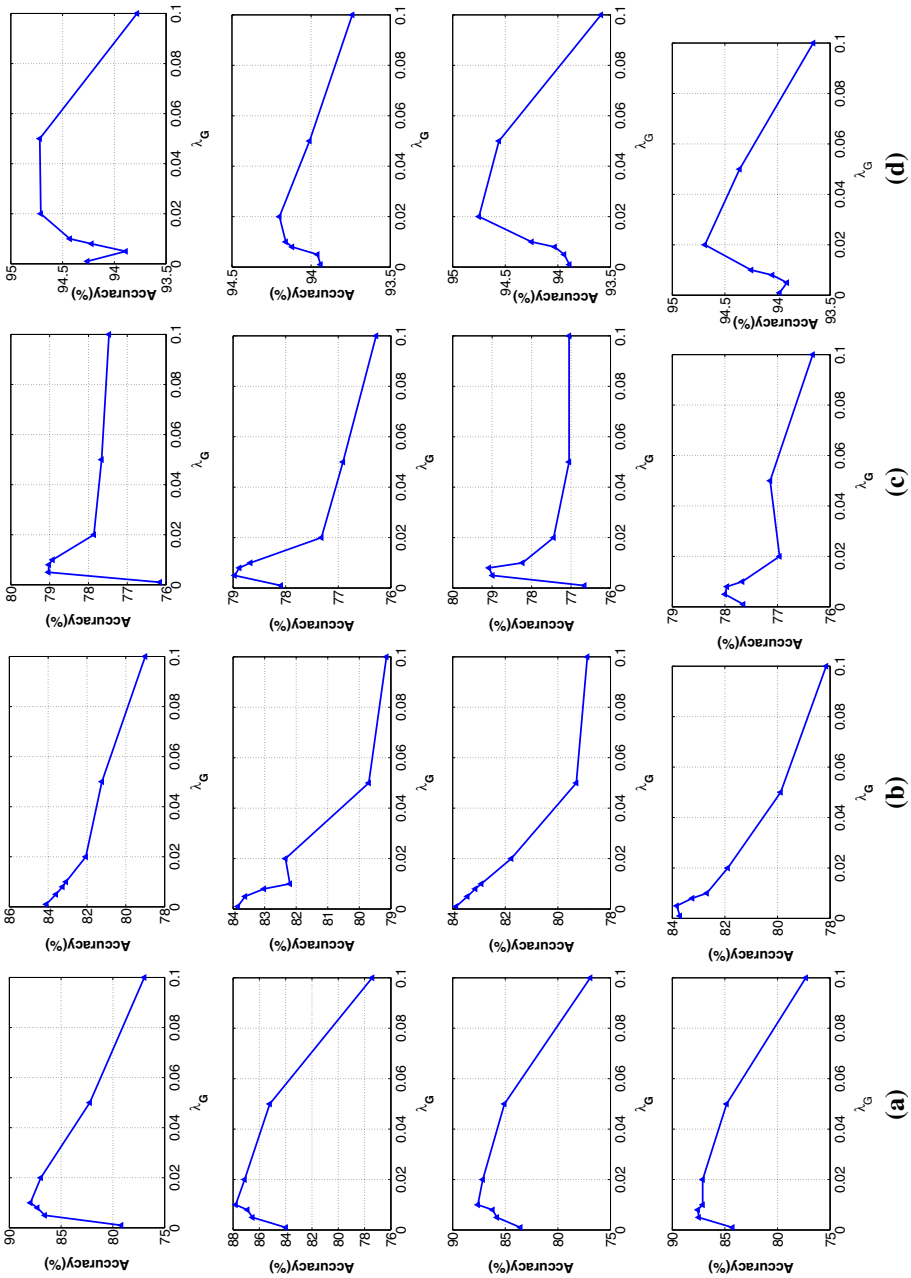


Fig. 2 Classification accuracy against λ_G under four circumstances of different N_k settings on different UCI datasets: **a** sonar, **b** heart, **c** Wpbc, and **d** vote

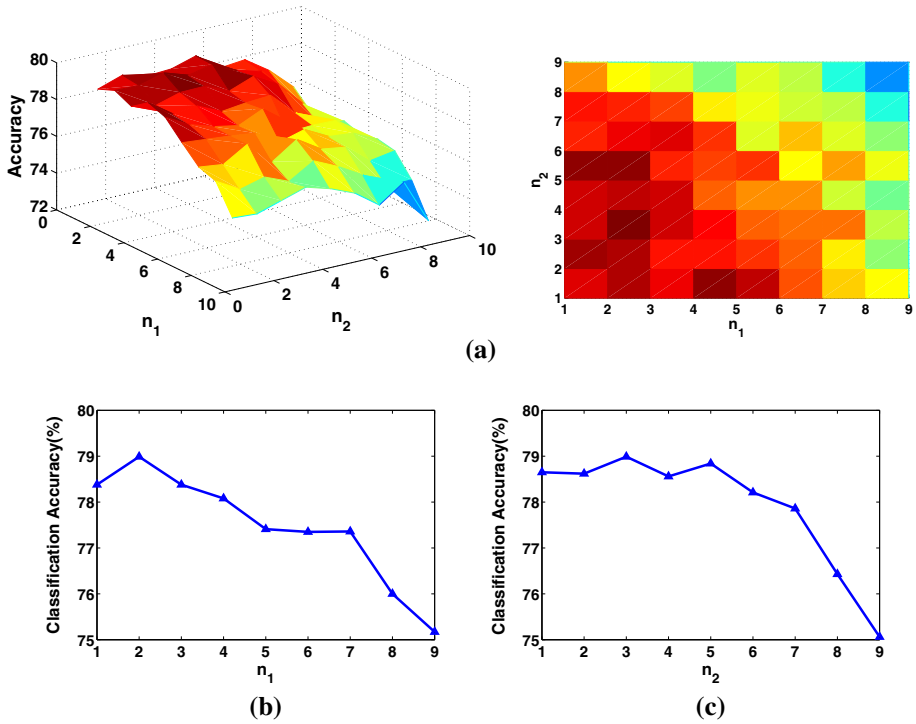


Fig. 3 Classification accuracy versus n_1 and n_2 . **a** Results on the Wpbc dataset with $\lambda_G = 0.005$, **b** classification accuracy versus n_1 for $n_2 = 3$, **c** classification accuracy versus n_2 for $n_1 = 2$

number of similar pairs and dissimilar pairs of samples is not proper when the number of N_k is settled, especially choosing more similar ones.

4.2 Result on the Synthetic Data Set

Two synthetic datasets with two classes whose distribution is nonlinear separable were created to evaluate our method. For the 2-dimensional dataset, feature x_1 and x_2 are drawn from a normal distribution of $\mathcal{N}(\mu_1, \mathbf{I})$ and $\mathcal{N}(\mu_2, \mathbf{I})$ with equal probability, where \mathbf{I} is an identity matrix. When $y = -1$, $\mu_1 = [-0.75, -3]^T$ and $\mu_2 = [0.75, 3]^T$. When $y = +1$, $\mu_1 = [3, -3]^T$ and $\mu_2 = [-3, 3]^T$. The distribution of x_1 and x_2 is shown in Fig. 5. This dataset contains 600 samples and for each sample \mathbf{x} , its label y has the equal probability of being $+1$ or -1 . We also generated a 50-dimensional dataset using the scikit-learn tool, which can introduce interdependence between features and add further noise to the data. It contains 1000 samples and each class is composed of 5 Gaussian clusters, where the 10 independent features, 10 useless features, 10 repeated features and noise are drawn from normal Gaussian distribution. We use m ($m = 10, 20, \dots, 90$) percent of samples to form the training set and the rest to form the test set. We evaluate our MLMKC with ℓ_1 regularization and radius information, namely MLMKC- ℓ_1 , on these nonlinear binary classification problem compared with SVM, GMKL, SimpleMKL, ℓ_p -MKL and SM1MKL as baseline methods. For the first dataset we use only ten basis kernels and for the second dataset we use forty basis kernels. The basis kernels used for MKL methods are Gaussian kernels with bandwidths selected from

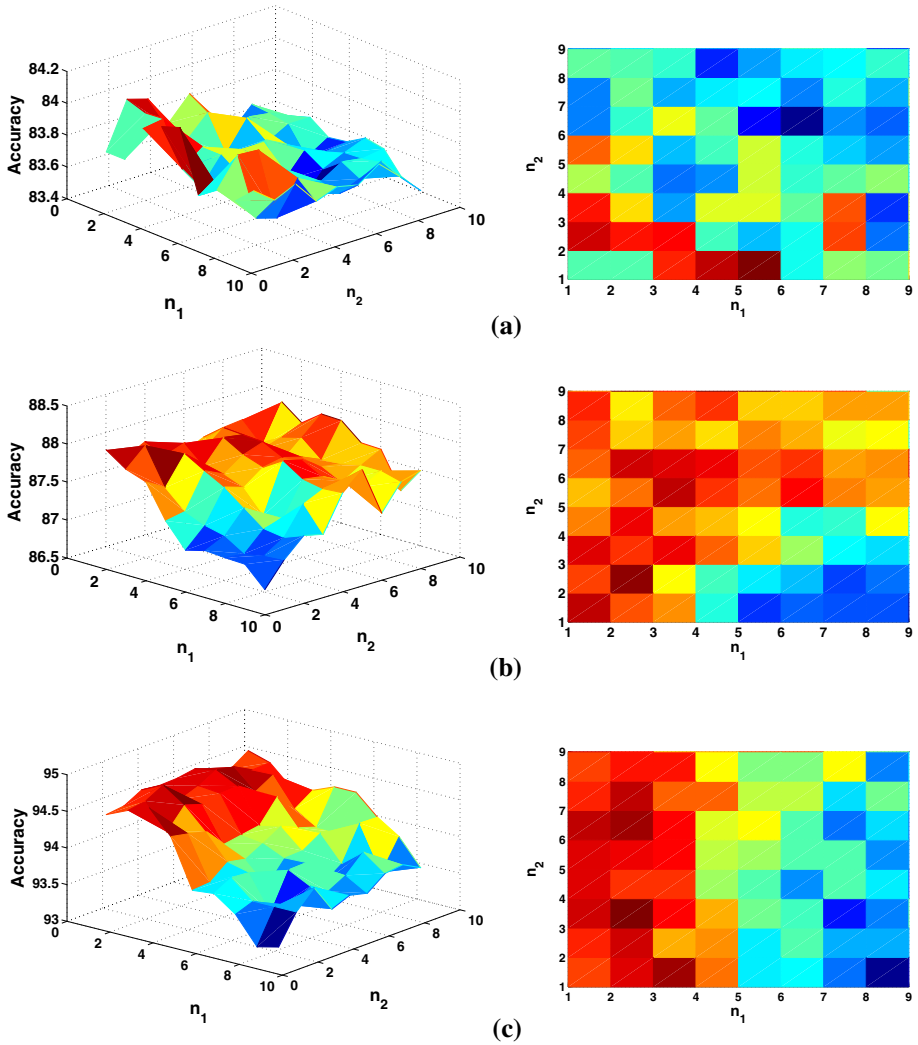
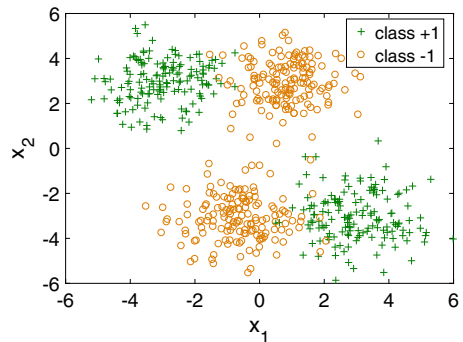


Fig. 4 Classification accuracy versus n_1 and n_2 with fixed λ_G : **a** results on the Heart dataset with $\lambda_G = 0.001$, **b** results on the Sonar dataset with $\lambda_G = 0.01$, and **c** results on the Vote dataset with $\lambda_G = 0.01$

Fig. 5 Two dimensional synthetic dataset



range $\{0.1, 0.2, 0.5, 0.7, 1.0, 2.0, 5.0, 7.0, 10.0, \dots, 20.0\}$ and polynomial kernels of degree 1 to 3. The parameter C used in SVM is chosen from a large range $\{10^{-1}, \dots, 10^3, 10^4\}$. Products of basis kernels are used as the combinations in GMKL framework. For ℓ_p -MKL and SM1MKL, the parameters p and θ are selected according to the setting in [24,46], respectively.

We show the decision boundaries of the first synthetic dataset, using 50% of the samples to train and others to test, for our method with different λ_G and N_k in Fig. 6, in which setting $\lambda_G = 0.1$ always result in over-fitting, while setting $\lambda_G = 0.01$ can achieve the highest classification accuracy. When choosing $N_k = 10$, the proposed MLMKC- ℓ_1 can achieve the highest accuracy with non-sparse and sparse solutions for kernels combination by setting $\lambda_G = 0.0$ and $\lambda_G = 0.01$, respectively. It is interesting that when $N_k = 4$, setting $\lambda_G = \{0.0, 0.001\}$ brings poor result, while $\lambda_G = 0.01$ provides accuracy of 97.67%, which demonstrates the significance of Gaussian RBF kernel. High classification accuracy no less than 97.33% can be achieved by setting $\lambda_G = 0.01$, but large N_k (e.g., $N_k = 16$ in Fig. 6) causes over-fitting. For the second synthetic dataset, choosing N_k within the range of $\{20, 40\}$ leads to higher accuracy, and setting larger N_k for all settings of λ_G makes no contribution to the classification accuracy. The accuracy curve with $\lambda_G = 0.02$ exceeds $\lambda_G = 0.0$ by nearly 10% as illustrated in Fig. 7, which demonstrates that the Gaussian RBF kernel cannot be replaced in more complex classification task. Above experimental results verify the effectiveness of our proposed parameterization scheme which incorporating the squared Mahalanobis distance into the Gaussian RBF kernels, and also reveal that by properly selecting parameter λ_G and N_k , the classification performance can benefit from the distance metric based kernels and the Gaussian RBF kernel in our proposed formulation.

We evaluate the proposed MLMKC algorithm using two synthetic datasets, and compare the proposed methods with other representative models, i.e., SVML,⁴ GMKL,⁵ SimpleMKL,⁶ ℓ_p -MKL,⁷ and SM1MKL⁸ in terms of classification accuracy. Experimental results are reported in Table 3, and the average number of kernels finally used in each method are reported in the last row. Comparatively, the proposed method achieves a higher accuracy compared with SVML and GMKL, especially on the first synthetic dataset. It demonstrates that utilizing the product combination framework of generalized MKL incorporated with distance metric makes our MLMKC flexible and effective for classification problem. Other MKL methods which focus on searching the optimal mixture of basis kernels, e.g., SimpleMKL, ℓ_p -MKL and SM1MKL get lower accuracy with the same number of input basis kernels, especially for the second synthetic dataset, which also verifies the effectiveness of the proposed kernel formulation and the robustness of our method. Sparse solution for kernels combination is obtained using SimpleMKL, GMKL, ℓ_p -MKL and our MLMKC, which we can see from the number of kernels finally used in Table 3. We find that the number of kernels finally used in our MLMKC is nearly the same as the number of features beneficial for the classification. The experimental results show that our method can get preferable results with less kernels which is more time and space efficient.

⁴ <http://www.cse.wustl.edu/~xuzx/research/code/code.html>.

⁵ <http://research.microsoft.com/en-us/um/people/manik/code/GMKL/download.html>.

⁶ <http://asi.insa-rouen.fr/enseignants/~arakoto/code/mklindex.html>.

⁷ http://doc.ml.tu-berlin.de/nonsparse_mkl/.

⁸ <https://sites.google.com/site/xinxingxu666/>.

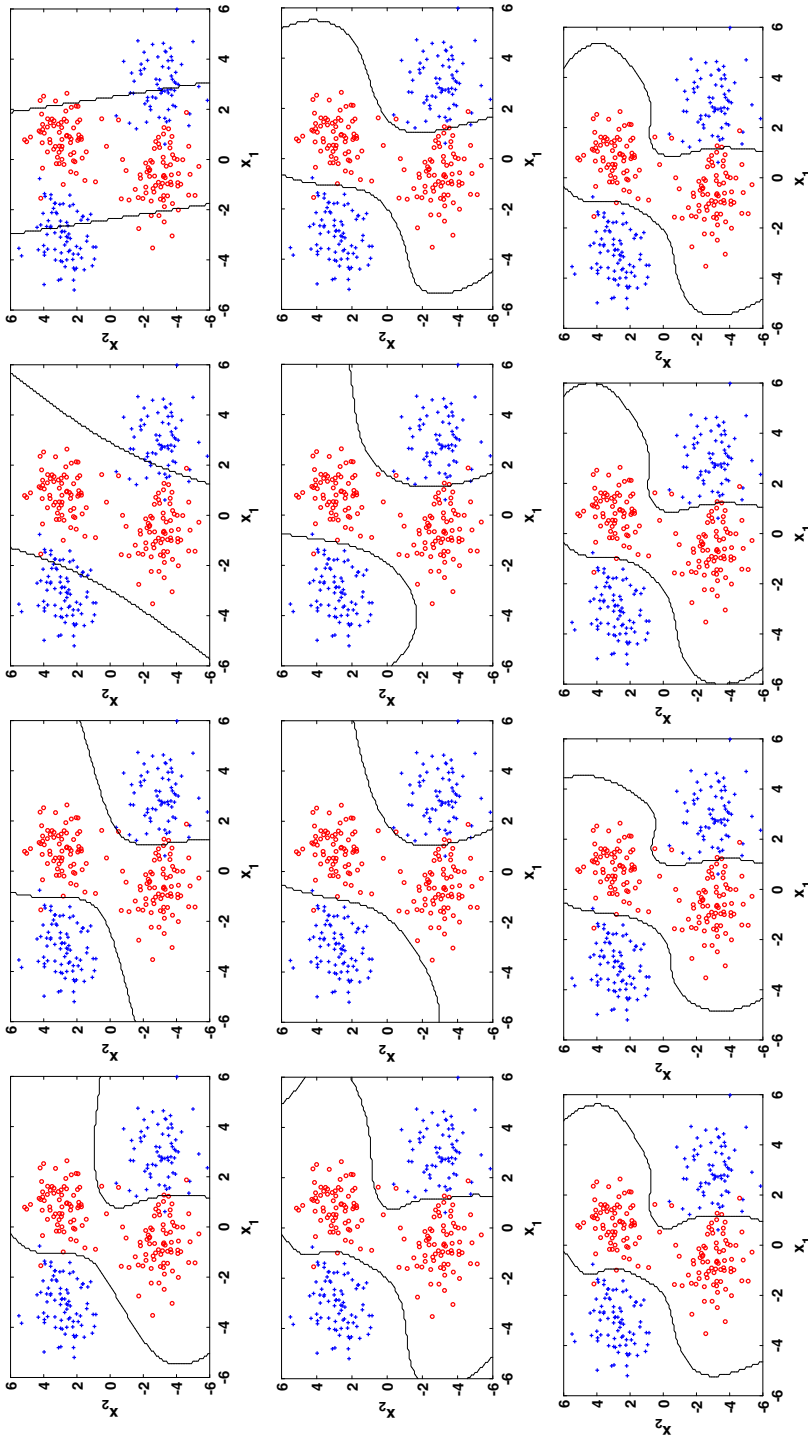


Fig. 6 Decision boundaries generated by our method with from top to bottom $N_k = 4, N_k = 10, N_k = 16$ and from left to right $\lambda_G = 0.1, \lambda_G = 0.01, \lambda_G = 0.001, \lambda_G = 0.0$

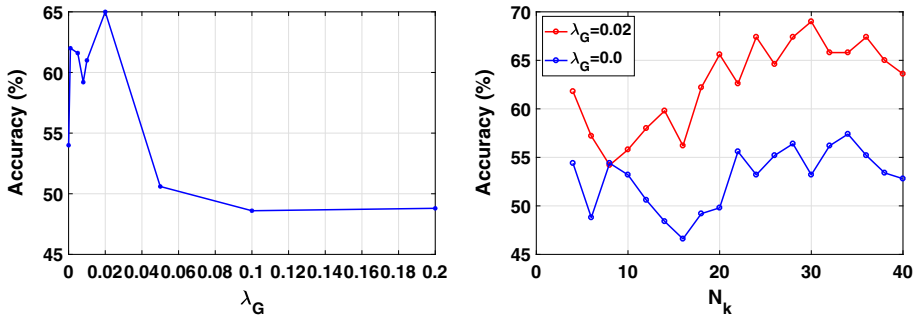


Fig. 7 Classification accuracy of the second synthetic dataset with different λ_G and N_k

Table 3 Classification accuracy (%) comparison of SVML, GMKL, SimpleMKL, ℓ_p -MKL, SM1MKL and our method on the synthetic datasets ($d = 2$) and ($d = 50$)

Method	SVML	GMKL	SimpleMKL	ℓ_p -MKL	SM1MKL	MLMKC- ℓ_1
d=2						
10	96.52	96.67	96.25	96.67	96.11	97.03
20	96.52	97.08	96.67	97.08	96.67	96.87
30	95.51	96.67	97.14	97.14	97.38	96.90
40	96.64	97.50	97.22	97.78	97.22	97.78
50	96.15	97.33	97.33	97.33	97.33	97.67
60	96.24	97.92	97.92	97.50	97.50	97.5
70	97.18	97.78	97.78	97.78	98.33	98.33
80	97.93	98.33	97.50	98.33	98.33	98.89
90	98.52	98.33	98.89	98.89	98.89	98.89
# of kernels	1	1	2	8	10	3
d=50						
10	55.04	57.60	51.33	50.45	50.31	56.63
20	59.92	54.39	52.17	52.95	51.07	58.85
30	59.97	61.49	50.50	50.22	50.70	62.10
40	62.00	59.58	52.77	53.03	50.26	61.57
50	66.18	65.20	52.50	53.90	50.90	66.70
60	65.60	66.00	52.90	51.80	49.70	66.10
70	67.80	67.0	52.90	51.70	49.20	67.30
80	66.80	67.10	53.20	52.50	50.40	68.20
90	68.40	70.45	53.80	53.30	50.60	71.80
# of kernels	1	3	8.5	13	39	10

Bold values indicate the best classification results over different methods

4.3 Comparison with State-of-the-Art Methods

In this experiment, we evaluate the proposed MLMKC algorithm using eleven UCI datasets, and compare the proposed methods with other representative models, i.e., SVML, GMKL, SimpleMKL, ℓ_p -MKL, SM1MKL, RMKL,⁹ EasyMKL⁹ and RM-GD⁹ in terms

⁹ <https://github.com/IvanoLauriola/MKLpy>.

Table 4 Classification accuracy (mean \pm SD) of different forms of MLMKC methods on the UCI datasets

Dataset	MLMKC			
	ℓ_1 w/ radius	ℓ_2 w/ radius	ℓ_1 w/o radius	ℓ_2 w/o radius
ACA	86.20 \pm 0.53	86.14 \pm 0.43	86.19 \pm 0.48	86.02 \pm 0.40
CMC	74.61 \pm 0.44	74.46 \pm 0.51	74.60 \pm 0.43	74.26 \pm 0.52
Mammo	83.95 \pm 0.37	83.86 \pm 0.30	83.89 \pm 0.30	83.82 \pm 0.30
Iono	94.61 \pm 0.50	94.52 \pm 0.54	94.66 \pm 0.56	94.54 \pm 0.57
Heart	84.10 \pm 0.45	83.79 \pm 0.54	84.01 \pm 0.54	83.57 \pm 0.59
Sonar	87.93 \pm 1.29	87.86 \pm 1.22	88.05 \pm 1.35	88.04 \pm 1.53
Liver	72.51 \pm 0.99	71.33 \pm 0.74	71.24 \pm 0.73	71.29 \pm 0.79
Wpbc	78.82 \pm 0.71	78.56 \pm 0.46	78.99 \pm 0.78	78.84 \pm 0.56
Pima	77.30 \pm 0.36	77.16 \pm 0.30	77.13 \pm 0.52	77.13 \pm 0.61
Vote	94.67 \pm 0.46	93.98 \pm 0.60	94.41 \pm 0.54	94.23 \pm 0.63
Musk	99.56 \pm 0.28	99.32 \pm 0.22	99.58 \pm 0.30	99.21 \pm 0.23

Bold values indicate the best classification results over different methods

of classification accuracy and training time. The designed RBF kernels are combined by taking their product for GMKL, and the parameters for ℓ_p -MKL and SM1MKL follow the setting in [24,46]. The parameter C used in SVM is chosen from a large range $\{10^{-3}, 10^{-2}, \dots, 10^6, 10^7\}$. Basis kernels used in SimpleMKL, ℓ_p -MKL and SM1MKL include Gaussian kernel and polynomial kernel following the construction method in SimpleMKL. The bandwidths of Gaussian kernel used in RMKL and RM-GD are selected as described in [18], where 20 scales are sampled with $\sigma_{min} = 0.1$ and $\sigma_{max} = 20$. For EasyMKL, we construct the RBF based weak kernel following [1]. Results are all obtained by using 5-folds CV and averaging over 20 runs. The classification results of four forms of proposed MLMKC methods are reported in Table 4, which includes MLMKC in Eq. (14) with ℓ_1 and ℓ_2 regularization and radius information, MLMKC in Eq. (10) with ℓ_1 and ℓ_2 regularization but without radius information. We can observe an interesting phenomenon that the proposed MLMKC method with ℓ_1 regularization but without radius information achieves good performance, which is almost the same as MLMKC with ℓ_1 regularization and radius information. Without loss of generality, we take MLMKC- ℓ_1 as the substitute of the proposed MLMKC in following experiments.

The comparison results with SVML, GMKL, SimpleMKL, SM1MKL, ℓ_p -MKL, RMKL, EasyMKL and RM-GD are listed in Table 5, where the highest accuracy of all are shown in solid black for each dataset, respectively. We do not report the accuracy of SVML on the Wpbc dataset, in that the released SVML code always collapsed when run on it. To compare the classification performance of these models, we list the average ranks of these models in the last row of Table 5. The average rank is defined as the mean rank of one method over the 11 datasets, which can provide a fair comparison of the algorithms [8].

In Table 5, our MLMKC- ℓ_1 greatly outperforms SVML and four MKL methods including GMKL, SimpleMKL, SM1MKL, and ℓ_p -MKL on most datasets with less than 10 kernels, while other methods need tens of basis kernels on small datasets and hundreds of basis kernels on large datasets. The classification accuracy of our method ranks third and is comparable with RMKL, EasyMKL and RM-GD on some datasets. These three methods solve different optimization problems instead of searching for the optimum combination directly of provided basis kernels. RMKL performs a max-variance projection-based learning, aiming at finding a

Table 5 Classification accuracy comparison (mean \pm SD) of different methods on the UCI datasets

Dataset	SVML	GMKL	SimpleMKL	ℓ_p -MKL	SMIMKL	RMKL	EasyMKL	RM-GD	MLMKC- ℓ_1
ACA	86.53 \pm 0.41	85.13 \pm 1.02	85.36 \pm 0.23	85.06 \pm 0.14	85.28 \pm 0.17	89.76 \pm 0.50	90.30 \pm 0.20	89.91 \pm 1.18	86.20 \pm 0.53
CMC	74.55 \pm 0.51	71.61 \pm 0.73	73.90 \pm 0.31	72.14 \pm 0.64	72.00 \pm 0.54	71.15 \pm 0.97	74.49 \pm 0.43	73.57 \pm 1.23	74.61 \pm 0.44
Mammo	83.23 \pm 0.32	78.25 \pm 0.58	82.65 \pm 0.19	82.63 \pm 0.42	82.64 \pm 0.35	83.60 \pm 0.82	83.67 \pm 0.62	83.76 \pm 0.42	83.95 \pm 0.37
Iono	94.01 \pm 0.46	93.19 \pm 0.92	92.59 \pm 1.34	90.81 \pm 0.80	91.06 \pm 0.59	98.95 \pm 0.13	98.87 \pm 0.16	98.47 \pm 0.17	94.61 \pm 0.50
Heart	82.56 \pm 0.54	79.43 \pm 0.96	84.21 \pm 0.78	83.40 \pm 0.64	83.69 \pm 0.55	89.20 \pm 0.56	86.86 \pm 0.93	86.64 \pm 0.40	84.10 \pm 0.45
Sonar	81.74 \pm 1.63	82.53 \pm 0.74	83.65 \pm 1.26	83.73 \pm 1.58	83.86 \pm 1.82	94.82 \pm 0.55	94.63 \pm 0.84	94.78 \pm 0.44	87.93 \pm 0.36
Liver	71.30 \pm 1.35	72.83 \pm 1.41	68.12 \pm 2.46	71.32 \pm 1.63	71.71 \pm 1.49	72.70 \pm 1.33	72.51 \pm 1.86	72.94 \pm 1.03	72.51 \pm 0.99
Wpbc	N/A	78.98 \pm 2.09	77.79 \pm 1.95	78.55 \pm 2.17	78.59 \pm 1.36	75.09 \pm 1.34	73.04 \pm 1.51	71.25 \pm 1.19	78.82 \pm 0.71
Pima	76.44 \pm 0.51	77.30 \pm 1.20	76.48 \pm 1.16	77.16 \pm 0.63	77.07 \pm 0.51	80.43 \pm 0.81	79.61 \pm 0.81	81.46 \pm 0.44	77.30 \pm 0.36
Vote	94.69 \pm 0.31	94.13 \pm 0.83	95.01 \pm 0.32	94.02 \pm 0.56	93.90 \pm 0.55	98.74 \pm 0.17	98.63 \pm 0.15	98.83 \pm 2.66	94.67 \pm 0.46
Musk	98.07 \pm 0.68	91.46 \pm 0.24	84.59 \pm 0.06	93.79 \pm 0.12	93.53 \pm 0.10	99.61 \pm 0.03	99.57 \pm 0.36	99.59 \pm 0.37	99.56 \pm 0.28
Rank	5.54	6.30	6.27	6.91	6.54	3.00	3.27	2.73	3.27

Bold values indicate the best classification results over different methods

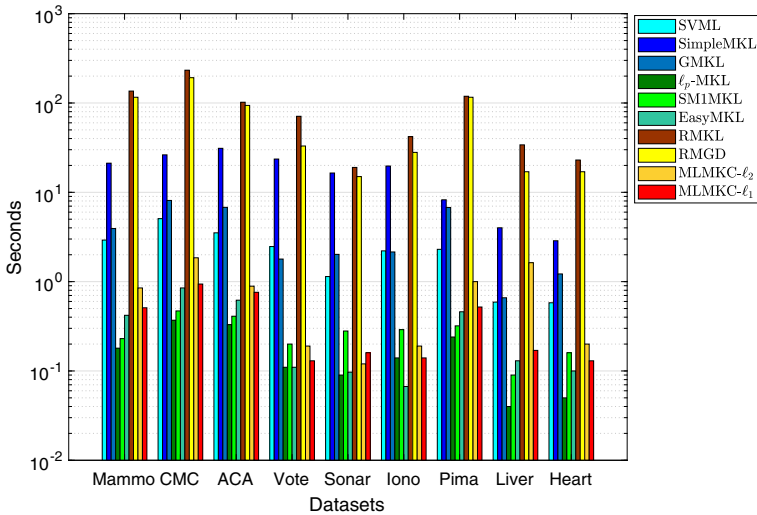


Fig. 8 Training time (s) of SVML, SimpleMKL, GMKL, ℓ_p -MKL, SM1MKL, EasyMKL, RMKL, RM-GD, MLMKC- ℓ_2 , and MLMKC- ℓ_1

low-dimensional representation which approximates the original space spanned by the basis kernels. It has capability of removing the redundancy of interscale kernel similarities, hence it is flexible with the scale of basis kernels. EasyMKL optimizes a min-max problem over probability distribution of positive and negative samples sets and the combination parameters of predefined kernel matrices, focusing on maximizing the margin. It also applies a feature selection in kernel computation, making it robust to noise features. RM-GD, a state-of-the-art margin-based MKL method, exploits a radius-margin ratio optimization based on gradient descent. It demonstrates that the minimization of the radius-margin bound achieves better results with respect to the margin maximization, while MLMKC minimizes the radius and maximizes the margin separately.

The experimental results reveal that: (i) the formulation of incorporating Mahalanobis into the Gaussian RBF kernel to make up the distance metric, and jointly learning the distance metric and the corresponding kernel classifier via GMKL framework is of great efficiency for SVM-based classifiers. (ii) The optimization result without considering the radius information achieves good performance indicate that the regularizer $r(\beta)$ related with β in our kernel function also imposes constraint on the radius of training samples in the feature space. Regardless of the classification results, these MKL methods need a number of appropriate basis kernels whether seeking the optimum combination parameters directly or not, comparing with our method. SimpleMKL, SM1MKL and ℓ_p -MKL methods need a large number of basis kernels, which often obtain sparse solution for kernels combination.

The training time of representative methods, and our proposed methods on several small datasets is showed in Fig. 8. All the experiments are executed in the same PC. The training time for MLMKC includes the doublets construction, k -means clustering and kernel computation. In general, we found the training time required for MLMKC- ℓ_1 outperforming SVML, GMKL and SimpleMKL, and 100 times faster than RM-GD and RMKL though achieving better classification accuracy than other MKL methods. ℓ_p -MKL and SM1MKL methods operate optimization in simple analytical update rules via block coordinate descent algorithm instead. The chunking optimization reduces the iteration and avoids the time consuming procedure for kernel combination coefficients in terms of the dual variable. EasyMKL replaces

Table 6 Classification error rates (%) comparison of different methods on the real-world datasets

Method	COIL20	USPS
SVML	2.66	5.79
SimpleMKL	3.09	11.16
GMKL	1.43	–
ℓ_p -MKL	1.78	10.00
SM1MKL	1.15	10.00
RMKL	1.10	9.07
EasyMKL	1.67	9.57
RM-GD	1.33	9.57
MLMKC– ℓ_2	2.67	10.65
MLMKC– ℓ_1	0.67	4.38

Bold values indicate the best classification results over different methods

the kernel matrix with the sum of the predefined weak kernel matrices, making it efficient in terms of time and space. By adopting SPG optimizer, our method performs efficiently and is even comparable to EasyMKL, ℓ_p -MKL and SM1MKL methods.

The classification error rates of multiclass real-world datasets are listed in Table 6. We use small number (about half of the feature dimension) of base kernels with SimpleMKL, ℓ_p -MKL and SM1MKL methods to avoid memory exceptions for the real-world datasets. We do not report the result of GMKL on USPS because it require memory space that our PC can not provide with. This further validates that on the large scale dataset our proposed method is competitive with other methods with less kernels. From Table 6, we can see that our proposed MLMKC achieves the lowest error rate with small number of kernels, $N_k < 10$, which verify the effectiveness of the method on the large scale dataset. As demonstrated by the classification results and the timing results on UCI datasets and real-world datasets, the proposed method produces overall better performance.

5 Conclusion

In this paper, we investigate metric learning via multiple kernel learning for SVMs. An effective MKL framework for joint learning of distance metric and kernel classifier is proposed, referred to MLMKC. We reformulate the matrix of Mahalanobis distance metric, and then incorporate it into the Gaussian RBF kernel to construct a novel kernel function formulation for the subsequent embedding to the GMKL framework. The regularization on the radius information, which shows significant improvement on MKL scenarios, although has been considered during the formulation of the proposed algorithm, shows limited ability on improving the final classification performance. Experimental results demonstrate that, dual regularizations, coming from both the regularization of distance metric imposed on the distance of samples in the original space, and regularizer $r(\beta)$ imposed on the radius of training samples in the feature space, guarantee the reliable performance of MLMKC. On the UCI datasets we demonstrate that our algorithm achieves preferable results in terms of classification accuracy and comparable training time with state-of-the-art methods. On the handwritten digit dataset USPS and object classification dataset COIL20, we achieve competitive results and show effectiveness of our proposed method on large scale data with less

number of kernels. All these aspects make MLMKC a competitive general-purpose metric learning-based multiple kernels algorithm for SVMs with Gaussian RBF kernels. The future work is to introduce efficient approximation algorithms to our proposed MLMKC model with suitable formulation, and finally make it feasible to tackle the scalability issue.

Acknowledgements This work is partly support by the National Science Foundation of China (NSFC) Project under the Contract Nos. 61671182, 61102037, 61471146 and 61871381.

References

1. Aiolli F, Donini M (2015) EasyMkL: a scalable multiple kernel learning algorithm. *Neurocomputing* 169:215–224
2. Bach FR (2009) Exploring large feature spaces with hierarchical multiple kernel learning. In: *Advances in neural information processing systems*, pp 105–112
3. 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
4. Boiman O, Shechtman E, Irani M (2008) In defense of nearest-neighbor based image classification. In: *IEEE Conference on computer vision and pattern recognition, 2008. CVPR 2008*, pp 1–8
5. Cao Q, Ying Y, Li P (2013) Similarity metric learning for face recognition. In: *IEEE international conference on computer vision*, pp 2408–2415
6. Cortes C, Mohri M, Rostamizadeh A (2009) Learning non-linear combinations of kernels. In: *Advances in neural information processing systems*, pp 396–404
7. Davis JV, Kulis B, Jain P, Sra S, Dhillon IS (2007) Information-theoretic metric learning. In: *Machine learning, proceedings of the twenty-fourth international conference*, pp 209–216
8. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7(1):1–30
9. Deng C, Tang X, Yan J, Liu W, Gao X (2016) Discriminative dictionary learning with common label alignment for cross-modal retrieval. *IEEE Trans Multimed* 18(2):208–218
10. Deng C, Chen Z, Liu X, Gao X, Tao D (2018) Triplet-based deep hashing network for cross-modal retrieval. *IEEE Trans Image Process* 27(8):3893–3903
11. Do H, Kalousis A (2013) Convex formulations of radius-margin based support vector machines. In: *Proceedings of the 30th international conference on Machine learning*, pp 169–177
12. Do H, Kalousis A, Wang J, Woznica A (2012) A metric learning perspective of SVM: on the relation of SVM and LMNN. *Eprint Arxiv* pp 308–317
13. Dong Y, Du B, Zhang L, Zhang L, Tao D (2017) Lam3l: locally adaptive maximum margin metric learning for visual data classification. *Neurocomputing* 235:1–9
14. Frank A, Asuncion A (2010) Uci machine learning repository [<http://archive.ics.uci.edu/ml>]. irvine, ca: University of california. School of Information and Computer Science 213
15. Gai K, Chen G, Zhang C (2010) Learning kernels with radiuses of minimum enclosing balls. In: *Advances in neural information processing systems*, pp 649–657
16. Gao X, Hoi SCH, Zhang Y, Wan J, Li J (2014) Soml: sparse online metric learning with application to image retrieval. In: *Twenty-eighth AAAI conference on artificial intelligence*, pp 1206–1212
17. Goldberger J, Roweis ST, Hinton GE, Salakhutdinov R (2004) Neighbourhood components analysis. *Adv Neural Inf Process Syst* 83(6):513–520
18. Gu Y, Wang C, You D, Zhang Y, Wang S, Zhang Y (2012) Representative multiple kernel learning for classification in hyperspectral imagery. *Neurocomputing* 50:215–224
19. Guillaumin M, Verbeek J, Schmid C (2009) Is that you? Metric learning approaches for face identification. In: *2009 IEEE 12th international conference on computer vision*. IEEE, pp 498–505
20. Hasan MA, Ahmad S, Molla MK (2017) Protein subcellular localization prediction using multiple kernel learning based support vector machine. *Mol Biosyst* 13(4):785
21. Hoi SCH, Liu W, Lyu MR, Ma WY (2006) Learning distance metrics with contextual constraints for image retrieval. In: *IEEE conference on computer vision and pattern recognition*, pp 2072–2078
22. Jain A, Vishwanathan SVN, Varma M (2012) Spg-gmkl: generalized multiple kernel learning with a million kernels. In: *ACM SIGKDD international conference on knowledge discovery and data mining*, pp 750–758

23. Kedem D, Tyree S, Weinberger KQ, Sha F, Lanckriet G (2012) Non-linear metric learning. In: Advances in neural information processing systems, pp 2573–2581
24. Kloft M, Brefeld U, Sonnenburg S, Zien A (2011) L_p-norm multiple kernel learning. *J Mach Learn Res* 12:953–997
25. Lanckriet GRG, Cristianini N, Bartlett P, El Ghaoui L, Jordan MI (2002) Learning the kernel matrix with semi-definite programming. *J Mach Learn Res* 5(1):323–330
26. Lauriola I, Polato M, Aiolli F (2017) Radius-margin ratio optimization for dot-product Boolean kernel learning. In: International conference on artificial neural networks, pp 183–191
27. Lim DKH, Mcfee B, Lanckriet G (2013) Robust structural metric learning. In: International conference on machine learning, pp 615–623
28. Lu X, Wang Y, Zhou X, Ling Z (2015) A method for metric learning with multiple-kernel embedding. *Neural Process Lett* 43(3):923–924
29. Mcfee B, Lanckriet G (2011) Learning multi-modal similarity. *J Mach Learn Res* 12(8):491–523
30. Nguyen B, Morell C, De Baets B (2016) Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing* 214:805–814
31. Nguyen N, Guo Y (2008) Metric learning: a support vector approach. In: Joint European conference on machine learning and knowledge discovery in databases, Springer, pp 125–136
32. Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y (2008) Simplemkl. *J Mach Learn Res* 9(11):2491–2521
33. Schölkopf B, Smola A (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
34. Shawe-Taylor J, Cristianini N (2006) Kernel methods for pattern analysis. *J Am Stat Assoc* 101(12):1730–1730
35. Sonnenburg S, Rätsch G, Schäfer C, Schölkopf B (2006) Large scale multiple kernel learning. *J Mach Learn Res* 7(7):1531–1565
36. Squarcina L, Castellani U, Bellani M, Perlini C, Lasalvia A, Dusi N, Bonetto C, Cristofalo D, Tosato S, Rambaldelli G (2017) Classification of first-episode psychosis in a large cohort of patients using support vector machine and multiple kernel learning techniques. *Neuroimage* 145:238–245
37. Torresani L, Kc L (2007) Large margin component analysis. *Adv Neural Inf Process Syst* 19:1385
38. Tran D, Sorokin A (2008) Human activity recognition with metric learning. In: European conference on computer vision, pp 548–561
39. Vapnik V, Chapelle O (2000) Bounds on error expectation for support vector machines. *Neural Comput* 12(9):2013–2036
40. Varma M, Babu BR (2009) More generality in efficient multiple kernel learning. In: International conference on machine learning, pp 1065–1072
41. Wang F, Zuo W, Zhang L, Meng D, Zhang D (2015) A kernel classification framework for metric learning. *IEEE Trans Neural Netw Learn Syst* 26(9):1950–1962
42. Wang J, Do HT, Woznica A, Kalousis A (2011) Metric learning with multiple kernels. In: Advances in neural information processing systems, pp 1170–1178
43. Wang J, Deng Z, Choi KS, Jiang Y, Luo X, Chung FL, Wang S (2016) Distance metric learning for soft subspace clustering in composite kernel space. *Pattern Recognit* 52:113–134
44. Weinberger KQ, Saul LK (2006) Distance metric learning for large margin nearest neighbor classification. *J Mach Learn Res* 10(1):207–244
45. Wu H, He L (2015) Combining visual and textual features for medical image modality classification with lp-norm multiple kernel learning. *Neurocomputing* 147(1):387–394
46. Xu X, Tsang IW, Xu D (2013) Soft margin multiple kernel learning. *IEEE Trans Neural Netw Learn Syst* 24(5):749–761
47. Xu Z, Jin R, King I, Lyu M (2009) An extended level method for efficient multiple kernel learning. In: Advances in neural information processing systems, pp 1825–1832
48. Xu Z, Weinberger KQ, Chapelle O (2012) Distance metric learning for kernel machines. *arXiv preprint arXiv:1208.3422*
49. Yang E, Deng C, Li C, Liu W, Li J, Tao D (2018) Shared predictive cross-modal deep quantization. *IEEE Trans Neural Netw Learn Syst* 99:1–12
50. Yi S, Jiang N, Wang X, Liu W (2016) Individual adaptive metric learning for visual tracking. *Neurocomputing* 191:273–285
51. Ying Y, Li P (2012) Distance metric learning with eigenvalue optimization. *J Mach Learn Res* 13(1):1–26

52. Zhang X, Mahoor MH, Mavadati SM (2015) Facial expression recognition using l_p -norm MKL multiclass-SVM. *Mach Vis Appl* 26(4):467–483
53. Zhao C, Chen Y, Wei Z, Miao D, Gu X (2018) Qrkiss: a two-stage metric learning via QR-decomposition and kiss for person re-identification. *Neural Process Lett* 2:1–24

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.