International Journal of Innovative
Computing, Information and Control
Volume **12**, Number **6**, December **2016**

ICIC International ©2016 ISSN 1349-4198
pp. 1835–1849

GROUP BASED LOCALIZED MULTIPLE KERNEL LEARNING ALGORITHM WITH *lp*-NORM

Guangyuan Fu, Qingchao Wang, Dongying Bai and Linlin Li

Department of Information Engineering
Xi'an Research Institute of Hi-Tech
Hongqing Town, Xi'an 710025, P. R. China
wangqingchaoa@yeah.net

Received May 2016; revised September 2016

Abstract. *Localized multiple kernel learning is a promising strategy for combining multiple features or kernels in terms of their discriminative power. However, learning specific combination kernel for each sample generally leads to expensive computation and less reliability caused by some dominant samples. In addition, traditional sparse constraint generally causes that one of the best kernels is selected, while some useful kernels may have not been used efficiently. In this paper, we proposed a groupbased non-sparse localized multiple kernel learning algorithm to tackle the issues above. In our algorithm, the samples are divided into groups and then the kernel weights are optimized for each group. Because the sparse constraint may lose useful kernels, we use an lp-norm constraint on the kernels and obtain non-sparse results to avoid losing useful kernels. The advantage of each kernel is adopted in various local spaces. Since some datasets consist of multiple features, we propose a multiple kernel clustering method to make the clustering result more reliable. Experiments state that our method performs better than other excellent MKL algorithms.*

Keywords: Multiple kernel learning, Support vector machine, *lp*-norm, Local learning

1. **Introduction.** Kernel method is an efficient tool for solving learning problems like classification and regression. The advantage of kernel method is realized by people due to the success of support vector machine (SVM) [1]. The key idea of kernel method is to introduce nonlinearity to represent the similarity of samples. However, the performance of kernel machine strongly depends on the selection of kernel function. More importantly, the kernel machine with single kernel performs not well while the samples consist of multiple sources. In such case, an interesting approach is to combine a series of kernels instead of a single one.

Learning both classifier and kernels simultaneously in a single optimization problem is known as multiple kernel learning (MKL) [2]. Learning the optimal combination of kernels could adopt the advantage of each kernel, so it generally achieves better performance than the single one. The main problem of MKL is how to learn the optimal combination of kernels. Then a lot of MKL algorithms are proposed to solve this problem [3-7]. These methods have enhanced the interpretability of the decision function and improved the performance of kernel method. The MKL algorithm is also widely used in clustering [8], dimension reduction [9], semi-supervised learning problems [10] and so on.

In recent years, the research of MKL mainly focuses on two strategies. On the one hand, a more efficient method is proposed to accommodate larger scale problems [12-14]. On the other hand, a more complex combination method is used to improve classification accuracy as far as possible [11,15,16]. Given N training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ and M kernels $K_m(\cdot, \cdot)$, the combination kernel can be rewritten as $K(x_i, x_j) = \sum_m \beta_m K_m(x_i,$

x_j). β_m is the weights of kernel $K_m(\cdot,\cdot)$. For SVM-based framework, the problem of learning a linear combination generally has the form as follows:

$$\max_{\boldsymbol{\beta}} \min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left(\sum_{m=1}^{M} \beta_m K_m(x_i, x_j) \right) - \sum_{i=1}^{N} \alpha_i \tag{1}$$

where alternation optimization is usually used to learn the support vector coefficient $\boldsymbol{\alpha}$ and kernel weights $\boldsymbol{\beta}$.

Nevertheless, canonical MKL basically adopts a uniform combination for the whole input space. Due to the diversity of samples, the optimal kernel weights may vary from different input spaces. This motivates us using different combination kernels in different local spaces. Under the idea of local learning, some promising results have been achieved [17-23].

Generally, by rewriting the combination of base kernel into a sample-based one, the combination kernel can be defined as $K(x_i, x_j) = \sum_m \beta_m^{x_i} \beta_m^{x_j} K_m(x_i, x_j)$. The kernel weight of $\beta_m^{x_i}$ is specific for each sample.

The final solution of SVM based on the LMKL is to solve the dual problem as follows:

$$\max_{\boldsymbol{\beta}} \min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left(\sum_{m=1}^{M} \beta_m^{x_i} \beta_m^{x_j} K_m(x_i, x_j) \right) - \sum_{i=1}^{N} \alpha_i \tag{2}$$

The solving of $\boldsymbol{\alpha}$ is actually a canonical SVM problem. The essential problem lies in the solving of $\boldsymbol{\beta}$ which is a difficult quadratic non-convex problem. We know that solving the quadratic programming problem needs expensive computation. Gonen and Alpaydin [17] defined a gating model to represent $\boldsymbol{\beta}$, and the updating of $\boldsymbol{\beta}$ was replaced by updating the parameters of the gating model, which can be obtained by gradient descent. This method is called localized multiple kernel learning (LMKL). LMKL is a great progress, although it achieves statistically similar accuracy results compared with canonical MKL. It provides an efficient solution for the difficult quadratic non-convex problem. Inspired by it, more efficient localized MKL algorithms were proposed [18-21].

In practice, such sample-based MKL methods have enhanced the discriminative power of classifiers. However, the expensive computation is incurred to learning kernel weights for each sample. More importantly, heavily respecting individual samples may overwhelm the intrinsic properties of a category so as to reduce the reliability of classifier [22]. In this case, Yang [22] proposed a group-sensitive multiple kernel learning (GS-MKL). The samples were divided into groups and then learnt the specific kernel weights for each group referring to LMKL [17]. However, the GS-MKL method uses a sparse constraint for the kernels, and thus some useful kernels may have not been used effectively. At the same time, Marius et al. [11] researched the non-sparse constraint on kernels and found it could generally achieve better performance than sparse methods. Then non-sparse methods became a new focus in the field of MKL.

In this paper, to inherit the advantage of group based MKL and non-sparse method, we propose a novel group based localized MKL algorithm with *lp*-norm constraint (*lp*-GLMKL). The main contributions of this paper consist of two points. One is that a non-sparse group based localized MKL framework is proposed. The other one is that we design a multi-kernel k-means algorithm. Thanks to the fact that the *lp*-norm constraint is employed, we can obtain a non-sparse local combination of kernels. The performance of classifier is improved by adjusting the sparsity of kernels. Because the samples are represented by multiple kernels/features, the multi-kernel clustering algorithm is more suitable for our *lp*-GLMKL framework. The flowchart of training the classifier is summarized in Figure 1.

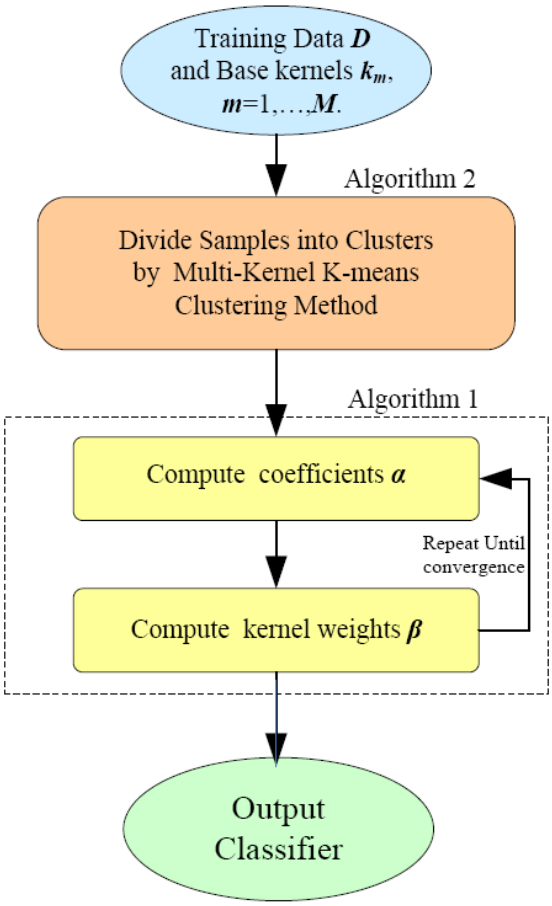

FIGURE 1. Flowchart of training the classifier

We carry out extensive experiment to compare the performance to some excellent MKL algorithms. The experiment on synthetic dataset indicates that our method achieves better decision boundary than the baseline approaches. Then we evaluate the performance of our method and test the influence of the parameters on eight UCI datasets. Finally, we apply our method to computer vision dataset and our method achieves higher classification accuracy than other excellent MKL algorithms.

The rest of this paper is organized as follows. In Section 2, the *lp*-GLMKL framework and optimization procedure for binary classification are introduced. The multi-kernel K-means clustering algorithm is presented in Section 3. Section 4 presents the experiments and analyses. Finally, Section 5 concludes this paper.

2. **Learning *lp*-GLMKL Based Classifier.** We take binary classification as an example to state the framework of *lp*-GLMKL method. Assume that we have N training samples, and the training dataset is represented by $D_L = \{x_i, y_i\}_{i=1}^N$, where x represents the N training samples, and $y \in \{-1, +1\}$ is the labels of the training samples. Noted that x_i could consist of multiple features which are represented by m kernels, or x_i is just one vector represented by m kernels. The theory is all the same. The discriminant function is defined as:

$$f(x) = \sum_{m=1}^{M} \beta_m^{c(x)} \langle \boldsymbol{w}_m, \phi_m(x) \rangle + b \tag{3}$$

where $\beta_m^{c(x)}$ represents the weight of the mth kernel in the group $c(x)$. By modifying the original SVM formulation with this new discriminant function, the training procedure can

be implemented by solving the following optimization problem:

$$\min_{\boldsymbol{w}_m,b,\xi,\beta} \frac{1}{2}\sum_{m}^{M}\|\boldsymbol{w}_m\|^2 + C\sum_{i=1}^{N}\xi_i \tag{4}$$

$$\begin{aligned} \text{s.t.} \quad & y_i\left(\sum_{m=1}^{M}\beta_m^{c(x_i)}\langle \boldsymbol{w}_m, \phi_m(x_i)\rangle + b\right) \geq 1-\xi_i \quad \forall i, \\ & \xi_i \geq 0 \quad \forall i, \\ & \textstyle\sum_m \left(\beta_m^{c(x_i)}\right)^p = 1 \quad \beta_m^{c(x_i)} \geq 0 \ \ \forall i, m \end{aligned}$$

where C is the regularization parameter and ξ is the slack variable. Inspired by SVM [1], Lagrange multiplier method is used to solve the dual problem of Equation (4). We first fix the kernel weights $\boldsymbol{\beta}$ and minimize the problem (4). The Lagrange object function is formulated as follows:

$$\begin{aligned} L = & \frac{1}{2}\sum_{m}^{M}\|\boldsymbol{w}_m\|^2 + \sum_{i=1}^{N}(C-\alpha_i-\gamma_i)\xi_i \\ & + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i\left(\sum_{m=1}^{M}\beta_m^{c(x)}\langle \boldsymbol{w}_m, \phi_m(x)\rangle + b\right) \end{aligned} \tag{5}$$

where α_i and γ_i are the Lagrange multipliers.

Taking the derivatives of L with respect to the primal variables, we get the following:

$$\frac{\partial L}{\partial \boldsymbol{w}_m} = 0 \Rightarrow \boldsymbol{w}_m = \sum_i \beta_m^{c(x_i)}\phi_m(x_i)\alpha_i y_i \tag{6}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0 \tag{7}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \beta_i \tag{8}$$

Eliminating the $\boldsymbol{w}$, ξ and b using Equations (6)-(8), we obtain the dual formulation of problem (4) as follows:

$$\begin{aligned} & \min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\sum_{m=1}^{M}\beta_m^{c(x_i)}\beta_m^{c(x_j)}K_m(x_i,x_j)\right) - \sum_{i=1}^{N}\alpha_i \\ & \text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i = 0, \quad \alpha_i \leq C \quad \forall i \end{aligned} \tag{9}$$

and the discriminant function of Equation (3) becomes:

$$f(x) = sign\left(\sum_i \alpha_i y_i K(x_i,x_j) + b\right) \tag{10}$$

where

$$K(x_i,x_j) = \sum_m \beta_m^{c(x_i)}\beta_m^{c(x_j)}K_m(x_i,x_j) \tag{11}$$

While we also should maximize the object function of Equation (9) over kernel weights $\boldsymbol{\beta}$, the optimization problem is finally a max-min problem as follows:

$$\begin{aligned} &\max_{\boldsymbol{\beta}} \min_{\boldsymbol{\alpha}} J = \tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left(\sum_{m=1}^{M}\beta_m^{c(x_i)}\beta_m^{c(x_j)}K_m(x_i,x_j)\right) - \sum_{i=1}^{N}\alpha_i \\ \text{s.t.}\quad &\sum_{i=1}^{N}\alpha_i y_i = 0, \quad 0 \le \alpha_i \le C \quad \forall i \\ &\textstyle\sum_m \left(\beta_m^{c(x_i)}\right)^p = 1 \quad \beta_m^{c(x_i)} \ge 0 \quad \forall i, m \end{aligned} \tag{12}$$

J is a multi-object function of coefficient $\boldsymbol{\alpha}$ and local kernel weights $\boldsymbol{\beta}$. When $\boldsymbol{\beta}$ is fixed, minimizing J over the coefficient is meant to minimize the global classification error and maximize the margin. When $\boldsymbol{\alpha}$ is fixed, maximizing J over the local kernel weights is meant to maximize the intra-class similarity and minimize the inter-class similarity simultaneously.

Similar to canonical MKL, we alternately optimize $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to solve the max-min problem. In the first stage, we fix $\boldsymbol{\beta}$ and optimize $\boldsymbol{\alpha}$. It is easy to know that this problem is a canonical SVM with the specific combination kernel, while there are many algorithms with it. In the second stage, we fix $\boldsymbol{\alpha}$ and optimize $\boldsymbol{\beta}$. J can be rewritten as follows:

$$J(\beta) = \sum_{g=1}^{G}\sum_{g'=1}^{G}\beta_m^g\beta_m^{g'}S_m^{gg'}(\alpha^*) - \sum_{i=1}^{N}\alpha_i^* \tag{13}$$

where

$$S_m^{gg'}(\boldsymbol{\alpha}^*) = \frac{1}{2}\sum_{\{i|c(x_i)=g\}}\sum_{\{i|c(x_i)=g'\}}\alpha_i^*\alpha_j^* y_i y_j K_m(x_i,x_j) \tag{14}$$

and $\boldsymbol{\alpha}^*$ is the optimization result of $\boldsymbol{\alpha}$.

Note that the solving of $J(\beta)$ in Equation (13) is independent of the latter item, though it equals solving the problem of Equation (15).

$$\max_{\boldsymbol{\beta}}\sum_{g=1}^{G}\sum_{g'=1}^{G}\sum_{m=1}^{M}\beta_m^g\beta_m^{g'}S_m^{gg'}(\alpha) \tag{15}$$

It is obvious that this is a quadratic non-convex problem. We know that solving the quadratic programming problem needs expensive computation. Inspired by [17,22]. A gating model is adopted to represent $\boldsymbol{\beta}$. The gating function is designed as follows:

$$\beta_m^g = \frac{\exp\left(a_m^g v_m^g + b_m^g\right)}{\left(\sum_{m'=1}^{M}\exp p\left(a_{m'}^g v_{m'}^g + b_{m'}^g\right)\right)^{\frac{1}{p}}} \tag{16}$$

where v_m^g is the kernel alignment [24] of the mth kernel in sample group of g, and it can be calculated by Equation (17).

$$v_m^g = \frac{\left\langle K_m^g, y_g y_g^T\right\rangle_F}{\sqrt{\left\langle K_m^g, K_m^g\right\rangle_F\left\langle y_g y_g^T, y_g y_g^T\right\rangle_F}} \tag{17}$$

where

$$\langle K_p, K_q\rangle_F = \sum_{i,j}K_p(x_i,x_j)K_q(x_i,x_j) \tag{18}$$

We can observe that $\sum_{m=1}^{M}(\beta_m^g)^p = 1$ from Equation (16), so there is actually a p norm constraint with β_m^g. We can optimize p according to dataset to change the sparseness of kernels. Considering the property varies from different local spaces, it is an interesting

idea to employ different p in each group. However, we have no idea to confirm so many p values, and this problem can be further researched in next work.

After representing the local kernel weights with the gating model, $J(\beta)$ becomes a convex function about $\boldsymbol{a}$ and $\boldsymbol{b}$. Thus, we can optimize $\boldsymbol{a}$ and $\boldsymbol{b}$ with the gradient-descent method to maximize $J(\beta)$. The derivative of $J(\beta)$ is as follows:

$$\frac{\partial J(\beta)}{\partial a_m^g} = 2\sum_{l}^{M}\left(\sum_{i=1}^{G}\left(\beta_l^i S_l^{ig}(\alpha)\right)\beta_m^g v_m^g\left(\delta_m^l - (\beta_l^g)^p\right)\right) \tag{19}$$

$$\frac{\partial J(\beta)}{\partial b_m^g} = 2\sum_{l}^{M}\left(\sum_{i=1}^{G}\left(\beta_l^i S_l^{ig}(\alpha)\right)\beta_m^g\left(\delta_m^l - (\beta_l^g)^p\right)\right) \tag{20}$$

where $\delta_m^l = 1$ if $l = m$ and $\delta_m^l = 0$ otherwise. We update $\boldsymbol{a}$ and $\boldsymbol{b}$ by gradient-descent method, and then update β_m^g with $\boldsymbol{a}$ and $\boldsymbol{b}$.

$$a_m^g + \lambda^t \frac{\partial J(\beta)}{\partial a_m^g} \to a_m^g \tag{21}$$

$$b_m^g + \mu^t \frac{\partial J(\beta)}{\partial b_m^g} \to b_m^g \tag{22}$$

where λ^t and μ^t are the step sizes, and they can be obtained by a line search method like [6] or fixed as a small constant.

Alternatively optimize $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ until some termination criterions are met. We use the duality gap as the termination criterion, which is written as follows:

$$\max_m \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K_m(x_i,x_j) - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\sum_{m=1}^{M}\beta_m^{c(x_i)}\beta_m^{c(x_j)}K_m(x_i,x_j)\right) \le \varepsilon \tag{23}$$

where ε is the tolerance threshold set in advance.

The complete optimization process of *lp*-GLMKL for binary classification is summarized in Algorithm 1.

Algorithm 1 *lp*-GLMKL method for binary classification

1: Divide the training samples into different groups
2: Initialize $\boldsymbol{a}$, $\boldsymbol{b}$ with small random numbers
3: Repeat
4: Calculate local kernel weights β_m^g as Equation (16);
5: Calculate kernel matrix as Equation (11);
6: Solve support vector coefficient $\boldsymbol{\alpha}$ using the canonical SVM
7: Update gating model parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ as Equations (21) and (22)
8: Until meet the termination criterion of Equation (23).

3. **Kernel K-Means Clustering for the *lp*-GLMKL.** Unlike sample-based localized method, our *lp*-GLMKL is a group based algorithm. There is a clustering stage before training the classifier. We need to design an effective clustering algorithm suitable for our *lp*-GLMKL. Because the training samples are represented by multiple features or kernels, traditional clustering algorithm cannot effectively solve this problem. In this section, we present a multi-kernel K-means clustering method. We merge the m kernel matrixes into one and then use the kernel K-means clustering algorithm. The weights of the kernel matrixes are obtained by centered KTA (CKTA) [25]. The CKTA is a novel kernel

alignment which performs well in evaluating the kernel matrixes. The kernel matrixes are centered as follows:

$$K_{ci} = \left[I - \frac{11^T}{n}\right] K_i \left[I - \frac{11^T}{n}\right] \tag{24}$$

Then we calculate the weights of the kernel matrixes as:

$$\eta_i = \frac{F(K_i, y)}{\sum_{j=1}^{m} F(K_j, y)} \tag{25}$$

where

$$F(K_{ci}, \boldsymbol{y}) = \frac{\left\langle K_{ci}, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F}{\sqrt{\langle K_{ci}, K_{ci} \rangle_F \left\langle \boldsymbol{y}\boldsymbol{y}^T, \boldsymbol{y}\boldsymbol{y}^T \right\rangle_F}} \tag{26}$$

The m kernel matrixes are merged into one matrix as follows:

$$K = \sum_{i=1}^{m} \eta_i K_i \tag{27}$$

The kernel matrixes K are taken as the distance of samples, then implementing the kernel clustering on K. The clustering error can be calculated by

$$E(C_1, \cdots, C_G) = \sum_{n=1}^{N_p} \sum_{g=1}^{G} I(x_n \in C_g) \left\| x_n - C_g \right\|^2 \tag{28}$$

where

$$\left\| x_n - C_g \right\|^2 = K_{nn} - \frac{2\sum_{l=1}^{N_p} I(x_n \in C_g) K_{ln}}{\sum_{l=1}^{N_p} I(x_n \in C_g)} + \frac{\sum_{i=1}^{N_p} \sum_{j=1}^{N_p} I(x_i \in C_g) I(x_j \in C_g) K_{ij}}{\sum_{i=1}^{N_p} \sum_{j=1}^{N_p} I(x_i \in C_g) I(x_j \in C_g)} \tag{29}$$

The clustering result of x_i is computed by:

$$C(x_i) = \arg\min_{g} \left(\left\| x_i - C_g \right\|^2 \right) \tag{30}$$

The process of clustering for *lp*-GLMKL is summarized in Algorithm 2.

Figure 2 shows the complete flowchart of using *lp*-GLMKL for training and testing.

Algorithm 2 Kernel K-means clustering with multiple views

Input: m kernel matrixes, number of groups G
Output: Final clustering results
1: Calculate the weights of the kernel matrixes as Equation (25);
2: Merge the m kernel matrixes to one matrix as Equation (27);
3: Initialize clusters $C_1, \cdots, C_G$;
4: Repeat
5: Calculate the clustering error as Equation (28);
6: Find $C(x_i)$ as Equation (30);
7: Update clusters;
8: Until convergence;

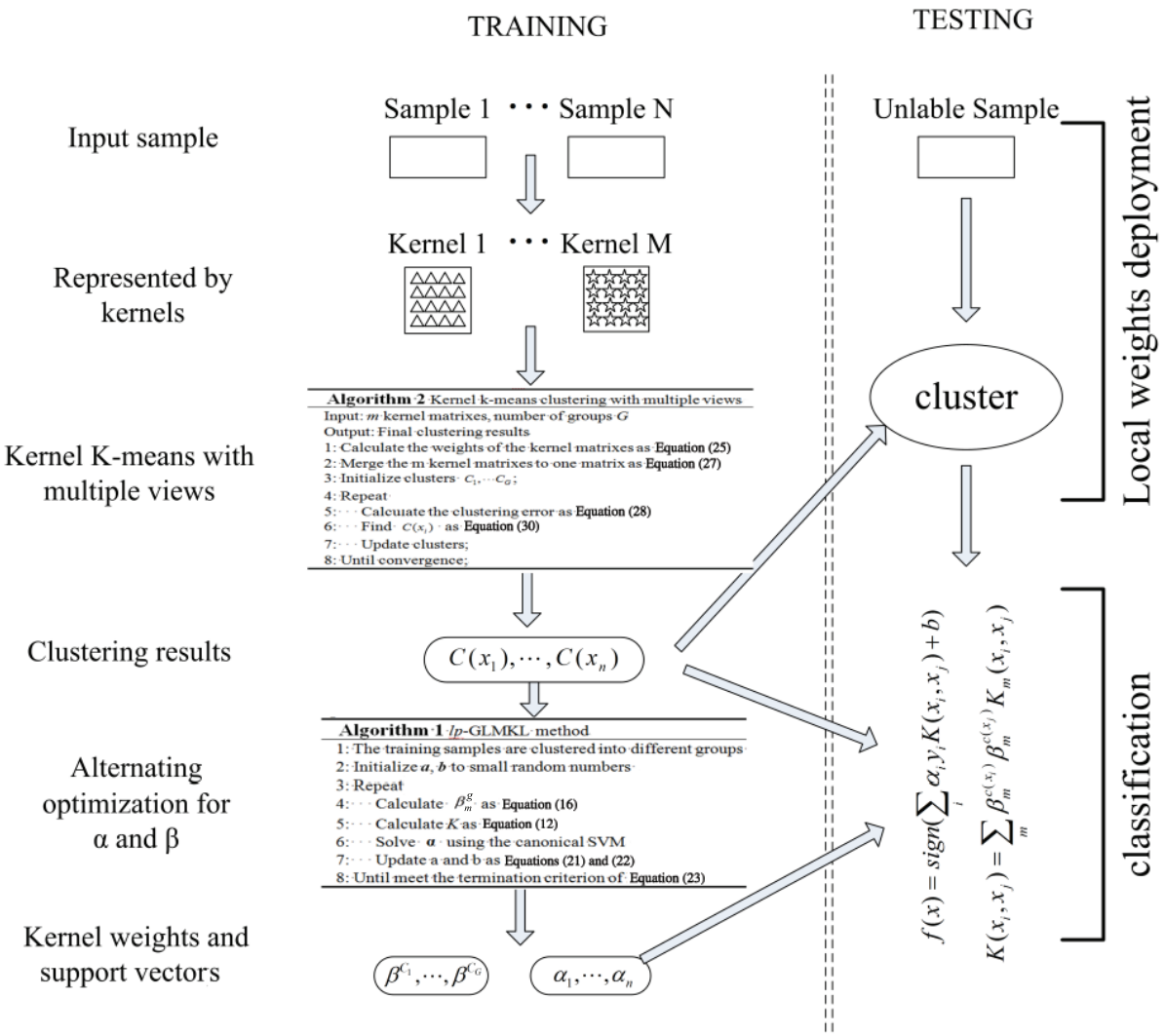

FIGURE 2. Flowchart of using *lp*-GLMKL for classification

4. **Experiments and Analyses.** The proposed algorithm was implemented in MATLAB and the canonical SVM problem was solved by SMO [4]. We take the canonical MKL (Simple MKL) [6], the canonical sample-based localized MKL (LMLK) [17], and the group-based MKL (GS-MKL) [22] as the baseline approaches.

We compared the performance with the baseline approaches in three sets of experiments. Firstly, the MKL methods were implemented on synthetic dataset to analyze the decision boundary. Secondly, the experiments were implemented on UCI datasets to evaluate the inference of different parameters and compare to baseline MKL methods. Finally, two image datasets were used to evaluate the performance of the MKL methods on computer vision.

4.1. **Experiments on synthetic dataset.** In this section, we created a two dimensions synthetic dataset like [17,19]. The dataset consists of 400 data samples generated from four Gaussian components (each class has two Gaussian components) with the following mean and covariance matrices.

$$\mu_{11} = \begin{pmatrix} -1.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{11} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$$

$$\mu_{12} = \begin{pmatrix} +3.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{12} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$$

$$\mu_{21} = \begin{pmatrix} -3.0 \\ -3.0 \end{pmatrix} \quad \Sigma_{21} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$$

$$\mu_{22} = \begin{pmatrix} +1.0 \\ -3.0 \end{pmatrix} \quad \Sigma_{22} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$$

We respectively use the four MKL algorithms (the three baseline methods and our algorithm) to calculate the boundary of the positive and negative samples. The candidate kernels consisted of linear kernel and polynomial kernel with one degree. For our algorithm and GS-MKL, the number of groups G was set to 4. The norm constraint of p was fixed to 2 for our algorithm. Because the data distribution is known to us, we could directly obtain the Bayesian decision boundary. It is obvious that the Bayesian boundary is the optimal solution for this classification problem. Hence, how well the decision boundaries match to the Bayesian boundary can be used to evaluate the performance of the classifier. The optimal boundary and experiment results are shown in Figure 3. The solid blue line is the Bayesian boundary and the dotted black line is the decision boundary learned by MKL algorithms. The solid red and blue points are the support vectors.

The decision boundary of canonical MKL is nearly a beeline, because it does not have the ability of local learning. The decision boundary learnt by the GS-MKL and LMKL match much better to the Bayesian boundary than the Simple MKL. However, there is

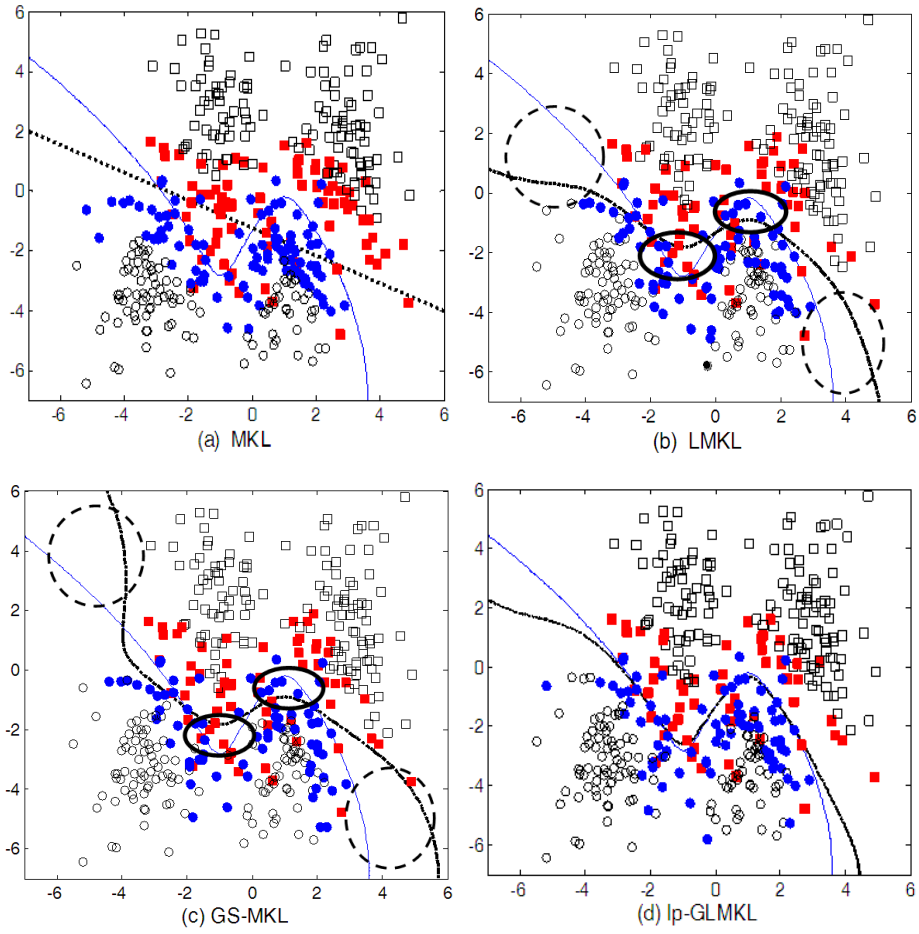

FIGURE 3. Bayesian boundary and decision boundaries learnt by the MKL methods

also a big gap in the sparse region signed as dotted circles. In addition, the decision boundary matches not well in the dense region signed as solid circles. Our algorithm obtained a much better decision boundary than the other three approaches. Because we use a non-sparse constraint on the kernels and we design a novel clustering algorithm with multiple kernels, the advantage of the two kernels is also adopted. Then we analyze the number of support vectors. Our method needs fewer support vectors than other MKL methods, which intuitively demonstrates the effectiveness in the testing processing.

4.2. **Experiments on UCI datasets.** Then we took some experiments on UCI dataset to evaluate the performance of our method. Since each dimension could be thought of a view of the dataset, we used one Gaussian kernel to represent each dimension of the datasets. For example, due to the fact that the "Ionosphere" dataset consisted of 34 dimensions, 34 kernels were used for the "Ionosphere" dataset. The width of Gaussian kernel was equal to half the mean-squared distance of the associated view [26]. In this section, three experiments were implemented. Firstly, we evaluated the influence of various g (number of groups). Secondly, the experiment was taken to test the influence of p on kernel weights and classification accuracy. Finally, the performance of the proposed algorithm was compared to the baseline approaches. For every dataset, 80% of samples were randomly selected for training and the rest for testing. All the experiments were repeated 10 times. Table 1 lists the detail of the eight UCI datasets.

Table 1. Parameter values of the UCI datasets

Datasets	Number of instances	Number of features	Number of classes
Heart	270	13	2
Ionosphere	351	34	2
Liver	345	6	2
Sonar	208	60	2
Spambase	4601	57	2
Wbdc	569	30	2
Pima	768	8	2
Glass	214	9	6

4.2.1. *Influence of G (number of groups).* In our method, the samples are divided into groups. So the influence of group number was evaluated in this experiment. We respectively used $[1, 5, 10, 20, N]$ (N is the number of samples) as the number of groups to train the classifier. And the results are shown in Figure 4. As shown in Figure 4, the classification accuracies vary with the value of G. While $G = 1$, the results are close to the canonical MKL method, and the results are close to the results of the LMKL method while $G = N$. Therefore, the optimal classification accuracies can be obtained by selecting the appropriate value of G.

4.2.2. *Influence of various p.* In this experiment, five different values of p composed of $\{1, 1.2, 1.4, 1.6, 1.8, 2\}$, were chosen to evaluate the influence of norm constraint. For a specific sample, Figure 5 shows the weights of the kernels on the training set of the "Ionosphere" with various p norms. We can observe that the sparsity of the learned kernels decreases as the increasing of p. While $p = 1$, a few kernels play a role in the classification task. That is to say, the result is sparse. As p increases, more kernels come into use. While $p \to \infty$, the kernel weights are similar to each other.

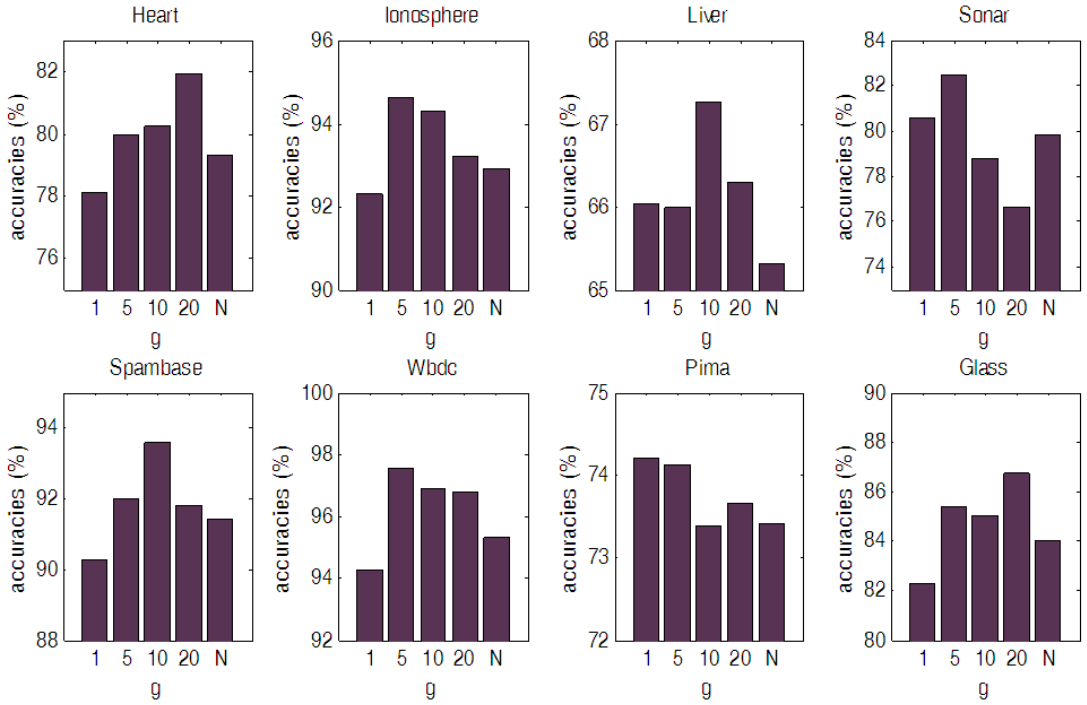

FIGURE 4. Classification accuracies (%) with various G (number of groups)

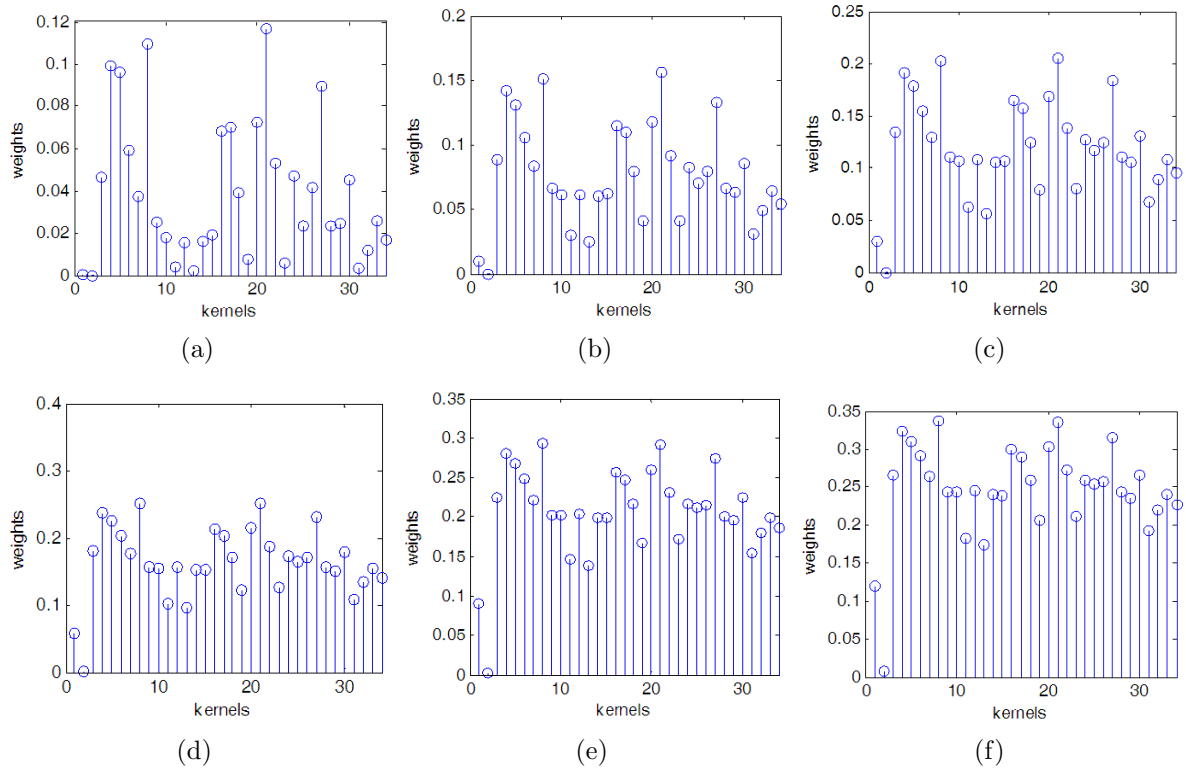

FIGURE 5. Combination weights of kernels with various p norms for a given sample

The testing accuracies with various p are shown in Figure 6. We observe that the classification accuracies vary with the value of p. By choosing appropriate p, our method can obtain better classification accuracy. The optimal p is not fixed for different datasets, which depends on the distribution of dataset. As given in [11], the optimal value of p is

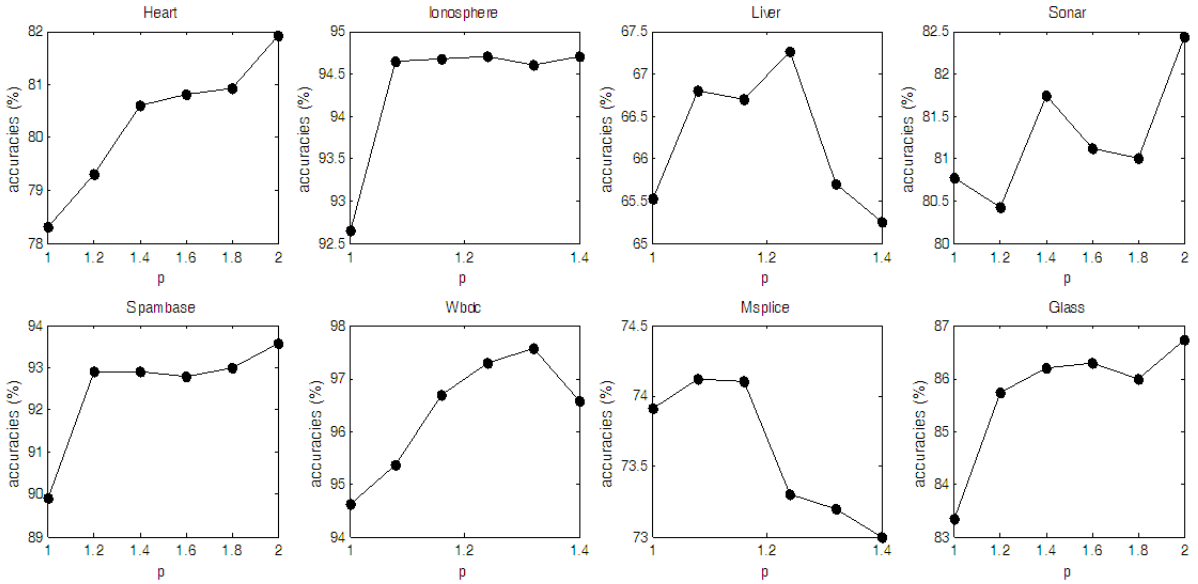

FIGURE 6. Classification accuracies (%) with various p

TABLE 2. Classification accuracies (%) of different algorithms

Dataset	Simple MKL	LMKL	GS-MKL	*lp*-GLMKL
Heart	78.12	79.33	78.32	**81.93**
Ionosphere	92.29	92.90	92.65	**94.65**
Liver	66.03	65.32	65.53	**67.26**
Sonar	80.60	79.83	80.78	**82.43**
Spambase	90.26	91.42	89.91	**93.57**
Wbdc	94.29	95.32	94.62	**97.58**
Pima	**74.21**	73.41	73.91	74.12
Glass	82.28	84.03	83.33	**86.74**

generally bigger than 1. Our experiments show that this rule is also suitable for localized MKL.

4.2.3. *Comparative evaluation.* In this experiment, we compared our algorithm to the baseline MKL methods. The parameters were acquired in Section 4.2.1 and Section 4.2.2. The results are listed in Table 2.

As we can see in Table 2, our method achieves the best classification accuracy in seven of the eight datasets and the result on "Pima" dataset is very close to the best one. The results show that our method performs better than other MKL methods. Because we select the optimal norm constraint and group number, the better combination kernel is learnt for each local space. For the "Pima" dataset, the canonical MKL obtained the best classification accuracy. The reason may be that this dataset is not suitable for local learning method since the kernels play a same role in each local space. However, by accommodating the parameters, the classification accuracy of our algorithm is very close to the best one.

4.3. **Experiments on visual image categorization.** We finally applied our algorithm to two visual image datasets: Scene-15 and Caltech-101. Scene-15 dataset contains 15 scene categories, while each category has 200 to 400 images. Caltech-101 involves 102 object categories, while each category contains 31 to 800 images. In order to make the

experimental results comparable, we followed the usual setting of training and testing sets. For each dataset, the number of images from each category was fixed, but the images were randomly selected while repeating the experiments. For Caltech-101, we used 10, 15, 20, 25, 30 images per category as training set and the rest for testing set. For Scene-15, the training set of each category was set to 10, 20, 30, 50, 100 and the rest were taken as testing set. The hyper-parameters of each algorithm were tuned using cross-validation procedures.

We used eight descriptors to represent the images, i.e., GIST, histogram of oriented gradient 2×2, dense SIFT, sparse SIFT, line histograms, self-similarity, texton, and geometric map. The associated kernels were computed based on radial basis function distance for GIST and $\chi 2$ distances for all the others. All kernels were normalized to unit trace. The one-versus-all rule was used to solve the multiclass classification problem.

The average classification accuracies are shown in Table 3 and Table 4. In terms of classification accuracies, *lp*-GLMKL performs better than the baseline method. When the number of training samples is small, the advantage of our algorithm is not obvious. For both the two datasets, when training number is 10, LMKL obtained the best accuracy. The reason is that the training set is too small to divide into groups. With the increasing of training samples, the advantage of our algorithm becomes more obvious. Then we look at the variance of accuracies. The variance of our algorithm is less than the other two local learning algorithms. This indicates our algorithm is more reliable.

Table 3. Classification accuracies on Caltech-101

Algorithm	**Number of positive training samples per category**				
	10	15	20	25	30
Simple MKL	66.4±1.1	70.5±1.1	73.7±1.0	75.2±0.8	75.7±0.8
LMKL	**69.2±1.5**	75.3±1.2	77.7±1.1	79.5±1.1	80.4±1.0
GS-MKL	66.3±1.3	75.3±1.1	81.6±1.1	83.6±1.0	84.4±0.9
***lp*-GMKL**	67.1±1.2	**75.4±1.1**	**82.1±1.0**	**84.1±1.0**	**85.0±0.8**

Table 4. Classification accuracies on Scene-15

Algorithm	**Number of positive training samples per category**				
	10	20	30	50	100
Simple MKL	59.7±1.2	65.9±1.1	71.0±1.0	77.2±0.8	82.9±0.6
LMKL	**62.5±1.6**	67.1±1.3	72.7±1.1	78.3±0.9	84.3±0.8
GS-MKL	61.3±1.4	68.4±1.2	74.3±1.1	80.8±1.0	86.6±0.9
***lp*-GMKL**	62.3±1.3	**69.1±1.1**	**74.9±0.9**	**82.0±0.8**	**88.1±0.7**

5. **Discussion.** In this paper, we proposed a group based localized multiple kernel learning algorithm with *lp*-norm constraint. Since the samples were represented by multiple kernels or features, we designed a multi-kernel K-means clustering algorithm for data pre-procedure. Thanks to the fact that the *lp*-norm constraint was employed for the local kernel weights, we obtained a non-sparse results of kernels. The performance of classifier is improved by adjusting the sparsity of kernels. The experiments confirm the improvement of our algorithm on classification accuracy.

However, there are more parameters in the new model, which cannot be obtained in advance. In next work, we will try to research the relationship between the parameter p and the property of specific dataset. If the optimal p can be directly calculated or

estimated, the cross-validation could be omitted. In addition, the proposed algorithm can also be extended to other learning settings and further application.

Acknowledgments. This work was jointly supported by the National Natural Science Foundation for Young Scientists of China (Grant Nos: 61202332, 61403397) and the Natural Science Foundation of Shaanxi Province, China (Grant No: 2015JM6313). The authors also gratefully acknowledge the helpful comments and suggestion of the reviewers, which have improved the presentation.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
[2] M. Gonen and E. Alpaydm, Multiple kernel learning algorithms, *Journal of Machine Learning Research*, vol.12, pp.2211-2268, 2011.
[3] K. P. Bennett, M. Momma and M. J. Embrechts, MARK: A boosting algorithm for heterogeneous kernel models, *Proc. of the 8th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, pp.24-31, 2002.
[4] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui and M. I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research*, vol.1, no.5, pp.27-72, 2004.
[5] F. R. Bach, G. R. G. Lanckriet and M. I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, *Proc. of the 21st International Conference on Machine Learning*, Banff, Canada, pp.41-48, 2004.
[6] S. Sonnenburg, G. Ratsch, C. Schafer and B. Scholkopf, Large scale multiple kernel learning, *Journal of Machine Learning Research*, vol.7, no.7, pp.1531-1565, 2010.
[7] A. Rakotomamonjy, F. R. Bach, S. Canu and Y. Grandvalet, Simple MKL, *Journal of Machine Learning Research*, vol.11, no.9, pp.2491-2521, 2008.
[8] Y. Lu, L. Wang and J. Lu, Multiple kernel clustering based on centered kernel alignment, *Pattern Recognition*, vol.47, no.11, pp.3656-3664, 2014.
[9] A. Nazarpour and P. Adibi, Two-stage multiple kernel learning for supervised dimensionality reduction, *Pattern Recognition*, vol.48, no.5, pp.1854-1862, 2015.
[10] X. Tian, G. Gasso and S. Canu, A multiple kernel framework for inductive semi-supervised SVM learning, *Neurocomputing*, vol.90, no.8, pp.46-58, 2012.
[11] K. Marius, B. Ulf, S. Soren and Z. Alexander, *lp*-norm multiple kernel learning, *Journal of Machine Learning Research*, vol.12, pp.953-997, 2011.
[12] T. Sun, L. Jiao, F. Liu et al., Selective multiple kernel learning for classification with ensemble strategy, *Pattern Recognition*, vol.46, no.11, pp.3081-3090, 2013.
[13] Z. Xu, R. Jin, H. Yang et al., Simple and efficient multiple kernel learning by group lasso, *Proc. of the 27th International Conference on Machine Learning*, Haifa, Israel, pp.1175-1182, 2010.
[14] F. Aiolli and M. Donini, EasyMKL: A scalable multiple kernel learning algorithm, *Neurocomputing*, vol.169, pp.215-224, 2015.
[15] B. Ni, T. Li and P. Moulin, Beta process multiple kernel learning, *IEEE Conference on Computer Vision and Pattern Recognition*, pp.963-970, 2014.
[16] I. Rebai, Y. Benayed and W. Mahdi, Deep multilayer multiple kernel learning, *Neural Computing & Applications*, pp.1-10, 2015.
[17] M. Gonen and E. Alpaydin, Localized algorithms for multiple kernel learning, *Pattern Recognition*, vol.46, no.3, pp.795-807, 2013.
[18] M. Varma, Local deep kernel learning for efficient non-linear SVM prediction, *International Conference on Machine Learning*, pp.486-494, 2013.
[19] Y. Han and G. Liu, Probability-confidence-kernel-based localized multiple kernel learning with $l(p)$ norm, *IEEE Trans. Systems, Man, and Cybernetics. Part B, Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society*, vol.42, no.3, pp.827-837, 2012.
[20] Y. Han, K. Yang, Y. Ma et al., Localized multiple kernel learning via sample-wise alternating optimization, *IEEE Trans. Cybernetics*, vol.4, no.1, pp.127-148, 2013.
[21] Y. Lei, A. Binder, Ü. Dogan and M. Kloft, Theory and algorithms for the localized setting of learning kernels, *Journal of Machine Learning Research*, vol.7, no.7, pp.173-195, 2015.

[22] J. J. Yang, Y. N. Li, Y. H. Tian, L. Y. Duan and W. Gao, Groupsensitive multiple kernel learning for object categorization, *Proc. of the 12th IEEE International Conference on Computer Vision and Pattern Recognition*, New York, USA, pp.436-443, 2012.

[23] G. Fu, Q. Wang et al., Group based non-sparse localized multiple kernel learning algorithm for image classification, *Proc. of the 4th IEEE International Conference on Cloud Computing and Intelligence Systems*, Beijing, China, pp.192-196, 2016.

[24] N. Cristianini, J. Shawe-Taylor, A. Elisseeff and J. Kandola, On kernel-target alignment, *Proc. of the Advances in Neural Information Processing Systems*, MIT Press, Cambridge, USA, pp.367-373, 2002.

[25] C. Cortes, M. Mohri and A. Rostamizadeh, Algorithms for learning kernels based on centered alignment, *Journal of Machine Learning Research*, vol.13, no.2, pp.795-828, 2012.

[26] M. Varewyck and J.-P. Martens, A practical approach to model selection for support vector machines with a Gaussian kernel, *IEEE Trans. Syst., Man, Cybern. B, Cybern*, vol.41, no.2, pp.330-340, 2011.