Ensemble of Multiple Kernel SVM Classifiers

Xiaoguang Wang[1], Xuan Liu[1], Nathalie Japkowicz[1], and Stan Matwin[2,3]

[1]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
{Bwang009,Nat}@eecs.uottawa.ca,
Xliu107@.uottawa.ca
[2]Faculty of Computer Science, Dalhousie University, Canada
[3]Institute for Computer Science, Polish Academy of Sciences, Poland
Stan@cs.dal.ca

Abstract. Multiple kernel learning (MKL) allows the practitioner to optimize over linear combinations of kernels and shows good performance in many applications. However, many MKL algorithms require very high computational costs in real world applications. In this study, we present a framework which uses multiple kernel SVM classifiers as the base learners for stacked generalization, a general method of using a high-level model to combine lower-level models, to achieve greater computational efficiency. The experimental results show that our MKL-based stacked generalization algorithm combines advantages from both MKL and stacked generalization. Compared to other general ensemble methods tested in this paper, this method achieves greater performance on predictive accuracy.

Keywords: multiple kernel learning, stacked generalization, ensemble learning.

1 Introduction

Ensembles of models are sets of models whose outputs are combined into a single output or prediction. Stacked generalization [1] is a heterogeneous ensemble method for combining multiple classifiers (base models) by learning a meta-level classifier based on the output of the base-level classifiers, estimated via cross-validation. When we choose the base learners for ensemble methods such as stacked generalization, Kernel based methods [5] [6] such as Support Vector Machines (SVMs) [4] could be one of the choices. Joachims et al. [24] show that combining two kernels is beneficial if both of them use different data instances as support vectors and achieve approximately the same performance. Recent developments on SVMs and other kernel methods have shown the need to consider multiple kernels. This provides flexibility and reflects the fact that typical learning problems often involve multiple, heterogeneous data sources.

The reasoning is similar to combining different classifiers: instead of choosing a single kernel function, it is better to have a set and let an algorithm do the picking or combination step. MKL can be useful in two aspects:

M. Sokolova and P. van Beek (Eds.): Canadian AI 2014, LNAI 8436, pp. 239–250, 2014.
© Springer International Publishing Switzerland 2014

- Since a kernel plays the role of defining the similarity between instances, different kernels correspond to different notions of similarity, and using a specific kernel may be a source of bias. To avoid this we can import a learning method to pick the best kernel or use a combination of a kernel set. In allowing a learner to choose from a set of kernels, a better solution can be found.
- Different kernels may use inputs from different representations. Since different kernels may have different measures of similarity, combining kernels can be done to combine multiple information sources.

There are many outstanding advantages of MKL. However, most MKL algorithms have some limitations in application. For instance, most MKL methods do not consider the group structure between the combined kernels. In multiple kernels learning (MKL), increasing the number of candidate kernels leads to better accuracy, but also increases the training time significantly [3]. Our idea is: if we separate the kernel set into subsets, each subset of kernels can lead to a different combination of kernels. Using an ensemble method such as majority voting or stacked generalization, the outputs of each MKL model can be combined into a single prediction. In this case, fewer kernels are required to be handled by each base MKL learner of this ensemble model than the number of kernels need to be handled by using a single MKL model. Since stacked generalization is an ideal method for parallel computation, using MKL as the base learner, this stacking MKL method can combine more kernels and process more instances in a fixed time than using a single MKL model. MKL performs better than the single kernel method but has a high cost. By enforcing sparse coefficients, MKL also generalizes feature selection to kernel selection. Stacked generalization is an efficient algorithm to combine heterogeneous classifiers and it can also benefit from diversity of data distribution. Our target is to combine advantages from both methods. The idea of combining two ensemble methods is frequently used. For instance, Wolpert et al. [15] present several ways that stacking can be used in agreement with the bootstrap procedure to achieve a further improvement on the performance of bagging for some regression problems. Kai et al. [16] also present ways of combining bagging and stacking for classification.

Contributions:

- In this paper, we combine the multiple kernel learning algorithms with stacked generalization (denoted as SMKL). Experimental results show that this algorithm can benefit from both methodologies.
- Results also show that even without using parallel computation, SMKL can process more instances with the same number of kernels or combine more kernels with the same number of instances than MKL. This makes SMKL adaptable to real world applications.
- We analyze this algorithm and compare it to other ensemble methods. A statistical explanation of how this method works is also given.

Section 2 is about the background and related work. In section 3, we present the algorithm SMKL and present some discussion and related work about it. After these

developments, we present an experimental section (section 4) that illustrates the efficiency of our algorithm and some concluding remarks. In section 5, we give a detailed analysis about ensemble methods using SVMs as a base learner. Section 6 is the conclusion followed by references.

2 Related Theory and Work

Stacked generalization [1] is a way of combining multiple models that have been learned for a classification task. This method has also been used for regression and unsupervised learning [14] [15].

In the most common form of stacked generalization, the first step is to collect the output of each model into a new set of data. For each instance in the original training set, this data set includes all models' predictions of that instance for every class along with its true classification. In the second step, this new data is treated as the data for another learning problem. A learning algorithm is employed to solve this problem.

The key idea of MKL is to learn a linear combination of a given set of base kernels by maximizing the margin between the two classes or by maximizing kernel alignment. We can think of kernel combination as a weighted average of kernels and consider the weight $\beta \in \mathbb{R}_+^P$ and $\sum_{m=1}^{P} \beta_m = 1$, where P denotes the number of weights. Suppose one is given n $m \times m$ symmetric kernel matrices $K_j, j = 1, \dots, n$, and m class class labels $y_i \in \{1, -1\}, i = 1, \cdots, m$. A linear combination of the n kernels under an ℓ_1 norm constraint is considered:

$$K = \sum_{j=1}^{n} \beta_j K_j, \beta \geq \emptyset, \|\beta\|_1 = 1 \tag{1}$$

where $\beta = (\beta_1, \dots, \beta_n)^T \in \mathbb{R}^n$, and $\emptyset$ is the n dimensional vector of zeros. Geometrically, different scaling of the feature spaces leads to different embeddings of the data in the composite feature space. The goal of MKL is to learn the optimal scaling of the feature spaces and maximize the so-called "separability" of the two classes in the composite feature space.

In kernel methods, the choice of a kernel function is critical, since it completely determines the embedding of the data in the feature space. Ideally, this embedding should be learnt from the training data. In practice, a simplified version of this very challenging problem is often considered: given multiple kernels capturing different "views" of the problem, an "optimal" combination of them must be learned.

Lanckriet et al. [9] have proposed to use the soft margin of SVM as a measure of separability, that is, to learn the weight β by maximising the soft margin between the two classes. Bach et al. [10] have reformulated the problem and then proposed a SMO algorithm for medium-scale problems. Cortes et al. [22] discuss the suitability of the 2-norm for MKL. In their paper they conclude that using the 1-norm improves the performance for a small number of kernels, but not for a large number of kernels. Meanwhile, the 2-norm increases the performance significantly for larger sets of candidate kernels and never decreases it.

The performance improvement of MKL comes at a price. Learning the entire set of models and then combining their predictions is computationally more expensive than learning just one simple model. The computational complexity of MKL is very high for two major reasons: 1). Similar to normal kernel based methods, MKL needs to compute kernel functions for each sample-pair over the training set; 2). MKL needs to

optimize the classifier parameters and kernel weights in an alternative manner, thus learning global optimal parameters would incur intensive computation. More specifically, MKL that use optimization approaches to learn combination parameters have high computational complexity, since they are generally modeled as a semi definite programming (SDP) problem, a quadratically constrained quadratic programming (QCQP) problem, or a second-order cone programming (SOCP) problem. MKL can also be modeled as a semi-infinite linear programming (SILP) problem [10], which uses a generic linear programming (LP) solver and a canonical SVM solver in the inner loop. This method is more efficient than previous methods [10], but the computational complexity is still very high.

In recent years, there has been an effort made to reduce the computational complexity of SVM algorithm [25] [26]. For MKL, Chen et al. [27] have proposed a method by dividing the global problem with multiple kernels into multiple local problems, each of which is optimized in a local processor with a single kernel. In this paper, we present an alternative method by combining stacked generalization with MKL. The following section will give the details of this framework.

3 Stacked Generalization on Multiple Kernel SVMs

Here we gave the detailed steps of the stacked generalization base on MKLs method as below:

- Step1: Given a data set $S = \{(y_n, x_n), n = 1, \ldots, N\}$, where y_n is the class value and and x_n is a vector representing the attribute values of the nth instance, randomly split the data into J almost equal parts $S_1, \ldots, S_J$. Define S_j and $S^{-j} = S - S_j$ to be the the test and training sets for the jth fold of a J-fold cross-validation.

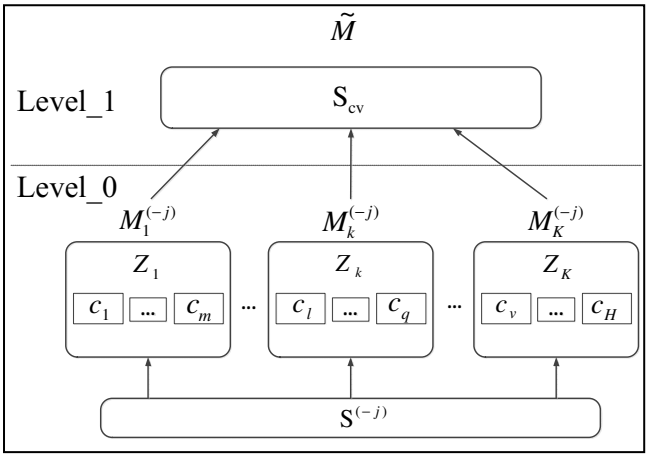

Fig. 1. This figure illustrates the j-fold cross-validation process in level-0; the level-1data set S_{cv} at the end is used to produce level-1 model $\tilde{M}$

- Step 2: Instead of choosing K learning algorithms$\{Z_1, Z_2, \dots, Z_K\}$ directly as the level-0 generalizers, here we choose H number of kernels $C = \{c_1, c_2, \dots, c_H\}$. Divide this set into K groups. Call the Multi-kernel algorithm as introduced in section II.b with each group of kernels to build the base learning algorithms $\{Z_1, Z_2, \dots, Z_K\}$. Invoke the kth algorithm on the data in the training set S^{-j} to induce a model M_k^{-j}, for k=1,...,K. These are called level-0 models. For each instance x_n in S_j, the test set for the jth cross-validation fold, let Z_{kn} denote the prediction of the model M_k^{-j} on x_n . At the end of the entire cross-validation process, the data set assembled from the outputs of the K models is: $S_{CV} = \{(y_n, Z_{1n}, \dots, Z_{Kn}), n = 1, \dots, N\}$. These are the level-1 data.
- Step3: Use the same Multi-kernel algorithm as in step 2 with all the kernels of $C = \{c_1, c_2, \dots, c_H\}$ to be the learning algorithm V^mas the level-1 generalizer. A model $\widetilde{M}$ can be derived for y as a function of $(Z_1, \dots, Z_K)$. This is the level-1 model. To complete the training process, the final level-0 models M_k, k=1,...,K, are are derived using all the data in S.Now let us consider the classification process, which uses the models M_k, k=1,...,K, in conjunction with $\widetilde{M}$. Let z_k denotes the prediction output vector of function Z_k. Given a new instance, models M_k produce a vector $(z_1, \dots, z_K)$. This vector is input to the level-1 model $\widetilde{M}$, whose output is the final classification result for that instance.

This algorithm is presented in Fig.2 and Fig.1 gives its workflow.

In this SMKL algorithm, we choose $MKL(C_k)$ as the level-0 generalizers instead of of using common classifiers and we use $MKL(C)$ as the level-1 generalizer. In the next sections we will discuss the relationship between the attributes used in the Meta learning set and the Meta learning algorithm used for learning the Meta model.

Input: Learning set S; Number of folds for meta-data generation J; Meta learning algorithms V^m; Base learning algorithms $\{Z_1, Z_2, \dots, Z_K\}$; Kernel set $C = \{c_1, c_2, \dots, c_H\}$; And Multi-kernel Function $MKL()$.
Output: Ensemble E
$E = \emptyset$
$\{S_1, S_2, \dots, S_J\} = SplitData(S, J)$
$L^m = \emptyset$
$\{C_1, C_2, \dots, C_K\} = SplitSet(C, K)$
for k= 1 to K **do**
  $Z_k = MKL(C_k)$
end for
$V^m = MKL(C)$
for j= 1 to J **do**
  for k= 1 to K **do**
  $M_k^j = Z_k(S - S_j)$
  end for

$$L_j^m = \bigcup_{x_i \in S_j} \left\{ \left(M_1^j(x_i), M_2^j(x_i), \ldots M_K^j(x_i), y_i \right) \right\}$$

end for

$$L^m = \bigcup_{j=1}^{J} L_j^m$$

$$M^m = V^m(L^m)$$

$$\{M_1, M_2, \ldots M_K\} = \{Z_1(S), Z_2(S), \ldots, Z_K(S)\}$$

$$E = (\{M_1, M_2, \ldots M_K\}, M^m)$$

return E

Fig. 2. Algorithm of SMKL

4 Experimental Result

In this section, we validate the usefulness of the proposed stacked generation on MKL (SMKL) with experimental evidence on datasets.

Table 1. Datasets details

Datasets	Table Column Head			
	dim	*n_pts*	*n_negative*	*n_positive*
Abalone(6,12)	8	526	259	267
Abalone(9,10)	8	1323	689	634
Balance-scale(1,3)	4	576	288	288
CMC(1,3)	9	1140	629	511
Glass(1,2)	9	146	70	76
Heart-statlog	13	270	150	120
Ionosphere	34	351	126	225
Liver-disorders	6	345	145	200
Monk3	6	122	62	60
Sonar	60	208	97	111
Tae(1,2)	5	99	49	50
Vehicle(1,2)	18	429	212	217

Table 2. Kernels details

1)Gaussian(2.1,1)	2)Polynomial(1)	3)Sigmoid(0.1)
4)Exponential(10,10)	5)Spherical(10)	6)Gaussian(20,1)
7)Circular(1)	8)Gaussian(100,10)	9)InverseMultiQuadric(1)
10)Gaussian(10,10)	11)T-Student(1)	12)Gaussian(1,1)
13)Linear	14)Spline	15)Chi-square(1)
16)RationalQuadratic(10)	17)Polynomial(10)	18)HistogramIntersection
19)ANOVA(1)	20)Distance(1)	21)Spherical(1)
22)Wavelet(1,1)	23)Sigmoid(0.01)	24)Polynomial(0.1)
25)Polynomial(2)	26)Gaussian(10,1)	27)Cauchy(1)
28)RationalQuadratic(1)	29)T-Student(0.1)	30)InverseMultiQuadric(10)

We use twelve real-world datasets from the UCI Repository of machine learning databases [7]. Details of these datasets are given in Table 1 while “dim” denotes feature number and “n_pts” denotes instance number. The numbers in the parentheses beside

the name of the datasets are the classes' numbers which are chosen from the original datasets which is multi-class. For example, Abalone (6, 12) denotes the class No. 6 and No. 12 which are chosen from the original Abalone dataset.

For the MKL method, Sonnenburg et al.'s algorithm [11] is used in our experiment so the multi-kernel method is modeled as a semi-infinite linear programming (SILP) problem for large scale MKL problem. To get robust performance on all kinds of datasets, 2-norm MKL algorithm [22] is used in our experiment. All experiments use 10 folds cross validation.

For stacked generalization and SMKL, we choose j=10 for the level_0 j-fold inner cross-validation process. 30 different kernels are chosen for MKL, SMKL and ensemble methods using single kernel SVMs. Table 2 gives the details of these kernels. The format of these kernels is kernel_name (gamma, cost). For all kernels the epsilon is 1e-5, and the coef0 is 0. For SMKL and Voting+MKL, we randomly divide the kernel set into a different number of groups and repeat this procession ten times for every cross validation to get the experimental result.

4.1 Comparing SMKL with Other Ensemble Methods

In this experiment, we first compare SMKL with all its base learners. Among the results, kernel number 18 (HistogramIntersection kernel) in the kernel list (TABLE II.) is the best of all base kernels using Nemenyi's post-hoc test [23] method. Then we compare SMKL with other ensemble methods. Table 3 gives the result. SVM(*best)* denotes using svm with the best kernel in our kernel list. Boosting_best denotes Adaboost [13] using SVM [3] with the kernel number 18 in our kernel list. Bagging_best denotes Bagging [12] using SVM [5] with kernel number 18 in our kernel list; Voting (30 kernels) denotes Majority voting on the 30 single kernel SVMs; SSVM (30 kernels) denotes stacked generalization on the 30 single kernel SVMs; MKL (30 kernels) denotes MKL using 30 kernels. Voting+MKL denotes dividing the kernel sets into different groups randomly and then using MKL (30 kernels) on each group to generate the base models; finally majority voting is used to combine the generated base models.

As Nemenyi's post-hoc test [23] is a non-parametric statistical test for multiple classifiers and multiple domains, we performed this test on the results in Table 3. We rank the accuracies for each domain with different classifiers using the following formula to calculate the q value between different classifiers [23].

$$q_{i,j} = \frac{\overline{R_i} - \overline{R_j}}{\sqrt{\frac{k(k+1)}{6n}}} \tag{2}$$

Where k is the number of classifiers and n is the number of datasets. The sums of the ranks of all tested classifiers are shown in Table 4. Therefore, we conclude that the difference between SMKL and other classifiers are significant and SMKL has better performance than all the other classifiers.

4.2 Experiments about Computational Complexity

In this experiment, we compare the running time of SMKL with MKL. m denotes the number of kernels and n denotes the number of instances. We report running time

Table 3. Experiment result of comparing SMKL with other methods

Datasets	methods name (accuracy is in percentage)							
	SVM (best)	*Adaboost _best*	*Bagging _best*	*Voting (30kernels)*	*MKL (30kernels)*	*SSVM (30kernels)*	*Voting +MKL*	*SMKL (30kernels)*
Abalone(6,12)	92.21±0.5	49.23±0.0	91.64±1.1	92.02±0.7	92.21±0.9	92.21±0.4	92.21±0.2	92.97±0.9
Abalone(9,10)	60.31±0.3	52.08±0.0	60.39±0.8	59.03±1.1	61.00±2.0	60.85±0.7	61.53±1.2	60.84±0.8
Balancescale	99.83±0.1	54.86±5.1	99.48±0.2	98.26±0.3	99.65±0.1	96.88±0.2	98.44±0.4	99.83±0.1
CMC	68.25±0.3	55.18±0.0	67.72±1.7	67.28±1.0	67.54±0.4	64.30±2.1	66.14±0.7	66.75±1.4
Glass	82.25±1.2	47.94±0.0	82.19±3.7	80.82±0.6	80.82±1.5	78.08±3.4	81.51±2.0	82.19±1.2
Heart-statlog	76.30±4.9	55.56±0.0	79.63±1.8	55.56±0.0	84.44±1.1	83.33±0.3	69.63±3.2	83.33±0.5
Ionosphere	92.59±1.3	76.64±6.9	92.88±0.9	94.87±0.7	94.30±0.6	94.30±1.4	94.87±0.5	96.01±0.7
Liver-disorders	69.86±1.2	42.03±0.0	67.83±4.8	59.71±3.0	72.17±1.0	73.62±0.9	70.43±2.3	71.30±0.4
Monk3	93.44±0.6	50.82±0.0	93.44±0.4	89.34±0.6	89.34±0.5	90.98±0.2	90.16±0.7	91.80±0.3
Sonar	87.02±1.4	87.02±1.1	87.02±0.5	83.17±0.9	87.50±0.1	87.02±0.2	86.06±0.9	87.50±0.3
Tae	72.73±0.4	49.49±0.0	74.75±1.8	74.75±2.7	71.72±3.2	74.75±2.2	76.77±1.8	77.78±0.3
Vehicle	61.07±1.1	61.07±0.5	64.59±0.2	50.58±5.7	64.57±0.4	63.00±1.0	49.65±1.9	65.73±0.4

Table 4. The rank of accuracies of all the datasets and classifiers (lower rank sum score is better)

Datasets	1 *SVM (best)*	2 *Adaboost _best*	3 *Bagging _best*	4 *Voting (30kernels)*	5 *MKL (30kernels)*	6 *SSVM (30kernels)*	7 *Voting +MKL*	8 *SMKL (30kernels)*
Abalone(6,12)	4	10	9	8	6	6	6	2.5
Abalone(9,10)	9	10	5	7	2	3	1	4
Balancescale	8	10	3	5	2	6	4	1
CMC	2.5	10	1	4	2.5	7	6	5
Glass	6	10	2.5	6	6	8	4	2.5
Heart-statlog	5	9.5	7	9.5	1	2.5	8	2.5
Ionosphere	8.5	10	6	2.5	4.5	4.5	2.5	1
Liver-disorders	5	10	6.5	9	2	1	4	3
Monk3	1.5	10	1.5	8	8	5	6	4
Sonar	9	4	4	7	1.5	4	6	1.5
Tae	9	10	4	4	6.5	4	2	1
Vehicle	6	5	2	9	3	4	10	1
Rank sums	71	107.5	48	75.5	40	51.5	58	27.5

results (Athlon ™ II X2 240 2.81G processor, 2.75G RAM) in Table 5. SMKL_p denotes SMKL using parallel computing on level_0.

Table 5 shows that even without using parallel computation on level_0, SMKL can process more instances with the same number of kernels or combine more kernels with the same number of instances than MKL. If we apply parallel computation on level_0, SMKL can get more efficiency on learning.

Table 5. Running times in seconds for SMKL and MKL. (Left) Ionosphere data with fixed number of data points n and varying number of kernels m; (Right) Ionosphere data with fixed number of kernels m and varying number of data points n.

Ionosphere, n=351			
m	*SMKL*	*SMKL_p*	*MKL*
6	62	50	33
12	170	142	147
24	562	423	738
48	1135	997	3393
96	3568	2781	*
192	*	*	*

Ionosphere, m=12			
n	*SMKL*	*SMKL_p*	*MKL*
351	170	142	147.5
702	432	291	459
1404	1497	988	1779
2808	9927	6519	19488
5616	36253	27436	*
11232	*	*	*

5 Discussions

5.1 Ensemble Methods Using SVMs as Base Learners

Although Evgeniou et al. [20] experimentally found that with accurate parameter tuning, single SVMs and ensembles of SVMs perform similarly, using ensemble methods or multiple kernel methods is always a more stable and robust methodology than using a single model. The statistical reason is that when we learn a model on the learning data, the resulting model can have more or less good predictive performance on these learning data. However, even if this performance is good, this does not guarantee good performance on the unseen data. Therefore, when learning single models, although there are evaluation techniques that minimize this risk, we can easily end up with a bad model. By taking into account several models and averaging their predictions, we can reduce the risk of selecting a very bad model.

In several real-world problems, SVM ensembles are reported to give improvements over single SVMs [17] [18], but a few works also showed negative experimental results about ensembles of SVMs [19].

The same situation occurs in our experiments. As shown in the experiment in section 4.1, stacking multiple SVMs, MKL and SMKL all show better predictive performance than all single SVMs while Voting+SVMs does not improve. Using the relative best single SVMs which is the kernel number 18 (in 30 kernels), bagging improves the predictive performance a little but boosting totally fails on this model.

We can give fundamental explanations to the above mentioned differences using the bias-variance analysis framework: the error of a learning algorithm can be divided into a part due to the functional form used by the algorithm (bias) and a part that is due to the instability of the algorithms (variance). Bagging, stacking and random forests reduce the variance part.

Boosting mainly reduces the bias part, but also reduces the variance portion. It can be viewed as an incremental forward stagewise regression procedure with regularization (Lasso penalty), which maximizes the margin between the two classes, much like the approach of support vector machines [21]. In boosting, when we increase the weights of examples that have not been correctly predicted by previous base models (or decrease the weights of the correctly predicted examples) and learn a new base model, the weight we increase may easily be above what the base learner needs because SVM is not a weak learner. Due to this, the complementary base models we try to generate by learning subsequent models may make the process hardly converge. In this situation, using SVM as the base learner for boosting may lead to worse performance, such as the result shown in our experiment. (Table 3 and Table 4)

Because of its simplicity, Voting allows for some theoretical analysis of its efficiency. However, since it requires neither cross-validation nor level-learning, unless level-0 generalizers perform comparably to one another, Voting cannot be comparable to other ensemble methods as shown in our experiments (Table 3 and Table 4)

Bagging reduces only the variance under the bootstrap assumption. On the other hand, SVMs can be strong, low-biased learners, but this property depends on the proper selection of the kernel and its parameters. If we can identify low-biased base

learners with a relatively high unbiased variance, bagging SVM can lower the error [17]. As shown in Table 3 and Table 4, kernel number 18 is selected as the best single SVM from the given kernel set, which means that we identify it as a low-biased base learner using cross-validation. Bagging SVM (kernel number 18) does improve the prediction performance as shown in Table 3 and Table 4.

5.2 The Reason Why SMKL Works

Although the idea of multiple kernel methods is similar to combining different classifiers, to be strict, MKL is not the same as traditional ensemble methods since it combines kernels instead of combining classifiers.

Giorgio et al. [17] analyze the bias-variance decomposition of error in SVMs. The analysis shows that the minimum of the overall error, bias, net-variance, unbiased and biased variance occurs often in different SVM models. These different behaviors of different SVM models could be in principle exploited to produce diversity in multiple kernel learning. As we have already mentioned, different kernels correspond to different notions of similarity of instances and may be using inputs coming from different representations. Thus by combining kernels, it is possible to combine multiple information sources and decrease the bias created by specific kernels. Moreover, by enforcing sparse coefficients, MKL also generalizes feature selection to kernel selection. These are the reasons why MKL is always more robust and has better predictive performance than single kernel classifier. By using MKL models as base learners for heterogeneous ensemble methods such as stacked generalization or voting, we can get the benefit of combining kernels. In addition, the diversity of different MKL models still exists. Since stacked generalization chooses level-1 generalizer to get the benefit of this diversity, using stacked generalization is a better choice than voting as the experiment result shows in Table 3 and Table 4 (compare Voting +MKL with SMKL).

The principle of how SMKL works is similar to Random forests [13], one of the most successful ensemble approaches. Random forests combine two sources of diversity of the base models: Variations in the learning data set (achieved through different bootstrap samples, as in bagging) and a randomized base-level learning algorithm. SMKL also combines multiple sources of diversity of the base models.

5.3 About the Computational Complexity

SMKL is ideal for parallel computation. The construction of each level-0 model proceeds independently; no communication with the other modeling processes is necessary. This feature makes SMKL applicable in the real world. Suppose the average computation time required for a MKL learning algorithm is t, SMKL requires g models and each model employs J-fold cross-validation. Assuming that time t is needed to derive each of the h level-0 models and the level-1 model, the learning time for SMKL is $T = (h(J + 1) + 1)t$. From Table 5 we find that the increase of running time of MKL is much higher than that of SMKL when we increase the number of kernels or instances. Since the experiment in Table 5 is set on one computer, SMKL does not benefit greatly from parallel computation. Higher efficiency can be expected if we implement SMKL on parallel computation with more processors.

6 Conclusions and Future Work

In this paper, we have introduced an approach which uses multiple kernel methods as base learners for stacked generalization process. The experimental results show that this method obtains the advantages from both the multiple kernel learning method and the stacked generalization method. Moreover, we have analyzed and compared many widely used ensemble methods which use kernel methods such as SVMs as base learners. Our future work includes implementation of this method for parallel computation with a large dataset and adopting more kernel functions, and testing the effect of kernel subsets arrangement in our method. Moreover, we will use further bias-variance decomposition analysis on this method.

References

1. Wolpert, D.H.: Stacked Generalization. Neural Networks 5, 241–259 (1992)
2. Kai, M.T., Ian Witten, H.: Issues in Stacked Generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
3. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the 21st International Conference on Machine Learning (2004)
4. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1999)
5. Scholkopf, B., Smola, A., Muller, K.: Kernel principal component analysis. Advances in Kernel Methods: Support Vector Learning, 327–352 (1999)
6. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
7. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2010), `http://archive.ics.uci.edu/ml`
8. Gehler, P.V., Nowozin, S.: Infinite kernel learning. Technical report, Max Planck Institute for Biological Cybernetics (2008)
9. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
10. Bach, F., Lanckriet, G., Jordan, M.: Multiple kernel learning, conic duality, and the smo algorithm. In: Proceedings of the 21st International Conference on Machine Learning, pp. 41–48 (2004)
11. Sonnenburg, S., Raetsch, G., Schaefer, C., Scholkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research 7, 1531–1565 (2006)
12. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
13. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
14. Schapire, R.E.: The strength of weak learnability. Machine Learning 5, 197–227 (1990)
15. Wolpert, D.H., Macready, W.G.: Combining stacking with bagging to improve a learning algorithm. Technical Report SFI-TR-96-03-123, Santa Fe Institute, Santa Fe, New Mexico (1996)
16. Kai, M.T., Witten, H.I.: Stacking Bagged and Dagged Models. In: ICML, pp. 367–375 (1997)

17. Valentini, G., Dietterich, T.G.: Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods. Journal of Machine Learning Research 5, 725–775 (2004)
18. Kim, H.C., Pang, S., Je, H.M., Kim, D., Bang, S.Y.: Pattern Classification Using Support Vector Machine Ensemble. In: Proceedings of the International Conference on Pattern Recognition, vol. 2, pp. 20160–20163. IEEE (2002)
19. Valentini, G., Dietterich, T.G.: Low Bias Bagged Support Vector Machines. In: Fawcett, T., Mishra, N. (eds.) Proceedings of the Twentieth International Conference, Machine Learning, pp. 752–759. AAAI Press, Washington (2003)
20. Evgeniou, T., Perez-Breva, L., Pontil, M., Poggio, T.: Bounds on the Generalization Performance of Kernel Machine Ensembles. In: Langley, P. (ed.) Proc. of the Seventeenth International Conference on Machine Learning, pp. 271–278. Morgan Kaufmann (2000)
21. Friedman, J.H., Hastie, T., Tibshirani, R.J.: Additive logistic regression: a statistical view of boosting. Technical report, Stanford University, Department of Statistics (1998)
22. Cortes, C., Mohri, M., Rostamizadeh, A.: L2 regularization for learning kernels. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (2009)
23. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press (2011)
24. Joachims, T., Cristianini, N., Shawe-Taylor, J.: Composite kernels for hypertext categorisation. In: Proceedings of the 18th International Conference on Machine Learning (2001)
25. Alham, N.K., Li, M., Liu, Y.: A distributed SVM ensemble for image: Classification and annotation. In: Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery, pp. 1581–1584. IEEE, Piscataway (2012)
26. Chang, E.Y., Zhu, K., Wang, H.: PSVM: Parallelizing support vector machines on distributed computers. Adv. Neural Inf. Process Syst. 20, 1–8 (2007)
27. Chen, Z.Y., Fan, Z.P.: Parallel multiple kernel learning: A hybrid alternating direction method of multipliers. Knowledge and Information Systems (2013)