Kernel Combination Versus Classifier Combination

Wan-Jui Lee[1], Sergey Verzakov[2], and Robert P.W. Duin[2]

[1] EE Department, National Sun Yat-Sen University, Kaohsiung, Taiwan
wrlee@water.ee.nsysu.edu.tw
[2] Information and Communication Theory Group, TU Delft, The Netherlands
serguei@edison.et.tudelft.nl, r.duin@ieee.org

Abstract. Combining classifiers is to join the strengths of different classifiers to improve the classification performance. Using rules to combine the outputs of different classifiers is the basic structure of classifier combination. Fusing models from different kernel machine classifiers is another strategy for combining models called kernel combination. Although classifier combination and kernel combination are very different strategies for combining classifier, they aim to reach the same goal by very similar fundamental concepts.

We propose here a compositional method for kernel combination. The new composed kernel matrix is an extension and union of the original kernel matrices. Generally, kernel combination approaches relied heavily on the training data and had to learn some weights to indicate the importance of each kernel. Our compositional method avoids learning any weight and the importance of the kernel functions are directly derived in the process of learning kernel machines. The performance of the proposed kernel combination procedure is illustrated by some experiments in comparison with classifier combining based on the same kernels.

1 Introduction

Traditional pattern recognition systems use a particular classification procedure to estimate the class of a given pattern. It has been observed that combining the decisions of different classifiers can be an efficient technique for improving the classification performance. If the combination function can take advantage of the strengths of the individual classifiers and avoid their weaknesses, the overall classification accuracy is expected to improve. Also, a larger stability for the classification system is highly anticipated. Many techniques have been proposed in last decade for combining classifiers [1].

A classifier combination system is usually composed of two phases, constructing individual classifiers and combining different classifiers. In the first phase, various models can be adopted to construct different classifiers, or the classifiers can be constructed on different features or from different sample datasets. In the second phase, the classifiers are combined by fixed or trained rules. This can be done on the basis of classifier outputs like posterior probabilities or using the crisp decisions (voting). Nevertheless, there are possibilities to combine the classifiers in an earlier stage in the classification system. In fact, the combination of models has attracted more and more attention recently, especially for kernel machines [2]. In kernel machine classifiers, different models can be built with different kernel functions. Combining models in kernel machine classifiers is thereby based on combining their kernels.

The support vector machine (SVM) [2,3], motivated by the results of statistical learning theory, is one of the most popular kernel machines. Most of the kernel combination research is based on it. In SVM, the decision boundary for pattern recognition problems is represented by a small subset of training examples, called support vectors. Unlike the traditional methods that minimize the empirical training errors, support vector machines implement the structural risk minimization principle. By adopting this principle, SVM can find the optimal discriminant hyperplane minimizing the risk of an erroneous classification of unseen test samples. When input data cannot be linearly separated in the original space, they should be mapped into a high dimensional feature space, where a linear decision surface separating the training data can be designed. The computation does not need to be performed in the feature space since SVM depends on the direct application of the kernel function over the input data. Therefore, the kernel function is a key component of SVM for solving nonlinear problems, and the performance of SVM classifiers largely depends on the choice of the kernels.

However, the selection of kernel functions, the model and the parameters, is one of the most difficult problem of designing a kernel machine. Recently, an interesting development seeks to construct a good kernel from a series of kernels. The most simple way to combine kernels is by averaging them. But not each kernel should receive the same weight in the decision process, and therefore the main force of the kernel combination study is to determine the optimal weight for each kernel. The criterion for searching these weights is mainly based on Fisher's discriminant that maximizes the ratio of the between-class variance and the within-class variance. Optimization methods and heuristic approaches are also used for obtaining the weights. In [5], the weights of kernels are derived by optimizing the measure of data separation in the feature space by the semidefinite programming. Kernel target alignment is used to match the kernels with the data labels. Using boosting, a weighted combination of base kernels is generated in [4]. Some methods [6,7,8,9] try to find the best weights by maximizing a class separability criterion. All the approaches above rely heavily on the training data and some weights have to be learned before the combination of kernels to indicate the importance of each kernel.

In order to avoid learning any weight, we propose a compositional method for kernel combination. In this method, the new kernel matrix is composed of the original, different kernel matrices, by constructing a larger matrix in which the original ones are still present. Thereby, the properties of the original kernel functions can be preserved and the importance of these kernel functions are directly derived in the process of training support vector machines. Herewith, weights for individual objects with respect to the base kernels are found integrated in a single classifier optimization procedure. This procedure will thereby not overfit the training dataset as the weighted kernels methods may do due to the fact that they use the data twice: for optimising the weights as well as for training the classifier.

In this paper we experimentally study the differences of our kernel compositional method with other kernel combination methods, and the differences and influences of combining models and combining decisions for a classifier combination system. Some considerations for selecting more suitable strategies under different situations will be discussed.

The rest of the paper is organized as follows. In Section 2, some background of support vector machine is recapitulated. The construction of the discriminant hyperplane in the feature space and the effect of kernel functions is shown. In Section 3, our kernel composition method is presented. Simulation results for comparing kernel combination and classifier combination methods are given in Section 4. Finally, conclusions are summarized in Section 5.

2 Overview of Support Vector Machine

For convenience, we introduce the support vector classifier with d input variables x_{i1}, x_{i2}, ..., x_{id} for 2-class problem with class labels $+1$ and -1 in this section. $\mathbf{x}_i$ and y_i represent i^{th} input datum (a vector) and its corresponding class label [2,3]. Extension to multi-class problems can be achieved by training multiple support vector machines.

2.1 Support Vector Machine

To control both training error and model complexity, the optimization problem of SVM is formalized as follows:

$$\begin{aligned} \text{minimize}\ & \frac{1}{2} < \mathbf{w}, \mathbf{w} > +C \sum_{i=1}^{n} \xi_i, \\ \text{subject to}\ & < \mathbf{w} \cdot \mathbf{x}_i > +b \geq +1 - \xi_i,\ \text{for}\ y_i = +1 \\ & < \mathbf{w} \cdot \mathbf{x}_i > +b \leq -1 + \xi_i,\ \text{for}\ y_i = -1 \\ & \xi_i \geq 0, \forall i. \end{aligned} \tag{1}$$

By using Lagrange multiplier techniques, Eq.(1) could lead to the following dual optimization problem:

$$\begin{aligned} \text{maximize}\ & \textstyle\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j >, \\ \text{subject to}\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \in [0, C]. \end{aligned} \tag{2}$$

Using Lagrange multipliers, the optimal desired weight vector of the discriminant hyperplane is $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$. Therefore the best discriminant hyperplane can be derived as

$$f(\mathbf{x}) =< \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i, \mathbf{x} > +b = (\sum_{i=1}^{n} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} >) + b, \tag{3}$$

where b is the bias of the discriminant hyperplane.

2.2 Kernel Functions

In Eq.(3), the only way in which the data appears is in the form of dot products $< \mathbf{x}_i, \mathbf{x} >$. The discriminant hyperplane is thereby linear and can only solve a linearly separable classification problem. If the problem is nonlinear, instead of trying to fit a nonlinear model, the problem can be mapped to a new space by a nonlinear transformation using a suitably chosen kernel function. The linear model used in the new space corresponds to a nonlinear model in the original space. To make the above model nonlinear, consider a mapping $\phi(\mathbf{x})$ from the input space into some feature space as

$$\phi : \mathbb{R}^d \rightarrow \mathcal{H}. \tag{4}$$

The training algorithm only depends on the data through dot products in $\mathcal{H}$, i.e. on functions of the form $< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >$. Suppose a kernel function K defined by

$$K(\mathbf{x}_i, \mathbf{x}_j) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >, \tag{5}$$

is used in the training algorithm. Explicit knowledge of ϕ is thereby avoided. The dot product in the feature space can be expressed as a kernel function. Similar to Eq.(3) in linear problems, for a nonlinear problem, we will have the following discriminant function

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b. \tag{6}$$

In this paper, we will use the Gaussian radial basis function as the kernel function, and therefore

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{(-\frac{\|x_i - x_j\|^2}{\sigma^2})}. \tag{7}$$

3 Composition of Kernel Matrices

Most kernel combination methods try to average out the kernel matrices in one way or another [5,4,6,7,8,9]. There is a risk, however, of losing information in the original kernel matrices. For example, if the dataset has varying local distributions, different kernels will be good for different areas. Averaging the kernel functions of such a dataset would lose some capability to describe these local distributions. In order to combine kernel matrices without losing any original information, we develop a kernel composition method which is an extension and aggregation of all the original kernel matrices.

Suppose the original kernel functions are $K_1, K_2, ...,$ and K_s and the feature functions of the original kernel functions are $\phi_1(\mathbf{x}), \phi_2(\mathbf{x})...,$ and $\phi_s(\mathbf{x})$. We would like to preserve and use all the feature functions to construct a new kernel function, so we should be able to compute inner products like $< \phi_p(\mathbf{x}), \phi_{p'}(\mathbf{x}') >$, where $\phi_p(\mathbf{x})$ and $\phi_{p'}(\mathbf{x}')$ are feature functions from different kernel spaces. We will show that this can be done if we can formulate $\phi_{\mathbf{p}}(\mathbf{x})$ as ${K_p}^{\frac{1}{2}}(\mathbf{x}, \mathbf{z})$ which is a function of $\mathbf{z}$ and belongs to the L_2 space. [1] Using the definition of inner products in the L_2 space, we define the compositional kernel function as

$$K_{p,p'}(\mathbf{x}, \mathbf{x}') \equiv< \phi_p(\mathbf{x}), \phi_{p'}(\mathbf{x}') >\equiv \int {K_p}^{\frac{1}{2}}(\mathbf{x}, \mathbf{z}) {K_{p'}}^{\frac{1}{2}}(\mathbf{x}', \mathbf{z}) d\mathbf{z}. \tag{8}$$

[1] ${K_p}^{\frac{1}{2}}(\mathbf{x}, \mathbf{z})$ can be computed based on eigenvalue decomposition: it has the same eigenfunctions as $K_p(\mathbf{x}, \mathbf{z})$ and its eigenvalues are the square roots of those of $K_p(\mathbf{x}, \mathbf{z})$.

Using the self-similarity property of Gaussian distributions, one can show that the square root of a radial basis kernel function is

$$K_p^{\frac{1}{2}}(\mathbf{x},\mathbf{z}) = (\frac{4}{\pi{\sigma_p}^2})^{\frac{d}{4}} e^{(-2\frac{\|\mathbf{x}-\mathbf{z}\|^2}{{\sigma_p}^2})}, \tag{9}$$

and the mixture of two kernel matrices can be derived as

$$K_{p,p'}(\mathbf{x},\mathbf{x}') = (\frac{2\sigma_p\sigma_{p'}}{{\sigma_p}^2+{\sigma_{p'}}^2})^{\frac{d}{2}} e^{(-2\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{{\sigma_p}^2+{\sigma_{p'}}^2})}. \tag{10}$$

Clearly, $K_{p,p} \equiv K_p$. Consequently, the compositional kernel matrix K is of the form

$$K = \begin{pmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,s} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ K_{s,1} & K_{s,1} & \cdots & K_{s,s} \end{pmatrix}_{s\times n, s\times n}, \tag{11}$$

where the original kernel matrices are on the diagonal. The other elements are mixtures of two different kernel matrices which are defined like $(K_{p,p'})_{i,j} = K_{p,p'}(\mathbf{x}_i,\mathbf{x}_j)$. It is obvious that entries of K are the inner products in L_2, and because of this K is positive semi-definite. The kernel matrix K is called the compositional kernel matrix. Also, the feature function $\phi(\mathbf{x})$ of the compositional kernel matrix can be defined as $\phi(\mathbf{x}) = [\phi_\mathbf{1}(\mathbf{x}), \phi_\mathbf{2}(\mathbf{x}), ..., \phi_\mathbf{s}(\mathbf{x})]$. Note that the size of the compositional kernel matrix is $(s\times n)\times(s\times n)$ while the sizes of the original kernel matrices are $n\times n$. After the construction of the compositional kernel matrix, the support vector machine can proceed the learning of support vectors and their corresponding coefficients. Objects have to be replicated as the compositional kernel matrix is s times larger than the base kernels.

With the compositional kernel matrix, we can reformulate the optimization problem as follows:

$$\begin{aligned} \text{minimize } & \frac{1}{2} < \mathbf{w},\mathbf{w} > + C\sum_{i=1}^{s\times n}\xi_i, \\ \text{subject to } & < \mathbf{w}\cdot\mathbf{x}_i > + b \geq +1-\xi_i, \text{ for } y_i=+1 \\ & < \mathbf{w}\cdot\mathbf{x}_i > + b \leq -1+\xi_i, \text{ for } y_i=-1 \\ & \xi_i \geq 0, \forall i. \end{aligned} \tag{12}$$

By using Lagrange multiplier techniques, Eq.(12) could lead to the following dual optimization problem:

$$\begin{aligned} \text{maximize } & \textstyle\sum_{i=1}^{s\times n}\alpha_i - (\sum_{i=1}^{n}\alpha_i y_i\phi_1(\mathbf{x}_i)+\cdots+\sum_{i=(s-1)\times n+1}^{s\times n}\alpha_i y_i\phi_s(\mathbf{x}_i)) \\ & \textstyle(\sum_{j=1}^{n}\alpha_j y_j\phi_1(\mathbf{x}_j)+\cdots+\sum_{j=(s-1)\times n+1}^{s\times n}\alpha_j y_j\phi_s(\mathbf{x}_j)), \\ \text{subject to } & \textstyle\sum_{i=1}^{s\times n}\alpha_i y_i = 0, \alpha_i\in[0,C]. \end{aligned} \tag{13}$$

After calculating Lagrange multipliers, an optimal weight vector for the discriminant hyperplane can be found by $\mathbf{w} = \sum_{i=1}^{s\times n} \alpha_i y_i \mathbf{x}_i$. Therefore the best discriminant hyperplane can be derived as

$$\begin{aligned} f(\mathbf{x}) = \quad & (\sum_{i=1}^{n} \alpha_i y_i \phi_1(\mathbf{x}_i) + \cdots + \sum_{i=(s-1)\times n+1}^{s\times n} \alpha_i y_i \phi_s(\mathbf{x}_i))\phi(\mathbf{x}) + s \times b \\ = \quad & \sum_{i=1}^{n} \alpha_i y_i (K_{1,1}(\mathbf{x}_i, \mathbf{x}) + K_{1,2}(\mathbf{x}_i, \mathbf{x}) + \cdots + K_{1,s}(\mathbf{x}_i, \mathbf{x})) + \cdots \\ + \quad & \sum_{i=n\times(s-1)+1}^{s\times n} \alpha_i y_i (K_{s,1}(\mathbf{x}_i, \mathbf{x}) + K_{s,2}(\mathbf{x}_i, \mathbf{x}) + \cdots + K_{s,s}(\mathbf{x}_i, \mathbf{x})) \\ + \; & s \times b \end{aligned} \tag{14}$$

where $s \times b$ is the bias of the discriminant hyperplane and $\mathbf{x}$ can be either a training or testing data pattern.

4 Experimental Results

In this section, we compare the experimental results obtained by our composition kernel combination method with those of another kernel combination and a classifier combination method. The kernel combination method is the weighted kernel for which the weights of the kernels are optimized by semidefinite programming [5]. The product rule is used to derive the classifier combiner. One synthetic dataset and three benchmark datasets [12] are used in the experiments. To test whether kernel combination methods are more capable of describing data with different local distributions than classifier combination methods, two of the four datasets used in the experiments are with different local distributions, and the other two datasets are regular real datasets. The single kernel and the combined kernel SVM classifiers in the experiments are implemented by LIBSVM [10] and the classifier combiners are built with the PRTOOLS [11]. In every experiment, several single kernel SVM classifiers are constructed, and kernel combination and classifier combination methods were used to combine these single classifiers. The sigma's of these single RBF kernels are assigned heuristically in the following way. The smallest sigma is the average distance of each data pattern to its nearest neighbor. The largest sigma is the average distance of each data pattern to its furthest neighbor. The other sigma's are determined by linear interpolation.

4.1 Experiment 1: Data with Varying Local Distributions

Banana and sonar datasets are used in experiment 1. The SVM parameter C is set to 1 in all experiments. The banana dataset is a synthetic 2-dimensional dataset with 400 data patterns in 2 classes, and it is rescaled in each dimension with

$$x_{ij} = x_{ij} \times e^{(-\frac{x_{ij}}{16})}, \text{ for } i = 1, 2, ..., 400, \text{ and } j = 1, 2, \tag{15}$$

to have a variation of scales in the dataset. The sonar dataset contains information of 208 objects, 60 attributes, and two classes, rock and mine. The attributes represent the energy within a particular frequency band integrated over a certain period of time. The results are averaged over 20 experiments. For the rescaled banana dataset, 2 single kernel classifiers are built and different methods are used to combine these 2 classifiers in each experiment. As for the sonar dataset, 4 single kernel classifiers are built and combined in the experiments.

The results for all single kernel classifiers, kernel combination methods and classifier combination methods with rescaled banana dataset are in Figure 1. Moreover, the

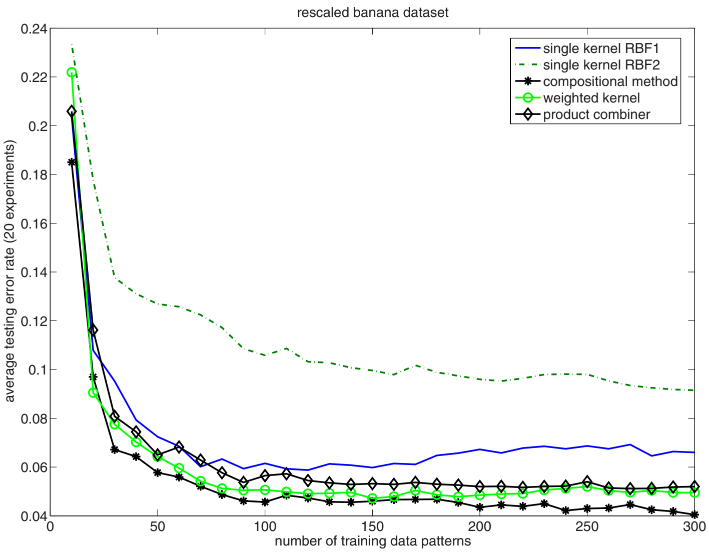

Fig. 1. Experiment results of rescaled banana dataset with our compositional method, weighted kernel, product combiner and their comparisons with the single kernel classifiers. Standard deviations of the mean error results vary from 0.1112 (left) to 0.0088 (right).

experimental results of the sonar dataset are given in Figure 2. From Figure 1 and Figure 2, we can see that the compositional method performs better than the weighted kernel, especially when a smaller size of training dataset is given. This is because it avoids the overtraining problem of the weighted kernel. Also, kernel combination methods outperform classifier combination methods when the dataset is composed of different local distributions. A possible reason is that a combined model is more capable of adapting to varying data distributions than can be realized by combining decision outputs.

4.2 Experiment 2: Benchmark Data

We use the glass and diabetes datasets in experiment 2. The glass dataset contains 214 instances, 9 features and 6 classes, and diabetes dataset is with 768 instances, 8 features and 2 classes. The SVM parameter C is set to 100 in all experiments. For both datasets, 4 single kernel classifiers are built and different methods are used to combine these 4 classifiers in each experiment, and the results are the averages of 20

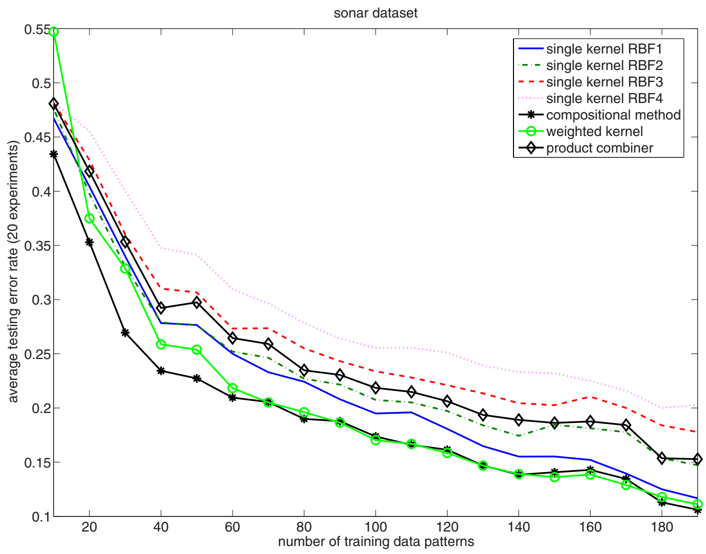

Fig. 2. Results of the sonar dataset for the proposed compositional method, weighted kernel, product combiner and their comparisons with the single kernel base classifiers. Standard deviations of the mean error results vary from 0.086 (left) to 0.038 (right).

Table 1. Diabetes dataset: number of support vectors generated with weighted kernel and our compositional method

method	number of training data patterns						
	10	50	100	150	200	250	300
	number of average support vectors (20 experiments)						
weighted kernel	9.7	40.3	65.7	88.2	111.4	133.4	153
compositional method	18.3	97.2	179.5	272.5	369.8	466.5	553.4

repeated experiments. The results of all single kernel classifiers, kernel combination methods and classifier combination methods with the glass dataset are shown in Figure 3. The results for the diabetes dataset are given in Figure 4, and the number of support vectors obtained with the kernel combination methods are given in Table 1. In Figure 3 and Figure 4, kernel combination methods and classier combination methods have similar performances if the number of training objects is large. When the size of the training set is small, kernel combination methods suffer more from overfitting, and therefore classier combination methods would be a better choice. Nevertheless, the compositional method performs better than all other methods. The number of support vectors, however, is about three times of those of the other kernel combination methods. This is related to the replication of the training set needed to apply the larger kernel matrix.

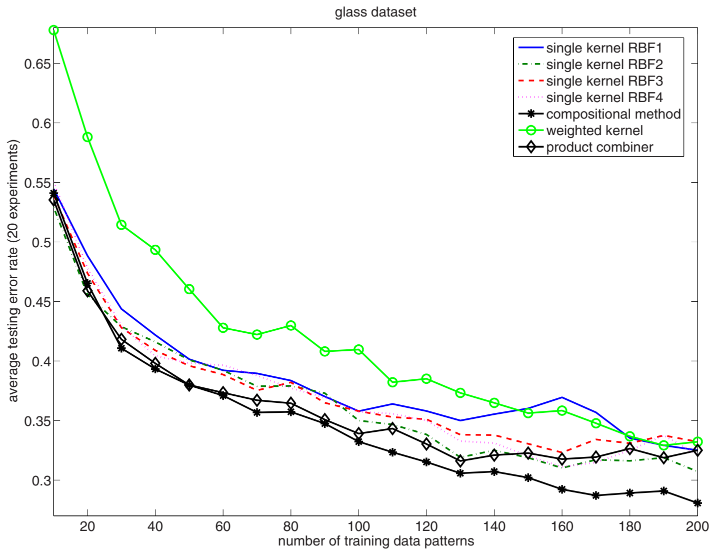

Fig. 3. Results for the glass dataset with the compositional method, weighted kernel, product combiner and their comparisons with the single kernel classifiers. Standard deviations of the mean error results vary from 0.13 (left) to 0.034 (right).

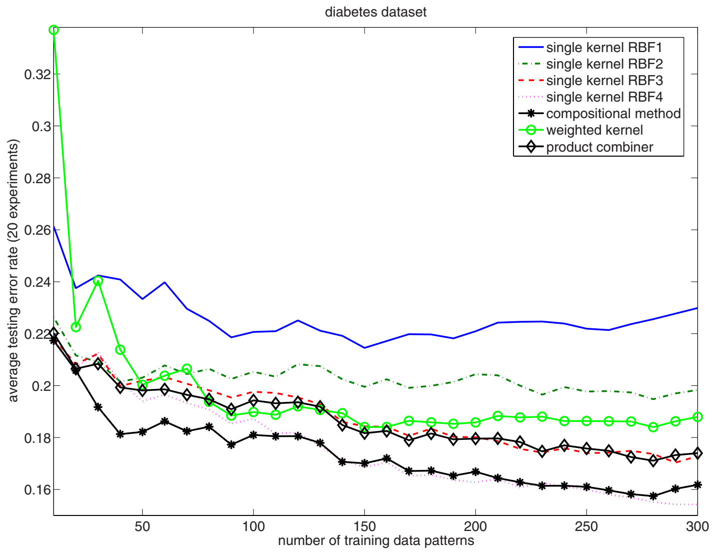

Fig. 4. Results of the diabetes dataset with the compositional method, weighted kernel, product combiner and their comparisons with the single kernel classifiers. Standard deviations of the mean error results vary from 0.092 (left) to 0.016 (right).

5 Conclusions

In this study we compared the performances of kernel combination and classifier combination methods for different types of data distribution. We also proposed a compositional kernel combination method to avoid the overfitting problem of the other kernel combination methods. When a dataset has a varying local data distributions, kernel combination methods are preferred. But classifier combination methods are more stable when the size of the training dataset is small. Nevertheless, the proposed compositional method is stable in all cases. If there is, however, a superior single kernel classifier, it will be very difficult to obtain a better classifier by classifier or kernel combining.

References

1. L. I. Kuncheva, *Combining Pattern Classifiers. Methods and Algorithms*, Wiley, 2004.
2. V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer Verlag, 1995.
3. C. J. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” *Knowledge Discovery and Data Mining*, vol. 2, no. 2, pp. 1-43, 1998.
4. K. P. Bennett, M. Momma, and M. J. Embrechts, “MARK: A boosting algorithm for heterogeneous kernel models,” in *Proc. 8th ACMSIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 24-31, 2002.
5. 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. 5, pp. 27-72, 2004.
6. G. Fung, M. Dundar, J. Bi, and B. Rao, “A fast iterative algorithm for fisher discriminant using heterogeneous kernels,” in *Proc. 21st Int. Conf. Machine Learning* , 2004.
7. I. M. de Diego, J. M. Moguerza, and A. Muoz, “Combining Kernel Information for Support Vector Classification,” in *Proc. Multiple Classifier Systems*, pp. 102-111, 2004.
8. J. M. Moguerza, A. Muoz, and I. M. de Diego, “Improving Support Vector Classification via the Combination of Multiple Sources of Information,” SSPR/SPR, pp. 592-600, 2004.
9. C. S. Ong, A. J. Smola, and R. C. Williamson, “Learning the kernel with hyperkernels,” *Journal of Machine Learning Research*, pp. 1043-1071 , 2005.
10. C. C. Chang and C. J. Lin, *LIBSVM : a library for support vector machines*, [http://www.csie.ntu.edu.tw/ cjlin/libsvm], Taiwan, National Taiwan University, 2001.
11. R. P. W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. M. J. Tax, *PRTools4, A Matlab Toolbox for Pattern Recognition*, [http://www.prtools.org], the Netherlands, Delft University of Technology, ICT Group, 2004.
12. D. J. Newman, S. Hettich, C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases* , [http://www.ics.uci.edu/ mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science, 1998.