Investigating the Performance of a Linear Regression Combiner on Multi-class Data Sets*

Chun-Xia Zhang[1,2] Robert P.W. Duin[2]

[1]School of Science and State Key Laboratory for Manufacturing Systems Engineering,
Xi'an Jiaotong University, Xi'an Shaanxi 710049, China
[2]Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
cxzhang@mail.xjtu.edu.cn r.duin@ieee.org

Keywords: Ensemble classifier; Multi-response linear regression (*MLR*); Trainable combiner; Decision template (*DT*); Fisher linear discriminant (*FLD*)

Abstract

*Although Multi-response Linear Regression (*MLR*) has been proposed as a trainable combiner to fuse heterogeneous base-level classifiers into an ensemble classifier, thus far it has not yet been evaluated extensively. In this paper, we employ learning curves to investigate the relative performance of* MLR *for solving multi-class classification problems in comparison with some other combiners. Meanwhile, several strategies (namely,* Reusing, Validation *and* Stacking*) are considered for using the available data to train both the base-level classifiers and the combiner. The experimental results show that due to the limited complexity of* MLR*, it can outperform the other combiners for small sample sizes when the* Validation *or* Stacking *strategy is adopted. Therefore,* MLR *should be a preferential choice of trainable combiners when solving a multi-class task with small sample size.*

1 Introduction

Classifier combination strategies, often termed as ensemble classifiers, currently have received much attention in pattern recognition and machine learning communities due to their potential to significantly improve the generalization capability of a learning system. These techniques have been proved to be quite versatile in a broad range of real applications such as remote sensing data processing, gene expression data analysis, face recognition and so on [1–3].

In general, the task of constructing an ensemble classifier [4, 5] can be broken into two steps: generation of multiple base-level classifiers and combination of their outputs. Up to now, many approaches for creating a diverse set of base-level classifiers as well as for fusing their predictions have been proposed [6–15]. In the present study, we will mainly study the case of employing trainable combiners to combine the predictions of heterogeneous base-level classifiers that are obtained by applying some different learning algorithms to the same data set [5, 9–15].

With respect to trainable combiners, we are faced with the following problem: how to utilize the available data to train the models for both levels, the base-level classifiers as well as the combiner? Thus far, three strategies (that is, *Reusing*, *Validation* and *Stacking*) [16] have been proposed and they will be briefly described in Section 2.

In recent years, multi-response linear regression

*An extended version of the paper has been accepted by MCS2009 Workshop

(*MLR*) has been recommended as a trainable combiner for merging heterogeneous base-level classifiers. So far, there have been some variants of it [9, 10, 12, 13, 17] and the approach proposed in [10] may be the prominent one which has been shown to be effective for handling multi-class problems [12]. To the best of our knowledge, however, the previous researchers only considered the situation that the training set size is supposed to be fixed and the *Stacking* method is employed to construct its meta-level data (namely, the data for training the combiner).

In order to evaluate the trainable combiner *MLR* extensively, in this paper we employ learning curves to investigate the relative performance of *MLR* for solving multi-class classification problems in comparison with other combiners *FLD* (Fisher Linear Discriminant), *DT* (Decision Template) and *MEAN*. At the same time, several strategies are considered for using the available data to train the base-level classifiers and the combiner. The experimental results show that for small sample sizes, *MLR* can generally outperform the other combiners when the *Validation* or *Stacking* strategy is adopted. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data.

The remainder of the paper is organized as follows. Section 2 briefly introduces the working mechanism of *MLR* and some feasible strategies to utilize the given data to derive both base-level classifiers and combiner. In Section 3, the experimental results obtained on some multi-class data sets are presented and discussed. Finally, the main conclusions are summarized in Section 4.

2 *MLR* and Strategies to Use the Training Data

2.1 Working mechanism of *MLR*

Denote by $\mathscr{L} = \{(y_n, \mathbf{x}_n)\}_{n=1}^{N}$ a given set consisting of N examples, where y_n is a class label taking value from $\Phi = \{\omega_1, \omega_2, \cdots, \omega_m\}$ and $\mathbf{x}_n \in R^d$ is the feature vector of the nth example. Suppose that a set $\mathscr{C} = \{C_1, C_2, \cdots, C_L\}$ of L base-level classifiers is generated by applying the heterogeneous learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$. When applying $C_i (i = 1, 2, \cdots, L)$ to classify an example $\mathbf{x}$, the output of it is assumed to be a probability distribution vector

$$\mathbf{P}_{C_i}(\mathbf{x}) = (P_{C_i}(\omega_1|\mathbf{x}), P_{C_i}(\omega_2|\mathbf{x}), \cdots, P_{C_i}(\omega_m|\mathbf{x}))^T, \tag{1}$$

where $P_{C_i}(\omega_j|\mathbf{x})$ indicates the probability that $\mathbf{x}$ belongs to class ω_j as estimated by the classifier C_i. To simplify the notations, we abbreviate the above formula as

$$\mathbf{P}_i(\mathbf{x}) = (P_1^i(\mathbf{x}), P_2^i(\mathbf{x}), \cdots, P_m^i(\mathbf{x}))^T,\ i = 1, 2, \cdots, L.$$

Furthermore, we define $\mathbf{P}(\mathbf{x})$ as an mL-dimensional column vector, namely,

$$\mathbf{P}(\mathbf{x}) = (\mathbf{P}_1^T(\mathbf{x}), \mathbf{P}_2^T(\mathbf{x}), \cdots, \mathbf{P}_L^T(\mathbf{x}))^T. \tag{2}$$

Based on the intermediate feature space constituted by the outputs of each base-level classifier, the *MLR* method [10] firstly transforms the original classification task with m classes into m regression problems: the problem for class ω_j has examples with responses equal to one when they indeed have class label ω_j and zero otherwise. For each class ω_j, *MLR* constructs a linear model through selecting only the probabilities $P_j^1(\mathbf{x}), P_j^2(\mathbf{x}), \cdots, P_j^L(\mathbf{x})$ as the prediction variables,

$$LR_j(\mathbf{x}) = \sum_{i=1}^{L} \alpha_j^i P_j^i(\mathbf{x}),\ j = 1, 2, \cdots, m, \tag{3}$$

where the coefficients $\{\alpha_j^i\}_{i=1}^{L}$ are constrained to be non-negative and the non-negative-coefficient least-squares algorithm described in [18] is exploited to estimate them. When classifying a new example $\mathbf{x}$, all the values of $LR_j(\mathbf{x})$ for each of the m classes are computed and $\mathbf{x}$ is then assigned to the class ω_k that has the greatest value.

2.2 Strategies to use the training data

In order to train the combiner *MLR*, that is, to estimate the coefficients $\{\alpha_j^i\}_{i=1}^{L}$ in formula (3), we must construct the meta-level data firstly. In practical applications, however, we are only given a single set $\mathscr{L}$ which should be used to train the base-level classifiers as well as the combiner. In this situation, there are mainly three feasible approaches – *Reusing*, *Validation* and *Stacking* – to make full use of $\mathscr{L}$ to form an ensemble classifier.

The *Reusing* strategy simply applies the given learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ to $\mathscr{L}$ to train the base-level classifiers $C_1, C_2, \cdots, C_L$ which are then used to predict the objects in $\mathscr{L}$ to form the meta-level data. Since the same set $\mathscr{L}$ is utilized twice to derive both base-level classifiers and combiner, the obtained combiner will inevitably be biased.

The *Validation* strategy splits the training set $\mathscr{L}$ into two disjoint subsets, one of which is for deriving the base-level classifiers $C_1, C_2, \cdots, C_L$ and the other one is for constructing the meta-level data.

As for the *Stacking* strategy, it takes advantage of the cross-validation method to form the meta-level data. Firstly, the training set $\mathscr{L}$ is partitioned into K disjoint subsets $\mathscr{L}_1, \mathscr{L}_2, \cdots, \mathscr{L}_K$ with each of them having almost equal size and roughly preserving the class probability distribution in $\mathscr{L}$. In order to obtain the base-level predictions on examples in $\mathscr{L}_k$, say, $\mathscr{L}'_k = \{(y_i, \mathbf{P}^T(\mathbf{x}_i)) | (y_i, \mathbf{x}_i) \in \mathscr{L}_k\}$, the learning algorithms $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_L$ are applied to the set $\mathscr{L} \backslash \mathscr{L}_k = \{\mathscr{L}_1, \cdots, \mathscr{L}_{k-1}, \mathscr{L}_{k+1}, \cdots, \mathscr{L}_K\}$ to derive the classifiers $\{C_{j,k}\}_{j=1}^{L}$ which are then used to predict the examples in $\mathscr{L}_k$. After repeating this process K times (once for each set $\mathscr{L}_k$), we can obtain the final meta-level data set $\mathscr{L}^{CV} = \bigcup_{k=1}^{K} \mathscr{L}'_k$. The readers can refer to [5, 9, 10, 16] for more details about this technique.

It is worthwhile to mention that for a new example $\mathbf{x}$, there may be two different ways to construct the input $\mathbf{P}(\mathbf{x})$ of the combiner trained by the *Stacking* method. The *Stacking I method*, commonly utilized by many researchers [9, 10, 12, 13, 19], is to retrain the base-level classifiers $C_1, C_2, \cdots, C_L$ on the full training set $\mathscr{L}$ to produce $\mathbf{P}(\mathbf{x})$. The *Stacking II method*, proposed in [16], applies all the classifiers $\{C_{j,k}\}_{j=1}^{L}{}_{k=1}^{K}$ which are trained in the process of cross-validation to predict $\mathbf{x}$. For each base-level classifier, it firstly averages the predictions that are obtained in each fold, namely, $\bar{C}_j(\mathbf{x}) = (1/K) \sum_{k=1}^{K} C_{j,k}(\mathbf{x})$. Then, the input of the trained combiner is formed as $\mathbf{P}(\mathbf{x}) = (\bar{C}_1^T(\mathbf{x}), \bar{C}_2^T(\mathbf{x}), \cdots, \bar{C}_L^T(\mathbf{x}))^T$ where $\bar{C}_i^T(\mathbf{x})$ $(i = 1, 2, \cdots, L)$ is an $m \times 1$ vector.

3 Experimental Studies

Since learning curves can give a good picture to study the performance of an algorithm at various sample sizes, in this section we employ them to investigate the relative performance of *MLR* for solving multi-class classification tasks in comparison with several other combiners. Furthermore, each of the different strategies described in subsection 2.2 will be considered here to utilize the given set to train the base-level classifiers and the combiner.

3.1 Experimental settings

We conducted experiments on a collection of 10 multi-class data sets from the UCI repository [20]. The data sets and some of their characteristics (number of examples, input attributes, classes and the considered training and test sizes per class) are summarized in Table 1. This selection includes data sets from a variety of fields and each set contains approximately equal number of objects per class.

Table 1. Summary of the data sets used in the experiments.

Data set	# Examples	# Input Attributes	# Classes	Training size (Per class)	Test size (Per class)
Abalone	4177	10	3	5,10,20,30,50,80,100	500
Cbands	12000	30	24	10,20,30,50,80,100	100
Digits	2000	240	10	5,10,20,30,50,80,100	50
Letter	20000	16	26	10,20,30,50,80,100	200
Pendigits	7494/3498	16	10	5,10,20,30,50,80,100	3498(total)
Satellite	6435	36	6	5,10,20,30,50,80,100	500
Segmentation	2310	19	7	5,10,20,30,50,80,100	100
Vehicle	846	18	4	5,10,20,30,50,80,100	50
Vowelc	990	12	11	10,20,30,50	30
Waveform	5000	21	3	5,10,20,30,50,80,100	500

Here, we totally considered 4 different types of base-level classifiers and 4 different combiners which were selected as follows. These classifiers and combiners were implemented based on our own codes together with routines available in version 4.1.4 of PRTools [21], a Matlab Toolbox specialized for sta-

tistical pattern recognition.

Classifiers
- Fisher linear discriminant (*fisherc*),
- Parzen density classifier (*parzenc*),
- Nearest neighbor (*knnc*),
- Logistic linear classifier (*loglc*);

Combiners
- Multi-response linear regression (*MLR*),
- Decision template (*DT*),
- Fisher linear discriminant (*FLD*),
- *MEAN*.

In order to constitute the intermediate feature space in a homogeneous way, the outputs of each base-level classifier were scaled to fall into a [0,1] interval. Furthermore, the parameters included into the base-level classifiers and the combiners were all taken to be their default values in PRTools. In the process of employing each strategy described in subsection 2.2 to utilize the available data to train base-level classifiers and combiners, the *Validation* method uses a $50\%/50\%$ split and the two *Stacking* methods employ 10-fold cross-validation to form the meta-level data.

To see clearly the behavior of different combiners at various sample sizes, we estimated the learning curves in the following way. For each data set (except for "Pendigits") listed in Table 1, a training set and an independent test set with desired sizes were randomly sampled. As for the "Pendigits" data set which has separate training and testing data, all of its testing data were used as the test set. On each of the obtained training set, the base-level classifiers and the combiner were constructed according to each of the strategies described in subsection 2.2. The trained combiner was then executed on the test set and the estimated classification error was taken as a measure to evaluate its performance. It should be noted that all steps required for building the base-level classifiers and the combiner, including the cross-validation procedure utilized by the *Stacking* method, were performed on the training set only. For each obtained training set size, the above process was repeated 10 times and the obtained results were averaged.

3.2 Results and discussion

For each combiner, we plotted the mean of the test errors over 10 replications as a function of the sample size. Meanwhile, the standard deviations were also shown to compare the performance of each combiner more detailedly. Due to the limited space of this paper, the learning curves were only given for "Cbands", "Digits" and "Satellite" data sets (Figs. 1-3) out of the 10 investigated ones and the plots displayed here were representative of all the cases studied. Additionally, the remaining plots are available from the authors. With regard to each data set, the scales of the axes for the plots which were obtained by each strategy to use the given data (except for the *Reusing* method on "Digits") have been adjusted to be identical to facilitate the comparisons. From these plots, the following observations can be made:

- For small sample sizes, *MLR* can generally outperform the other combiners when the *Validation* or *Stacking* method is adopted. However, the superiority of *MLR* is not very obvious when the classification task has many classes ("Cbands") or high dimensionality of the input space ("Digits"), which may be caused by the inadequate diversity among the linear models established by *MLR*.
- When the sample size is large, there is little difference between the compared combiners no matter what strategy is used to form the meta-level data.
- The *Reusing* strategy should be avoided as much as possible anyway except for large sample sizes because the test errors corresponding to it are generally larger than those obtained by using the *Validation* or *Stacking* technique.
- Some combiners are observed in Fig. 2 to exhibit a dramatic error in some situations, which may be caused by the bad performance of base-level classifier *fisherc* or combiner *FLD* when the training set size is comparable to the dimension of the feature space and this phenomenon has also been indicated by other researchers [16, 22].
- For each combiner, the results obtained by *Stacking II* are slightly better than those derived by *Stacking I*. The reasons for this may be explained as follows. It should be noted that the only difference between these two methods lies in the way to construct the input of the trained combiner when classifying a new example $\mathbf{x}$, that is, *Stacking I* uses the original full training set to

retrain the base-level classifiers which are then utilized to predict $\mathbf{x}$ whereas *Stacking II* employs the mean combiner to fuse the predictions for each base-level classifier that are obtained in each fold of the cross-validation method. Thus, *Stacking II* benefits from more robust base-level classifiers [16].

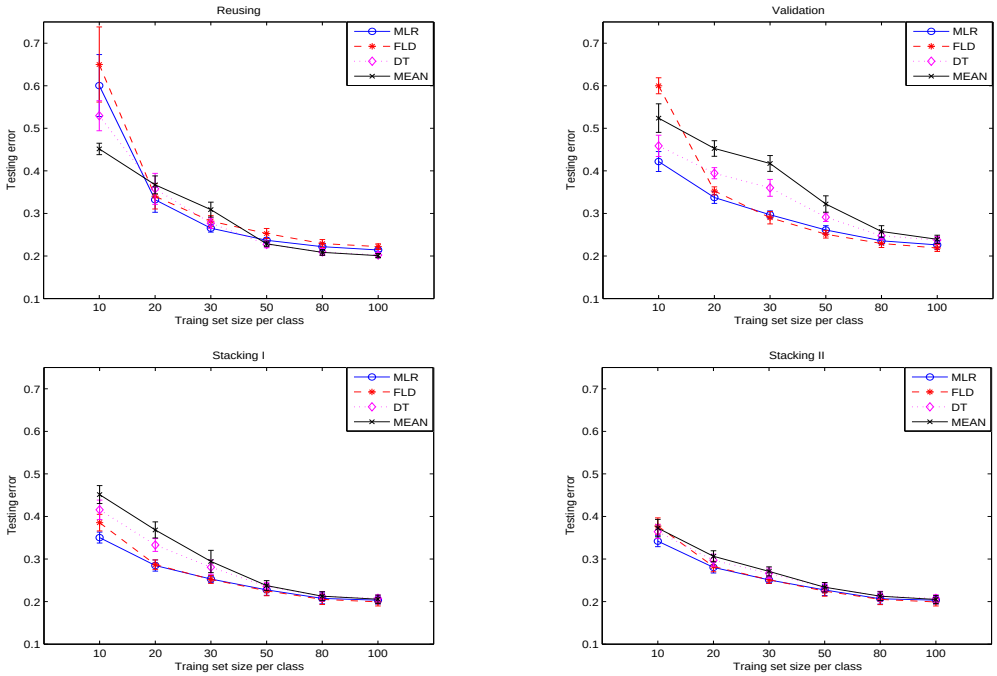

Fig. 1. On "**Cbands**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner.

In order to assess the relative performance of the compared combiners on the used 10 data sets, Table 2 provides some comparative summaries of the mean test errors for the combiner *MLR* in comparison with other ones. For each considered sample size, the geometric means of error ratios and significant Wins-Losses of *MLR* compared with other combiners at each considered sample size were listed here and these statistics were computed in the following way. Take the notation "*MLR/FLD*" as an example, at each sample size we firstly computed the error ratio of *MLR* to *FLD* on each data set, and then calculated the geometric mean of the obtained error ratios across all the data sets. Therefore, the value smaller than 1 indicates the better performance of *MLR*. With respect to the Win-Loss statistic, a paired t-test was utilized to check whether the performance of *MLR* is significantly better than that of the other ones at the significance level 0.05 for each combination of data set and sample size and the numbers listed in the table should be read as the number of data sets on which *MLR* performs significantly better than *FLD*. For instance, *MLR* behaves significantly better than *FLD* on 9 data sets and never losses to *FLD* when the sample size per class is 10 and the *validation* strategy is adopted.

From the results reported in Table 2, we can draw almost the same conclusions as those obtained from the previous figures. Furthermore, it can be found that for large sample sizes, *FLD* performs slightly better than *MLR* when the *Validation* or *Stacking* method is used, which may be due to the fact that in these cases there are enough data for *FLD* to behave well.

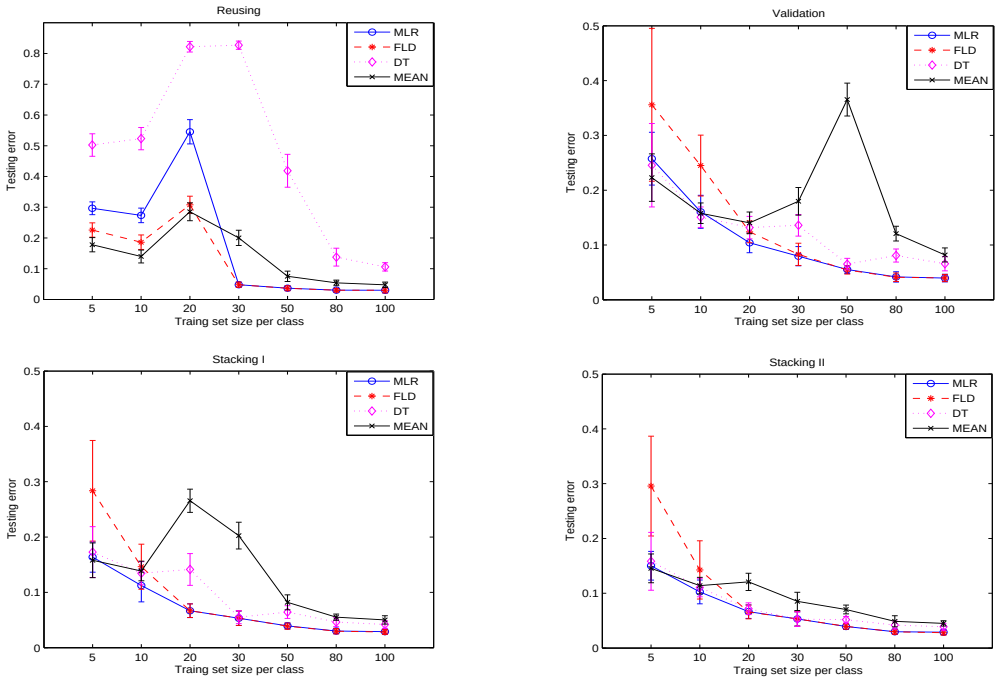

Fig. 2. On "**Digits**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner.

3.3 Complexity analysis

Now let us analyze the complexity of each considered combiner in terms of the number of parameters to be estimated to investigate the reasons for the good performance of *MLR*.

Apparently, the combiner *MEAN* have not any parameters to estimate since it simply averages the probability distributions predicted by each base-level classifier and assigns $\mathbf{x}$ to the class having the largest probability.

With respect to the combiner *MLR*, since it needs to establish m linear models and each of them has L parameters, so there are totally mL parameters required to be estimated.

If we use $\overline{\mathbf{P}}^k = (P^1_{k,1}, P^1_{k,2}, \cdots, P^1_{k,m}, P^2_{k,1}, P^2_{k,2}, \cdots, P^2_{k,m}, \cdots, P^L_{k,1}, P^L_{k,2}, \cdots, P^L_{k,m})^T$ to denote the mean vector of class ω_k in the intermediate feature space, the combiner *DT* estimates the class label of $\mathbf{x}$ as

$$\begin{aligned} \omega_{dt}(\mathbf{x}) &= \underset{1\le k\le m}{\operatorname{argmin}} \sum_{i=1}^{L}\sum_{j=1}^{m}[P^i_{k,j} - P^i_j(\mathbf{x})]^2 \\ &= \underset{1\le k\le m}{\operatorname{argmin}} ||\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})||^2. \end{aligned} \tag{4}$$

Thus, we have to estimate m^2L parameters in this case.

As for the combiner *FLD*, it decides the class label of $\mathbf{x}$ as

$$\omega_{fld}(\mathbf{x}) = \underset{1\le k\le m}{\operatorname{argmin}} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x}))^T \Sigma^{-1} (\overline{\mathbf{P}}^k - \mathbf{P}(\mathbf{x})), \tag{5}$$

here Σ indicates the sample estimate of the $mL \times mL$ covariance matrix supposed to be common for each class. Since PRTools utilizes one-against-all strategy to implement the combiner *FLD* when solving a multi-class task, it needs to estimate $m(\frac{mL(mL+1)}{2} + 2mL)$ parameters in total.

Based on the above analysis, we can see that the compared combiners can be ordered from simple to complex as *MEAN*, *MLR*, *DT* and *FLD*. Thus, one of the reasons for the better performance of the combiner *MLR* at small sample sizes may be attributed to its limited complexity.

4 Conclusions

In this paper, we utilized learning curves to investigate the relative performance of *MLR* for solving multi-class problems in comparison of other trainable combiners *FLD*, *DT* and the fixed combiner *MEAN*. The *Reusing*, *Validation* and two versions of *Stacking* method were respectively considered for using the given data to train the base-level classifiers as well

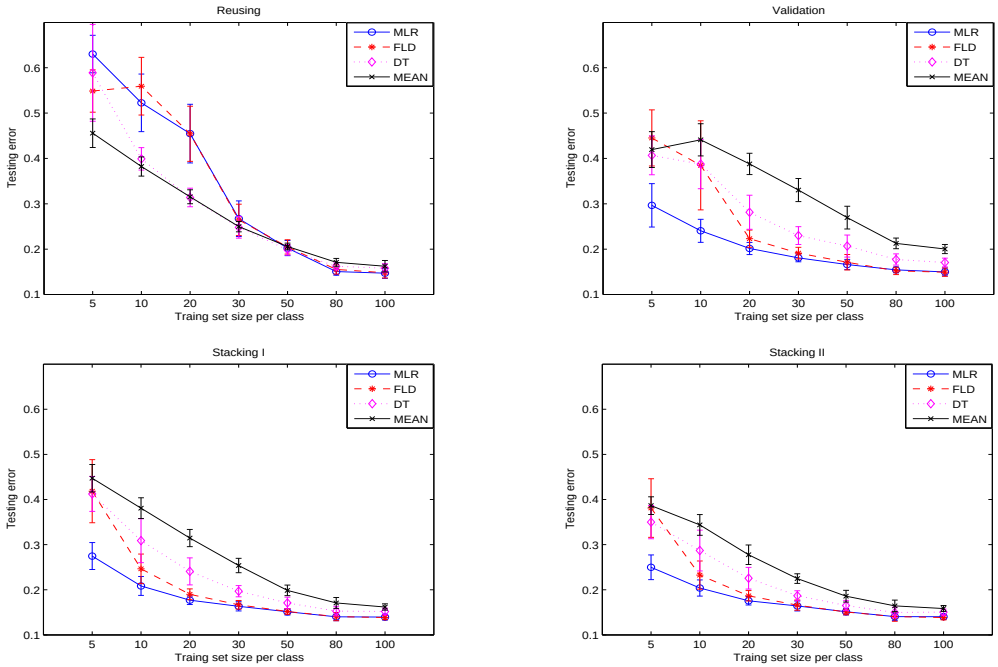

Fig. 3. On "**Satellite**" data set, learning curves for the compared combiners when different strategies are utilized to train base-level classifiers and combiner.

as the combiner. The experimental results show that *MLR* can outperform the other combiners for small sample sizes when *Validation* or *Stacking* method is employed. Meanwhile, the *Reusing* strategy should be avoided as much as possible anyway. When the sample size is large, however, there is little difference between the compared combiners no matter what strategy is employed to form the meta-level data. As for the two *Stacking* methods, *Stacking II* may be preferred over *Stacking I* for its robustness and relatively smaller computational cost.

Acknowledgements This research was supported in part by China State Scholarship Fund, National Basic Research Program of China (973 Program) (Grant No. 2007CB311002) and National Natural Science Foundations of China (Grant Nos. 60675013 and 10531030). This work was conducted when the first author was in Pattern Recognition Group, Delft University of Technology.

References

[1] M. Dettling, BagBoosting for tumor classification with gene expression data. *Bioinformatics*, 20(18): 3583-3593, 2004.

[2] R. Lawrence, A. Bunn, S. Powell, M. Zambon, Classification of remotely sensed imagery using stochastic gradient boosting as a refinement of classification tree analysis. *Remote Sensing of Environment*, 90(3): 331-336, 2004.

[3] T.K. Kim, O. Arandjelović, R. Cipolla, Boosted manifold principal angles for image set-based recognition. *Pattern Recognition*, 40(9): 2475-2484, 2007.

[4] L.I. Kuncheva, J.C. Bezdek, R.P.W. Duin, Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2): 299-314, 2001.

[5] L. Todorovski, S. Džeroski, Combining classifiers with meta decision trees. *Machine Learning*, 50(3): 223-249, 2003.

[6] L. Breiman, Bagging predictors. *Machine Learning*, 24(2): 123-140, 1996.

[7] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm. *In: 13th International Conference on Machine Learning*, pp. 148-156. Morgan Kaufmann Press, San Francisco, 1996.

[8] L. Breiman, Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3): 229-242, 2000.

Table 2. On the used 10 data sets, geometric means of error ratios as well as significant Wins-Losses of *MLR* in comparison with other combiners.

		Training set size per class						
		5	10	20	30	50	80	100
Reusing method	*MLR/FLD*	0.9842	0.9898	1.0269	0.9757	0.9760	0.9952	0.9886
		3-2	5-1	4-1	4-0	2-0	4-1	3-1
	MLR/DT	1.0116	1.0575	1.0532	0.7557	0.7810	0.8404	0.8840
		1-3	1-6	2-4	3-5	2-4	3-5	3-5
	MLR/MEAN	1.2563	1.2221	1.1510	0.8289	0.9123	0.9066	0.9380
		0-6	0-8	1-5	4-2	2-4	3-5	4-5
Validation method	*MLR/FLD*	0.7045	0.7201	0.9517	1.0305	1.0367	1.0318	1.0371
		6-0	9-0	5-2	1-2	1-5	1-4	0-3
	MLR/DT	0.8454	0.8718	0.8897	0.8717	0.9150	0.8721	0.9004
		4-0	9-0	5-0	5-2	5-1	5-0	4-1
	MLR/MEAN	0.9051	0.8391	0.7779	0.7496	0.7070	0.7909	0.8326
		3-0	7-1	7-0	8-0	7-0	5-1	5-1
Stacking I method	*MLR/FLD*	0.6983	0.8535	0.9782	0.9908	1.0158	1.0211	1.0316
		7-0	10-0	2-1	1-1	1-4	0-4	0-4
	MLR/DT	0.8290	0.8569	0.8495	0.9239	0.9162	0.9037	0.9123
		5-0	6-0	5-0	4-1	4-0	4-1	4-1
	MLR/MEAN	0.8922	0.8091	0.7500	0.7622	0.8525	0.8532	0.8620
		3-0	6-0	5-0	5-1	6-1	4-1	4-1
Stacking II method	*MLR/FLD*	0.6919	0.8709	0.9869	0.9963	1.0171	1.0253	1.0347
		7-0	10-0	2-1	1-1	0-3	0-3	0-5
	MLR/DT	0.8499	0.9041	0.9377	0.9579	0.9385	0.9213	0.9216
		4-0	5-0	4-0	4-1	6-1	5-0	4-0
	MLR/MEAN	0.9299	0.8684	0.8513	0.8728	0.8743	0.8821	0.8816
		3-2	4-1	4-1	4-1	6-1	4-1	4-1

[9] K.M. Ting, I.H. Witten, Stacking bagged and dagged models. *In: 14th International Conference on Machine Learning*, pp. 367-375. Morgan Kaufmann Press, San Francisco, 1997.

[10] K.M. Ting, I.H. Witten, Issues in stacked generalization. *Journal of Artificial Intelligent Research*, 10: 271-289, 1999.

[11] C.J. Merz, Using corresponding analysis to combine classifiers. *Machine Learning*, 36(1/2): 33-58, 1999.

[12] A.K. Seewald, How to make stacking better and faster while also taking care of an unknown weakness. *In: 19th International Conference on Machine learning*, pp. 554-561. Morgan Kaufmann Press, San Francisco, 2002.

[13] S. Džeroski, B. Ženko, Is combining classifiers with stacking better than selecting the best ones? *Machine Learning*, 54(3): 255-273, 2004.

[14] S. Raudys, Trainable fusion rules: I. Large sample size case. *Neural Networks*, 19(10): 1506-1516, 2006.

[15] S. Raudys, Trainable fusion rules: II. Small sample-size effects. *Neural Networks*, 19(10): 1517-1527, 2006.

[16] P. Paclík, T.C.W. Landgrebe, D.M.J. Tax, R.P.W. Duin, On deriving the second-stage training set for trainable combiners. *In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS-2005. LNCS*, vol. 3541, pp. 136-146. Springer, Heidelberg, 2005.

[17] M. Liu, B.Z. Yuan, J.F. Chen, Z.J. Miao, Does linear combination outperform the k-NN rule? *In: 8th International Conference on Signal Processing*, vol. 3. IEEE Press, Beijing, 2006.

[18] C.J. Lawson, R.J. Hanson, *Solving Least Squares Problems*. SIAM Publications, Philadephia, 1995.

[19] D.H. Wolpert, Stacked generalization. *Neural Networks*, 5(2): 241-259, 1992.

[20] UCI machine larning respository, http://www.ics.uci.edu/~mlearn/MLRespository.html

[21] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. Ridder, D.M.J. Tax, S. Verzakov, *PRTools4. A Matlab Toolbox for Pattern Recognition*. Delft University of Technology, Delft, 2007.

[22] C. Lai, *Supervised classification and spatial dependency analysis in human cancer using high throughput data*. Ph.D Thesis, Delft University of Technology, 2008.