Solving Regression Problems Using Competitive Ensemble Models

Yakov Frayman, Bernard F. Rolfe, and Geoffrey I. Webb

School of Information Technology
Deakin University
Geelong, VIC, Australia
{yfraym,brolfe,webb}@deakin.edu.au

Abstract. The use of ensemble models in many problem domains has increased significantly in the last few years. The ensemble modeling, in particularly boosting, has shown a great promise in improving predictive performance of a model. Combining the ensemble members is normally done in a co–operative fashion where each of the ensemble members performs the same task and their predictions are aggregated to obtain the improved performance. However, it is also possible to combine the ensemble members in a competitive fashion where the best prediction of a relevant ensemble member is selected for a particular input. This option has been previously somewhat overlooked. The aim of this article is to investigate and compare the competitive and co–operative approaches to combining the models in the ensemble. A comparison is made between a competitive ensemble model and that of MARS with bagging, mixture of experts, hierarchical mixture of experts and a neural network ensemble over several public domain regression problems

that have a high degree of nonlinearity and noise. The empirical results show a substantial advantage of competitive learning versus the co–operative learning for all the regression problems investigated. The requirements for creating the efficient ensembles and the available guidelines are also discussed.

1 Introduction

The main motivation for combining models in ensembles is to improve their generalization ability. The idea of combining models in order to achieve a better prediction has a long history, and has emerged independently in a number of different areas. For example, in econometrics, better forecasting results can be achieved by combining forecasts (model mixing) than by choosing the best model [2]. In machine learning this can be traced back to evidence combination [1].

Recently there was a resurgence of interest in model aggregation particularly in machine learning and data mining. One of the aggregation methods, bagging, is aimed at reducing the variance of predictive models [5]. Another one, stacking [24], attempted to decrease prediction bias in addition to variance. But the most popular ensemble method is boosting [20], aimed on transforming a collection of weak models into one strong model. The recently developed AdaBoost algorithm [11] sequentially fits weak models to different weighting of the observations in a data set. The observations that are predicted poorly receive greater weighting on the next iteration. The resulting AdaBoost model is

R.I. McKay and J. Slaney (Eds.): AI 2002, LNAI 2557, pp. 511–522, 2002.

Page 2

a weighted average of all the weak predictors. The AdaBoost is shown to be effective for reducing bias and variance in a wide range of classification problems [3]. However, the recent work on boosting [13] have shown that AdaBoost algorithm is an optimization method for finding a model that minimizes a particular exponential loss function, namely the Bernoulli likelihood. Recent investigations of the boosting algorithm from the statistical viewpoint [18] have found that while boosting algorithm appears complex on surface, it is similar to the ways linear models are fitted in statistics. This opens the possibility to unite seemingly different, but essentially similar approaches to ensemble

modeling from machine learning and statistical and engineering viewpoints.

The current trend in boosting with major developments in boosted regression [9] is going in a direction similar to the statistical approach to model mixing. The boosted regression approach follows the spirit of AdaBoost algorithm by repeatedly performing weighted tree regression followed by increasing weighting of the poorly predicted observations and decreasing the weighting of the better predicted samples. The work on Gradient Boosting Machine [14] uses the connection between boosting and optimization more explicitly. At each iteration the algorithm determines the direction, the gradient, in which it needs to improve the fit to the data and selects a particular model from the available class of functions that is most in agreement with the direction. While this is a significant improvement on the approach of AdaBoost, the Gradient Boosting Machine algorithm is, however, increasingly similar to other non–parametric regression models such as neural networks. As such the Gradient Boosting Machine still does not provide a reliably guidance to selection of ensemble members and effective means of combining such ensemble members. This leads us to analysis of the available methods for creation of ensemble members and their combination for efficient ensembles.

2 Ensemble Modeling

The effectiveness of an ensemble can be measured by the extent to which the members are error–independent (show different patterns of generalization) [19]. The ideal would be a set of models where each of the models generalize well, and when they do make errors on new data, these errors are not shared with any other models [19].

2.1 Creating Ensemble Members

In order to create efficient ensembles we need to consider the relative merits of methods of creating ensemble members, and to choose and apply one that is likely to result in models that generalize differently. There are several possible ways to achieve this objective that include the following:

Sampling data: A set of models for an ensemble is commonly created by using some form of sampling, such that each model in the ensemble is trained on a different sub–sample of the training data. Re-sampling methods which have been used for this purpose include cross-validation, and bootstrapping [5]. In bagging [5], a training set containing N cases is perturbed by sampling with replacement (bootstrap) N times from the training set. The perturbed data set may contain repeats. This procedure can be repeated several times to create a number of different, although overlapping, data sets. A method similar

to the sampling is the use of disjoint training sets, that is sampling without replacement. There is then no overlap between the data used to train different models.

However, the presence of correlation between the errors of the ensemble members could reduce the effectiveness of the ensemble [15]. While sampling the data might be an effective way of producing models that generalize differently, this will not necessarily result in low error correlations as it requires a representative training set where a function being inferred is similar to that which generated the test set [8]. However, two representative training sets could lead to very similar functions being inferred, so their pattern of errors on the new data would be very similar.

On the other hand, if ensemble members are trained using unrepresentative training sets, the resulting generalization performance would be poor. Each member might show different patterns of generalization, but as the amount of errors increases so does the probability that their errors will overlap.

Boosting and adaptive resampling: In boosting, as discussed previously, a series of weak learners could be converted to a strong learner as a result of training the members of an ensemble on patterns that have been filtered by previously trained members [20], [23]. AdaBoost algorithm [11] has training sets adaptively resampled, such that the weights in the resampling are increased for those cases which are most often predicted incorrectly.

*Varying the learning method employed:*The learning method used to train the models could be varied while holding the data constant when creating ensemble members. An ensemble might be constructed from models generated by a combination of learning techniques such as various statistical methods, linear regression, neural networks, k–nearest neighbors, decision trees, and Markov chains [7].

While this approach is not commonly used, in our opinion, it is a very promising approach to ensemble learning, as the use of different learning methods for ensemble members is more likely to result in different patterns of generalization than sampling the data. Furthermore, there is a possibility to combine varying both the learning method and the data, for example, with adaptive re–sampling of the data.

In this paper we will investigate the approach of varying the learning method for creation of ensemble members and compare it with approaches that use sampling the data.

2.2 Combining Ensemble Members

The next step in ensemble learning is to find an effective way of combining model outputs. While there exists several possible ways of combining models in an ensemble, the co–operative combination is the most dominant one. In co–operative combination it is assumed that all of the ensemble members will make some contribution to the ensemble decision, even though this contribution may be weighted in some way.

Methods of combining the models in co–operative fashion include the following:

Averaging and weighted averaging: Linear combination of the outputs of the ensemble members are one of the most popular aggregation methods. A single output can be created from a set of model outputs via simple averaging, or by means of a weighted average that takes account of the relative accuracy of the models to be combined.

Stacked generalization: Stacked generalization [24] uses an additional model that learns how to combine the models with weights that vary over the feature space. The

Page 4

outputs from a set of level 0 generalizers are used as the input to a level 1 generalizer, which is trained to produce the appropriate output. It is also possible to view other methods of combining, such as averaging, as instances of stacking with a simple level 1 generalizer.

In competitive combination, on the other hand, it is assumed that for each input only the most appropriate ensemble member will be selected based on either the inputs or outputs of the models [21].

The two main methods for a competitive combination (selection) are:

Gating: Under the divide and conquer approach employed by mixtures-of-experts [16] and hierarchical mixtures-of-experts [17] the complex problem is decomposed into a set of simpler problems. The data is partitioned into regions and the simple surfaces are fitted to the data into each region. The regions have soft boundaries where data points may lie simultaneously in multiple regions. Such decomposition ensures that the errors made by the expert models will not be correlated as they deal with different data points. A gating model is used to output a set of scalar coefficients that weights the contributions of the various inputs.

Rule–based switching: In this case, the switching between the models can be triggered on the basis of the input or the output of one of the models. For example, in the study on the diagnosis of myocardial infarction (heart attack), two models were optimized separately by varying the proportion of high risk and low risk patients in the training sets. The first model was trained to make as few positive errors as possible, and the second model was trained to make as few negative errors as possible [4]. The output of the first model was used unless it exceeds a threshold in which case the output of the second model was used.

There exists other examples of rule–based switching, for example, switching of control to the most appropriate model depending on the current situation as it is exploited in behavior–based robotics [6].

Our empirical observation [10] has been that the better results can be obtained through the use of a more explicit rule–based switching between models.

3 Computational Experiments

3.1 Experimental Set Up

To evaluate the performance of a competitive ensemble model, several public domain regression data sets were selected from DELVE (Data for Evaluating Learning in Valid Experiments) (see http://www.cs.toronto.edu/~delve/). DELVE is a standardized environment designed to evaluate the performance of methods that learn relationships based primarily on empirical data. DELVE makes it possible for users to compare their learning methods with other methods on many data sets. The DELVE learning methods and evaluation procedures are well documented, such that meaningful comparisons can be made. Since our approach involves ensembles, we compared the performance of our competitive ensemble model to that of multivariable adaptive regression splines (MARS) with bagging, mixture of experts, hierarchical mixture of experts, neural network ensemble and also with a standard linear regression as a baseline method. The competitive ensemble model was used in accordance with DELVE guidelines. The performance of other

methods are available from DELVE. The splitting of data sets into training and testing was done using a DELVE software environment, which allows to manipulate data sets and do statistical analysis of method's performance.

We have selected from DELVE environment all the regression data sets where there are results available of the MARS with bagging, the mixture of experts, the hierarchical mixture of experts, the neural network ensemble and the linear regression as follows:

a) *Boston Housing data-set.* The Boston Housing data-set is a small but widely used data-set derived from information collected by the U.S. Census Service concerning housing in the Boston, Massachusetts area. It has been used extensively throughout the literature to benchmark algorithms. The task is to predict the median value of a home (price).

b) *Pumadyn family of data sets.*The Pumadyn family of data sets is a realistic simulation of the dynamics of a Puma 560 robot arm. The task is to predict angular acceleration of one of the robot arm's links. The inputs include angular positions, velocities

and torques of the robot arm. The family has been specifically generated for the DELVE environment and so the individual data sets span the corners of a cube whose dimensions represent: (a) number of inputs (8 or 32), (b) degree of non-linearity (fairly linear or non-linear), (c) amount of noise in the output (moderate or high).

c) *Kin family of data sets.* The Kin family of data sets is a realistic simulation of the forward dynamics of an 8 link all–revolute robot arm. The task is to predict the distance of the end–effector from a target. The inputs are factors like joint positions and twist angles. The family has been also specifically generated for the DELVE environment and has the same dimensions as Pumadyn family of data sets.

We have considered both dimensionalities of the input (8 or 32) and have chosen high amount of noise and a large amount of non–linearity to make the tasks similar to other real–world tasks we have investigated. In addition, we have only considered the larger size of training and testing sets (1024 for pumadyn and kin, and 128 for boston housing) out of that available in the DELVE environment. Training and testing sets were generated randomly from the respective data sets using a DELVE software environments. Thus, there are 2 different training and testing sets for Boston Housing data set, and 4 for all the Pumadyn and Kin data sets. Table 1 summarizes the data characteristics.

Table 1. Data Characteristics.

Name	Size	Train	Test	Inputs	Noise	Non-Linearity
Boston Housing	506	128	128	13	–	–
Pumadyn-8nh	8192	1024	1024	8	high	non–linear
Pumadyn-32nh	8192	1024	1024	32	high	non–linear
Kin-8nh	8192	1024	1024	8	high	non–linear
Kin-32nh	8192	1024	1024	32	high	non–linear

3.2 Regression Methods Used

The competitive (selection) model used in this paper was created following the guidelines discussed previously.

A non–linear model (neural network) was trained to select the appropriate output of the ensemble members as a final output of the ensemble model based on the performance of ensemble members on a particular data tuple. The aim here is basically to create a global model (ensemble model) where each of the ensemble members is acting as a local predictor in the area of its best performance. In such a rule–based switching [22], the control is switched between the ensemble members depending on the output of one of the members.

For the ensemble members (level 0 generalizers) we have selected a linear method (linear regression), an efficient nonlinear method (multilayer perceptron (MLP) with two hidden layers with 20 nodes each), a logistic regression (MLP with a single hidden node) and a clustering algorithm (KNN) with k equal 10. The reasons for selecting these learning methods for creation of the ensemble members are that these methods are very different in nature and as such have different learning biases. In our experience, these learning methods have a different pattern of generalization and as such can produce an efficient ensemble.

As a level 1 generalizer (selector model) another MLP consisting of 2 hidden layers of 20 hidden nodes each was used. All MLPs were fully connected with hyperbolic tangent hidden units and linear output units. All ensemble members were trained using the back–propagation learning algorithm with early stopping to avoiding over–fitting. The pattern (on–line) learning was used. The learning rate of 0.05 and momentum of 0.99 were used for all MLPs to avoid local minima. The structure and the parameters of both MLPs and the KNN were selected based on our experience with other data sets and no attempt was made to optimize the level 0 models to the data sets considered.

The structure of the competitive model is in Fig 1. In this case all the available inputs in a training set were supplied to the level 0 models and the selection model. Selection model also receives the output of the level 0 models. In the learning phase a binary value is assigned to the output of the selection model that represents the best model or otherwise for a particular data tuple. The selection model is solving a classification problem which is to choose the appropriate output of the one of the level 0 models as the final output of the ensemble. Supplying the input data to the selection model is not strictly necessary as the MLP with 2 hidden layers is a very efficient classifier, but may help the selection model to distinguish between the level 0 models in some cases.

For comparison with our selection model we have used several popular regression methods: multivariable adaptive regression splines with bagging, mixture–of-experts, hierarchical mixture–of-experts and ensemble of neural networks. We have already discussed mixture–of-experts and hierarchical mixture–of-experts methods. In the following, we will briefly describe the rest of the methods.

Multivariable adaptive regression splines (MARS) [12] were created to provide the advantages of tree based regression methods without the disadvantages of the response being discontinuous along the boundaries. Here each step basis function in the predictor space is replaced by a pair of linear basis functions. The new splits are not required to

Page 7

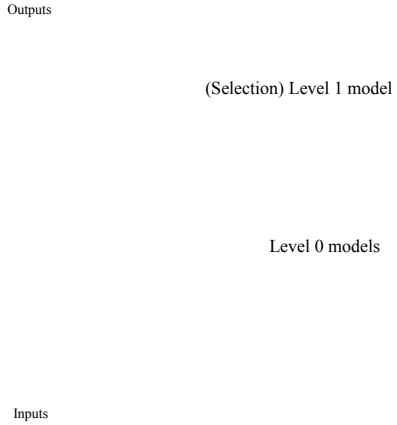

Fig. 1. Selection Model.

depend on the previous splits. The final solution is made smooth by replacing the linear functions with cubic functions after backward deletion of unnecessary basic functions.

Table 2. Results from Boston data-set.

Method	Standardized expected loss	Standard error estimated for difference of estimate	Significance of difference (F-test)
Selection ensemble	0.05663	–	–
Linear regression	0.28123	0.02920	< 0.05
HME (ensemble learning)	0.16204	0.01416	< 0.05
HME (early stopping)	0.17312	0.02075	< 0.05
HME (growing and early stopping)	0.17623	0.02063	< 0.05
MARS version 3.6 with Bagging	0.15713	0.02104	< 0.05
ME (ensemble learning)	0.15937	0.02431	< 0.05
ME (early stopping)	0.16006	0.01464	< 0.05
MLP (early stopping)	0.21014	0.01697	< 0.05

MARS was used in conjunction with the bagging procedure [5]. Using this method one trains MARS on a number of bootstrap samples of the training set and averages the resulting predictions. The bootstrap samples are generated by sampling the original training set with replacement. Samples of the same size as the original training set are used.

The ensemble of neural networks from DELVE repository trains ensembles of MLPs using early stopping to avoiding over–fitting. The networks all have identical architecture: fully connected with a single hidden layer of hyperbolic tangent units and linear output units. The minimization algorithm is based on conjugate gradients.

A fraction of the training examples are held out for validation, and performance on this set is monitored while the iterative learning procedure is applied. The learning is stopped as soon as minimum in validation error is achieved.

The number of hidden units is chosen to be the smallest such that the number of weights is at least as large as the total number of training cases after removal of the validation cases. One third of the training examples were used for validation and the rest for training.

Table 3. Results from Kin-8nh and Kin-32nh data sets.

	Kin-8nh		
Method	Standarized expected loss	Standard error estimated for difference	Significance of difference (T-test)
Selection ensemble	0.22896	–	–
Linear regression	0.62930	0.00929	0.00002
HME (ensemble learning)	0.52120	0.00987	0.00008
HME (early stopping)	0.50874	0.01167	0.00016
HME (growing and early stopping)	0.54611	0.01005	0.00007
MARS version 3.6 with Bagging	0.57910	0.00827	0.00003
ME (ensemble learning)	0.45127	0.01050	0.00023
ME (early stopping)	0.46433	0.01149	0.00025
MLP (early stopping)	0.40543	0.00701	0.00013
	Kin-32nh		
Method	Standarized expected loss	Standard error estimated for difference	Significance of difference (T-test)

Selection ensemble	0.57230	–	–
Linear regression	0.81556	0.00948	0.00013
HME (ensemble learning)	0.79264	0.00750	0.00003
HME (early stopping)	0.78321	0.01413	0.00065
HME (growing and early stopping)	0.79076	0.01591	0.00084
MARS version 3.6 with Bagging	0.84432	0.01551	0.00040
ME (ensemble learning)	0.80461	0.00894	0.00002
ME (early stopping)	0.78322	0.01413	0.00065
MLP (early stopping)	0.79613	0.01170	0.00031

3.3 Experimental Results and Discussion

For each of the data sets the attribute variables were normalized to a median of zero and an average absolute deviation from the median of one. This enabled each level 0 model to learn from the same set of data as certain models have limits on the input and output data ranges.

A standard squared loss function between the target output value and the predicted value was used to calculate the error for each ensemble member. The predicted values were then un–normalised before they were compared to the target values.

We have only run the competitive model once over each training and testing sets using the same parameters for the ensemble members, in accordance with DELVE guidelines. All the results obtained are thus the averages of these runs.

DELVE uses an ANOVA (ANalysis Of VAriance) model to estimate the statistics in evaluating the performance of different models. Tables 2–6 contain the resulting statistical values from the competitive (selection) ensemble model and other ensemble models on the five data sets considered. Each table includes the standardized estimated squared error loss that is calculated against a simple baseline method within DELVE. The second statistic is the standard error of the estimated expected difference. This gives an indication of the variability of the estimated expected difference between the two

methods. The final statistic is either F–test or T–test value which is a probability that the null–hypothesis (no difference between the expected loss of the two methods) is true. This means that high values of the F–test or of the T–test would indicate the two methods are probably similar, and low values of the tests indicate that the difference between the performances of different methods are statistically significant.

Table 4. Results from Pumadyn-8nh and Pumadyn-32nh data sets.

	Pumadyn-8nh		
Method	Standardized Standard error estimated for difference of difference		Significance
	expected loss	estimate	(F-test)
Selection ensemble	0.21343	–	–
Linear regression	0.63146	0.01172	0.00005
HME (ensemble learning)	0.36389	0.01085	0.00081
HME (early stopping)	0.44881	0.01749	0.00089
HME (growing and early stopping)	0.51206	0.02107	0.00076
MARS version 3.6 with Bagging	0.33650	0.00705	0.00041
ME (ensemble learning)	0.36034	0.00950	0.00059
ME (early stopping)	0.40625	0.01084	0.00039
MLP (early stopping)	0.34271	0.01358	0.00246
	Pumadyn-32nh		
Method	Standarized Standard error estimated for difference of difference		Significance
	expected loss	estimate	(T-test)
Selection ensemble	0.26010	–	–
Linear regression	0.86604	0.01413	0.00002
HME (ensemble learning)	0.88020	0.01460	0.000002
HME (early stopping)	0.87137	0.01600	0.00004
HME (growing and early stopping)	0.85896	0.01397	0.00002
MARS version 3.6 with Bagging	0.34236	0.00711	0.00139
ME (ensemble learning)	0.88173	0.01489	0.000002
ME (early stopping)	0.86792	0.01426	0.00003
MLP (early stopping)	0.70891	0.02542	0.00040

In Tables 2–6, HME (ensemble learning) is a Hierarchical mixture of experts trained using Bayesian methods, HME (early stopping) is a Hierarchical mixtures of experts trained using early stopping, HME (growing and early stopping) is a Hierarchical mixtures of experts trained using growing and early stopping, ME (ensemble learning) is a Mixtures of experts trained using Bayesian methods, ME (early stopping) is a Mix-

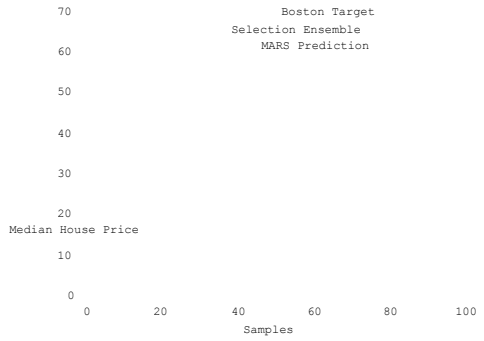

Fig. 2. First 100 samples of the Boston data, plot of the selection ensemble model and the MARS model version 3.6 with bagging versus the actual target.

tures of experts trained using early stopping and MLP (early stopping) is a Multilayer perceptron ensembles trained with early stopping.

The actual predictions of a competitive model and the best of the comparison models are shown in Figs 2–4. For clarity only the first 100 samples of testing set are shown for respective data sets.

Tables 2–6 show that the competitive (selection) model is significantly better, to a confidence of 95% ($p = 0.05$), than any of the other methods. This demonstrates that the competitive (selection) combination in conjunction with varying the learning method approach for creating ensemble members is working better than the co–operative combination in conjunction with sampling the data approach for creating ensemble members.

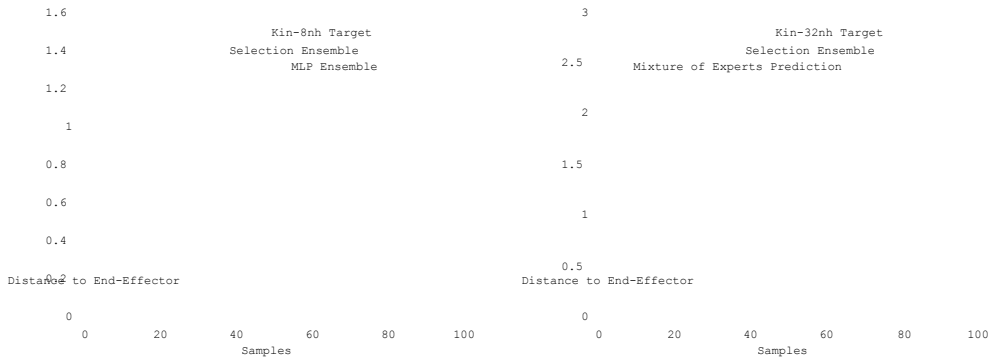

Fig. 3. First 100 samples of the (a) Kin-8nh data, plot of the selection ensemble model and the MLP ensemble model with early stopping versus the actual target (b) the Kin-32nh data, plot of the selection ensemble model and the mixture-of-experts model with early stopping versus the actual target.

Page 11

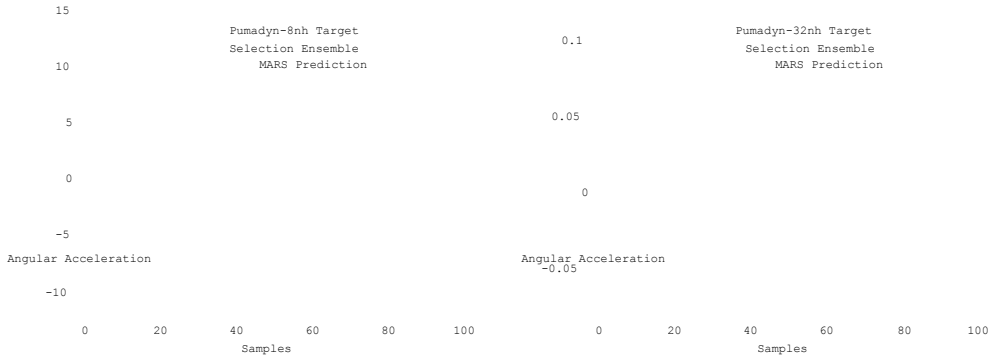

Fig. 4. First 100 samples of the (a) Pumadyn-8nh data, plot of the selection ensemble model and the MARS model version 3.6 with bagging versus the actual target, (b) the Pumadyn-32nh data, plot of the selection ensemble model and the MARS model version 3.6 with bagging versus the actual target.

However, while the competitive model is much better than the other models, there is still much room for improvement considering the actual prediction accuracy. The prediction of the competitive (selection) model for Boston Housing data set can be considered good, as seen in Fig. 2, even though there are some predictions that are not quite accurate. In case of Figs. 3 and 4, the results of the selection model are not as good, especially in case of Fig. 3(b), even though the selection model out–performs the other ensemble models.

We obviously need to keep in mind that we explicitly selected the most difficult data sets of that available in DELVE. However, based on our experience with real data sets [10] the selected regression problems are no more difficult than the many real non–linear problems with a high degree of noise. This just shows that the ultimate goal of the prediction is not just to achieve better results than the competing methods, but to achieve the best possible prediction.

References

1. Barnett, J. A. “Computational methods for a mathematical theory of evidence”, *Proceedings of IJCAI*, pp. 868–875, 1981.
2. Bates, J. M. and C. W. J. Granger. “The combination of forecasts”. *Operations Research Quaterly*, 20:451–468, 1969.
3. Bauer, E. and Kohavi, R. “An empirical comparison of voting classification algorithms: bagging, boosting and variants”. *Machine Learning*, 36(1,2), 105–139, 1999.
4. Baxt, W. G. “Improving the accuracy of an artificial neural network using multiple differently trained networks”. *Neural Computation*, 4:772–780, 1992.
5. Breiman, L. “Bagging predictors”. *Machine Learning*, 26(2):123–140, 1996.
6. Brooks, R. A. “A robust layered control system for a mobile robot”. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.
7. Catfolis, T. and Meert, K. “Hybridization and specialization of real–time recurrent learning–based neural networks”, *Connectionist Science*, 9(1):51–70, 1997.

8. Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L. and Hopfield, J. “Large automatic learning, rule extraction and generalisation”. *Complex Systems*, 1:877–922, 1987.
9. Drucker, H. “Improving regressors using boosting techniques”. *Proceedings of the 14th International Conference on Machine Learning*, pp. 107–115, 1997.
10. Frayman, Y., Rolfe B. F., Hodgson, P. D. and Webb G. I. “Predicting the rolling force in hot steel rolling mill using an ensemble model”. *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2002)*, 2002. (in press).
11. Freund, Y. and R.Schapire. “A decision–theoretic generalization of on–line learning and an application to boosting”. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
12. Friedman J. “Multivariate adaptive regression splines (with discussion)”. *Annals of Statistics*, 19(1), 1–82, 1991.
13. Friedman, J., Hastie, T., and Tibshirani, R. “Additive logistic regression: a statistical view of boosting (with discussion)”, *Annals of Statistics*, 28(2), 337–374, 2000.
14. Friedman, J. “Greedy function approximation: a gradient boosting machine”.*Annals of Statistics*, 29(4). 2001.
15. Hashem, S. “Optimal linear combinations of neural networks”. *Neural Networks*, 10(4):599–

614, 1997.

16. Jacobs, R. A, Jordan, M. I., Nowlan, S. J., and Hinton, G. E. “Adaptive mixtures of local experts”. *Neural Computation*, 3:79–97, 1991.
17. Jordan, M. I. and Jacobs R. A. “Hierarchical mixtures of experts and the em algorithm”. *Neural Computation*, 6(2):181–214, 1994.
18. Ridgeway, G. “The state of boosting”. *Computing Science and Statistics*, 31:172–7181, 1999.
19. Rogova, G. “Combining the results of several neural network classifiers”. *Neural Networks*, 7(5):777–781, 1994.
20. Schapire, R. E. “The strength of weak learnability”. *Machine Learning*, 5:197–227, 1990.
21. Sharkey, A.J.C. (Ed.) *Combining artificial neural nets: ensemble and modular multi-net systems,* Springer-Verlag, 1999.
22. Ting, K. M. “The characterisation of predictive accuracy and decision combination”. *Proceedings of the 13th International Conference on Machine Learning*, pp. 498–506, 1996.
23. Webb, G. “MultiBoosting: a technique for combining boosting and wagging”.*Machine Learning*, 40(2): 159–196, 2000.
24. Wolpert, D.H. “Stacked generalization”. *Neural Networks*, 5:241–259, 1992.