Combining the Advantages of Neural Networks and Decision Trees for Regression Problems in a Steel Temperature Prediction System

Miroslaw Kordos[1], Piotr Kania[1], Pawel Budzyna[1],
Marcin Blachnik[2], Tadeusz Wieczorek[2], and Slawomir Golak[2]

[1] University of Bielsko-Biala, Department of Mathematics and Computer Science,
Bielsko-Biala, Willowa 2, Poland
mkordos@ath.bielsko.pl
[2] Silesian University of Technology, Department of Management and Informatics,
Katowice, Krasinskiego 8, Poland
marcin.blachnik@polsl.pl

Abstract. Simple decision trees enable obtaining simple logical rules with a limited accuracy in regression tasks. Neural networks as highly non-linear systems can map much more complex shapes and thus can obtain higher prediction accuracy in regression problems, that is however at the cost of the poor comprehensibility of the decision process. We present a hybrid system which incorporates the features of both a regression tree and a neural network. This system allowed for achieving high prediction accuracy supported by comprehensive logical rules for the practical problem of temperature prediction in electric arc furnace at one of the steelwors.

Keywords: neural networks, decision tree, regression, logical rules.

1 Introduction

There are some differences between extracting logical rules for classification and for regression tasks. A good strategy in data mining for the classification task is to extract simplest crisp logical rules first, because they provide hyper-rectangular decision borders in the feature space and that kind of rules are easy to understand. In the regression tasks also simple logical rules are preferred, however the issue is more complex because of the continuous output space. Several approaches to that issue have been attempted so far, including simple decision trees, which really provide hyper-rectangular decision borders, but which are frequently not the most accurate solution. Also rule extraction from standard multilayer perceptrons was performed, which frequently provides quite a good approximation, but the extracted logical rules can be of very high complexity and difficult to understand by experts in a given domain. In order to address this issue we tried two approaches: to improve the accuracy of a decision tree and to improve the comprehensibility of rules extracted from a neural network. Although hybrid methods can obtain very good accuracy [1,2], the customers are not always happy with them, because of the level of knowledge required to understand the systems. Thus, each of the

E. Corchado et al. (Eds.): HAIS 2012, Part II, LNCS 7209, pp. 36–45, 2012.

solutions was aimed to be used as the only one kind of prediction model, because this homogeneity is more preferred by our customers. The purpose of the methodology we present in this paper is to achieve both goals; high accuracy and low rule complexity as far as possible by using only a single neural network-based model.

This section outlines the problems associated with neural network-based rule extraction for regression tasks and the problems connected with temperature control in electric arc steel-making. In the Methodology section we present our system and in the Experimental Evaluation section we compare it to some other systems on the two metallurgical datasets that can be obtained from [3] and on two datasets from the UCI repository [4,5].

1.1 Neural Network-Based Rule Extraction for Regression Problems

In our paper presented at the previous HAIS conference [6] we considered optimization of regression tree parameters. Although the logical rules were very clear and allowed for a better understanding of the process, we were able to obtain much higher accuracy for the same task using an ensemble of neural networks with evolutionary optimization [8]. Thus, for the practical purposes, where the system had to be used in production environment in the metallurgical industry, we decided to use a committee of several models. In our current approach we want to simplify this committee by introducing a new regression and rule extraction model based on a specially designed and specially trained neural network.

Most of the research concerns rule extraction for classification tasks, which is much easier. So far only a few approaches to rule extraction from neural networks for regression tasks have been described in the literature.

Setiono et. al. [9] and Wang et. al. [10] proposed an approach where each rule in the extracted rule set corresponds to a subregion of the input space. The nonlinear activation function (hyperbolic tangent or logistic sigmoid) of each hidden neuron is approximated locally by a three-piece linear function based on least squares estimation and then the regression rules are generated. Although in our experiments the accuracy of the rules obtained from that approach was only little below the accuracy of the underlying neural network, the complexity of the rules, when applied to our data was too high to allow understanding them quickly (in the time range below 1 minute, which is the time, the electric arc operator has to make decisions about how to further continue the process).

Kazumi [11] proposed a method where each regression rule is expressed as a pair of a logical formula on the conditional part over nominal variables and a polynomial equation on the action part over numeric variables. The proposed extraction method first generates one such regression rule for each training sample, then utilizes the k-means algorithm to generate a much smaller set of rules having more general conditions, where the number of distinct polynomial equations is determined through cross-validation. Finally, this method invokes decision tree induction to form logical formulae of nominal conditions as conditional parts of the final regression rules.

Markowska and Mularczyk [12] proposed o methodology based on two hierarchical evolutionary algorithms with multiobjective Pareto optimization. The lower level algorithm searches for rules that are optimized by the upper level algorithm. The conclusion

of the rule takes the form of a tree, where non-terminal nodes contain functions and operators and leaves contain identifiers of attributes and numeric constants.

1.2 Temperature Control in the Electric Arc Furnace

In the electric arc furnace the steel scrap is melted using the electric arc to generate most of the heat. Additional heat is obtained from gas that is inserted and burnt in the furnace. The optimal temperature of the melted steel that is to be tapped out from the furnace is about 1900K, however it must be kept at proper temperature enough long so that all the solid metal gets melted. If the heating lasts too long, unnecessary time and energy is wasted and additional wear of the furnace is caused. Modern EAFs have the melt times of 30 minutes, older ones up to one hour.

The temperature is measured a few times during every melt by special lances with thermocouple that are inserted into the liquid steel. Every measurement takes about one minute and in this time the arc has to be turn off and the process suspended. Waste of time and energy for three or even more measurements is thus quite significant. There are many problems with the continuous measurement of the steel temperature. The temperatures are very high and the radiant heat and the electro-magnetic radiation from the furnace creates major problems for the measuring equipment.

Therefore there was a need to build a temperature prediction system that would allow us to limit the number of temperature measurements and thus shorten the EAF process to make it more economical. We previously built a system based on a single regression model [6], as a part of the whole intelligent system for steel production optimization [14]. However, as our recent experiments showed, a significant improvement can be obtained with a committee of regression models. But on the other hand using complex committees of hybrid regression models made the extraction of simple comprehensive rules unrealistic in practice.

2 Methodology

2.1 Data Preprocessing

The presence of outliers and wrong data may dramatically reduce generalization abilities and limit the convergence of training process of any learning algorithm. Proper data preprocessing is helpful not only to deal with outliers but also to achieve variable sensitivity of the model in different ranges of input variables. First we standardize the data before the training, according to the following formula:

$$x_{std} = \frac{x - \bar{x}}{\sigma} \quad \sigma = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x - \bar{x})^2} \tag{1}$$

and then we transform the data be the hyperbolic tangent function

$$y2 = \frac{1 - exp\,(-\beta \cdot y + \theta)}{1 + exp\,(-\beta \cdot y + \theta)} \tag{2}$$

where y is the output value before the transformation and $y2$ - after. In practical problems, it is frequently desired to obtain a model with higher sensitivity in the intervals with more dense data (as it was in our case) or in other intervals of special interests. To address his issue, we transfer the data through a hyperbolic tangent function. The other advantage of the transformation is the automatic reduction of the outliers' influence on the model. Because if the complexity of our data and lot of error in particular value measurements, before the values were recorded into the database it is very difficult to properly apply known outliers removal methods, such as ENN or other. For that reason we do not do not reject the outliers [13], but rather reduce their influence on the final model, because it is frequently not clear whether a given value is already an outlier or wrong value or is still correct. The hyperbolic tangent transformation allows for a smooth reduction of the outliers, because no matter how big the value is, after the transformation it will never be grater than one or smaller than minus one. however, in the case of multimodal data distribution the data should be first divided into several single-mode distribution datasets.

2.2 Network Construction

The neural network is based on a 3-layer MLP architecture. There are two kind of networks used in parallel. At the beginning of the training neurons implement hyperbolic tangent transfer functions, then in the rule-based network the transfer functions are gradually changed to step functions, while in the classical network they remain unchanged. The rule-based network requires discrete input data. If the data is continuous, what is frequently the case in regression tasks, it must be discretized prior to the training.

We tried using equi-distance and equal-number bins during the discretization. However, it turns out, that the best results can be achieved if the continuous attributes are discretized taking into account not only the input but also the output value. That prevents grouping into one bin the attribute values which correspond to a quite different output values and prevents making a split in the points for which output value does not differ. For that reason we use a decision tree as described in [6] for which there is only one input - the attribute under consideration. The decision tree divides the input attribute into bins in such a way, which minimizes the variance of each node. The division is performed so long as the predefined stopping criteria are met: the variance of the output value in the node, minimum number of vectors inn the node and maximum tree depth (For more details see [6]). In practice, with the metallurgical datasets on average the number of continuous feature bins (corresponding to the number of the tree leaves) was 4 to 16.

In the network for rule extraction a separate input neuron for each discretized feature value is used. Thus the number of all input neurons equals the sum of all distinct values for all features. The input values are 1 if the feature has the value represented by this neuron and 0 otherwise.

In practice we use a separate network for each output value bin. One hidden neuron per each output bin is created at the beginning. The second hidden neuron per each output value bin is added, if the results with only one neuron are not satisfactory and then the weights of the first neuron are frozen and only the newly added neuron is trained. If the results are still unsatisfactory then the next hidden neuron is added. The number of

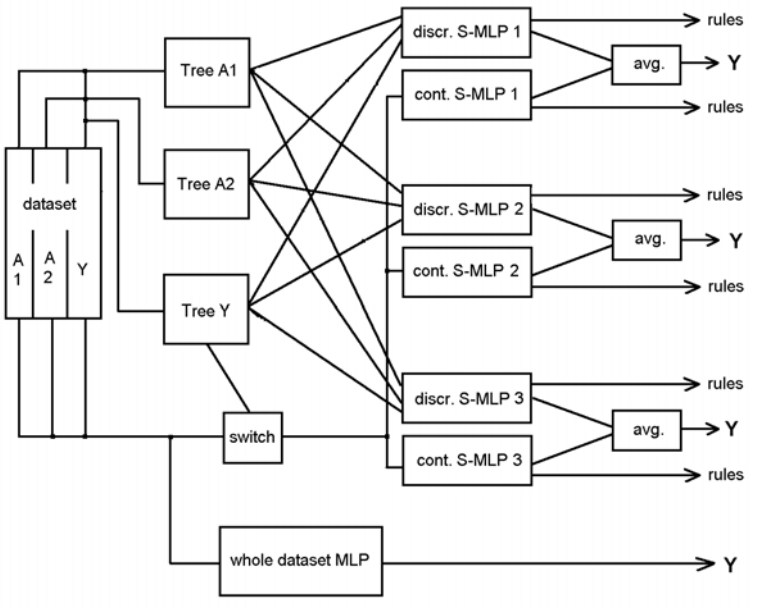

Fig. 1. The system used for temperature prediction in the EAF process

hidden neurons per one output bin should equal the number of data clusters that correspond to that output value bin. The first hidden neuron corresponds to the biggest cluster and thus generates the most general rules, while the second hidden neuron generates the rules for the second largest cluster and so on.

After the network is converted to step transfer function at the final part of the training, each hidden layer neuron performs M-of-N operation, where frequently N=M and thus the operation can be reduced to AND operation. The output neuron combines the partial rules given by hidden neurons for a given output value bin (OR operation). The bias and weights of output neurons are constant; bias = 0.5 each weight = 1, what implements the OR operator.

2.3 Network Training

Only one network can be used for the whole training. That network would consist of 10 output neurons (if we assume, the output value is discretized into 10 bins), each being responsible for a different bin of the output value. However, it is much easier to train 10 separate networks, where each of them has only one output neuron and thus the regression problem can be reduced to 10 classification problems with two classes only (this bin and the rest). In the dataset we replace the value of the output with one of it is within the current bin and with zero otherwise (in a similar way as the input values).

We begin the training with hyperbolic tangent transfer functions at the hidden and output layer neurons. In the final stage of the training the transfer functions are gradually converted to step functions in order to be able to extract simple logical rules from the network. For that purpose a regularization tern t is added to the error function to gradually force the network weights to take the values of -1, 0 or 1.

$$t = k_w * sum((w-1)(w+1)w) + k_b * sum((b-x+0.5)(b-x+1.5)...(b+x-0.5)(b+x+0.5)) \quad (3)$$

where the first part of the equation regularizes the weight values and the second part the biases and x is the number of attributes in the discretized dataset.

From certain point of the training we begin gradually increasing the transfer function slopes as long until we practically obtain step transfer functions, which enables simple extraction of logical rules.

In parallel we use another ("classical") network, which is trained on continuous values and for which we do not increase the slopes of transfer functions. That network is trained only on those vectors, which have the output value within the current bin. From this network we extract logical rules in a method similar to that proposed by Setiono [9]. However, because we add the regularization term to the error function proportional to the sum of the weight squares and we use only vectors from one output bin, we can significantly simplify the system of equations used to extract the logical rules by assuming that all the neurons operates only within the linear part of the hyperbolic transfer function (-1,1). That can be achieved also, because the output value is 10 times less differentiated within one bin than within the whole range.

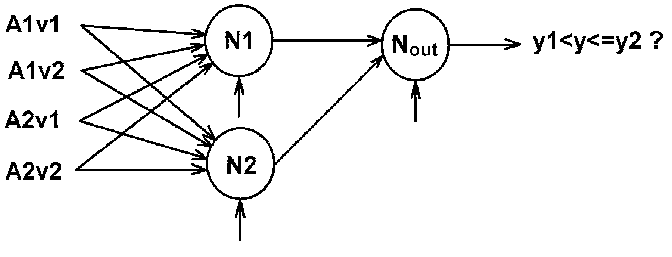

Fig. 2. The single discrete network for rule extraction (for one output bin)

In this way we obtain two sets of neural networks with two different sets of rules. The rules generated by the first set are simpler and frequently are sufficient for the expert. If the experts wants to have more accurate rules, they can use the second model (which generates more complex rules, but still much simpler as the approach presented in [9]). In practice a good combination is to provide the expert with two sets of rules, the two values predicted by the two networks and a final value, which we suggest to use

$$y = (y_1 + 2 * y_2)/3 \quad (4)$$

If several consecutive values of one feature have the same weights assigned at one neuron, then the rule can easily be simplified by merging all the values into one bin.

2.4 Network Test

At the test phase, the test vector is given as an input to all the 10 logical networks and it is assigned to that output bin, which network's output neuron sets its signal to 1. The signals of all other networks' output neurons should produce the value of -1 in response to that vector. However, that is not always the case and sometimes may happen that the networks responses don't allow us to unambiguously assign the vector to one output bin. In this cases we use a standard MLP network trained on the whole dataset. That network predicts some output value of the vector and we assign it to the appropriate bin according to that prediction.

3 Experimental Evaluation

3.1 Experimental Methodology

In this section we compare our method to a regression tree, to a forest of regression trees [6], an MLP network, a hierarchical committee of MLP networks and to the MLP-based method proposed by Setiono [9].

The single MLP neural network had the structure $n - n - 1$, where n is the number of attributes and was trained with the VSS algorithm [16]. In the hierarchical MLP Committee, the whole dataset was split into several clusters, as in the Mixture of Expert approach, however there was a hierarchy of clusters, where the clusters of higher levels contained the clusters of lower levels. Several neural networks were created and trained for each cluster, using bagging to select the vectors. Then an evolutionary algorithm was used to decide how a final decision of the committee must be evaluated based on the test vector properties and on particular neural network responses [7]. The last compared method was a method for extracting logical rules form an MLP network as described in [9], where the sigmoid transfer functions were approximated by a three straight lines and depending on the current point of the sigmoid a proper linear approximation was used for rule generation. The decision tree and forest were constructed as described in our paper presented at the previous HAIS conference [6].

We created software that implements all the methods in C#, Java and Delphi languages and made it available together with the datasets used in the experiments from our web page at [3]. All the methods were run in a 10-fold crossvalidation.

3.2 Datasets

The datasets we used are: two datasets depicting the metallurgical problem; one of them refers to the temperature prediction in the EAF process as described in the introduction and one refers to predicting the amount of carbon to be added in the steel refinement process. The next two datasets are the Crime and Concrete datasets from the UCI Machine Learning Repository.

Concrete Compressive Strength. There are 8 input attributes (variables) in the dataset reflecting the amount of particular substances in the concrete mixture, such as cement, slag, water, etc. The task is to predict the concrete compressive strength. There are

1030 instances in the dataset. The dataset is available from the UCI Machine Learning Repository [4].

Communities and Crime. There are 127 input attributes in the data set, describing various social, economical and criminal factors. The attribute to predict is per capita violent crime. After removing the instances with missing attributes, 121 instances were left. The dataset is available from the UCI Machine Learning Repository [5].

Steel-Carbon. The dataset comes from a real metallurgical process at the phase of refining the melted steel in a ladle arc furnace to achieve desired steel properties. The input variables represent various measured parameters, such as temperature, energy, amount of particular elements in the steel etc. The amount of carbon that should be added to the steel refinement process is the output variable. The data was standardized, only 12 attributes were left from the original dataset with over 100 attributes and the names of 12 input attributes were changed to $x_1 \ldots x_{12}$. There are 1440 instances in the dataset. The dataset is available from [3].

Steel-Temperature. The dataset comes from a real metallurgical process at the phase of melting the steel scrap in the electric arc furnace. The input variables represent various measured parameters, such as temperature, energy, amount of gases, etc. The temperature of the liquid steel is the output variable. The data was standardized, only 14 attributes were left from the original dataset with over 100 attributes and the names of 14 input attributes were changed to $x_1 \ldots x_{14}$. There are 7400 instances in the dataset. The dataset is available from [3].

3.3 Experimental Results

In regression and classification problems, there is no single model which is best for all datasets and models must be chosen for a given task. As it can be seen from table 1, usually the lowest MSE for the large metallurgical datasets is obtained with hierarchical MLP committee. For simple datasets, this model is too complex. However, the purpose of the work was to create a model fitted for rule extraction for large and complex datasets, especially in the application to temperature prediction in electric arc furnace. Thus, it cannot be stated that a model with a lower MSE error is better, because there are situation where a model that can provide simple rule is preferred over a model

Table 1. Experimental results; mean square error (MSE) and standard deviation in 10-fold cross-validation

dataset		single tree	tree forest	discrete Split-MLP	continues Split-MLP	single MLP (values)	single MLP (rules)	hierarchical MLP committee
Concrete	MSE	0.153	0.121	0.133	0.125	0.124	0.125	0.120
	std. dev.	0.020	0.018	0.015	0.020	0.016	0.017	0.014
Crime	MSE	0.35	0.30	0.32	0.28	0.28	0.30	0.28
	std. dev.	0.04	0.04	0.03	0.03	0.03	0.03	0.03
Steel-C	MSE	0.140	0.122	0.120	0.115	0.118	0.128	0.110
	std. dev.	0.017	0.015	0.018	0.014	0.015	0.017	0.011
Steel-Temp	MSE	0.70	0.60	0.56	0.53	0.57	0.60	0.51
	std. dev.	0.08	0.06	0.06	0.05	0.09	0.11	0.05

with low prediction accuracy. In our method we tried to obtain the best possible balance between the two.

3.4 Logical Rules

In this section we discuss the logical rules extracted from the steel-temperature dataset for three methods: the regression tree, the method of Setiono and our method. Because it would require about several pages to print out all the obtained rules, only some chosen examples are presented.

In the Setiono's method, first we have to calculate the position of signals on transfer functions generated by the network as a response to a given vector and obtain the coefficients by which each input feature will be multiplied and then we obtain a rule for each vector in the form:

y=0.297*x1+0.114*x2+0.018*x3+0.275*x4+0.077*x5-0.190*x6+...+0.197

The main problem is the coefficients are different depending on a given vector and they can take a lot of different values, even more than 100 for our dataset. That makes the rules too difficult to understand, especially in limited amount of time, as it is required in the production environment.

The logical rules generated by our method are similar to the rules generated by a regression tree. They are generated in a different way, but they have very similar form; first we decide to what cluster a particular vector belongs (the number of cluster is below 10) and then the final rule has the following form:

if (x1<0.207 and 0.334<x4<0.565) y=0.478*x3+0.232

The difference is that first the number of rule sets is below 10 and second that the rule has the crisp part, which is easier to understand and a very short linear part, which in first approximation maybe also replaced with a constant value.

4 Conclusions

The noticeable difference between a neural network and a decision tree is that the neural network considers all the attributes at each training stage. An univariate decision tree makes a choice of the most important attribute at every node according to some predefined criteria. As a result the split points are not always chosen inn a globally optimal way but in a way that is optimal for the current node and it is not guarantied that the sum of local maxima gives a global maximum. The next issue with decision trees are the predefined shapes of boundaries and the final mapping of leaves, even if done by linear regression models do not cover smoothly all the output space.

For that reason we decided to build the model based on a neural network and to overcome the neural network limitation of providing simple logical rules, especially for regression tasks, we incorporated the split idea known from decision trees. The rules generated by the model achieve only a little poorer accuracy than the best regression model, which we were able to build (committee of MLP networks), while the rule comprehensibility is comparable to that of decision trees with a significantly higher accuracy

at the same time. The system is currently being tested at the real technological data at one of polish steelworks for the purpose of steel temperature prediction in the electric arc furnace.

Acknowledgment. The work was sponsored by the Polish Ministry of Science and Higher Education, projects No. 4866/B/T02/2010/38 and 4421/B/T02/2010/38.

References

1. Corchado, E., et al.: Hybrid intelligent algorithms and applications. Information Science 180(14), 2633–2634 (2010)
2. Abraham, A., Corchado, E., Corchado, J.M.: Hybrid learning machines. Neurocomputing 72(13-15), 2729–2730 (2009)
3. http://www.kordos.com/his.html
4. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases (1998-2011), http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
5. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases (1998-2011), http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime
6. Kordos, M., Blachnik, M., Perzyk, M., Kozłowski, J., Bystrzycki, O., Gródek, M., Byrdziak, A., Motyka, Z.: A Hybrid System with Regression Trees in Steel-Making Process. In: Corchado, E., Kurzyński, M., Woźniak, M. (eds.) HAIS 2011, Part I. LNCS (LNAI), vol. 6678, pp. 222–230. Springer, Heidelberg (2011)
7. Kordos, M., Blachnik, M., Wieczorek, T., Golak, S.: Neural Network Committees Optimized with Evolutionary Methods for Steel Temperature Control. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part I. LNCS (LNAI), vol. 6922, pp. 42–51. Springer, Heidelberg (2011)
8. Kordos, M., Blachnik, M., Wieczorek, T.: Evolutionary Optimization of Regression Model Ensembles in Steel-Making Process. In: Yin, H., Wang, W., Rayward-Smith, V. (eds.) IDEAL 2011. LNCS, vol. 6936, pp. 369–376. Springer, Heidelberg (2011)
9. Setiono, R., Thong, J.: An approach to generate rules from neural networks for regression problems. European Journal of Operational Research 155(1) (2004)
10. Wang, J., et al.: Regression rules extraction from artificial neural network based on least squares. In: 7th Int. Conference on Natural Computation (ICNC), Shanghai (2011)
11. Saitoa, K., Nakano, R.: Extracting regression rules from neural networks. Neural Networks 15, 1279–1288 (2002)
12. Markowska-Kaczmar, U., Mularczyk, K.: GA-Based Rule Extraction from Neural Networks for Approximation. In: Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 141–148 (2006)
13. Kordos, M.: Neural Network Regression for LHF Process Optimization. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) ICONIP 2008. LNCS, vol. 5506, pp. 453–460. Springer, Heidelberg (2009)
14. Blachnik, M., Mączka, K., Wieczorek, T.: A Model for Temperature Prediction of Melted Steel in the Electric Arc Furnace(EAF). In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010. LNCS, vol. 6114, pp. 371–378. Springer, Heidelberg (2010)
15. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for understanding of data. Proceedings of the IEEE 92(5), 771–805 (2008)
16. Kordos, M., Duch, W.: Variable Step Search Algorithm for Feedforward Networks. Neurocomputing 71(13-15), 2470–2480 (2008)