A Hybrid System with Regression Trees in Steel-making Process

Mirosław Kordos[1], Marcin Blachnik[2], Marcin Perzyk[3], Jacek Kozłowski[3], Orestes Bystrzycki[1], Mateusz Gródek[1], Adrian Byrdziak[1], Zenon Motyka[1]

[1] University of Bielsko-Biala, Department of Mathematics and Informatics, Bielsko-Biała, 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
[3] Warsaw University of Technology, Faculty of Production Engineering, Warsaw, Pl. Politechniki 1, Poland;

Abstract. The paper presents a hybrid regresseion model with the main emphasis put on the regression tree unit. It discusses input and output variable transformation, determining the final decision of hybrid models and node split optimization of regression trees. Because of the ability to generate logical rules, a regression tree maybe the preferred module if it produces comparable results to other modules, therefore the optimization of node split in regression trees is discussed in more detail. A set of split criteria based on different forms of variance reduction is analyzed and guidelines for the choice of the criterion are discussed, including the trade-off between the accuracy of the tree, its size and balance between minimizing the node variance and keeping a symmetric structure of the tree. The presented approach found practical applications in the metallurgical industry.

1 Introduction

There are several issues concerning regression tasks in data mining. One of the problems is caused by the fact, that there is usually no clear rule about which data point is an outlier or wrong point and thus should be excluded from the training set. Another problem is the choice of the appropriate regression model (as neural network, support vector machine for regression (SVR), decision tree, etc.) Most of nonlinear models, as neural networks, kernel methods or even k-NN are very powerful and are able to model any shape of decision surface as they are universal approximators. However, using them we lose the possibility to understand in a simple way what they have learned. Another issue is human verification of the models. It is not only required to provide the user with extracted rules, but also the user must be given a chance to provide feedback deciding whether extracted statistical knowledge is in accordance to the specific expert knowledge. The authors faced that kind of problems while building models for optimizing steel production process in two of polish steelworks [1, 2]. The control of the process was critical for the safety and economical reasons. Since hybrid systems are known to perform usually better than single-model systmes [3–5], we used a hybrid system built

of several parallel modules: MLP neural network, support vector regression, multivariate linear regression and decision tree. But the best results were obtained by combining outputs of all the modules as follows:

$$y = \frac{\sum_i y_i w_i}{\sum_i w_i} \tag{1}$$

where y_i is the output predicted by the $i - th$ module and w_i is the weight assigned to this module. The better the module performs on the learning data, the higher weight is assigned to it. A reasonable first choice is to assign the weights that are inversely proportional to the mean squared error of the modules, but in general that is another parameter to optimize. However, the comprehensibility of the model was not less important than the model accuracy; if the operator of the process doesn't understand the predictive system decisions, he is not likely to apply them to the process. Therefore it was necessary to provide decision rules explaining the rationale behind predicting a given value.

Two common groups of models that can address the comprehensibility problems [6] are:

- Rule extraction by sequential covering
- Decision tree induction

Sequential covering is based on extracting decision rules in such a way that each new rule covers new examples not covered by previously defined rules. In other words rules are created starting from the most general to the most specific one. Decision tree induction is based on the divide and conquer paradigm, which defines the hierarchical structure of the learning problem. Decision trees, depending on how tree nodes get split, can be defined as one dimensional, multidimensional (oblique) or mixed. In the first type of trees the nodes of the tree split the dataset based on the value of a single variable, comparing it against a threshold or "belong to" operation in case of symbolic values. The choice of the variables and thresholds is discussed in the following chapters. In multidimensional trees the nodes split the dataset based on a combination of several variables. That can be a linear or quadratic model, or in general any decision making algorithm (LMT [7], NBtree [8], etc.). Mixed trees constitute a combination of one- and multi-dimensional trees [9], in which for each node the best node splitting function is selected. Although mixed and multidimensional trees are more flexible, they lose their comprehensibility, making obtained rules less readable.

One of the features of decision trees is the ability to extract decision rules, which converts the tree structure into a flat list of rules. So each branch of the tree is converted into an independent decision rule. Decision tree algorithms use the divide and conquer concept, making the tree induction process very fast. The complexity of sequential covering algorithms is much higher, so that it becomes one of limitations of the method. On the other hand rules obtained from decision tree tend to be more complex because they all have a common part (at least one common promise defined by the root node) as the tree is a hierarchical structure. This makes the rules less readable, and may require further pruning as in C4.5 rules [10].

Another undesirable property of decision trees is the possible insignificance of the root node and other top level for the interpretation of obtained results, because at the top nodes the criterion function may be highly impure.

As a remedy to that problem we can redefine the criterion function. For that purpose many different criteria have been defined such as entropy based functions: Shannon, Renyi or Tsallis [11] and others. On the other hand instead of defining new criterion functions, typical criteria may be parameterized such that the promise of the top level nodes will become more significant. This can be obtained by not splitting the node symmetrically (i.e. with equal error rate in the two subsets), but rather at the top level nodes the split would be asymmetric, and travelling further down the tree the split would be getting more and more symmetric.

The paper is divided into two theoretic sections. The first section presents a simple data transformation for reducing the influence of outliers for the system and the second section discusses parameterized split criteria function for the regression tree module. The next sections presents empirical results obtained on various real-word datasets and discuss the influence of different parameters on the quality of obtained results, including final results obtained for the metallurgical problem. The last section concludes the paper.

2 Outliers and Data Transformation

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. A good practice is to standardize the data before the training, e.g. according to the following formula:

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

to make the influence of particular inputs independent of their physical range. One dimensional regression trees do not need the standardization of inputs, because they consider only a single attribute at a time. However, they can still benefit from the standardization of the output variable, which can be performed as the first step of outlier removal and sensitivity improvements. 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 the problem, we transfer the data through a hyperbolic tangent function (fig. (2), eq. (3)). The other advantage of the transformation is the automatic reduction of the outliers' influence on the model. We do not consider the outliers as erroneous values and thus do not reject them [1], but rather reduce their influence on the final model, because it is frequently not clear whether a given value is already an outlier or w 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. That approach

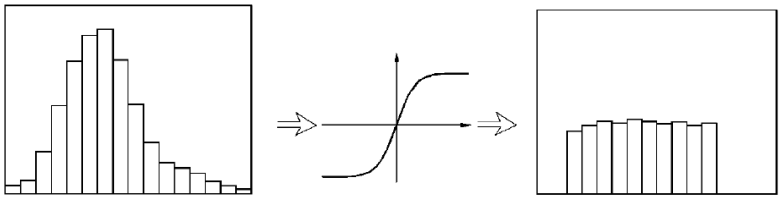


Fig. 1. The idea of transforming output data to uniform distribution

does not work well in the case of multimodal data distribution. That data must be first divided into several single-mode distribution datasets or a more complex transformation function must be used.

After the input and output attributes are standardized, we transform them by $\tanh$ function, as shown in fig.(2) using the formula

$$y = \frac{1 - exp\left(-\beta \cdot y + \theta\right)}{1 + exp\left(-\beta \cdot y + \theta\right)} \tag{3}$$

In practice the θ value could be estimated using the mean value of the output variable y (that is 0 after standardization).

3 Parameterization of split criteria

The parameterization function used to split the data into two children of every node can take various forms. The following pseudo-code shows how the split points are determined: The purpose here is to find such a split point s_0 and a feature (variable) f_0 over

Algorithm 1 Tree optimization pseudocode

Require: $\mathbf{F} = [f_1, f_2, \ldots, f_s]$
Ensure: $\underset{i=1:s}{\forall}$ sizeof$(f_i) \leftarrow p$
  for $i = 1 \ldots s$ **do**
    $f_i = SortFeatureElements(f_i)$
    for $j = 1 \ldots p$ **do**
      $p_L = j/p$
      $p_R = (p - j)/p$
      $v = v_0 - p_L^m \cdot v_L^n - p_R^m \cdot v_R^n$
      if $v \geq q$ **then**
        $q = v$
        $s_0 = j$
        $f_0 = f_i$
      end if
    end for
  end for
  return s_0, f_0

all possible features f_i, which maximize the variance reduction v for each tree node. v_0 is the node variance, i.e. the variance of all vector output values Y in the node. We search for the optimal split point (s_0) iterating over each input feature f_j and each value of that feature j. For that reason the vectors must be sorted in the increasing order of

each feature separately, before the search for the optimal split is attempted. v_L is the variance of the left side of the node (the potential left child) and v_R of the right side. The simplest statement for v could be:

$$v = v_0 - (v_L + v_R) \tag{4}$$

Although, this maximizes variance reduction in a single node, it turns out to be only a local minimum and the entire tree created with this split criterion tends to be large and with poor generalization ability. That happens, because this split usually divides the node in two children with very different number of vectors. One child comprises very few vectors, frequently only one and the other one very many.

The solution is to multiply the variances of each child node by the number of vectors in the child node to enforce the split into two sets with more equal number of vectors.

$$v = v_0 - (p_L^n \cdot v_L^m - p_R^n \cdot v_R^m) \tag{5}$$

Where p is the number of vectors in the given node, p_L is the ratio of the number of vectors in the left child node and the total number of vectors in that node (p) and respectively p_R is the ratio in the right mode. We raised these values to different powers m and n and examine the influence of the powers on the complexity and accuracy of the decision tree. The results are discussed in the experimental result section.

4 Experimental results

4.1 Experimental Methodology

The purpose of the experiments was to find out how the split criteria influence the structure and accuracy of regression trees. We conducted the experiments on about 10 different datasets. However, here we present results only on three of them, because the results on the other datasets showed similar dependencies.

The source code of the software is available from [12].

The experiments described in this chapter were performed with the following parameters:

- minimum node variance: $0.002/\beta^2$
- minimum number of instances in the current node: 2% of the number of instances in the training data
- minimum number of instances in a child node: 0.5% of the number of instances in the training data
- maximum number of levels in the trees: 24

4.2 Results

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 database. We used 687 instances for the training data and 343 instances

for the test data. The dataset is available from the UCI Machine Learning Repository . The results are shown in Fig. 1.

Communities and Crime. There are 120 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. We used 81 instances for the training data and 40 instances for the test data. The dataset is available from the UCI Machine Learning Repository . The results are shown in Fig. 2.

Metallurgical problem. The dataset comes from a real metallurgical process atthe phase of refining the melted steel in a ladle arc furnace to achieve desired steel properties. The inputs 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 represented by variable to precict (C). The data was standardized and the names of 12 input attributes were changed to $x_1 \dots x_{12}$. There are 1440 instances in the data set. We used 960 instances for training and 480 instances for the test dataset. The dataset is available at [13]. The results are shown in Fig. 3.

Output Transformations. As discussed in the previous chapter, we transformed the output column by hyperbolic tangent and sought for the optimal parameter "β in the $\tanh$ equation (3).

The tree was constructed on the transformed data, however, the MSE was measured on the original test data in order to obtain a direct comparison with the results obtained on the original, untransformed data. In order to keep the algorithm properties unchanged, the parameters describing the minimum variance in the current and child nodes were changed so to keep a constant ratio of the mean deviation (square root of the current and child node variance) to the derivative of $\tanh$ function at the zero point (for y=0). It was found that the optimal β was close to one in most cases (table 1.). However, it significantly improved the results only in the case of a single mode distribution. For example, the distribution of the steel data was similar to a sum of three Gaussian distributions with different mean values and therefore this transformation didn't work well in this case. It was found that the transformation didn't have a significant influence on the tree size.

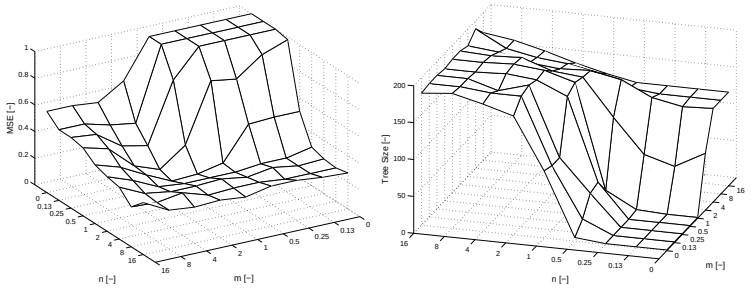
dataset	Desc.	none	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$	$\beta = 5$
Concrete	MSE	0.20	0.20	0.18	0.16	0.15	0.15	1.16	0.21
	Tree size	167	164	157	167	169	177	168	149
Crime	MSE	0.58	0.58	0.56	0.54	0.46	0.35	0.58	0.71
	Tree size	95	95	85	69	85	95	95	65
Steel	MSE	0.14	0.14	0.15	0.14	0.14	0.15	0.23	0.35
	Tree size	121	121	77	95	105	111	107	59

Table 1. Influence of β for the accuracy (for $n = m = 1$)

Forest of Decision Trees. There are ways to improve the prediction ability of a single tree. One of the methods is to create a forest of trees. In our experiments we used a forest of 10 trees. The whole training set consisted each time of a 90% of the total set and the whole model was tested in 10-fold crossvalidation, each time on the remaining

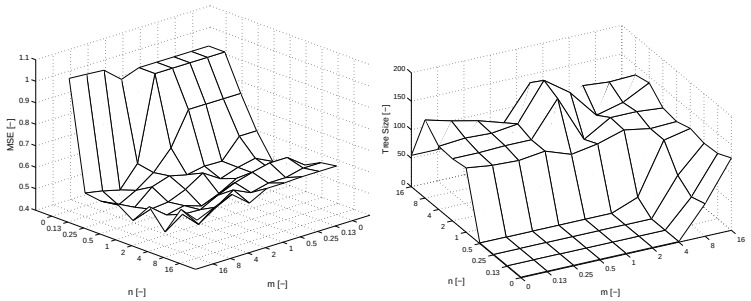
10% of vectors. One third of training vectors were randomly chosen to build each of thee trees. Each tree was then tested on the remaining 2/3 of the training vectors and the inverse of MSE achieved on this set was the quality measure of the tree. Then the final decision y was taken based a weighted average of the values predicted by the 10 trees:

$$y = \left(\sum_{i=1}^{k} \frac{1}{MSE_i} \right)^{-1} \sum_{i=1}^{k} \frac{y}{MSE_i} \tag{6}$$



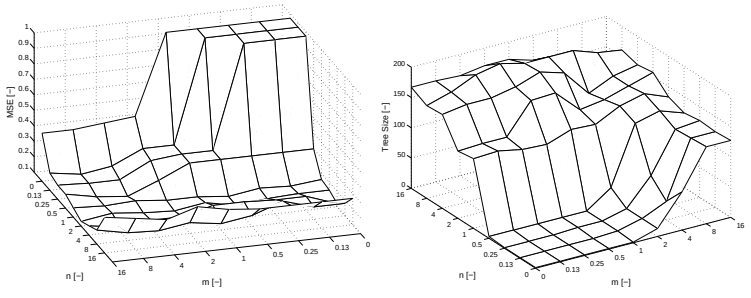
(a) MSE (b) Tree size

Fig. 2. n and m influence on the MSE (a) and on the tree size (b) for Concrete Compressive Strength



(a) MSE (b) Tree size

Fig. 3. n and m influence on the MSE (a) and on the tree size (b) for Communities and Crime dataset



(a) MSE (b) Tree size

Fig. 4. n and m influence on the MSE (a) and on the tree size (b)for Steel dataset

5 Conclusions

The output variable transformation as discussed in section 2 allowed reducing the MSE for most datasetst. In standardized data with Gaussian distribution of the output variable the assumption $\theta = 0$ is correct. For non Gaussian distributions s θ and β should be optimized simultaneously. The decision tree module, although not always achieves the highest accuracy in the whole hybrid system, is very usable because of it easily generates comprehensive logical rules. The goal of the proposed split criteria was to control the influence of variance reduction and the position of the splitting threshold/cut-off. According to the formula (5) increasing n value increases the role of the position in the split. In other words it forces the split threshold to be placed in the middle such that $p_L \approx p_R$. Reducing the value n below m $n < m$ also reduces the importance of that factor and focuses more on the pure variance reduction. To improve the decision tree accuracy the best m and n values could be derived from the formula $m = 2 - n$ where the value of n were in ranges $n \in [0.5, 2]$. Our experiments showed that the most optimal m and n values oscillate around 1 in (4). The results can be further improved with output variable transformation and with tree forest was presented as well as with post-training optimization of the tree.

Acknowledgment

The work was sponsored by the Polish Ministry of Science and Higher Education, projects No. 4866/B/T02/2010/38 and R07/001504.

References

1. M. Kordos, "Neural Network Regression for LHF Process Optimization", LNCS, vol. 5506, pp. 453-460, 2008
2. M. Blachnik, K. Mączka, T. Wieczorek, "A model for temperature prediction of melted steel in the electric arc furnace(EAF), LNCS, vo.l 6614, 2010
3. E. Corchado, et. al., "Hybrid intelligent algorithms and applications", Information Science, vol. 180(14): pp. 2633-2634, 2010
4. A. Abraham, E. Corchado, J. M. Corchado, "Hybrid learning machines", Neurocomputing, vol. 72(13-15), pp. 2729-2730, 2009
5. , M. Wozniak, M. Zmyslony, "Designing Fusers on the Basis of Discriminants - Evolutionary and Neural Methods of Training", HAIS 2010, pp. 590-597
6. W. Duch, R. Setiono, J.Zurada, "Computational intelligence methods for understanding of data", Proceedings of the IEEE, vol. 92, no. 5, pp. 771-805, 2008
7. N. Landwehr, M. Hall, E. Frank, "Logistic Model Trees", Machine Learning, vol. 95, 2005
8. R, Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid", 2nd Int. Conf. on Knowledge Discovery and Data Mining, pp. 202-207, 1996
9. K. Grąbczewski, W. Duch, "Heterogenous forests of decision trees", LNCS, vol. 2415, pp. 504–509, 2002
10. J. R. Quinlan, "Simplifying decision trees", Int. Journal of Man-Machine Studies, vol. 27. no. 3, 1987
11. T. Maszczyk, W. Duch, "Comparison of Shannon, Renyi and Tsallis Entropy used in Decision Trees", LNCS, vol. 5097, pp. 643-651, 2008
12. `www.kordos.com/tree-source.zip`
13. `www.kordos.com/datasets/steel.zip`