Solving Regression by Learning an Ensemble of Decision Rules

Krzysztof Dembczyński[1], Wojciech Kotłowski[1], and Roman Słowiński[1,2]

[1] Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, Poland
{kdembczynski, wkotlowski, rslowinski}@cs.put.poznan.pl
[2] Institute for Systems Research, Polish Academy of Sciences, 01-447 Warsaw, Poland

Abstract. We introduce a novel decision rule induction algorithm for solving the regression problem. There are only few approaches in which decision rules are applied to this type of prediction problems. The algorithm uses a single decision rule as a base classifier in the ensemble. Forward stagewise additive modeling is used in order to obtain the ensemble of decision rules. We consider two types of loss functions, the squared- and absolute-error loss, that are commonly used in regression problems. The minimization of empirical risk based on these loss functions is performed by two optimization techniques, the gradient boosting and the least angle approach. The main advantage of decision rules is their simplicity and good interpretability. The prediction model in the form of an ensemble of decision rules is powerful, which is shown by results of the experiment presented in the paper.

1 Introduction

A *decision rule* is a logical expression of the form: *if* [*conditions*], *then* [*decision*]. If an object satisfies conditions of the rule, then the decision is taken; otherwise no action is performed. A rule can be treated as a simple classifier that gives a constant response for the objects matching the conditions, and abstains from the response for all other objects.

Induction of decision rules has been widely considered in the early machine learning approaches. The most popular algorithms were based on a sequential covering procedure (also known as separate-and-conquer approach) [26, 7, 8, 17]. Apart from the sequential covering, some other approaches to rule induction exist. For instance, the apriori-based algorithms are also used for induction of predictive rules [24, 33]. There are several rule-based approaches of lazy learning type, possibly combined with instance-based methods [10, 18]. Other algorithms based on Boolean reasoning and mathematical programming try to select the most relevant rules – this is the case of Logical Analysis of Data [3]. Let us also notice that decision rule models are strongly associated with rough set approaches to knowledge discovery [27, 31, 20, 32, 19], where also Boolean reasoning has been applied [30]. Such wide interest in decision rules may be explained by their simplicity and good interpretability. It seems, however, that decision trees (e.g. C4.5 [29], CART [6]) are more popular in data mining and machine learning applications. Nevertheless, recently, a growing interest in decision rule models is observed. As an example, let us mention such algorithms as RuleFit [15], SLIPPER [9], Lightweight Rule Induction (LRI) [35], and ensemble of decision rules [1, 2]. All these algorithms can be explained within the framework of *forward stagewise additive modeling* (FSAM) [21], a greedy procedure for minimizing a loss function on the dataset.

Let us notice that there are only few rule induction algorithms tailored to the problem of regression. An example is an extension of LRI [23], in which the decision attribute is discretized and then the problem is solved via classification rules. Another example is RuleFit, which uses FSAM framework explicitly, therefore can utilize a variety of loss functions, including those adapted to regression problems, like squared-error loss.

However, in RuleFit the decision rules are not generated directly – trees are used as base classifiers instead. Rules are produced from each node (interior or terminal) of each resulting tree. This is set up by conjunction of conditions associated with all the edges on the path from the root of the tree to the considered node. Rule ensemble is then fitted by gradient directed regularization [14]. LRI, in turn, uses a specific reweighting schema (cumulative error), similar to Breiman's Arc-xf algorithm [5], which can also be explained in the context of loss function minimization [25]. Single rules are in the form of DNF-formulas.

The algorithm described here, called ENDER (from ENsemble of DEcision Rules), benefits from the achievements in boosting machines [16, 11, 25, 12]. The main contribution of this paper is transmission of these achievements to the ground of a specific weak learner being a decision rule. Similarly to FSAM, our approach is stated as a greedy minimization of a loss function on the training set. However, contrary to RuleFit and LRI, the method introduced in this paper generates simple single rules (conjunctions of elementary conditions) directly, one rule in each iteration of the algorithm. Our approach is also distinguished by the fact of using the same single measure (value of the empirical risk) at all stages of the learning procedure: setting the best cuts (conditions), stopping the rule's growth and determining the response (weight) of the rule; no additional features (e.g. impurity measures, pruning procedures) are considered. Our research includes detailed analysis of the algorithm, both from the theoretical and experimental point of view. We report experiments with two types of loss functions, squared- and absolute-error loss, using two optimization techniques, the gradient boosting [12] and least angle approach [25].

The paper is organized as follows. In section 2, the regression problem is formulated. Section 3 presents a general framework for construction of an ensemble of decision rules. Section 4 is devoted to the problem of a single rule generation. Derivation of particular algorithms for different optimization techniques and loss functions is described in section 5. Section 6 contains experimental results and comparison with other methods. The last section concludes the paper and outlines further research directions.

2 Problem Statement

In the regression problem, the aim is to predict the unknown value of an attribute y (called *decision attribute*, *output* or *dependent variable*) of an object using known joint values of other attributes (called *condition attributes*, *predictors*, or *independent variables*) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The decision attribute is quantitative and it is assumed that $y \in \mathbb{R}$, where $\mathbb{R}$ is a set of real numbers.

The aim is to find a function $F(\mathbf{x})$ that predicts accurately value of y. The accuracy of a single prediction is measured in terms of the *loss function* $L(y, F(\mathbf{x}))$ which is the penalty for predicting $F(\mathbf{x})$ when the actual value is y. The overall accuracy of the function $F(\mathbf{x})$ is measured by the expected loss (*prediction risk*) over the joint distribution of variables $P(y, \mathbf{x})$ for the data to be predicted:

$$R(F) = E_{y\mathbf{x}} L(y, F(\mathbf{x})) = E_{\mathbf{x}}[E_{y|\mathbf{x}} L(y, F(\mathbf{x})].$$

Therefore, the optimal (risk-minimizing) decision function (or Bayes optimal decision) is given by:

$$F^* = \arg\min_F E_{y\mathbf{x}} L(y, F(\mathbf{x})) = \arg\min_F E_{\mathbf{x}}[E_{y|\mathbf{x}} L(y, F(\mathbf{x})]. \quad (1)$$

Since $P(y, \mathbf{x})$ is generally unknown, the learning procedure uses only a set of training examples $\{y_i, \mathbf{x}_i\}_1^N$ to construct $F(\mathbf{x})$ to be the best possible approximation of $F^*(\mathbf{x})$. Usually, it is performed by minimization of the *empirical risk*:

$$R_{\text{emp}}(F) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i)),$$

where function F is chosen from some restricted family of functions.

The regression problem is solved typically by using the squared-error loss:

$$L_{\text{se}}(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2. \quad (2)$$

The Bayes optimal decision for the squared-error loss has the following form:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y|\mathbf{x}} L_{\text{se}}(y, F(\mathbf{x})) = E_{y|\mathbf{x}}(y). \quad (3)$$

It follows that minimization of expected squared-error loss can be seen as estimation of expected value of y for given $\mathbf{x}$. The absolute-error loss is also considered:

$$L_{\text{ae}}(y, F(\mathbf{x})) = |y - F(\mathbf{x})|. \quad (4)$$

The Bayes optimal decision in this case is:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y|\mathbf{x}} L_{\text{ae}}(y, F(\mathbf{x})) = \text{median}_{y|\mathbf{x}}(y). \quad (5)$$

Thus, minimization of expected absolute-error loss can be seen as estimation of median of y for given $\mathbf{x}$. It is often underlined that the absolute-error makes the fitting procedures less sensitive to outliers, because the error grows linearly, not quadratically, with the distance between actual and predicted value.

In order to solve the regression problem, other loss functions can also be employed, for example, Huber loss [22] or ϵ-insensitive error loss, well-known from support vector regression [34]. In this paper, however, we restrict our considerations to the squared- and absolute-error loss only.

3 Learning of an Ensemble of Decision Rules

This section describes the general scheme for decision rule induction. Let us remind that decision rule is a specific classifier having the form of logical expression: *if* [*conditions*], *then* [*decision*].

Let X_j be a value set of attribute j, i.e. the set of all possible values for attribute j. Conditions of the rule consist of elementary expressions of the general form $x_j \in S_j$, where x_j is the value of object $\mathbf{x}$ on attribute j and S_j is some subset of X_j, $j \in \{1, \ldots, n\}$. We assume that in the case of ordered value sets, S_j has the form of the interval $[s_j, \infty)$ or $(-\infty, s_j]$ for some $s_j \in X_j$, so that the elementary expression takes the form $x_j \geq s_j$ or $x_j \leq s_j$. For nominal attributes, we consider elementary expression of the form $x_j = s_j$ or $x_j \neq s_j$. Let $\phi_{S_j}(\mathbf{x}) = I(x_j \in S_j)$, where $I(a)$ is an indicator function, i.e. if a is true then $I(a) = 1$, otherwise $I(a) = 0$. Let further Φ denote the set of elementary expressions constituting the conditions of the rule. Moreover, let $\Phi(\mathbf{x}) = \prod_{\phi \in \Phi} \phi_{S_j}(\mathbf{x})$ be a function that indicates (is non-zero) if an object satisfies conditions of the rule. It is easy to see that conjunction of elementary expressions defines an arbitrary axis-parallel region in the attribute space. Decision, denoted by α, is a real non-zero value assigned to this region. Therefore, we define a decision rule as:

$$r(\mathbf{x}) = \alpha \Phi(\mathbf{x}). \tag{6}$$

Notice that the decision rule takes only two values, $r(\mathbf{x}) \in \{\alpha, 0\}$.

We assume that the optimal decision function $F^*(\mathbf{x})$ is approximated by a linear combinations of M decision rules:

$$F(\mathbf{x}) = \alpha_0 + \sum_{m=1}^{M} r_m(\mathbf{x}), \tag{7}$$

where α_0 is a constant value, which can be interpreted as a default rule, covering the whole attribute space. Construction of optimal combination of rules minimizing the empirical risk is an extremely hard optimization task. That is why we follow here FSAM [21], i.e. the rules are added one by one, greedily minimizing the loss function. We start with the default rule defined as:

$$\alpha_0 = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha). \tag{8}$$

In each next iteration, the new rule is added taking into account previously generated rules. Let $F_{m-1}(\mathbf{x})$ be a prediction function after $m-1$ iterations,

Algorithm 1: Ensemble of decision rules – ENDER

```
input  : set of training examples $\{y_i, \mathbf{x}_i\}_1^N$,
         $M$ – number of decision rules to be generated.
output: default rule $\alpha_0$, ensemble of decision rules $\{r_m(\mathbf{x})\}_1^M$.

$\alpha_0 = \arg\min_\alpha \sum_{i=1}^N L(y_i, \alpha)$; //default rule
$F_0(\mathbf{x}) = \alpha_0$;
for $m = 1$ to $M$ do
    $r_m(\mathbf{x}) = \arg\min_{\Phi,\alpha} \Big\{\sum_{\Phi(\mathbf{x}_i)=1} L(y_i, F_{m-1}(\mathbf{x}_i) + \alpha) +$
                $\sum_{\Phi(\mathbf{x}_i)=0} L(y_i, F_{m-1}(\mathbf{x}_i))\Big\}$;
    $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot r_m(\mathbf{x})$;
end
$ensemble = \{r_m(\mathbf{x})\}_1^M$;
```

consisting of first $m-1$ rules and the default rule. In the m-th iteration, a decision rule is obtained from:

$$r_m(\mathbf{x}) = \arg\min_{\Phi,\alpha} \left\{ \sum_{\Phi(\mathbf{x}_i)=1} L(y_i, F_{m-1}(\mathbf{x}_i) + \alpha) + \sum_{\Phi(\mathbf{x}_i)=0} L(y_i, F_{m-1}(\mathbf{x}_i)) \right\} \quad (9)$$

This is the main step of the algorithm. Different approaches to solution of (9) are described in the next two sections.

It has been shown that in order to improve the accuracy of the ensemble, the base classifiers should be shrunk towards α_0 [21]. That is why we use a shrinkage parameter $\nu \in (0, 1]$ when the ensemble is augmented by the newly generated rule:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot r_m(\mathbf{x}).$$

In other words, values of ν determine the degree to which previously generated rules $r_k(\mathbf{x})$, $k = 1, \ldots, m-1$, affect the generation of the successive one in the sequence, i.e., $r_m(\mathbf{x})$.

It has also been observed that training base classifiers on a subsample of the training set leads to improvement of both accuracy and computational complexity [13]. This is due to the fact that classifiers trained on the random subsamples become diversified and less correlated. We also apply this technique and take subsample of size $\eta \leq N$ randomly drawn with or without replacement from the original training set, when a single rule is generated.

The whole procedure for constructing an ensemble of decision rules is presented as Algorithm 1. We called this procedure ENDER (from ENsemble of DEcision Rules).

4 Generation of a Single Rule

Exact solution of (9) is still computationally hard. That is why, in order to generate a single rule, we are using a fast approximate algorithm. The general scheme of this algorithm is the following. At the m-th iteration:

1. Find Φ_m by minimizing some functional $\mathcal{L}_m(\Phi)$ in a greedy manner (the particular form of the functional depends on the loss function and minimization technique used; see next section):
$$\Phi_m = \arg\min_{\Phi} \mathcal{L}_m(\Phi). \tag{10}$$
2. Find α_m as a solution of the following line-search problem:
$$\alpha_m = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \alpha\Phi_m(\mathbf{x}_i)). \tag{11}$$
In the case of the squared- and absolute-error loss, computation of α_m is straightforward, because analytical expressions for (11) can be given (see next section).

The greedy procedure for finding Φ_m is the following:

– At the beginning, Φ_m is empty (no elementary expressions are specified),
– In the next step, an elementary expression ϕ_{S_j} is added to Φ_m that minimizes $\mathcal{L}_m(\Phi)$ (if it exists). Such expression is searched by consecutive testing of elementary expressions, attribute by attribute. For ordered attributes, let $x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(N)}$ be a sequence of ordered values of attribute j, such that $x_j^{(i-1)} \geq x_j^{(i)}$, for $i = 2, \ldots, N$. Then, each elementary expression $x_j \geq s_j$ or $x_j \leq s_j$, for each $s_j = \frac{x_j^{(i-1)} + x_j^{(i)}}{2}$, is tested. For nominal attributes, we test each expression $x_j = s_j$ or $x_j \neq s_j$, for each value $s_j \in X_j$.
– The above step is repeated until $\mathcal{L}_m(\Phi)$ cannot be decreased.

The output of the whole procedure is the decision rule $r_m(\mathbf{x}) = \alpha_m\Phi_m(\mathbf{x})$. The details of the algorithm for two loss functions and two optimization techniques are given in the next section.

Let us underline that this procedure is very fast and proved to be efficient in computational experiments. The ordered attributes can be sorted once before generating any rule. The procedure for finding optimal Φ resembles the way the decision trees are generated. Here, we look for only one branch instead of the whole decision tree. Moreover, let us notice that minimal value of $\mathcal{L}_m(\Phi)$ is a natural stop criterion in building a single rule and we do not use any other measure (e.g. impurity measures) for choosing the optimal cuts.

5 ENDER Algorithms with Different Optimization Techniques and Loss Functions

We use two minimization techniques that determine the form of $\mathcal{L}_m(\Phi)$.

Gradient boosting [12]. In this method, the rule is fitted to the negative gradient of loss function for each training object:

$$\tilde{y}_i = - \left. \frac{\partial L(y_i, F(\mathbf{x}))}{\partial F(\mathbf{x})} \right|_{F(\mathbf{x}) = F_{m-1}(\mathbf{x}_i)}, \quad i = 1, \ldots, N. \tag{12}$$

The fitting procedure is defined by minimization of the squared-error between rule response and negative gradient:

$$r(\mathbf{x}) = \arg\min_{\Phi, \alpha} \left(\sum_{\Phi(\mathbf{x}_i)=1} (\tilde{y}_i - \alpha)^2 + \sum_{\Phi(\mathbf{x}_i)=0} \tilde{y}_i^2 \right). \tag{13}$$

The term in brackets can be solved for:

$$\alpha = \frac{\sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i}{\sum_{i=1}^N \Phi(\mathbf{x}_i)}, \tag{14}$$

where $\sum_{i=1}^N \Phi(\mathbf{x}_i)$ is the number of objects satisfying $\Phi(\mathbf{x}_i) = 1$. Thus, α is an average of the negative gradient in the region covered by the rule. Putting (14) into (13), expanding the term in the sum for $\Phi(\mathbf{x}_i) = 1$, and performing some simple calculations, we obtain:

$$r(\mathbf{x}) = \arg\min_{\Phi} \left(\sum_{i=1}^N \tilde{y}_i^2 - \frac{1}{\sum_{i=1}^N \Phi(\mathbf{x}_i)} \left(\sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i \right)^2 \right).$$

After removing the first term, which is constant, and taking the square root of the second term (which does not affect the minimization) we get:

$$- \frac{\left| \sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i \right|}{\sqrt{\sum_{i=1}^N \Phi(\mathbf{x}_i)}} \tag{15}$$

which plays the role of $\mathcal{L}_m(\Phi)$, to be minimized with respect to Φ.

Least angle approach [25]. In this method, the rule is also fitted to the negative gradients of loss functions. However, contrary to gradient boosting, the angle between negative gradient vector (12) and rule response vector is minimized. This can be formulated as minimization of the dot product with fixed value (norm) of the rule response:

$$r(\mathbf{x}) = \arg\min_{\Phi} \left(\sum_{\Phi(\mathbf{x}_i)=1} \alpha \tilde{y}_i \right) = |\alpha| \arg\min_{\Phi} \left(\pm \sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i \right) \tag{16}$$

which can be expressed in the following form, independent of α:

$$r(\mathbf{x}) = \arg\min_{\Phi} - \left| \sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i \right|. \tag{17}$$

The term $-\left|\sum_{\Phi(\mathbf{x}_i)=1} \tilde{y}_i\right|$ in (17) plays the role of $\mathcal{L}_m(\Phi)$. Notice the similarity between this term and (15). The latter is divided by the square root of the rule size, which results in more specific rules covering smaller regions in the attribute space. Thus, minimizing the angle results in more general rules.

Below, we present how the above techniques apply to the squared- and absolute-error loss functions.

Squared-error loss. The negative gradient is in this case:

$$\tilde{y}_i = -\left.\frac{\partial L_{se}(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right|_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} = \frac{y_i - F_{m-1}(\mathbf{x}_i)}{2}, \; i = 1, \ldots, N. \quad (18)$$

Putting (18) to (15) or (17), one obtains Φ_m by the gradient boosting or the least angle approach, respectively.

Finally, we find α_m as a solution of the line-search (11). The optimal value is obtained by:

$$\alpha_m = \arg\min_{\alpha} \sum_{i=1}^{N} L_{se}(y_i, F_{m-1}(\mathbf{x}_i) + \alpha\Phi_m(\mathbf{x}_i)) = \frac{\sum_{\Phi(\mathbf{x}_i)=1}(y_i - F_{m-1}(\mathbf{x}_i))}{\sum_{i=1}^{N} \Phi(\mathbf{x}_i)}. \quad (19)$$

It is interesting that gradient boosting gives in this case an exact solution to (9) (but still we have to use the greedy procedure for finding Φ_m). This is due to the fact that putting (19) into (9), after some simple calculations, we get the following expression to be minimized

$$-\frac{\left|\sum_{\Phi(\mathbf{x}_i)=1}(y_i - F_{m-1}(\mathbf{x}_i))\right|}{\sqrt{\sum_{i=1}^{N} \Phi(\mathbf{x}_i)}}. \quad (20)$$

(20) is equivalent to (15) with negative gradient (18) up to the constant $\frac{1}{2}$ that does not affect the solution.

Absolute error loss. In this case, the negative gradient is:

$$\tilde{y}_i = -\left.\frac{\partial L_{ae}(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right|_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} = \operatorname{sgn}(y_i - F_{m-1}(\mathbf{x}_i)), \; i = 1, \ldots, N, \quad (21)$$

and the optimal α_m is obtained by:

$$\alpha_m = \arg\min_{\alpha} \sum_{i=1}^{N} L_{ae}(y_i, F_{m-1}(\mathbf{x}_i) + \alpha\Phi_m(\mathbf{x}_i)) = \operatorname{median}_{\Phi(\mathbf{x}_i=1)}(y_i - F_m(\mathbf{x}_i)). \quad (22)$$

For this loss function, there is no simple and exact solutions to (9) and $\mathcal{L}_m(\Phi)$ has to be determined by gradient boosting or least angle techniques.

6 Experimental Results

We designed an experiment to compare performance of ENDER algorithm with other regression methods. We collected eight benchmark datasets taken from `http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html`.

We tested four types of ENDER algorithm: least angle with absolute- (LA) and squared-error (LS), and gradient boosting with absolute- (GA) and squared-error (GS). We used 200 rules, shrinkage parameter ν set to 0.5, and resampling in which 50% of training examples were drawn without replacement. These parameters were found to work well in a few previous experimental runs on several datasets. In order to perform the comparison, we chose five methods implemented in Weka [36]. These are: linear regression (LR), M5P trees [28] (M5P), additive regression [13] (AR) with decision stumps (DS) and with reduced-error pruning trees (RT), and also bagging [4] (BA) with RT. To perform the experiment in a fair manner, we ran several times these algorithms with different parameters, and the best result of each method for each dataset was reported. This is in fact a real challenge for ENDER algorithms that used one set of parameters for all datasets.

We performed 10-fold cross validation that was repeated 10 times. The measured error rate was root mean squared-error (RMSE). The results are shown in Table 1. For each dataset the best result among ENDER algorithms and among the competitive ones is marked in bold, separately. We compared the best results with each other and tested if the difference is significant. We used paired t-test with confidence level 0.05. If the difference is significant then the better result is marked by '$*$', otherwise by '+'.

In Table 2 we present the won-loss-tied analysis. For results given in Table 1, we tested ENDER algorithms against the competitive ones in order to get how many times there is significant difference (obtained by paired t-test with confidence level 0.05) in favor or against a given method. A "win" indicates that a method was significantly better, a "loss" − significantly worse, and a "tie" indicates the situation when the difference was not significant.

The results show that the best variant of ENDER algorithm is gradient boosting with squared-error loss. Let us remind that this approach gives an

Table 1. Experimental results on eight benchmark datasets.

Classifier	Pyrim	CPU	Boston	Abalone	Bank	Comp.	Calif.	Census
ENDER LS	0.0979	55.092	3.6209	2.2031	0.0341	3.5584	59087.7	31442.2
ENDER LA	**0.0892**	102.277	4.4022	2.399	0.0399	4.2805	62636.9	36569.7
ENDER GS	0.1024	**53.399**$^{+}$	**3.6036**	**2.1994**	**0.033**	**3.105**	**54565.5**	**30851.4**
ENDER GA	0.0904	70.053	3.8305	2.2343	0.034	3.3936	56913.7	32337.3
LR	0.1197	68.404	4.8750	2.2176	0.039	9.8972	69631.5	41605.9
M5P	0.1065	**53.512**	**3.5933**$^{+}$	**2.1376***	**0.0302***	3.166	55705.2	31396.1
AR w/ DS	**0.0880**$^{+}$	56.297	3.8560	2.2318	0.036	3.1943	64697.8	32899.9
AR w/ RT	0.1299	93.343	3.8070	2.1943	0.031	**2.8427***	**48393.2***	30142.6
BA w/ RT	0.1162	71.326	3.7311	2.1483	0.032	3.0756	50420.6	**30027.9***

Table 2. Won-loss-tied analysis of the results obtained in the experiment.

	LR	M5P	AD w/ DS	AR w/ RT	BA w/ RT
ENDER LS	8-0-0	1-6-1	6-2-1	3-5-0	3-5-0
ENDER LA	5-3-0	1-7-0	1-6-1	1-7-0	1-7-0
ENDER GS	8-0-0	4-2-2	7-1-0	3-4-1	3-5-0
ENDER GA	6-1-1	1-7-0	3-1-4	2-5-1	1-6-1

exact solution to (9). The second best is least angle variant with the same loss function. The superiority of the squared-error loss is due to the fact that the chosen measured error rate was RMSE in this experiment. The approaches based on the absolute-error loss obtained rather low results. The best variant of ENDER algorithm can be seen as competitive to the other methods. ENDER GS is distinctly better than linear regression and additive regression with decision stumps, slightly better than M5P and rather comparable to additive regression or bagging with reduced-error pruning trees.

There is, however, still a place for improvements for the ensemble of decision rules. In Figure 1 the RMSE on training and testing set for Abalone and Boston datasets is given for up to 1000 rules. The results are obtained by 30 runs. Datasets were split into training and testing set in the proportion 60% to 40%. One can observe that ENDER is quite stable and results given in Table 1 can be better for higher number of rules. One can also see that in the case of Abalone dataset, ENDER GS has small ability to overfit. The other methods are rather insensitive to overfitting.

7 Conclusions and Future Plans

We introduced a novel algorithm for solving the regression problems that generates an ensemble of decision rules. We derived four different variants of the algorithm incorporating two optimization techniques with two loss functions. Let us underline that there are only few approaches in which decision rules are used for solving this type of prediction problems. Examples of such methods are LRI and RuleFit. Unfortunately, we still did not compare our algorithm to these methods. This is included in our future research plans. Experiment performed in this paper shows that the algorithm is competitive to other methods commonly used in regression problems. Moreover, the algorithm has an additional advantage. Its output is a set of decision rules that are simple and interpretable. However, the ensemble of 1000 rules could lose this advantage. That is why, an additional tool for post-processing of rules is desirable.

References

1. Błaszczyński, J., Dembczyński, K., Kotłowski, W., Słowiński, R., Szeląg, M.: Ensemble of decision rules. *Foundations of Computing and Decision Sciences* **31**, 1 (2006).

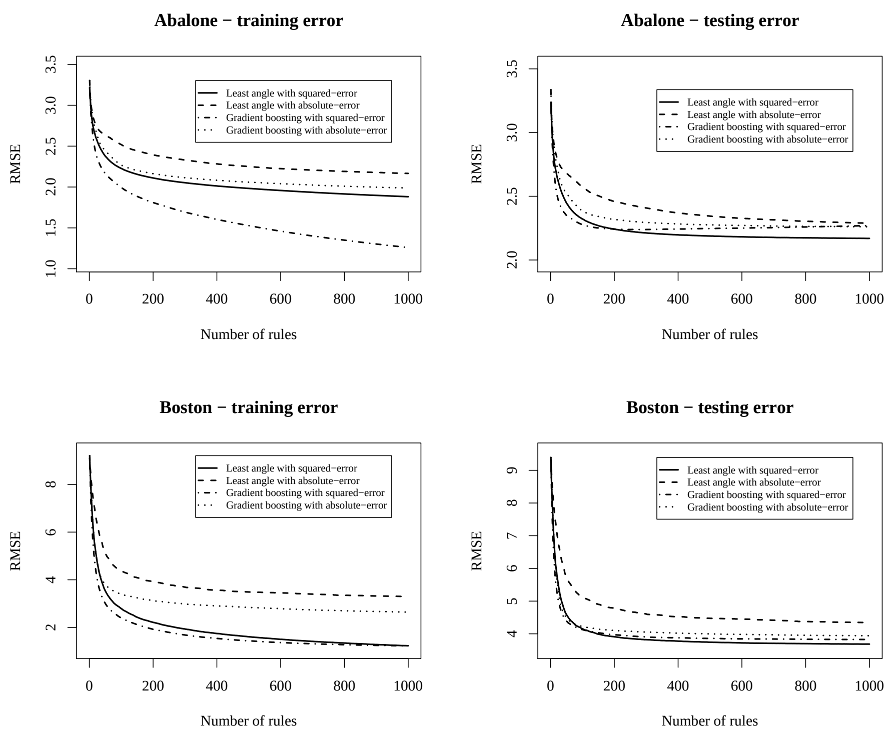

Fig. 1. Errors on Abalone and Boston datasets over 30 runs.

2. Błaszczyński, J., Dembczyński, K., Kotłowski, W., Słowiński, R., Szeląg, M.: Ensembles of Decision Rules for Solving Binary Classification Problems with Presence of Missing Values. *LNAI*, Springer **4259** (2006) 224–234
3. Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I.: An Implementation of Logical Analysis of Data. *IEEE Trans. on Knowledge and Data Engineering* **12** (2000) 292–306
4. Breiman, L.: Bagging Predictors. *Machine Learning* **24** 2 (1996) 123–140
5. Breiman, L.: Arcing classifiers. *The Annals of Statistics* **26** 3 (1998) 801–824.
6. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: *Classification and Regression Trees*. Wadsworth (1984)
7. Clark, P., Nibbet, T.: The CN2 induction algorithm. *Machine Learning* **3** (1989) 261–283
8. Cohen, W. W.: Fast effective rule induction. *Proc. of ICML* (1995) 115–123
9. Cohen, W. W., Singer, Y.: A simple, fast, and effective rule learner. Proc. of AAAI (1999) 335–342
10. Domingos, P.: Unifying instance-based and rule-based induction. *Machine Learning*, **24** (1996) 141–168
11. Friedman, J. H., Hastie, T. and Tibshirani, R.: Additive logistic regression: a statistical view of boosting. *The Annals of Statistics* 2 **28** (2000) 337–407

12. Friedman, J. H.: Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics*, 5 **29** (2001) 1189-1232
13. Friedman, J. H.: Stochastic Gradient Boosting. *Computational Statistics & Data Analysis* **38** 4 (2002), 367–378.
14. Friedman, J. H., Popescu, B. E.: Gradient directed regularization. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (2004)
15. Friedman, J. H., Popescu, B. E.: Predictive Learning via Rule Ensembles. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (2005)
16. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and System Sc.*, 55 **1** (1997) 119–139
17. Fürnkranz, J.: Separate-and-conquer rule learning. *AI Review*, 13 **1** (1996) 3–54
18. Góra, G., Wojna. A.: RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning. *Fundamenta Informaticae* **54** (2002) 369–390
19. Greco, S., Matarazzo, B., Słowiński, R., Stefanowski, J.: An Algorithm for Induction of Decision Rules Consistent with the Dominance Principle. *LNAI*, Springer **2005** (2000) 304–313
20. Grzymala-Busse, J. W.: LERS — A system for learning from examples based on rough sets. In [31], Kluwer Academic Publishers (1992) 3–18
21. Hastie, T., Tibshirani, R., and Friedman, J. H.: *Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer (2003)
22. Huber, P. J.: Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* **35** (1964) 73–101
23. Indurkhya, N., Weiss, S.,: Solving Regression Problems with Rule-based Ensemble Classifiers. *Proc. of the ACM SIGKDD* (2001) 287–292
24. Jovanoski, V., Lavrac, N.: Classification Rule Learning with APRIORI-C. *LNAI*, Springer **2258** (2001) 44–51
25. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Functional gradient techniques for combining hypotheses. In: *Advances in Large Margin Classifiers*, MIT Press (1999) 33–58
26. Michalski, R.S.: A Theory and Methodology of Inductive Learning. In Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.): *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, Tioga Publishing (1983) 83–129
27. Pawlak, Z.: *Rough Sets. Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers (1991)
28. Quinlan, J. R.: Learning with continuous classes. Proc. of the Australian Joint Conference on Artificial Intelligence, Singapore (1992) 343–348.
29. Quinlan, J. R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann (1993)
30. Skowron, A.: Extracting laws from decision tables - a rough set approach. *Computational Intelligence* **11** 371–388
31. Słowiński R. (ed.): *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory*, Kluwer Academic Publishers (1992)
32. Stefanowski, J.: On rough set based approach to induction of decision rules. In Skowron, A., Polkowski, L. (eds.): *Rough Set in Knowledge Discovering*, Physica Verlag (1998) 500–529
33. Stefanowski, J., Vanderpooten, D.: Induction of decision rules in classification and discovery-oriented perspectives. Int. J. on Intelligent Systems **16** (2001) 13–27
34. Vapnik, V.: *The Nature of Statistical Learning Theory (Second Edition).* Springer (1998)
35. Lightweight rule induction. *Proc. of ICML* (2000) 1135–1142
36. Witten, I. H., Frank, E.: *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco (2005)