Ensembles of Decision Rules

Jerzy Błaszczyński, Krzysztof Dembczyński,
Wojciech Kotłowski, Roman Słowiński and Marcin Szeląg
Research Report RA-011/06, Poznań University of Technology

July 28, 2006

Abstract

In most approaches to ensemble methods, base classifiers are decision trees or decision stumps. In this paper, we consider an algorithm that generates an ensemble of decision rules that are simple classifiers in the form of logical expression: *if* [*conditions*], *then* [*decision*]. Single decision rule indicates only one of the decision classes. If an object satisfies conditions of the rule, then it is assigned to that class. Otherwise the object remains unassigned. Decision rules were common in the early machine learning approaches. The most popular decision rule induction algorithms were based on sequential covering procedure. The algorithm presented here follows a different approach to decision rule generation. It treats a single rule as a subsidiary, base classifier in the ensemble. First experimental results have shown that the presented algorithm is competitive with other methods. Additionally, generated decision rules are easy in interpretation. which is not the case of other types of base classifiers.

1 Introduction

Ensemble methods became a very popular approach to classification problems. These methods consist in forming an ensemble of classifiers that are simple learning and classification procedures often referred to as base (or weak) learners. The ensemble members (i.e., base learners or classifiers) are applied to a classification task and their individual outputs are then aggregated to one output of the whole ensemble. The aggregation is computed as a linear combination of outputs. The most popular methods that are used as base learners are decision trees, for example C4.5 [18] or CART [6], and decision stumps (that are one level decision trees). There are several approaches to construction of the ensemble, the most popular are bagging [4] and boosting [19, 9]. These algorithms have proven to be effective in reducing classification error of a base learner. In other words, a committee of low performance learners creates a powerful and quite simple solution for

1

the classification problem. That is why these methods are often treated as off-the-shelf methods-of-choice.

In this paper, we consider an algorithm that generates an ensemble of decision rules that are simple classifiers in the form of logical expression: $if\ [conditions],\ then\ [decision]$. Single decision rule indicates only one of the decision classes. If an object satisfies conditions of the rule, then it is assigned to recommended class. Otherwise the object remains unassigned. Decision rules were common in the early machine learning approaches [1, 7]. The most popular decision rule induction algorithms were based on a sequential covering procedure. This procedure builds rules that cover objects coming from one given decision class only. While buiding these rules, the positive and negative examples are distinguished. The positive examples are those to be covered by the rules. The negative examples are all the others. All objects already covered by rules generated so far are deleted from the set of positive examples. Decision rule models are widely considered in the rough set approaches to knowledge discovery [17, 15, 20, 21]. They are also considered in Logical Analysis of Data [3] where they are called patterns.

The algorithm described here follows a different approach to decision rule generation. It treats a single rule as a subsidiary, base classifier in the ensemble. As it was mentioned above, it is a specific base classifier that indicates only one of the decision classes. In our approach, the ensemble of decision rules is constructed using a variation of forward stagewise additive modeling [11]. Similar technique is also used by Friedman and Popescu [14]. However, one can observe substantial differences between their algorithm and the one presented in this paper. In Friedman and Popescu's algorithm, the decision trees are used as base classifiers, and then each node (interior and terminal) of each resulting tree produces a rule. It is setup by the conjunction of conditions associated with all of the edges on the path from the root to that node. Rule ensemble is then fitted by gradient directed regularization [13]. The algorithm presented here generates rules directly. Single rule is created in each iteration of forward stagewise additive modeling. The rules are then used in a classification procedure by linear combination of their outputs. This simpler way is as efficient as other main machine learning methods. Usually, it is enough to generate around 50 rules to achieve satisfying accuracy and, moreover, the rules are easy in interpretation. Our algorithm is also similar to SLIPPER introduced by Cohen and Singer [8]. The difference is that SLIPPER uses AdaBoost [19] schema to produce an ensemble of decision rules. Let us notice that AdaBoost is a specific case of the forward stagewise additive modeling, so the latter is a more general approach [11].

The paper is organized as follows. In Section 2, the problem is formulated. Section 3 presents the algorithm for construction of an ensembles of decision rules. Section 4 contains experimental results and comparison to other methods. The last section concludes the paper and outlines further

research directions.

2 Problem Statement

Let us define the classification problem in a similar way as in [12, 14]. The aim is to predict the unknown value of an attribute y (sometimes called *output*, *response variable* or *decision attribute*) of an object using the known joint values of other attributes (sometimes called *predictors*, *condition attributes* or *independent variables*) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. We consider binary classification problem, and we assume that $y \in \{-1, 1\}$. In other words, all objects for which $y = -1$ constitute decision class Cl_{-1}, and all objects for which $y = 1$ constitute decision class Cl_1. The goal of a learning task is to find a function $F(\mathbf{x})$ using a set of training examples $\{y_i, \mathbf{x}_i\}_1^N$ that classifies accurately objects to these classes. The optimal classification procedure is given by:

$$F^*(\mathbf{x}) = \arg \min_{F(\mathbf{x})} E_{y\mathbf{x}} L(y, F(\mathbf{x}))$$

where the expected value $E_{y\mathbf{x}}$ is over joint distribution of all variables $(y, \mathbf{x})$ for the data to be predicted. $L(y, F(\mathbf{x}))$ is a loss or cost for predicting $F(\mathbf{x})$ when the actual value is y. The typical loss in classification tasks is:

$$L(y, F(\mathbf{x})) = \begin{cases} 0 & y = F(\mathbf{x}), \\ 1 & y \neq F(\mathbf{x}). \end{cases} \tag{1}$$

The learning procedure tries to construct $F(\mathbf{x})$ to be the best possible approximation of $F^*(\mathbf{x})$.

3 Ensembles of Decision Rules

Forward stagewise additive modeling [11] is a general schema of algorithm that creates an ensemble. The variation of this schema used by Friedman and Popescu [14] is presented as Algorithm 1. In this procedure, $L(y_i, F(\mathbf{x}))$ is a loss function, $f_m(\mathbf{x}, \mathbf{p})$ is the base learner characterized by a set of parameters $\mathbf{p}$ and M is a number of base learners to be generated. $S_m(\eta)$ represents a different subsample of size $\eta \leq N$ randomly drawn with or without replacement from the original training data. ν is so called "shrinkage" parameter, usually $0 \leq \nu \leq 1$. Values of ν determine the degree to which previously generated base learners $f_k(\mathbf{x}, \mathbf{p})$, $k = 1, \ldots, m$, affect the generation of the successive one in the sequence, i.e., $f_{m+1}(\mathbf{x}, \mathbf{p})$.

Classification procedure is performed according to:

$$F(\mathbf{x}) = sign(a_0 + \sum_{m=1}^{M} a_m f_m(\mathbf{x}, \mathbf{p})). \tag{2}$$

3

Algorithm 1: Ensemble of base learners [14]

> **input** : set of training examples $\{y_i, \mathbf{x}_i\}_1^N$,
> M – number of base learners to be generated.
> **output**: ensemble of base learners $\{f_m(\mathbf{x})\}_1^M$.
> $F_0(\mathbf{x}) := \arg\min_{\alpha \in \Re} \sum_{i=1}^N L(y_i, \alpha)$;
> **for** $m = 1 \; to \; M$ **do**
> > $\mathbf{p_m} := \arg\min_{\mathbf{p}} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + f(\mathbf{x}_i, \mathbf{p}))$;
> > $f_m(\mathbf{x}) = f(\mathbf{x}, \mathbf{p_m})$;
> > $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot f_m(\mathbf{x})$;
>
> **end**
> $ensemble = \{f_m(\mathbf{x})\}_1^M$;

In other words, it is a linear classifier in a very high dimensional space of derived variables that are highly nonlinear functions of the original predictor variables $\mathbf{x}$. These functions are generated by base learners, for example, they are decision trees. Parameters $\{a_m\}_0^M$ can be obtained in many ways. For example, they can be set to fixed values (for example, a_0=0 and $\{a_m = 1/M\}_1^M$), computed by some optimization techniques (see, for example, [13]), fitted in cross-validation experiments or estimated in a process of constructing the ensemble (like in AdaBoost [19]).

According to Friedman and Popescu [14], several ensemble methods represent special cases of Algorithm 1. Bagging method [4], for example, may be represented by this algorithm and classification procedure (2) by setting $\nu = 0$, using resampling with replacement and η is given by a user, $a_0 = 0$ and $\{a_m = 1/M\}_1^M$. Random Forest [5] is a specific variant of bagging, where the base learner is constructed by randomized decision tree induction algorithm. AdaBoost uses exponential loss, $L(y, F(\mathbf{x})) = exp(-y \cdot F(\mathbf{x}))$, for $y \in \{-1, 1\}$, and corresponds to Algorithm 1 by setting $\nu = 1$ and $S_m(\eta)$ to be a whole set of training examples. In AdaBoost, as it was mentioned above, $\{a_m\}_0^M$ are tuned during the process of constructing an ensemble.

Let us consider a base classifier that is a decision rule, the definition of which is as follows. Decision rule is a logical statement: $if \; [conditions]$, $then \; [decision]$. Condition part of a decision rule is represented by a complex $\Phi = \phi_1^\propto \wedge \phi_2^\propto \wedge \ldots \wedge \phi_t^\propto$, where $\phi^\propto$ is a selector and t is a number of selectors in the complex, also referred to as a length of the rule. Selector $\phi^\propto$ is defined as $x_j \propto v_j$, where v_j is a single value or a subset of values from the domain of the j-th attribute; and $\propto$ is specified as $=, \in, \geq$ or $\leq$, depending on the characteristic of the j-th attribute. In other words, complex Φ is a set of selectors that allows to select a subset of objects. Objects covered by complex Φ are denoted by $cov(\Phi)$ and referred to as cover of a complex Φ. Decision part of a rule indicates one of the decision classes and is denoted by $d(\mathbf{x}) = -1$ or $d(\mathbf{x}) = 1$. Let as denote a rule by $r(\mathbf{x}, \mathbf{c})$, where $\mathbf{c}$ represents

---
Algorithm 2: Ensemble of decision rules
---
input : set of training examples $\{y_i, \mathbf{x}_i\}_1^N$,
$\quad\quad\quad$ M – number of decision rules to be generated.
output: ensemble of decision rules $\{r_m(\mathbf{x})\}_1^M$.
$F_0(\mathbf{x}) := \arg\min_{\alpha \in \{-1,1\}} \sum_{i=1}^N L(y_i, \alpha)$; or $F_0(\mathbf{x}) := 0$; //default rule
$F_0(\mathbf{x}) := \nu \cdot F_0(\mathbf{x})$;
for $m = 1$ *to* M **do**
$\quad\quad$ $\mathbf{c} := \arg\min_{\mathbf{c}} \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$;
$\quad\quad$ $r_m(\mathbf{x}) = r(\mathbf{x}, \mathbf{c})$;
$\quad\quad$ $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot r_m(\mathbf{x})$;
end
$ensemble = \{r_m(\mathbf{x})\}_1^M$;
---

both complex and decision of the rule, $\mathbf{c} = (\Phi, d(\mathbf{x}))$. Then, the output of the rule may be defined as follows:

$$r(\mathbf{x}, \mathbf{c}) = \begin{cases} d(\mathbf{x}) & \mathbf{x} \in cov(\Phi), \\ 0 & \mathbf{x} \notin cov(\Phi). \end{cases} \quad (3)$$

The loss of a single decision rule takes a specific form:

$$L(y, r(\mathbf{x}, \mathbf{c})) = \begin{cases} 0 & y \cdot r(\mathbf{x}, \mathbf{c}) = 1, \\ 1 & y \cdot r(\mathbf{x}, \mathbf{c}) = -1, \\ l & r(\mathbf{x}, \mathbf{c}) = 0, \end{cases} \quad (4)$$

where $0 \geq l \geq 1$ is a penalty for specificity of the rule. This means, the lower the value of l, the smaller the number of objects covered by the rule from the opposite class.

The ensemble algorithm creating decision rules (presented as Algorithm 2) is an extension of Algorithm 1 suited for this task. In the algorithm, in each consecutive iteration m we augment the function $F_{m-1}(\mathbf{x})$ by one additional rule $r_m(\mathbf{x})$ weighted by shrinkage parameter ν. This gives a linear combination of rules $F_m(\mathbf{x})$. The additional rule $r_m(\mathbf{x}) = r(\mathbf{x}, \mathbf{c})$ is chosen to minimize $\sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$. $F_0(\mathbf{x})$ corresponds to the default rule in the ensemble generation process. It is set to $F_0(\mathbf{x}) := \arg\min_{\alpha \in \{-1,1\}} \sum_i^N L(y_i, \alpha)$ (i.e., it corresponds to the default rule indicating the majority class) or there is no default rule (then $F_0(\mathbf{x}) := 0$). The default rule is taken with the same "shrinkage" parameter ν as all other rules.

The loss of the linear combination of rules $F_m(\mathbf{x})$ takes the following form in the simplest case:

$$L(y, F_m(\mathbf{x})) = \begin{cases} 0 & y \cdot F_m(\mathbf{x}) > 0, \\ 1 & y \cdot F_m(\mathbf{x}) < 0, \\ l & y \cdot F_m(\mathbf{x}) = 0. \end{cases} \quad (5)$$

Nevertheless, the interpretation of l in the above definition is not as easy as in the case of a single rule. It depends on all other parameters used in Algorithm 2. $L(y, F_m(\mathbf{x}))$ takes value equal to l in two cases. The first case is, when $F_0(\mathbf{x})$ is set to zero (there is no default rule) and no rule generated in m iterations covers object $\mathbf{x}$. The second case is when rules covering object $\mathbf{x}$ indicate equally two classes Cl_{-1} and Cl_1. The interpretation of l is similar to the case of a single rule, when $F_0(\mathbf{x})$ is set to zero and $\nu = 1/M$, for example. Note that $\nu = 1/M$ means that each next rule is more important than all previously generated.

Another possible formulation for the loss function are as follows (for a wide discussion on different formulas for loss function see [11]):

$$L(y, F_m(\mathbf{x})) \;=\; \frac{1}{1 - \beta \cdot exp(y \cdot F_m(\mathbf{x}))}, \tag{6}$$

$$L(y, F_m(\mathbf{x})) \;=\; \begin{cases} 0 & y \cdot F_m(\mathbf{x}) \geq 1, \\ 1 - y \cdot F_m(\mathbf{x}) & y \cdot F_m(\mathbf{x}) < 1, \end{cases} \tag{7}$$

$$L(y, F_m(\mathbf{x})) \;=\; exp(\beta \cdot y \cdot F_m(\mathbf{x})), \tag{8}$$

$$L(y, F_m(\mathbf{x})) \;=\; \log(1 + exp(\beta \cdot y \cdot F_m(\mathbf{x})). \tag{9}$$

Algorithm 2 is characterized by parameters that make it flexible. For example, one can parameterize it to obtain a procedure similar to sequential covering. This procedure in each phase builds rules that cover objects coming from one given decision class only. In each phase, the positive and negative examples are distinguished. The positive examples are those to be covered by the rules. The negative examples are all others. In the given phase, all objects already covered by rules generated so far are deleted from the set of positive examples. Usually, the output of the procedure is a decision list (i.e., ordered list of rules from the most important to the less important one). The classification is performed in such a way that it is sequentially checked whether $\mathbf{x}$ is covered by rules from the list. If $\mathbf{x}$ is covered by the first rule, then the class indicated by this rule is the output of the classification. If not, then the next rule from the list is checked, until the procedure arrives to the default rule indicating the majority class. To parametrize the Algorithm 2 to obtain sequential covering procedure one should choose a specific heuristic algorithm for creating single rule, set $F_0(\mathbf{x}) = 0$, and $\nu = 2^{M-m}$. Such value of ν means that the rule first generated is more important than all rules generated later. To conduct classification consistent with the decision list, it is performed according to (2) by setting a_0 to indicate the majority class and $a_m = 2^{M-m+1}$.

To perform our experiments, we have used simple greedy heuristic to construct a single decision rule. It consists in searching for $\mathbf{c}$ such that $L_m = \sum_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$ is minimal. At the beginning, the complex contains an universal selector (i.e., selector that covers all objects). In the next step, a new selector is added to the complex and the decision of

the rule is set. The selector and the decision are chosen to give the minimal value of L_m. This step is repeated until L_m is minimized. Additionally, another stop criterion can be introduced. It may be the length of the rule.

As the results show in the next section, it is enough to generate around fifty rules. Such a number of rules gives satisfying classification accuracy and allows to interpret them. Let us notice, however, that the whole procedure may generate the same (or very similar) rule several times. For the interpretation purposes, it will be necessary to find a method that cleans and compresses the set of rules. Construction of such a method is included in our future plans.

4   Experimental Results

We designed an experiment to compare performance of the ensemble of decision rules to other classifiers. To conduct this experiment we implemented presented here method using Weka package [22]. Other methods that were used in experiment were also implemented in Weka and they are included in the standard Weka package. We used in the experiment a wide range of methods starting from regular classifiers: naive Bayes, logistic regression (Logistic), normalized Gaussian radial basis function network (RBF Network), sequential minimal optimization algorithm for training linear support vector machines (SMO), instance based learning (IBk), C4.5 trees (J48) and PART rules. We also compared our algorithm to classifiers trained in boosting and bagging. Specifically, we used AdaBoost with decision stumps, reduced-error pruning trees (REPTrees), C4.5 (J48) and PART rules. The same classifiers exempt for decision stumps were used in bagging. In Logistic Boost we used two classifiers: decision stumps and REPTrees. Boosting and bagging were always set to perform fifty iterations. Another method to compare was random forest, also used with fifty iterations. Other settings were kept to default values suggested by Weka.

We set the paremeters of the ensemble of decision rules as follows. We have decided to generate default rule indicating the majority class. The function (6) with $\beta = 1$ was used as the loss functions. The "shrinkage" parameter was set to $\nu = 0.5$. Each rule was generated from the sample drawn with replacement (bootstrap sample) for $\eta = N$. We have decided to generate fifty rules. The classification is performed according to (2), where $a_0 = F_0(\mathbf{x})$ and $\{a_m\}_1^M$ are set to 1. Precise information about values of classifiers' parameters is presented in Table 1.

For purpose of the experiment, we used six data sets taken from UCI [16] repository of machine learning data sets. These sets contain two-class classification problems with no missing values. The data sets that we chosen are presented in Table 2. To estimate classifiers error rates, we used leaving-one-out technique. Results of the experiment are presented in Tables 3–5.

7

Table 1: Classification methods and parameters that were used.

Classifier	Abbrev.
NaiveBayes	NB
Logistic -R 1.0E-8 -M -1	Log
RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1	RBFN
SMO -C 1.0 -E 1.0 -G 0.01 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1	SMO
IBk -K 5 -W 0	IBL
AdaBoostM1 -P 100 -S 1 -I 50 -W DecisionStump	AB DS
AdaBoostM1 -P 100 -S 1 -I 50 -W REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1	AB REPT
AdaBoostM1 -P 100 -S 1 -I 50 -W J48 -C 0.25 -M 2	AB J48
AdaBoostM1 -P 100 -S 1 -I 50 -W PART -M 2 -C 0.25 -Q 1	AB PART
Bagging -P 100 -S 1 -I 50 -W REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1	BA REPT
Bagging -P 100 -S 1 -I 50 -W J48 -C 0.25 -M 2	BA J48
Bagging -P 100 -S 1 -I 50 -W PART -M 2 -C 0.25 -Q 1	BA PART
LogitBoost -P 100 -F 0 -R 1 -L -1.8e308 -H 1.0 -S 1 -I 50 -W DecisionStump	LB DS
LogitBoost -P 100 -F 0 -R 1 -L -1.8e308 -H 1.0 -S 1 -I 50 -W REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1	LB REPT
J48 -C 0.25 -M 2	J48
RandomForest -I 50 -K 0 -S 1	RF
PART -M 2 -C 0.25 -Q 1	PART
Ensemble of Decision Rules, $L(y, F_m(\mathbf{x})) = \frac{1}{1 - exp(y \cdot F_m(\mathbf{x}))}$, $M = 50$, bootstrap sample $\eta = N$, $\nu = 0.5$, $a_0 = F_0(\mathbf{x})$, $\{a_m\}_1^M = 1$	EDR

In those tables we show performance of classifiers on factors (given as percents): correctly classified examples (C), true positive in class +1 (TP +1), precision in class +1 (P +1), true positive in class -1 (TP -1), precision in class -1 (P -1). First results of our algorithm are promising. It is comparable to other methods. In one case it gave the best result in terms of correctly classified examples (ionosphere) and in terms of precision (credit-g). The rest of results did not differ importantly from results of other methods. Even more, they were comparable to the best results for each of considered problems. It is worth noting that C4.5 trees and PART rules generated in fifty iterations of boosting and bagging are more complex classifiers than decision stumps and ensemble of decision rules (also generated in fifty iterations). This justifies worse results of our classifier for some of problems, for example for kr-vs-kp (see Table 5). We can obtain better results by increasing the number of iterations.

Further experiments on ensembles of decision rules with different number of rules and on other classification problems are planned.

Table 2: Number of attributes and objects for data sets included in the experiment.

Data set	Attributes	Obj. class -1	Obj. class 1
German Credit (credit-g)	21	300	700
Pima Indians Diabetes (diabetes)	9	268	500
Heart Statlog (heart-statlog)	14	120	150
J. Hopkins University Ionosphere (ionosphere)	35	126	225
King+Rook vs. King+Pawn on a7 (kr-vs-kp)	37	1527	1669
Sonar, Mines vs. Rocks (sonar)	61	97	111

5 Conclusions and Future Plans

We have described a general algorithm creating an ensemble of decision rules. First experimental results show that the algorithm is comparable to other methods that are popular in machine learning. Moreover, the algorithm has also an additional advantage. Its output is a set of decision rules that are easy in interpretation. The idea of the algorithm comes from considerations on Friedman and Popescu's paper [14], where rules are obtained from decision trees. As we have started working on our algorithm, we have also found another similar approach to decision rule generation proposed by Cohen and Singer [8]. The algorithm presented here generates directly (not from a decision trees) decision rules and is more flexible in comparison to [8]. Unfortunately, we had not yet managed to compare our algorithm to these two approaches. We will try to make this comparison in the future. Additionally, our research plans include deep insight into the algorithm, from the theoretical and experimental point of view. We plan to conduct experiments with other types of loss function, and to check how the algorithm works with different values of other parameters. We also plan to verify other heuristics to generate single rule. An interesting issue may be to randomize the process of generating single rule as it is done in random forest. The classification, in this paper, was performed by setting $\{a_m\}_0^M$ to fixed values. We want to check some other solutions, also some optimizations techniques of these parameters. As it was mentioned above the set of rules generated by the ensemble should be cleaned and compressed. Construction of a method preparing a set of rules that is easy in interpretation is also included in our future research plans.

Table 3: Classification results in percents [%], part 1; C indicates *Correctly classified examples*, TP +1 *True Positive* in class +1, P +1 *Precision* in class +1, TP -1 *True Positive* in class -1, P -1 *Precision* in class -1.

Classifier	credit-g					diabetes				
	C	TP +1	P +1	TP -1	P -1	C	TP +1	P +1	TP -1	P -1
NB	75.1	86.4	79.7	48.7	60.6	75.7	84	79.7	60.1	66.8
Log	75.1	86.3	79.8	49	60.5	**77.7**	88.6	79.5	57.5	73
RBFN	72.6	86.6	77.1	40	56.1	73.7	85.2	76.9	52.2	66.4
SMO	74.1	86.6	78.6	45	59	76.8	**89.8**	78	52.6	**73.4**
IBL	73.6	89.1	76.8	37.3	59.6	74.1	83.2	78.3	57.1	64.6
AB DS	71	84.1	76.7	40.3	52.2	76.7	76	79.8	59.3	69.4
AB REPT	73.1	84.7	78.5	46	56.3	73.6	80.8	79.1	60.1	62.6
AB J48	73.7	85.9	78.6	45.3	57.9	73.2	80.8	78.6	59	62.2
AB PART	73.7	85.1	78.9	47	57.6	76.3	84.4	80.2	61.2	67.8
BA REPT	74.9	87	79.2	46.6	60.6	75.4	83.4	79.7	60.4	66.1
BA J48	73.5	86.6	78	43	58.8	76.3	84	**80.5**	61.9	67.5
BA PART	76.1	87.9	80	48.7	63.2	75.3	85.8	78.3	55.6	67.7
LB DS	**76.4**	87.9	**80.3**	**49.7**	63.7	74.7	84.8	78.2	56	66.4
LB REPT	71.5	86.3	76.2	37	53.6	75.4	83.4	79.7	60.4	66.1
J48	70.9	84.7	76.3	38.7	52	73.8	81.6	78.9	59.3	63.3
RF	75.7	89.1	78.9	44.3	63.6	75.3	84.4	79	58.2	66.7
PART	69.9	80.3	77.5	45.7	49.8	67.7	69	78.8	**65.3**	53
EDR	75.3	**90.4**	77.9	40	**64.2**	75.5	86	78.5	56	68.2

References

[1] 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

[2] Booker, L. B., Goldberg, D. E., Holland, J. F.: Classifier systems and genetic algorithms. In Carbonell, J. G. (ed.): Machine Learning. Paradigms and Methods. The MIT Press, Cambridge, MA (1990) 235–282

[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.: Random Forests. Machine Learning **45** 1 (2001) 5–32

[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., Singer, Y.: A simple, fast, and effective rule learner. Proc. of 16th National Conference on Artificial Intelligence (1999) 335–342

[9] Friedman, J. H., Hastie, T. and Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Dept. of Statistics, Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), August (1998)

[10] Friedman, J. H., Popescu, B. E.: Importance Sampled Learning Ensembles. Dept. of Statistics, Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), September (2003)

[11] Friedman, J. H., Hastie, T., Tibshirani, R.: Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer (2003)

[12] Friedman, J. H.: Recent advances in predictive (machine) learning. `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), November (2003)

[13] Friedman, J. H., Popescu, B. E.: Gradient directed regularization. Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), February (2004)

[14] Friedman, J. H., Popescu, B. E.: Predictive Learning via Rule Ensembles. Dept. of Statistics, Stanford University Technical Report, `http://www-stat.stanford.edu/~jhf/` (last access: 1.05.2006), February (2005)

[15] Grzymala-Busse, J. W.: LERS — A system for learning from examples based on rough sets. In Słowiński, R. (ed.): Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers (1992) 3–18

[16] Newman, D. J., Hettich, S., Blake, C. L., Merz, C. J. (UCI) Repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (last access: 01.05.2006), Dept. of Information and Computer Sciences, University of California, Irvine (1998)

[17] Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, (1991)

[18] Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)

[19] Schapire, R. E., Freund, Y., Bartlett, P., Lee, W. E.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics **26** 5 (1998) 1651–1686

[20] Skowron, A.: Extracting laws from decision tables - a rough set approach. Computational Intelligence **11** 371–388

[21] Stefanowki, J.: On rough set based approach to induction of decision rules. In Skowron, A. and Polkowski, L. (eds): Rough Set in Knowledge Discovering, Physica Verlag, Heidelberg (1998) 500–529

[22] Witten, I. H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco (2005)

Table 4: Classification results in percents [%], part 2; C indicates *Correctly classified examples*, TP +1 *True Positive* in class +1, P +1 *Precision* in class +1, TP -1 *True Positive* in class -1, P -1 *Precision* in class -1.

Classifier	heart-statlog					ionosphere				
	C	TP +1	P +1	TP -1	P -1	C	TP +1	P +1	TP -1	P -1
NB	**83**	**86.7**	83.3	78.3	**82.5**	82.6	86.5	71.2	80.4	91.4
Log	**83**	**86.7**	83.3	78.3	**82.5**	89.2	78.6	90	95.1	88.8
RBFN	81.5	84.7	82.5	77.5	80.2	91.7	**90.5**	87	92.4	**94.5**
SMO	**83**	86	83.8	79.2	81.9	88	71.4	93.8	97.3	85.9
IBL	80	82.7	81.6	76.7	78	85.5	63.5	94.1	97.8	82.7
AB DS	82.6	86	83.2	78.3	81.7	93.2	84.1	96.4	98.2	91.7
AB REPT	76.7	76.7	80.4	76.7	72.4	91.7	84.9	91.5	95.6	91.9
AB J48	78.9	82	80.4	75	76.9	94	85.7	97.3	98.7	92.5
AB PART	**83**	84.7	**84.7**	**80.8**	80.8	92.6	84.9	93.9	96.9	92
BA REPT	82.6	**86.7**	82.8	77.5	82.3	91.2	84.9	89.9	94.7	91.8
BA J48	81.1	84.7	81.9	76.7	80	93.4	82.5	**99**	**99.6**	91.1
BA PART	81.9	84	83.4	79.2	79.8	92.9	84.1	95.5	97.8	91.7
LB DS	81.1	84.7	81.9	76.7	80	91.5	81.7	93.6	96.9	90.5
LB REPT	79.3	84.7	79.4	72.5	79.1	91.5	88.9	87.5	92.9	93.7
J48	75.2	77.3	77.9	72.5	79.9	87.5	81	83.6	91.1	89.5
RF	82.2	85.3	83.1	78.3	81	93.4	86.5	94.8	97.3	92.8
PART	77	78.7	79.7	75	73.8	89.5	81.7	88	93.8	90.2
EDR	82.2	85.3	83.1	78.3	81	**94.3**	88.1	95.7	97.8	93.6

Table 5: Classification results in percents [%], part 3; C indicates *Correctly classified examples*, TP +1 *True Positive* in class +1, P +1 *Precision* in class +1, TP -1 *True Positive* in class -1, P -1 *Precision* in class -1; "-" means that no results were obtained.

Classifier	kr-vs-kp					sonar				
	C	TP +1	P +1	TP -1	P -1	C	TP +1	P +1	TP -1	P -1
NB	87.9	89.7	87.5	86	88.4	67.3	81.4	61.2	55	77.2
Log	97.7	97.6	97.9	97.7	97.4	75.5	70.1	76.5	80.2	75.4
RBFN	84.3	86.2	84.2	82.3	84.5	71.2	72.2	68	70.3	74.2
SMO	95.5	95.6	95.8	95.4	95.2	76.9	71.1	77.5	82	76.5
IBL	96.6	98.4	95.1	94.5	98.2	82.2	76.3	84.1	87.4	80.8
AB DS	94.1	96.9	92.2	91	96.5	83.7	79.4	84.6	87.4	82.9
AB REPT	99.4	99.5	99.3	99.3	99.5	80.3	73.2	82.6	86.5	78.7
AB J48	**99.7**	**99.8**	99.6	99.5	99.7	85.6	78.4	**89.4**	**91.9**	82.9
AB PART	**99.7**	99	**99.6**	**99.5**	**99.9**	85.6	**82.5**	86	88.3	**85.2**
BA REPT	99.1	99.4	98.9	98.8	99.3	76.4	67	79.3	84.7	74.6
BA J48	99.4	99.5	99.3	99.3	99.5	80.3	73.2	82.6	86.5	78.7
BA PART	99.5	99.6	99.4	99.3	99.5	83.7	78.4	85.4	88.3	82.4
LB DS	96	96.4	95.9	95.5	96	83.2	78.4	84.4	87.4	82.2
LB REPT	-	-	-	-	-	71.6	64.9	71.6	77.5	71.7
J48	99.5	99.6	99.5	**99.5**	99.5	68.3	66	66	73	73
RF	99.4	99.6	99.2	99.1	99.5	**86.1**	80.4	88.6	91	84.2
PART	99	99.3	98.9	98.8	99.2	73.6	74.2	70.6	73	76.4
EDR	94.1	97	92.1	91	96.5	80.7	73.2	83.5	87.4	78.9