Data Min Knowl Disc (2010) 21:52–90
DOI 10.1007/s10618-010-0177-7

ENDER: a statistical framework for boosting decision rules

Krzysztof Dembczyński · Wojciech Kotłowski · Roman Słowiński

Received: 30 March 2009 / Accepted: 18 March 2010 / Published online: 30 April 2010

Abstract Induction of decision rules plays an important role in machine learning. The main advantage of decision rules is their simplicity and human-interpretable form. Moreover, they are capable of modeling complex interactions between attributes. In this paper, we thoroughly analyze a learning algorithm, called ENDER, which constructs an ensemble of decision rules. This algorithm is tailored for regression and binary classification problems. It uses the boosting approach for learning, which can be treated as generalization of sequential covering. Each new rule is fitted by focusing on examples which were the hardest to classify correctly by the rules already present in the ensemble. We consider different loss functions and minimization techniques often encountered in the boosting framework. The minimization techniques are used to derive impurity measures which control construction of single decision rules. Properties of four different impurity measures are analyzed with respect to the trade-off between misclassification (discrimination) and coverage (completeness) of the rule. Moreover, we consider regularization consisting of shrinking and sampling. Finally,

Responsible editor: Johannes Fürnkranz and Arno Knobbe.

K. Dembczyński (✉) · W. Kotłowski · R. Słowiński
Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland
e-mail: kdembczynski@cs.put.poznan.pl

W. Kotłowski
e-mail: wkotlowski@cs.put.poznan.pl

R. Słowiński
e-mail: rslowinski@cs.put.poznan.pl

R. Słowiński
Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland

we compare the ENDER algorithm with other well-known decision rule learners such as SLIPPER, LRI and RuleFit.

Keywords Decision rules · Impurity measures · Ensemble · Boosting · Forward stagewise additive modeling

1 Introduction

A *decision rule* is a simple logical pattern of the form:

if *condition* then *decision*.

The condition part of the rule is a conjunction of elementary conditions defined on values of particular attributes, and the decision part specifies a value of the function being learned under the stated conditions. The main advantage of rules is their simplicity and human-interpretable form. Moreover, they are an aggregation model able to represent complex interactions between attributes. From the machine learning perspective, a rule can be treated as a simple classifier that gives a constant response to examples satisfying the condition part, and abstains from the response for all other examples.

The problem of learning a set of decision rules has been widely considered in machine learning. The most popular algorithms were based on a sequential covering procedure (Michalski 1983; Clark and Niblett 1989; Grzymala-Busse 1992; Cohen 1995; Fürnkranz 1996), also known as separate-and-conquer approach. According to Fürnkranz (1996), this procedure can be described as follows: "learn a rule that covers a part of the given training examples, remove the covered examples from the training set (the separate part) and recursively learn another rule that covers some of the remaining examples (the conquer part) until no examples remain".

In this paper,[1] we follow a more general approach that treats decision rules as base classifiers in the ensemble. The ensemble is constructed using boosting (Freund and Schapire 1997), also called forward stagewise additive modeling (Hastie et al. 2003). A boosting-like approach can be seen as a generalization of sequential covering, because it approximates the solution of the prediction task by sequentially adding new rules to the ensemble without adjusting those that have already been added. Each rule is fitted by focusing on examples which were the hardest to classify correctly by the rules already present in the ensemble. This is accomplished in terms of minimization of the empirical loss, i.e., the loss over the training set.

The sequential covering approach to rule induction encounters several problems. For example, there is no clear picture how to classify new examples by a set of decision rules. Usually, the rules overlap and do not cover the entire attribute space. In many rule induction algorithms, the rule generation phase and the classification phase are considered separately. First, the algorithm generates a set of rules, and then, one of

[1] This is an extended version of the paper "A general framework for learning an ensemble of decision rules" presented by the authors at the ECML/PKDD 2008 Workshop "From Local Patterns to Global Models". It also generalizes some previous results published in Błaszczyński et al. (2006) and Dembczyński et al. (2008b).

several possible strategies is used to classify new examples. Usually, different voting schemes are considered. The weights of the rules used in the voting are simple statistics computed over training examples covered by the rules. A popular choice is a relative number of covered examples or an estimate of conditional class probability within the rule. The latter suggests that the rules are treated as independent ones, which may not be true taking into account how the sequential covering procedure works.

The above problem can easily be solved in the boosting approach to rule induction. The general form of the classifier is given as a linear combination of rules, and the algorithm finds iteratively the best rules minimizing the empirical risk. Thus, the weights of rules are computed in a natural way, taking into account the interdependencies among the rules. As written by Cohen and Singer (1999), boosted rule ensembles are in fact simpler and better-understood formally than other state-of-the-art rule learners.

Let us also remark that classifiers in the form of a linear combination of decision rules have an additional interesting feature. In such areas as medicine or economics, one often uses decision procedures based on expert scoring that is in many cases just a linear combination of experts' rules. In other words, the expert scoring uses similar representation to that given by the boosting algorithm. This similarity in representation enables also a straightforward combination of expert knowledge with rules induced from data.

Induction of decision rules can be seen as an instance of the LeGo approach (Knobbe et al. 2008), a paradigm which relies on building global models by using local patterns. This is particularly the case of mining large databases, for which there exist many techniques for discovering local patterns. The aim is then to combine them into a global model. The typical rule learning algorithms (based on sequential covering or boosting) differ a bit from LeGo main stream approaches, since the rules being the local patterns are usually generated with the aim of minimizing some general criterion concerning the global model. However, optimization of a global criterion, broken down to sequential optimization of local loss functions, could be still considered as an example of the "local to global" modeling framework. Nonetheless, the rule learning algorithms can also be tailored to the case in which the previously generated patterns are combined together into a powerful ensemble. In this case, instead of building rules from data, the algorithm chooses from a set of pre-generated patterns the one that best fits the data in a given iteration.

1.1 Main contribution

We introduce and characterize an algorithm for learning an ensemble of decision rules, referred to as ENDER, that can be used in regression and binary classification tasks. ENDER consistently minimizes the empirical risk in all stages of the learning procedure: setting the best conditions, stopping the rule's growth and determining the response (decision) of the rule.

In particular, impurity measures that control construction of single decision rules are derived by using various minimization techniques often encountered in boosting. The minimal value of the impurity measure is a natural stopping criterion for building a single rule, as it represents a trade-off between misclassification and coverage. No

additional parameters are needed, contrary to the case of decision trees for which one has to set, for example, the maximal number of nodes, or the minimal number of training examples in terminal nodes.

Our main theoretical results concern the above-mentioned trade-off between misclassification (discrimination) and coverage (completeness). We investigate in depth four techniques of empirical loss minimization with respect to this trade-off. The first minimization technique, called in this paper simultaneous minimization, can be applied with the squared-error or the exponential loss function (Friedman 2001; Schapire and Singer 1999). Two others, the gradient descent (Mason et al. 1999) and the gradient boosting (Friedman 2001), can be used with any differentiable loss function. The last one, called constant-step minimization, can be used with any loss function and is particularly well-suited for decision rules. It relies on building a rule for a fixed constant value of the rule response, equal to the step length in the optimization process. We show that the coverage of the rule can be controlled by the step length. When the length approaches zero, this technique is equivalent to the gradient descent technique. Moreover, it follows that the gradient descent technique produces the most general rules (i.e., rules with the highest coverage) in comparison to other techniques. All these results are also confirmed experimentally.

We also verify the performance of ENDER with three different loss functions often used in binary classification: the exponential, the logit and the sigmoid loss. It appears that choice of the loss function has only a little impact on the performance. However, the use of a proper regularization can significantly increase the accuracy of the rule ensemble. We consider two forms of regularization: shrinkage and sampling. This result is rather common for boosting algorithms, however, it is interesting to observe what parameters of shrinkage and sampling are the best in the case of rules. Moreover, we test whether computation of a rule response on *all* training examples increases the performance, independently of the fact whether a condition part of the rule was built using a subsample or not. Such an approach usually decreases the value of response, so it plays also the role of regularization, and avoids overfitting the rule to the training set.

The three elements: shrinking, sampling, and calculating response of the rule on the whole training set, can be treated as an alternative to post-pruning, very often used in induction of decision rules.

1.2 Related work

The main competitors of the ENDER algorithm are SLIPPER (Cohen and Singer 1999), LRI (Weiss and Indurkhya 2000), RuleFit (Friedman and Popescu 2008), and MLRules (Dembczyński et al. 2008a). The interesting fact is that all these algorithms fall into the general framework adopted in this paper. This framework permits a thorough analysis of different rule ensemble algorithms which, to our knowledge, has not yet been done. As we will see, the main differences between above-mentioned algorithms lay in the chosen loss function and minimization technique.

Theoretical analysis of the trade-off between discrimination and completeness of a decision rule has been already performed in Janssen and Fürnkranz (2008), however,

while the referred paper discusses some classic rule impurity measures, our analysis concerns impurity measures obtained within boosting.

There are several theoretical studies concerning rule induction from the point of view of statistical learning theory. For instance, the generalization bounds for the sequential covering approach can be given using the analysis of the set covering machine (Marchand and Shawe-Taylor 2002). Rückert and Kramer (2008) gave generalization bounds based on margin minus variance objective, which is minimized in their procedure constructing a rule ensemble. On the other hand, bounds for the ensemble approach can be given by a slight modification of the margin theorem (Schapire et al. 1998; Koltchinskii and Panchenko 2006) used to explain the generalization ability of convex combinations of classifiers. Unfortunately, all those bounds cannot be used directly in practice, as they tend to be very loose (and thus overpessimistic) for real-life data. Therefore, experimental analysis is needed to test the generalization power of rule learning.

Let us also remark that some other approaches to rule induction exist. For instance, the a priori-based algorithms (i.e., algorithms that resemble the way in which association rules are generated) are also used for induction of predictive rules (Jovanoski and Lavrac 2001; Stefanowski and Vanderpooten 2001). There are also several rule-based approaches of lazy learning type (Bazan 1998; Góra and Wojna 2002b), possibly combined with instance-based methods (Domingos 1996; Góra and Wojna 2002a). 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 (Boros et al. 2000), where rules are called patterns. Let us also notice that decision rule models are strongly associated with rough set approaches to knowledge discovery (Pawlak 1991; Słowiński 1992; Grzymala-Busse 1992; Stefanowski 1998; Greco et al. 2000, 2001), where also Boolean reasoning has been applied (Skowron 1995).

1.3 Content

The paper is organized as follows. In Section 2, we formulate regression and binary classification problems. Section 3 outlines the ENDER algorithm. Different loss functions are considered in Section 4. Derivation of rule impurity measures is presented in Section 5. Theoretical analysis concerning the trade-off between misclassification (discrimination) and coverage (completeness) of a rule is performed in Section 6. Construction of a single rule in ENDER is described in Section 8. Section 9 discusses some connections with other rule induction algorithms, as well as with decision tree induction. Section 10 contains results of a large computational experiment on artificial data, and compares ENDER to other rule-based methods. Section 11 concludes the paper.

2 Problem statement

In the *prediction problem*, the aim is to predict the unknown value of a *decision attribute* y of an object, using a vector of known values of other attributes $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. In the following, we consider two types of the prediction problem, *regression* and *binary classification*. In the former, the decision attribute is quantitative and it is

assumed that $y \in \mathbb{R}$, where $\mathbb{R}$ is a set of real numbers. In the latter, the decision attribute is qualitative and values are taken from a finite set, $y \in \{-1, 1\}$. One class is then called "negative" (for which $y = -1$), and the second is called "positive" ($y = 1$).

The task is to find a function $f(\mathbf{x})$, called *classifier*, that predicts accurately the value of y. The accuracy of a single prediction $\hat{y}$ is measured by *loss function* $L(y, \hat{y})$ which determines the penalty for predicting $\hat{y}$ when the true value is y. The overall accuracy of function $f(\mathbf{x})$ is measured by the expected loss, called *risk*, over joint distribution $P(y, \mathbf{x})$:

$$R(f) = \mathbb{E}_{y\mathbf{x}} L(y, f(\mathbf{x})).$$

Therefore, the optimal risk-minimizing classification function, called the *Bayes optimal decision*, is given by:

$$f^* = \arg\min_f \mathbb{E}_{y\mathbf{x}} L(y, f(\mathbf{x})).$$

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

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

where f is chosen from a restricted family of functions.

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

$$L_{\text{se}}(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \tag{1}$$

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

$$f^*(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \mathbb{E}_{y|\mathbf{x}} L_{\text{se}}(y, f(\mathbf{x})) = \mathbb{E}_{y|\mathbf{x}}(y). \tag{2}$$

It follows that minimization of the squared-error loss on the data set can be seen as estimation of the expected value of y for a given $\mathbf{x}$.

The most natural loss function in binary classification is the *0–1 loss*. If we assume that the classifier gives continuous responses, $f(\mathbf{x}) \in \mathbb{R}$, the loss function can be expressed by $L(yf(\mathbf{x}))$, where $yf(\mathbf{x})$ is called *margin*. The positive margin means correct classification, and, intuitively, its magnitude tells what is the credibility of assigning an object to a given class. The *margin 0-1 loss* function is therefore:

$$L_{0-1}(yf(\mathbf{x})) = [\![yf(\mathbf{x}) < 0]\!],$$

where $[\![\pi]\!]$ is the Boolean test equal to 1 if predicate π is true, and 0 otherwise. The expected value of this loss function is simply a misclassification error of $f(\mathbf{x})$ defined by $\Pr(yf(\mathbf{x}) < 0)$. Thus, the Bayes optimal decision has the following form:

$$f^*(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \mathbb{E}_{y|\mathbf{x}} L_{0-1}(yf(\mathbf{x})) = \arg\max_{k\in\{-1,1\}} \Pr(y = k|\mathbf{x}). \tag{3}$$

By minimizing the 0-1 loss function, we estimate two regions in the attribute space: the positive one, for which the conditional probabilities of $y = 1$ for given $\mathbf{x}$ are greater than $1/2$, and the negative one, for which in turn the conditional probabilities of $y = -1$ for given $\mathbf{x}$ are greater than $1/2$.

3 Ensemble of decision rules: the ENDER framework

A single decision rule, denoted by $r(\mathbf{x})$, can be formally defined as the following function:

$$r(\mathbf{x}) = \alpha\Phi(\mathbf{x}),$$

where $\Phi(\mathbf{x})$ corresponds to condition part and α to decision part of the rule.

The condition part $\Phi(\mathbf{x})$ is defined as a conjunction of elementary conditions concerning particular attributes. An elementary condition for the attribute $j \in \{1, \ldots, n\}$ has the general form:

$$x_j \in S_j,$$

where x_j is the value of an object $\mathbf{x}$ on the attribute j and S_j is a subset of a domain of this attribute. In particular, elementary conditions are of the form $x_j \geq s_j$, $x_j \leq s_j$, for quantitative attributes, and $x_j = s_j$, $x_j \neq s_j$, for qualitative attributes, where s_j is taken from a domain of the attribute j. Let Φ be the set of elementary conditions, then $\Phi(\mathbf{x})$ indicates whether an object $\mathbf{x}$ satisfies the conjunction of elementary conditions Φ. In other words, $\Phi(\mathbf{x})$ defines an axis-parallel region in the attribute space. We say that a rule *covers* object $\mathbf{x}$ if it belongs to this region, i.e. $\Phi(\mathbf{x}) = 1$; otherwise, $\Phi(\mathbf{x}) = 0$. The number of training examples covered by the rule is referred to as *rule coverage*.

The decision (or response) α, is a real value assigned to the region defined by $\Phi(\mathbf{x})$. Depending on the way in which the rules are combined together, the interpretation of α is different. In the following, we consider a linear combination of rules. In such a case, the decision α defines the change of the prediction function if the condition $\Phi(\mathbf{x})$ is met. For regression, this is just the change of the predicted real value. For binary classification, $\text{sgn}(\alpha)$ can be interpreted as the class for which the rule "votes", and $|\alpha|$ as the weight of the rule.

The prediction function being the linear combination of M decision rules is given by:

$$f_M(\mathbf{x}) = \alpha_0 + \sum_{m=1}^{M} r_m(\mathbf{x}), \tag{4}$$

where α_0 is a constant value, which can be interpreted as a default rule, covering the entire attribute space. In the case of regression, prediction is made directly by value $f_M(\mathbf{x})$, and in the case of binary classification, by $\text{sgn}(f_M(\mathbf{x}))$.

The construction of an optimal combination of rules minimizing the empirical risk is a hard optimization problem. Therefore, we follow forward stagewise additive modeling. This results in an iterative procedure in which rules are added one by one. We start with the default rule defined as:

$$\alpha_0 = \arg\min_{\alpha} R_{emp}(\alpha) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha). \tag{5}$$

In each subsequent iteration, a new rule is constructed taking into account previously generated rules. Let $f_{m-1}(\mathbf{x})$ be a prediction function after $m-1$ iterations involving the first $m-1$ rules and the default rule. In the m-th iteration, a decision rule can be obtained by solving:

$$r_m = \arg\min_{r} R_{emp}(f_{m-1} + r) = \arg\min_{\Phi,\alpha} \sum_{i=1}^{N} L(y_i, f_{m-1}(\mathbf{x}_i) + \alpha\Phi(\mathbf{x}_i)).$$

Since $\Phi(\mathbf{x})$ takes two values only, 0 or 1, it is convenient to rewrite the above formulation to:

$$r_m = \arg\min_{\Phi,\alpha} \left(\sum_{\Phi(\mathbf{x}_i)=1} L(y_i, f_{m-1}(\mathbf{x}_i) + \alpha) + \sum_{\Phi(\mathbf{x}_i)=0} L(y_i, f_{m-1}(\mathbf{x}_i)) \right). \tag{6}$$

Unfortunately, the exact solution of Eq. 6 is still computationally hard. That is why we proceed in two steps.

1. Find Φ_m by minimizing an impurity measure $\mathcal{L}_m(\Phi)$ derived from Eq. 6 in such a way that it does not depend on α:

$$\Phi_m = \arg\min_{\Phi} \mathcal{L}_m(\Phi). \tag{7}$$

2. Find α_m by solving the following line-search problem:

$$\alpha_m = \arg\min_{\alpha} \sum_{\Phi_m(\mathbf{x}_i)=1} L(y_i, f_{m-1}(\mathbf{x}_i) + \alpha), \tag{8}$$

   where Φ_m is a solution of the first step.

These two steps depend on a chosen loss function and minimization technique used for deriving the impurity measure. In the next two sections, we will consider several loss functions and minimization techniques commonly used within the boosting framework for regression and binary classification tasks.

The general framework for learning the rule ensemble, called ENDER, is given as Algorithm 1.

Algorithm 1: Ensemble of decision rules – the ENDER framework

input : set of training examples $\{y_i, \mathbf{x}_i\}_1^N$,
$L(y, f(\mathbf{x}))$ – loss function,
M – number of decision rules to be generated.
output: ensemble of decision rules $f_M(\mathbf{x})$.

$\alpha_0 = \arg\min_\alpha \sum_{i=1}^N L(y_i, \alpha)$
$f_0(\mathbf{x}) = \alpha_0$;
for $m = 1$ *to* M **do**
  $\Phi_m = \arg\min_\Phi \mathcal{L}_m(\Phi)$
  $\alpha_m = \arg\min_\alpha \sum_{\Phi_m(\mathbf{x}_i)=1} L(y_i, f_{m-1}(\mathbf{x}_i) + \alpha)$
  $r_m(\mathbf{x}) = \alpha_m \Phi_m(\mathbf{x})$
  $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + r_m(\mathbf{x})$
end

4 Loss functions and decision of the rule

In this section, we review different loss functions that can be used within the framework of ENDER. We also present the calculation of the decision α_m of rule r_m for given Φ_m, for all considered loss functions.

In the case of regression, we consider the squared-error loss (Eq. 1). The decision α_m of rule r_m follows from the problem (Eq. 8) that has in this case the closed-form solution:

$$\begin{aligned} \alpha_m &= \arg\min_\alpha \sum_{\Phi_m(\mathbf{x}_i)=1} (y_i - f_{m-1}(\mathbf{x}_i) - \alpha)^2 \\ &= \frac{\sum_{\Phi_m(\mathbf{x}_i)=1} (y_i - f_{m-1}(\mathbf{x}_i))}{\sum_{i=1}^N \Phi_m(\mathbf{x}_i)}, \end{aligned} \tag{9}$$

which is the average over the residuals $y_i - f_{m-1}(\mathbf{x}_i)$ of examples covered by the rule.

For binary classification, the direct goal is to minimize the 0-1 loss. There is, however, a problem with minimization of this loss function, since it is neither convex, nor differentiable. Moreover, it is insensitive to the value of the margin $yf(\mathbf{x})$. Therefore, instead of this function, convex surrogates (upper-bounding the 0-1 loss) are commonly used, such as the exponential and the logit loss, which makes the minimization process more tractable. The *exponential loss* is defined as:

$$L_{\exp}(yf(\mathbf{x})) = \exp(-yf(\mathbf{x})). \tag{10}$$

This loss function is used in AdaBoost (Freund and Schapire 1997). The *logit loss*

$$L_{\log}(yf(\mathbf{x})) = \log(1 + \exp(-2yf(\mathbf{x}))) \tag{11}$$

is commonly used in statistics. These two loss functions have the same Bayes optimal decision (Hastie et al. 2003):

$$f^*(\mathbf{x}) = \frac{1}{2}\log\frac{\Pr(y=1|\mathbf{x})}{\Pr(y=-1|\mathbf{x})}, \tag{12}$$

which is the logit transform of conditional probabilities. The expression (Eq. 12) can be inverted to give:

$$\Pr(y=1|\mathbf{x}) = \frac{1}{1+\exp(-2f^*(\mathbf{x}))}. \tag{13}$$

Therefore, minimization of these loss functions on the training set can be seen as estimation of conditional probabilities $\Pr(y=1|\mathbf{x})$. The sign of $f(\mathbf{x})$ estimates in turn the class with a higher probability.

The additional advantage of the exponential loss is that there exists a closed-form solution to Eq. 8:

$$\begin{aligned}\alpha_m &= \arg\min_{\alpha} \sum_{\Phi_m(\mathbf{x})=1} \exp(-y_i(f_{m-1}(\mathbf{x}_i)+\alpha)) \\ &= \frac{1}{2}\log\frac{\sum_{\Phi_m(\mathbf{x}_i)=1, y_i=1}\exp(-f_{m-1}(\mathbf{x}_i))}{\sum_{\Phi_m(\mathbf{x}_i)=1, y_i=-1}\exp(f_{m-1}(\mathbf{x}_i))}.\end{aligned} \tag{14}$$

The logit loss is better motivated statistically (since it is a proper log-likelihood), but there is no analytical solution to Eq. 8. To speed up the computations, instead of numerically solving the line search problem, a single Newton–Raphson step is performed, similarly as in (Friedman 2001):

$$\alpha_m = -\left.\frac{\sum_{\Phi_m(\mathbf{x}_i)=1}\frac{\partial}{\partial\alpha}L_{\log}(y_i(f_{m-1}(\mathbf{x}_i)+\alpha))}{\sum_{\Phi_m(\mathbf{x}_i)=1}\frac{\partial^2}{\partial\alpha^2}L_{\log}(y_i(f_{m-1}(\mathbf{x}_i)+\alpha))}\right|_{\alpha=0}. \tag{15}$$

One can also consider to use the *sigmoid loss* that is a continuous approximation of the 0-1 loss:

$$L_{\text{sigm}}(yf(\mathbf{x})) = \frac{1}{1+\exp(yf(\mathbf{x}))}. \tag{16}$$

Although not convex, it is differentiable. The Bayes optimal decision for Eq. 16 is rather aberrant:

$$f^*(\mathbf{x}) = \begin{cases} +\infty, & \text{if } \Pr(y=1|\mathbf{x}) > \frac{1}{2}, \\ -\infty, & \text{if } \Pr(y=-1|\mathbf{x}) < \frac{1}{2}, \\ \text{arbitrary}, & \text{otherwise.} \end{cases}$$

There are, however, theoretical justifications for using this loss function. It is shown in Mason et al. (1999) that the upper bound of the misclassification error for such a

loss function is tighter than the bound obtained by Schapire et al. (1998). Moreover, contrary to the exponential and logit loss functions, this loss function is bounded within the range (0,1), and is therefore less sensitive to outliers.

Unfortunately, no analytical solution to Eq. 8 exists for the sigmoid loss. Moreover, due to non-convexity of this function one should avoid the Newton–Raphson method. Nevertheless, to speed up the computations, we do not solve the line search problem, but instead we perform one step of a small constant length γ in the direction of the negative gradient.

5 Impurity measures

In this section, we derive from Eq. 6 several impurity measures $\mathcal{L}_m(\Phi)$. There have been different minimization techniques considered within boosting. We review some of them in the following subsections in the context of deriving impurity measures for decision rules. Moreover, a new technique, the constant-step minimization is introduced.

Let us remind the form of the optimization problem (Eq. 6):

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

By using a certain loss function in Eq. 6, and applying one of the minimization techniques in order to make the problem independent of α, we obtain an impurity measure $\mathcal{L}_m(\Phi)$.

5.1 Simultaneous minimization

In the case of the squared-error and the exponential loss we have a closed-form solution for α_m. That is why one can perform in this case simultaneous minimization of both parameters, Φ_m and α_m.

Let us start with the squared-error loss. By putting Eq. 9 into Eq. 6, removing constant terms, and taking the square root, we obtain the following expression to be minimized:

$$-\frac{\left|\sum_{\Phi(\mathbf{x}_i)=1}(y_i - f_{m-1}(\mathbf{x}_i))\right|}{\sqrt{\sum_{i=1}^{N}\Phi(\mathbf{x}_i)}}, \tag{17}$$

which depends only on Φ and plays the role of $\mathcal{L}_m(\Phi)$.

In the case of the exponential loss, one puts the optimal value of α_m given by Eq. 14 into Eq. 6, and obtains the impurity measure $\mathcal{L}_m(\Phi)$:

$$\mathcal{L}_m(\Phi) = 2 \times \sqrt{\sum_{\Phi(\mathbf{x}_i)=1, y_i=1} w_i^{(m)} \sum_{\Phi(\mathbf{x}_i)=1, y_i=-1} w_i^{(m)}} + \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)}, \tag{18}$$

where $w_i^{(m)} = e^{-y_i f_{m-1}(\mathbf{x}_i)}$ can be treated as the weight of the i-th training example in the m-th iteration.

Let us remark that the boosting approach based on simultaneous minimization of the squared-error loss is well-known in statistics as forward stagewise additive modeling (Hastie et al. 2003). It is also used in Gradient Boosting Machine (Friedman 2001). Simultaneous minimization of the exponential loss is implicitly used in AdaBoost with confidence rated predictions (Schapire and Singer 1999).

5.2 Gradient descent

Contrary to the simultaneous minimization, the gradient descent technique can be used with any differentiable loss function. It approximates Eq. 6 up to the first order with respect to α:

$$r_m \simeq \arg\min_{\Phi,\alpha} \left(\sum_{\Phi(\mathbf{x}_i)=1} \left(L(y_i, f_{m-1}(\mathbf{x}_i)) - \alpha \tilde{w}_i^{(m)} \right) + \sum_{\Phi(\mathbf{x}_i)=0} L(y_i, f_{m-1}(\mathbf{x}_i)) \right),$$

where

$$\tilde{w}_i^{(m)} = -\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\bigg|_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i).}$$

One can observe that the optimal solution with respect to Φ is obtained by minimizing:

$$\mathcal{L}_m(\Phi) = -\sum_{\Phi(\mathbf{x}_i)=1} \alpha \tilde{w}_i^{(m)}, \tag{19}$$

since the sum over all $L(y_i, f_{m-1}(\mathbf{x}_i))$ is constant in a given iteration, and thus does not change the solution. Observe that for a given value of α, the solution depends only on $\sum_{\Phi(\mathbf{x}_i)=1} \tilde{w}_i^{(m)}$, so the minimization of Eq. 19 can be finally reformulated to the minimization of the following term:

$$\mathcal{L}_m(\Phi) = -\left| \sum_{\Phi(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right|, \tag{20}$$

because the sign and the magnitude of α may be established afterwards.

The gradient descent technique applied to the exponential loss is in fact used in the original AdaBoost algorithm. Mason et al. (1999) have widely considered this technique for different loss functions.

5.3 Gradient boosting

Gradient boosting can be treated as an adaptation of simultaneous minimization of the squared-error loss to other loss functions. It is defined as minimization of the squared-error between the base classifier response and the negative gradient of any differentiable loss function. In the case of decision rules, this can be expressed as:

$$r_m \simeq \arg\min_{\Phi,\alpha} \left(\sum_{\Phi(\mathbf{x}_i)=1} \left(\tilde{w}_i^{(m)} - \alpha \right)^2 + \sum_{\Phi(\mathbf{x}_i)=0} \left(\tilde{w}_i^{(m)} \right)^2 \right). \tag{21}$$

The minimization problem defined by Eq. 21 can be solved for:

$$\alpha = \frac{\sum_{\Phi(\mathbf{x}_i)=1} \tilde{w}_i^{(m)}}{\sum_{i=1}^{N} \Phi(\mathbf{x}_i)}, \tag{22}$$

and by putting Eq. 22 into Eq. 21, and performing some simple calculations, we obtain:

$$\mathcal{L}_m(\Phi) = -\frac{\left| \sum_{\Phi(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right|}{\sqrt{\sum_{i=1}^{N} \Phi(\mathbf{x}_i)}}. \tag{23}$$

This technique has been used by Friedman (2001) in Gradient Boosting Machine for a wide spectrum of loss functions.

5.4 Constant-step minimization

Finally, we consider a novel technique that consists in minimization of the loss function with a constant step. In other words, we restrict α in Eq. 6 to $\alpha \in \{-\beta, \beta\}$, where β is a fixed parameter of the algorithm. Then, Eq. 6 becomes:

$$r_m(\mathbf{x}) \simeq \arg\min_{\Phi,\pm\beta} \left(\sum_{\Phi(\mathbf{x}_i)=1} L(y_i (f_{m-1}(\mathbf{x}_i) \pm \beta)) + \sum_{\Phi(\mathbf{x}_i)=0} L(y_i f_{m-1}(\mathbf{x}_i)) \right). \tag{24}$$

The above formula can be used with any loss function, since it involves calculation of two loss values at points $f_{m-1}(\mathbf{x}_i) \pm \beta$. This technique is natural for sigmoid loss (Eq. 16), for which we approximate the rule response (Eq. 8) by a constant step γ in the direction of the negative gradient.

Constant-step minimization is the simplest possible optimization procedure, however, it seems to be well-tailored for decision rules, as stated by results given in the next section.

6 Rule coverage

In this section, we investigate connections between impurity measures obtained by minimization techniques described above. This theoretical analysis focuses on the trade-off between misclassification (discrimination) and coverage (completeness) of the rule.

First, we consider the connection between the gradient descent and gradient boosting techniques. One can easily notice the similarities between impurity measures Eq. 20 and Eq. 23. On this basis, we can prove the following theorem that states that the gradient descent produces more general rules (covering more training examples) than the gradient boosting.

Theorem 1 *Consider minimization of any differentiable loss function on the training set. Let* Φ_m^{GD} *be the optimal condition part obtained by minimization of Eq. 20, i.e.:*

$$\Phi_m^{GD} = \arg\min_{\Phi} - \left| \sum_{\Phi_m(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right|, \tag{25}$$

and let Φ_m^{GB} *be the optimal condition part obtained by minimization of Eq. 23, i.e.:*

$$\Phi_m^{GB} = \arg\min_{\Phi} - \frac{\left| \sum_{\Phi_m(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right|}{\sqrt{\sum_{i=1}^{N} \Phi_m(\mathbf{x}_i)}}, \tag{26}$$

where

$$\tilde{w}_i^{(m)} = -\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\bigg|_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i).}$$

Then, the following holds:

$$\sum_{i=1}^{N} \Phi_m^{GD}(\mathbf{x}_i) \geq \sum_{i=1}^{N} \Phi_m^{GB}(\mathbf{x}_i).$$

Proof The proof is straightforward. From Eq. 25 and Eq. 26 we have that:

$$-\left| \sum_{\Phi_m^{GD}(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right| \leq -\left| \sum_{\Phi_m^{GB}(\mathbf{x}_i)=1} \tilde{w}_i^{(m)} \right|,$$

and

$$-\frac{\left|\sum_{\Phi_m^{GD}(\mathbf{x}_i)=1} \tilde{w}_i^{(m)}\right|}{\sqrt{\sum_{i=1}^N \Phi_m^{GD}(\mathbf{x}_i)}} \geq -\frac{\left|\sum_{\Phi_m^{GB}(\mathbf{x}_i)=1} \tilde{w}_i^{(m)}\right|}{\sqrt{\sum_{i=1}^N \Phi_m^{GB}(\mathbf{x}_i)}}.$$

From the above, we immediately get:

$$\sum_{i=1}^N \Phi_m^{GD}(\mathbf{x}_i) \geq \sum_{i=1}^N \Phi_m^{GB}(\mathbf{x}_i),$$

as claimed. □

The above result concerns both regression and binary classification problems. Some further theoretical results can be obtained by limiting the considerations to binary classification. In the remainder of this section, we give some of these results. Let us remark, however, that the rule coverage is expressed in the following as a sum of weights of the covered examples, and not as a number of them.

In the case of binary classification, we use the margin loss functions $L(yf(\mathbf{x}))$, so the weights are given by:

$$w_i^{(m)} = -\frac{\partial L(y_i f(\mathbf{x}_i))}{\partial (y_i f(\mathbf{x}_i))}\bigg|_{y_i f(\mathbf{x}_i)=y_i f_{m-1}(\mathbf{x}_i)}.$$

One can easily observe that

$$\tilde{w}_i^{(m)} = y_i w_i^{(m)}.$$

Let us also note that for the exponential loss, we have

$$w_i^{(m)} = e^{-y_i f_{m-1}(\mathbf{x}_i)}.$$

That is why the same symbol $w_i^{(m)}$ is used here and in Eq. 18.

Let us introduce a set

$$R_+ = \{i : \alpha y_i \Phi(\mathbf{x}_i) > 0\}$$

that contains examples "correctly classified" by the rule (for $y_i = 1$, α should be > 0, and for $y_i = -1$, α should be < 0). Analogously, we introduce a set

$$R_- = \{i : \alpha y_i \Phi(\mathbf{x}_i) < 0\}$$

that contains examples "misclassified" by the rule.

Let us investigate the connection between the simultaneous minimization and the gradient descent technique applied to the exponential loss function. Using the sets

defined above, we can transform the functional minimized in the gradient descent technique (Eq. 20) to the following form:

$$\mathcal{L}_m(\Phi) = -\sum_{i \in R_+} w_i^{(m)} + \sum_{i \in R_-} w_i^{(m)}. \tag{27}$$

We can also transform the formula applied in the simultaneous minimization (Eq. 18) to the following from:

$$\mathcal{L}_m(\Phi) = -\sqrt{\sum_{i \in R_+} w_i^{(m)}} + \sqrt{\sum_{i \in R_-} w_i^{(m)}}. \tag{28}$$

This is accomplished by using the fact that

$$\sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)} = \sum_{i=1}^{N} w_i^{(m)} - \sum_{\Phi(\mathbf{x}_i)=1, y_i=1} w_i^{(m)} - \sum_{\Phi(\mathbf{x}_i)=1, y_i=-1} w_i^{(m)},$$

and by applying short multiplication formulas, removing the constant term $\sum_{i=1}^{N} w_i^{(m)}$, and imputing the summations over R_+ and R_- instead of $\Phi(\mathbf{x}_i) = 1, y = 1$ and $\Phi(\mathbf{x}_i) = 1, y = -1$.

Now, we can state the following theorem.

Theorem 2 *Consider minimization of the exponential loss on the training set. Let* Φ_m^{GD} *be the optimal condition part obtained by minimization of Eq. 27, i.e.:*

$$\Phi_m^{GD} = \arg\min_{\Phi} \left(-\sum_{i \in R_+} w_i^{(m)} + \sum_{i \in R_-} w_i^{(m)} \right), \tag{29}$$

and let Φ_m^{SM} *be the optimal condition part obtained by minimization of Eq. 28, i.e.:*

$$\Phi_m^{SM} = \arg\min_{\Phi} \left(-\sqrt{\sum_{i \in R_+} w_i^{(m)}} + \sqrt{\sum_{i \in R_-} w_i^{(m)}} \right), \tag{30}$$

where

$$w_i^{(m)} = e^{-y_i f_{m-1}(\mathbf{x}_i)}.$$

Then, the following holds:

$$\sum_{\Phi_m^{SM}(\mathbf{x}_i)=1} w_i^{(m)} \leq \sum_{\Phi_m^{GD}(\mathbf{x}_i)=1} w_i^{(m)}.$$

Proof Let us use the following notation:

$$W_+^{GD} = \sum_{i \in R_+^{GD}} w_i^{(m)}, \quad W_-^{GD} = \sum_{i \in R_-^{GD}} w_i^{(m)},$$
$$W_+^{SM} = \sum_{i \in R_+^{SM}} w_i^{(m)}, \quad W_-^{SM} = \sum_{i \in R_-^{SM}} w_i^{(m)},$$

where R_+^{GD} and R_-^{GD} denote sets of correctly and incorrectly classified examples for Φ_m^{GD}, respectively. Analogously, R_+^{SM} and R_-^{SM} denote sets of correctly and incorrectly classified examples for Φ_m^{SM}, respectively. From Eq. 29, we have:

$$-W_+^{GD} + W_-^{GD} \leq -W_+^{SM} + W_-^{SM}, \tag{31}$$

and from Eq. 30, we have:

$$-\sqrt{W_+^{GD}} + \sqrt{W_-^{GD}} \geq -\sqrt{W_+^{SM}} + \sqrt{W_-^{SM}}. \tag{32}$$

From Eq. 32 we obtain:

$$\sqrt{W_-^{GD}} \geq \sqrt{W_+^{GD}} - \sqrt{W_+^{SM}} + \sqrt{W_-^{SM}}.$$

Further, from Eq. 31 we obtain:

$$\begin{aligned} -W_+^{GD} &\leq -W_+^{SM} + W_-^{SM} - W_-^{GD} \\ &\leq -W_+^{SM} + W_-^{SM} - \left(\sqrt{W_+^{GD}} - \sqrt{W_+^{SM}} + \sqrt{W_-^{SM}}\right)^2 \\ &= -2W_+^{SM} - W_+^{GD} + 2\sqrt{W_+^{GD}}\left(\sqrt{W_+^{SM}} - \sqrt{W_-^{SM}}\right) + 2\sqrt{W_+^{SM}}\sqrt{W_-^{SM}}. \end{aligned}$$

From the above we get:

$$2\sqrt{W_+^{GD}}\left(\sqrt{W_+^{SM}} - \sqrt{W_-^{SM}}\right) \geq 2\sqrt{W_+^{SM}}\left(\sqrt{W_+^{SM}} - \sqrt{W_-^{SM}}\right).$$

Thus, $\sqrt{W_+^{GD}} \geq \sqrt{W_+^{SM}}$, because the term in parentheses is always positive (since it is equal to minus optimum of the objective function (Eq. 30), which is always greater then the value of the objective function of "empty" rule, equal to zero). From this, and Eq. 32, we have immediately that $\sqrt{W_-^{GD}} \geq \sqrt{W_-^{SM}}$. Thus, we get:

$$W_+^{SM} + W_-^{SM} \leq W_+^{GD} + W_-^{GD},$$

and finally

$$\sum_{\Phi_m^{SM}(\mathbf{x}_i)=1} w_i^{(m)} \leq \sum_{\Phi_m^{GD}(\mathbf{x}_i)=1} w_i^{(m)},$$

as claimed. □

This theorem states that the gradient descent technique generates more general rules than the simultaneous minimization.

In the case of the gradient descent technique and any differentiable loss function, we can precisely determine what is the trade-off between discrimination and completeness of the decision rule.

Theorem 3 *Minimization of Eq. 27 is equivalent to minimization of:*

$$\mathcal{L}_m(\Phi) = \sum_{i \in R_-} w_i^{(m)} + \frac{1}{2} \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)}, \tag{33}$$

where

$$w_i^{(m)} = -\frac{\partial L(y_i f(\mathbf{x}_i))}{\partial (y_i f(\mathbf{x}_i))}\bigg|_{y_i f(\mathbf{x}_i)=y_i f_{m-1}(\mathbf{x}_i).}$$

Proof Let us remark that

$$\sum_{i \in R_+} w_i^{(m)} = \sum_{i=1}^{N} w_i^{(m)} - \sum_{i \in R_-} w_i^{(m)} - \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)}. \tag{34}$$

Since $\sum_{i=1}^{N} w_i^{(m)}$ is constant in a given iteration, it can be added or subtracted from Eq. 27 without any influence on the optimization process. Thus, we finally obtain that a subject to minimize is:

$$2 \sum_{i \in R_-} w_i^{(m)} + \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)},$$

and we get Eq. 33 after dividing it by 2. □

This theorem has a nice interpretation: the first term of Eq. 33 corresponds to examples "misclassified" by the rule, while the second term—to examples which are not classified by the rule at all. Value $\frac{1}{2}$ plays the role of a penalty for abstaining from classification and defines a trade-off between discrimination and completeness of the decision rule.

In the case of the exponential loss, we can additionally prove that the constant-step minimization generalizes the gradient descent technique in such a way that a larger step length results in a smaller rule coverage. That is why we claim that the constant-step minimization is particularly well-tailored for decision rules.

Theorem 4 *The solution of Eq. 24 for the exponential loss and step length β is equivalent to minimization of:*

$$\mathcal{L}_m(\Phi) = \sum_{i \in R_-} w_i^{(m)} + \ell \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)}, \tag{35}$$

where

$$w_i^{(m)} = e^{-y_i f_{m-1}(\mathbf{x}_i)}, \quad \ell = \frac{1 - e^{-\beta}}{e^{\beta} - e^{-\beta}}, \quad \beta = \log \frac{1-\ell}{\ell}.$$

Proof The first part of the theorem follows from putting the exponential loss formula Eq. 10 into Eq. 24:

$$r_m = \arg\min_{\Phi, \pm\beta} \left(\sum_{i \in R_+} w_i^{(m)} e^{-\beta} + \sum_{i \in R_-} w_i^{(m)} e^{\beta} + \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)} \right). \tag{36}$$

Applying similar decomposition as in the case of Eq. 34, we can equivalently minimize:

$$\left(e^{\beta} - e^{-\beta}\right) \sum_{i \in R_-} w_i^{(m)} + (1 - e^{-\beta}) \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)} + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)}. \tag{37}$$

The last element does not change the solution (because β is constant), so it suffices to minimize the first two terms. Moreover, dividing Eq. 37 by $\left(e^{\beta} - e^{-\beta}\right)$ we obtain:

$$\mathcal{L}_m(\Phi) = \sum_{i \in R_-} w_i^{(m)} + \ell \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)},$$

where

$$\ell = \frac{1 - e^{-\beta}}{e^{\beta} - e^{-\beta}}, \qquad \beta = \log \frac{1-\ell}{\ell},$$

as claimed. □

Let us observe that for $\beta > 0, \ell \in [0, 0.5)$. Expression Eq. 35 has a similar interpretation to Eq. 33, but with a varying value of ℓ. Increasing ℓ (or decreasing β) results in more general rules, covering more examples. For $\beta \to 0$ we get the gradient descent technique applied to the exponential loss. This means that the gradient descent produces the most general rules (in the sense of coverage).

Finally, we consider a more general form of Theorem 4. This theorem also speaks about a trade-off between misclassification and coverage of the rules, but in the case of any twice differentiable margin loss function.

Theorem 5 *The solution of Eq. 24 for any twice differentiable loss function $L(yf(\mathbf{x}))$ and step length β is equivalent to the minimization of:*

$$\mathcal{L}_m(\Phi) = \sum_{i \in R_-} w_i^{(m)} + \frac{1}{2} \sum_{\Phi(\mathbf{x}_i)=0} \left(w_i^{(m)} - \beta v_i^{(m)} \right), \tag{38}$$

where

$$w_i^{(m)} = -\left. \frac{\partial L(y_i f(\mathbf{x}_i))}{\partial (y_i f(\mathbf{x}_i))} \right|_{y_i f(\mathbf{x}_i) = y_i f_{m-1}(\mathbf{x}_i)},$$
$$v_i^{(m)} = \frac{1}{2} \left. \frac{\partial^2 L(y_i f(\mathbf{x}_i))}{\partial (y_i f(\mathbf{x}_i))^2} \right|_{y_i f(\mathbf{x}_i) = y_i f_{m-1}(\mathbf{x}_i) \pm \lambda_i y_i},$$

for some $\lambda_i \in [0, \beta]$.

Proof It follows from Taylor's expansion of $L(y(f(\mathbf{x}) \pm \beta))$ with respect to $\pm\beta$ up to the second order. The formula for $\mathcal{L}_m(\Phi)$ is then:

$$\mathcal{L}_m(\Phi) = \sum_{i \in R_+} \left(L_i - \beta w_i^{(m)} + \beta^2 v_i^{(m)} \right) + \sum_{i \in R_-} \left(L_i + \beta w_i^{(m)} + \beta^2 v_i^{(m)} \right) + \sum_{\Phi(\mathbf{x}_i)=0} L_i,$$

where $L_i = L(y_i f_{m-1}(\mathbf{x}_i))$. After some simple transformations similar to the ones from the proof of Theorem 4, one proves the thesis. □

As above, β defines a trade-off between misclassified and unclassified examples. Values $w_i^{(m)}$ are always positive, since the loss function is decreasing. If the loss function is convex (e.g., the exponential or the logit loss), $v_i^{(m)}$ is also positive, therefore, increasing β decreases the penalty for abstaining from classification, which leads to smaller and more specific rules. Notice that for $\beta \to 0$ expression (Eq. 38) boils down to the gradient descent technique.

The situation changes if the loss function is not convex, which is the case of the sigmoid loss. This loss function is convex for $yf(\mathbf{x}) > 0$ and concave for $yf(\mathbf{x}) < 0$, therefore, as β increases, uncovered examples satisfying $y_i f_{m-1}(\mathbf{x}_i) > 0$ ("correctly classified" by previous rules) are penalized less, while the penalty for uncovered "misclassified" examples ($y_i f_{m-1}(\mathbf{x}_i) < 0$) increases. This leads to the following conclusion: although the rule covers only a part of the examples, with respect to uncovered examples it still tries to make a small error; remark that the weights of the uncovered examples depend on the curvature of the function (second derivative) rather than on the slope (first derivative).

7 Regularization

A decision rule has the form of an n-dimensional rectangle, where n is the number of attributes. It can be shown, that the class of n-dimensional rectangles has the VC dimension equal to $2n$ (Kearns and Vazirani 1994), and it does not depend on the

number of cuts. This is contrary to the tree classifier, for which the VC dimension grows to infinity with increasing number of cuts (nodes). Therefore, in the case of tree ensembles, one usually specifies some constraints on tree complexity, e.g., the maximal number of nodes, while in the case of a rule ensemble, such constraints are not necessary.

The theoretical results reported in Schapire et al. (1998) and Koltchinskii and Panchenko (2006) suggest that an ensemble with a simple base classifier (with low VC dimension) and high prediction confidence (margin) on the data set generalizes well, regardless of the size of the ensemble. Nevertheless, the computational experiments have shown that the performance of a rule ensemble can deteriorate as the number of rules grows, especially for the problems with a high level of noise. Similar phenomenon has been observed for other boosting algorithms, in particular for AdaBoost (Mason et al. 1999; Friedman et al. 2000; Dietterich 2000). Therefore, the algorithm should be used with some kind of regularization.

The form of regularization which is particularly useful in the case of rule ensembles is the L_1-penalty, also called *lasso* (Hastie et al. 2003). This leads to the problem of learning linear combination of all possible rules with additional term $\sum_m |\alpha_m|$ that penalizes absolute values of the combination coefficients α_m. Lasso penalty is indifferent to dispersion of coefficient values and tends to produce solutions with a large variation in the absolute values of the coefficients, with many of them set to zero. This is especially appropriate in the rule ensemble context because among all possible rules only a small number is likely to represent very good predictors.

To approximate a solution of such a regularized problem, one can follow a strategy that is called shrinkage (Hastie et al. 2003). It consists in shrinking a newly generated rule $r_m(\mathbf{x}) = \alpha_m \Phi_m(\mathbf{x})$ towards rules already present in the ensemble:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu \cdot r_m(\mathbf{x}),$$

where $\nu \in (0, 1]$ is a shrinkage parameter that can be regarded as the learning rate. For small ν, one can obtain a solution that is close to the regularized one. Note, however, that small ν requires large M, and the resulting rule ensemble is then less interpretable.

Such an approach gives even better results, when decision rules are less correlated. That is why the procedure for finding Φ_m works on a subsample of original data, that is a fraction ζ of all training examples, drawn without replacement (Friedman and Popescu 2003). Such an approach leads to a set of rules that are more diversified and less correlated. Moreover, finding Φ_m on a subsample reduces the computational complexity. However, we pay once again the price of the interpretability.

Independently on the fact whether Φ_m was found using a subsample or not, the value of α_m is calculated on *all* training examples in the introduced algorithm. This usually decreases $|\alpha_m|$, so it plays also the role of regularization, and avoids overfitting the rule to the training set.

These three elements: shrinking, sampling, and calculating α_m on the entire training set, constitute a competitive technique to post-pruning, often used in rule induction algorithms. Our experiments have shown that this technique improves significantly the accuracy of the classifier.

8 Single rule construction

Having defined a functional $\mathcal{L}_m(\Phi)$, a greedy procedure for finding Φ works in the way resembling generation of decision trees. In this case, however, the algorithm constructs only one path from the root to the leaf. At the beginning, Φ_m is empty and in each subsequent step an elementary condition $x_j \in S_j$ minimizing $\mathcal{L}_m(\Phi)$ is added to Φ_m. This procedure ends if $\mathcal{L}_m(\Phi)$ cannot be decreased anymore. Contrary to the generation of decision trees, a minimal value of $\mathcal{L}_m(\Phi)$ is a natural stop criterion. No other parameters are necessary for decision rules.

From computational perspective, this is the most critical part of the algorithm. In order to speed up computations, training examples are sorted once before generating any rule. In the worst case, the greedy procedure constructing the condition part of the rule may require $nN(N+1)/2$ steps. This occurs, when each added elementary condition makes Φ to cover one training example less, and the rule construction ends with one covered object only. Such a situation is rather uncommon. First, the addition of a new elementary condition cuts-off rather more than one training example, and the average size of the rules l (number of elementary conditions involved in Φ) is also much smaller than N. In fact, l differs depending on the complexity of the problem, since the rule is built until $\mathcal{L}_m(\Phi)$ cannot be decreased anymore. Concluding this, the construction of the condition part of the rule roughly scales with nNl, which is similar to the complexity of decision tree learning.

9 Discussion and related work

ENDER provides a novel view on rule induction. The main advantage of ENDER is that it generalizes to some extent different approaches to rule learning. In this section, we discuss main features of ENDER and relate it to other algorithms, pointing out similarities and differences.

9.1 Sequential covering

Initially, almost all algorithms for rule learning were based on sequential covering. The most popular are AQ (Michalski 1983), CN2 (Clark and Niblett 1989), Ripper (Cohen 1995), and LEM (Grzymala-Busse 1992). Sequential covering relies on learning a rule that covers a part of given training examples, removing the covered examples from the training set, and repeating this step until no examples remain. This procedure is repeated separately for each class. In each turn, the rules cover examples from one class only.

One can observe some similarities between this approach and the stagewise minimization of a loss function. In fact, ENDER can be seen as a generalization of sequential covering. Let us take a simple heuristic that covers examples from one class only, and let us use the margin 0-1 loss. For such a setting, the value of the loss function decreases down to 0 for all correctly covered training examples and there is no need for another rule to cover them again. This corresponds to removing such objects from the training set. In the case of ENDER, all the rules will obtain the same absolute

value of rule response $|\alpha|$, and the sign decides for which class the rule votes. The classification is then simply a majority voting.

9.2 SLIPPER

SLIPPER (Cohen and Singer 1999) is the first boosted rule learner, and it is often referred to as weighted sequential covering. This is because it was originally introduced as an instance of AdaBoost with confidence-rated predictions (Schapire and Singer 1999). The examples are not removed totally but their weights decrease (or increase in the case of misclassification). Since AdaBoost was also explained in terms of minimization of the exponential loss (Friedman et al. 2000), SLIPPER can be considered as an instance of ENDER solving Eqs. 14 and 18.

The main difference is that SLIPPER uses post-pruning when generating a single rule. The training set is randomly split into two disjoint sets. The first set is used for rule growing, and the second for rule pruning. In ENDER, instead, specific regularization is used that consists in shrinking, resampling, and computing a rule response over all training examples. The latter approach results in a higher accuracy, as demonstrated later.

In SLIPPER, the optimal number of rules to be generated is determined by an internal cross-validation on the training test. Such an approach can also be applied in ENDER, but the regularization technique used instead should ensure that the rules do not overfit. The experiment shows several error curves that indicate that ENDER is quite insensitive to overfitting.

9.3 LRI

LRI (Weiss and Indurkhya 2000) generates a single rule in the form of a DNF-formula, i.e. disjunction of conjunctions of elementary conditions, instead of a simple conjunction.

LRI uses a specific reweighting scheme (cumulative error), similar to Arc-xf algorithm (Breiman 1996). For the two-class problem, however, this method can also be explained in the context of the loss function minimization, as it was done in Mason et al. (1999) for Arc-xf. It follows that LRI minimizes a specific polynomial loss function using a slightly modified gradient descent technique.

In the m-th iteration of LRI, the classification error is originally measured by:

$$FP^{(m)} + k \cdot FN^{(m)}. \tag{39}$$

In the above formula, $FP^{(m)}$ is a sum of false positive examples:

$$FP^{(m)} = \sum_{i \in R_-} w_i^{(m)},$$

and $FN^{(m)}$ is a sum of false negative examples:

$$FN^{(m)} = \sum_{\Phi(\mathbf{x}_i)=0 \wedge y_i \alpha > 0} w_i^{(m)},$$

with weights $w_i^{(m)}$ being a cumulative number of errors (taken to the power of 3) for each training example. Such weights are elements of a negative gradient of the polynomial loss function minimized on a training set. The value of k is doubled if there are still true positive examples ($y_i \alpha > 0$) to be covered, so $k = 1, 2, 4, \ldots$. It can be easily shown that minimization of the above is equivalent to minimization of:

$$\sum_{i \in R_-} w_i^{(m)} - \frac{k}{k+1} \sum_{\Phi(\mathbf{x}_i)=0} w_i^{(m)}, \tag{40}$$

being a slightly modified version of Eq. 33 with $\frac{k}{k+1}$ instead of $\frac{1}{2}$. This equivalence holds due to the fact that one can add to Eq. 39 the following term:

$$k \sum_{y_i \alpha < 0} w_i^{(m)} = k \left(\sum_{i \in R_-} w_i^{(m)} + \sum_{\Phi(\mathbf{x}_i)=0 \wedge y_i \alpha < 0} w_i^{(m)} \right),$$

which is constant in a given iteration, and divide altogether by $k+1$, without changing the solution of the optimization problem. Let us note that $\frac{k}{k+1} \in [0.5, 1)$, which is a little bit aberrant with respect to the analysis given in Sect. 6. This minimization works properly, however, because the rules are generated for each class separately, using one-versus-all strategy.

For each class, the same number of decision rules is generated. A new example is classified by majority voting, in which each rule has the same strength. LRI can also freeze the set of attributes used to generate rules. After generating a given number of rules, attributes not selected by these rules are ignored in building subsequent ones.

9.4 MLRules

MLRules (Dembczyński et al. 2008a) are derived from the maximum likelihood principle. In the case of binary classification, this algorithm is in fact an instance of ENDER with logit loss (Eq. 11) minimized by the gradient descent technique. In the general case, one can use an elegant generalization of this algorithm to a multi-class problem.

9.5 RuleFit and ensemble of decision trees

RuleFit (Friedman and Popescu 2008) differs from the above algorithms, since decision rules are not generated directly. First, decision trees are used as base classifiers in a forward stagewise procedure, and then the rules are produced from the resulting

trees. Finally, a rule ensemble is fitted by gradient directed regularization that aims at selecting the most relevant rules by using lasso regularization. There is also a possibility to include original attributes as basis functions to complement the rule ensemble with a linear part.

RuleFit can utilize a variety of loss functions, because it uses gradient boosting technique for fitting trees. Originally, in order to solve regression problems, RuleFit uses the Huber loss, and classification problems are solved by using the squared-error ramp loss.

Since RuleFit uses decision trees as base classifiers, this is a right place to discuss the similarities and differences between tree- and rule-based ensembles. Decision rule models share many of the advantages of decision trees. Rules can either work on numerical and categorical attributes. They are also invariant to monotone transformations of them. This invariance provides immunity to the presence of extreme values "outliers" and to change of the measurement scales of the attributes. Also irrelevant attributes are not taken into account by rules. Moreover, the computational issues are similar for both methods.

The main difference in favor of decision rules is that there exists a natural stop criterion for rule construction. This is just the minimal value of $\mathcal{L}_m(\Phi)$ that takes into account the trade-off between discrimination and completeness of rules. In the case of decision trees, one has to define several additional parameters, such as the number of terminal nodes, the minimal number of training examples in a terminal node, or to perform post-pruning. Thanks to this natural stop criterion, the size of decision rules (number of elementary conditions involved in Φ) adapts to the problem. For simple problems, rules contain short condition parts, and for hard problems, the number of elementary conditions gets higher.

Obviously, a single decision rule is a very poor classifier acting on covered objects only. However, the performance of a rule ensemble is comparable with the performance of a tree ensemble. In fact, both models are quite similar being a linear combination of regions Φ in the attribute space:

$$f(\mathbf{x}) = \alpha_0 + \sum_{m=1}^{M} \alpha_m \Phi_m(\mathbf{x}).$$

One can just consider decision trees as a special case of the general rule ensemble.

The difference is in learning procedures. In the case of rule ensembles, each region defined by Φ is built to be optimal, taking into account all previously generated rules. This is not the case of decision trees, where in each iteration several regions are produced. That is why the rule ensemble can contain smaller number of regions Φ, and can be easier in interpretation than the tree ensemble. Moreover, using rules one can generate regions that are not easily obtained by decision trees.

9.6 Ensemble of decision rules and knowledge discovery

The main advantage of rules is their simplicity and interpretability. A question arises, however, whether an ensemble of a high number of rules is still interpretable.

We believe that ENDER can still be used for interpretation purposes. One way is to follow the approach given in Friedman and Popescu (2008) that relies on a post-processing phase in which the rules are refitted by using the lasso regularization. Moreover, also in Friedman and Popescu (2008), a simple measure has been introduced that can be used in order to sort the rules according to their interestingness. A similar approach is also used in association rule mining. Many interestingness measures that were already introduced and characterized in the literature (Hilderman and Hamilton 2001; Greco et al. 2004; Brzezińska et al. 2007) can also be used here.

Another way is to define the learning problem as a multiple criteria decision problem. In this case, one can constrain the number of rules (as a criterion of interpretability) to the constant value (like $M = 10$), and try to set other parameters in order to maximize the performance.

In order to improve the interpretability, one can also try to fit the β parameter of the constant-step minimization technique. This parameter controls the rule coverage, so it can be used in order to find rules with different characteristics: general ones, but less discriminative, or specific ones being "pure".

Another important aspect of the interpretability of decision rules is the following. When a new unseen example is classified, only few rules in the ensemble are "fired" (exactly those covering the example), so they all can be easily interpreted by the user. This also means that in the case of each unseen example, one can give a justification of the final prediction of the rule ensemble in terms of the few "fired" rules. In real applications, like medicine, this is of crucial importance, since the user usually wants to know the reasons of a recommended decision (e.g., therapy or diagnosis).

Let us also mention that the linear combination of rules resembles the expert's scoring procedures often used in medicine or economics. In such procedures, if a certain condition is satisfied, some "points" are added to the final score. In the case of the rule ensemble, the satisfied condition corresponds to a rule covering the unseen example. Basing of the value of the final score the decision is made. Thus, we claim that the model produced by ENDER is close to the practice of many domain experts.

10 Results of a computational experiment

The experiment has been constrained to binary classification problems since the main rule-based competitors of ENDER are usually tailored for this type of the prediction problem. Some of the results on regression data sets for a particular instance of the ENDER framework have been already published in Dembczyński et al. (2008b).

The experiment has been mainly focused on the performance of the rule ensemble. In the first part of the experiment, the ENDER algorithm has been tested with different settings on artificial data. From this experiment, one can draw conclusions about the values of the parameters. In the second part, four variants of ENDER have been compared with existing rule ensemble learning methods: SLIPPER, LRI and RuleFit. The comparison has been carried out on the benchmark data sets taken from the UCI repository (Asuncion and Newman 2007). In all the experiments, the misclassification error has been measured.

At the end of this section, we also present some experimental results confirming that the ENDER algorithm produces models that are easy in interpretation.

10.1 Artificial data

The artificial data have been generated using the following model. Let examples $\mathbf{x} \in \mathbb{R}^n$ be drawn according to the normal distribution, $\mathbf{x} \sim N(0, \mathbf{I})$, where $\mathbf{I}$ is a unit matrix of size n. Assume that the target function $f^*(\mathbf{x}) \in \mathbb{R}$ can be transformed to conditional probabilities $\Pr(y|\mathbf{x})$ in the following way:

$$\log \frac{\Pr(y = 1|\mathbf{x})}{\Pr(y = -1|\mathbf{x})} = \pi f^*(\mathbf{x}),$$

where π corresponds to the level of noise, measured by the Bayes risk of misclassification $R^* = \mathbb{E}_{y\mathbf{x}}[L_{0-1}(y, \text{sgn}(f^*(\mathbf{x})))]$ (i.e., there is one-to-one correspondence between π and R^*). In the main experiment, we set $R^* = 0.1$. The target function $f^*(\mathbf{x})$ has been defined as:

$$f^*(\mathbf{x}) = x_1 - x_2 + 0.2(x_3 - x_4) + 5e^{-\left(x_5^2 + x_6^2 + 0.2x_7^2\right)} - 5\prod_{j=8}^{10} I(-0.5 \leq x_j \leq 0.5)$$
$$+ I(x_{11} \geq 0 \wedge x_{12} \geq 0) - I(x_{13} \geq 0 \wedge x_{14} \geq 0) + \theta, \tag{41}$$

where threshold θ is chosen so that the prior probabilities of both classes are equal: $\Pr(y = 1) = \Pr(y = -1)$. Notice that the target function contains linear terms (which are hard to approximate by trees and rules), a Gaussian term, a cube and two rectangles. Later, we have also added some irrelevant attributes $x_{15}, x_{16}, \ldots$ which do not affect the target function.

First, we have examined the effect of regularization. We compared unregularized algorithms with regularized ones. For the former, the shrinkage parameter has been set to $\nu = 1$, and the fraction of training examples drawn without replacement to $\zeta = 1$. For the latter, some ad hoc values corresponding to high regularization have been taken, namely $\nu = 0.1$, and $\zeta = 0.25$. The rule response has been computed over all training examples.

The ENDER algorithm based on minimization of the exponential (Exp), logit (Log) and sigmoid (Sigm) loss has been applied with constant-step (CS) (with four different values of the step length, $\beta \in \{1, 0.5, 0.2, 0.1\}$), gradient descent (GD) and gradient boosting (GB) minimization techniques. Additionally, simultaneous minimization (SM) has been used to minimize the exponential loss. The size M of the rule ensemble varied in each case from 1 to 1,000 rules, thus for each classifier the error on a testing set has been a function of M. Such functions can be shown in the form of "error curves". Figure 1 shows the error curves being averages over 30 trials; in each of them a training set and a testing set of size $N = 1{,}000$ have been drawn.

One can observe that for the exponential and the logit loss, the regularization results in a slower decrease of the error, but the curves are much smoother, and the overall accuracy is much better.

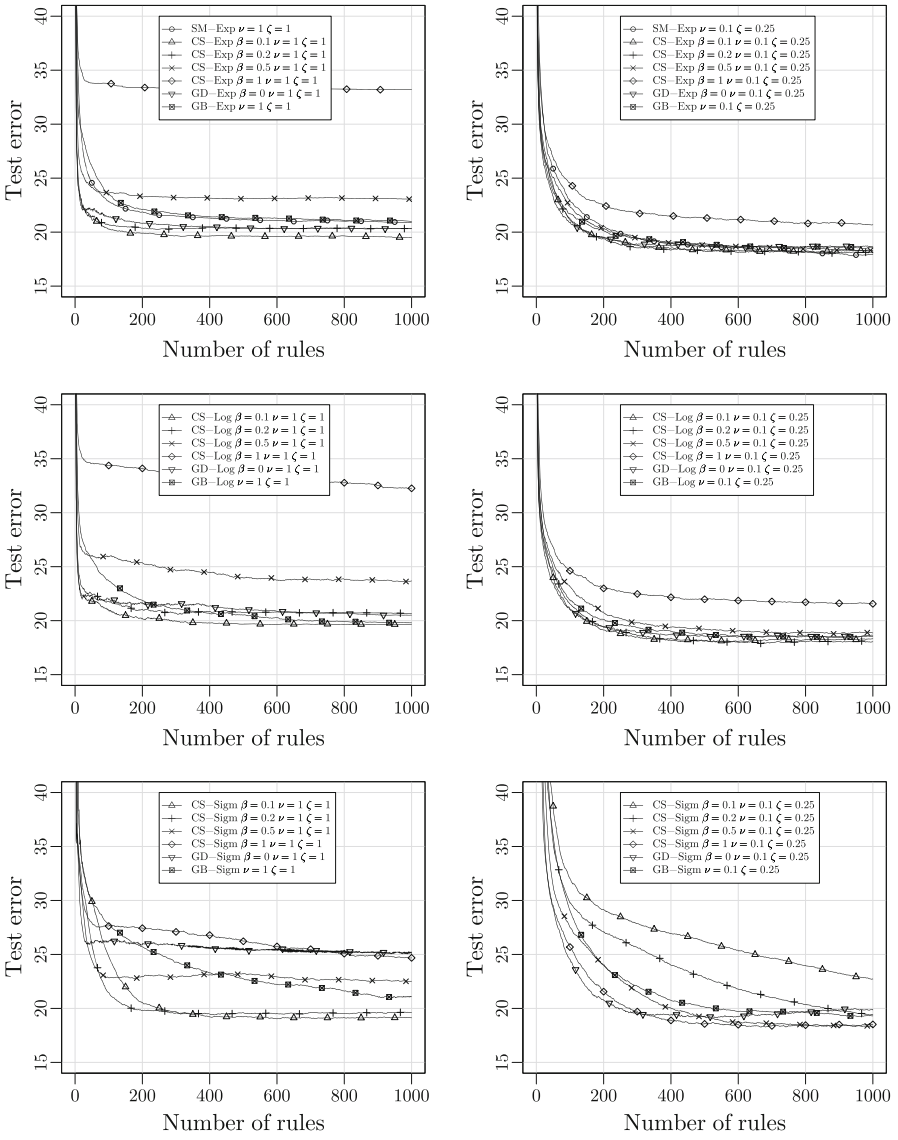

Fig. 1 Experiments on artificial data. Comparison of unregularized (on *left*) and regularized (on *right*) rule ensembles minimizing different loss functions: (from *top*) exponential, logit and sigmoid loss

In the case of the constant step minimization, the best prediction accuracy has been achieved for the small (but non-zero) step length: 0.1–0.2. Let us remind that an increase of β in the case of the exponential and the logit loss results in smaller rules making less mistakes. Thus, neither small and well fitted, nor very general rules (remember that the gradient descent corresponds to the constant step technique with $\beta \to 0$) have achieved the best prediction performance. Notice, however, that the

performance of the algorithms is partially biased by the greedy heuristic building the condition part of the rule. Even the first elementary condition found by this heuristic has to minimize $\mathcal{L}(\Phi)$. This causes that only a subspace of possible condition parts is explored. On the other hand, however, this speeds up the algorithm.

It seems that regularization highly improves the simultaneous minimization technique, which can also be observed when comparing ENDER to SLIPPER, which will be reported later. Gradient boosting works better with regularization, but gets moderate results.

In the case of sigmoid loss, the situation has been a little bit different. For the constant step, that seems to be the most natural minimization technique for this loss function, one can observe an interesting fact. Namely, parameters β and ν are closely related with each other. The length of step β is also a value of the rule response α that is in turn multiplied by ν. One can observe that the solution for $\beta = 0.1$ and $\nu = 1$ is quite similar to that for $\beta = 1$ and $\nu = 0.1$ (with additional sampling, $\zeta = 0.25$). It seems that the step length is the main factor influencing the performance. Sampling in this case does not play an important role. One can also notice that gradient descent is not well adapted to this loss function. The condition part of the rule is constructed for $\alpha \to 0$, but the final response of the rule α is set to be β multiplied by ν. Indeed, the regularized algorithm tends to obtain better results, but it has tendency to overfit. This is the only case in which overfitting occurred.

Table 1 presents the same results as Fig. 1. Test errors for $M = 1{,}000$ are given with standard errors. Only in the case of sigmoid loss, the regularized algorithm may not result in better performance as discussed above. The table contains also information about computation time. The algorithm appeared to be quite efficient. Generation of 1,000 rules ended on commodity hardware (running MS Windows Server 2003, AMD Opteron Processor 250, 2.39 GHz, 8 GB of RAM) around 15 s in the worst case. Duration of rule generation is closely related to rule coverage. Algorithms that produce more general rules take longer time (later on we will discuss the experiment concerning the relation between impurity measures and the rule coverage). It is worth stressing that sampling reduces the computation time in all the cases.

In the next experiment, the best parameters for each loss function have been selected. All minimization techniques have been tested with all combinations of the following values of the parameters: $\nu \in \{1, 0.5, 0.2, 0.1\}$, $\zeta \in \{1, 0.75, 0.5, 0.25\}$. The constant-step minimization has been used with $\beta \in \{1, 0.5, 0.2, 0.1\}$, as before. For each algorithm, an error curve has been drawn showing misclassification error for M varying from 1 to 1,000. Using these curves, the best minimization technique and the best values of the parameters have been chosen for each loss function. The exponential loss has been an exception, where the simultaneous minimization was treated separately from the other techniques. Thus, the following four algorithms have been selected:

- simultaneous minimization with exponential loss (SM-Exp): $\nu = 0.1$, $\zeta = 0.25$,
- constant-step with exponential loss (CS-Exp): $\beta = 0.2$, $\nu = 0.1$, $\zeta = 0.25$,
- constant-step with logit loss (CS-Log): $\beta = 0.2$, $\nu = 0.1$, $\zeta = 0.25$,
- constant-step with sigmoid loss (CS-Sigm): $\beta = 0.2$, $\nu = 0.2$, $\zeta = 0.5$.

For the exponential and the logit loss, the algorithms with the highest regularization have been selected (the same parameters as in the previous experiment, see Fig. 1 and

Table 1 Test errors and standard errors for regularized and unregularized rule ensembles. The computation time is also given

ENDER	Unregularized		Regularized	
	Test error (%)	Time (s)	Test error (%)	Time (s)
SM-Exp	20.877 ± 0.255	4.625	17.940 ± 0.229	1.969
CS-Exp $\beta = 0.1$	19.513 ± 0.286	8.063	18.300 ± 0.235	5.399
CS-Exp $\beta = 0.2$	20.320 ± 0.234	5.296	18.110 ± 0.212	4.735
CS-Exp $\beta = 0.5$	23.040 ± 0.306	3.703	18.240 ± 0.239	2.890
CS-Exp $\beta = 1.0$	33.203 ± 0.687	3.047	20.683 ± 0.267	1.813
GD-Exp $\beta = 0.0$	20.333 ± 0.290	15.515	18.670 ± 0.282	6.062
GB-Exp	20.993 ± 0.240	5.937	18.573 ± 0.227	3.063
CS-Log $\beta = 0.1$	19.653 ± 0.275	5.781	18.323 ± 0.276	5.453
CS-Log $\beta = 0.2$	20.667 ± 0.297	5.312	18.033 ± 0.258	4.359
CS-Log $\beta = 0.5$	23.677 ± 0.277	4.640	18.863 ± 0.240	2.594
CS-Log $\beta = 1.0$	32.257 ± 0.640	2.570	21.560 ± 0.286	1.844
GD-Log $\beta = 0.0$	20.513 ± 0.313	13.625	18.653 ± 0.235	6.219
GB-Log	19.793 ± 0.291	6.093	18.547 ± 0.260	3.125
CS-Sigm $\beta = 0.1$	19.143 ± 0.274	9.265	22.720 ± 0.309	5.521
CS-Sigm $\beta = 0.2$	19.640 ± 0.281	6.968	19.437 ± 0.279	5.484
CS-Sigm $\beta = 0.5$	22.520 ± 0.309	5.203	18.370 ± 0.282	4.641
CS-Sigm $\beta = 1.0$	24.683 ± 0.335	4.704	18.517 ± 0.253	3.484
GD-Sigm $\beta = 0.0$	25.227 ± 0.322	10.187	19.860 ± 0.240	6.484
GB-Sigm $\beta = 0.0$	21.090 ± 0.298	5.953	19.307 ± 0.224	3.187

Table 1). For the sigmoid loss, the parameters have been a little bit different. Among minimization techniques, the constant step minimization with $\beta = 0.2$ appeared to be the best. The error curves for each of the best classifiers are shown in top left panel of Fig. 2. It follows from the figure that none of the classifiers outperforms the others in a significant way. The sigmoid loss tends to minimize the error slower than other loss functions, however, for $M = 1{,}000$ it achieved the same accuracy. Notice that SM-Exp does not decrease the training error as rapidly as CS-Exp, yet the characteristics of both algorithms on the testing set are similar. There is almost no difference between constant step minimization applied for the exponential and the logit loss. The top right panel of Fig. 2 shows learning curves, the errors of the classifiers as functions of the sample size up to $N = 10{,}000$; all classifiers decreased the testing error, but SM-Exp seems to gain the most while N increases.

From the above experiments, one can conclude that regularization has a strong influence on the performance of the rule ensemble, but it is hard to observe a supremacy of one of the loss functions. In the next step, the impact of shrinkage ν and subsample size ζ have been analyzed independently. We focused on SM-Exp, but the relationships appeared to be similar for the logit loss function and other minimization techniques. The bottom panels of Fig. 2 show both dependencies. It follows that shrinkage did

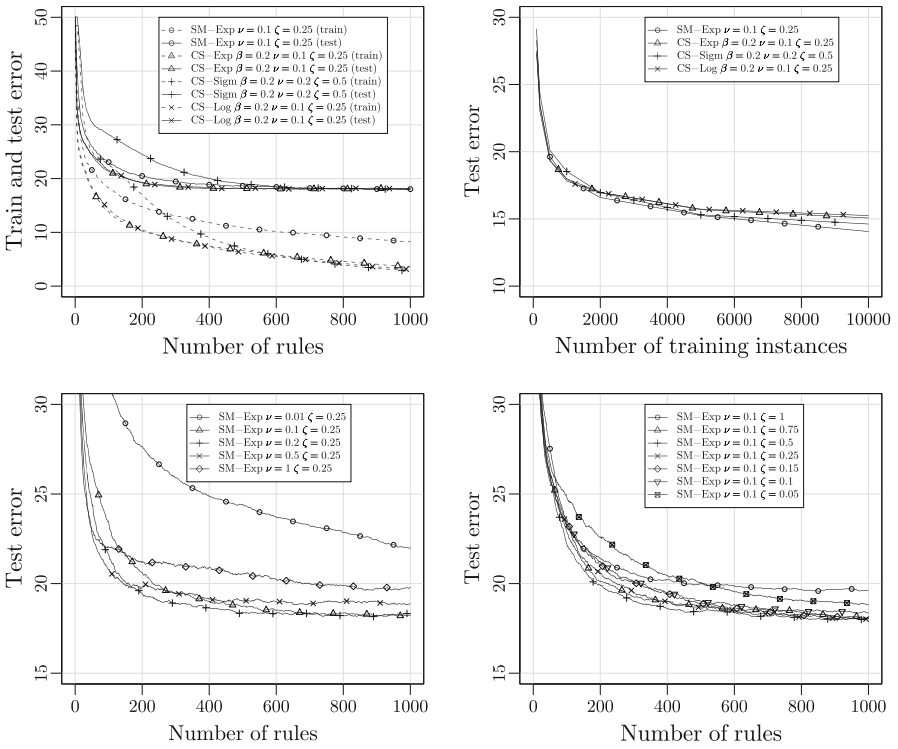

Fig. 2 Experiments on artificial data. *Top left*: error curves for the best algorithms, *top right*: learning curves, *bottom left*: SM-Exp with varying ν and constant $\zeta = 0.25$, *bottom right*: SM-Exp with varying ζ and constant $\nu = 0.1$

improve the accuracy, similarly as it was shown in Hastie et al. (2003). Nevertheless, too strong shrinkage may lead to a very slow learning rate (see the black curve for $\nu = 0.01$). We found out that the optimal range of shrinkage is 0.1–0.2. It also follows that sampling has a positive impact on the accuracy—even very small values of ζ ($\leq$0.25) seem to work very well.

We have also examined the behavior of classifiers when the Bayes risk increases up to the level of 0.3. This is shown in the top left panel in Fig. 3. Obviously, the testing error of the classifiers has increased. However, the shape of the error curve for all the classifiers did not change: we did not observe any significant overfitting. At first sight, it looks contrary to what has usually been observed in boosting experiments: the exponential loss tends to be prone to overfitting since it focuses too much on the incorrectly classified examples, which are typically noisy ones; the sigmoid loss should behave best, since it gives up on the hardest examples (because of the shape of this loss function). No such behavior has been observed. The top right panel in Fig. 3 sheds some light on this phenomenon. The only difference is that the decision of the rule is now calculated on the subsample rather than on the entire training set. This leads to severe overfitting of all classifiers except the one with the sigmoid loss. Thus, calculating the response on all examples makes the classifier robust to noise:

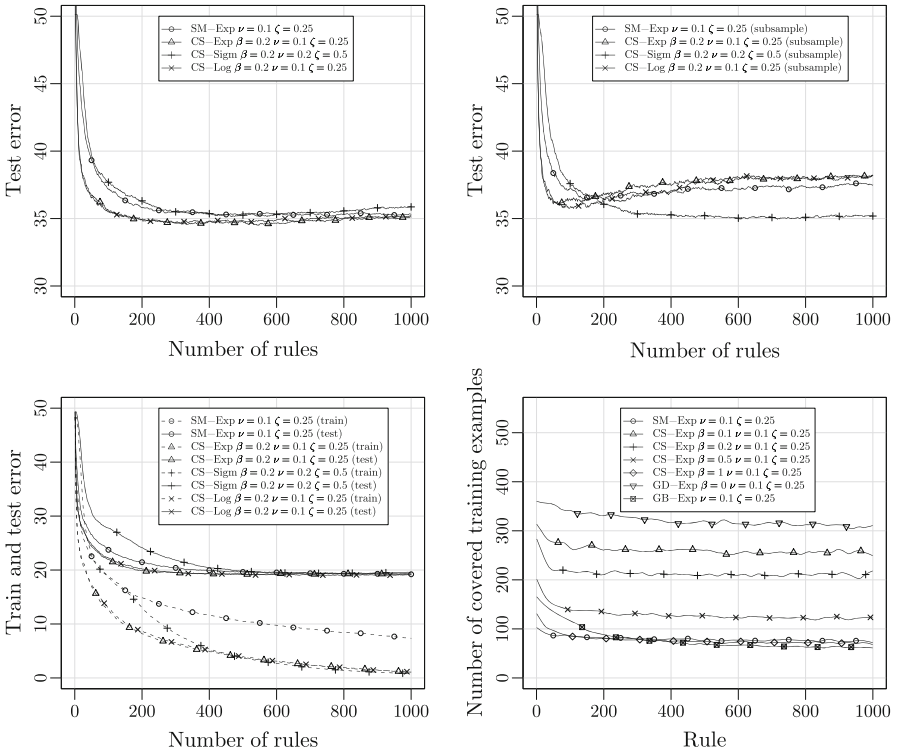

Fig. 3 Experiments on artificial data. *Top left*: error curves for Bayes risk 0.3 (rule response calculated over the whole training set), *top right*: rule response calculated on subsample (Bayes risk 0.3), *bottom left*: error curves for a problem with 20 additional irrelevant features, *bottom right*: coverage of rules (lines are smoothed)

the rules overfitted to the data, will get their responses (decisions) close to 0. This also explains, at least partially, why all loss functions behaved roughly the same. Since the values of the rule responses are relatively small, the ensemble function $f(\mathbf{x})$ takes values in the vicinity of 0 for most of the examples. In this range, all loss functions share a similar characteristic (the main difference between the loss functions is for large negative values of the margin).

Next, some irrelevant attributes have been added to the problem. The bottom left panel in Fig. 3 shows that the presence of 20 irrelevant attributes did not affect ENDER much, regardless of the kind of the loss function used. Notice the similarity between this plot and the top left one in Fig. 2.

Finally, the bottom right plot in Fig. 3 shows the coverage of the rules for different minimization techniques applied to the exponential loss. One can observe that the simultaneous minimization and the gradient boosting have produced rules with much lower coverage than the gradient descent, as stated in Theorem 1 and 2. One can also observe, what has already been anticipated by Theorem 4, that step length determines the coverage of the rule. We have already noticed that the best prediction accuracy is achieved for the small (but non-zero) step length 0.1–0.2: neither small and well fitted,

nor very general rules achieved the best prediction performance. Notice that the possibility of controlling the coverage can also be very helpful in getting a comprehensible set of rules.

10.2 Benchmark data

In the second part of the experiment, ENDER has been compared to other rule ensemble algorithms: LRI, SLIPPER and RuleFit. The following parameters have been used for each method:

- SLIPPER: the maximum number of iterations was set to 500, the rest of parameters remained default (the internal cross validation for choosing the optimal number of rules was switched on).
- LRI: according to the experiment in Weiss and Indurkhya (2000), the rule consisted of 2 disjuncts of length 5, feature selection was frozen after 50 rounds, and 200 rules were generated per class.
- RuleFit: according to the experiment in Friedman and Popescu (2008), the mixed rule-linear mode was chosen, average tree size was set to 4, the number of trees was increased to 500, and sample fraction was set as default.
- ENDER: the best four classifiers from the artificial data experiment were taken, for all classifiers $M = 500$.

The experiment has been performed on 20 binary classification problems, all taken from the UCI Repository (Asuncion and Newman 2007). The description of each data set is given in Table 2. Each test has been performed using 10-fold cross validation (with exactly the same train/test splits for each classifier) and the average 0-1 loss on testing folds was calculated. The results are shown in Table 3.

To compare multiple classifiers on multiple data sets, the Friedman test has been applied as suggested by Demšar (2006). This test uses ranks of each algorithm to check whether all the algorithms perform equally well (null hypothesis). Friedman statistics gave 35.636 which exceeds the critical value 12.592 (for confidence level 0.05), and thus the null hypothesis has been rejected. Next, we passed to a post-hoc analysis and calculated the *critical difference* (CD) according to the Nemenyi statistics. The value obtained is CD $= 2.015$, which means that algorithms with difference in average ranks greater than 2.015, are significantly different. In Fig. 4, average ranks have been marked on a line, and groups of the classifiers that are not significantly different were connected. This shows that all ENDER algorithms outperform all of the competitors. However, CS-Exp and SM-Exp are significantly better than SLIPPER and RuleFit, but CS-Sigm and CS-Log are significantly better than RuleFit. None of the ENDER algorithms is significantly better than LRI. On the other hand, none of the three well-known rule ensemble algorithms (LRI, SLIPPER, RuleFit) has appeared to be significantly better than any other.

The results have confirmed the main issues of the experiment performed on the artificial data. The choice of the loss function does not seem to be a critical issue in the learning process. More important is the use of regularization that consists of shrinking, resampling and calculating the response of the rule on entire data set rather than on

Table 2 Data sets used in the experiment

Data set	Number of training examples	Number of attributes
HABERMAN	306	3
BREAST-C	286	9
DIABETES	768	8
CREDIT-G	1,000	20
CREDIT-A	690	15
IONOSPHERE	351	34
COLIC	368	22
HEPATITIS	155	19
SONAR	208	60
HEART-STATLOG	270	13
LIVER-DISORDERS	345	6
VOTE	435	16
HEART-C-2	303	13
HEART-H-2	294	13
BREAST-W	699	9
SICK	3,772	29
TIC-TAC-TOE	958	9
SPAMBASE	4,601	57
CYLINDER-BANDS	540	39
KR-VS-KP	3,196	36

the subsample only. This conclusion can be drawn from the fact that SM-Exp is very similar to SLIPPER, but SM-Exp applying these techniques gave much better results.

10.3 Interpretability of rule ensembles

The last part of the experiment examines the interpretability of the rule ensemble. In the previous experiments, ENDER generated up to 1,000 rules. Such an ensemble is certainly not easy in interpretation. However, one can sort the rules using some interestingness measure, or try to refit the ensemble to choose the most important rules only. Another approach taken here relies on parameterizing ENDER in such a way that only few rules maintaining good performance are generated.

Consider the BREAST- CANCER data set. We have used ENDER with the exponential loss and the constant-step minimization technique. We have limited the number of rules to 3. We have not performed any regularization, i.e. no shrinking and sampling has been used. We have modified the parameter β only that controls the trade-off between misclassification and the coverage of the rules. It has been set up to obtain good performance and interpretable rules.

Table 3 Test errors and ranks (in parenthesis). In the last row, the average rank is computed for each classifier

Data set	CS-Log	SM-Exp	CS-Exp	CS-Sigm	SLIPPER	LRI	RuleFit
HABERMAN	26.8 (4.5)	25.5 (1.0)	26.2 (3.0)	25.8 (2.0)	26.8 (4.5)	27.5 (7.0)	27.2 (6.0)
BREAST-C	28.3 (5.0)	27.9 (3.0)	27.2 (1.0)	27.3 (2.0)	27.9 (4.0)	29.3 (6.0)	29.7 (7.0)
DIABETES	24.5 (2.0)	24.6 (3.5)	24.6 (3.5)	23.6 (1.0)	25.4 (6.0)	25.4 (5.0)	26.2 (7.0)
CREDIT-G	23.3 (2.0)	23.5 (3.0)	22.8 (1.0)	24.2 (5.0)	27.7 (7.0)	23.9 (4.0)	25.9 (6.0)
CREDIT-A	13.5 (4.5)	13.5 (4.5)	12.3 (2.0)	13.8 (6.0)	17.0 (7.0)	12.2 (1.0)	13.2 (3.0)
IONOSPHERE	6.3 (3.0)	6.0 (2.0)	5.7 (1.0)	6.5 (4.5)	6.5 (4.5)	6.8 (6.0)	8.5 (7.0)
COLIC	15.0 (5.0)	14.7 (3.5)	14.4 (2.0)	12.8 (1.0)	15.0 (6.0)	16.1 (7.0)	14.7 (3.5)
HEPATITIS	19.5 (7.0)	18.2 (4.0)	18.8 (5.0)	16.2 (1.0)	16.7 (2.0)	18.0 (3.0)	19.4 (6.0)
SONAR	16.8 (5.0)	15.4 (3.0)	16.4 (4.0)	14.5 (1.0)	26.4 (7.0)	14.9 (2.0)	19.7 (6.0)
HEART-STATLOG	16.7 (1.0)	17.0 (2.0)	17.4 (3.5)	17.4 (3.5)	23.3 (7.0)	19.6 (6.0)	18.5 (5.0)
LIVER-DISORDERS	26.4 (4.0)	25.8 (3.0)	24.8 (1.0)	24.9 (2.0)	30.8 (7.0)	26.6 (5.0)	30.7 (6.0)
VOTE	3.2 (1.0)	3.4 (2.5)	3.4 (2.5)	4.6 (5.0)	5.0 (6.0)	3.9 (4.0)	5.0 (7.0)
HEART-C-2	16.9 (4.0)	15.5 (3.0)	15.2 (1.0)	15.5 (2.0)	19.5 (7.0)	18.5 (5.0)	18.9 (6.0)
HEART-H-2	17.0 (1.0)	17.6 (3.0)	17.3 (2.0)	19.3 (6.0)	20.0 (7.0)	18.2 (4.0)	18.3 (5.0)
BREAST-W	3.9 (4.5)	3.9 (4.5)	3.6 (3.0)	3.1 (1.0)	4.3 (7.0)	3.3 (2.0)	4.1 (6.0)
SICK	1.5 (1.0)	1.6 (3.0)	1.8 (4.0)	6.1 (7.0)	1.6 (2.0)	1.8 (5.0)	1.9 (6.0)
TIC-TAC-TOE	0.9 (1.0)	4.2 (3.0)	8.1 (5.0)	19.0 (7.0)	2.4 (2.0)	12.2 (6.0)	5.3 (4.0)
SPAMBASE	5.2 (4.0)	4.6 (2.0)	4.5 (1.0)	5.2 (5.0)	5.9 (7.0)	4.9 (3.0)	5.9 (6.0)
CYLINDER-BANDS	21.9 (6.0)	18.7 (3.0)	19.4 (4.0)	15.4 (1.0)	21.7 (5.0)	16.5 (2.0)	38.1 (7.0)
KR-VS-KP	0.9 (2.0)	0.9 (3.0)	1.0 (4.0)	3.5 (7.0)	0.6 (1.0)	3.1 (6.0)	2.9 (5.0)
AVG. RANK	3.38	2.98	2.68	3.5	5.3	4.45	5.73

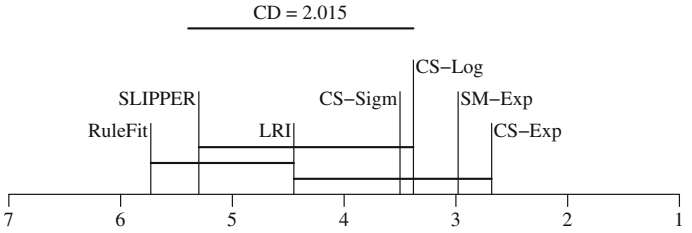

Fig. 4 Critical difference diagram

Table 4 Decision rules for the BREAST- CANCER data set. The decision part of the rule specifies the class (sign of the rule response) and the weight (absolute value of the rule response) of the rule (in the first parentheses), as well as the number of training examples that are correctly classified and misclassified by the rule (in the second parentheses)

#	Rule
	if DEFAULT RULE
	then NO-RECURRENCE EVENTS (0.43) (201:85)
1	*if* TUMOR SIZE $\leq$14.5
	and INV-NODES $\leq$5.5
	and AGE $\geq$40
	then NO-RECURRENCE EVENTS (1.57) (34:0)
2	*if* DEG-MALIG $\geq$2.5
	and INV-NODES $\geq$2.5
	and TUMOR SIZE $\leq$44
	then RECURRENCE EVENTS (0.91) (27:9)
3	*if* TUMOR SIZE $\leq$14.5
	and INV-NODES $\leq$5.5
	and BREAST-QUAD IS NOT CENTER
	then NO-RECURRENCE EVENTS (0.96) (32:0)

The rules are presented in Table 4. Parameter β has been finally fixed on 0.6. In 10-fold cross-validation, the algorithm has obtained the misclassification error equal to 26.9, which is even better than results obtained by other algorithms (see Table 3).

As we can see, the first and the third rule are supporting the default rule, and only the second rule indicates the "recurrence events". Such a model is still appropriate, since we obtain different levels of no-recurrence and recurrence events.

There is, however, still some room for improving the interpretability of rule ensembles. One can try, for example, to prune the obtained rules (i.e., remove some elementary conditions). As we can see from the example, the first and the third rule are very similar. A deeper insight shows that the last elementary conditions change slightly the coverage of these rules. One can also consider the use of another heuristic for building condition parts of decision rules. This is, however, postponed to a future work.

11 Conclusions

The main contribution of this paper is a proposal of a general framework for rule induction, that we called ENDER. In this framework, an ensemble of decision rules is constructed by boosting or forward stagewise additive modeling. We have shown that several other approaches to rule induction, including the sequential covering, fall into this framework. The ENDER algorithm has been analyzed, both theoretically and experimentally.

From the theoretical analysis, it follows that the minimization technique influences the coverage of the rule. The most general rules (i.e., rules with the highest coverage) are produced by the gradient descent. This technique is a particular case of the constant-step minimization, with the step length tending to 0. The constant-step minimization technique is particularly well-tailored for decision rules, since the step length controls the rule coverage. This feature is interesting from knowledge discovery point of view. The simultaneous minimization produces smaller rules than the gradient descent. The gradient boosting and the gradient descent result in a similar form of the functional to be minimized, but the former penalizes larger rules.

It follows that the choice of the loss function has only a little impact on the accuracy of predictions. We showed, however, that the use of regularization may significantly improve the performance. The regularization in ENDER consists in shrinkage and sampling. Moreover, the rule response is computed over all training examples, independently of the fact whether a rule was built using a subsample or not. This regularization constitutes an alternative to post-pruning that is often used in induction of decision rules. In the experiments, ENDER outperformed all other rule induction algorithms. Let us finally remark that the learning procedure is very fast and proved to be efficient in computational experiments.

We hope that this paper will increase the interest in rule-based classifiers that seem to be at least as attractive as decision trees.

Acknowledgements The authors wish to acknowledge financial support from the Polish Ministry of Science and Higher Education, grant no. N N519 314435.

References

Asuncion A, Newman DJ (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html

Bazan JG (1998) Discovery of decision rules by matching new objects against data tables. In: Polkowski L, Skowron A (eds) Rough sets and current trends in computing, volume 1424 of Lecture notes in artificial intelligence. Springer, Warsaw, pp 521–528

Błaszczyński J, Dembczyński K, Kotłowski W, Słowiński R, Szeląg M (2006) Ensembles of decision rules. Found Comput Decis Sci 31(3–4):221–232

Boros E, Hammer PL, Ibaraki T, Kogan A, Mayoraz E, Muchnik I (2000) An implementation of logical analysis of data. IEEE Trans Knowl Data Eng 12:292–306

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Brzezińska I, Greco S, Słowiński R (2007) Mining Pareto-optimal rules with respect to support and confirmation or support and anti-support. Eng Appl Artif Intell 20(5):587–600

Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3:261–283

Cohen WW (1995) Fast effective rule induction. In: Proceedings of the twelfth international conference of machine learning (ICML 1995). Morgan Kaufmann, Tahoe City, pp 115–123

Cohen WW, Singer Y (1999) A simple, fast, and effective rule learner. In: Proceedings of the sixteenth national conference on artificial intelligence. AAAI Press/The MIT Press, Orlando, pp 335–342

Dembczyński K, Kotłowski W, Słowiński R (2008a) Maximum likelihood rule ensembles. In: Proceedings of the twenty-fifth international conference on machine learning (ICML 2008). Omnipress, Helsinki, pp 224–231

Dembczyński K, Kotłowski W, Słowiński R (2008b) Solving regression by learning an ensemble of decision rules. In: Rutkowski L, Tadeusiewicz R, Zadeh LA, Zurada JM (eds) Artificial intelligence and soft computing, volume 5097 of Lecture notes in artificial intelligence. Springer, Zakopane, pp 533–544

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Dietterich TG (2000) An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Mach Learn 40(2):139–158

Domingos P (1996) Unifying instance-based and rule-based induction. Mach Learn 24(2):141–168

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232

Friedman JH, Popescu BE (2003) Importance sampled learning ensembles. Technical report, Department of Statistics, Stanford University

Friedman JH, Popescu BE (2008) Predictive learning via rule ensembles. Ann Appl Stat 2(3):916–954

Friedman JH, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting (with discussion). Ann Stat 28(2):337–407

Fürnkranz J (1996) Separate-and-conquer rule learning. Artif Intell Rev 13(1):3–54

Góra G, Wojna A (2002a) Local attribute value grouping for lazy rule induction. In: Peters JF, Skowron A, Zhong N (eds) Rough sets and current trends in computing, volume 2475 of Lecture notes in artificial intelligence. Springer, Malvern, pp 405–412

Góra G, Wojna A (2002b) A new classification system combining rule induction and instance-based learning. Fundam Inform 54(4):369–390

Greco S, Matarazzo B, Słowiński R, Stefanowski J (2000) An algorithm for induction of decision rules consistent with the dominance principle. In: Ziarko W, Yao Y (eds) Rough sets and current trends in computing, volume 2005 of Lecture notes in artificial intelligence. Springer, Banff, pp 304–313

Greco S, Matarazzo B, Słowiński R (2001) Rough sets theory for multicriteria decision analysis. Eur J Oper Res 129:1–47

Greco S, Pawlak Z, Słowiński R (2004) Can Bayesian confirmation measures be useful for rough set decision rules. Eng Appl Artif Intell 17(4):345–361

Grzymala-Busse JW (1992) 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, Dordrecht, pp 3–18

Hastie T, Tibshirani R, Friedman JH (2003) Elements of statistical learning: data mining, inference, and prediction. Springer, New York

Hilderman RJ, Hamilton HJ (2001) Knowledge discovery and measures of interest. Kluwer, Boston

Janssen F, Fürnkranz J (2008) An empirical investigation of the trade-off between consistency and coverage in rule learning heuristics. In: Boulicaut J-F, Berthold MR, Horváth T (eds) Discovery science, volume 5255 of Lecture notes in artificial intelligence. Springer, Budapest, pp 40–51

Jovanoski V, Lavrac N (2001) Classification rule learning with APRIORI-C. In: Brazdil P, Jorge A (eds) Progress in artificial intelligence, volume 2258 of Lecture notes in artificial intelligence. Springer, Berlin , pp 111–135

Kearns MJ, Vazirani UV (1994) An introduction to computational learning theory. MIT Press, Cambridge

Knobbe A, Crémilleux B, Fürnkranz J, Scholz M (2008) From local patterns to global models: the LeGo approach to data mining. In: Fürnkranz J, Knobbe A (eds) Proceedings of the ECML/PKDD 2008 workshop "From local patterns to global models", Antwerp, Belgium

Koltchinskii V, Panchenko D (2006) Complexities of convex combinations and bounding the generalization error in classification. Ann Stat 33(4):1455–1496

Marchand M, Shawe-Taylor J (2002) The set covering machine. J Mach Learn Res 3:723–746

Mason L, Baxter J, Bartlett P, Frean M (1999) Functional gradient techniques for combining hypotheses. In: Bartlett P, Schölkopf B, Schuurmans D, Smola AJ (eds) Advances in large margin classifiers. MIT Press, Cambridge, pp 33–58

Michalski RS (1983) A theory and methodology of inductive learning. In: Michalski RS, Carbonell JG, Mitchell TM (eds) Machine learning: an artificial intelligence approach. Tioga Publishing, Palo Alto, PP 83–129

Pawlak Z (1991) Rough sets. Theoretical aspects of reasoning about data. Kluwer, Dordrecht

Rückert U, Kramer S (2008) Margin-based first-order rule learning. Mach Learn 70(2–3):189–206

Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336

Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. Ann Stat 26(5):1651–1686

Skowron A (1995) Extracting laws from decision tables—a rough set approach. Comput Intell 11:371–388

Słowiński R (ed) (1992) Intelligent decision support. Handbook of applications and advances of the rough set theory. Kluwer, Dordrecht

Stefanowski J (1998) On rough set based approach to induction of decision rules. In: Skowron A, Polkowski L (eds) Rough set in knowledge discovering. Physica Verlag, Heidelberg, pp 500–529

Stefanowski J, Vanderpooten D (2001) Induction of decision rules in classification and discovery-oriented perspectives. Int J Intell Syst 16(1):13–27

Weiss SM, Indurkhya N (2000) Lightweight rule induction. In: Proceedings of the seventeenth international conference on machine learning (ICML 2000). Morgan Kaufmann, Stanford, pp 1135–1142