Towards Tight Bounds for Rule Learning

Ulrich Rückert RUECKERT@IN.TUM.DE
Stefan Kramer KRAMER@IN.TUM.DE
Technische Universität München, Institut für Informatik/I12, Boltzmannstr. 3, D-85748 Garching b. München, Germany

Abstract

While there is a lot of empirical evidence showing that traditional rule learning approaches work well in practice, it is nearly impossible to derive analytical results about their predictive accuracy. In this paper, we investigate rule-learning from a theoretical perspective. We show that the application of McAllester's PAC-Bayesian bound to rule learning yields a practical learning algorithm, which is based on ensembles of weighted rule sets. Experiments with the resulting learning algorithm show not only that it is competitive with state-of-the-art rule learners, but also that its error rate can often be bounded tightly. In fact, the bound turns out to be tighter than one of the "best" bounds for a practical learning scheme known so far (the Set Covering Machine). Finally, we prove that the bound can be further improved by allowing the learner to abstain from uncertain predictions.

1. Introduction

Rule learning has a long history within the field of machine learning. One of the main reasons for the popularity of rule learning is that rules are considered to be a convenient representation of regularities and interrelations for humans. Many learning settings can be adapted to allow for the application of rule learning systems. Usually, the goal in rule learning is to find a set of rules that has high predictive accuracy and that is as small as possible. Historically, most research on rule learning has concentrated on separate-and-conquer approaches, often combined with sophisticated pruning and postprocessing methods to avoid overfitting. While there is a lot of empirical evidence showing that those approaches work very well in practice, it is very hard to investigate them analytically. This is due to the combinatorial complexity inherent in rule learning and the fact that the impact of pruning and postprocessing methods on predictive accuracy is hard to estimate a priori. In contrast, decision trees have been the subject of many empirical and theoretical investigations (e.g. (Breiman, 2001; Golea et al., 1998; Mansour & McAllester, 2000)). The main goal of this work is to design a rule learning system that is not only competitive with established rule learning systems in terms of predictive accuracy, but that is also accessible to an analytical investigation. In particular, we would like to give non-trivial upper bounds on the prediction error of the proposed algorithm.

Of course, rule learning algorithms are not suited for all learning settings. If a data set features many (possibly related or noisy) continuous attributes, a conjunction of literals of the form "attribute $<$ value" allows only for a rough and angular separation of the instance space. In those cases it is usually more sensible to use smooth separating functions as in Support Vector Machines or Neural Networks. In this work we restrict ourselves therefore to nominal-valued data sets. Another problem is that pure rule learning algorithms tend to be *unstable*, in the sense that a small change of the training set often leads to a large change in the induced rule set. This is due to the combinatorial nature of rule learning. The instability can increase the variance component of the prediction error. We therefore use ensembles of rule sets to keep the variance low and improve predictive accuracy.

The paper is organized as follows: after introducing the learning setting in section 2, we describe a novel rule learning system in sections 3 and 4. In section 5 we investigate the system theoretically, and then present empirical results in section 6.

2. The Learning Setting

To give a precise description of the learning algorithm and the underlying theory, we need to introduce a few definitions and state basic assumptions. First of all, training and test data are given as sets of labeled *instances*, taken from an *instance space* $\mathcal{X}$. The instance space is structured as the cartesian product of n domains A_i, so that each instance x

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

is represented by an n-tuple $(x_1, x_2, \ldots, x_n)$, where each x_i denotes a value taken from A_i. We require that all domains are finite, so continuous attributes need to be discretized in order to fit into our framework. Furthermore, let $\mathcal{Y}$ be a finite set of *classes*. We assume a fixed, but unknown distribution D on the pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. A training set S is generated by drawing m *labeled instances* (x, y) independently according to D. The learning algorithm is given the sample S. Upon termination it outputs a concept c that maps instances to classes. The main goal, of course, is to come up with an algorithm that generates a concept c with provably low error $\mathbf{Pr}_{(x,y)\sim D}[c(x) \neq y]$.

Since we are working in the field of rule learning, we use a particular representation for the concepts to be learned. First of all, a *literal* is of the form $(a_i = v_i)$ or $\neg(a_i = v_i)$ for any $1 \leq i \leq n$, where a_i identifies the attribute i and v_i is a value taken from the corresponding domain A_i. A conjunction of literals is a *rule*. A rule is said to *cover* an instance, if the conditions imposed by all literals in the rule are met by the instance. Not surprisingly, a *rule set* is a set of rules. A rule set covers an instance, if any rule in the rule set covers it. If we have p distinct class labels, a *labeled ruleset* $r_i := (r, y_i)$ is a ruleset together with a class label. We can use a labeled rule set r_i to predict the class $r_i(x)$ given an instance x:

$$r_i(x) := \begin{cases} +1 & \text{if } x \text{ is covered by } r_i \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

The class of all possible y_i-labeled rule sets is denoted by R_i. In the next section we deal with the problem of finding rule sets that agree with the labeled instances of a training set.

3. Stochastic Local Search

One can easily transform any learning problem with nominal attributes into an equivalent problem containing only boolean attributes. If we do so, the problem of finding rule sets of a fixed size with low error on a given training set becomes the optimization version of the *k-term DNF learning problem*:

Given

- a set of n attributes $A_1, \ldots, A_n$, where all $A_i \in \{-1, +1\}$,
- a training set of m labeled instances $(x_1, y_1), \ldots, (x_m, y_m)$, where all $x_i \in \prod_i A_i$ and all $y_i \in \{-1, +1\}$,
- a natural number k

Find a +1-labeled rule set r containing exactly k rules over the literals on $A_1, \ldots, A_n$, so that the number of misclassified instances $\Sigma_{1\leq i\leq n}\mathrm{I}(r(x_i) \neq y_i)$ is minimal (I(.) is the indicator function).

Unfortunately this problem is known to be NP-hard (Kearns & Vazirani, 1994). We are therefore forced to use an approximation algorithm to keep the computational costs of learning low. The set covering approach, that is taken by most traditional rule learning systems, does not allow to restrict the size of the induced rule set and is therefore not suited for our purposes. Instead, we use a *stochastic local search* (SLS) approach (Hoos, 1998). SLS algorithms differ from other approaches in that they perform a local, random walk search. In its basic incarnation, an SLS algorithm starts with a randomly generated solution candidate. It then iterates in a two-step loop: in the first step it examines a fixed set of "neighboring" candidates according to some predefined neighborhood relation. Each neighbor is evaluated according to a global scoring function. In the second step the SLS algorithm selects the neighbor with the best score as next candidate. Such a greedy hill climbing approach is obviously susceptible to getting caught in local optima. Most SLS algorithms therefore select with a fixed probability p a random neighboring candidate instead of the candidate with the highest score. In this way they can escape local optima through random steps. In the following we describe an SLS algorithm for k-term DNF learning, that has been shown to work well on hard learning problems (Rückert & Kramer, 2003).

For k-term DNF learning, the score to be minimized is the number of misclassified instances. The SLS algorithm starts with a random rule set r of k rules. It then randomly picks a misclassified instance x. Assume x is labeled +1, but not covered by r. Thus, r is obviously too specific: we have to remove at least one literal in order to let r cover x. Consequently, the algorithm generalizes the rule t in r that differs in the smallest number of literals from x. It generates for each literal l_i in t a new "neighboring" rule set by removing l_i from t. The neighboring rule set with the lowest score is the new candidate to be further optimized. If the randomly chosen instance x is labeled -1, but erroneously covered by r, the current rule set r is too general. Let t be the rule that covers x. We have to add a literal to t in order to make x uncovered. Again we can generate a set of neighbors by adding one literal to t and then choose that neighbor whose score is the lowest. In order to escape local optima, the algorithm replaces each decision step with a random decision from time to time. Algorithm 1 sketches the idea. A more elaborate description of the algorithm can be found in (Rückert & Kramer, 2003).

4. Learning Ensembles of Rule Sets

Learning algorithms inducing individual rule sets are unstable in the sense that small perturbations in the training data

Algorithm 1 An SLS algorithm for k-term DNF learning. k is the number of rules per rule set. Global variables taking default values: $maxSteps$ specifies the maximal length of search, p_{g1}, p_{g2}, and p_s are the probabilities for random steps.

procedure SLSearch(k)
  $r \leftarrow$ a randomly generated rule set with k rules
  $steps \leftarrow 0$
  while $score(r) \neq 0$ and $steps < maxSteps$ **do**
    $steps \leftarrow steps + 1$
    $x \leftarrow$ a random instance that is misclassified by r
    if x is labeled +1 **then**
      $t \leftarrow$ { with probability p_{g1}: a random rule in r ; otherwise: the rule in r that differs in the smallest number of literals from x
      $l \leftarrow$ { with probability p_{g2}: a random literal in t ; otherwise: the literal in t whose removal decreases $score(r)$ most
      $r \leftarrow r$ with l removed from t
    else if x is labeled -1 **then**
      $t \leftarrow$ a (random) term in r that covers x
      $l \leftarrow$ { with probability p_s: a random literal m so that $t \wedge m$ does not cover x ; otherwise: a random literal whose addition to t decreases $score(r)$ most
      $r \leftarrow r$ with l added to t
    end if
  end while
  return the best r found so far
end procedure

Algorithm 2 The rule learning algorithm. S is the training set, k_{max} the maximal number of rules per rule set, o the number of rule sets per ensemble and rule set size k, and ν is the noise probability.

procedure LearnEnsemble(S, k_{max}, o, ν)
  for $cl = 1$ to p **do**
    $Q_{cl} \leftarrow$ empty ensemble
    for $k = 1$ to k_{max} **do**
      for $s = 1$ to o **do**
        $rs \leftarrow$ SLSearch(k)
        $w \leftarrow \exp(-\nu\, \hat{l}(rs, S))$
        $Q_{cl} \leftarrow Q_{cl} \cup \{(rs, w)\}$
      end for
    end for
    Normalize the weights in e_{cl} to sum up to one
  end for
  return $\{Q_1, \ldots, Q_p\}$
end procedure

may result in large changes in the rules. In order to avoid unstable behavior, we now consider learning ensembles of rule sets. Algorithm 2 learns one ensemble per class. The rule sets in the ensemble are sampled using the SLS algorithm. Since we do not know the the optimal number of rules per rule set k in advance, we sample a constant number of rule sets for each k up to a user-specified limit k_{max}.

Of course, the sampled rule sets differ in their ability to explain the training set. It makes sense to keep, along with each rule set, a probability $Q(r_i)$ depending on the R_i's error on the training set. If the user has information about the noise behavior of the underlying target distribution, she can use this information to generate a matching noise model and generate Q accordingly. If, for example, the noise level is higher for some parts of the instance space $\mathcal{X}$, one would want to base $Q(r_i)$ on the misclassified instances at hand: if a rule set misclassifies only "noisy" instances, a higher probability is assigned than for a rule set that misclassifies relatively stable instances. As a conservative default strategy we use a Q that assigns a probability that is exponentially decreasing with its empirical error on S to each r_i:

$$Q(r_i) := \frac{\exp(-\nu\, \hat{l}(r_i, S))}{\Sigma_{Q(r_i)\neq 0} \exp(-\nu\, \hat{l}(r_i, S))}, \qquad (2)$$

where $\hat{l}(r_i, S)$ denotes the empirical error of r_i on the sample S (cf. section 5), and ν specifies the rate of decay. This corresponds to a white noise model, where the error probability is constant $\alpha := \exp(-\nu/m)$ across all instances. Similar models have been studied, e.g., in (Littlestone & Warmuth, 1994).

The learning algorithm should also be able to handle multi-class problems directly. Up to now, we only consider the case of two-class problems. Dealing with multi-class problems has traditionally been problematic for rule learning systems. For the sake of simplicity, we employ a simple one-vs-all scheme here: we learn p ensembles Q_i, where ensemble Q_i distinguishes between class label y_i and $\mathcal{Y} \setminus \{y_i\}$. As the scores can be seen as a measure of how certain an ensemble is about its prediction (cf. section 5.2), it makes sense to choose that class y_i with the largest score $c(Q_i, x)$.

5. Bounds for Rule Learning

In the following, we will present a theoretical analysis of the above algorithm. The goal is to bound its expected prediction error. In the first part, we show how McAllester's PAC-Bayesian theorem can be used to bound the error in ensemble rule learning. In the second part, we prove that the PAC-Bayesian bound can be further improved by allowing the learner to abstain from uncertain predictions. Together, these results are more generally applicable, but they serve well the purpose of bridging the gap between theory and practice for rule learning.

5.1. The PAC-Bayesian Bound for Rule Learning

Two-class problems are generally easier to handle than multi-class problems. We therefore use *characteristic functions* χ_i to split a p-class problem into p 2-class problems: let $\mathcal{Y}$ be a set of p class labels. Just like $r_i(x)$, $\chi_i(y)$ is defined to be +1, if $y = y_i$, and -1 otherwise. To estimate the rule set's predictive behavior we are especially interested in the cases where the class y_i assigned by the rule set disagrees with the "true" class y of an instance x. To measure the rate of misclassification, we introduce a *loss function*: let (x, y) be a labeled instance drawn according to D, and r_i a labeled ruleset. Then the *loss* $l(r_i, x, y) := \mathrm{I}(r_i(x) \neq \chi_i(y))$ is 0, if the prediction agrees with the "true" class y, and 1 otherwise. Intuitively, l is a 0-1 loss for measuring whether the prediction of a rule set agrees with an observation drawn from D, when we are only distinguishing between class label y_i and the remaining class labels.

The *expected loss* $l(r_i) := \mathbf{E}_{(x,y)\sim D}[l(r_i, x, y)]$ indicates, how often the rule set r_i disagrees with the "true class" on average. Given a sample S of size m, the *empirical loss* $\hat{l}(r_i, S) := \frac{1}{m}\Sigma_{(x,y)\in S} l(r_i, x, y)$ yields the fraction of instances in S that are misclassified by r_i. The PAC-Bayesian theorem deals with (prior and posterior) distributions on the space R_i of y_i-labeled rule sets. If Q is an arbitrary probability measure on R_i, we can extend the notion of a loss to distributions instead of single rule sets: $\hat{l}(Q, S) := \mathbf{E}_{r_i\sim Q}[\hat{l}(r_i, S)]$ and $l(Q) := \mathbf{E}_{r_i\sim Q}[l(r_i)]$. Furthermore, to compare prior and posterior distributions, we need the well known Kullback-Leibler divergence, denoted by $D(Q\|P) := \Sigma_{r\in R}(Q(r)\ln\frac{Q(r)}{P(r)})$. As a notational shortcut we write $\forall^\delta S \quad \Phi(S)$ instead of $\mathbf{Pr}_{S\sim D}[\Phi(S)] \leq 1 - \delta$ to express that Φ holds for all but a fraction δ of the cases. With this we can state the PAC-Bayesian theorem:

Theorem 1 (PAC-Bayesian, (McAllester, 1999)). *Let $y_i \in \mathcal{Y}$ be a class label, let P be a distribution over the space of y_i-labeled rule sets R_i, let $\delta > 0$, and define:*

$$B(Q, P, m, \delta) := \hat{l}(Q, S) + \sqrt{\frac{D(Q\|P) + \ln\frac{1}{\delta} + \ln m + 2}{2m - 1}}$$

Then, where Q ranges over all distributions on R_i:

$$\forall^\delta S \quad \forall Q \quad l(Q) \leq B(Q, P, m, \delta)$$

This theorem can be used to upper-bound the expected error of an ensemble classifier in the following way: the user provides a prior distribution P_i on the rule set space R_i. If she has some information about which rule sets are most likely to resemble the "true" target distribution D, she can use this information by selecting a P_i that assigns high probabilities to those rule sets. If she does not have any such information, she can simply select an uninformative, flat prior. In the next step she draws a sample S of size m from D. It is now the task of the learning algorithm to select a distribution Q_i on R_i for each class label y_i. The algorithm aims at finding Q_i that minimize the right hand side of the inequality and thus provides a tight upper bound on the expected error of rule sets drawn according to Q_i.

Using those Q_i and the PAC-Bayesian theorem one can construct a voting ensemble of weighted rule sets, and bound its generalization error. Let $\bar{Q} := (Q_i, \ldots, Q_p)$. Then:

$$c(Q_i, x) := \mathop{\mathbf{E}}_{r_i\sim Q_i}[r_i(x)] \tag{3}$$

$$c_V(\bar{Q}, x) := \mathop{\mathrm{argmax}}_{y_i\in\mathcal{Y}} c(Q_i, x) \tag{4}$$

$c(Q, x)$ is the *score* of Q on x and $c_V(\bar{Q}, x)$ denotes the *voting classifier* for $\bar{Q}$. The expected error of the voting classifier is thus $l_V(\bar{Q}) := \mathbf{E}_{(x,y)\sim D}[\mathrm{I}(c_V(\bar{Q}, x) \neq y)]$. The following theorem bounds the expected error of the voting classifier for two-class problems, i.e. $\bar{Q} = (Q_1, Q_2)$. Bounds for multi-class problems can be derived in a similar fashion.

Theorem 2. *For $i \in \{1, 2\}$ let $y_i \in \mathcal{Y}$ be class labels, P_i be distributions over the spaces of y_i-labeled rule sets R_i, let $\delta > 0$, and $\bar{Q} = (Q_1, Q_2)$ be as above. Then:*

$$\forall^\delta S \quad \forall Q \quad l_V(\bar{Q}) \leq B(Q_1, P_1, m, \delta) + B(Q_2, P_2, m, \delta)$$

Proof. First, observe that for $\bar{Q} = (Q_1, Q_2)$:

$$l_V(\bar{Q}) = \mathop{\mathbf{Pr}}_{D}[\chi_1(y)c(Q_1, x) + \chi_2(y)c(Q_2, x) \leq 0] \tag{5}$$

Additionally,

$$\begin{aligned}
1 - 2l(Q_i) &= 1 - 2\mathop{\mathbf{E}}_{D}\big[\mathop{\mathbf{E}}_{Q_i}[\mathrm{I}(r_i(x) \neq \chi_i(y))]\big] \\
&= 1 - 2\mathop{\mathbf{E}}_{D}\big[\mathop{\mathbf{E}}_{Q_i}[\tfrac{1}{4}(r_i(x) - \chi_i(y))^2]\big] \qquad (6)\\
&= 1 - \frac{1}{2}\mathop{\mathbf{E}}_{D}\big[\mathop{\mathbf{E}}_{Q_i}[r_i(x)^2 - 2r_i(x)\chi_i(y) + \chi_i(y)^2]\big] \\
&= 1 - \frac{1}{2}\Big(1 - 2\mathop{\mathbf{E}}_{D}\big[\mathop{\mathbf{E}}_{Q_i}[r_i(x)\chi_i(y)]\big] + 1\Big) \qquad (7)\\
&= \mathop{\mathbf{E}}_{D}[\chi_i(y)\, c(Q_i, x)]
\end{aligned}$$

(6) uses the fact that $\mathrm{I}(a \neq b) = \frac{1}{4}(a - b)^2$ for $a, b \in \{-1, +1\}$, (7) uses $a^2 = 1$ for $a \in \{-1, +1\}$. It follows from theorem 1:

$$\begin{aligned}
\forall^\delta S: \quad \mathop{\mathbf{E}}_{D}[\chi_i(y)c(Q_i, x)] &= 1 - 2l(Q_i) \qquad (8)\\
&\geq 1 - 2B(Q_i, P_i, m, \delta)
\end{aligned}$$

Now, let $C := 2 - \chi_1(y)c(Q_1,x) - \chi_2(y)c(Q_2,x)$ be a random variable. Since $C \geq 0$ for all x, y, Q_1, Q_2, we can apply Markov's inequality:

$$\forall \varepsilon > 0 : \Pr_D \left[C \geq \varepsilon \mathop{\mathbf{E}}_D [C] \right] \leq \frac{1}{\varepsilon} \tag{9}$$

By definition of C,

$$\forall \varepsilon > 0 : \Pr_D \left[\chi_1(y)c(Q_1,x) + \chi_2(y)c(Q_2,x) \leq \right.$$
$$\left. 2 - 2\varepsilon + \varepsilon \mathbf{E}[\chi_1(y)c(Q_1,x)] + \varepsilon \mathbf{E}[\chi_2(y)c(Q_2,x)] \right] \leq \frac{1}{\varepsilon}$$

and because of (8)

$$\forall \varepsilon > 0 \quad \forall^\delta S : \Pr_D \left[\chi_1(y)c(Q_1,x) + \chi_2(y)c(Q_2,x) \leq \right.$$
$$\left. 2 - 2\varepsilon(B(Q_1,P_1,m,\delta) + B(Q_2,P_2,m,\delta)) \right] \leq \frac{1}{\varepsilon} \tag{10}$$

The theorem follows from (5) by setting

$$\varepsilon = \frac{1}{B(Q_1,P_1,m,\delta) + B(Q_2,P_2,m,\delta)}.$$

□

In order to keep this bound as tight as possible, one would like to find Q_i that have low empirical error on the sample S and that differ from the P_i only to a small degree. While this may be possible theoretically, such a "PAC-Bayesian-optimal" algorithm would require calculating the empirical error of all possible concepts in the underlying concept class. Of course, this is not practical for our purposes, because the space of rule sets is way too large to be evaluated exhaustively.

We try to keep the computational costs within a reasonable range by taking two measures: first, we limit the maximum size of the rule sets to be considered. This is necessary anyway, because rule sets of arbitrary size can represent all possible dichotomies. If a flat prior is used and we choose Q only depending on the training set, this is effectively bias-free learning, which is known to be equivalent to rote learning. Thus, we restrict our concept space to the set of all rule sets with at most k_{max} rules per rule set. This bias towards short hypothesis is motivated by the principle of William of Ockham and – in one form or the other – included in virtually any existing rule learning algorithm. Second, instead of considering all possible rule sets for a Q_i, we sample a small number of rule sets from R_i, and set the probability measure Q_i to zero for all rule sets outside the sample. In this way, we have to deal with only a rather small number of concepts; the calculation of $c(Q_i,x)$ involves only the summation over the few rule sets with non-zero $Q_i(r_i)$ and the task of predicting and estimating the expected error becomes feasible.

Unfortunately, it is a bad idea to sample uniformly or according to P_i from R_i, because the bound depends on the empirical error $\hat{l}$ on the training set. Thus, if the sampled rule sets have a high error on the training set, the bound will be loose. It is therefore essential that we select rule sets with low empirical error. As stated above, we employ an SLS algorithm that finds DNFs with low empirical error in a randomized fashion to achieve this goal.

5.2. Improving the Bound Through Abstaining

Most rule learning systems are designed to assign a class to any instance that was input for classification. In practice, though, it is often the case that some instances clearly belong to one class, while other instances are just in between two classes or particularly susceptible to noise. A classifier that abstains from classification for instances of the latter kind might feature a much higher predictive accuracy, because it avoids errors on uncertain predictions. Sometimes, abstaining can give important hints to the user, e.g. about the existence of a previously unknown class label.

These considerations lead to a different approach for getting tighter bounds: allowing the classifier to abstain from a classification for uncertain instances. In our case we can assess the deviation of the votes in an ensemble as a measure of how certain the corresponding prediction is. If all rule sets in an ensemble vote for the same class label, the classification is quite certain. If, on the other hand, the weight of the rule sets that vote for y_i differs only by a small margin from the weight of the rule sets that vote for a different y_j, the classification can be regarded as uncertain. Thus, it might make sense to abstain from a classification, if the absolute value of the margin is lower than a certain threshold $\theta > 0$. For the two-class setting, the following definition gives the *abstaining voting classifier* c_V^θ. Again, the concept can be easily extended to the multi-class setting.

$$c_V^\theta(\bar{Q},x) := \begin{cases} y_1 & \text{if } c(Q_1,x) - c(Q_2,x) \geq \theta \\ 0 & \text{if } -\theta < c(Q_1,x) - c(Q_2,x) < \theta \\ y_2 & \text{if } c(Q_1,x) - c(Q_2,x) \leq -\theta \end{cases}$$

The expected error of this abstaining classifier is

$$\begin{aligned} l_B^\theta(\bar{Q}) &:= \mathop{\mathbf{E}}_{(x,y)\sim D} [\mathrm{I}(c_V^\theta(\bar{Q},x) \neq 0 \wedge c_V^\theta(\bar{Q},x) \neq y)] \\ &= \Pr_{(x,y)\sim D} [\chi_1(y)c(Q_1,x) + \chi_2(y)c(Q_2,x) \leq -\theta]. \end{aligned} \tag{11}$$

The following adaption of theorem 2 improves the PAC-Bayesian bound for an ensemble $\bar{Q}$, if the abstaining voting classifier is used instead of the voting classifier.

Theorem 3. *Let the y_1, y_2, P_1, P_2, Q_1, Q_2, and δ be as*

above, let $\theta > 0$. Then:

$$\forall^{\delta} S \quad \forall \bar{Q} \quad l_V^{\theta}(\bar{Q}) \leq \frac{B(Q_1, P_1, m, \delta) + B(Q_2, P_2, m, \delta)}{1 + \frac{1}{2}\theta}$$

Proof. The result follows from (11) and by setting

$$\varepsilon = \frac{1 + \frac{1}{2}\theta}{B(Q_1, P_1, m, \delta) + B(Q_2, P_2, m, \delta)} \tag{12}$$

in (10). □

6. Experiments

In this section we describe an empirical evaluation of the presented rule learning algorithm. The goal of the first experiment is to investigate how the presented algorithm compares to modern rule learning algorithms. To get results on learning problems with varying characteristics, we select 34 data sets from the UCI repository (Blake & Merz, 1998). Since the presented rule learning algorithm works only on nominal attributes, we discretize continuous attributes using a frequency-based discretization with ten intervals. If a data set contains unknown values, we simply add a new value "unknown" to the corresponding domains, so that unknown values are treated just like any other value.

To estimate the predictive accuracy of the algorithms we averaged over ten runs of tenfold cross-validation. The presented algorithm was set up to build rule sets with up to eight rules per rule set, and we set the o parameter to twenty rule sets per level. To calculate the $Q(r)$ probabilities, we chose the "white noise" model described in section 4 with the noise parameter α set to 0.9. The SLS algorithm was set up to search for 5000 iterations, with p_{g1}, p_{g2}, p_S set to the default values of 0.1, 0.2, and 0.1, respectively. We compare the results for the presented algorithm with the results of a support vector machine with RBF kernel and two state-of-the art rule learning systems. PART (Frank & Witten, 1998) is a separate-and-conquer-based rule learning algorithm, that avoids over pruning by obtaining rules from partial decision trees. JRIP (an implemention of Cohen's RIPPER (Cohen, 1995) in the WEKA workbench (Witten & Frank, 1999)) combines separate-and-conquer with incremental reduced error pruning and an iterated post-processing optimization step. To include ensemble-based approaches, we estimated the predictive accuracy of twentyfold-bagged versions of the two algorithms. The results are given in table 1. Table 2 shows how different methods compare to each other. Each entry indicates the number of data sets for which the method associated with its row is significantly more accurate than the method associated with its column according to a paired two-sided t-test on a 1% significance level over the runs. As can be seen, the presented algorithm performs favorably. It clearly outperforms SVM, PART, JRIP, and Bagged JRIP, and is slightly worse than Bagged PART.

				Bagged	Bagged	Rule
	SVM	PART	JRIP	PART	JRIP	Ens.
SVM		16	16	10	11	9
PART	18		16	2	5	7
JRIP	14	9		0	1	4
Bagged PART	22	28	27		14	14
Bagged JRIP	19	22	26	6		7
Rule Learn	22	23	24	11	16	

Table 2. Results: each number identifies the number of data sets, on which the method in the row significantly outperforms the method in the column.

	Rule Ens.		Consistent		Simple	
Data Set	Bound	CV	Bound	CV	Bound	CV
breast-w	30.6	3.4	43.6	4.8	**15.8**	**2.3**
bupa	**54.3**	**30.7**	100.0	38.0	84.6	30.7
credit-a	**57.7**	**13.9**	100.0	39.1	87.6	29.9
diabetes	**64.1**	**26.1**	99.9	29.9	78.6	26.8
glassg2	76.8	26.8	91.4	**22.1**	**63.8**	**22.1**
haberman	**68.0**	**29.3**	99.7	38.1	78.2	31.6
vote	**36.5**	**4.1**	71.2	11.5	44.2	11.5

Table 3. Results: the prediction error in percent as estimated by ten runs of tenfold cross validation and the size of the bound for the rule set ensemble algorithm, the consistent Set Covering Machine, and the simple Set Covering Machine.

For the second experiment, the main goal is to investigate the gap between the PAC-Bayesian bound and the true error as estimated by tenfold cross-validation and to compare our results with related approaches. Multi-class problems require the application of the union bound on the PAC-Bayesian bounds for the p ensembles, so the resulting bound is rather loose. We therefore focus on two-class problems. We apply the presented algorithm with the same parameters as above and a flat prior P to a selection of two-class problems taken from the UCI repository (Blake & Merz, 1998). To the best of our knowledge, there are no comparable results on theoretical bounds for rule learning systems in the literature. The closest approaches in the literature are SLIPPER (Cohen & Singer, 1999), LRI (Weiss & Indurkhya, 2000), and the Set Covering Machine (Marchand & Shawe-Taylor, 2001; Sokolova et al., 2003; Marchand et al., 2003). SLIPPER and LRI are rule learning algorithms based on ensembles of individual rules instead of rule sets. Since they employ voting schemes, they are amenable to theoretical analysis and also would be able to abstain from predictions. However, standard approaches to bounding the error applied to SLIPPER and LRI give rather loose bounds. Like a rule learning system, the Set Cover-

Data Set	SVM	PART	JRIP	Bagged PART	Bagged JRIP	Rule Ens.
anneal	92.9	97.2	98.0 ± 0.4	98.2 ± 0.2	98.4 ± 0.2	98.3 ± 0.1
audiology	43.8	78.8	72.8 ± 1.6	82.6 ± 0.6	77.6 ± 1.3	81.5 ± 0.9
autos	62.0	70.7	74.2 ± 2.1	82.2 ± 0.9	82.3 ± 1.3	83.4 ± 0.4
balance-scale	90.7	77.3	71.7 ± 0.9	85.1 ± 0.6	80.9 ± 1.2	83.6 ± 0.5
breast-cancer	70.3	71.0	71.8 ± 1.4	72.8 ± 1.7	74.2 ± 0.9	73.8 ± 0.5
breast-w	97.3	94.3	94.1 ± 0.5	95.3 ± 0.2	94.8 ± 0.3	96.6 ± 0.2
bupa	58.0	58.3	62.1 ± 0.9	60.9 ± 1.8	59.7 ± 1.2	69.3 ± 0.9
colic	85.9	83.4	84.3 ± 0.4	85.1 ± 0.5	85.2 ± 0.4	82.9 ± 0.5
credit-a	86.2	85.9	85.4 ± 0.4	87.0 ± 0.5	85.7 ± 0.5	86.1 ± 0.6
credit-g	71.3	70.7	69.6 ± 0.7	74.7 ± 0.6	73.1 ± 0.6	73.6 ± 0.5
diabetes	72.4	73.4	71.5 ± 0.4	73.7 ± 0.7	72.5 ± 0.6	73.9 ± 0.9
glass	54.2	52.3	64.2 ± 2.5	65.2 ± 2.0	70.2 ± 1.8	73.2 ± 1.1
haberman	73.5	73.5	71.4 ± 0.7	70.9 ± 0.7	72.1 ± 0.5	70.7 ± 0.4
heart-c	84.8	80.9	78.7 ± 1.9	83.0 ± 1.3	81.1 ± 0.9	78.5 ± 0.7
heart-h	83.3	78.9	79.4 ± 1.5	81.3 ± 1.2	81.0 ± 0.6	79.8 ± 0.6
heart-statlog	84.1	78.5	77.6 ± 1.3	81.2 ± 0.8	81.3 ± 0.8	77.1 ± 1.2
hypothyroid	96.8	97.9	97.9 ± 0.1	98.2 ± 0.1	98.1 ± 0.1	98.1 ± 0.1
ionosphere	90.3	88.0	90.8 ± 0.9	90.4 ± 0.4	91.9 ± 0.5	91.8 ± 0.3
iris	90.7	92.0	88.1 ± 1.5	92.6 ± 0.6	90.9 ± 1.3	91.1 ± 0.3
kr-vs-kp	91.4	99.1	99.2 ± 0.0	99.4 ± 0.1	99.4 ± 0.1	97.8 ± 0.1
labor	70.2	87.7	76.7 ± 2.8	84.4 ± 2.0	83.3 ± 1.2	93.3 ± 1.3
lymph	80.4	79.1	78.6 ± 1.9	85.3 ± 1.6	80.1 ± 1.2	84.9 ± 0.7
mushroom	99.9	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
primary-tumor	24.8	40.7	38.7 ± 1.0	45.3 ± 1.1	42.0 ± 0.8	43.6 ± 0.6
segment	94.4	94.2	91.8 ± 1.0	95.7 ± 0.2	96.4 ± 0.2	97.2 ± 0.1
sick	97.6	98.1	97.6 ± 0.1	98.3 ± 0.1	97.7 ± 0.0	98.1 ± 0.0
sonar	76.9	62.0	64.2 ± 3.3	72.1 ± 2.1	70.4 ± 1.2	76.2 ± 1.4
soybean	88.4	91.8	92.2 ± 0.5	93.6 ± 0.3	93.5 ± 0.3	93.2 ± 0.2
splice	96.1	92.5	94.3 ± 0.4	94.9 ± 0.1	95.8 ± 0.1	93.8 ± 0.3
tic-tac-toe	76.2	94.5	97.5 ± 0.4	99.7 ± 0.2	98.2 ± 0.1	99.2 ± 0.1
vehicle	68.4	67.0	58.8 ± 1.0	69.9 ± 1.3	70.0 ± 0.6	71.8 ± 0.7
vote	95.2	95.9	95.3 ± 0.3	96.0 ± 0.3	95.8 ± 0.2	95.9 ± 0.1
waveform-5000	84.6	73.7	72.6 ± 0.7	80.3 ± 0.3	80.7 ± 0.3	82.0 ± 0.3
zoo	73.3	92.1	87.7 ± 0.8	92.7 ± 0.9	90.7 ± 0.8	95.0 ± 0.9

Table 1. Results: percentage of correct classifications, together with standard deviation.

ing Machine uses disjunctions[1] of boolean-valued features as concepts. However, unlike rule learners, it disjunctively joins data-dependent features such as generalized balls and half-spaces instead of conjunctions of literals. Marchard *et al.* derive a bound based on a compression scheme, that can be compared to the PAC-Bayesian bound. They report empirical and analytical results for a whole range of parameter settings. In table 3 we reproduce the values for two particular settings: the "consistent" columns give the results for the unparameterized version of the data-dependent ball SCM, which induces only consistent classifiers. The "Simple SCM" is able to derive inconsistent classifiers, but the results are given for experiments that use only the best parameter settings among an exhaustive scan of many values for each data set. Those values are thus much more optimistic than our results, which are based on default parameter values that are fixed for all data sets. Nevertheless, the PAC-Bayesian bound is better in five out of seven cases, and the presented algorithm achieves a lower prediction error in five out of the seven cases.

We also performed preliminary experiments with abstaining ensembles of rule sets. To test the validity of this approach empirically, we estimate the prediction error of the abstaining Bayes algorithm on the Haberman data set using tenfold cross-validation. We used the same parameters as

[1]The Set Covering Machine can induce disjunctions or conjunctions. Since we are dealing with rule sets, we consider the disjunctive case only.

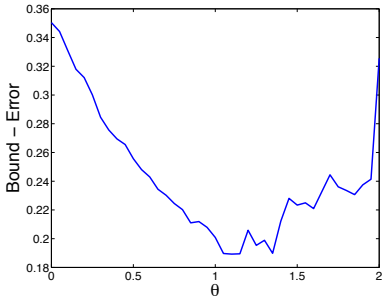

Figure 1. The difference between the bound and the estimated prediction error depending on the abstaining parameter θ.

before, but varied θ between 0 and 2. Figure 1 shows the difference between bound and estimated error. As can be seen, the relative accuracy of the bound is optimal for values of θ near 1. Thus, the bound can in fact be improved on this dataset for a certain level of abstinence.

7. Discussion

This paper showed that the empirical error of a practical rule learning algorithm can be bounded theoretically by applying McAllester's PAC-Bayesian theorem. We proved that the PAC-Bayesian bound can be further improved by allowing the model to abstain from making uncertain predictions. A preliminary experiment also indicates empirically the benefit of abstaining from uncertain predictions. Based on these theoretical considerations, we designed a new algorithm learning ensembles of rule sets. The performance of the algorithm on standard UCI datasets compares very favorably with state-of-the-art rule learning algorithms and their bagged variants. Experiments showed that the calculated bounds are reasonably close to the empirical error estimated in tenfold cross-validation. In most cases the ratio of the bound to the empirical error is smaller than for the Set Covering Machine, for which one of the tightest bounds is known. It should be noted that the bound (and with it the algorithm's predictive accuracy) can be improved in various ways: one can (1) provide an informative prior instead of a flat prior, (2) use a matching noise model instead of the white noise default, (3) increase the ensemble size, or (4) allow for abstaining. Thus, the algorithm can be easily adopted to a particular setting in the presented framework. Moreover, it should be possible to extract comprehensible consensus rules and statistics over frequently used features and feature combinations along the lines of Pfahringer *et al.*'s work on ADTrees (Pfahringer et al., 2001). Overall, we hope that this work helps narrowing the gap between theory and practice for rule learning, one of the most important practical machine learning schemes.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Cohen, W., & Singer, Y. (1999). A simple, fast, and effective rule learner. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 335–342). AAAI Press.

Cohen, W. W. (1995). Fast effective rule induction. *Proc. 12th International Conf. on Machine Learning* (pp. 115–123). Morgan Kaufmann.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proc. 15th International Conf. on Machine Learning* (pp. 144–151). Morgan Kaufmann.

Golea, M., Bartlett, P., Lee, W. S., & Mason, L. (1998). Generalization in decision trees and DNF: Does size matter? *Advances in Neural Information Processing Systems*. The MIT Press.

Hoos, H. H. (1998). *Stochastic local search - methods, models, applications*. Doctoral dissertation, TU Darmstadt.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory*. Cambridge, MA: The MIT Press.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation, 108*, 212–261.

Mansour, Y., & McAllester, D. (2000). Generalization bounds for decision trees. *Proc. 13th Annu. Conference on Comput. Learning Theory* (pp. 69–80). Morgan Kaufmann.

Marchand, M., Shah, M., Shawe-Taylor, J., & Sokolova, M. (2003). The set covering machine with data-dependent half-spaces. *Proc. 20th International Conf. on Machine Learning* (pp. 520–527). Morgan Kaufmann.

Marchand, M., & Shawe-Taylor, J. (2001). Learning with the set covering machine. *Proc. 18th International Conf. on Machine Learning* (pp. 345–352). Morgan Kaufmann.

McAllester (1999). PAC-bayesian model averaging. *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann.

Pfahringer, B., Holmes, G., & Kirkby, R. (2001). Optimizing the induction of alternating decision trees. *Proceedings of the Fith Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD2001)* (pp. 477–487).

Rückert, U., & Kramer, S. (2003). Stochastic local search in k-term DNF learning. *Proc. 20th International Conf. on Machine Learning* (pp. 648–655). AAAI Press.

Sokolova, M., Marchand, M., Japkowicz, N., & Shawe-Taylor, J. (2003). The decision list machine. *Advances in Neural Information Processing Systems 15* (pp. 921–928). MIT-Press, Cambridge, MA, USA.

Weiss, S., & Indurkhya, N. (2000). Lightweight rule induction. *Proc. 17th International Conf. on Machine Learning* (pp. 1135–1142). Morgan Kaufmann.

Witten, I. H., & Frank, E. (1999). *Data mining: Practical machine learning tools and techniques with java implementations*. Morgan Kaufmann. hardcopy.