Probability Estimation and Aggregation for Rule Learning

Jan-Nikolas Sulzmann, Johannes Fürnkranz

Abstract

Rule learning is known for its descriptive and therefore comprehensible classification models which also yield good class predictions. For different classification models, such as decision trees, a variety of techniques for obtaining good probability estimates have been proposed and evaluated. However, so far, there has been no systematic empirical study of how these techniques can be adapted to probabilistic rules and how these methods affect the probability-based rankings. In this paper we apply several basic methods for the estimation of class membership probabilities to classification rules. We also study the effect of a shrinkage technique for merging the probability estimates of rules with those of their generalizations. Finally, we compare different ways of combining probability estimates from an ensemble of rules. Our results show that for probability estimation it is beneficial to exploit the fact that rules overlap (i.e., rule averaging is preferred over rule sorting), and that individual probabilities should be combined at the level of rules and not at the level of theories.

1 Introduction

The main focus of symbolic learning algorithms such as decision tree and rule learners is to produce a comprehensible explanation for a class variable. Thus, they learn concepts in the form of crisp IF-THEN rules. On the other hand, many practical applications require a finer distinction between examples than is provided by their predicted class labels. For example, one may want to be able to provide a confidence score that estimates the certainty of a prediction, to rank the predictions according to their probability of belonging to a given class, to make a cost-sensitive prediction, or to combine multiple predictions.

All these problems can be solved straight-forwardly if we can predict a probability distribution over all classes instead of a single class value. A straight-forward approach to estimate probability distributions for classification rules is to compute the fractions of the covered examples for each class. However, this naïve approach has obvious disadvantages, such as that rules that cover only a few examples may lead to extreme probability estimates. Thus, the probability estimates need to be smoothed.

There has been quite some previous work on probability estimation from decision trees (so-called *probability-estimation trees (PETS)*). A very simple, but quite powerful technique for improving class probability estimates is the use of m-estimates, or their special case, the Laplace-estimates [Cestnik, 1990]. It has been shown that unpruned decision trees with Laplace-corrected probability estimates at the leaves produce quite reliable decision tree estimates [Provost and Domingos, 2003]. A recursive computation of the m-estimate, which uses the probability disctribution at level l as the prior probabilities for level $l + 1$, was proposed in [Ferri *et al.*, 2003]. In [Wang and Zhang, 2006], a general shrinkage approach was used, which interpolates the estimated class distribution at the leaf nodes with the estimates in interior nodes on the path from the root to the leaf.

An interesting observation is that, contrary to classification, class probability estimation for decision trees typically works better on unpruned trees than on pruned trees. The explanation for this is simply that, as all examples in a leaf receive the same probability estimate, pruned trees provide a much coarser ranking than unpruned trees. In [Hüllermeier and Vanderlooy, 2009], a simple but elegant analysis of this phenomenon was provided, which shows that replacing a leaf with a subtree can only lead to an increase in the area under the ROC curve (AUC), a commonly used measure for the ranking capabilities of an algorithm. Of course, this only holds for the AUC estimate on the training data, but it still may provide a strong indication why unpruned PETs typically also outperform pruned PETs on the test set.

Despite the amount of work on probability estimation for decision trees, there has been hardly any systematic work on probability estimation for rule learning. Despite their obvious similarility, we nevertheless argue that a separate study of probability estimates for rule learning is necessary.

A key difference is that in the case of decision tree learning, probability estimates will not change the prediction for an example, because the predicted class only depends on the probabilities of a single leaf of the tree, and such local probability estimates are typically monotone in the sense that they all maintain the majority class as the class with the maximum probability. In the case of rule learning, on the other hand, each example may be classified by multiple rules, which may possibly predict different classes. As many tie breaking strategies depend on the class probabilities, a local change in the class probability of a single rule may change the global prediction of the rule-based classifier, even if the order of all local estimates is maintained.

Because of such non-local effects, it is not evident that the same methods that work well for decision tree learning will also work well for rule learning. Indeed, as we will see in this paper, our conclusions differ from those that

have been drawn from similar experiments in decision tree learning. For example, the above-mentioned argument that unpruned trees will lead to a better (training-set) AUC than pruned trees, does not straight-forwardly carry over to rule learning, because the replacement of a leaf with a subtree is a local operation that only affects the examples that are covered by this leaf. In rule learning, on the other hand, each example may be covered by multiple rules, so that the effect of replacing one rule with multiple, more specific rules is less predictable. Moreover, each example will be covered by some leaf in a decision tree, whereas each rule learner needs to induce a separate default rule that covers examples that are covered by no other rule.

The rest of the paper is organized as follows: In Section 2 we briefly describe the basics of probabilistic rule learning and discuss the used estimation techniques used for rule probabilities. In Section 3 we describe the rule learning algorithm that we used in our experiments, contrast two approaches for the generation of a probabilistic rule set, and describe how they are used for classification. Experimental results, which compare the different probability estimation techniques in these two scenarios, are described in Section 4. We then discuss four techniques for obtaining rule probabilities from a bagging ensemble (Section 5), and compare them experimentally in Section 6. In the end, we summarize our conclusions in Section 7.[1]

2 Rule Learning and Probability Estimation

This section is divided into two parts. The first one describes briefly the properties of conjunctive classification rules and of its extension to a probabilistic rule. In the second part we introduce the probability estimation techniques used in this paper. These techniques can be divided into basic methods, which can be used stand-alone for probability estimation, and the meta technique shrinkage, which can be combined with any of the techniques for probability estimation.

2.1 Probabilistic Rule Learning

In classification rule mining one searches for a set of rules that describes the data as accurately as possible. As there are many different generation approaches and types of generated classification rules, we do not go into detail and restrict ourselves to conjunctive rules. The *premise* of these rules consists of a conjunction of number of conditions, and in our case, the *conclusion* of the rule is a single class value. So a conjunctive classification rule r has basically the following form:

$$condition_1 \wedge \cdots \wedge condition_{|r|} \implies class \qquad (1)$$

The size of a rule $|r|$ is the number of its conditions. Each of these conditions consists of an attribute, an attribute value belonging to its domain and a comparison determined by the attribute type. For our purpose, we consider only nominal and numerical attributes. For nominal attributes, this comparison is a test of equality, whereas in the case of numerical attributes, the test is either less (or equal) or greater (or equal). If all conditions are met by an instance, the instance is covered by the rule ($r \supseteq x$) and the class value of the rule is predicted for the instance. Consequently, the rule is called a *covering rule* for this instance.

[1]Parts of this paper have previously appeared as [Sulzmann and Fürnkranz, 2009].

This in mind, we can define some statistical values of a data set which are needed for later definitions. A data set consists of $|C|$ classes and n instances from which n^c belong to the class c respectively ($n = \sum_{c=1}^{|C|} n^c$). A rule r covers n_r instances which are distributed over the classes, so that n_r^c instances belong to class c ($n_r = \sum_{c=1}^{|C|} n_r^c$).

A probabilistic rule r is an extension of a classification rule, which does not only predict a single class value, but a set of *class probabilities*, which form a probability distribution over the classes. This probability distribution estimates all probabilities that a covered instance belongs to any of the class in the data set, so we get one class probability per class. The example is then classified with the most probable class. The probability that an instance x covered by rule r belongs to c can be viewed as a conditional probability $\Pr(c|r \supseteq x)$. Thus the set of class probabilities can be noted as vector of probabilities sorted by the class ordering:

$$\vec{\Pr}(r \supseteq x) = (\Pr(c_1|r \supseteq x), \cdots, \Pr(c_{|C|}|r \supseteq x)) \qquad (2)$$

On the vector $\vec{\Pr}(r \supseteq x)$, abbreviated $\vec{\Pr}_r(x)$, we define the following maximum function

$$\max\left(\vec{\Pr}_r(x)\right) = \max_{c \in C} \Pr(c|r \supseteq x). \qquad (3)$$

On sets of class probability vectors $\bigcup_{j=1}^{k} \vec{\Pr}_i(x)$ we define the average function

$$\text{avg}\left(\bigcup_{j=1}^{k} \vec{\Pr}_j(x)\right) = \frac{1}{k} \sum_{j=0}^{k} \vec{\Pr}_j(x) \qquad (4)$$

and the multiplication

$$\text{mult}\left(\bigcup_{j=1}^{k} \vec{\Pr}_j(x)\right) = \left(\prod_{j=0}^{k} \Pr(c_1|r_j \supseteq x), \cdots, \prod_{j=0}^{k} \Pr(c_{|C|}|r_j \supseteq x)\right) \qquad (5)$$

Obviously the results of these functions are also class probability vectors.

In the next section, we discuss some approaches for estimating these class probabilities.

2.2 Basic Probability Estimation

In this subsection we will review three basic methods for probability estimation. Subsequently, in Section 2.3, we will describe a technique known as shrinkage, which is known from various application areas, and show how this technique can be adapted to probabilistic rule learning.

All of the three basic methods we employed, calculate the relation between the number of instances covered by the rule n_r and the number of instances covered by the rule but also belong to a specific class n_r^c. The differences between the methods are the minor modifications of the calculation of this relation.

The simplest approach to rule probability estimation directly estimates a class probability distribution of a rule with the fraction of examples that belong to each class.

$$\underset{\text{naïve}}{\Pr}(c|r \supseteq x) = \frac{n_r^c}{n_r} \qquad (6)$$

This naïve approach has several well-known disadvantages, most notably that rules with a low coverage may be lead to extreme probability values. For this reason, the use of

the Laplace- and m-estimates was suggested in [Cestnik, 1990].

The Laplace estimate modifies the above-mentioned relation by adding one additional instance to the counts n_r^c for each class c. Hence the number of covered instances n_r is increased by the number of classes $|C|$.

$$\Pr_{\text{Laplace}}(c|r \supseteq x) = \frac{n_r^c + 1}{n_r + |C|} \tag{7}$$

It may be viewed as a trade-off between $\Pr_{\text{naïve}}(c|r \supseteq x)$ and an *a priori* probability of $\Pr(c) = 1/|C|$ for each class. Thus, it implicitly assumes a uniform class distribution.

The m-estimate generalizes this idea by making the dependency on the prior class distribution explicit, and introducing a parameter m, which allows to trade off the influence of the *a priori* probability and$\Pr_{\text{naïve}}$.

$$\Pr_{m}(c|r \supseteq x) = \frac{n_r^c + m \cdot \Pr(c)}{n_r + m} \tag{8}$$

The m-parameter may be interpreted as a number of examples that are distributed according to the prior probability, which are added to the class frequencies n_r^c. The prior probability is typically estimated from the data using $\Pr(c) = n^c/n$ (but one could, e.g., also use the above-mentioned Laplace-correction if the class distribution is very skewed). Obviously, the Laplace-estimate is a special case of the m-estimate with $m = |C|$ and $\Pr(c) = 1/|C|$.

2.3 Shrinkage

Shrinkage is a general framework for smoothing probabilities, which has been successfully applied in various research areas.[2]Its key idea is to "shrink" probability estimates towards the estimates of its generalized rules r_k, which cover more examples. This is quite similar to the idea of the Laplace- and m-estimates, with two main differences: First, the shrinkage happens not only with respect to the prior probability (which would correspond to a rule covering all examples) but interpolates between several different generalizations, and second the weights for the trade-off are not specified *a priori* (as with the m-parameter in the m-estimate) but estimated from the data.

In general, shrinkage estimates the probability $\Pr(c|r \supseteq x)$ as follows:

$$\Pr_{Shrink}(c|r \supseteq x) = \sum_{k=0}^{|r|} w_c^k \Pr(c|r_k) \tag{9}$$

where w_c^k are weights that interpolate between the probability estimates of the generalized rules r_k. In our implementation, we use only generalizations of a rule that can be obtained by deleting a final sequence of conditions. Thus, for a rule with length $|r|$, we obtain $|r| + 1$ generalizations r_k, where r_0 is the rule covering all examples, and $r_{|r|} = r$.

The weights w_c^k can be estimated in various ways. We employ a shrinkage method proposed in [Wang and Zhang, 2006] which is intended for decision tree learning but can be straight-forwardly adapted to rule learning. The authors propose to estimate the weights w_c^k with an iterative procedure which averages the probabilities obtained by removing training examples covered by this rule. In effect, we obtain two probabilities per rule generalization and class: the removal of an example of class c leads to a decreased probability $\Pr_-(c|r_k \supseteq x)$, whereas the removal of an example of a different class results in an increased probability $\Pr_+(c|r_k \supseteq x)$. Weighting these probabilities with the relative occurrence of training examples belonging to this class we obtain a smoothed probability

$$\Pr_{Smoothed}(c|r_k \supseteq x) = \frac{n_r^c}{n_r} \cdot \Pr_-(c|r_k \supseteq x) \tag{10}$$

$$+ \frac{n_r - n_r^c}{n_r} \cdot \Pr_+(c|r_k \supseteq x) \tag{11}$$

Using these smoothed probabilities, this shrinkage method computes the weights of these nodes in linear time (linear in the number of covered instances) by normalizing the smoothed probabilities separately for each class.

$$w_c^k = \frac{\Pr_{Smoothed}(c|r_k \supseteq x)}{\sum_{i=0}^{|r|} \Pr_{Smoothed}(c|r_i \supseteq x)} \tag{12}$$

Multiplying the weights with their corresponding probability we obtain "shrinked" class probabilities for the instance.

Note that all instances which are classified by the same rule receive the same probability distribution. Therefore the probability distribution of each rule can be calculated in advance.

3 Rule Learning Algorithm

For the rule generation we employed the rule learner Ripper [Cohen, 1995], arguably one of the most accurate rule learning algorithms today. We used Ripper both in ordered and in unordered mode:

Ordered Mode: In ordered mode, Ripper learns rules for each class, where the classes are ordered according to ascending class frequencies. For learning the rules of class c_i, examples of all classes c_j with $j > i$ are used as negative examples. No rules are learned for the last and most frequent class, but a rule that implies this class is added as the default rule. At classification time, these rules are meant to be used as a decision list, i.e., the first rule that fires is used for prediction.

Unordered Mode: In unordered mode, Ripper uses a one-against-all strategy for learning a rule set, i.e., one set of rules is learned for each class c_i, using all examples of classes $c_j, j \neq i$ as negative examples. At prediction time, all rules that cover an example are considered and the rule with the maximum probability estimate is used for classifying the example. If no rule covers the example, it classified by the default rule predicting the majority class.

We used JRip, the Weka [Witten and Frank, 2005] implementation of Ripper. Contrary to William Cohen's original implementation, this re-implementation does not support the unordered mode, so we had to add a re-implementation of that mode.[3] We also added a few other minor modifications which were needed for the probability estimation, e.g. the collection of statistical counts of the sub rules.

In addition, Ripper (and JRip) can turn the incremental reduced error pruning technique [Fürnkranz and Widmer, 1994; Fürnkranz, 1997] on and off. Note, however,

[2]Shrinkage is, e.g., regularly used in statistical language processing [Chen and Goodman, 1998; Manning and Schütze, 1999]

[3]Weka supports a general one-against-all procedure that can also be combined with JRip, but we could not use this because it did not allow us to directly access the rule probabilities.

that with turned off pruning, Ripper still performs prepruning using a minimum description length heuristic [Cohen, 1995]. We use Ripper with and without pruning and in ordered and unordered mode to generate four set of rules. For each rule set, we employ several different class probability estimation techniques.

In the test phase, all covering rules are selected for a given test instance. Using this reduced rule set we determine the most probable rule. For this purpose we select the most probable class of each rule and use this class value as the prediction for the given test instance and the class probability for comparison. Ties are solved by predicting the least represented class. If no covering rules exist the class probability distribution of the default rule is used.

4 Experimental Results

4.1 Experimental Setup

We performed our experiments within the WEKA framework [Witten and Frank, 2005]. We tried each of the four configuration of Ripper (unordered/ordered and pruning/no pruning) with 5 different probability estimation techniques, Naïve (labeled as Precision), Laplace, and m-estimate with $m \in \{2, 5, 10\}$, both used as a stand-alone probability estimate (abbreviated with B) or in combination with shrinkage (abbreviated with S). As a baseline, we also included the performance of pruned or unpruned standard JRip accordingly. Our unordered implementation of JRip using Laplace stand-alone for the probability estimation is comparable to the unordered version of Ripper (Cohen, 1995), which is not implemented in JRip.

We evaluated these methods on 33 data sets of the UCI repository [Asuncion and Newman, 2007] which differ in the number of attributes (and their categories), classes and training instances. As a performance measure, we used the weighted area under the ROC curve (AUC), as used for probabilistic decision trees in [Provost and Domingos, 2003]. Its key idea is to extend the binary AUC to the multiclass case by computing a weighted average the AUCs of the one-against-all problems N_c, where each class c is paired with all other classes:

$$\mathrm{AUC}(N) = \sum_{c \in C} \frac{n_c}{|N|} \mathrm{AUC}(N_c) \tag{13}$$

For the evaluation of the results we used the Friedman test with a post-hoc Nemenyi test as proposed in [Demsar, 2006]. The significance level was set to 5% for both tests. We only discuss summarized results here, detailed result tables can be found in [Sulzmann and Fürnkranz, 2009] and [Sulzmann and Fürnkranz, 2010].

4.2 Ordered Rule Sets

In the first two test series, we investigated the ordered approach using the standard JRip approach for the rule generation, both with and without pruning. The basic probability methods were used standalone (B) or in combination with shrinkage (S).

The Friedman test showed that in both test series, the employed combinations of probability estimation techniques showed significant differences. Considering the CD chart of the first test series (Figure 1(a)), one can identify three groups of equivalent techniques. Notable is that the two best techniques, the m-Estimate used stand-alone with $m = 2$ and $m = 5$ respectively, belong only to the best group. These two are the only methods that are significantly better than the two worst methods, Precision used stand-alone and Laplace combined with shrinkage. On the other hand, the naïve approach seems to be a bad choice as both techniques employing it rank in the lower half. However our benchmark JRip is positioned in the lower third, which means that the probability estimation techniques clearly improve over the default decision list approach implemented in JRip.

Comparing the stand-alone techniques with those employing shrinkage one can see that shrinkage is outperformed by their stand-alone counterparts. Only Precision is an exception as shrinkage yields increased performance in this case. In the end shrinkage is not a good choice for this scenario.

The CD-chart for ordered rule sets with pruning (Figure 1(b)) features four groups of equivalent techniques. Notable are the best and the worst group which overlap only in two techniques, Laplace and Precision used stand-alone. The first group consists of all stand-alone methods and JRip which dominates the group strongly covering no shrinkage method. The last group consists of all shrinkage methods and the overlapping methods Laplace and Precision used stand-alone. As all stand-alone methods rank before the shrinkage methods, one can conclude that they outperform the shrinkage methods in this scenario as well. Ripper performs best in this scenario, but the difference to the stand-alone methods is not significant.

4.3 Unordered Rule Sets

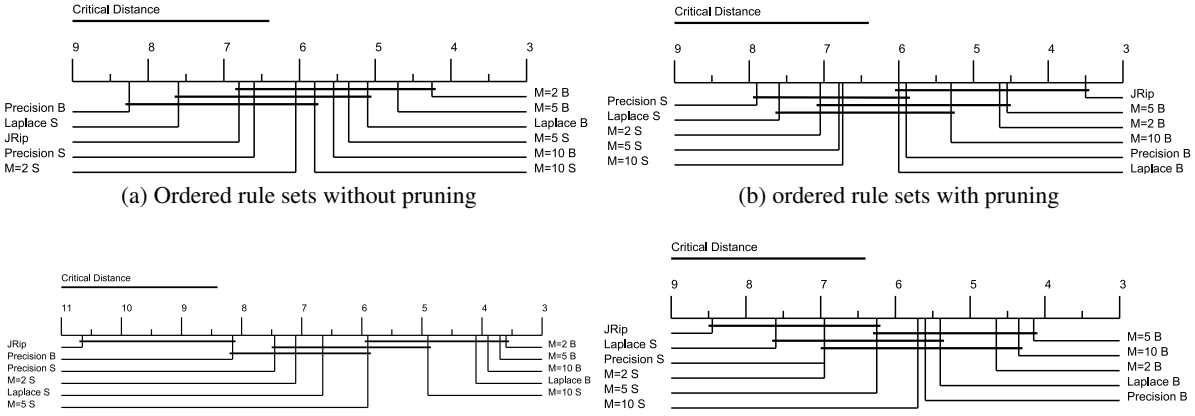
Test series three and four used the unordered approach employing the modified JRip which generates rules for each class. Analogous to the previous test series the basic methods are used as stand-alone methods or in combination with shrinkage (left and right column respectively). Test series three used no pruning while test series four did so. The results of the Friedman test showed that the techniques of test series three and test series four differ significantly.

Regarding the CD chart of test series three (Figure 1(c)), we can identify four groups of equivalent methods. The first group consists of all stand-alone techniques, except for Precision, and the m-estimates techniques combined with shrinkage and $m = 5$ and $m = 10$, respectively. Whereas the stand-alone methods dominate this group, $m = 2$ being the best representative. Apparently these methods are the best choices for this scenario. The second and third consist mostly of techniques employing shrinkage and overlap with the worst group in only one technique. However our benchmark JRip belongs to the worst group being the worst choice of this scenario. Additionally the shrinkage methods are outperformed by their stand-alone counterparts.

The CD chart of test series four (Figure 1(d)) shows similar results. Again four groups of equivalent techniques groups can be identified. The first group consists of all stand-alone methods and the m-estimates using shrinkage and $m = 5$ and $m = 10$ respectively. This group is dominated by the m-estimates used stand-alone with $m = 2$, $m = 5$ or $m = 10$. The shrinkage methods are distributed over the other groups, again occupying the lower half of the ranking. Our benchmark JRip is the worst method of this scenario.

4.4 Unpruned vs. Pruned Rule Sets

Rule pruning had mixed results, which are briefly summarized in Table 1. On the one hand, it improved the results of the ordered approach, on the other hand it worsened the results of the unordered approach. In any case, in our ex-



(a) Ordered rule sets without pruning

(b) ordered rule sets with pruning

(c) Unordered rule sets without pruning

(d) Unordered rule sets with pruning

Figure 1: Critical distance charts of ordered rule sets ((a) and (b)) and unordered rule sets ((c) and (d))

periments, contrary to previous results on PETs, rule pruning was not always a bad choice. The explanation for this result is that in rule learning, contrary to decision tree learning, new examples are not necessarily covered by one of the learned rules. The more specific rules become, the higher is the chance that new examples are not covered by any of the rules and have to be classified with a default rule. As these examples will all get the same default probability, this is a bad strategy for probability estimation. Note, however, that JRip without pruning, as used in our experiments, still performs an MDL-based form of pre-pruning. We have not yet tested a rule learner that performs no pruning at all, but, because of the above deliberations, we do not expect that this would change the results with respect to pruning.

5 Probability Estimates from Rule Ensembles

Instead of trying to improve the probability estimates for each individual leaf, one can also resort to averaging multiple estimates, thereby reducing the variance of the resulting probability estimates. For example, a technique based on bagging multiple unpruned decision trees was used in [Domingos, 1999] to obtain improved probability estimates, which were subsequently used for cost-sensitive classification. An adaptation of this technique to rule learning again has the effect that an example may be covered by a varying number of rules, whereas in decision tree learning, each example will be covered by exactly s leafs (where s is the number of trees learned).

For investigating the performance ensemble-based probability estimates, we combined our rule learner with bagging [Breiman, 1996]. We generated s bootstrap samples $S_1, \cdots, S_s$ by repeatedly drawing n instances with replacement. The rule algorithm described in Section 3 was applied to each sampled data set s_i obtaining s classifiers $C_1, \cdots, C_s$ and their corresponding rule sets $R_1, \cdots, R_s$. The probability estimation for each is computed using the previously introduced basic estimation methods.

For prediction, we determine the covering rules of each sampled rule set R_i for a given example x

$$R_i(x) = \{r \in R_i | r \supseteq x\} \tag{14}$$

Let $Cov_i(x)$ denote the set of all class probability distributions that originate from a rule in the set of covering rules

$$Cov_i(x) = \left\{ \vec{\Pr}_r(x) | r \in R_i(x) \right\}. \tag{15}$$

From this set of class probability distributions of the covering rules, we try to estimate a class probability distribution for the given example x. For this purpose we have to decode the probability estimations of these covering rules $\vec{\Pr}_r(x)$ into a single normalized global class probability distribution $\vec{\Pr}_{global}(x)$.

For our approach we considered four decoding methods. The first three methods have in common that they average the class probability distribution of (some of) the covering rules.

Best Rule: Only the most confident covering rule $\vec{\Pr}_i(x)$ of each covering rule set $R_i(x)$ is determined.

$$\vec{\Pr}_i(x) = \underset{\vec{\Pr}(x) \in Cov_i}{\arg\max} \left(\vec{\Pr}(x) \right) \tag{16}$$

Afterwards the class probability distributions of these rules are averaged and the result is normalized:

$$\vec{\Pr}_{global}(x) = \frac{\text{avg}\left(\vec{\Pr}_1, \cdots, \vec{\Pr}_s\right)}{\left\| \text{avg}\left(\vec{\Pr}_1, \cdots, \vec{\Pr}_s\right) \right\|_1}. \tag{17}$$

Macro Averaging: All covering rules are determined and their class probability distributions are macro-averaged in two steps. First the class probability distributions $Cov_i(x)$ of each covering rule set $R_i(x)$ are averaged and normalized:

$$\vec{\Pr}_i = \frac{\text{avg}\left(Cov_i(x)\right)}{\left\| \text{avg}\left(Cov_i(x)\right) \right\|_1}. \tag{18}$$

These local class probabilities are then averaged as above (17).

Micro Averaging: All covering rules are determined and their class probability distributions are micro-averaged. Essentially, this means that all learned rules are pooled, and the average is formed over the resulting set set of rules:

$$\vec{\Pr}_{global}(x) = \frac{\text{avg}\left(Cov_1(x) \cup \cdots \cup Cov_s(x)\right)}{\left\| \text{avg}\left(Cov_1(x) \cup \cdots \cup Cov_s(x)\right) \right\|_1} \tag{19}$$

Bayesian Decoding: All covering rules are pooled as above, but their class probability distributions are multiplied with each other and with the vector of the a priori class probabilities

$$\vec{\Pr}_{prior} = \left(\Pr(c_1), \cdots, \Pr(c_{|C|})\right).$$

Table 1: Unpruned vs. pruned rule sets: Win/Loss for ordered (top) and unordered (bottom) rule sets

	JRip	Precision		Laplace		M 2		M 5		M 10	
Win	26	23	19	20	19	18	20	19	20	19	20
Loss	7	10	14	13	14	15	13	14	13	14	13
Win	26	21	9	8	8	8	8	8	8	8	6
Loss	7	12	24	25	25	25	25	25	25	25	27

Thus $\vec{\Pr}_{global}(x)$ is calculated as follows

$$\vec{\Pr}_{global}(x) = \frac{\text{mult}\left(\bigcup_{i=1}^{s} Cov_i(x) \cup \left\{\vec{\Pr}_{prior}\right\}\right)}{\left\|\text{mult}\left(\bigcup_{i=1}^{s} Cov_i(x) \cup \left\{\vec{\Pr}_{prior}\right\}\right)\right\|_1} \tag{20}$$

In all cases, the resulting class probability distribution $\vec{\Pr}_{global}(x)$ is used for the prediction. For this purpose the most probable class according to $\vec{\Pr}_{global}(x)$ is selected and this class value is used as the prediction for the given test instance x. Ties are solved by predicting the least represented class. If no covering rules ($Cov_i(x) = \emptyset$) exist for a sampled rule set R_i the class probability distribution of the default rule is used accordingly.

6 Results on Ensemble-Based Probability Estimates

6.1 Experimental Setup

The above methods were integrated into the framework described in Section 4.1. For sampling, we employed the unsupervised random sampling of WEKA for the generation of the bootstrap samples and applied the Bagging implementation of WEKA to JRip. In accordance with the results obtained above, we only used unordered rule sets for these experiments because these produce better probability estimates than ordered rule sets. Furthermore we know that the unpruned rule sets perform better for the Laplace- and m-estimates than pruned rule sets if the rule sets are generated by the unordered JRip. For Precision the opposite is true. So we employed these basic probability estimation techniques on either unpruned or pruned rule sets according to these observations. As the employed shrinkage method worsened the probability estimation we abstained from using shrinkage in these experiments. All previously mentioned decoding methods - Best rule, Macro and Micro Averaging, and Bayesian Decoding - were employed. So we computed all combinations of the basic probability estimation and decoding methods on a different number of bootstrap samples - 10, 20, 50 and 100 samples accordingly. These methods were compared to the default configuration of JRip (pruned, ordered rule sets) and to its bagged version (with or without pruning) using the same bootstrap samples.

All methods were evaluated on the 33 previously used data sets of the UCI repository. As a performance measure, we used the weighted area under the ROC curve (AUC) also. For our comparison we calculated the average weighted AUC over all data sets for all combinations (combining a probability estimation technique and a decoding method to a given number of boostrap samples), the summarized results are depicted in Figure 2. Detailed results can be found in [Sulzmann and Fürnkranz, 2010].

6.2 Comparison to JRip

In our experiments we compared JRip and its bagged versions to the basic probability estimation techniques employing the four decoding methods. Figure 2 shows the results of unpruned bagged JRip and the basic probability estimation techniques. We omitted to depict the results of JRip and the pruned bagged JRip because they both had a worse performance than the unpruned bagged version which has the worst performance of the depicted methods. The weighted AUC of their best representative, unpruned bagged JRip, was always at least two absolute percentage points lower than the weighted AUC of the basic probability estimation techniques for all decoding methods and all numbers of samples. So we can conclude that the performance of the basic probability estimation techniques improve over the default probability estimation integrated in JRip. As this observation was also made in the basic rule learning experiments, we see our approach reconfirmed.

6.3 Comparison of the base probability estimation techniques

Each of the results of Figure 2 is the average performance over several different base probability estimation techniques: Precision applied to pruned rule sets and the Laplace and the m-estimate ($m \in \{2, 5, 10\}$) applied to unpruned rule sets. The applied decoding methods seem to have only a small impact on the ranking (according the weighted AUC) of the basic probability estimation techniques. Especially for a higher number of bootstrap samples, 50 and 100, these rankings are always the same - Precision having the best performance followed by the m-estimate using $m = 2$, $m = 5$, or $m = 10$ and the Laplace estimate in this order. For the smaller numbers of samples, 10 and 20, the ranking is a little bit more dynamic. Although the best two methods, m-estimate with $m = 2$ and $m = 5$ (in this order), and the worst method, Laplace estimate, are always the same, the methods in the center, Precision and the m-estimate using $m = 10$, switch places dependent on the decoding methods. So we can conclude that the m-estimate using $m = 2$ is the best choice for a low number of samples just as Precision is the best choice for a higher number of samples. For all probability estimation techniques holds that an increase in the number of samples leads to an improvement in the weighted AUC but the gain is a bit lower than the gain of the bagged JRip.

6.4 Comparison of the decoding methods

In this section, we want to compare the four employed decoding methods: Best Rule, Macro and Micro Averaging, and Bayesian Decoding. For this purpose we calculated the average weighted AUC over all data sets for all combinations obtained by combining a probability estimation technique, a decoding method and a number of bootstrap samples. Afterwards, we determined on this data the average rank of each decoding method (Table 2) and used this information for a Friedman test with a significance of 5%. According to this test, the decoding methods differ significantly, so we applied a post-hoc Nemenyi test which is depicted in a CD chart (Figure 3).

Table 2: Decoding Methods: Count of each rank and the average rank

Decoding Method	Ranked 1st	2nd	3rd	4th	Average Rank
Best Rule	0	0	4	16	3.80
Macro Averaging	0	17	3	0	2.15
Micro Averaging	20	0	0	0	1.00
Bayesian Decoding	0	3	13	4	3.05

Obviously, Micro Averaging is the best decoding methods in our experiments as it always placed first according to the average weighted AUC. This observation is reconfirmed by the CD chart since the Micro Averaging is the only member of the best group of methods. Macro Averaging and Bayesian Decoding do not differ significantly being both in the second best group of decoding methods. Nevertheless Bayesian Decoding is also in the worst group together with Best Rule which is the worst choice in our experiments.

The observed ranking of the decoding methods can be attributed to the individual exploitation of the covering rules. As the decoding method Best Rule only uses the best rule of each bootstrap sample, a great deal of evidence has no influence on the probability estimation. Thus, it is not surprising that Best Rule ranks behind the methods that make better use of the ensemble.

The methods Macro and Micro Averaging both average the probability distributions of a number of covering rules but their averaging approaches differ. So the influence of a high evidence for a class in a sampled rule set is also different for the two methods. For Micro Averaging the aforementioned evidence has a direct effect on resulting probability distribution as all covering rules have the same weight in its calculation. As our rule sets only have a low redundancy, this effect is desirable. For Macro Averaging a high evidence in a sampled rule set influences only the probability distribution of this bootstrap sample. So the number of covering rules has no effect on the global probability distribution. As Macro Averaging partially discards the available information Micro Averaging should perform better than Macro Averaging as observed in our experiments.

Bayesian Decoding uses all the information contained in the covering rules as Micro Averaging does. These two methods differ only the way how they combine the information of the covering rule, averaging or multiplying their probability distributions. The multiplication used in the Bayesian approach has a tendency to prefer a number of medium probabilities to a balanced number of low and high probabilities. This bias has a negative effect on the calculation of the global probability distribution. Averaging is more desirable as high probabilities have a greater impact on its calculation.

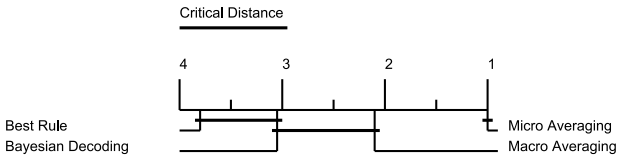


Figure 3: CD chart of the decoding methods

7 Conclusions

The most important result of our study is that probability estimation is clearly an important part of a good rule learning algorithm. The probabilities of rules induced by JRip can be improved considerably by simple estimation techniques. In unordered mode, where one rule is generated for each class, JRip is outperformed in every scenario. On the other hand, in the ordered setting, which essentially learns decision lists by learning subsequent rules in the context of previous rules, the results were less convincing, giving a clear indication that the unordered rule induction mode should be preferred when a probabilistic classfication is desirable.

Among the tested probability estimation techniques, the m-estimate typically outperformed the other methods in our version of the JRip algorithm. The superiority of the m-estimate was not sensitive to the choice of its parameter. When combined with an ensemble-based approach, the m-estimate maintained its superiority for smaller number of bootstrap samples, but typically lower value of m performed better. For higher numbers of bootstrap samples, precision, which corresponds to a value of $m = 0$, outperformed the other methods. Thus, it seems to be the case that the use of the m-estimate primarily helps to reduce the variance of the probability estimates.

The employed shrinkage method did, in general, not improve the simple estimation techniques. It remains to be seen whether alternative ways of setting the weights could yield superior results. Rule pruning did not produce the bad results that are known from ranking with pruned decision trees, presumably because unpruned, overly specific rules will increase the number of uncovered examples, which in turn leads to bad ranking of these examples.

Ensemble-based probability estimation based on a bagging approach further improved the probability estimates. The improvement increases with the number of bootstrap samples. In every case, the probabilistic approach outperformed Bagged JRip using the same number of bootstrap samples. Amongst the employed decoding methods, Micro Averaging of the probability distributions of all covering rules was without exceptions superior to the other methods, indicating that the predictions of rule-based ensembles should be combined at the level of individual rules and not at the level of theories.

Acknowledgements

This research was supported by the *German Science Foundation (DFG)* under grant FU 580/2.

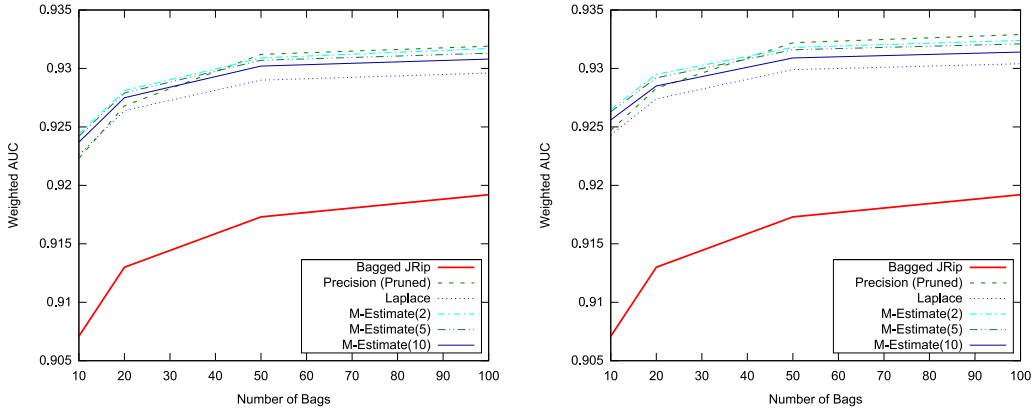
References

A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Bojan Cestnik. Estimating probabilities: A crucial task in Machine Learning. In L. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, pages 147–150, Stockholm, Sweden, 1990. Pitman.

Stanley F. Chen and Joshua T. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, 1998.



(a) Best Rule

(b) Macro Averaging

(c) Micro Averaging

(d) Bayesian Decoding

Figure 2: Average Weighted AUC of the employed decoding methods on unpruned rule sets (where stated the pruned rule set is used)

William W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 155–164, San Diego, CA, 1999. ACM.

César Ferri, Peter A. Flach, and José Hernández-Orallo. Improving the AUC of probabilistic estimation trees. In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *Proceedings of the 14th European Conference on Machine Learning*, pages 121–132, Cavtat-Dubrovnik, Croatia, 2003. Springer.

Johannes Fürnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In William W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77, New Brunswick, NJ, 1994. Morgan Kaufmann.

Johannes Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.

Eyke Hüllermeier and Stijn Vanderlooy. Why fuzzy decision trees are good rankers. *IEEE Transactions on Fuzzy Systems*, 17(6):1233–1244, 2009.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.

Foster J. Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

Jan-Nikolas Sulzmann and Johannes Fürnkranz. An empirical comparison of probability estimation techniques for probabilistic rules. In João Gama, Vítor Santos Costa, A. Jorge, and Pavel B. Brazdil, editors, *Proceedings of the 12th International Conference on Discovery Science (DS-09)*, pages 317–331. Springer-Verlag, 2009. Winner of Best Student Paper Award.

Jan-Nikolas Sulzmann and Johannes Fürnkranz. Probability estimation and aggregation for rule learning. Technical Report TUD-KE-2010-03, TU Darmstadt, Knowledge Engineering Group, 2010.

Bin Wang and Harry Zhang. Improving the ranking performance of decision trees. In Johannes Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the 17th European Conference on Machine Learning (ECML-06)*, pages 461–472, Berlin, Germany, 2006. Springer-Verlag.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005.