Estimating Prediction Certainty in Decision Trees

Eduardo P. Costa[a], Sicco Verwer[b], and Hendrik Blockeel[a,c]

[a] Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium
[b] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands
[c] Leiden Institute of Advanced Computer Science, Universiteit Leiden, Leiden, The Netherlands
eduardo.costa@cs.kuleuven.be,siccoverwer@gmail.com,hendrik.blockeel@cs.kuleuven.be

Abstract. Decision trees estimate prediction certainty using the class distribution in the leaf responsible for the prediction. We introduce an alternative method that yields better estimates. For each instance to be predicted, our method inserts the instance to be classified in the training set with one of the possible labels for the target attribute; this procedure is repeated for each one of the labels. Then, by comparing the outcome of the different trees, the method can identify instances that might present some difficulties to be correctly classified, and attribute some uncertainty to their prediction. We perform an extensive evaluation of the proposed method, and show that it is particularly suitable for ranking and reliability estimations. The ideas investigated in this paper may also be applied to other machine learning techniques, as well as combined with other methods for prediction certainty estimation.

Keywords: Decision trees, prediction certainty, soft classifiers

1 Introduction

In classification, it is often useful to have models that not only have high accuracy, but can also tell us how certain they are about their predictions. Classifiers that output some kind of reliability or likelihood of the quality of predictions are called soft classifiers [1]. A common example of a soft classifier is a probability estimator, which estimates the probability that a data instance belongs to a certain class. Rankers and reliability estimators are other forms of soft classifiers.

The standard way of turning decision trees (DTs) into soft classifiers consists of inferring the certainty of a prediction from the class distribution in the leaf responsible for the prediction. For example, if an instance x is classified in a leaf node with 90% of positive examples, we say that that x has 90% probability of being positive. However, it has been shown that these proportions can be misleading: the smaller the leaf is, the more likely the proportion is accidental, and not inherent to the population distribution in the leaf [2, 3]; and, as DT

learners try to make the leaves as pure as possible, the observed frequencies are systematically shifted towards 0 and 1 [3].

In this paper, we propose an alternative method to estimate prediction certainty in DTs. Assume that, given data set D and k classes $c_1, \ldots, c_k$, we want to make a prediction for an unseen instance x. Suppose that we learn a tree T that predicts with high "certainty" (according to leaf proportions) that x has class c_1. Logically speaking, if we would give the learner the prior knowledge that x has class c_1 (by simply adding (x, c_1) to D), the learner should not return a tree T_{c_1} that predicts with less certainty that the class of x is c_1. If it does, there is a logical contradiction, in the sense that more evidence about some fact being true leads to less certainty about it. Moreover, if we add (x, c_2) to D, and it turns out that the tree learned from the new dataset T_{c_2} makes a different prediction than T_{c_1}, also with high certainty, then we, as observers, know that there is actually high uncertainty about the class.

More specifically, our method works as follows. Given an unseen instance x, we can, for $i = 1, \ldots, k$, add (x, c_i) to D, giving a dataset D_i from which a tree T_{c_i} is learned, and look at the prediction T_i makes for x. If all T_{c_i} predict the same class c, we can be quite certain that c is the correct class. If multiple trees predict different classes, each with high certainty, we must conclude that these predictions are highly uncertain. We also propose a way to combine the predictions of the resulting trees.

We perform an extensive evaluation of the proposed method on 48 randomly selected UCI datasets. We compare our results to those of a standard DT learner and a standard ensemble method. The results show that our method tends to produce better ranking and reliability estimates than the other methods, while producing better probability estimates than standard trees and comparable probability estimates to ensembles. Additionally, compared to a closely related method for reliability estimation, we show that our method produces better estimates.

The remainder of the text is organized as follows. In Section 2 we discuss basic concepts related to prediction certainty in soft classifiers, with special focus on DTs; we also discuss related work and point the main differences w.r.t our method. In Section 3 we describe our new method in detail. In Section 4 we present experiments and results, and in Section 5 we conclude.

2 Background and related work

We first discuss three different ways of interpreting prediction certainty and how to evaluate them. Then, we briefly recall DTs and discuss related work.

2.1 Prediction certainty in soft classifiers

The notion of certainty associated to soft classifiers has been defined in different ways in the literature. We discuss three of them: probability, ranking and reliability estimations. For each one of them we present a measure to evaluate it; we use these measures to evaluate our method in the experimental section.

We say that a soft classifier is a probability estimator when it estimates for every possible class the true probability that a random instance with given attribute values belongs to that class. Probability estimations are usually evaluated with the Brier score [4], which is also known as mean squared error.

A ranking estimator orders the instances from high to low expectation that the instance belongs to a certain class c. For every pair of instances (x_1,x_2), the ranking defines if x_1 is more likely, equally likely or less likely to belong to c than x_2. As ranking estimation is defined in terms of pairs of sequences, we can say that this is a relative estimation. Probability estimation, on the other hand, is an absolute estimation, since the estimation for each prediction can be interpreted on its own. The ranking ability of a classifier is usually assessed using ROC analysis [5], by calculating the area under the ROC curve (AUC).

Finally, Kukar and Kononenko [6] use the term "reliability" to define the probability that a prediction is correct. This is, in principle, the same as probability estimation for the predicted class. However, Kukar and Kononenko [6] consider reliability in a more general way. They consider a prediction to be more "reliable" when it is has a higher probability to be correct, and evaluate the reliability skill of a classifier by assessing how well it can distinguish between correct and incorrect predictions based on the calculated reliability for each prediction. This is in fact a ranking evaluation where the predictions define only one rank over all classes together (in terms of correct and incorrect predictions), instead of internally to each class, as for standard ranking evaluation. For this evaluation, we can use the AUC; we call it AUC reliability to avoid confusion with the aforementioned AUC calculation.[1]

2.2 Decision Trees and certainty estimates

A decision tree (DT) [7] is a tree-shaped predictive model that assigns to a given instance a prediction by determining the leaf that the instance belongs to, and making a prediction based on this leaf. In a classification context, the predicted class is typically the one that occurs most frequently among the training instances covered by that leaf.

When a certainty estimate is needed, the relative frequency of the predicted class among the training instances covered by the leaf is often used as a probability estimate. However, it is well-known that standard DT learners do not yield very good probability estimates [2, 3]. Especially for small leaves, the sample class distribution can significantly deviate from the population class distribution, and it typically deviates towards lower class entropy (or higher purity), due to the learning bias of the tree learner. Moreover, DTs assign the same certainty estimates for instances falling into the same leaf, and do not exploit the fact that even in the same leaf there might be prediction certainty differences. For example, it is reasonable to assume that a borderline prediction in a leaf is more likely to be a misclassification than the other predictions in that leaf.

[1] In their original evaluation framework, Kukar and Kononenko [6] use information gain to evaluate the rank of reliability scores. The advantage of using AUC is that the evaluation is not dependent on a fixed discrimination threshold.

Several methods have been proposed in the literature to improve estimates in DTs. One approach is to apply a smoothing correction (e.g., the Laplace or m-estimate smoothing) to unpruned trees [2, 1]. These corrections are used to avoid extreme estimates (i.e, 0 or 1). Other group of methods either modify the decision tree learning (e.g., by learning fuzzy [8] or lazy DTs [9, 10]) or the way in which the predictions are made (e.g., by propagating the test instances across multiples branches of the tree and combining estimates from different leaf nodes [11], or by using internal nodes to make predictions [3]). Other methods use alternative probability calculations, e.g., by combining the class distribution from different nodes in the path taken by the test instance [12].

In contrast to these methods, which either develop a new type of DT learner or use different probability estimations, we propose a new way of using the results that can be obtained using any traditional DT learner. We do this by learning multiple trees, and combining their predictions. In contrast to ensemble methods, which also learn multiple trees, we modify the training data in a very restricted and controlled way to obtain different trees. We do this by just complementing the training data with a labeled version of the instance to be classified.

Our method is similar to the method for reliable classification proposed by Kukar and Kononenko [6]. Their method estimates a reliability score for each prediction based on a two-step approach: first an inductive learning step is performed, followed by a transductive step. More specifically, given an unseen instance x, the method makes a prediction for x using a standard machine learning method (e.g., a DT learner). The output of this inductive step is then used as input to the transductive step: x is added to the training data with the predicted label c and a new classifier is learned. Finally, the probability distributions output by both steps are compared in order to infer the reliability of predicting x as belonging to class c. The idea behind this reliability estimation is that the more different the probability distributions are, the less reliable the prediction is. This idea is based on the theory of randomness deficiency [13, 14]. Once a reliability score has been calculated for every prediction, the predictions are ranked according to their reliability, and a threshold is defined to separate the predictions into two populations: unreliable and reliable predictions. Kukar and Kononenko [6] propose a supervised procedure to find this threshold automatically.

In contrast to this method, we measure the sensitivity of the learned model w.r.t. all possible labels, instead of only using the label which is believed (predicted) to be the correct one. Our hypothesis is that measuring this sensitivity is crucial to obtaining good certainty estimates for DTs.

3 Proposed method

We start by giving the intuition of our method. Then, we describe its algorithm.

3.1 Intuition of the proposed method

We want to investigate the following questions w.r.t. the instance we want to classify: (1) "If we add the test instance to the training set with a different label

than the correct one, will the DT learner find a tree that is consistent with this instance according to the wrong label?"; (2) "How certain will the tree be about this prediction?"; (3) "How does this situation compare to the one where the instance is added to the training set with the correct label?".

Our method is therefore based on the idea that the learned DT can be dependent on the label of a single instance. One example of this effect can be seen in Fig. 1, which shows two DTs built to classify an instance x_1 from the Iris dataset [15]. For the left tree, x_1 was included in the training data with the correct label, *Iris-virginica*, while for the right one, x_1 was included with the wrong label *Iris-versicolor*. Observe that the trees make different predictions based on the pre-defined label for x_1. Intuitively, we cannot be very certain about the predicted label when the prediction model itself depends on the label we give to the instance. This uncertainty is not reflected in a certainty measure based only on leaf proportions. This intuition leads to the method we discuss next.

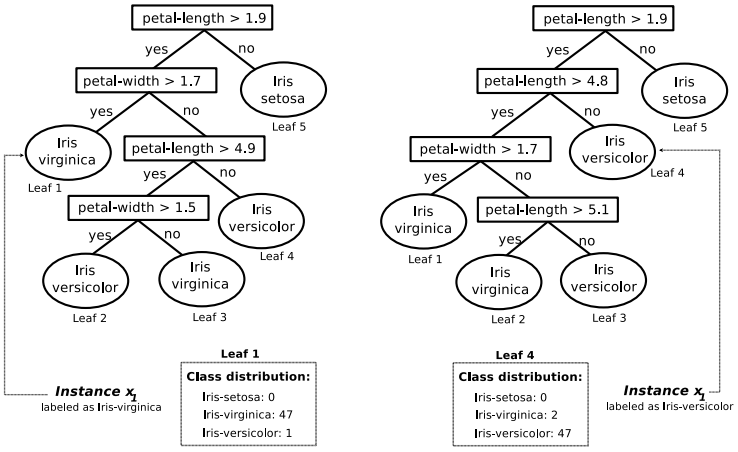

Fig. 1. Two DTs built for the same instance (x_1) of the Iris dataset. Left: x_1 was added in the training data with the correct label, *Iris-virginica*. Right: x_1 was added with label *Iris-versicolor*. For both cases the tree predicts x_1 with the pre-defined label.

3.2 Description of the method

The proposed method estimates the prediction certainty by comparing trees generated for the test instance x using different labels. As the correct label of x is not known to the method, we try all possibilities. More specifically, to classify an instance x, the method builds k trees, where k is the number of possible labels for the target attribute. For each label, we insert x in the training set with that label and induce a DT. In the end, we combine the prediction of all the trees.

To combine the predictions, we first calculate the prediction estimates for each tree by applying the Laplace smoothing (with $\alpha = 1$), then average over

the predicted values (Equation 1). In Equation 1, $\mathsf{Pred}(c)$ is the prediction for (probability of) class c, L_i is the leaf node responsible for the prediction in tree T_i, $|T|$ is the number of trees ($|T|$ equals the number of classes), $|L_i(c)|$ is the number of instances belonging to class c within L_i, and $|L_i(\neg c)|$ is the number of instances belonging to a different class within L_i.

$$\mathsf{Pred}(c) = \frac{\sum_{i=1}^{|T|} \frac{|L_i(c)|+1}{|L_i(c)|+|L_i(\neg c)|+|T|}}{|T|} \tag{1}$$

The algorithm is described in pseudocode in Fig. 2, where x is the test instance, D is the original training data and C is the set of possible classes. The procedure LearnTree builds a DT for the given training data, and the procedure CombinePred combines the predictions of the resulting trees $\mathbf{T}$.

```
procedure ObtainPrediction (x, D, C)
    for all c_i ∈ C:
        D' := D ∪ {(x, c_i)}
        T_i := LearnTree(D')
    return CombinePred({T_1, ..., T_k}, x, D, C)

procedure CombinePred(T, x, D, C)
    for all c_i ∈ C:
        pred_{c_i} := 0.0
        for all T_i ∈ T:
            pred_{c_i} := pred_{c_i} + (|L_i(c_i)|+1) / (|L_i(c_i)|+|L_i(¬c_i)|+|T|)
        pred_{c_i} := pred_{c_i} / |T|
    return {pred_{c_1}, ..., pred_{c_k}}
```

Fig. 2. Pseudocode to obtain the prediction for an instance

This procedure works fine in many cases, but sometimes an additional modification of the tree induction procedure LearnTree is needed. DTs typically handle continuous attributes by comparing them with a threshold. This threshold is put between two values belonging to instances with different classes. If we add a test instance to the training set, this introduces an additional possible threshold. This can have undesired effects, as shown in Fig. 3. In this example, the instance x, which is represented in Fig. 3.a as a circle, is a positive instance. This instance is far enough from the negative class (so that a standard DT would not have problems classifying it correctly), but is the positive instance the closest to the negative ones. If we allow our method to use the attribute values of x, it will always choose a decision boundary that perfectly separates the instances depending on the label attributed to x (as shown in Figs. 3.b and 3.c). This would lead our method to conclude that x is a difficult case to be classified, while actually it is not. To avoid this, the attribute values of x are not used when determining the possible split (test) values of the tree. However, they are still used when determining the heuristic value (information gain) of these possible splits.

4 Empirical evaluation

We present an extensive evaluation of our method, which we implemented as an extension of the DT learner Clus (http://dtai.cs.kuleuven.be/clus/); we call it Clus-TPCE (transductive prediction certainty estimation).

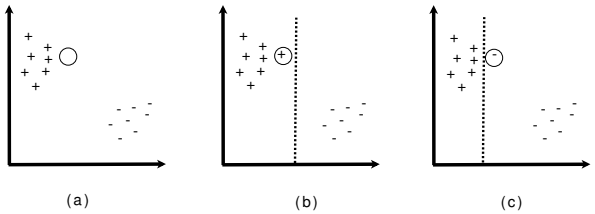

Fig. 3. Undesired effects of using a test instance when constructing decision boundaries.

4.1 Experimental setup

In total we used 48 randomly selected UCI datasets [15] in the experiments. For the datasets with no pre-defined test set, we used leave-one-out validation.

With these experiments we want to answer the following question: "Does the proposed method yields better prediction certainty estimates than a standard DT?"; for this comparison we use the original version of Clus, which we refer to as Clus-Orig. One could argue, however, that this comparison is not entirely fair since our method uses multiple trees to make the final prediction, and it is known that ensembles tend to yield better estimates than single trees [2]. Therefore, to ensure that an improvement of Clus-TPCE is not simply due to "ensemble effects", we also compare to standard bagging (Clus-Ens) with the same number of trees as we use for Clus-TPCE. For all methods, we use information gain as the splitting criterion to induce the trees and we do not apply pruning.

We evaluate the results as (a) ranking estimates, (b) probability estimates, and (c) reliability estimates. For the reliability estimation evaluation, we include the results for the procedure proposed by Kukar and Konenko [6] (c.f., Section 2.2). We implemented this procedure to estimate the reliability estimation of the predictions given by the original version of Clus; we call it Clus-K&K.

For each comparison, additionally to the results themselves, we also report the p-value of a two-sided Wilcoxon signed-rank. With this test we verify the hypothesis that the method with the largest number of wins, in terms of the evaluation measure in consideration, is statistically superior to the other one.

4.2 Evaluating probability estimation

We start by evaluating the results in terms of probability estimation, using the Brier score. The results are shown in Fig. 4 and summarized in Table 1.

Clus-TPCE obtains 36/0/12 wins/ties/losses compared to Clus-Orig, and a smaller average Brier score. When compared to Clus-Ens, Clus-TPCE obtains a smaller number of wins (21 wins against 27 wins for Clus-Ens), but has a smaller average Brier score. Interestingly, for the cases where both Clus-TPCE and Clus-Ens have a larger Brier score, the advantage is always in favor of Clus-TPCE (c.f. Fig. 4). For the cases with low scores, the results are in favor of Clus-Ens, but with a smaller difference. This explains why Clus-TPCE has a smaller average Brier score, even though it has a smaller number of wins.

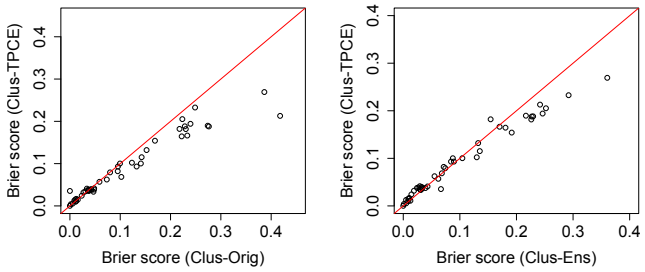

Fig. 4. Brier scores: Clus-TPCE vs. Clus-Orig (left); Clus-TPCE vs. Clus-Ens (right).

Table 1. Number of wins for each method, in terms of the Brier score, along with the p-value resulting from a two-sided Wilcoxon signed-rank test and the average estimates.

Clus-TPCE vs. Clus-Orig			Clus-TPCE vs. Clus-Ens			Average Brier score		
Clus-TPCE	Ties	Clus-Orig	Clus-TPCE	Ties	Clus-Ens	Clus-TPCE	Clus-Orig	Clus-Ens
36	0	12	21	0	27	0.087	0.109	0.096
p-value < 0.0001			p-value $= 0.5686$					

4.3 Evaluating ranking estimation

We now compare the methods w.r.t. their ranking ability. We generate a ROC curve for each class against all the other ones and report the average AUC value. The results are shown in Fig. 5 and summarized in Table 2.

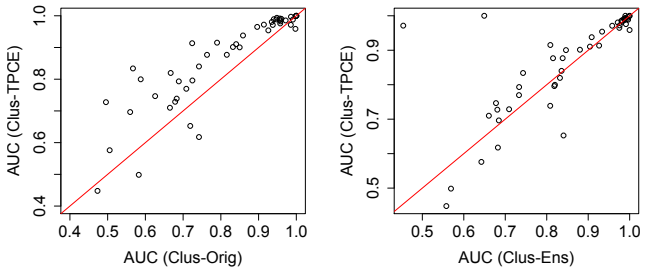

Fig. 5. AUC: Clus-TPCE vs. Clus-Orig (left); Clus-TPCE vs. Clus-Ens (right).

The results show that Clus-TPCE has a better ranking ability than the other methods. It obtains 37/3/8 wins/ties/losses compared to Clus-Orig and 29/1/18 wins/ties/losses compared to Clus-Ens. Note that these results are more in favor of Clus-TPCE than for the probability estimation evaluation. This is not completely unexpected. Clus-TPCE tends to shift the probability distribution output for some instances away from 1 (or 0), and towards the uniform probability (0.5, in case of a binary problem). This effect is stronger for the cases for which the methods finds evidence that there is a high degree of uncertainty

Table 2. Number of wins for each method, in terms of AUC, along with the p-value resulting from a two-sided Wilcoxon signed-rank test and the average estimates.

Clus-TPCE vs. Clus-Orig			Clus-TPCE vs. Clus-Ens			Average AUC		
Clus-TPCE	Ties	Clus-Orig	Clus-TPCE	Ties	Clus-Ens	Clus-TPCE	Clus-Orig	Clus-Ens
37	3	8	29	1	18	0.868	0.814	0.849
p-value < 0.0001			p-value $= 0.1706$					

associated to their predictions. However, as the Brier score assumes that the method should ideally report a probability of 1 for the true class and 0 for the other classes, having a number of predictions with probabilities closer to the uniform probability will have a negative effect on the final Brier score.

4.4 Evaluating reliability estimation

Finally, we evaluate Clus-TPCE w.r.t. reliability estimation. To that aim, we use the probability output for the predicted class as the reliability that the prediction is correct. We apply the same procedure for Clus-Orig and Clus-Ens. For Clus-K&K, we use the procedure proposed by Kukar and Kononenko [6]. To evaluate the reliability estimations, we use AUC reliability, as discussed in Section 2.1. The results are shown in Fig. 6 and summarized in Table 3.

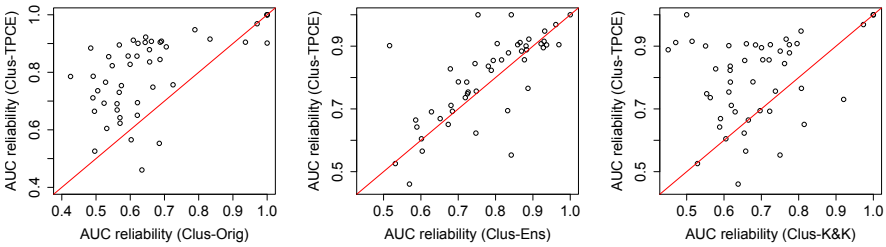

Fig. 6. AUC reliability: Clus-TPCE vs. Clus-Orig" (left); Clus-TPCE vs. Clus-Ens (center); Clus-TPCE vs. Clus-K&K (right).

Clus-TPCE outperforms the other three methods in terms of reliability estimation: it obtains 39/3/6 wins/ties/losses compared to Clus-Orig, 33/1/14 wins/ties/losses compared to Clus-Ens, and 33/2/13 wins/ties/losses compared to Clus-K&K. Furthermore, Clus-TPCE obtains the largest average AUC reliability, and the statistical tests indicate that the difference in the results is statistically significant.

5 Conclusions

We proposed a method for estimating prediction certainty. The new method was implemented as an extension of the DT learner Clus, but the ideas investigated in

Table 3. Number of wins for each method, in terms of AUC reliability, along with the p-value resulting from a two-sided Wilcoxon signed-rank test and the average estimates.

Clus-TPCE vs. Clus-Orig			Clus-TPCE vs. Clus-Ens			Clus-TPCE vs. Clus-K&K			Average AUC reliability	
Clus-TPCE	Ties	Clus-Orig	Clus-TPCE	Ties	Clus-Ens	Clus-TPCE	Ties	Clus-K&K	Clus-TPCE	Clus-Orig
39	3	6	33	1	14	33	2	13	0.7969	0.6498
p-value < 0.0001			p-value $= 0.0208$			p-value < 0.0001			Clus-Ens	Clus-K&K
									0.7681	0.6836

this paper can also be applied to other machine learning methods, in particular those where the label of a single example can influence the learned model.

Our new method builds a DT for an input data consisting of the training data plus the instance to be classified, which is labeled with one of the possible class values. This procedure is repeated for every class, and in the end all induced trees are compared. This comparison allows us to identify instances that might present difficulties to be correctly classified, and to attribute some uncertainty to their predictions. We evaluated our method on 48 UCI datasets, and compared it to the original Clus and to standard bagging. The results showed that the new method yields better ranking and reliability estimates than the other methods. Regarding probability estimation, the proposed method yields better estimates than the original method and comparable estimates to the ensemble method. We also compared to the method by Kukar and Kononenko [6] w.r.t. reliability estimation, and show that our method produces better estimates.

Since our method is complementary to the other methods suggested in the literature (c.f. Section 2.2), they can be easily combined: simply use those methods instead of Clus to learn multiple trees for different versions of the test instances. Investigating these combinations is a very interesting avenue for future work.

Note that our method makes predictions for only one instance at a time, which increases its computational cost. This raises the question if the method can be extended in order to be able to perform the whole process once for a whole batch of unseen instances. A straightforward extension of the method consists of generating the same number of trees as the number of possible labels (once for all instances), where for every tree each instance to be classified receives a random label, with the constraint that the same instance will never receive the same label in two or more trees. This constraint assures that each instance receives each possible label once, allowing us to apply the same strategy to combine predictions used in this paper. However, as each generated tree is not only subject to the influence of the labeling of a single instance (but a batch of instances instead), it is not trivial to analyze if the prediction obtained for an instance is a result of how that instance was labeled, and/or how the other instances were labeled. In fact, we have performed preliminary experiments with this extended method to test its ranking ability, and the results showed that it produces better results

than a standard tree learner, but slightly worse results than standard bagging. Therefore, the extension of the proposed method remains for future research.

Acknowledgments

This research was supported by projects G.0413.09 "Learning from data originating from evolution" and G.0682.11 "Declarative experimentation", funded by the Research Foundation - Flanders (FWO-Vlaanderen).

References

1. Ferri, C., Flach, P., Hernández-Orallo, J.: Improving the auc of probabilistic estimation trees. In: Proc. of the 14th European Conference on Machine Learning. Volume 2837 of LNCS. Springer (2003) 121–132
2. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning **52**(3) (2003) 199–215
3. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: Proc. of the 18th International Conference on Machine Learning, Morgan Kaufmann (2001) 609–616
4. Brier, G.W.: Verification of forecasts expressed in terms of probability. Monthly weather review **78**(1) (1950) 1–3
5. Fawcett, T.: An introduction to roc analysis. Pattern recognition letters **27**(8) (2006) 861–874
6. Kukar, M., Kononenko, I.: Reliable classifications with machine learning. In: Proc. of the 13th European Conference on Machine Learning, Springer (2002) 1–8
7. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1) (1986) 81–106
8. Hüllermeier, E., Vanderlooy, S.: Why fuzzy decision trees are good rankers. IEEE Transactions on Fuzzy Systems **17**(6) (2009) 1233–1244
9. Margineantu, D.D., Dietterich, T.G.: Improved class probability estimates from decision tree models. In Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B., eds.: Nonlinear Estimation and Classification; Lecture Notes in Statistics. Volume 171. Springer-Verlag (2001) 169–0184
10. Liang, H., Yan, Y.: Improve decision trees for probability-based ranking by lazy learners. In: Proc. of the 18th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society (2006) 427–435
11. Ling, C.X., Yan, R.J.: Decision tree with better ranking. In: Proc. of the 20th International Conference on Machine Learning, Morgan Kaufmann (2003) 480–487
12. Wang, B., Zhang, H.: Improving the ranking performance of decision trees. In: Proc. of the 17th European conference on Machine Learning, Springer (2006) 461–472
13. Li, M., Vitanyi, P.: An introduction to Kolmogorov complexity and its applications. Springer Verlag (1997)
14. Vovk, V., Gammerman, A., Saunders, C.: Machine-learning applications of algorithmic randomness. In: Proc. of the 16th International Conference on Machine Learning. (1999) 444–453
15. Bache, K., Lichman, M.: UCI machine learning repository [http://archive.ics.uci.edu/ml] (2013) University of California, Irvine, School of Information and Computer Sciences.