A Comparison of Approaches for Learning Probability Trees

Daan Fierens, Jan Ramon, Hendrik Blockeel, and Maurice Bruynooghe

Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium
{daanf, janr, hendrik, maurice}@cs.kuleuven.be

Abstract. Probability trees (or Probability Estimation Trees, PET's) are decision trees with probability distributions in the leaves. Several alternative approaches for learning probability trees have been proposed but no thorough comparison of these approaches exists.

In this paper we experimentally compare the main approaches using the relational decision tree learner Tilde (both on non-relational and on relational datasets). Next to the main existing approaches, we also consider a novel variant of an existing approach based on the Bayesian Information Criterion (BIC). Our main conclusion is that overall trees built using the C4.5-approach or the C4.4-approach (C4.5 without post-pruning) have the best predictive performance. If the number of classes is low, however, BIC performs equally well. An additional advantage of BIC is that its trees are considerably smaller than trees for the C4.5- or C4.4-approach.

Keywords: (Relational) Decision trees, probability estimation.

1 Introduction

Probability trees (or Probability Estimation Trees, PET's) are decision trees with in the leaves probability distributions on a set of classes [11]. They are useful in a number of ways, e.g. for ranking instances according to the probability of belonging to a certain class [11] or as a compact way of specifying conditional probability distributions (for instance in Bayesian networks) [5].

Several alternative approaches for learning probability trees have been proposed in the literature but currently no thorough comparison of these approaches exists. Hence, it is unclear which approaches are preferable under which circumstances. The goal of this paper is to compare the main existing approaches and a novel variant. We incorporated them in the relational decision tree learner Tilde [2] and evaluate them by performing experiments on benchmark datasets and on manipulated datasets. We use both non-relational and relational datasets.

In Section 2 we give a high-level algorithm for learning probability trees, of which the main existing approaches are instantiations. In Section 3 we experimentally compare these approaches. In Section 4 we conclude.

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 556–563, 2005.

2 Learning Probability Trees

Probability trees are learned from a dataset D of instances labelled with their true class. Tilde [2], the *relational* decision tree learner we use, represents instances as first-order logic interpretations and tests in internal nodes as Prolog queries (since such tests either succeed or fail, trees are binary). We use Tilde because it can handle relational datasets in addition to non-relational ones.

Probability trees are typically learned in two steps. In the first step we top-down induce a tree $\mathcal{T}$ as follows. We start from the empty tree and for each candidate-test T compute the heuristic value $h(T)$. Call T_{best} the best of all candidate-tests, i.e. $T_{best} = argmax_T(h(T))$. If $h(T_{best}) < Thr$ with Thr a certain threshold we return a leaf (so Thr determines a kind of stopping-criterion). Otherwise we make T_{best} the root of the tree and apply the same procedure recursively to learn the left- and right-subtrees. In the second step we can apply bottom-up *post-pruning* (to avoid overfitting): we first prune the left- and right-subtrees giving $\mathcal{T}_{pruned}$ and then check whether $\mathcal{T}_{pruned}$ is 'better' than a single leaf according to some pruning-criterion.

The main approaches all fit into this generic two-step approach and correspond to different choices of the heuristic function $h(.)$, Thr and the pruning-criterion (if post-pruning is used). We now briefly discuss these approaches. Some more details are given in [4].

C4.5 (error-based post-pruning) Provost and Domingos [11] discuss learning probability trees using C4.5. This means that $h(T)$ is information-gain of T ($gain(T)$), Thr is 0 (any information-gain is acceptable) and error-based post-pruning is applied[1]. We refer to Tilde applied with these parameters as **C4.5**.

C4.4 (no pruning) Provost and Domingos [11] argue that pruning is harmful for probability trees. The idea is that probability estimation is conceptually different from majority-classification (the focus of C4.5). Hence they propose to use C4.4, i.e. C4.5 without any post-pruning. We refer to Tilde applied with these parameters as **C4.4**. Obviously, **C4.4** builds extremely large trees.

Minimum Description Length (MDL) Friedman and Goldszmidt [5] define an MDL-score for probability trees and use it to derive a *stopping-criterion* for the tree-building. Concretely this means that $h(T)$ is $N_{node}.gain(T)$ and Thr is $0.5\ (NbClasses - 1)\ log_2 N + log_2 NbTests + 2$, where N is the total number of examples, N_{node} is the number of examples in the current node and $NbTests$ is the number of candidate-tests considered. In terms of MDL, $h(T)$ is the decrease in description length of the data and Thr is the increase in description length of the tree due to adding T to the tree [5]. We refer to this approach as **MDLs**.

Using MDL as a stopping-criterion (using the above Thr) we easily get stuck in local optima of the MDL-score. As an alternative we can use $Thr = 0$ and

[1] Like Provost and Domingos, we do not apply 'collapsing' [11] since it harms probability estimates too much.

apply *post-pruning* based on MDL-reasoning [5]. We refer to this approach as **MDLp**. **MDLp** builds trees at least as large as those for **MDLs**.

Bayesian Information Criterion (BIC) Inspired on the above MDL-score, we can define a BIC-score for probability trees (as far as we know, we are the first to apply BIC to probability trees). BIC [12] is a general approach equivalent to a form of MDL where the the description length of the model only depends on its number of independent parameters. In the context of probability trees this means that $h(T)$ is the same as for **MDLs** but Thr now is 0.5 $(NbClasses - 1)$ $log_2 N$. We refer to this approach as **BICs**. **BICs** builds trees at least as large as those for **MDLs** (since Thr is strictly lower for **BICs**).

As an alternative we can again use $Thr = 0$ and apply post-pruning based on BIC-reasoning. We refer to this approach as **BICp**. **BICp** builds trees at least as large as those for **BICs**.

Chi-square score Neville et al. [10] discuss learning probability trees using the chi-square (χ^2) statistic. Concretely, $h(T)$ is the χ^2-score of T and Thr is determined by the sampling distribution of χ^2 for significance level $p = \frac{0.1}{NbTests}$ and degrees of freedom $df = NbClasses - 1$. No post-pruning is used. We refer to this approach as **Chi**.

The above list is not complete. Some other existing approaches and the reasons for not considering them in our work are given in [4].

3 Experimental Comparison

To the best of our knowledge, **C4.5** and **C4.4** are the only of the above approaches that have already been compared (Provost and Domingos [11] conclude that neither of the two is significantly better than the other). In this section we make a thorough comparison of all approaches mentioned above.

3.1 Experiments on Benchmark Datasets: Setup and Results

Table 1 gives an overview of the datasets used. All non-relational datasets are from the UCI-repository [8], except *asm* [6]. All relational datasets are standard ILP-benchmarks [1, 7, 13] (*trains* was artificially generated [9]; for *hiv* the classes 'inactive' and 'moderately active' were taken together).

To evaluate predictive performance of probability trees we use the *Area Under the ROC-curve (AUC)*, or *Expected AUC* for multi-class problems [11]. As noted in [11], AUC can be used as a quality measure for probability estimates since a high AUC indicates that, with proper re-calibration of probabilities, probability estimates will be good. To evaluate the size of the trees we use the *number of leaf nodes* (this is the number of internal nodes plus one since trees are binary). We perform 10-fold cross-validation (except for datasets smaller than 500 examples where we perform five times 3-fold cross-validation to keep test-sets large enough) and report averages and standard deviations of results over the test-sets.

Table 2 shows the experimental results (the upper half of each table shows two-class problems, the lower half shows multi-class problems). We compared

Table 1. Characteristics of the non-relational (left) and relational (right) datasets: number of examples, number of classes and number of candidate-tests for the root

	N	NbClasses	NbTests
asm	999	2	170
audiology	226	24	125
pen digits	7494	10	160
primary tumor	339	22	29
voting	435	2	16
yeast	1484	10	45

	N	NbClasses	NbTests
biodegradability	328	2	47
carcinogenesis	330	2	305
diterpenes	1504	23	210
hiv	41768	2	49
mutagenesis	230	2	139
trains	25000	2	73

Table 2. Experimental results: AUC (upper table, in %) and tree size (lower table)

	C4.4	**C4.5**	**MDLs**	**MDLp**	**BICs**	**BICp**	**Chi**
asm	58.7±4.7	**62.8±3.7**	**69.6±4.4**	**69.6±4.4**	**67.4±3.8**	**66.0±3.5**	**69.5±4.3**
biodegr.	74.9±6.7	75.4±4.8	63.9±4.0	64.1±2.9	73.0±4.7	71.6±5.3	64.5±4.0
carcinog.	59.1±5.8	59.3±6.7	50.0±0.0	50.0±0.0	54.0±2.7	57.2±3.2	50.0±0.0
hiv	74.8±3.3	53.9±1.1	64.1±2.1	70.5±3.1	66.8±3.3	72.4±3.3	67.0±3.2
mutagenesis	77.0±5.0	71.7±4.9	71.6±7.0	74.5±4.4	72.8±8.2	75.8±6.6	71.9±6.0
trains	86.3±0.5	**89.2±0.6**	**89.0±0.6**	**89.2±0.5**	**89.2±0.6**	**89.4±0.5**	**89.3±0.5**
voting	98.6±0.7	96.5±2.1	97.4±1.1	97.7±1.4	98.3±1.5	98.4±0.9	98.5±1.2
audiology	98.8±0.8	98.7±0.7	75.3±2.7	75.6±2.4	80.9±5.5	81.2±5.0	97.4±1.1
diterpenes	85.4±2.5	85.8±1.8	70.4±3.7	71.2±2.6	71.5±3.8	72.0±2.9	82.0±2.5
pen digits	99.5±0.1	99.4±0.1	98.7±0.2	98.7±0.2	98.9±0.3	98.9±0.3	99.4±0.1
pr. tumor	71.5±3.1	73.3±4.3	65.8±5.3	67.7±1.8	67.9±1.9	67.5±1.9	72.7±4.0
yeast	75.8±3.2	**79.6±3.5**	**78.3±2.5**	**78.3±2.5**	**78.3±2.5**	**78.3±2.5**	**79.1±4.0**

	C4.4	**C4.5**	**MDLs**	**MDLp**	**BICs**	**BICp**	**Chi**
asm	352±14	64±11	3±0	3±0	6±2	7±3	3±0
biodegradability	72±5	30±5	2±1	2±1	10±2	12±4	4±2
carcinogenesis	95±8	35±7	1±0	1±0	5±2	8±3	1±0
hiv	1391±76	15±2	8±3	32±4	26±3	55±4	33±3
mutagenesis	42±4	9±5	2±0	2±0	5±2	5±2	2±0
trains	4664±55	484±29	38±2	49±4	68±6	92±6	64±3
voting	21±3	6±3	3±1	3±1	6±1	6±1	6±1
audiology	24±1	24±2	2±0	2±0	3±1	3±1	18±3
diterpenes	127±11	64±5	3±1	4±1	4±1	4±1	14±2
pen digits	254±8	214±7	36±2	36±2	42±3	42±3	126±4
primary tumor	129±5	79±7	2±0	2±0	2±0	2±0	5±1
yeast	447±34	123±13	6±0	6±0	6±0	6±0	23±3

AUC's for each of the approaches to AUC's for **C4.4** (which performs best) by means of two-tailed paired t-tests (p=0.05). An AUC in boldface (resp. underlined) indicates that this AUC is significantly higher (resp. lower) than that for **C4.4**. A more detailed statistical analysis is given in [4].

As for running times, the approaches using an explicit stopping criterion (**MDLs**, **BICs** and **Chi**) are a factor 8 to 52 faster than the others [4].

3.2 Discussion and Further Experiments

As mentioned we used **C4.4** as the reference method for performing significance tests on the AUC's in Table 2. Hence, when reporting wins/ties/losses for a certain method we always mean wins/ties/losses of that method versus **C4.4**.

Overall Observations. From Table 2 we see that overall **C4.4** performs best although there are some datasets where it is outperformed (most notably *asm*). **C4.5** performs almost as well (the number of wins/ties/losses for **C4.5** is 3/5/4). This confirms the conclusions of Provost and Domingos [11]. Trees for **C4.5**, however, are always significantly smaller than trees for **C4.4**, except on *audiology* (see [4] for a statistical analysis). Note that the dramatic performance of **C4.5** on *hiv* is probably due to the strongly skewed class-distribution (only 3.6% of positive examples). In [3] we discuss a controlled experiment showing that **C4.5** indeed performs badly on strongly skewed datasets.

MDL and **Chi** overall perform clearly worse than **C4.4** (wins/ties/losses for **MDLs** are 3/0/9, for **MDLp** 3/2/7, for **Chi** 3/3/6). Trees for MDL and **Chi** are always significantly smaller than trees for **C4.4** or **C4.5**, except on *voting* and *hiv* [4]. Results for BIC are discussed in the next section.

Influence of the Number of Classes. An interesting observation from Table 2 is that BIC performs well for two-class problems but not for the multi-class problems we considered (that all have $NbClasses \geq 10$). On the two-class problems, wins/ties/losses for **BICs** are 2/3/2, for **BICp** 2/4/1. This means that on two-class problems, BIC performs at least as well as **C4.4** (or any other approach), while BIC has the additional advantage of building much smaller trees.

On the multi-class problems, the picture looks rather different, however. Wins/ties/losses for **BICs** and **BICp** are both 1/0/4: **C4.4** clearly outperforms BIC here. One explanation for this is the fact that Thr for **BICs** is $0.5\ (NbClasses - 1)\ log_2 N$. So if $NbClasses$ is high, then Thr is high as well and only tests T with a very high heuristic value $h(T)$ (larger than Thr) are accepted and hence very small trees are built. This suggests that **BICs** could be improved by making the dependency of Thr on $NbClasses$ less strong (i.e. less than linear) since then trees built for a high $NbClasses$ will be larger (although such a modification would deviate from the original theoretical foundations of BIC [12]). Similar remarks apply to **BICp**. Note that these observations (and their explanation) also hold for MDL but to a smaller extent, e.g. for **MDLp** wins/ties/losses on two-class problems are 2/2/3, on multi-class problems 1/0/4.

We performed an additional controlled experiment to investigate the influence of the number of classes. We started from *diterpenes*, a dataset having 23 classes on which **BICs** and **BICp** performed badly. In each step we merged the two least frequent classes, until only three classes were left. Figure 1 shows the results obtained from 10-fold cross-validation. In the top panels we show AUC, in the bottom panels tree size (on a logarithmic axis). We show **MDLs**, **MDLp**, **BICs** and **BICp** in the left panels and **BICp** (the best of the previous four), **C4.5**,

C4.4 and **Chi** in the right panels. We see that results for **MDLs**, **MDLp**, **BICs** and **BICp** are always very close to each other. For these approaches both tree size and AUC quickly decrease as the number of classes increases. Interestingly, there is almost no such decrease for **C4.5**, **C4.4** and **Chi**. Figure 1 shows that as a consequence **MDLs**, **MDLp**, **BICs** and **BICp** can compete with the other approaches when *NbClasses* is 3 or 5, but are outperformed when *NbClasses* goes higher. This confirms the above observation that MDL and BIC do not work well for multi-class problems, and its explanation.

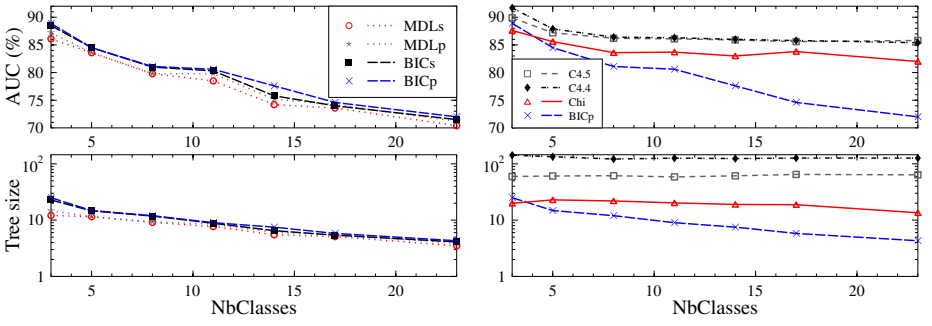

Fig. 1. Influence of the number of classes for *diterpenes*

Influence of the Number of Examples We also investigated the influence of the number of examples in the dataset. We learned trees from subsets of *hiv* containing a variable number of examples (we evaluate them on a separate test-set of 6768 examples). Figure 2 shows the results. We see that results (both AUC and tree size) for **MDLs**, **MDLp**, **BICs** and **Chi** are very close to each other for all sizes of the dataset. For small datasets (N<15000) also **BICp** is very close to the previous four. Interestingly however, for larger datasets **BICp** learns larger trees than the others, resulting in higher AUC's. This suggests that for larger datasets BIC for post-pruning (**BICp**) is more useful as compared to BIC as a stopping-criterion (**BICs**).

We performed the same experiment for *trains* (using a test-set of 5000 examples), see Figure 3. Again results for **MDLs**, **MDLp**, **BICs** and **Chi** are close to each other for all sizes of the dataset (except the very low ones, $N \leq 3000$). Unlike for *hiv*, however, **BICp** is very close to the previous four for all sizes of the dataset and does not become better than these four for larger datasets. Also we see that **C4.4** seems to overfit for all sizes of the dataset (it builds the largest trees but has the lowest AUC). The degree of overfitting is not heavily influenced by the size of the datasets. This is probably due to two competing effects [11]. On the one hand: if the dataset grows, trees grow as well (Figure 3), increasing the probability of overfitting. On the other hand, if the dataset grows, the number of examples in the leaves would increase, making probability estimates more reliable, thus decreasing the probability of overfitting. Why **C4.4** overfits on some datasets but not on others is currently an open question.

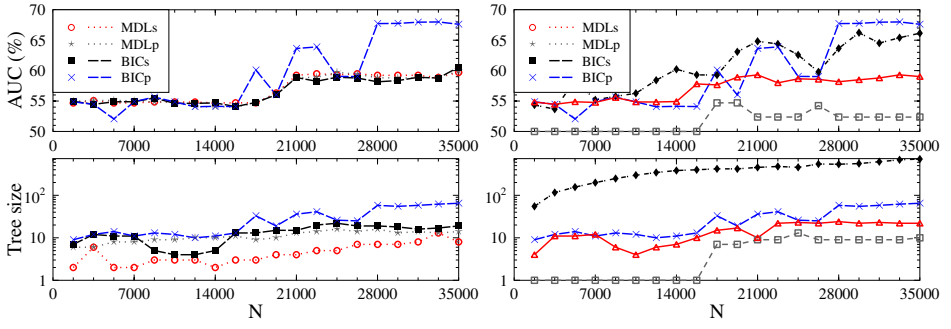

Fig. 2. Influence of the number of examples N for *hiv*

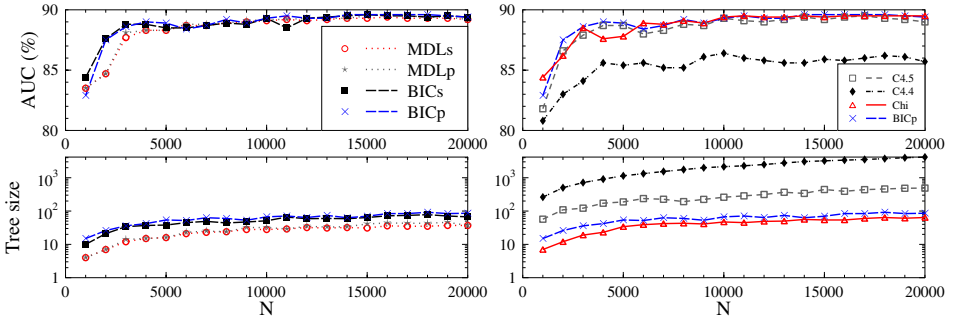

Fig. 3. Influence of the number of examples N for *trains*

Influence of the Number of Tests. We also investigated the influence of the number of candidate-tests. This seemed interesting since this parameter occurs in the definition of Thr for MDL and **Chi** but no interesting trends were found [4].

4 Conclusions

We reviewed and experimentally compared the main approaches for learning probability trees including a novel variant based on the Bayesian Information Criterion (BIC). We conclude that overall the C4.4-approach performs best, and the C4.5-approach second best. However, trees are much smaller for the latter than for the former. Interestingly, if the number of classes is low, BIC performs equally well. An additional advantage of BIC is that its trees are considerably smaller than trees for the C4.5- or C4.4-approaches. If the number of classes is too high (≥ 8 in our experiments), BIC fails because trees are too small.

An interesting idea for future research is to try to improve performance of BIC on multi-class problems by decreasing the influence of the number of classes on the stopping- or post-pruning-criterion.

Acknowledgements

DF is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen). JR and HB are post-doctoral fellows of the Fund for Scientific Research (FWO) of Flanders. The authors thank Kristian Kersting and the reviewers for useful comments.

References

[1] ILPnet2 applications descriptions. `http://www-ai.ijs.si/~ilpnet2/apps/`.
[2] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
[3] D. Fierens, J. Ramon, H. Blockeel, and M. Bruynooghe. A comparison of approaches for learning first-order logical Probability Estimation Trees. In *Inductive Logic Pogramming, 15th Int. Conference (ILP05), Late-breaking Papers*, 2005.
[4] D. Fierens, J. Ramon, H. Blockeel, and M. Bruynooghe. A comparison of approaches for learning probability trees. Technical Report CW 418, Department of Computer Science, Katholieke Universiteit Leuven, 2005.
[5] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Learning and Inference in Graphical Models*. Cambridge: MIT Press, 1998.
[6] A. J. Knobbe. Data mining for adaptive system management. In *Proceedings of the 1st International Conference and exhibition on the Practical Application of Knowledge Discovery and Data Mining (PADD97)*, 1997.
[7] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD01)*, pages 136–143, 2001.
[8] C. Merz and P. Murphy. UCI repository of machine learning databases `http://www.ics.uci.edu/~mlearn/mlrepository.html`, 1996. Irvine, CA: University of California, Department of Information and Computer Science.
[9] D. Michie, S. Muggleton, D. Page, and A. Srinivasan. To the international computing community: A new east-west challenge. Technical report, Oxford University Computing Laboratory, Oxford, UK, 1994. Available at `ftp.comlab.ox.ac.uk`.
[10] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD03)*, 2003.
[11] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52:199–216, 2003.
[12] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
[13] A. Srinivasan, R. King, and D. Bristol. An assessment of ILP-assisted models for toxicology and the PTE-3 experiment. In *Proceedings of the seventh international conference on Inductive Logic Programming (ILP99)*, 1999.