Lazy Learning for Improving Ranking of Decision Trees

Han Liang[1⋆], Yuhong Yan[2]

[1]Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada E3B 5A3
han.liang@unb.ca
[2] National Research Council of Canada
Fredericton, NB, Canada E3B 5X9
yuhong.yan@nrc.gc.ca

Abstract. Decision tree-based probability estimation has received great attention because accurate probability estimation can possibly improve classification accuracy and probability-based ranking. In this paper, we aim to improve probability-based ranking under decision tree paradigms using AUC as the evaluation metric. We deploy a lazy probability estimator at each leaf to avoid uniform probability assignment. More importantly, the lazy probability estimator gives higher weights to the leaf samples closer to an unlabeled sample so that the probability estimation of this unlabeled sample is based on its similarities to those leaf samples. The motivation behind it is that ranking is a relative evaluation measurement among a set of samples, therefore, it is reasonable to yield the probability for an unlabeled sample with reference to its extent of similarities to its neighbors. The proposed new decision tree model, *LazyTree*, outperforms C4.5, its recent improvement C4.4 and their *state-of-the-art* variants in AUC on a large suite of benchmark sample sets.

1 Introduction

A learning model is induced from a set of labeled samples represented by the vector of an attribute set $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$ and a class label C. Classic decision trees are typical decision boundary-based models. When computing class probabilities, decision trees use the observed frequencies at leaves for estimation. For instance, if a leaf contains 100 samples, 60 of which belong to the positive class and the others are in the negative class, then for any unlabeled sample that falls into the leaf, decision trees will assign the same positive probability of $\hat{p}(+|\mathbf{A_p} = \mathbf{a_p})$=60%, where $\mathbf{A_p}$ is the set of attributes from the leaf to the root. However, this incurs two problems: high bias (traditional tree inductive algorithm tries to make leaves homogeneous, therefore, the class probabilities are systematically shifted toward zero or one) and high variance (if the number of samples at a leaf is small, the class probabilities are unreliable) [9].

⋆ This work was done when the author was a visiting worker at Institute for Information Technology, National Research Council of Canada.

Accurate probability estimation is important for problems like classification and probability-based ranking. Provost and Domingos' *Probability Estimation Trees* (PETs) [7] turn off pruning and collapsing in C4.5 to keep some branches that may not be useful for classification but are crucial for accurate probability estimation. The final version is called C4.4. PETs also use *Laplace* smoothing to deal with the pure nodes that contain samples from the same class. Instead of assigning a probability of 1 or 0, smoothing methods try to give a more modest estimation. Other smoothing approaches, such as m-Branch [2] and *Ling&Yan*'s algorithm [6], are also developed. Using a probability density estimator at each leaf is another improvement to tackle the "uniform probability distribution" problem of decision trees. Kohavi [4] proposed an Naïve Bayes Tree (NBTree), in which a naïve Bayes is deployed at each leaf to produce probabilities. The intuition behind it is to take advantage of leaf attributes $\mathbf{A_l}(l)$ for probability estimation. Therefore, $p(c|\mathbf{e_t}) \approx p(c|\mathbf{A_p}(l), \mathbf{A_l}(l))$.

The main objective of this paper is to improve the ranking performance of decision trees. The improvement comes from two aspects. Firstly, decision trees work better when the sample set is large. After several splits of attributes, the samples at the subspaces can be too few on which to base the probability. Therefore, although employing a traditional tree inductive process, we stop the splits once the samples are reduced to some extent and deploy probability estimators at leaves. The probability estimators assign distinct probabilities to different samples. Thus, the probability generated by such a tree is more accurate than assigning a uniform probability for the samples falling into the same leaf. Secondly, and more importantly, we observe that probability-based ranking is indeed a relative evaluation measurement where the correctness of ranking depends on the relative position of a sample among a set of other samples. In our paper, a lazy probability estimator that calculates the probability of an unlabeled sample based on its neighbors is designed for better ranking quality. The lazy probability estimator finds m closest neighbors at a leaf for the unlabeled sample and calculate a weight for each neighbor using a newly proposed similarity score function. We generate the probability estimates for this unlabeled sample by normalizing all weights of the neighbors at the leaf in terms of their class values. The new model is called *LazyTree*. AUC [3] is used to evaluate our method. On a large suite of 36 standard sample sets, empirical results indicate that *LazyTree* performs substantially better than C4.5, C4.4 and their variants with other methods designed for optimal ranking, such as m-Branch, *Ling&Yan*'s algorithm and a voting strategy–*bagging*, in yielding accurate ranking.

2 Using Lazy Learner to Improve Tree-Based Ranking

2.1 The Lazy Learner

Our work aims to calibrate probability-based ranking of decision trees. Improvement comes from two aspects. Firstly, we want to trade-off between *bias* and *variance* of decision trees by deploying a probability estimator at each leaf. *Probability Estimator* is defined as:

Definition 1. *Given a set of unlabeled samples E and a set of class labels $C = \{c_i\}$, a* ***Probability Estimator*** *is a set of functions $p_i : E \mapsto [0, 1]$, such that $\forall \mathbf{e} \in E$, $\sum p_i(\mathbf{e}) = 1$.*

The probability estimators give distinct probabilities to different samples. As a result, the probability estimation generated by such trees is more precise than the uniform probability assignment for the samples falling into the same leaf.

Secondly, and more essentially, we observe that probability-based ranking is indeed a ***relative evaluation measurement*** where the correctness of ranking depends on the relative position of a sample among a set of other samples. For instance, for a binary-class problem, if assigned class probabilities of a positive sample $\mathbf{e}_+$ and a negative sample $\mathbf{e}_-$ satisfy $p(+|\mathbf{e}_+) > p(+|\mathbf{e}_-)$, it is a correct ranking. For multi-classes, the right ranking means $p_i(\mathbf{e} \in c_i) > p_i(\mathbf{e}' \notin c_i)$.

These inspire us to use a ***lazy probability estimator*** which calculates the probability of a sample based on its neighbors. The lazy probability estimator finds m closest neighbors at a leaf (here m means all the samples at this leaf) for an unlabeled sample and calculate a weight for each neighbor using a newly proposed similarity metric.

Assume that sample $\mathbf{e}$ can be represented by an attribute vector as $< a_1(\mathbf{e}), a_2(\mathbf{e}), ..., a_n(\mathbf{e}) >$, where $a_i(\mathbf{e})$ denotes the value of ith attribute. The distance between two samples $\mathbf{e_1}$ and $\mathbf{e_2}$ is calculated in (1):

$$d(\mathbf{e_1}, \mathbf{e_2}) = \sqrt{\sum_{i=1}^{n} \delta(a_i(\mathbf{e_1}), a_i(\mathbf{e_2}))}, \tag{1}$$

$\delta(a_i(\mathbf{e_1}), a_i(\mathbf{e_2}))$ outputs zero if $a_i(\mathbf{e_1})$ is equivalent to $a_i(\mathbf{e_2})$, otherwise it outputs one.

For an unlabeled sample $\mathbf{e_t}$ and a set of labeled samples $\{\mathbf{e_i}|i = 1, ..., n\}$ falling in a leaf, we assign a weight to each of the labeled sample $\mathbf{e_i}$ based on its distance to $\mathbf{e_t}$ (as in 2):

$$w_i = 1 - \frac{d_i}{\sum_{i=1}^{n} d_i}. \tag{2}$$

In (2), $d_i = d(\mathbf{e_t}, \mathbf{e_i})$ is the distance of an unlabeled sample $\mathbf{e_i}$ to $\mathbf{e_t}$. Notice that for any two labeled samples $\mathbf{e_i}$ and $\mathbf{e_j}$, the shorter the distance to $\mathbf{e_t}$ (assuming that $d_i \leq d_j$), the larger weight the sample ($w_i \geq w_j$). That implies a labeled sample nearest to $\mathbf{e_t}$ contribute most when calculating the probability for $\mathbf{e_t}$.

We generate the probability estimates of the unlabeled sample by normalizing all weights of the labeled samples at a leaf in terms of their class values, as in (3):

$$p(c_j|\mathbf{e_t}) = \frac{\sum_{k=1}^{m} w_k^j + \frac{1}{|C|}}{\sum_{k=1}^{n} w_k + 1}, \tag{3}$$

where n represents the number of samples at a leaf and m is the number of samples that belong to c_j.

2.2 *LazyTree* Induction Algorithm

We deploy a lazy probability estimator at each leaf of a decision tree and call this model *LazyTree*. To induce the model, we adopt a heuristic search process, in which we exhaustively build all possible trees in each step and keep only the best one for the next level expansion. Suppose that finite k attributes are available. When expanding the tree at level q, there are *k-q+1* attributes to be chosen. On each iteration, each candidate attribute is chosen as the root of the (sub) tree, the generated tree is evaluated, and we select the attribute that achieves the highest *gain ratio* as the next level node to grow the tree. We consider two criteria for halting the search process. We could stop splitting when none of the alternative attributes can statistically significantly upgrade the classification accuracy. Or, to avoid the "fragmentation" problem, there are at least 30 samples at the current node. Besides, we still permit splitting if the relative increment in accuracy is not a negative value, which is greedier than C4.5. The tree model is represented as T. An unlabeled sample $\mathbf{e_t}$ is assigned a set of class probabilities as in Algorithm 1.

Algorithm 1 $LazyTrees(T, \mathbf{e_t})$ return $\{p(c_j|\mathbf{e_t})\}$

T: a model with a set of leaves L
S_l: a set of labeled samples at a leaf l
$\mathbf{e_t}$: an unlabeled sample
$\{p(c_j|\mathbf{e_t})|c_j \in C\}$: a set of probability estimates of $\mathbf{e_t}$

Dispatch e_t into one leaf l according to its attributes
for each labeled sample $\mathbf{e_{train}} \in S_l$ **do**
  Calculate the distance d_{train} between $\mathbf{e_{train}}$ and $\mathbf{e_t}$, by utilizing Equation 1
  Calculate the sample weight w_{train}, in terms of its similarity to $\mathbf{e_t}$, by utilizing Equation 2
for each class value $c_j \in C$ **do**
  Use Equation 3 to compute $p(c_j|\mathbf{e_t})$
Return a set of probability estimates $\{p(c_j|\mathbf{e_t})\}$ for the unlabeled sample $\mathbf{e_t}$

3 Empirical Study

More details of empirical study can be found in [5]. We used 36 standard sample sets from the UCI repository [1] and conducted three groups of experiments in terms of ranking within the *Weka* [8] Platform. Table 1 lists the properties of the sample sets. Numeric attributes were handled by decision trees themselves. Missing values were processed using the mechanism in *Weka*, which replaced all missing values with the modes and means from the training set. Besides, due to the relatively high time complexity of *LazyTree*, we made a re-sampling within

Weka in sample set *Letter* and generated a new sample set named *Letter-2000*. The selection rate is 10%.

Table 1. Brief description of sample sets used in our experiments.

Data Set	Size	Classes	Missing	Numeric	Sample Set	Size	Classes	Missing	Numeric
anneal	898	6	Y	Y	ionosphere	351	2	N	Y
anneal.ORIG	898	6	Y	Y	iris	150	3	N	Y
audiology	226	24	Y	N	kr-vs-kp	3196	2	N	N
autos	205	7	Y	Y	labor	57	2	Y	Y
balance	625	3	N	Y	letter-2000	2000	26	N	Y
breast	286	2	Y	N	lymph	148	4	N	Y
breast-w	699	2	Y	N	mushroom	8124	2	Y	N
colic	368	2	Y	Y	p.-tumor	339	21	Y	N
colic.ORIG	368	2	Y	Y	segment	2310	7	N	Y
credit-a	690	2	Y	Y	sick	3772	2	Y	Y
credit-g	1000	2	N	Y	sonar	208	2	N	Y
diabetes	768	2	N	Y	soybean	683	19	Y	N
glass	214	7	N	Y	splice	3190	3	N	N
heart-c	303	5	Y	Y	vehicle	846	4	N	Y
heart-h	294	5	Y	Y	vote	435	2	Y	N
heart-s	270	2	N	Y	vowel	990	11	N	Y
hepatitis	155	2	Y	Y	waveform-5000	5000	3	N	Y
hypoth.	3772	4	Y	Y	zoo	101	7	N	Y

In the first group of our experiments, *LazyTree* was compared to C4.5 and its PET variants including C4.5-L (C4.5 with *Laplace* estimation), C4.5-M (C4.5 with m-Branch) and C4.5-LY (C4.5 with *Ling&Yan*'s algorithm). In the second group, we compared *LazyTree* with C4.4 and its PET variants, which contain C4.4-nLa (C4.4 without *Laplace* estimation), C4.4-M (C4.4 with m-Branch) and C4.4-LY (C4.4 with *Ling&Yan*'s algorithm). In the last group, we made a comparison between LazyTree-B (*LazyTree* with *bagging*) and C4.5-B (C4.5 with *bagging*) and C4.4-B (C4.4 with *bagging*). Multi-class AUC was calculated by M-measure[3]. The AUC value on each sample set was measured via a ten-fold cross validation ten times, and we performed two-tailed t-tests with a significantly different probability of 0.95 to compare our model with others. Now, our observations are highlighted as follows.

1. *LazyTree* achieves remarkably good performance in AUC among C4.5 and its variants. *LazyTree* performs significantly better than C4.5 (31 wins and 0 loss). As the results show, C4.5 variants can improve the AUC values of C4.5. However, *LazyTree* considerably outperforms these models in AUC. In addition, decision trees with m-Branch or with *Ling&Yan*'s algorithm can not generate multiple probabilities for the samples falling into the same leaf.
2. *LazyTree* is the best model among C4.4 and its variants in AUC. *LazyTree* significantly outperforms C4.4 (10 wins and 1 loss). Since C4.4 is the *state-of-art* decision tree model designed for yielding accurate ranking, this comparison provides strong evidence to the ranking performance of *LazyTree*. *LazyTree* also outperforms most of C4.4 variants in AUC.
3. LazyTree-B performs greatly better than C4.5-B and C4.4-B. C4.5-B has a good ranking performance compared with other typical models. However,

LazyTree-B produces better AUC values than C4.5-B (15 wins and 0 loss) and also performs better than C4.4-B (7 wins and 2 losses).

4. Besides having good performances on ranking, *LazyTree* also has better robustness and stability than other models. The average standard deviation of *LazyTree* in AUC is 4.37, the lowest among all models.

Generally speaking, *LazyTree* is a trade-off between the quality of probability-based ranking and the comprehensibility of results when selecting the best model.

4 Conclusion

In this paper, we analyzed that traditional decision trees have inherent defects in achieving precise ranking, and proposed to resolve those issues by representing the similarity between each sample at a leaf and an unlabeled sample. One key observation is that for a leaf, deploying a lazy probability estimator is an optimal alternative to produce a unique probability estimate for a specific sample, compared with directly using frequency-based probability estimation based on the leaf samples. Experiment results prove our expectation that *LazyTree* outperforms typical decision tree models in ranking quality. In our future work, other parameter-learning methods could be used to tune the probability estimation at a leaf. Additionally, we can find the right tree size for our model, i.e. based on C4.5 we use the *leave-one-out* technique to learn a weight for each sample at leaves, then continue fully splitting the tree, and at each leaf we normalize the weights and produce probability estimation of that leaf.

References

1. C. Blake and C.J. Merz. Uci repository of machine learning database.
2. P. A. Flach C. Ferri and J. Hernandez-Orallo. Improving the auc of probabilistic estimation trees. In *Proceedings of the 14th European Conference on Machine Learning (ECML2003)*. Springer, 2003.
3. D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45, 2001.
4. Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
5. H. Liang and Y. Yan. Lazy learning for improving ranking of decision trees. www.flydragontech.com/publications/2006/LazyLeaveTree_long.pdf, 2006.
6. C. X. Ling and R. J. Yan. Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning (ICML2003)*. Morgan Kaufmann, 2003.
7. F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(30), 2003.
8. I. H. Witten and E. Frank. *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.
9. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning (ICML2001)*. Springer, 2001.