Learning Conditional Independence Tree for Ranking

Jiang Su and Harry Zhang
Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca

Abstract

Accurate ranking is desired in many real-world data mining applications. Traditional learning algorithms, however, aim only at high classification accuracy. It has been observed that both traditional decision trees and naive Bayes produce good classification accuracy but poor probability estimates. In this paper, we use a new model, conditional independence tree (CITree), which is a combination of decision tree and naive Bayes and more suitable for ranking and more learnable in practice. We propose a novel algorithm for learning CITree for ranking, and the experiments show that the CITree algorithm outperforms the state-of-the-art decision tree learning algorithm C4.4 and naive Bayes significantly in yielding accurate rankings. Our work provides an effective data mining algorithm for applications in which an accurate ranking is required.

1 Introduction

In data mining, a classifier is built from a set of training examples with class labels and its performance is measured by its predictive accuracy (or error rate, 1 $-$ accuracy). Some classifiers can also produce the class probability estimates $p(c|E)$ that is the probability of an example E in the class c. This information is largely ignored, since the error rate considers only the class with the largest probability estimate. In some data mining applications, however, classification and error rate are not enough. For example, in direct marketing, we often need a ranking of customers in terms of their likelihood of buying.

If we are aiming at an accurate ranking based on the class probability, the area under the ROC (Receiver Operating Characteristics) curve [9, 6], or simply AUC, has been recently used as the performance measure. AUC compares the classifiers' performance cross the entire range of class distributions and error costs, and provides a good "summary" for comparing two classifiers. Hand and Till [1] show that, for binary classification, AUC is equivalent to the probability that a randomly chosen example of class $-$ will have a smaller estimated probability of belonging to class + than a randomly chosen example of class +. They present a simple approach to calculating the AUC of a classifier G below.

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}, \tag{1}$$

where n_0 and n_1 are the numbers of negative and positive examples respectively, and $S_0 = \sum r_i$, where r_i is the rank of i_{th} positive example in the ranking.

From Equation 1, it is clear that AUC is essentially a measure of the quality of a ranking. For example, the AUC of a ranking is 1 (the maximum value of AUC) if there is no positive example preceding a negative example.

If we are aiming at the accurate probability-based ranking, what is the performance of the traditional learning algorithms, such as decision trees and naive Bayes? In a decision tree, the class probability $p(c|E)$ is the fraction of the examples of class c in the leaf that E falls into. While decision trees perform quite well in classification, it is also found that their probability estimates are poor [5, 7]. Building decision trees with accurate probability estimates, called probability estimation trees (PETs), has received a great deal of attention recently [8]. Some researchers ascribe the poor probability estimates of decision trees to the decision tree learning algorithms. Thus, many techniques have been proposed to improve the learning algorithms in producing accurate probability estimates [8]. Provost and Domingos propose propose the following techniques to improve the AUC of C4.5 [8].

1. Smooth probability estimates by Laplace correction. Assume that there are p examples of the class at a leaf, N total examples, and C total classes. The frequency-based estimation calculates the estimated probability as $\frac{p}{N}$. The Laplace estimation calculates the estimated probability as $\frac{p+1}{N+C}$.

2. Turn off pruning. Provost and Domingos [8] show that pruning a large tree damages the probability estima-

tion. Thus, a simple strategy to improve the probability estimation is to build a large tree without pruning.

Provost and Domingos call the resulting algorithm C4.4. They compared C4.4 to C4.5 by empirical experiments, and found that C4.4 is a significant improvement over C4.5 with regard to AUC.

Ling and Yan also propose a method to improve the AUC of a decision tree [3]. They present a novel probability estimation algorithm, in which the class probability of an example is an average of the probability estimates from all leaves of the tree, instead of only using the leaf into which it falls. In other word, each leaf contributes to the class probability estimate of an example.

To our observation, however, the representation of decision trees also plays an important role. In a decision tree, the class probabilities of all the examples in the same leaf are equal. However, an accurate ranking needs that different examples have different probability. This is an obstacle in building an accurate PET, because two contradictory factors are in play at the same time. On one hand, traditional decision tree algorithms, such as C4.5, prefer a small tree. Thus, a leaf has more examples and the class probability estimates are more reliable. A small tree, however, has a small number of leaves, thus more examples will have the same class probability. That prevents the learning algorithm from building an accurate PET. On the other hand, if the tree is large, not only may the tree overfitting the training data, but the number of examples in each leaf is also small, and thus the probability estimates would not be accurate and reliable. Such a contradiction does exist in traditional decision trees.

A CITree [11] (Conditional Independence Tree) is a novel representation model, which uses a traditional decision tree to explicitly represent conditional independences among attributes and a naive Bayes on each leaf to represent the local distribution. Thus, a CITree represents a joint distribution among all attributes. CITree is different from NBTree [2] in that the structure of a CITree represents conditional independence; that is, given the attributes (path attributes) that occur on the path from the root to a leaf, all the other attributes (leaf attributes) are independent. Since conditional dependence can be relaxed by the tree structure, so the probability estimates given by leaf naive Bayes will be accurate. In the mean while, naive Bayes can assign different probability to each example, which could lead to an accurate ranking. In other words, its AUC should be high.

2 A Novel Algorithm for Learning CITree

We believe that CITree is a suitable model for building an accurate PET, and thus yields an accurate ranking. But in practice, learning the structure of an accurate CITree is intractable, just as learning an optimal decision tree. However, a good approximation of a CITree, which gives good estimates of class probabilities, is satisfiable in many data mining applications. If the structure of a CITree is sufficiently well, the probability estimates given by its leaf naive Bayes is also accurate, since the conditional dependences among attributes are relaxed by the structure of the CITree.

In building a CITree, we are looking for an attribute, given which all other attributes have the maximum independence. Thus, the key is to find such an attribute. The process of building a CITree could be also a greedy and recursive process, similar to building a decision tree. At each step, choose the “best” attribute as the root of the (sub)tree, split the associated data into disjoint subsets corresponding to the values of the attribute, and then recur this process for each subset until certain criteria are satisfied.

Notice as well, however, the difference between learning a CITree and learning a decision tree. The process of building a decision tree is guided by the purity of the (sub)dataset, measured by information gain. That is, we are looking for a sequence of attributes that leads to the maximum purity in all leaves of the tree. However, the process of building a CITree is guided by relaxing the conditional dependences among attributes. More precisely, we wish that, by choosing a sequences of attributes, all other attributes are independent. In practice, we intend to choose a set of attributes that make the local conditional independence among the rest of attributes true as much as possible. Thus, even though the impurity of a (sub)dataset is high, it could still be desired, as long as the conditional dependence is minimum. Thus, traditional decision tree learning algorithms are not directly suitable for learning CITrees.

One problem in learning a CITree is how to find the attribute (if it exists), given which all other attributes have the maximum conditional independence. We adopt a heuristic search process, in which we choose an attribute with the greatest improvement on the performance of the resulting CITree. Another reason for using this strategy is that, not all the conditional dependences will influence naive Bayes. Naive Bayes works well even when strong dependences exist. Thus a heuristic search guided by the performance of the CITree is suitable. More precisely, we try each possible attribute as the root at each step, evaluate the resulting tree, and choose the attribute that achieves the highest AUC. To avoid overfitting, the AUC of a leaf naive Bayes is conducted by a 5-fold cross validation.

Similar to C4.5, our learning algorithm has two separate steps: growing a tree and post-pruning. In growing a tree, each possible attribute is evaluated at each step, and the attribute that gives the most improvement in AUC is selected. The algorithm is depicted below.

Algorithm AUC-CITree ($\mathbf{T}$, $\mathbf{S}$, $\mathbf{A}$)

Input : CITree $\mathbf{T}$, a set $\mathbf{S}$ of labeled examples, a set of attributes $\mathbf{A}$

```
Output : a CITree.
    1. For all attributes $A$ in $\mathbf{A}$
        • Partition $\mathbf{S}$ into $\mathbf{S_1}, \cdots, \mathbf{S_k}$, each of which
          corresponds to a value of $A$.
        • Create a leaf naive Bayes for each $\mathbf{S_i}$.
        • Evaluate the resulting CITree in terms of
          AUC.
    2. Choose the attribute $A_{opt}$ with the highest AUC.
    3. For all values $a$ of $A_{opt}$
       CITree($\mathbf{T}_a$, $\mathbf{S}_a$, $\mathbf{A} - \{A_{opt}\}$).
       Add $\mathbf{T}_a$ as a child of $\mathbf{T}$.
    4. Return $\mathbf{T}$.
```

Notice that in the AUC-CITree algorithm described above, we grow a tree as large as possible until we are out of data or attributes, and then start a post-pruning process as following:

Apply pruning based on the AUC of leaf naive Bayes, in which the children of a node are removed only if the resulting pruned tree (making it a leaf node and deploying a naive Bayes at it) performs no worse than the original tree. Notice the AUC for the children of a node is computed using instances from all the children.

Our AUC-CITree algorithm is different from the NBTree algorithm [2] in several aspects:

1. Our AUC-CITree algorithm is based on AUC, instead of accuracy.

2. Our algorithm is less greedy than the NBTree algorithm, in which if the accuracy improvement is less than 5%, stop growing the tree. We always choose the best attribute at each step, even no improvement has been achieved.

3. The AUC-CITree algorithm adopts the post-pruning strategy, rather than early stop.

3 Experiments

We conduct experiments to compare our algorithm CITree with C4.4 and naive Bayes. Notice that the implementation of naive Bayes and C4.4 is from Weka [10], C4.4 is J48 in Weka with Laplace correction and turning off pruning. All algorithms are evaluated by using 29 datasets from the UCI repository [4]. Considering that large data sets seem more important in data mining, we also choose large data sets to show CITree's advantage in practical environment. In our experiment, multi-class AUC has been calculated by M-measure [1], and the average AUC on each data set is obtained by using 10-fold stratified cross validation 10 times. Some details in our implementation are summarized below.

1. In Step 1 of the AUC-CITree algorithm, we adopt an inner 5-fold cross-validation on the training data S which fall into a leaf to evaluate the AUC for a leaf naive Bayes, and choose the best one. For example, if a attribute has 4 attribute value which will result four leaf naive Bayes, the inner 5-fold cross-validations will be run in four leafs. Note that, we compute AUC by putting the instances from all the leaves together rather than computing the AUC for each leaf separately.

2. In Step 1, when the instances falling into a leaf are less than 5, all the instances will be assigned the random probability. This is a strategy to avoid overfitting.

3. Numeric attributes are discretized using ten-bin discretization implemented in Weka [10]. Missing value are also replaced by ReplaceMissingValues class in Weka.

Table 1 shows the average AUC obtained by the three algorithms. The comparison of the three algorithms on these datasets, in which a paired t-test with a confidence of 95% has been used, are summarized in Table 2. Our observations are summarized below.

1. The AUC-CITree algorithm outperforms naive Bayes significantly in terms of AUC: It wins in 10 datasets, ties in 19 datasets and loses in 0 dataset. The average AUC for CITree is 92.66%, higher than the average AUC 90.91% of naive Bayes.

2. The CITree algorithm also outperforms C4.4 significantly in terms of AUC: It wins in 16 datasets, ties in 11 datasets and loses in 2 datasets. The average AUC for decision trees is 89.02%, lower than CITree's.

3. The sizes of CITrees (not shown in Table 1) are significantly smaller than the sizes of decision trees over most of these datasets. Here the size of a tree is the number of nodes. The average tree size for CITrees is 64, and for C4.4 it is 610.

4 Conclusions

In this paper, we study using an extended decision tree model CITree for ranking, and present and implement a novel algorithm AUC-CITree which is based on AUC to build a CITree for ranking by exploring the conditional independence among attributes, different from traditional decision tree learning algorithms. Our experiments show that the AUC-CITree algorithm performs better than C4.4 and naive Bayes significantly in terms of ranking, measured by AUC. Our work provides an effective data mining algorithm for applications in which an accurate ranking is required.

Table 1. Experimental results on AUC.

Dataset	AUC-CITree	NB	C4.4
Abalone	79.25 ± 1.6	78.73 ± 1.58	77.86 ± 1.8
Adult	90.44 ± 0.3	90.1 ± 0.33	86.64 ± 0.42
Anneal	96.33 ± 0.61	95.9 ± 1.3	93.8 ± 2.9
B.-scale	84.46 ± 4.1	84.46 ± 4.1	59.4 ± 5.53
Wis.-breast	99.18 ± 1	99.26 ± 0.82	97.83 ± 1.54
Car	98.41 ± 1.47	92.06 ± 2.55	95.83 ± 1.74
Horse-colic	85.04 ± 7.59	84.04 ± 5.38	82.01 ± 5.25
Crd-rating	91.7 ± 3.85	92.05 ± 3.46	87.46 ± 3.46
G.-credit	78.25 ± 5.27	79.27 ± 4.74	68.59 ± 4.81
Pima	82.31 ± 5.17	82.31 ± 5.17	73.27 ± 4.84
Hypothy.	87.35 ± 7.04	87.37 ± 8.52	82.88 ± 8.95
Ionosphere	96.33 ± 2.7	93.61 ± 3.36	91.95 ± 4.31
Iris	98.58 ± 2.67	98.58 ± 2.67	97.42 ± 2.1
Kr-vs-kp	99.8 ± 0.16	95.17 ± 1.29	99.96 ± 0.05
Letter	98.57 ± 0.14	96.86 ± 0.24	95.27 ± 0.41
Mushroom	100 ± 0	99.79 ± 0.04	100 ± 0
Nursery	98.68 ± 3.27	98.91 ± 1.02	96.89 ± 6.04
Pendigits	99.77 ± 0.07	98.7 ± 0.19	98.23 ± 0.28
Satellite	66.44 ± 27.02	66.61 ± 17.63	83.31 ± 14.08
Segment	99.06 ± 0.34	98.51 ± 0.46	98.95 ± 0.27
Sick	98.53 ± 1.25	95.91 ± 2.35	99.07 ± 0.42
Soybean	99.55 ± 0.64	99.53 ± 0.6	91.35 ± 2.7
Splice	99.36 ± 0.32	99.41 ± 0.22	97.81 ± 0.55
Tic-tac-toe	90.33 ± 1.31	73.91 ± 2.18	93.99 ± 2.88
Vehicle	87.06 ± 2.58	80.81 ± 3.51	86.5 ± 2.61
Vote	98.64 ± 1.15	96.56 ± 2.09	97.46 ± 2.46
Vowel	99.29 ± 0.5	95.81 ± 0.84	91.01 ± 2.3
Waveform	95.27 ± 0.58	95.27 ± 0.58	80.44 ± 1.18
Yeast	89.2 ± 1.53	86.91 ± 1.91	76.35 ± 5.3
Average	92.66	90.91	89.02

Table 2. Summary of the experimental results. An entry w-t-l means that the algorithm at the corresponding row wins in w datasets, ties in t datasets, and loses in l datasets, compared to the algorithm at the corresponding column.

	NB	C4.4
AUC-CITree	10-19-0	16-11-2
NB		14-8-7

CITree can be viewed as a bridge between probabilistic models, such as Bayesian networks, and non-parametric models, such as decision trees [11]. However, a more effective CITree learning algorithm is desired. Currently, our learning algorithm is based on heuristics, similar to the NBTree algorithm [2]. We believe that if a better learning algorithm is found, a CITree will benefit much from its structure, and thus will be a good model for many data mining applications.

References

[1] D. J. Hand and R. J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[2] R. 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*, pages 202–207. AAAI Press, 1996.

[3] C. X. Ling and R. J. Yan. Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning*, pages 480–487. Morgan Kaufmann, 2003.

[4] C. Merz, P. Murphy, and D. Aha. UCI repository of machine learning databases. In *Dept of ICS, University of California, Irvine*. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1997.

[5] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proceedings of the 11th International conference on Machine Learning*, pages 217–225. Morgan Kaufmann, 1994.

[6] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48. AAAI Press, 1997.

[7] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, 1998.

[8] F. J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[9] J. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293, 1988.

[10] I. H. Witten and E. Frank. *Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

[11] H. Zhang and J. Su. Conditional independence trees. In *to appear in Proceedings of the 15th European Conference on Machine Learning*. Springer, 2004.