Decision Tree for Data Uncertainty with Pruning

Jayesh R Solanki[1] Sonal P Rami[2]
[1]M.E. student [2]Assistant Professor
[1,2] KIRC, Kalol, Gujarat-India

Abstract— Decision Tree is a widely used data classification technique. This paper proposes a decision tree based classification method on uncertain data. Data uncertainty is common in emerging applications, such as sensor networks, moving object databases, medical and biological bases. Data uncertainty can be caused by various factors including measure- ments precision limitation, outdated sour ces, sensor errors, and network latency and transmission problems. In this paper, we enhance the traditional decision tree algorithms and extend measures, including entropy and information gain, considering the uncertain data interval and probability distribution function. Our algorithm can handle both certain and uncertain datasets. The experiments demonstrate the utility and robust ness of the proposed algorithm as well as its satisfactory prediction accuracy. We propose a series of pruning techniques that can greatly improve the efficiency of the construction of decision trees.

I. INTRODUCTION

Decision trees are a simple yet widely used method for classification and predictive modeling. A decision tree partitions data into smaller segments called terminal nodes.

Each terminal node is assigned a class label The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to differentiate each record from others that have different characteristics

This process terminates when the subsets cannot be partitioned further. In this paper we study how to control data uncertainty by using decision tree classification. A simple way to handle data uncertainty is to abstract probability distribution by means and variances. This approach is called Averaging. Here we have another approach also that is called Distribution based approach in this complete information is utilized. In this paper the Decision tree can handle both numerical and categorical data, while many other techniques are usually specialized in analyzing datasets that have only one type data. Our goals are 1) To invent an algorithm for building decision tree for uncertain data using Distribution based approach.2) To check whether the Distribution approach could lead to higher accuracy when compared with averaging 3) Pruning techniques are derived to significantly improve the computational efficiency of Distribution based algorithm. Data uncertainty arises basically in many applications due to various reasons. We have three categories here: measurement errors, data staleness, and repeated measurements.

A. *Measurement errors:*

Data obtained from measurements by physical devices are often not precise due to measurement errors. For example, thermometer measures body temperature by measuring the temperature of the ear drum via an infrared sensor. It is having some calibration error.

B. *Data staleness:*

In some of the applications, data values are no longer fresh that means the values are continuously changing. Example for this is location based tracking system.

C. *Repeated measurements:*

Basically the most common uncertainty comes from repeated measurements. Example for this is a patient's body temperature could be taken multiple times during a day

II. RELATED WORKS

Decision trees are one of the most important aspects for Decision making. Classification is one of the most widespread data mining problems found in real life. There has been a growing interest in uncertain data mining. The well known k means clustering algorithm has been extended to the UK

means algorithm and various pruning techniques have been proposed, Decision tree classification with missing data has been addressed for decades in the form of missing values. In C4.5, these are handled by using fractional tuples. In this work, we adopt the technique of fractional tuple for splitting tuples into subsets when the domain of its pdf spans across the split point. However, handling data values represented as pdf's is unprecedented. we give a distribution telling how likely it belongs to each class. Building a decision tree on tuples with numerical, point valued data is computationally demanding. Finding the best split point is computationally expensive. To improve efficiency, many techniques have been proposed to reduce the number of candidate split points. Our work can be considered an extension of these optimization techniques

III. ALGORITHM

In this section, we discuss two approaches for handling uncertain data. The first approach, called Averaging transforms an uncertain dataset to a point valued one by replacing each pdf with its mean value. A decision tree can then be built by applying a traditional tree construction algorithm. To exploit the full information carried by the pdf's, our second approach, called Distribution based, considers all the sample points that constitute each pdf

A. *Averaging*

We simply replace each pdf with its expected value, thus effectively converting the data tuples to point-valued tuples. This reduces the problem back to that for point valued data, and hence C4.5 [3] (with pre-pruning and post pruning) can be reused. We call this approach AVG (for averaging).AVG is a greedy algorithm that builds a tree top-down. When processing a node, we examine a set of tuples S. The algorithm starts with the root node and with S being the set of all training tuples. At each node n, we first check if all the tuples in S have the same class label c. If so, we

make n a leaf node and set Pn(c) = 1, Pn(c′) = 0 for all c′≠ c. Otherwise, we select an attribute Ajn and a split point zn and divide the tuples into two subsets: left and right. All tuples with vi,jn≤zn are put in the left subset L; the rest go to the right subset R. If either L or R is empty (even after exhausting all possible choices of Ajn and zn), it is impossible to use the available attributes to further discern the tuples in S. In that case, we make n a leaf node. Moreover, the population of the tuples in S for each class label induces the probability distribution Pn. In particular, for each class label C, we assign to Pn(c) the fraction of tuples in S that are labeled c. If neither L nor R is empty, we make n an internal node and create child nodes for it. We recursively invoke the algorithm on the left child and the right child, passing to them the sets L and R, respectively.

To build a good decision tree, the choice of Ajn and Zn is crucial. At this point, we may assume that this selection is performed by a black box algorithm Best Split, which takes a set of tuples as parameter, and returns the best choice of attribute and split point for those tuples. We will examine this black box in details. Typically, Best Split is designed to select the attribute and split point that minimizes the degree of dispersion. The degree of dispersion can be measured in many ways, such as entropy (from information theory) or Gini index [4]. The choice of dispersion function affects the structure of the resulting decision tree. In this paper we assume that entropy is used as the measure since it is predominantly used for building decision trees. The minimizations are taken over the set of all possible attributes Aj (j = 1… k), considering all possible split points in dom (Aj). Given a set S = {t1… tm} of m tuples with point values, there are only m-1 ways to partition S into two non-empty L and R sets. For each attribute Aj , the split points to consider are given by the set of values of the tuples under attribute Aj , i.e., {v1,j ,….,vm,j}. Among these values, all but the largest one give valid split points.

B. Distubution-based Approach

For finding uncertain data, we adopt the same decision tree building framework as described above for handling point data. After an attribute Ajn and a split point zn has been chosen for a node n, we have to split the set of tuples S into two subsets L and R. The major difference from the point-data case lies in the way the set S is split. Recall that the pdf of a tuple ti ϵ S under attribute Ajn spans the interval [ai,jn, bi,jn]. If bi,jn ≤zn, the pdf of ti lies completely on the left of the split point and thus ti is assigned to L. Similarly, we assign ti to R if zn < ai,jn. If the pdf properly contains the split point, i.e., Ai,jn≤zn < bi,jn, we split ti into two fractional tuples tL and tR in the same way as described in previous Section and add them to L and R, respectively.The key to building a good decision tree is a good choice of an attribute Ajn and a split point zn for each node n. With uncertain data, however, the number of choices of a split point given an attribute is not limited to m-1 point values. This is because a tuple ti's pdf spans a continuous range [ai,j , bi,j]. Moving the split point from ai,j to bi,j continuously changes the probability pL =∫Znai,jn Fi,jn(x) dx (and likewise for pR). This changes the fractional tuples tL and tR, and thus changes the resulting tree. If we model a pdf [9,13] by s sample values, we are approximating the pdf by a discrete distribution of s points. In this case, as the split point moves from one end-point ai,j to another end-point bi,j of the interval, the probability pL changes in s steps. With m tuples, there are in total ms sample points. So, there are at most ms-1 possible split points to consider. Considering all k attributes, to determine the best (attribute, split-point) pair thus require us to examine k(ms -1) combinations of attributes and split points. Comparing to AVG, UDT is s time more expensive. Note that splitting a tuple into two fractional tuples involves a calculation of the probability pL, which requires an integration. We remark that by storing the pdf in the form of a cumulative distribution, the integration can be done by simply subtracting two cumulative probabilities. Let us re-examine the example tuples in Table I to see how the distribution-based algorithm can improve classification accuracy. By taking into account the probability distribution, UDT builds the tree shown in Figure 3 before pre-pruning and post-pruning are applied. This tree is much more elaborate than the tree shown in Figure 1(a), because we are using more information and hence there are more choices of split points. The tree in Figure 3 turns out to have 100% classification accuracy! After post-pruning, we get the tree in Figure 1(b). Now, let us use the 6 tuples in Table I as testing tuples4 to test the tree in Figure 1(b). For instance, the classification result of tuple 3 gives P(A) = 5/8*0.80 + 3/8 * 0.212 = 0.5795 and P(B) = 5/8 * 0.20 + 3/8 * 0.788 = 0.4205. Since the probability for A is higher, we conclude that tuple 3 belongs to class A. All the other tuples are handled similarly, using the label of the highest probability as the final classification result. It turns out that all 6 tuples are classified correctly. This hand-crafted example thus illustrates that by considering probability distributions rather than just expected values, we can potentially build a more accurate decision tree.

C. Algorithm for Both Certain and Uncertain Data

Input: The training dataset D; the set of candidate attributes att-list
Output: An uncertain decision tree Begin

1: create a node N;
2: if (D are all of the same class, C) then
3: return N as a leaf node labeled with the class C;
4: else if (attribute-list is empty) then
5: return N as a leaf node labeled with the highest weight class in D;
6: end if;
7: select a test-attribute with the highest probabilistic information gain ratio to label node N;
8: if (test-attribute is numeric or uncertain numeric or categorical data for various data uncertain) then
9: binary split the data from the selected position y;
10: for (each instance Rj) do
11: if (test-attribute ≤y) then
12: put it into Dl with weight Rj .w;
13: else if (test-attribute > y) then
14: put it into Dr with weight Rj .w;
15: else
16: put it into Dl with weight Rj .w *∫yx1f(x)dx;
17: put it into Dr with weight Rj .w *∫x2yf(x)dx;
18: end if;
19: end for;
20: else

21: for (each value ai(i = 1, . . . , n) of the attribute) do
22: grow a branch Di for it;
23: end for;
24: for (each instance Rj) do
25: if (test-attribute is uncertain) then
26: put it into Di with Rj .ai.w*Rj .w weight;
27: else
28: put it into a certain Di with weight Rj .w;
29: end if
30: end for;
31: end if;
32: for each Di do
33: attach the node returned by DTU (Di, att-list);
34: end for;
End

IV. DISSCUSSIONS

A. *Uncertain model*

In our discussion, uncertainty models of attributes have been assumed known by some external means. In practice, finding a good model is an application-dependent endeavor. For example, manufacturers of some measuring instruments do specify in instruction manuals the error of the devices, which can be used as a source of information for modeling error distributions. In some other cases, repeated measurements can be taken and the resulting histogram can be used to approximate the pdf .In the case of random noise, for example, one could fit a Gaussian distribution5 using the sample mean and variance, thanks to the Central Limit Theorem. During the search for datasets appropriate for our experiments, we have hit a big obstacle: There are few datasets with complete uncertainty information. Although many datasets with numerical attributes have been collected via repeated measurements, very often the raw data has already been processed and replaced by aggregate values, such as the mean. The pdf information is thus not available to us. One example is the ―Breast Cancer‖ dataset from the UCI repository. This dataset actually contains 10 uncertain numerical features collected over an unspecified number of repeated measurements. However, when the dataset is deposited into the repository, each of these 10 features is replaced by 3 attribute values, giving the mean, the standard score and the mean of the three largest measured values. With these 3 aggregate values, we are unable to recover the distribution of each feature. Even modeling a Gaussian distribution is impossible. These 3 aggregate values are insufficient for us to estimate the variance. Had the people preparing this dataset provided the raw measured values, we would be able to model the pdf's from these values directly, instead of injecting synthetic uncertainty and repeating this for different parameter values for w. Now that we have established in this work that using uncertainty information modeled by pdf's can help us construct more accurate classifiers, it is highly advisable that data collectors preserve and provide complete raw data, instead of a few aggregate values, given that storage is nowadays very affordable.

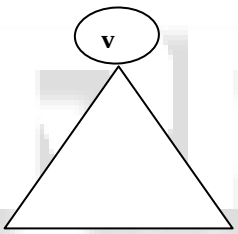
B. *Handling categorical attributes*

We have been focusing on processing uncertain numerical attributes in this paper. How about uncertain categorical attributes? Like their numerical counterparts, uncertainty can arise in categorical attributes due to ambiguities, data staleness, and repeated measurements. For example, to cluster users based on access logs of HTTP proxy servers using (besides other attributes such as age) the top-level domain names (e.g. .com, .edu, .org, .jp, .de, .ca) as an attribute, we obtain repeated measurements of this attribute from the multiple log entries generated by each user. The multiple values collected from these entries form a discrete distribution, which naturally describes the uncertainty embedded in this categorical attribute. The colour of a traffic light signal, which is green at the time of recording, could have changed to yellow or even red in 5 seconds, with probabilities following the programmed pattern of the signal. This is an example of uncertainty arising from data staleness. Colours of flowers recorded in a survey may divide human- visible colours into a number of categories, which may overlap with one another. Such ambiguities could be recorded as a distribution, e.g. 80% yellow and 20% pink. In all these cases, using a distribution to record the possible values (wit corresponding probabilities) is a richer representation than merely recording the most likely value.

V. PRUNING

A. *Reduced Error Pruning*

Use pruning set to estimate accuracy of sub-tree and accuracy at individual nodes.



Where T be a sub tree rooted at node v.
V= misclassification in T – misclassification v
Sum the error over entire sub tree.
Calculate error on same example if converted to a leaf with majority class lable.
Prune node with highest reduction in error.
Repeat until error no longer reduced.

B. *Minimum error pruning*

The following equation is used to predict the expected error rate of pruning at node t.

$$E(t) = \frac{n_t - n_{t,c} + k - 1}{n_t + k}$$

Where, k= no of class

nt =no of examples in node t

nt .c =no of example

C. *Cost – complexity pruning*

In this method calculated error cost of a node.
It finds error complexity at each node.
The error cost of the node is calculated using following equation.

R(t) = r(t) * p(t)

Where,

r(t) is error cost of the node is calculated using following equation.

$$r(t) = \frac{\text{no of examples misclassified in node}}{\text{no of all example in node}}$$

p(t) is probability on occurrence on a node.

$$p(t) = \frac{\text{no of example in node}}{\text{no of total examples}}$$

D. Pessimistic error

$n'(t) = e(t) + (1/2)$ ……..(1)

$n'(Tt) = e(Tt) + (NT/2)$……(2)

Equation (1) is the number of misclassifications for node t

Equation (2) is the number of misclassifications for sub tree T.

Where:

NT is the number of leaves for sub tree T,

e(t) is the number of misclassifications at node t

e(Tt) is the number of misclassifications for sub tree T.

The 1/2 in the equation (1) and (2) is a constant which indicates the contribution of a leaf to the complexity of the tree.

VI. CONCLUSION

In this work, we have seen the uncertain data is handled through "Averaging" by using means and variances. But in "Distribution-based" the accuracy of a uncertain data is detected through decision trees. Decision trees calculate the entropy measure and enhance the information gain for better accuracy. Several procedures and algorithm handles data uncertainty. We exploit data uncertainty that leads to decision trees with remarkably higher accuracies.

REFERENCES

[1] Jiawei Han, MichelineKamber, "Data Mining Concepts and Techniques", pp. 279-328, 2001.
[2] Anuja Priyam, Rahul gupta, Saurabh srivastava "comparative Analysis of decision tree Classification Algorithms" (2013).
[3] J.R. Quinlan, "Induction of Decision Trees", Machine Learning 1(1986) pp.81-106.
[4] SamDrazin and Matt Montag,"Decision Tree Analysis using Weka", Machine Learning-Project II, University of Miami.
[5] Dipti D. Patil, V. M. Wadhai, J. A. Gokhale, "Evaluation of Decision Tree Pruning Algorithms for Complexity and Classification Accuracy", IJCSE, volume-II.
[6] John Mingers."An Empirical Comparison of Pruning Methods for Decision Tree Induction" Machine Learning, 4, 227-243 (1989).
[7] Biao Qin, Yuni Xia, Fang Li. "DTU: A Decision Tree for Uncertain Data".
[8] Biao Qin, Yuni Xia, Fang Li. "DTU: A Decision Tree for Uncertain Data" Full paper.
[9] Tom. M. Mitchell, "Machine Learning", McGraw-Hill Publications, 1997
[10] J. Quinlan," Simplifying decision trees", Int. J. Human-Computer Studies.
[11] J.R. Quinlan, "C4.5: programs for Machine Learning", Morgan Kaufmann, New York,1993.
[12] Xiaoping pang, Haoran Guo and Jianmin pang" Performance analysis between different decision tree for uncertain data" IEEE 2012
[13] Smith Tsang, Ben Kao, Kevin Y. Yip, Wai-Shing Hok, Sau Dan Lee. "Decision Trees for Uncertain Data."
[14] Krishna Mohan, Surekha Alokam , MHM Krishna Prasad , "An Efficient Decision Tree For Uncertain Data ", International Journal of Engineering Research and Applications (IJERA) , Vol. 2, Issue 3, May-Jun 2012
[15] M. Suresh Krishna Reddy, R. Jayasree," Extending Decision Tree Clasifiers For Uncertain Data" International Journal Of Engineering Science & Advanced Technology (IJESAT), Volume-2, Issue-4, 1030 – 1034.

