IOSR Journal of Engineering (IOSRJEN)
ISSN: 2250-3021 Volume 2, Issue 6 (June 2012), PP 41-49
www.iosrjen.org

Improving Computational efficiency of Tentative Data set

Manobendu Kesari Jena[1], Tapaswini Nayak[2] ,Dr.Maya Nayak[3],
Ashok Kumar Panda[4]

[1](M.Tech(Scolar),OEC Engg. College ,IACSIT Member)[2](Asst.Professor,Centurion University)
[3](Dean,Orissa Engg College ,BBSR) [4](Dean(ES & A),MITS Engg College,Rayagada)

ABSTRACT : - *The aim of data mining is to find novel, interesting and useful patterns from data using algorithms that will do it in a way that is more computational efficient than previous method. Current data mining practice is very much driven by practical computational concerns. In focusing almost exclusively on computational issues; it is easy to forget that statistics is in fact a core component. The Conventional decision tree classifies the data whose values are known and particular, but in this paper such classifiers handle data with tentative information. The value vagueness arises in many applications during the data collection process. Example sources of vagueness include currency conversion(Rupees to Dollar), role of packet data transfer, Daily weather report , protein design, Flight reservation, production company data due to fluctuation in raw material. With vagueness, the value of a data item is often represented not by one single value, but by multiple values forming a likelihood distribution. Rather than abstracting tentative data by statistical derivatives (such as mean and median), we discover that the accuracy of a decision tree classifier can be much improved if the "complete information" of a data item (taking into account the probability density function) is utilized .We extend conventional decision tree building algorithms to handle data tuples with tentative values and make a comparison between average based approach and distribution based approach. Since processing probability density function's is computationally more costly than processing single values (e.g., averages), decision tree construction on tentative data is more CPU demanding than that for certain data. To handle this problem, we propose a series of pruning techniques that can greatly improve edifice efficiency.*

Keywords – *data mining, probability density function, tentative data, decision tree classifier*

I. INTRODUCTION

Classification,which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications. Example include Detecting spam email messages based upon the message header and content, categorizing cells as malignant or benign based upon the results of MRI scans and classifying galaxies based upon their shapes.The classification of large datasets is an important problem in data mining[1] .For a database with a number of records and for a set of classes such that each record belongs to one of the given classes, the problem of classification is to decide the class to which a given record belongs. The classification problem is also concerned with generating a description or a model for each class from the given dataset.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

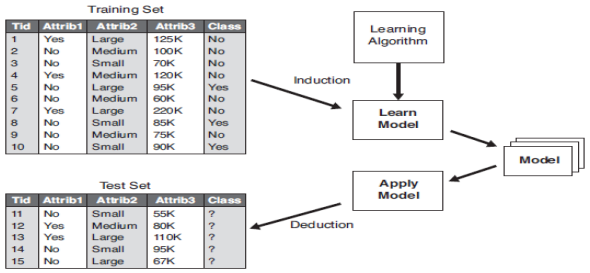
Fig1:shows the attributes presented are mostly discrete

Here we are concerned with a given type of classification called supervised classification. In supervised classification we have a training dataset of records and for each record of this set, the classification process attempts to generate the description of classes, and these descriptions help to classify the unknown records.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
gila monster	cold-blooded	scales	no	no	no	yes	yes	?

Fig2:Class label is unknown.

In addition to the training set, we can also have a test dataset which is used to determine the effectiveness of a classification. There are several approaches of supervised classifications. Decision trees are especially attractive in the data mining environment as they represent rules. Rules can also be extracted from decision trees easily. Many algorithms, such as SLIQ,ID3[2] and C4.5[3],SPRINT have been devised for decision tree construction. In recent years many new techniques for collecting data have resulted in an increase in the availability of uncertain data



While many applications lead to data which contains errors, we refer to *tentative data sets* as those in which the level of vagueness can be quantified in some way. Data uncertainty arises naturally in many applications due to various reasons. Some examples of applications which create uncertain data are as data staleness, measurement error, and repeated measurements.

i)Data staleness: In some applications, data values are continuously changing and recorded information is always stale. One example is currency trading. The currency exchange rate is the rate at which one currency can be exchanged for another. It is always quoted in pairs like the EUR/USD (the Euro and the US Dollar). Exchange rates fluctuate based on economic factors like inflation, industrial production and geopolitical events. The whereabouts of currency trading can only be approximated by imposing an uncertainty model on its last reported value. A typical uncertainty model requires knowledge about the fluctuating speed of currency.whetheher its movement is restricted or unrestricted. Typically a 2D bounded probability density function is defined over a bounded region to model such uncertainity.±

ii) Measurement Error: Data obtained from measurements by physical devices are often imprecise due to measurement errors.As an example a tympanic thermometer measures body temperature by measuring the temperature of the ear drum via an infrared sensor. A typical ear thermometer has a quoted calibration error of ±0.2°C,which is about 6.7% of the normal range of operation, noting that the human body temperature ranges from 37°C (normal) and to 40 C (severe fever). Compound that with other factors such as placement

and technique , measurement error can be very high . For example it is reported in [5] that about 24% of measurements are off by more than 0.5℃ ,or about 17% of the operational range. Another source of error is quantization errors introduced by the digitization process. In many applications such as the tracking of mobile objects, the *future trajectory* of the objects is modeled by forecasting techniques. Small errors in current readings can get magnified over the forecast into the distant future of the trajectory.

This is frequently encountered in cosmological applications when one models the probabilityof encounters with Near-Earth-Objects (NEOs). Errors in forecasting are also encountered in non-spatial applications such as electronic commerce.Such errors can be properly handeled by assuming an appropriate error model such as a Gaussian error distribution for random noise or a uniform error distribution for quantization errors.

iii)Repeated measurement: The most common source of uncertainity comes from repeated measurements. For example a patient's body temperature could be taken multiple times during a day; an anemometer could record wind speed once every minute; the space shuttle has a large number of heat sensors installed all over its surface.When we inquire about a patients temperature or wind speed or temperature of a certain section of the

suttle,which value shall we use? or it would better to utilize all the information by considering the distribution given by the collected data values?

An source of vagueness comes from the limitation of the data collection process. For example, a survey may ask a question like, "How many hours of Internet do you surf each day?"A typical respondent would not reply with an exact precise answer. Rather, a range (e.g. "2-3 hours") is usually replied, possibly because the respondent is not so sure about the answer himself. In this example, the survey can restrict an answer to fall into a few pre-set categories (such as "2-4 hours","4-7 hours", etc). However, this restriction unnecessarily limits the respondents' choices and adds noise to the data.

From the above examples, we see that in many applications, information cannot be ideally represented by point data. More often, a value is best captured by a range possibly with a probability density function.Our concept of uncertainty refers to such ranges of values. Again, our goal is to investigate how decision trees are built over uncertain (range) data. Our contributions include:

1) A basic algorithm for constructing decision trees out of uncertain datasets.2) A study comparing the classification accuracy achieved by the Averaging approach and the Distribution-based approach.

II. RELATED WORK

In recent years, uncertain data has become ubiquitous because of new technologies for collecting data which can only measure and collect the data in an imprecise way. Furthermore, many technologies such as privacy-preserving data mining create data which is inherently tentative in nature. As a result there is a need for tools and techniques for mining and managing uncertain data. Data vagueness has been broadly classified as existential uncertainty and value uncertainty. Existential uncertainty appears when it is uncertain whether an object or a data tuple exists. For example, a data tuple in a relational database could be associated with a probability that represents the confidence of its presence [5]."Probabilistic databases" have been applied to semi-structured data and XML[6],[7]. Value uncertainty, on the other hand,appears when a tuple is known to exist, but its values are not known precisely. A data item with value uncertainty is usually represented by a pdf over a finite and bounded region of possible values[8], [9]. One well-studied topic on value uncertainty is "imprecise queries processing". The answer to such a query is associated with a probabilistic guarantee on its correctness. For example, indexing solutions for range queries on uncertain data[10], solutions for aggregate queries[11] such as nearest neighbour queries, and solutions for imprecise location-dependent queries[8] have been proposed.

There has been a growing interest in uncertain data mining.In [9], the well-known k-means clustering algorithm is extended to the UK-means algorithm for clustering uncertain data. As we have explained, data uncertainty is usually captured by pdf's, which are generally represented by sets of sample values. Mining uncertain data is therefore computationally costly due to information explosion (sets of samples vs. single values). To improve the performance of UK-means, pruning techniques have been proposed. Examples include min-maxdist pruning[12] and CK-means[13]. Apart from studies in partition-based uncertain data clustering, other directions in uncertain data mining include density-based clustering(e.g.,FDBSCAN[14]), frequent itemsetmining[15] and density based classification[16]. Density-based classification requires that the joint probability distribution of the data attributesbe known. In [16], each data point is given an error model.Upon testing, each test tuple is a point-valued data. These are very different from our data model, as we do not require the knowledge of the joint probability distribution of the data attributes. Each attribute is handled independently and may have its own error model. Further, the test tuples, like the training tuples, may contain uncertainty in our model.

Decision tree classification on uncertain data has been addressed for decades in the form of missing values[2], [3].Missing values appear when some attribute values are not available during data collection or due to data entry errors.Solutions include approximating missing values with the majority value or inferring the missing value (either by exact or probabilistic values) using a classifier on the attribute (e.g.,ordered attribute tree[17] and probabilistic attribute tree[18]).In C4.5[3] and probabilistic decision trees[19], missing values in training data are handled by using fractional tuples. During testing, each missing value is replaced by multiple values with probabilities based on the training tuples, thus allowing probabilistic classification results. 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.We have also adopted the idea of probabilistic classification results. We do not directly address the problem of handling missing values. Rather, we tackle the problem of handling data uncertainty in a more general form. Our techniques are general enough for the existing missing-value handling methods to be encapsulated naturally into our framework. Based on the previously described approaches, a simple method of "filling in" the missing values could be adopted to handle the missing values, taking advantage of the capability of handling arbitrary pdf's in our approach. We can take the average of the pdf of the attribute in question over the tuples where the value is present. The result is a pdf, which can be used as a

"guess" distribution of the attribute's value in the missing tuples. Then, we can proceed with decision tree construction.

Building a decision tree on tuples with numerical, point valued data is computationally demanding [20]. A numerical attribute usually has a possibly infinite domain of real or integral numbers, inducing a large search space for the best "split point". Given a set of n training tuples with a numerical. attribute, there are as many as n - 1 binary split points or ways to partition the set of tuples into two non-empty groups. Finding the best split point is thus computationally expensive. To improve efficiency, many techniques have been proposed to reduce the number of candidate split points[21],[20], [22]. These techniques utilise the convex property of well-known evaluation functions like Information Gain[2] and Gini Index[23]. For the evaluation function TSE (Training Set Error), which is convex but not strictly convex, one only needs to consider the "alternation points" as candidate split points.[24] An alternation point is a point at which the ranking of the classes (according to frequency) changes.

III. PROBLEM DEFINITION

This section formally defines the problem of decision-tree classification on uncertain data. We first discuss traditional decision trees briefly. Then, we discuss how data tuples with uncertainty are handled.

A. Traditional Decision Trees

In our model, a dataset consists of d training tuples,$\{t_1,t_2,....,\ t_d\}$,and k numerical (real-valued) feature attributes,$A_1,\ldots\ldots A_k$. The domain of attribute A_j is $dom(Aj)$.Each tuple t_i is associated with a feature vector $V_i = (v_{i,1},\ v_{i,2},\ .,.,.,\ .\ v_{i,k})$ and a class label c_i, where $v_{i,j} \in dom(A_j)$ and $c_i \in C$, the set of all class labels. The classification problem is to construct a model M that maps each feature vector $(v_{x,1},\ldots..\ v_{x,k})$ to a probability distribution P_x on C such that given a test tuple $t_0 = (v_{0,1},\ldots..,v_{0,k},\ c_0)$, $P_0 = M(v_{0,1},\ldots.,v_{0,k})$ predicts the class label c_0 with high accuracy.We say that P_0 predicts c_0 if $c_0 = \arg\max_{c \in C} P_0(c)$.In this paper we study binary decision trees with tests on numerical attributes. Each internal node n of a decision tree is associated with an attribute Aj_n and a split point $z_n \in dom(Aj_n)$, giving a binary test $v_0,j_n \leq z_n$. An internal node has exactly 2 children, which are labelled "left" and "right", respectively. Each leaf node m in the decision tree is associated with a discrete probability distribut1ion P_m over C. For each $c \in C$, $P_m(c)$ gives a probability reflecting how likely a tuple assigned to leaf node m would have a class label of c.

To determine the class label of a given test tuple $t_0 = (v_{0,1},\ \ldots\ldots\ ,\ v_{0,k},\ ?)$, we traverse the tree starting from the root node until a leaf node is reached. When we visit an internal node n, we execute the test $v_0,j_n \leq z_n$ and proceed to the left child or the right child accordingly. Eventually, we reach a leaf node m. The probability distribution P_m associated with m gives the probabilities that t_0 belongs to each class label $c \in C$. For a single result, we return the class label $c \in C$ that maximises $P_m(c)$.

t \ v		0	1	2	3	4	5	6	7	8	C
t_0	V_0	3.2	4.3	7.8	9.1	1.4	4.1	2.8	4.0	6.6	C_1
t_1	V_1	1.2	2.1	3.7	4.1	1.8	2.3	3.1.	8.1	9.4	C_2
t_2	V_2	4.4	2.3	5.5	7.8	8.8	9.9	9.3	4.5	6.6	C_3

t \ v		0	1	2	3	4	5	6	7	8	C
t_0	V_0	3.2	4.3	7.8	9.1	1.4	4.1	2.8	4.0	6.6	?

B. Handling Uncertainty Information

Under our uncertainty model, a feature value is represented not by a single value, $v_{i,j}$, but by a pdf, $f_{i,j}$. For practical reasons, we assume that $f_{i,j}$ is non-zero only within a bounded interval $[a_{i,j}, b_{i,j}]$.A pdf $f_{i,j}$ could be programmed analytically if it can be specified in closed form. More typically, it would beimplemented numerically by storing a set of s sample points $x \in [a_{i,j}, b_{i,j}]$ with the associated value $f_{i,j}(x)$, effectively approximating $f_{i,j}$ by a discrete distribution with s possible values. We adopt this numerical approach for the rest this paper. With this representation, the amount of information available is exploded by a factor of s. Hopefully, the richer information allows us to build a better classification model.On the down side, processing large numbers of sample point is much more costly. In this paper we show that accuracy can be improved by

considering uncertainty information. We also propose pruning strategies that can greatly reduce the computational effort.

A decision tree under our uncertainty model resembles that of the point-data model. The difference lies in the way the tree is employed to classify unseen test tuples. Similar to thetraining tuples, a test tuple t_0 contains uncertain attributes.Its feature vector is thus a vector of pdf's $(f_{0,1}, \ldots, f_{0,k})$.A classification model is thus a function M that maps such a feature vector to a probability distribution P over C. The probabilities for P are calculated as follows. During these calculations, we associate each intermediate tuple t_x with a weight $w_x \in [0, 1]$. Further, we recursively define the quantity $\phi_n(c;t_x,w_x)$, which can be interpreted as the conditional probability that t_x has class label c, when the sub tree rooted at n is used as an uncertain decision tree to classify tuple t_x with weight w_x.

For each internal node n (including the root node), to determine $\phi_n(c; t_x,w_x)$, we first check the attribute Aj_n andsplit point z_n of node n. Since the pdf of t_x under attribute Aj_n spans the interval z_n $[a_x,j_n, b_x,j_n]$, we compute the "left" probability $p_L = \int^{zn} f_{x,jn}(t)dt$(or $p_L = 0$ in case $z_n < a_{x,jn}$) and $a_x,j_n$the "right" probability $p_R = 1 - p_L$. Then, we split t_x into 2 fractional tuples t_L and t_R. (The concept of fractional tuples is also used in C4.5[3] for handling missing values.) Tuples t_L and t_R inherit the class label of t_x as well as the pdf's of t_x for all attributes except Aj_n. Tuple t_L is assigned a weight of $w_L = w_x \_ p_L$ and its pdf for Ajn is given by

$$f_{L,j_n}(x) = \begin{cases} f_{x,j_n}(x)/w_L & \text{if } x \in [a_{x,j_n}, z_n] \\ 0 & \text{otherwise} \end{cases}$$

Tuple t_R is assigned a weight and pdf analogously. We define $\phi_n(c; t_x,w_x) = {}_L\phi_{nL}(c,t_L,w_L)+p_R.\phi_{nR}(c; t_R,w_R)$ where n_L and n_R are the left child and the right child of node n,respectively.

For every leaf node m, recall that it is associated with a probability distribution Pm over C. We define $\phi_m(c; t_x,w_x) = w_x .P_m(c)$. Finally, for each class c, let P(c) = _r(c; t0; 1:0),where r is the root node of the decision tree. Obtained this way, each probability P(c) indicates how likely it is that

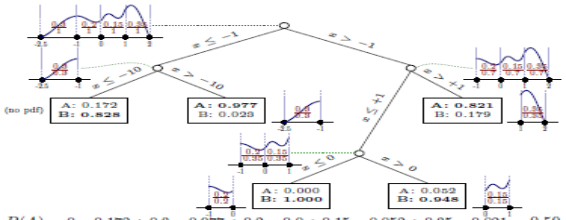


Fig. 1. Classifying a test tuple

the test tuple t_0 has class label c. These computations are illustrated in Figure 1, which shows a test tuple t_0 with one feature whose pdf has the domain [*-2.5, 2*]. It has a weight of 1.0 and is first tested against the root node of the decision tree. Based on the split point -1, we find that $P_L = 0.3$ and $P_R = 0.7$. So, t_0 is split into two tuples t_L and t_R with weights $w_L = 0.3$ and $w_R = 0.7$. The tuple t_L inherits the pdf from t_0 over the sub-domain [-2.5,-1], normalised by multiplying by a factor of $1/w_L$. Tuple t_R inherits the pdf from t_0 in a similar fashion. These tuples are then recursively tested down the tree until the leaf nodes are reached. The weight distributed in such a way down to each leaf node is then multiplied with the probability of each class label at that leaf node. These are finally summed up to give the probability distribution (over the class labels) for t_0, giving P(A) = 0.59; P(B) = 0.41.

If a single class label is desired as the result, we select the class label with the highest probability as the final answer. In the example in Figure 1, the test tuple is thus classified as class "A" when a single result is desired.

The most challenging task is to construct a decision tree based on tuples with uncertain values. It involves finding a good testing attribute A_{jn} and a good split point z_n for each internal node n, as well as an appropriate probability distribution Pm over C for each leaf node m. We describe algorithms for constructing such trees in the next section.

IV. ALGORITHMS

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. More specifically, for each tuple t_i and attribute A_j , we take the mean value[1]

$$v_{i,j} = \int_{a_{i,j}}^{b_{i,j}} x f_{i,j}(x)\, dx$$

as its representative value. The feature vector of ti is thus transformed to ($v_{i,1}, \ldots, v_{i,k}$). 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. The challenge here is that a training tuple can now "pass" a test at a tree node probabilistically when its pdf properly contains the split point of the test. Also, a slight change of the split point modifies that probability, potentially altering the tree structure. We present details of the tree-construction algorithms under the two approaches in the following subsections.

A. Averaging

A straight-forward way to deal with the uncertain information is to 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 traditional decision tree algorithms such as ID3 and C4.5[3] can be reused. We call this approach AVG (for Averaging). We use an algorithm based on C4.5. Here is a brief description.

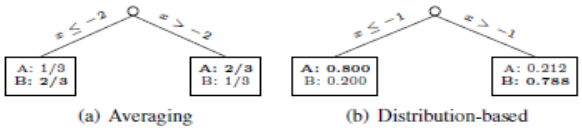
AVG is a greedy algorithm that builds a tree top-down.When processing a node, we examine a set of tuples S. The algorithm starts with the root node and with S being the set of all training tuples. At each node n, we first check if all the tuples in S have the same class label c. If so, we make n a leaf node and set $P_n(c) = 1,\ P_n(c') = 0\ \forall c' \neq c.$ Otherwise, we select an attribute Ajn and a split point z_n and divide the tuples into two subsets: "left" and "right". All tuples with $v_{i,j\ n} \leq z_n$ 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 A_{jn} and z_n), 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 cЄC, we assign to $P_n(c)$ the fraction of tuples in S that are labelled 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 A_{jn} and z_n is crucial. At this point, we may assume that this selection is performed by a blackbox algorithm BestSplit, 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 blackbox in details. Typically, BestSplit is designed to select the attribute and split point that minimises the degree of dispersion. The degree of dispersion can be measured in many ways,such as entropy (from information theory) or Gini index[30].The choice of dispersion function affects the structure of the resulting decision tree.[2] In this paper we assume that entropy is used as the measure since it is predominantly used for building decision trees.The minimisation is taken over the set of all possible attributes A_j (j = 1,……,k), considering all possible split points in dom(A_j). Given a set S = {t_1,……,t_m} of m tuples with point values, there are only m-1 ways to partition.

Probability distributions

Tuple	Class	Mean	-10	-1.0	0.0	+1.0	+10
1	A	+2.0		8/11			3/11
2	A	-2.0	1/9	8/9			
3	A	+2.0		5/8		1/8	2/8
4	B	-2.0	5/19	1/19		13/19	
5	B	+2.0			1/35	30/35	4/35
6	B	-2.0	3/11			8/11	

TABLE 1



(a) Averaging (b) Distribution-based

Fig. 2. Decision tree built from example tuples in Table I

into two non-empty L and R sets. For each attribute A_j , the split points to consider are given by the set of values of the tuples under attribute A_j , i.e., $\{v_{1,j}, \ldots, v_{m,j}\}$. Among these values, all but the largest one give valid split points. (The largest one gives an empty R set, so invalid.) For each of the (m-1)k combinations of attributes (A_j) and split points (z), we divide the set S into the "left" and "right" subsets L and R. We then compute the entropy for each such combination:

$$H(z, A_j) = \sum_{X=L,R} \frac{|X|}{|S|} \left(\sum_{c \in C} -p_{c/X} \log_2 p_{c/X} \right)$$

where P_c/X is the fraction of tuples in X that are labeled c. We take the pair of attribute A_{j*} and split point z^* that minimises $H(z,A_j)$ and assign to node n the attribute $A_{j*}$with split point z^*.[3]

Let us illustrate this classification algorithm using the example tuples shown in Table I. This set consists of 6 tuples of 2 class labels "A" and "B". Each tuple has only 1 attribute,whose (discrete) probability distribution is shown under the column "probability distribution". For instance, tuple 3 has class label "A" and its attribute takes the values of -1, +1, +10 with probabilities 5/8, 1/8, 2/8 respectively. The column "mean" shows the expected value of the attribute. For example,tuple 3 has an expected value of +2.0. With Averaging, there is only 1 way to partition the set: the even numbered tuples go to L and the odd-numbered tuples go to R. The tuples in each subset have the same mean attribute value, and hence cannot be discerned further. The resulting decision tree is shown in Figure 2(a). Since the left subset has 2 tuples of class B and 1 tuple of class A, the left leaf node L has the probability distribution $P_L(A) = 1/3$ and $P_L(B) = 2/3$ over the class labels. The probability distribution of class labels in the right leaf node R is determined analogously. Now, if we use the 6 tuples in Table I as test tuples[4] and use this decision tree to classify them, we would classify tuples 2, 4, 6 as class "B"(the most likely class label in L) and hence misclassify tuple 2. We would classify tuples 1, 3, 5 as class "A", thus getting the class label of 5 wrong. The accuracy is 2/3.

B. Distribution-based

For uncertain data, we adopt the same decision tree building framework as described above for handling point data. After an attribute A_{jn} and a split point z_n 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 t_i Є S under attribute A_{jn} spans the interval $[a_{i,jn}, b_{i,jn}]$. If $b_{i,jn} \leq z_n$, the pdf of t_i lies completely on the left of the split point and thus t_i is assigned to L. Similarly, we assign t_i to R if $z_n < a_{i,jn}$. If the pdf properly contains the split point, i.e.,$a_{i,jn} \leq zn < b_{i,jn}$, we split t_i into two fractional tuples T_L and t_R in the same way as described in Section III-B and add them to L and R, respectively. We call this algorithm UDT (for Uncertain Decision Tree).

Again, the key to building a good decision tree is a good choice of an attribute Ajn and a split point z_n 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 t_i's pdf spans a continuous range [$a_{i,j}$, $b_{i,j}$]. Moving the split point from a $_{i,j}$ to $b_{i,j}$ continuously changes the probability $p_L = \int_{a_{t,jn}}^{z_n} f_{i,j_n}(x)\,dx$ (and likewise for p_R). This changes the fractional tuples t_L and t_R, and thus changes the resulting tree. If we model a pdf by s sample values, we are approximating the pdf by a discrete distributionof s points. In this case, as the split point moves from one end-point $a_{i,j}$ to another end-point $b_{i,j}$ of the interval, the probability p_L changes in s steps. With m tuples, there arein 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.

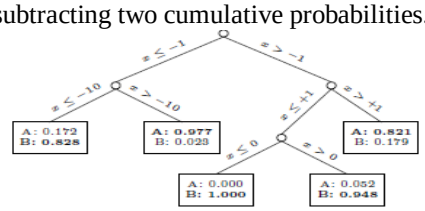


Fig. 3. Example decision tree before post-pruning

TABLE II

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 2(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 a 100% classification accuracy! After post-pruning, we get the tree in Figure 2(b).Now, let us use the 6 tuples in Table I as testing tuples4 to test the tree in Figure 2(b). For instance, the classification result of tuple 3 gives P(A) = 5/8 x 0.80 + 3/8 x 0.212 = 0.5795 and P(B) = 5/8 x 0.20 + 3/8 x 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.

V. CONCLUSIONS

We have extended the model of decision-tree classification to accommodate data tuples having numerical attributes with vagueness described by arbitrary pdf's. We have modified classical decision tree building algorithms (based on the framework of C4.5[3]) to build decision trees for classifying such data. We have found empirically that when suitable pdf's are used, exploiting data vagueness leads to decision trees with remarkably higher accuracies. We therefore advocate that data be collected and stored with the pdf information intact. Performance is an issue, though, because of the increased amount of information to be processed, as well as the more complicated entropy computations involved. Therefore, we have devised a series of pruning techniques to improve tree construction efficiency. Their execution times are of an order of magnitude comparable to classical algorithms.Although our novel techniques are primarily designed to handle uncertain data, they are also useful for building decision trees using classical algorithms when there are tremendous amounts of data tuples.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami, "Database mining: A performance perspective," IEEE Trans. Knowl. Data Eng., vol. 5, no. 6,pp. 914–925, 1993.
[2] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1,no. 1, pp. 81–106, 1986.
[3] C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993,ISBN 1-55860-238-0.
[4] G. L. Freed and J. K. Fraley, "25% "error rate" in ear temperature sensing device," Pediatrics, vol. 87, no. 3, pp. 414–415, Mar. 1991.
[5] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," The VLDB Journal, vol. 16, no. 4, pp. 523–544, 2007.
[6] E. Hung, L. Getoor, and V. S. Subrahmanian, "Probabilistic interval XML," ACM Transactions on Computational Logic (TOCL), vol. 8,no. 4, 2007.
[7] A. Nierman and H. V. Jagadish, "ProTDB: Probabilistic data in XML,"in VLDB. Hong Kong, China: Morgan Kaufmann, 20-23 Aug. 2002,pp. 646–657.
[8] J. Chen and R. Cheng, "Efficient evaluation of imprecise location dependent queries," in ICDE. Istanbul, Turkey: IEEE, 15-20 Apr. 2007,pp. 586–595.
[9] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain data mining: An example in clustering location data," in PAKDD, ser. Lecture Notes in Computer Science, vol. 3918. Singapore: Springer, 9–12 Apr. 2006, pp. 199–204.
[10] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter, "Efficient indexing methods for probabilistic threshold queries over uncertain data," in VLDB. Toronto, Canada: Morgan Kaufmann, 31 Aug.–3 Sept.2004, pp. 876–887.
[11] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," IEEE Trans. Knowl. Data Eng.,vol. 16, no. 9, pp. 1112–1127, 2004.
[12] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip,"Efficient clustering of uncertain data," in ICDM. Hong Kong, China:IEEE Computer Society, 18–22 Dec. 2006, pp. 436–445.
[13] S. D. Lee, B. Kao, and R. Cheng, "Reducing UK-means to K-means,"in The 1st Workshop on Data Mining of Uncertain Data (DUNE), in conjunction with the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE, USA, 28 Oct. 2007.
[14] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in KDD. Chicago, Illinois, USA: ACM, 21–24 Aug. 2005, pp.672–677.
[15] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in PAKDD, ser. Lecture Notes in Computer Science,vol. 4426. Nanjing, China: Springer, 22-25 May 2007, pp. 47–58.

[16] C. C. Aggarwal, "On density based transforms for uncertain data mining," in ICDE. Istanbul, Turkey: IEEE, 15-20 Apr. 2007, pp. 866–875.
[17] O. O. Lobo and M. Numao, "Ordered estimation of missing values," in PAKDD, ser. Lecture Notes in Computer Science, vol. 1574. Beijing,China: Springer, 26-28 Apr. 1999, pp. 499–503.
[18] L. Hawarah, A. Simonet, and M. Simonet, "A probabilistic approach to classify incomplete objects using decision trees," in DEXA, ser. Lecture Notes in Computer Science, vol. 3180. Zaragoza, Spain: Springer, 30 Aug.-3 Sep. 2004, pp. 549–558.
[19] J. R. Quinlan, "Learning logical definitions from relations," Machine Learning, vol. 5, pp. 239–266, 1990.
[20] T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes," Machine Learning, vol. 36, no. 3, pp. 201–244, 1999.
[21] U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," Machine Learning, vol. 8, pp.87–102, 1992.
[22] T. Elomaa and J. Rousu, "Efficient multisplitting revisited: Optimapreserving elimination of partition candidates," Data Mining and Knowledge Discovery, vol. 8, no. 2, pp. 97–126, 2004.
[23] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Wadsworth, 1984.
[24] T. Elomaa and J. Rousu, "Necessary and sufficient pre-processing in numerical range discretization," Knowledge and Information Systems, vol. 5, no. 2, pp. 162–182, 2003.