A Rule-Based Classification Algorithm for Uncertain Data

Biao Qin, Yuni Xia
Department of Computer Science
Indiana University - Purdue University Indianapolis, USA
{biaoqin,yxia}@cs.iupui.edu

Sunil Prabhakar
Department of Computer Science
Purdue University
sunil@cs.purdue.edu

Yicheng Tu
Department of Computer Science and Engineering
University of South Florida
ytu@cse.usf.edu

***Abstract*— Data uncertainty is common in real-world applications due to various causes, including imprecise measurement, network latency, outdated sources and sampling errors. These kinds of uncertainty have to be handled cautiously, or else the mining results could be unreliable or even wrong. In this paper, we propose a new rule-based classification and prediction algorithm called uRule for classifying uncertain data. This algorithm introduces new measures for generating, pruning and optimizing rules. These new measures are computed considering uncertain data interval and probability distribution function. Based on the new measures, the optimal splitting attribute and splitting value can be identified and used for classification and prediction. The proposed uRule algorithm can process uncertainty in both numerical and categorical data. Our experimental results show that uRule has excellent performance even when data is highly uncertain.**

I. Introduction

In many applications, data contains inherent uncertainty. A number of factors contribute to the uncertainty, such as the random nature of the physical data generation and collection process, measurement and decision errors, unreliable data transmission and data staling. For example, in location based services, moving objects of interest are attached with locators and this location information is periodically updated and streamed to the control center. Based on this location information, many location-based services can be provided including real-time traffic based routing, public transportation planning and accident prediction. In these types of applications, location data is typically inaccurate due to locator energy and precision constraint, network bandwidth constraint and latency. There are also massive uncertain data in sensor networks such as temperature, humidity and pressure. All of them are uncertain numerical data [1].

Uncertainty can also arise in categorical data [2]. For example, a tumor is typically classified as benign or malignant in cancer diagnosis and treatment. In practice, it is often very difficult to accurately classify a tumor at the initial diagnosis due to the experiment precision limitation. Inevitably the lab results are sometimes false positive or false negative results. Therefore, doctors may often decide tumors to be benign or malignant with certain probability or confidence.

Since data uncertainty is ubiquitous, it is important to develop data mining algorithms for uncertain datasets. In this paper, we focus on developing a rule-based classification algorithm for data with uncertainty. Rule-based data mining algorithms have a number of desirable properties. Rule sets are relatively easy for people to understand [3], and rule learning systems outperform decision tree learners on many problems [4], [5]. Rule sets have a natural and familiar first order version, namely Prolog predicates, and techniques for learning propositional rule sets can often be extended to the first-order case [6], [7]. However, when data contains uncertainty - for example, when some numerical data are, instead of precise value, an interval with probability distribution function with that interval - these algorithms can not process the uncertainty properly.

In the paper, we propose a new rule-based algorithm for classifying and predicting both certain and uncertain data. We integrate the uncertain data model into the rule-based mining algorithm. We propose a new measure called probabilistic information gain for generating rules. We also extend the rule pruning measure for handling data uncertainty. We perform experiments on real datasets with both uniform and Gaussian distribution, and the experimental results demonstrate that uRule algorithm perform well even on highly uncertain data.

This paper is organized as follows. In the next section, we will discuss related work on data mining with uncertainty. Section 3 describes the uncertainty model for both numerical and categorical data. Section 4 presents the rule-based algorithm. In Section 5, we show the details for computing the probabilistic information gain using the uncertain data model. Section 6 explains the measures for rule pruning based on unceratain data. Section 7 discusses the class prediction for uncertain data. The experimental results are shown in Section 8 and Section 9 concludes the paper.

II. Related Work

Classification is a well-studied area in data mining. Numerous classification algorithms have been proposed in the literature, such as decision tree classifiers [8], rule-based classifiers [9], Bayesian classifiers [10], support vector machines (SVM) [11], artificial neural networks [12], Lazy Learners, and ensemble methods [13]. Decision tree induction is the learning of a decision tree from class-labeled training tuples.

1084-4627/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDE.2009.164

A rule-based classifier is a technique for classifying records using a collection of "if ... then ..." rules. Bayesian classifiers are statistical classifiers and are based on Bayes theorem. SVM has its roots in statistical learning theory and has shown promising empirical results in many practical applications, from handwritten digit recognition to text categorization. An artificial neural network is a computational model based on biological neural networks. An ensemble method constructs a set of base classifiers from training data and performs classification by taking a vote on the predictions made by each base classifier.

In spite of the numerous classification algorithms, building classifiers based on uncertain data has remained a great challenge. There are early work performed on developing decision trees when data contains missing or noisy values [14], [15], [16]. Various strategies have been developed to predict or fill missing attribute values. However, the problem studied in this paper is different from before. Instead of assuming part of the data has missing or noisy values, we allow the whole dataset to be uncertain. Furthermore, the uncertainty is not shown as missing or erroneous values but represented as uncertain intervals and probability distribution functions. There are also some previous work performed on classifying uncertain data in various applications [17], [18], [19]. These methods try to solve specific classification tasks instead of developing a general algorithm for classifying uncertain data.

Recently, there have been studies on clustering of uncertain data. In [20], [21], the Kmeans clustering algorithms are extended so that the distance between objects are computed as the Expected Distance using a probability distribution function. For uncertain versions of k-means and k-median, Cormode et al [22] show reductions to their corresponding weighted versions on data with no uncertainties. The FDBSCAN and FOPTICS [23], [24] algorithms are based on DBSCAN and OPTICS, respectively. Instead of identifying regions with high density, these algorithms identify regions with high expected density, based on the pdfs of the objects. Aggarwal and Yu [25], [26] propose an extension of their micro-clustering technique to uncertain data. [27] introduces a new conceptual clustering algorithm for uncertain categorical data. There is also research on identifying frequent itemset and association mining [28], [29] from uncertain datasets. The support of itemsets and confidence of association rules are integrated with the existential probability of transactions and items. However, none of them address the issue of developing classification and predication algorithms for uncertain datasets.

III. The Uncertainty Data Model

In this section, we will discuss the uncertain model for both numerical and categorical attributes, which are the most common types of attributes encountered in data mining applications. In this paper, we focus on the uncertain attributes and assume the class type is certain.

TABLE I
Training Set for predicting borrowers who will default on loan payments

Rowid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	110-120	No
2	No	Single	60-85	No
3	Yes	Married	110-145	No
4	No	Divorced	110-120	Yes
5	No	Married	50-80	No
6	Yes	Divorced	170-250	No
7	No	Single	85-100	Yes
8	No	Married	80-100	No
9	No	Single	120-145	Yes
10	No	Divorced	95-115	Yes
11	No	Divorced	80-95	No

A. A model for uncertain numerical data

When the value of a numerical attribute is uncertain, the attribute is called an uncertain numerical attribute (UNA), denoted by $A_i^{u_n}$. Further, we use $A_{ij}^{u_n}$ to denote the jth instance of $A_i^{u_n}$. The concept of UNA has been introduced in [1]. The value of $A_i^{u_n}$ is represented as a range or interval and the probability distribution function (PDF) over this range. Note that $A_i^{u_n}$ is treated as a continuous random variable. The PDF $f(x)$ can be related to an attribute if all instances have the same distribution, or related to each instance if each instance has different distributions.

Table I shows an example of UNA. The data in this table are used to predict whether borrowers will default on loan payments. Among all the attributes, the Annual Income is a UNA, whose precise value is not available. We only know the range of the Annual Income of each person and the PDF $f(x)$ over that range. The probability distribution function of the UNA attribute Annual Income is assumed to be uniform distribution and PDF $f(x)$ is not shown in Table I.

Definition 1. An uncertain interval instance of $A_i^{u_n}$, denoted by $A_{ij}^{u_n}.U$, is an interval $[A_{ij}^{u_n}.l, A_{ij}^{u_n}.r]$ where $A_{ij}^{u_n}.l, A_{ij}^{u_n}.r \in R$, $A_{ij}^{u_n}.r \geq A_{ij}^{u_n}.l$.

Definition 2. The uncertain PDF of $A_{ij}^{u_n}$, denoted by $A_{ij}^{u_n}.f(x)$, is a probability distribution function of $A_{ij}^{u_n}$, such that $\int_{A_{ij}^{u_n}.l}^{A_{ij}^{u_n}.r} A_{ij}^{u_n}.f(x)dx = 1$ and $\int_{A_{ij}^{u_n}.l}^{A_{ij}^{u_n}.r} A_{ij}^{u_n}.f(x)dx = 0$ if $x \notin A_{ij}^{u_n}.U$.

The exact realization of this model is application-dependent. For example, in modeling sensor measurement uncertainty, $A_i^{u_n}.U$ is an error bound and $f(x)$ is a Gaussian distribution. The specification of uncertain PDF also depends on applications. For convenience, one may assume that the uncertain PDF $f(x)$ is a uniform distribution i.e. $f(x) = \frac{1}{A_{ij}^{u_n}.r - A_{ij}^{u_n}.l}$ for $A_{ij}^{u_n} \in [A_{ij}^{u_n}.l, A_{ij}^{u_n}.r]$; essentially, this implies a "worst-case" scenario where we have no knowledge of which point in the uncertain interval possesses a higher probability.

B. A model for uncertain categorical data

Under the uncertainty categorical model, a dataset can have attributes that are allowed to take uncertain values. We shall

TABLE II
TRAINING SET FOR PREDICTING PATIENT FOR 10 YEAR SURVIVAL

Rowid	Sex	Symptom	Tumor	Class
1	M	a	(Benign, 0.3) (Malignant, 0.7)	1
2	M	b	(Benign, 0.2) (Malignant, 0.8)	0
3	M	c	(Benign, 0.9) (Malignant, 0.1)	1
4	F	b	(Benign, 0.3) (Malignant, 0.7)	1
5	F	a	(Benign, 0.8) (Malignant, 0.2)	1
6	F	a	(Benign, 0.4) (Malignant, 0.6)	1
7	F	a	(Benign, 0.1) (Malignant, 0.9)	0
8	M	c	(Benign, 0.4) (Malignant, 0.6)	0
9	M	b	(Benign, 0.1) (Malignant, 0.9)	0
10	F	b	(Benign, 0.2) (Malignant, 0.8)	0
11	M	a	(Benign, 0.4) (Malignant, 0.6)	0

call such an attribute an uncertain categorical attribute (UCA), denoted by $A_i^{u_c}$. Further, we use $A_{ij}^{u_c}$ to denote the jth instance of $A_i^{u_c}$. The notion of UCA was proposed in [2].

$A_{ij}^{u_c}$ takes values from the categorical domain Dom with cardinality $|Dom| = n$. Within a regular relation, the value of an attribute A is a single value v_k in Dom, $P(A = v_k) = 1$. In the case of an uncertain relation, we record the information by a probability distribution over Dom instead of a single value. We have the following definition:

Definition 3. Given a categorical domain $Dom = \{v_1, \ldots, v_n\}$, an uncertain categorical attribute (UCA) $A_i^{u_c}$ is characterized by probability distribution over Dom. It can be represented by the probability vector $\mathbf{P} = \{p_{j1}, \ldots, p_{jn}\}$ such that $P(A_{ij}^{u_c} = v_k) = p_{jk}$ and $\sum_{k=1}^{n} p_{jk} = 1$ $(1 \leq k \leq n)$.

Table II shows an example of UCA. This dataset is used for a medical diagnosis and it contains information for cancer patients with a tumor. The type of tumor for each patient is a UCA attribute, whose exact value is unobtainable. It may be either benign or malignant, each with associated probability. The class type of each instance is either 0 or 1, representing whether the patient survived 5 years after the diagnosis.

IV. URULE ALGORITHM

To build a rule-based classifier, we need to extract a set of rules that show the relationships between the attributes of a dataset and the class label. Each classification rule is a form $R : (Condition) => y$. Here the (Condition) is called the rule antecedent, which is a conjunction of the attribute test condition. y is called the rule consequent and it is the class label. A rule set can consist of multiple rules $R_S = \{R_1, R_2, ..., R_n\}$

A rule R covers an instance I_j if the attributes of the instance satisfy the condition of the rule. The **Coverage** of a rule is the number of instances that satisfy the antecedent of a rule. The **Accuracy** of a rule is the fraction of instances that satisfy both the antecedent and consequent of a rule, normalized by those satisfying the antecedent. Ideal rules should have both high coverage and high accuracy rates.

The uRule algorithm is shown in Algorithm 1. It uses the sequential covering approach to extract rules from the data. The algorithm extracts the rules one class at a time for a data set. Let $(y_1, y_2, \ldots, y_n)$ be the ordered classes according to their frequencies, where y_1 is the least frequent class and y_n is the most frequent class. During the ith iteration, instances that belong to y_i are labeled as positive examples, while those that belong to other classes are labeled as negative examples. A rule is desirable if it covers most of the positive examples and none of the negative examples. Our uRule algorithm is based on the RIPPER algorithm [9], which was introduced by Cohen and considered to be one of the most commonly used rule-based algorithms in practice.

Algorithm 1 uRule(Dataset D, ClassSet C)

begin

1: $RuleSet$ = $\emptyset$; //initial set of rules learned is empty
2: **for** Each Class $c_i \in C$ **do**
3: newRuleSet = uLearnOneRule(D, c_i);
4: Remove tuples covered by newRuleSet from DataSet D;
5: $RuleSet$ += $newRuleSet$;
6: **end for**;
7: return $RuleSet$;

end

The uLearnOneRule() procedure, shown in Algorithm 2, is the key function of the uRule algorithm. It generates the best rule for the current class, given the current set of uncertain training tuples. The uLearnOneRule() includes two phases: growing rules and pruning rules. We will explain the first phase, growing rules, in more detail, while the other pruning rules is similar to regular rule-based classifier, thus will not be elaborated. After generating a rule, all the positive and negative examples covered by the rule are eliminated. The rule is then added into the rule set as long as it does not violate the stopping condition, which is based on the minimum description length (DL) principle. uRule also performs additional optimization steps to determine whether some of the existing rules in the rule set can be replaced by better alternative rules.

The process of growing rules, uGrow(), is shown in Algorithm 3. The basic strategy is as follows:

1. It starts with an initial rule : {}-> y, where the left hand side is an empty set and the right hand side contains the target class. The rule has poor quality because it covers all the examples in the training set. New conjuncts will subsequently be added to improve the rule's quality.

2. The probabilistic information gain is used as a measure to identify the best conjunct to be added into the rule antecedent. The algorithm selects the attribute and split point which has the highest probabilistic information gain and add them as

Algorithm 2 uLearnOneRule(Dataset D, Class c_i)

begin

1: stop = false;
2: RuleSet = $\emptyset$;
3: **repeat**
4: Split D into growData and pruneData;
5: Rule = uGrow(growData);
6: Prune Rules based on pruneData;
7: Add Rules to RuleSet;
8: Remove data covered by Rule from D;
9: **until** Stop Condition is true
10: **return**(RuleSet);

end

an antecedent of the rule (steps 3-4). The details of how to compute probabilistic information gain for uncertain data will be shown in the next section.

3. If an instance is covered by the rule, Function splitUncertain(), is invoked (steps 5-9). Function splitUncertain() returns part of the instance that is covered by the rule. Then, the part of the instance that is covered by the rule is removed from the dataset, and the rule growing process continues, until either all the data are covered or all the attributes have been used as antecedents.

Algorithm 3 uGrow(Instances growData)

begin

1: coverData = $\emptyset$;
2: **while** (growData.size() $>$ 0) $\wedge$ (numUnusedAttributes $>$ 0) **do**
3: Find the attribute A_i and the split point sp, which has the highest probabilistic information gain;
4: Antecedent += RuleAntecedent(A_i, sp);
5: **for** (each instance I_j) **do**
6: **if** (covers(I_j)) **then**
7: inst = splitUncertain(I_j, A_i, sp);
8: coverData += inst;
9: **end if**;
10: **end for**;
11: growData -= coverData;
12: **end while**;

end

Function splitUncertain() is shown in Algorithm 4. As the data is uncertain, a rule can partly cover an instance. For example, for an instance with income [100, 110], a rule "income $>$ 105 $\Rightarrow$ defaultBorrower = No" only partly covers it. For an instance with tumor [Benign, 0.8; Malignant, 0.2], a rule "tumor = benign $\Rightarrow$ Survive = yes" also partly covers it. Function splitUncertain() computes what proportion of the instances is covered by a rule based on the uncertain attribute interval and probabilistic distribution function.

Algorithm 4 splitUncertain(Instance I_i, attribute A_i, splitPoint sp)

begin

1: **if** the rule fully covers instance I_i **then**
2: return I_i;
3: **end if**;
4: **if** (A_i is numerical or uncertain numerical) **then**
5: **if** the rule partially covers instance I_i on the right side **then**
6: I_i.w = I_i.w*$\int_s f(x)dx / \int_{min}^{max} f(x)dx$;
7: **end if**;
8: **if** the rule partially covers instance I_i on the left side **then**
9: I_i.w = I_i.w*$\int^s f(x)dx / \int_{min}^{max} f(x)dx$;
10: **end if**;
11: **end if**;
12: **if** (A_i is categorical or uncertain categorical) **then**
13: I_i.w = I_i.w*att.value(sp).w*P(I_i, R_k);
14: **end if**;
15: return I_i;

end

V. Probabilistic Information Gain

In the previous section, it is shown that when rules are being generated from training dataset, the goal is to determine the best split attribute and the best split point. We use a measure called probabilistic Information Gain to identify the optimal split attribute and split point for uncertain training dataset. In this section, we will explain the detail of this procedure.

A. Uncertain numerical attributes

As described earlier, the value of an uncertain numeric attribute is an interval with associated PDF. Each uncertain interval has a maximal value and a minimal value, which are called critical points. For each UNA, we can order all critical points of an uncertain numeric attribute in an ascending sort with duplicate elimination. Suppose there are N critical points after eliminating duplicates, then this UNA can be divided into $N+1$ partitions. Since the leftmost and rightmost partitions do not contain data instances at all, a split definitely will not occur within them. We need only consider the rest $N-1$ partitions. For a UNA A, assume it can be divided into n partitions. Let $S = [A.min, A.max]$, then $S = \cup_{i=1}^{n} Pa_i$.

One partition may overlap with the UNA of many instances. The partition [110, 120) on the Annual Income attribute has overlap with 4 instances, namely instances 1, 3, 4 and 10. Among them, instances 1 and 3 are in the class NO, while instances 4 and 10 are in class YES. When an instance with UNA overlaps with a partition [a, b], the probability of its UNA that actually falls in that partition is $\int_a^b f(x)dx$. For example, for instance 3, the probability that its actual Annual Income in the partition [110,120) is $\int_{110}^{120} f(x)dx$. If its PDF is a uniform distribution, then the probability is $\int_{110}^{120} f(x)dx = (120-110)/(145-110) = 0.286$. Based on the probability

TABLE III
GROW DATA FOR ONE RULE

RowId	Home Owner	Marital Status	Annual Income	Defaulted Borrower
2	No	Single	60-85	No
4	No	Divorced	110-120	Yes
5	No	Married	50-80	No
6	Yes	Divorced	170-250	No
8	No	Married	80-100	No
9	No	Single	120-145	Yes
10	No	Divorced	95-115	Yes

of each individual instance falling in a partition [a, b), we can compute the probabilistic cardinality of a dataset falling in that partition.

Definition 4. The probabilistic cardinality of the dataset over a partition $Pa = [a, b)$ is the sum of the probabilities of each instance whose corresponding UNA falls in [a, b). That is, $PC(Pa) = \sum_{j=1}^{n} P(A_{ij}^{u_n} \in [a, b)) = \sum_{j=1}^{n} \int_a^b A_{ij}^{u_n}.f(x)dx$.

For example, for the dataset in Table I. The probabilistic cardinality of the partition [110, 120) on the Annual Income is the sum of the probabilities of instances with Annual Income falling in [110, 120). Suppose each instance of Annual Income is uniformly distributed over its uncertain interval; instances 1, 3, 4 and 10 have overlap with [110,120), and the probability for instance 1 with Annual Income in [110, 120) is $P(I1 \in [110, 120)) = 1$. Similarly, $P(I3 \in [110, 120)) = 0.286$, $P(I4 \in [110, 120)) = 1$, and $P(I10 \in [110, 120)) = 0.25$. Therefore, the probabilistic cardinality of this dataset over partition [110, 120) is 2.536.

Definition 5. The probabilistic cardinality for class C_i of the dataset over a partition $Pa = [a, b)$ is the sum of the probability of each instance I_k in class C_i whose corresponding UNA falls in [a, b). That is, $PC(Pa, C_i) = \sum_{k=1}^{n} P(A_{jk}^{u_n} \in [a, b) \wedge C_{I_k} = C_i)$, where C_{I_k} denotes the class label of instance I_k.

Let us continue on the previous example. The probabilistic cardinality for class "DefaultBorrower = NO" over the partition [110, 120) on Annual Income is the sum of the probabilities of instances who are not a DefaultBorrower with Annual Income falling in [110, 120). Among instances 1, 3, 4 and 10 who overlap with [110,120), only instance 1 and 3 are in class NO. Therefore, the probabilistic cardinality for "DefaultBorrower = NO" over partition [110, 120) is 1.286. Similarly, the probabilistic cardinality for "DefaultBorrower = Yes" over partition [110, 120) is 1.25. The class distributions of each partition can be denoted by a vector $PC(Pa, C) = (PC(Pa, C_1), PC(Pa, C_2), \ldots, PC(Pa, C_n))^T$, which we call a class distribution vector (CDV). The CDV is $PC(Pa \in [110, 120), C) = (1.286, 1.25)$.

Example 1: We perform a 3-fold stratified holdout, taking out 2/3 the data by random sampling. We use those data to build rules and use the remaining 1/3 data for pruning. So the grow data for Table I is shown in Table III. The uncertain partitions and their CDVs of attribute Annual Income are shown in Table IV.

Definition 6. Let $y_1 = log_2 \frac{PC(p_1)}{PC(p_1)+PC(n_1)}$ and $y_0 = log_2 \frac{PC(p_0)}{PC(p_0)+PC(n_0)}$ The **probabilistic information gain** for a dataset D when adding an antecedent A_i is

$$ProbInfo(D, A_i) = PC(p_1) \times (y_1 - y_0) \quad (1)$$

Here p_0 and n_0 are the probabilistic cardinalities of positive and negative instances covered by the rule before adding antecedent A_i, and p_1 and n_1 are the probabilistic cardinalities of positive and negative instances covered by the rule after adding antecedent A_i. Since the measure is proportional to $PC(p_1)$ and $PC(p_1)/(PC(p_1) + PC(n_1))$, it prefers rules that have both a high support count (coverage) and accuracy. The probabilistic information gain will be used in the process of rule generation. It can determine what is the best antecedent to be added into a rule as well as the best splitting point. For example, based on the data in Table III, when generating rules for the "DefaultedBorrower=No" class, the best splitting point for the attribute Annual Income is at 95.

B. Uncertain categorical data

A rule related to an uncertain categorical attribute only covers its one value, which is called the split point. An uncertain categorical attribute (UCA) $A_i^{u_c}$ is characterized by probability distribution over Dom. As mentioned earlier, it can be represented by the probability vector $\mathbf{P} = \{p_{j1}, \ldots, p_{jn}\}$ such that $P(A_{ij}^{u_c} = v_k) = p_{jk}$ $(1 \leq i \leq n)$.

Assuming m is the total number of instances and n is the number of candidate values for the attribute, under the definition of UCA, we can record a UCA for a dataset using an $m*n$ matrix, which we call a categorical probability matrix, as shown in Matrix (2).

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & p_{m3} & \cdots & p_{mn} \end{bmatrix} \quad (2)$$

Datasets without uncertainty can be treated as a special case of data with uncertainty. When using a matrix to represent a categorical attribute of a dataset without uncertainty, there is at most one element per row to be 1. Similar to UNA, we have the following definitions:

Definition 7. The probabilistic cardinality of the dataset over v_k is the sum of the probability of each instance whose corresponding UCA equals v_k. That is, $PC(v_k) = \sum_{j=1}^{n} P(A_{ij}^{u_c} = v_k) = \sum_{j=1}^{n} p_{jk}$.

For example, for the dataset in Table II, the probabilistic cardinality over "Tumor = benign" is the overall probabilities of each instance whose Tumor is "benign", which is 4.1, and the probabilistic cardinality of "Tumor = malignant" is 6.9.

Definition 8. The probabilistic cardinality for class C_i of the dataset over v_k is the sum of the probability of each instance in class C_i whose corresponding UCA equals to v_k. That is, $PC(v_k, C_i) = \sum_{j=1}^{n} P(A_{ij}^{u_c} = v_k \wedge C_{I_j} = C_i)$.

For the dataset in Table II, the probabilistic cardinality for class **0** over "Tumor = benign" is the overall probabilities of instances in class **0** whose Tumor attribute is "benign",

TABLE IV
UNCERTAIN PARTITIONS AND CDVS

	[50,60]	[60,80]	[80,85)	[85,95)	[95,100)
$PC(Pa, C)$	(0,0.333)	(0,1.467)	(0,0.45)	(0,0.5)	(0.25,0.25)
	[100,110)	[110,115]	[115,120]	[120,145)	[170,250]
$PC(Pa, C)$	(0.5,0)	(0.75,0)	(0.5,0)	(1,0)	(0,1)

which is 1.4, and the probabilistic cardinality for class **1** over "Tumor = benign" is 4.6. The class distribution vector (CPV) is $PC(benign, C) = (1.4, 4.6)$. Based on the CPV, we can compute the probabilistic information gain if the categorical attribute is selected as the antecedent of a rule, following in the same process for uncertain numerical data.

VI. RULE PRUNING

uRule employs a general-to-specific strategy to grow a rule and the probabilistic information gain measure to choose the best conjunct to be added into the rule antecedent. The new rule is then pruned based on its performance on the validation set. We use the following metric for rule pruning.

Definition 9. The *probabilistic prune* for a rule R is

$$ProbPrune(R, p, n) = \frac{PC(p) - PC(n)}{PC(p) + PC(n)}$$

Here PC(p) and PC(n) is the probabilistic cardinality of positive and negative instances covered by the rule. This metric is monotonically related to the rule's accuracy on the validation set. If the metric improves after pruning, then the conjunct is removed. Like RIPPER, uRule starts with the most recently added conjunct when considering pruning. Conjuncts are pruned one at a time as long as this results in an improvement.

Example 2: Refer to the dataset shown in Table I. Using uRule Algorithm, after rule pruning, the following rule set is generated:
(annual_income $\geq$ 95) and (home_owner = No)=> class=Yes (3.58/0.25)
{} => class=No (7.42/0.67)
The first is a regular rule, whose accuracy is around 93%, since it covers 3.58 positive instances and 0.25 negative instances. Please note that for uncertain data, a rule may partly cover instances, therefore, the number of positive and negative instances covered by a rule are no longer integers but real values. The second rule is a default rule. Like traditional rule-based classifier, uRule also generate a default rule, which can be applied to instances which do not match any rules in the rule set. This default rule has an accuracy around 91%.

VII. URULE PREDICTION

Once the rules are learned from a dataset, they can be used for predicting class types of unseen data. Like a traditional rule classifier, each rule of uRule is in the form of "IF Conditions THEN Class = C_i". Because each instance I_i can be covered by several rules, a vector can be generated for each instance $(P(I_i, C) = (P(I_i, C_1), P(I_i, C_2), ..., P(I_i, C_n))^T)$, in which $P(I_i, C_j)$ denotes the probability for an instance to be in class C_j. We call this vector an Class Probability Vector (CPV).

As an uncertain data instance can be *partly* covered by a rule, we denote the degree an instance I covered by a rule R_j by $P(I, R_j)$. When $P(I, R_j)$ = 1, R_j fully covers instance I_i; when $P(I_i, R_j)$ = 0, R_j does not cover I_i; and when $0 < P(I_i, R_j) < 1$, R_j partially covers I_i.

An uncertain instance may be covered or partially covered by more than one rule. We allow a test instance to trigger all relevant rules. We use $w(I_i, R_k)$ to denote the weight of an instance I_i covered by the kth rule R_k. The weight of an instance I_i covered by different rules is as follows:

$$\begin{aligned} w(I_i, R_1) &= I_i.w * P(I_i, R_1) \\ w(I_i, R_2) &= (I_i.w - w(I_i, R_1)) * P(I_i, R_2) \\ &\cdots \\ w(I_i, R_n) &= (I_i.w - \sum_{k=1}^{n-1} w(I_i, R_k)) * P(I_i, R_n) \end{aligned}$$

For the first rule R_1 in the rule set, $w(I_i, R_1)$ should be the weight of the instance, $I_i.w$, times the degree the instance covered by the rule, $P(I_i, R_1)$. For the second rule R_2, $w(I_i, R_2)$ is the remained probability cardinality of instance I_i, which is $(I_i.w - w(I_i, R_1))$, times the rule coverage, $P(I_i, R_2)$. Similarly, $w(I_i, R_n)$ should be the remained probability cardinality of instance I_i, which is $(I_i.w - \sum_{k=1}^{n-1} w(I_i, R_k))$, times $P(I_i, R_n)$.

Suppose an instance I_i is covered by m rules, then it class probability vector $P(I_i, C)$ is computed as follows:

$$P(I_i, C) = \sum_{k=1}^{m} P(R_k, C) * w(I_i, R_k)$$

Where $P(R_k, C)$ is a vector $P(R_k, C) = (P(R_k, C_1), P(R_k, C_2), \ldots, P(R_k, C_n))^T$ and denotes the class distribution of the instances covered by rule R_k. $P(R_k, C_i)$ is computed as the fraction of the probabilistic cardinality of instances in class C_i covered by the rule over the overall probabilistic cardinality of instances covered by the rule.

After we compute the CPV for instance I_i, the instance will be predicted to be of class C_j, which has the largest probability in the class probability vector. This prediction procedure is different from a traditional rule based classifier. When predicting the class type for an instance, a traditional rule based classifier such as RIPPER usually predicts with the first rule in the rule set that covers the instance. As an uncertain data instance can be fully or partially covered by

multiple rules, the first rule in the rule set may not be the rule that covers it best. uRule will use all the relevant rules to compute the probability for the instance to be in each class and predict the instance to be the class with the highest probability.

Example 4. Suppose we have a test instance {No, Married, 90-110}, when applying the rules shown in Example 2 on the test instance, the class probability vector (CPV) is

$$\begin{aligned} P(I,C) &= P(R_1,C)*w(I,R_1)+P(R_2,C)*w(I,R_2) \\ &= \begin{pmatrix} 0.93017 \\ 0.06983 \end{pmatrix} \times 0.75 + \begin{pmatrix} 0.0903 \\ 0.9097 \end{pmatrix} \times 0.25 \\ &= \begin{pmatrix} 0.6976 \\ 0.0524 \end{pmatrix} + \begin{pmatrix} 0.0226 \\ 0.2274 \end{pmatrix} \\ &= \begin{pmatrix} 0.7202 \\ 0.2798 \end{pmatrix} \end{aligned}$$

This shows that the instance have 0.7202 probability to be in class "Yes" and 0.2798 probability in class "No". Thus the test instance will be predicted to be in class "Yes".

VIII. Experiments

In this section, we present the experimental results of the proposed rule-based uRule algorithm. We studied the uRule classifier accuracy and classifier construction time over multiple datasets.

A. Setup

We used 5 real-world benchmark datasets to evaluate the performance of uRule - diabetes, glass, iris, segment and sonar datasets. All of these datasets are available from the UCI Repository [30]. The experiments are executed on a PC with an Intel Pentium IV 3.2 GHz CPU and 2.0 GB main memory.

To make numerical attributes uncertain, we convert each numerical value to an uncertain interval. For each numerical attribute, we scan all of its value and get its maximum X_{max} and minimum value X_{min}, respectively. Let $Length = X_{max} - X_{min}$. If the uncertain interval is within 10% of the length, we call the dataset with 10% uncertainty and denote it by U10.

We make categorical attributes uncertain by converting them into probability vectors. For example, a categorical attribute A_i may have k possible values $v_j, 1 \leq j \leq k$. For an instance I_j, we convert its value A_{ij} into a probability vector $\mathbf{P} = (p_{j1}, p_{j2}, ..., p_{ji}, ..., p_{jk})$, while p_{jl} is the probability of $A_{ij}^{u_c}$ to be equal to v_l, that is, $P(A_{ij}^{u_c} = v_l) = p_{jl}$. For example, when we introduce 10% uncertainty, this attribute will take the original value with 90% probability, and 10% probability to take any of the other values. Suppose in the original accurate dataset $A_{ij} = v_1$, then we will assign $p_{j1} = 90\%$, and assign $p_{jl}(2 \leq l \leq k)$ to ensure $\sum_{i=2}^{k} p_{jl} = 10\%$. Similarly, we denote this dataset with 10% uncertainty in categorical data by U10.

B. Accuracy

In the following experiments, we use ten-fold cross validation. Data is split into 10 approximately equal partitions; each one is used in turn for testing while the rest is used for training, that is, 9/10 of data is used for training and 1/10 for testing. The whole procedure is repeated 10 times, and the overall accuracy rate is counted as the average of accuracy rates on each partition. In the figures, UX denotes the uncertainty in the datasets is within X%. We use U0 to denote accurate or certain datasets. When uRule is applied on certain datasets, it works as a traditional rule classifier RIPPER.

Figure 1 and 2 show the performance of uRule when uncertainty ranges from 0 to 20%. Figure 1 shows the result for uncertain data whose probability distribution function is uniform and Figure 2 is for data whose pdf is Gaussian. Overall speaking, the accuracy of uRule classifier remain relatively stable. Even when the extent of uncertainty reaches 20%, the accuracy is still quite comparable to one over certain or precise data. It shows that uRule is quite robust against data uncertainty. We observe similar trends for both uniform and Gaussian distribution uncertain data. As uncertainty further increases, uRule classification and prediction accuracy will degrade to some extent.

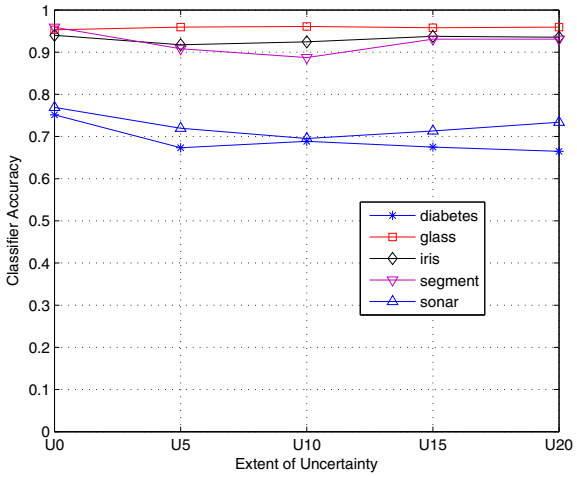

Fig. 1. uRule accuracy for uncertain data with uniform pdf

C. Computation time

We investigate the time it takes to construct a rule classifier out of uncertain data using the uRule algorithm. Tables V depicts the absolute run time in seconds when all instances of a dataset are used to build the rule set. It is shown that it generally takes longer to construct a classifier as the uncertainty in data increases. The reason is two-fold. First, for uncertain data, more candidate splitting points are available and require more comparisons. Second, uncertain data can be partly covered by rules, resulting in record splitting, weight and probabilistic cardinalities computation, all of which do not exist in classifier construction over certain datasets. Furthermore, it is shown that it takes longer to generate classifier from uncertain data with Gaussian PDF than with uniform PDF. The reason is that the calculation of cumulative distribution is more complicated

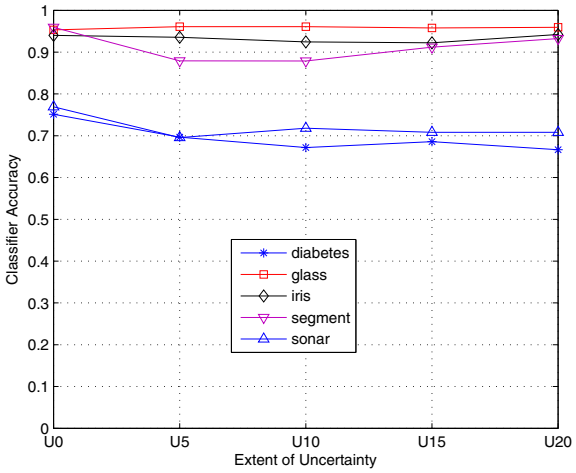

Fig. 2. uRule accuracy for uncertain data with Gaussian pdf

and more computation intensive for Gaussian distribution than for uniform distribution.

TABLE V
CLASSIFIER CONSTRUCTION TIME

	Dataset	U0	U5	U10	U15	U20
Uniform	diabetes	0.13	0.47	0.49	0.58	0.59
	glass	0.1	0.12	0.18	0.13	0.16
	iris	0.03	0.08	0.09	0.06	0.06
	segment	1.59	14.18	18	20.62	19.5
	sonar	0.12	0.52	0.49	0.73	0.75
Guassian	diabetes	0.13	2.85	4.75	5.63	5.04
	glass	0.1	0.6	0.78	0.99	1.12
	iris	0.03	0.11	0.16	0.14	0.21
	segment	1.59	126	249.1	408.5	302.2
	sonar	0.12	1.1	1.73	2	2.62

IX. CONCLUSIONS AND FUTURE WORK

Uncertain data often occur in modern applications, including sensor databases, spatial-temporal databases, and medical or biology information systems. In this paper, we propose a new rule-based algorithm for classifying and predicting uncertain datasets. We propose new approaches for deriving optimal rules out of highly uncertain data, pruning and optimizing rules, and class prediction for uncertain data. This algorithm follows the new paradigm of directly mining uncertain datasets. The avenues of future work include developing uncertain data mining techniques for various applications, including sequential pattern mining, association mining, spatial data mining and web mining, where data can be commonly uncertain.

REFERENCES

[1] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *ACM SIGMOD International Conference on Management of Data*, 2003, pp. 551–562.

[2] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, "Indexing categorical data with uncertainty," in *International Conference on Data Engineering (ICDE) 2007*, pp. 616–625.

[3] J. Catlett, "Megainduction: A test flight," in *ML*, 1991, pp. 596–599.

[4] G. Pagallo and D. Haussler, "Boolean feature discovery in empirical learning," *Machine Learning*, vol. 5, pp. 71–99, 1990.

[5] S. M. Weiss and N. Indurkhya, "Reduced complexity rule induction," in *IJCAI*, 1991, pp. 678–684.

[6] J. R. Quinlan, "Learning logical definitions from relations," *Machine Learning*, vol. 5, pp. 239–266, 1990.

[7] J. R. Quinlan and R. M. Cameron-Jones, "Induction of logic programs: Foil and related systems," *New Generation Comput.*, vol. 13, no. 3&4, pp. 287–312, 1995.

[8] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers, 1993.

[9] W. W. Cohen, "Fast effective rule induction," in *Proc. of the 12th Intl. Conf. on Machine Learning*, 1995, pp. 115–123.

[10] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *National Conf. on Artigicial Intelligence*, 1992, pp. 223–228.

[11] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[12] R. Andrews, J. Diederich, and A. Tickle, "A survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge Based Systems*, vol. 8, no. 6, pp. 373–389, 1995.

[13] T. G. Dietterich, "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, vol. 1857, pp. 1–15, 2000.

[14] Q. JR, *Probabilistic decision trees, in Machine Learning:An Artificial Intelligence Approach*. Morgan Kaufmann, 1990.

[15] L. O and N. M, "Ordered estimation of missing values," in *PAKDD*, 1999, pp. 499–503.

[16] H. L, S. A, and S. M, "Dealing with missing values in a probabilistic decision tree during classification," in *The Second International Workshop on Mining Complex Data*, 2006, pp. 325–329.

[17] B. J and Z. T, "Support vector classification with input data uncertainty," in *Advances in Neural Information Processing Systems*, 2004, pp. 161–168.

[18] E. V. Gonzalez, I. A. E. Broitman, E. E. Vallejo, and C. E. Taylor, "Targeting input data for acoustic bird species recognition using data mining and hmms," in *ICDMW 2007*, pp. 513–518.

[19] C. Jebari and H. Ounelli, "Genre categorization of web pages," in *ICDMW 2007*, pp. 455–464.

[20] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip, "Efficient clustering of uncertain data," in *IEEE International Conference on Data Mining (ICDM) 2006*, pp. 436–445.

[21] M. Chau, R. Cheng, B. Kao, and J. Ng, "Data with uncertainty mining: An example in clustering location data," in *Proc. of the Methodologies for Knowledge Discovery and Data Mining, Pacific-Asia Conference (PAKDD 2006)*, 2006.

[22] C. G and McGregor, "Approximation algorithms for clustering uncertain data," in *PODS*, 2008, pp. 191–199.

[23] H. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005, pp. 672–677.

[24] H. . Kriegel and M. . Pfeifle, "Hierarchical density-based clustering of uncertain data," in *IEEE International Conference on Data Mining (ICDM) 2005*, pp. 689–692.

[25] A. C, "On density based transforms for uncertain data mining," in *Proceedings of IEEE 23rd International Conference on Data Engineering*, 2007, pp. 866–875.

[26] A. C and Y. PS, "A framework for clustering uncertain data streams," in *Proceedings of IEEE 24rd International Conference on Data Engineering*, 2008, pp. 150–159.

[27] Y. Xia and B. Xi, "Conceptual clustering categorical data with uncertainty," in *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2007, pp. 329–336.

[28] Z. Yu and H. Wong, "Mining uncertain data in low-dimensional subspace," in *International Conference on Pattern Recognition (ICPR) 2006*, pp. 748–751.

[29] C. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in *Proc. of the Methodologies for Knowledge Discovery and Data Mining, Pacific-Asia Conference (PAKDD) 2007*, pp. 47–58.

[30] *http://archive.ics.uci.edu/ml/datasets.html*. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html