Direct Mining of Discriminative Patterns for Classifying Uncertain Data

Chuancong Gao, Jianyong Wang
Department of Computer Science and Technology
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing 100084, China
gaocc07@mails.tsinghua.edu.cn, jianyong@tsinghua.edu.cn

ABSTRACT

Classification is one of the most essential tasks in data mining. Unlike other methods, associative classification tries to find all the frequent patterns existing in the input categorical data satisfying a user-specified minimum support and/or other discrimination measures like minimum confidence or information-gain. Those patterns are used later either as rules for rule-based classifier or training features for support vector machine (SVM) classifier, after a feature selection procedure which usually tries to cover as many as the input instances with the most discriminative patterns in various manners. Several algorithms have also been proposed to mine the most discriminative patterns directly without costly feature selection. Previous empirical results show that associative classification could provide better classification accuracy over many datasets.

Recently, many studies have been conducted on uncertain data, where fields of uncertain attributes no longer have certain values. Instead probability distribution functions are adopted to represent the possible values and their corresponding probabilities. The uncertainty is usually caused by noise, measurement limits, or other possible factors. Several algorithms have been proposed to solve the classification problem on uncertain data recently, for example by extending traditional rule-based classifier and decision tree to work on uncertain data. In this paper, we propose a novel algorithm uHARMONY which mines discriminative patterns directly and effectively from uncertain data as classification features/rules, to help train either SVM or rule-based classifier. Since patterns are discovered directly from the input database, feature selection usually taking a great amount of time could be avoided completely. Effective method for computation of expected confidence of the mined patterns used as the measurement of discrimination is also proposed. Empirical results show that using SVM classifier our algorithm uHARMONY outperforms the state-of-the-art uncertain data classification algorithms significantly with 4% to 10% improvements on average in accuracy on 30 categorical datasets under varying uncertain degree and uncertain attribute number.

KDD'10, July 25–28, 2010, Washington, DC, USA.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*Data Mining*

General Terms

Algorithm, Experimentation

Keywords

Associative Classification, Uncertain Data, Frequent Pattern Mining, Expected Confidence

Code and Datasets

All the code and datasets are available at `http://dbgroup.cs.tsinghua.edu.cn/chuancong/uharmony/`.

1. INTRODUCTION

As one of the most essential tasks in data mining and machine learning area, classification has been studied for many years. Many effective models and algorithms have been proposed to solve the problem in different aspects, including decision tree, rule-based classifier, support vector machine, etc.

Unlike some traditional rule-based algorithms like Ripper [7] or FOIL [19], associative classification tries to mine the complete set of frequent patterns from the input dataset, given the user-specified minimum support threshold and/or discriminative measurements like minimum confidence threshold. Sequential covering technology is further employed to select the most discriminative patterns while covering most input training instances. A test instance is classified later using classifier trained based on the mined patterns. CBA [15] is one of the most classical associative classification algorithms. Empirical results show that associative classification algorithm could provide better classification accuracy than other algorithms on categorical datasets. However, this approach takes a great amount of running time in both pattern mining and feature selection, since most of the mined frequent patterns are not the most discriminative ones and will be dropped later.

To improve the efficiency of associative classification, several algorithms have been proposed in recent years to try to mine the most discriminative patterns directly during the pattern mining step. Different discriminative measures and different instance covering technologies have also been devised. One of the most typical algorithms is HARMONY [21] which uses confidence to evaluate the discrimination of patterns. It employs a so-called instance-centric framework to find one most discriminative pattern for each instance. Effective pruning methods have also been proposed to enhance the

algorithm efficiency. To classify a test instance, a rule-based classifier is built based on the mined patterns. From then on, several other algorithms [4, 9] mining pattern directly have also been proposed. The main differences of these algorithms in comparison with HARMONY include the replacement of confidence with information gain or information gain ratio as the pattern quality measure, the adoption of SVM model, and more advanced covering technologies such as the one using decision tree to partition data. Among these changes, the adoption of SVM model contributes the most in improving the classification accuracy.

Recently, more and more research has been conducted on uncertain data mining to solve the uncertainty usually caused by noise, measurement precisions, etc. Several algorithms have already been proposed to solve the frequent itemset mining problem where each item has a probability to appear, using either expected support [6] or frequentness probability [2] to measure the pattern frequentness. Classification for uncertain data has also been studied recently. For uncertain data classification, the values of the uncertain attributes are now represented using a probabilistic distribution function, for uncertain numeric attribute or uncertain categorical attribute. Several classical algorithms like C4.5 and Ripper have been extended to process uncertain attributes [17, 20, 16]. Some of them try to convert the uncertain attributes into certain ones by discretization using sample points, while others adopt probabilistic cardinalities of entropy, information gain and information gain ratio.

In this paper, we propose a new algorithm called uHARMONY to solve the problem of classifying uncertain categorical data. The algorithm adopts the same framework as algorithm HARMONY. Expected support is adopted to represent pattern frequentness, while expected confidence is employed to represent the discrimination of the mined patterns. We also devise a novel method to calculate the expected confidence efficiently. A new instance covering strategy has been devised to try to ensure that the instances are covered with a probability higher than a user-specified cover threshold. Evaluation on 30 public datasets with different number of uncertain attributes and different uncertain degrees (which is defined as the probability the attribute takes values other than the original single value) shows that our algorithm outperforms two state-of-the-art algorithms significantly with 4% to 10% improvements in accuracy on average while using SVM as the classification model.

Our contributions could be summarized as follows.

- We devise a new associative classification algorithm mining the most discriminative patterns directly on uncertain data. To our best knowledge, this is the first associative classification algorithm for classifying uncertain data.
- We adopt the expected confidence as the measurement of discriminative degree, instead of other probabilistic cardinalities without reasonable theoretical explanations. A novel upper-bound based approach is also proposed to speedup the calculation of expected confidence.
- Unlike covering instance with only one pattern having the corresponding maximum confidence in HARMONY, we devise a novel instance covering strategy to assure the probability of each training instance covered by at least one pattern is higher than a user-specified threshold. Evaluation shows that this technology could improve the accuracy significantly.
- We conducted a comprehensive experiment using 30 public datasets under varying uncertain parameters. The empirical results validate that our algorithm outperforms two state-of-the-art algorithms significantly with 4% to 10% improvements in accuracy on average.

For the rest of this paper, we first introduce the related work in Section 2. Preliminaries are described in Section 3. The computation of expected confidence is provided in Section 4. Algorithm details are discussed in Section 5. The evaluation part is presented in Section 6. Our paper concludes in Section 7.

2. RELATED WORK

Various algorithms have been proposed for categorical data classification. Most of them could be classified into two types – the traditional rule-induction (or decision tree) based methods and the association-based methods. The rule-induction-based classifiers such as Ripper [7], C4.5 [18], FOIL [19], and CPAR [22] use heuristics like information-gain or gini index to grow the current rule. Sequential covering paradigm may also be adopted for speedup. While for association-based classifiers, efficient associative rule mining algorithms are first applied to find the complete set of candidate rules. A set of rules are selected later based on several covering paradigms and discrimination heuristics. Some typical examples include CBA [15] and CMAR [14]. [3] proposes a method recently using the discovered rules as SVM training features and achieves higher accuracy. [11] includes an application on associative classification using frequent itemset generators mined on stream data.

In recent years, several studies have been conducted on how to mine associative rules directly and effectively from the input database without costly feature selection step. HARMONY [21] is an instance-centric algorithm which mines directly for each instance a covering rule with the highest confidence. [8] proposes a method to discover top-k covering rules for the input gene expression data. Extended from their previous study in [3], the authors of [4] further introduced a feature-centered mining approach to generate discriminative patterns sequentially by incrementally eliminating training instances on a progressively shrinking FP-Tree. [9] also proposes a method to solve the same problem. Unlike DDPMine [4], it builds a decision tree to partition the data onto different nodes. Then at each node, one discriminative pattern is discovered directly to further divide its covering examples into purer subsets.

Uncertain data mining attracts much attention recently. Several research efforts focus on frequent itemset mining. [6] proposes the U-Apriori algorithm using expected support to find all the frequent itemsets on uncertain data. Later a probabilistic filter for earlier candidate pruning was further devised in [5]. The UF-Growth algorithm was proposed in [13]. Besides using expected support like U-Apriori, UF-Growth uses the FP-Tree [12] approach to avoid expensive candidate generation. [1] discusses the frequent pattern mining for uncertain datasets and shows how to extend a broad classes of algorithms to uncertain data. Trying to solve the inaccuracy in measuring frequentness using expected support, [2] proposes to use frequentness probability under possible worlds semantics and devises an efficient computing technology.

In recent years, several classification algorithms have been proposed for uncertain data too. [20] proposes to use decision tree for classifying uncertain numeric data where the value uncertainty is represented by multiple values forming a probability distribution function. Simultaneously uRule [17] tries to solve the problem using rule-based classifier extended from classical algorithm Ripper [7]. The authors also extended the entropy and information gain measure for uncertain data. Extending from classical decision tree algorithm C4.5 [18] and adopting the same discrimination measure of uRule, DTU [16] achieves close accuracy to uRule while runs much faster in most cases. Both uRule and DTU support uncertain numeric data and uncertain categorical data. However, the new measurements of probabilistic entropy and probabilistic information gain are only probabilistic cardinalities, which means that they do not have reasonable theoretical explanations.

3. PRELIMINARIES

3.1 The Uncertain Data Model

We adopt the same uncertain model of categorical data used in both [17] and [16]. For each input dataset, it is composed of a set of attributes $\mathbb{A}$. For each attribute $A_i \in \mathbb{A}(1 \leq i \leq |\mathbb{A}|)$, if it contains values which are uncertain it is called an uncertain attribute and is denoted by A_i^u, or else a certain attribute which is denoted by A_i^c. The set of all uncertain and the set of all certain attributes are denoted by $\mathbb{A}^u$ and $\mathbb{A}^c$, respectively. The value of attribute A_i in the jth instance is denoted by $a_{i,j}$. For an uncertain attribute A_i^u, its value in each instance is represented as a probability distribution function $pdf_{i,j}$ which records the possibility for each possible value in the categorical domain dom_{A_i} for A_i, instead of a single value for a certain attribute. Given the domain of A_i, $dom_{A_i} = \{v_1, \cdots, v_k, \cdots, v_n\}$, $pdf_{i,j}$ could be represented using a probability vector $P_{i,j} = \{p_1, \cdots, p_k, \cdots, p_n\}$ such that $P(a_{i,j} = v_k) = p_k$ and $\sum_{k=1}^{n} p_k = 1$. There is also a class label attribute C containing class label for each instance. c_j is used to denote the class label of the jth instance. In this paper, the class label attribute C is not included in the set of attributes $\mathbb{A}$.

Table 1 provides a toy example of an uncertain database about computer buying evaluation with one uncertain attribute on quality.

Evaluation	Price	Looking	Tech. Spec.	Quality
Unacceptable	+	-	/	{-: 0.8, /: 0.1, +: 0.1}
Acceptable	/	-	/	{-: 0.1, /: 0.8, +: 0.1}
Good	-	+	/	{-: 0.1, /: 0.8, +: 0.1}
Very Good	/	+	+	{-: 0.1, /: 0.1, +: 0.8}

Table 1: Example of an Uncertain Database (+: Good, /: Medium, -: Bad)

3.2 Frequent Itemset Mining

Since the foundation of associative classification is frequent pattern mining, we also introduce the definitions and notations related to frequent pattern mining. Specifically, we discuss frequent itemset mining on uncertain categorical data. Given a set of items $\mathbb{I}$ [1] in input database $input\_db$, an itemset x is defined as a subset of $\mathbb{I}$. A transaction t_j is defined as the set of values on each attribute A_i in the jth instance $A_{i,j}$ and a class label c_j. The complete set of transactions in any database $db \subseteq input\_db$ is denoted by T^{db} or simply T when db is clear in context.

Traditionally an itemset x is said to be supported by a transaction t_i if $x \subseteq t_i$. $|\{t_i | t_i \subseteq T \wedge x \subseteq t_i\}|$ is called the absolute support of x with respect to db, denoted by sup_x^{db} or sup_x in clear context, while $sup_x / \left|T^{input\_db}\right|$ is called the relative support. When it is clear, absolute support and relative support could be used interchangeably. We also use $sup_x^{db^c}$ or ${sup_x}^c$ to denote $|\{t_i | c_i = c \wedge t_i \subseteq T \wedge x \subseteq t_i\}|$, the support value of x under class c. x is said to be frequent iff $sup_x \geq sup_{min}$, where sup_{min} is a user specified minimum (absolute/relative) support threshold.

While on uncertain database, there exists a probability of $x \subseteq t_i$ when x contains at least one item of uncertain attribute, and the support of x is no longer a single value but a probability distribution function instead. In this paper we use expected support to represent the support value on uncertain data. The expected support $E(sup_x)$ of itemset x is defined as $E(sup_x) = \sum_{t \in T} P(x \subseteq t)$. For example, in Table 1 we have an itemset $\{/@Price, +@Quality\}$ with expected support of $0.1 + 0.8 = 0.9$. $E({sup_x}^c)$ could be defined similarly. The concept of frequent itemset is the same as on certain database. When the context is clear $E(sup_x)$ could also be denoted by sup_x for unified representation.

Finally, we summarize the notations used mostly through this paper in Table 2.

Notation	Description
$\mathbb{A}$ ($\mathbb{A}^c$ / $\mathbb{A}^u$)	Set of (certain / uncertain) attributes
A_i (A_i^c / A_i^u)	ith attribute in $\mathbb{A}$ (which is certain / uncertain)
$a_{i,j}$	Value of A_i in the jth instance
C	Class attribute
c_j	Class label in the jth instance
dom_{A_i}	Set of possible values on A_i
$\mathbb{I}$	Set of Items on the whole input database
T	Set of transactions in current database
t_j	jth transaction t in T
sup_x ($sup_x{}^c$)	(Expected) Support of itemset x (on class c)
sup_{min}	Minimum support threshold
$conf_x^c$	(Expected) Confidence of itemset x on class c
$E_i(conf_x{}^c$ / $sup_x{}^c)$	Part of Expect on $conf_x{}^c$ / $sup_x{}^c$ when $sup_x = i$
$E_{i,n}(conf_x{}^c$ / $sup_x{}^c)$	Part of Expect on $conf_x{}^c$ / $sup_x{}^c$ on the first n transactions when $sup_x = i$
$bound_i(conf_x{}^c)$	ith upper bound on $conf_x{}^c$
IS	Set of discovered candidate itemsets through the algorithm
$Ux@y$	With y uncertain attribute(s) under uncertain degree of $x\%$

Table 2: Summary of Notations

4. EXPECTED CONFIDENCE

In our algorithm uHARMONY we use expected confidence of a discovered itemset to measure its discrimination. Unlike probabilistic cardinalities like probabilistic entropy and probabilistic information gain used in [17] and [16] which may be not precise and are lack of theoretical explanations and statistical meanings, expected confidence is guarantied to be statistical meaningful in theory while providing relatively accurate measure of discrimination. However, the calculation of expected confidence is non-trivial and requires careful consideration. On uncertain database expected confidence $E(conf_x{}^c) = E(sup_x{}^c / sup_x)$ of itemset x on class c is not simply equal to $E(sup_x{}^c)/E(sup_x)$, although we have $conf_x{}^c = sup_x{}^c / sup_x$ on certain database. For example in Table 1, for itemset $x = \{-@Looking, -@Quality\}$ and class $c = Unacceptable$ we have $E(conf_x{}^c) = 1.0 \times (0.8 \times 0.9) + 0.5 \times (0.8 \times 0.1) = 0.76$ while $E(sup_x{}^c)/E(sup_x) = 0.8/(0.8+0.1) \approx 0.89$. Obviously, $E(conf_x{}^c)$ is not equal to $E(sup_x{}^c)/E(sup_x)$.

4.1 Definition of Expected Confidence

DEFINITION 1. *Given a set of transactions T and the set of possible worlds W with respect to T, the expected confidence of an itemset x on class c is*

$$E(conf_x{}^c) = \sum_{w_i \in W} {conf_{x,w_i}}^c \times P(w_i) = \sum_{w_i \in W} \frac{{sup_{x,w_i}}^c}{sup_{x,w_i}} \times P(w_i)$$

where $P(w_i)$ is the probability of world w_i. ${conf_{x,w_i}}^c$ is the respected confidence of x on class c in world w_i, while sup_{x,w_i} (${sup_{x,w_i}}^c$) is the respected support of x (on class c) in world w_i.

However, this formula could not be used directly to calculate the expected confidence, due to the extremely large number of possible world $|W|$. Actually, there are $O((1 + \prod_{A_k \in \mathbb{A}^u} |dom_{A_k}|)^{|T|})$

[1] Note that items appeared in different attributes are different even if they are identical literally.

possible worlds, where 1 stands for not taking the transaction while $\prod_{A_k \in \mathbb{A}^u} |dom_{A_k}|$ stands for combinations of values in uncertain attributes when taking the transaction. Hence, more efficient computation technology is needed.

4.2 Efficient Computation of Expected Confidence

In order to devise an efficient method for computing expected confidence, we first introduce a lemma.

LEMMA 1. *Since* $0 \leq {sup_x}^c \leq sup_x \leq |T|$, *we have:*

$$\begin{aligned} E({conf_x}^c) &= \sum_{w_i \in W} {conf_{x,w_i}}^c \times P(w_i) \\ &= \sum_{i=0}^{|T|} \sum_{j=0}^{i} \frac{j}{i} \times P(sup_x = i \wedge {sup_x}^c = j) \\ &= \sum_{i=0}^{|T|} \frac{E_i({sup_x}^c)}{i} = \sum_{i=0}^{|T|} E_i({conf_x}^c) \end{aligned}$$

, where $E_i({sup_x}^c)$ *and* $E_i({conf_x}^c)$ *denote the part of expected support and confidence of itemset* x *on class* c *when* $sup_x = i$.

Given $0 \leq n \leq |T|$, we define $E_n({sup_x}^c) = \sum_{i=0}^{|T|} E_{i,n}({sup_x}^c)$ as the expected support of x on class c on the first n transactions of T, and $E_{i,n}({sup_x}^c)$ as the part of expected support of x on class c with support of i on the first n transactions of T. We have the following theorem.

THEOREM 1. *Denoting* $P(x \subseteq t_i)$ *as* p_i *for each transaction* $t_i \in T$, *we have*

$$\begin{aligned} E_{i,n}({sup_x}^c) = p_n \times E_{i-1,n-1}({sup_x}^c) \\ + (1 - p_n) \times E_{i,n-1}({sup_x}^c) \end{aligned}$$

when $c_n \neq c$, *and*

$$\begin{aligned} E_{i,n}({sup_x}^c) = p_n \times E_{i-1,n-1}({sup_x}^c + 1) \\ + (1 - p_n) \times E_{i,n-1}({sup_x}^c) \end{aligned}$$

when $c_n = c$, *where* $1 \leq i \leq n \leq |T|$.

$$E_{i,n}({sup_x}^c) = 0$$

for $\forall n$ *where* $i = 0$, *or where* $n < i$.

PROOF. If $x \not\subseteq t_n$, we have $E_{i,n}({sup_x}^c) = E_{i,n}({sup_x}^c) + (1 - p_n) \times E_{i,n-1}({sup_x}^c)$ since both sup_x and ${sup_x}^c$ in each possible world remains the same. If $x \subseteq t_n$, there exist two situations:

When $c_n \neq c$, ${sup_x}^c$ in each possible world remains the same while $sup_x = sup_x + 1$ and we have $E_{i,n}({sup_x}^c) = E_{i,n}({sup_x}^c) + p_n \times E_{i-1,n-1}({sup_x}^c)$.

When $c_n = c$, ${sup_x}^c = {sup_x}^c + 1$ and $sup_x = sup_x + 1$ in each possible world. Hence we have $E_{i,n}({sup_x}^c) = E_{i,n}({sup_x}^c) + p_n \times E_{i-1,n-1}({sup_x}^c + 1)$ similarly.

Thus, the theorem is proved. □

Defining $P_{i,n}$ as the probability of x having support of i on the first n transactions of T, we have

$$\begin{aligned} E_{i,n}({sup_x}^c) &= p_n \times (E_{i-1,n-1}({sup_x}^c + 1)) \\ &\quad + (1 - p_n) \times E_{i,n-1}({sup_x}^c) \\ &= p_n \times (E_{i-1,n-1}({sup_x}^c) + P_{i-1,n-1}) \\ &\quad + (1 - p_n) \times E_{i,n-1}({sup_x}^c) \end{aligned}$$

when $c_n = c$, since we have:

$$E_{i-1,n-1}({sup_x}^c + 1) = E_{i-1,n-1}({sup_x}^c) + P_{i-1,n-1}$$

For computing $P_{i,n}$, we also introduce the following theorem.

THEOREM 2. *Denoting* $P(x \subseteq t_i)$ *as* p_i *for each transaction* $t_i \in T$, *we have*

$$P_{i,n} = p_n \times P_{i-1,n-1} + (1 - p_n) \times P_{i,n-1}$$

, where $1 \leq i \leq n \leq |T|$.

$$P_{i,n} = \begin{cases} 1 & for\ n = 0 \\ P_{i,n-1} \times (1 - p_n) & for\ 1 \leq n \leq |T| \end{cases}$$

where $i = 0$.

$$P_{i,n} = 0$$

where $n < i$.

PROOF. The proof is similar to that of Theorem 1. □

Now we could compute the expected confidence of itemset x on class c, since $E({conf_x}^c) = E_{|T|}({conf_x}^c) = \sum_{i=0}^{|T|} E_{i,|T|}({conf_x}^c)$. The whole computation is divided into $|T|+1$ steps with $E_{i,|T|}({conf_x}^c) = E_{i,|T|}({sup_x}^c)/i$ $(0 \leq i \leq |T|)$ computed in ith step. Figure 1 shows the computation process.

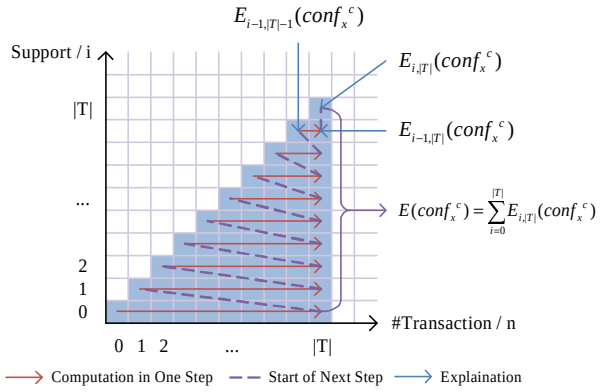


Figure 1: Computation Process of Expected Confidence

Finally, we prove the computation complexity of the expected confidence $E({conf_x}^c)$ in terms of time and space in Theorem 3.

THEOREM 3. *The computation of the expected confidence* $E({conf_x}^c)$ *requires at most* $O(|T|^2)$ *time and at most* $O(|T|)$ *space.*

PROOF. The number of computation iterations is bounded by the size of matrix depicted in Figure 1 containing $\sum_{i=0}^{|T|} |T| + 1 - i$ cells. Each cell represents a computation iteration performed in $O(1)$ time. Hence, the total computation requires $O(|T|^2)$ time.

Since only two rows in Figure 1 need to be reserved to complete the computation, only $O(|T|)$ space is required. Together with the $O(|T|)$ space to store $p_i = P(x \subseteq t_i)$ for each $t_i \in T$, totally $O(|T|)$ space is required to finish the computation. □

Comparing with the complexity using the definition of expected confidence, we could see that our computation strategy is very efficient and reduces the time complexity significantly.

4.3 Upper Bound of Expected Confidence

We further develop a theorem to compute the upper bound of expected confidence. Given a class c, if the upper bound of a pattern x is smaller than the maximum (expected) confidence of another pattern y we have mined previously having $x \subset y$, the computation could be stopped since x would never be provided as a more discriminative pattern with higher confidence value.

THEOREM 4. *Given* $1 \leq i \leq |T|$,

$$bound_i({conf_x}^c) = \sum_{k=0}^{i-1} E_{k,|T|}({sup_x}^c) \times (\frac{1}{k} - \frac{1}{i}) + \frac{E({sup_x}^c)}{i}$$

is an upper bound of $E({conf_x}^c)$.

PROOF. For $\forall i (1 \leq i \leq |T|)$, we have

$$E({conf_x}^c) = E_{|T|}({conf_x}^c)$$

$$= \sum_{k=0}^{i-1} \frac{E_{k,|T|}({sup_x}^c)}{k} + \sum_{k=i}^{|T|} \frac{E_{k,|T|}({sup_x}^c)}{k}$$

$$\leq \sum_{k=0}^{i-1} \frac{E_{k,|T|}({sup_x}^c)}{k} + \sum_{k=i}^{|T|} \frac{E_{k,|T|}({sup_x}^c)}{i}$$

$$= \sum_{k=0}^{i-1} \frac{E_{k,|T|}({sup_x}^c)}{k} + \sum_{k=0}^{|T|} \frac{E_{k,|T|}({sup_x}^c)}{i} - \sum_{k=0}^{i-1} \frac{E_{k,|T|}({sup_x}^c)}{i}$$

$$= \sum_{k=0}^{i-1} E_{k,|T|}({sup_x}^c) \times (\frac{1}{k} - \frac{1}{i}) + \frac{E({sup_x}^c)}{i}$$

$$= bound_i({conf_x}^c)$$

Hence $bound_i({conf_x}^c)$ is an upper bound of $E({conf_x}^c)$. □

COROLLARY 1. *For* $1 \leq i \leq |T|$, *we have:*

$$E({sup_x}^c) = bound_1({conf_x}^c) \geq \cdots \geq bound_i({conf_x}^c) \geq \cdots \geq bound_{|T|}({conf_x}^c) = E({conf_x}^c)$$

PROOF. It is easy to get that $bound_1({conf_x}^c) = E({sup_x}^c)$ and $bound_{|T|}({conf_x}^c) = E({conf_x}^c)$ using the definition. Since

$$bound_{i-1}({conf_x}^c) - bound_i({conf_x}^c)$$

$$= (\frac{1}{i-1} - \frac{1}{i}) \times (E({sup_x}^c) - \sum_{k=0}^{i-1} E_{k,|T|}({sup_x}^c)) \geq 0$$

for $1 < i \leq |T|$, the corollary is proved. □

For ith step in computing $E_{i,|T|}({conf_x}^c)$, we could compute the respective upper bound $bound_i({conf_x}^c)$ using $E_{k,|T|}({sup_x}^c)$ $(0 \leq k \leq i-1 < i \leq |T|)$ which all have been computed previously. Actually since

$$bound_i({conf_x}^c) = bound_{i-1}({conf_x}^c) - (\frac{1}{i-1} - \frac{1}{i}) \times (E({sup_x}^c) - \sum_{k=0}^{i-1} E_{k,|T|}({sup_x}^c))$$

, we could compute $bound_i({conf_x}^c)$ more efficiently with previous upper bound $bound_{i-1}({conf_x}^c)$.

With the corollary, we know that the upper bound in the current step would be smaller than the one in the previous step and is more close to the value of expected confidence. Figure 2 illustrates the computation process of expected confidence using upper bound.

We also provide an example on computation of upper bound shown in Figure 3, conducted on one of the evaluation datasets. (Details of the evaluation parameters could be found in Section 6.)

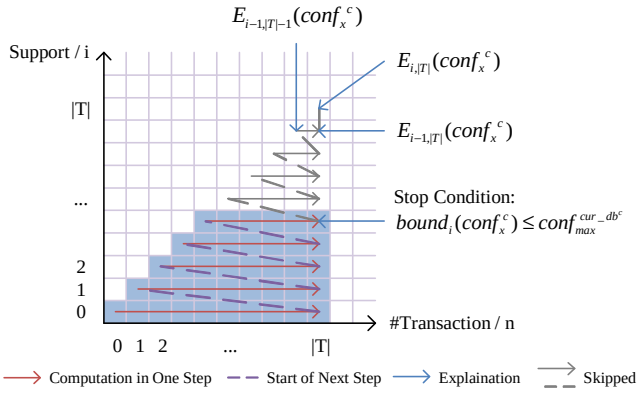


Figure 2: Computation Process of Expected Confidence using Upper Bound ($\underline{conf_{max}^{cur\_db^c}}$: Maximum (expected) confidence in current database on current class)

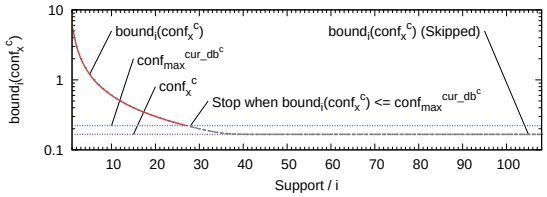


Figure 3: Example on Upper Bound Computation (car, U10@1, $sup_{min} = 0.01$, $x = \{vhigh@buying, 3@doors, med@safety\}$, $c = acc$)

5. ALGORITHM DETAILS

Now we give the details of the whole algorithm of uHARMONY. First we present the frequent itemset mining algorithm on uncertain categorical data. Then we will discuss the instance covering technology using minimum cover probability. Finally, details of using either SVM classifier or rule-based classifier are presented.

5.1 Mining Algorithm

The framework of our frequent itemset mining algorithm is similar to that of HARMONY [21]. However, there also exist significant differences. First, the infrequent pattern pruning technology used in HARMONY is no longer applicable on uHARMONY. Since on uncertain data even if the expected support is equal to or higher than the minimum support, there still remain situations where the itemset support is less than the minimum support. Second, items on uncertain attributes need carefully consideration since on uncertain attributes the pattern searching space could not shrink when the current prefix pattern gets extended.

Algorithm 1 gives the details of our frequent itemset mining algorithm in uHARMONY. Note that before running the algorithm, we need to first sort the attributes to place all the certain attributes before uncertain attributes. Hence when we traverse the attributes for extending items, uncertain attributes which would not help shrink the pattern searching space will be encountered at last. This helps to speedup the algorithm. $calcExpConf$ is the function of expected confidence computation of current itemset x with upper bound computation used. Function $coverInstances$ is used to cover instances with the current itemset, whose details will be provided later in Section 5.2. The variable IS is used throughout the algorithm to maintain all the discovered candidate itemsets as the output.

Algorithm 1: Itemset Mining Algorithm of uHARMONY

Function: $mine(T, index, x, IS, \{{conf_{max}}^c | c \in dom_C\})$
Input: Current set of transactions, Current attribute index, Current itemset pattern, Set of discovered candidate itemsets through the algorithm, Set of maximum confidences for each class [2]

```
1  if A_index ∉ 𝔸^u then
2  |  foreach c ∈ dom_C do
3  |  |  conf_x^c ← sup_x^c / sup_x;
4  else
5  |  foreach c ∈ dom_C do
6  |  |  conf_x^c ← calcExpConf(T, x, conf_max^c);
7  coveredNum ← 0;
8  foreach c ∈ dom_C do
9  |  if conf_x^c > conf_max^c then
10 |  |  coveredNum^c ← coverInstances(T, x, IS);
11 |  |  conf_max^c ← conf_x^c;
12 |  coveredNum ← coveredNum + coveredNum^c;
13 if coveredNum > 0 then
14 |  IS ← IS ∪ {x};
15 for A_index* ∈ {A_index* | A_index* ∈ 𝔸 ∧ index < index*}
   do
16 |  for i ∈ dom_A_index* do
17 |  |  x* ← x ∪ {i@A_index*};
18 |  |  if sup_x* ≥ sup_min then
19 |  |  |  if A_index* ∉ 𝔸^u then
20 |  |  |  |  T* ← {t | t ∈ T ∧ x* ⊆ t};
21 |  |  |  else
22 |  |  |  |  T* ← T;
23 |  |  |  mine(T*, index*, x*, IS, {conf_max^c | c ∈
   |  |  |  dom_C});
```

5.2 Instance Covering Strategy

HARMONY adopts a simple strategy for instance covering. It tries to find one most discriminative covering pattern with the highest confidence for each instance. However, this strategy is not practical for uncertain data, since each itemset has a probability being contained in the instance. If we just find the itemset with the highest confidence for each instance, the probability of the instance being covered could be very low.

In uHARMONY, we propose an instance covering strategy by applying a threshold of minimum cover probability $coverProb_{min}$. We try to assure that the probability of each instance not covered by anyone itemset is less than $1 - coverProb_{min}$. For each instance t with class label c, we sort the covering itemsets in the descending order with respect to confidence on class c. When a new itemset is discovered we insert it into that list $IS_t = \{x | x \in IS \wedge x \subseteq t\} \subseteq IS$. Then only the first k itemsets $IS_t[1,k] \subseteq IS_t$ with $\prod_{x \in IS_t[1,k]}(1 - P(x \subseteq t)) < 1 - coverProb_{min}$ $(1 \leq k \leq |IS_t|)$ are selected to remain in the list. For each removed itemsets in $IS_t - IS_t[1,k]$, we decrease its total covered number on all instances, and remove it from the candidate itemset set IS when its total covered number reaches 0.

5.3 Classification Algorithm

The mined frequent itemsets could be used either as training features of SVM classifier or as classification rules of rule-based classifier. In this section, we will discuss the details of algorithms classifying instances using SVM classifier and rule-based classifier.

5.3.1 SVM Classifier

It is very simple to convert the mined patterns to training features of SVM classifier. Each pattern is a feature with the feature weight for an instance as the probability of the instance containing the itemset. According to the accuracy evaluation, this approach could provide 4% to 10% improvement on average in terms of classification accuracy comparing with two state-of-the-art algorithms.

5.3.2 Rule-based Classifier

To use the mined itemsets as classification rules, we adopt the similar classifier construction algorithm of HARMONY. For each test instance we just sum up the product of the confidence of each itemset on each class and the probability of the instance containing the itemset. The class with the largest value is the predicted class of the instance. Although this algorithm is simple, it is effective in classification. Accuracy evaluation shows that this algorithm outperforms two state-of-the-art baselines too.

6. EVALUATION RESULTS

In this section, we will present the evaluation results of our algorithm uHARMONY. Our algorithm is implemented in C#. All the experiments were conducted on a computer with Intel Core Duo 2 E6550 CPU (2.33GHz) and 2GB memory installed.

6.1 Datasets

Due to the unavailability of public uncertain categorical datasets, we conducted our evaluation on 30 public certain datasets from UCI Machine Learning Repository [3], by converting them into uncertain ones with varying uncertain degree (defined as the probability of each instance on each uncertain attribute taking values other than the original value in the certain dataset) over different attribute numbers. Details of the converting procedure on categorical datasets could be found in [17]. Missing values appearing in the uncertain attributes are converted to uncertain ones with the same probability for each possible value. For some datasets containing not only categorical but also continuous attributes, discretization was applied using the entropy method proposed in [10] and adopted in [15] using weka [4]. Attributes containing unique identifier for each instance and instances with missing class label have also been removed. Detailed characteristics of datasets are listed in Table 3. We could see that those datasets cover most of the common areas.

6.2 Classification Accuracy Evaluation

Now we give the evaluation results of our algorithm on classification accuracy, comparing with two state-of-the-art classification algorithms uRule [17] and DTU [16], which are extended from the famous rule-based classifier Ripper [7] and decision tree classifier C4.5 [18]. To our best knowledge, the two algorithms are the only available algorithms supporting uncertain categorical data. For training and classifying of SVM, svm^{light} and svm^{multiclass} are used for 2-class and multi-class situations, respectively. [5]

[2] A new set is created each time to avoid overwriting previous values.

[3] All the datasets and their detailed descriptions could be found at `http://archive.ics.uci.edu/ml/index.html`.

[4] The software is available at `http://www.cs.waikato.ac.nz/ml/weka/`.

[5] They are available at `http://www.cs.cornell.edu/people/tj/svm%5Flight/`. For svm^{multiclass}, the parameter of trade-off ("-c") is set to 1000.

Dataset	#Instance	#Attribute	#Class	Area
australian	690	14	2	Financial
balance	635	4	3	Social
bands	539	38	2	Physical
breast	699	9	2	Life
bridges-v1	106	11	6	N/A
bridges-v2	106	10	6	N/A
car	1728	6	4	N/A
contraceptive	1473	9	3	Life
credit	690	15	2	Financial
echocardiogram	131	12	2	Life
flag	194	28	8	N/A
german	1000	19	2	Financial
heart	920	13	5	Life
hepatitis	155	19	2	Life
horse	368	27	2	Life
monks-1	556	6	2	N/A
monks-2	601	6	2	N/A
monks-3	554	6	2	N/A
mushroom	8124	22	2	Life
pima	768	8	2	Life
postoperative	90	8	3	Life
promoters	106	57	2	Life
spect	267	22	2	Life
survival	306	3	2	Life
ta_eval	151	5	3	N/A
tic-tac-toe	958	9	2	Game
vehicle	846	18	4	N/A
voting	435	16	2	Social
wine	178	13	3	Physical
zoo	101	16	7	Life

Table 3: Dataset Characteristics

6.2.1 Accuracy Comparison

Table 4 shows the evaluation results in terms of classification accuracy, with 1, 2, or 4 uncertain attribute(s) using SVM classifier. The upper part of the table is for uncertain degree of 10%, while the lower part is for uncertain degree of 20%. Ux@y denotes the datasets with y uncertain attribute(s) under uncertain degree x%; N/A^{time} denotes that the algorithm did not finish in an acceptable time; N/A^{mem} denotes that the algorithm ran out of memory. All the uncertain attributes are selected from all the non-class attributes with the highest information gain values. All the experiments were conducted using 10-fold cross validation. For the same dataset, the same parameter values like sup_{min} were used. Values in bold stand for the highest accuracies for the corresponding datasets. We could easily find that on most situations, our algorithm uHARMONY outperforms the state-of-the-art algorithms DTU and uRule significantly, with up to 28% improvement on dataset *balance*, and on average 4% to 10% improvements on all the 30 datasets under varying uncertain parameters. Besides, uHARMONY is also more memory-efficient than both DTU and uRule, especially on datasets whose uncertain attributes have many possible values. For example, on dataset *bands*, DTU and uRule ran out of the memory and could not finish properly in situations with more uncertain attributes. Actually, the experiments with 8 uncertain attributes and under uncertain degree of 40% show the same results.

Since uHARMONY supports the rule-based classifier besides SVM classifier. We also conducted experiments on all the datasets with four uncertain attributes under uncertain degree of 10%. From Table 5 we could see that although uHARMONY^{rule} (stands for the rule-based uHARMONY classifier) could not outperform uHARMONY (stands for the SVM-based uHARMONY classifier), it provides near 1% to 4% higher accuracy on average than DTU and uRule, which validates the effectiveness of our algorithm. Experimental results with other uncertain attributes and under other uncertain degrees also show similar results.

Dataset	uHARMONY^{rule}	DTU	uRule	sup_{min}
australian	**85.37**	83.6232	84.3478	0.05
balance	**89.3**	56.32	62.88	0.1
bands	**58.63**	N/A^{mem}	N/A^{mem}	0.25
breast	65.52	91.2732	**94.5637**	0.05
bridges-v1	**62**	59.434	55.6604	0.1
bridges-v2	62.2	**64.1509**	57.5472	0.1
car	**77.72**	70.0231	70.0231	0.01
contraceptive	47.59	**50.1018**	44.2634	0.01
credit	**85.95**	84.3478	74.3478	0.1
echocardiogram	**93.29**	92.3664	87.0229	0.1
flag	52.42	**59.2784**	44.8454	0.1
german	69.6	**72.3**	70.1	0.1
heart	**56.64**	53.0435	52.3913	0.25
hepatitis	**82.52**	80	79.3548	0.1
horse-colic	82.88	**85.3261**	N/A^{time}	0.1
monks-1	**91.36**	74.6403	70.6835	0.1
monks-2	65.72	**65.7238**	**65.7238**	0.1
monks-3	**96.4**	79.9639	68.0505	0.1
mushroom	97.45	**100**	99.9877	0.1
pima	65.11	65.1042	**67.3177**	0.25
postoperative	69.75	**70**	**70**	0.25
promoters	69	**71.6981**	61.3208	0.25
spect	80.19	79.0262	**81.6479**	0.1
survival	**73.53**	73.5294	72.549	0.1
ta_eval	45.04	**48.3444**	33.7748	0.01
tic-tac-toe	76.2	72.6514	**81.524**	0.05
vehicle	63.44	**64.7754**	N/A^{mem}	0.01
voting	92.86	94.4828	**94.9425**	0.25
wine	**51.11**	42.1348	41.573	0.01
zoo	88.76	**92.0792**	89.1089	0.1
Average	**73.2517**	72.2670	69.4649	

Table 5: Accuracy (in %) Comparison of U10@4 for uHarmony using Rule-based Classifier

6.2.2 Sensitivity Test

We also evaluated uHARMONY on two popular datasets *breast* and *wine*, under varying minimum supports and varying minimum cover probabilities. Figure 4 shows the results with varying minimum supports. We could find that the sup_{min} is crucial to the accuracy. If a too high sup_{min} is specified, few patterns could be found. But a too low sup_{min} could also hurt the accuracy. In our experiments, sup_{min} ranging from 0.01 to 0.25 were chosen. On many datasets, a minimum support of 0.1 could provide the best results. However, the accuracy is insensitive to sup_{min} under varying uncertain degree and uncertain attribute number. This means we just need to choose one proper sup_{min} which will work well for all the uncertain datasets derived from the same certain dataset. We also tested the algorithm sensitivity against minimum cover probability. The results are shown in Figure 5. We see on average a minimum cover probability of 90% could provide the best results (Minimum cover probability of 90% is used during all this paper if not specified explicitly.) Note that a minimum cover probability of 0% works as selecting at most one pattern for each instance like HARMONY.

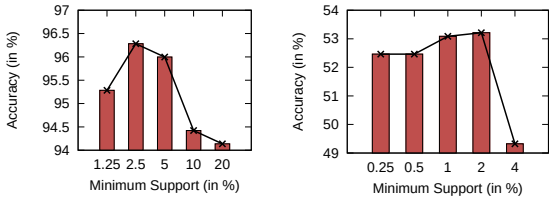
6.3 Runtime Efficiency Evaluation

6.3.1 Efficiency Test

Figure 6 provides the evaluation results of algorithm efficiency on six of the datasets in terms of both running time and memory us-

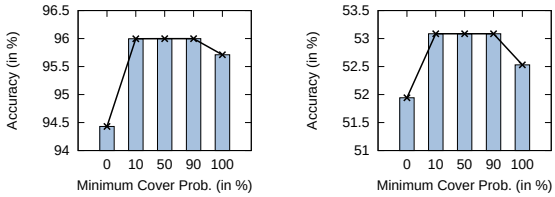
Dataset	uHARMONY	DTU	uRule	uHARMONY	DTU	uRule	uHARMONY	DTU	uRule	sup_{min}
	U10@1			U10@2			U10@4			
australian	**86.542**	85.942	84.2029	**86.109**	86.087	85.2174	**86.821**	83.6232	84.3478	0.05
balance	**90.577**	65.12	69.92	**90.736**	66.08	71.2	**90.736**	56.32	62.88	0.1
bands	**69.939**	65.1206	N/A^{mem}	**69.939**	N/A^{mem}	N/A^{mem}	**68.609**	N/A^{mem}	N/A^{mem}	0.25
breast	**95.998**	91.1302	93.7053	**94.569**	91.2732	93.5622	93.999	91.2732	**94.5637**	0.05
bridges-v1	**65.068**	51.8868	59.434	**65.068**	53.7736	60.3774	**67.012**	59.434	55.6604	0.1
bridges-v2	64.143	**65.0943**	60.3774	63.032	**67.9245**	55.6604	64.144	**64.1509**	57.5472	0.1
car	88.66	**91.1458**	85.5324	**89.82**	72.9745	69.0394	**87.907**	70.0231	70.0231	0.01
contraceptive	**51.797**	50.1018	44.2634	49.895	**51.5954**	43.9919	49.426	**50.1018**	44.2634	0.01
credit	85.517	**87.3913**	84.2029	85.517	**86.9565**	86.5217	**86.382**	84.3478	74.3478	0.1
echocardiogram	**93.289**	92.3664	92.3664	**92.52**	92.3664	90.8397	**92.52**	92.3664	87.0229	0.1
flag	65.667	**67.0103**	62.3711	62.544	**65.9794**	59.7938	**59.868**	59.2784	44.8454	0.1
german	**72.8**	69	70.6	72.7	70.7	70.5	**72.9**	**72.3**	70.1	0.1
heart	**57.943**	54.1304	50	**58.166**	54.2391	50.9783	**57.854**	53.0435	52.3913	0.25
hepatitis	**83.772**	79.3548	78.0645	**83.772**	79.3548	77.4194	79.264	**80**	79.3548	0.1
horse	86.108	85.3261	**87.5**	86.108	85.3261	**87.2283**	**86.1**	85.3261	N/A^{time}	0.1
monks-1	**100**	97.8417	95.6835	**100**	74.6403	97.1223	**100**	74.6403	70.6835	0.1
monks-2	**69.554**	65.7238	64.2263	**72.187**	65.7238	63.0616	**76.7**	65.7238	65.7238	0.1
monks-3	96.402	**98.917**	98.1949	**96.402**	79.9639	79.9639	**96.402**	79.9639	68.0505	0.1
mushroom	99.618	**100**	**100**	99.766	**100**	**100**	99.717	**100**	99.9877	0.1
pima	**68.106**	65.1042	68.099	**68.106**	65.1042	67.4479	**68.106**	65.1042	67.3177	0.25
postoperative	**69.614**	68.8889	68.8889	69.614	**70**	**70**	69.198	**70**	**70**	0.25
promoters	**87.166**	76.4151	71.6981	**86**	73.5849	69.8113	**77.667**	71.6981	61.3208	0.25
spect	**85.846**	79.7753	83.8951	**86.956**	79.7753	83.5206	**85.474**	79.0262	81.6479	0.1
survival	73.529	**73.5294**	70.2614	73.529	**73.5294**	70.915	73.529	**73.5294**	72.549	0.1
ta_eval	**54.298**	44.3709	40.3974	**52.263**	48.3444	34.4371	45.746	**48.3444**	33.7748	0.01
tic-tac-toe	**100**	85.6994	97.7035	**100**	78.81	96.6597	**99.895**	72.6514	81.524	0.05
vehicle	**65.323**	64.0662	N/A^{mem}	**65.441**	64.0662	N/A^{mem}	64.02	**64.7754**	N/A^{mem}	0.01
voting	**96.099**	93.7931	91.954	**94.475**	93.5632	90.8046	**95.406**	94.4828	94.9425	0.25
wine	**53.086**	39.8876	43.2584	**53.601**	39.8876	39.3258	**50.525**	42.1348	41.573	0.01
zoo	**93.954**	92.0792	89.1089	**92.074**	90.099	90.099	**93.045**	92.0792	89.1089	0.1
Average	**79.0138**	74.8738	75.2111	**78.6970**	73.1629	73.4107	**77.9657**	72.2670	69.4649	
	U20@1			U20@2			U20@4			
australian	**85.384**	84.4928	78.8406	**85.535**	84.058	80.5797	**88.416**	77.971	79.1304	0.05
balance	**91.212**	66.08	69.92	**91.212**	66.24	71.2	**91.527**	56.32	63.04	0.1
bands	**69.939**	65.1206	N/A^{mem}	**69.939**	N/A^{mem}	N/A^{mem}	**69.73**	N/A^{mem}	N/A^{mem}	0.25
breast	**94.71**	91.2732	93.8484	**94.853**	91.2732	93.7053	94.287	90.9871	**94.5637**	0.05
bridges-v1	**65.068**	51.8868	59.434	**65.068**	53.7736	60.3774	**65.532**	60.3774	55.6604	0.1
bridges-v2	64.143	**65.0943**	60.3774	63.033	**67.9245**	55.6604	**64.087**	57.5472	55.6604	0.1
car	88.953	**90.7986**	81.25	**88.26**	70.0231	70.0231	**82.413**	70.0231	70.0231	0.01
contraceptive	**51.525**	50.9165	44.1276	50.506	**50.9165**	43.5166	**49.422**	47.0468	44.1276	0.01
credit	**85.361**	84.3478	80.7246	**85.217**	84.058	81.7391	**86.959**	83.3333	73.913	0.1
echocardiogram	**93.289**	92.3664	92.3664	**92.52**	86.2595	79.3893	**92.52**	77.8626	87.0229	0.1
flag	65.667	**67.0103**	63.4021	62.544	**65.9794**	59.7938	**57.689**	49.4845	40.2062	0.1
german	**72.5**	69.3	69.4	**72.8**	69.7	68.3	**72.8**	71.2	68.3	0.1
heart	**56.965**	54.1304	48.913	**57.627**	54.2391	50.8696	**58.397**	53.3696	50.4348	0.25
hepatitis	**83.772**	79.3548	78.0645	**83.772**	79.3548	77.4194	**81.3**	79.3548	79.3548	0.1
horse	86.108	85.3261	**87.5**	86.108	85.3261	**87.5**	**86.633**	85.3261	N/A^{time}	0.1
monks-1	**100**	95.5036	95.3237	**100**	74.6403	90.8273	**100**	74.6403	69.2446	0.1
monks-2	**75.028**	65.7238	64.7255	**74.852**	65.7238	63.0616	**73.205**	65.7238	65.7238	0.1
monks-3	96.402	**98.917**	97.8339	**96.402**	79.9639	79.9639	**96.402**	79.9639	70.5776	0.1
mushroom	99.79	**100**	**100**	99.74	**100**	**100**	99.975	**100**	**100**	0.1
pima	**68.106**	65.8854	68.099	**68.106**	65.1042	67.4479	**68.106**	65.1042	67.3177	0.25
postoperative	**69.864**	68.8889	68.8889	69.614	**70**	**70**	69.614	**70**	**70**	0.25
promoters	**85.667**	76.4151	71.6981	**87**	75.4717	70.7547	**81.333**	59.434	54.717	0.25
spect	**85.489**	80.8989	83.1461	**85.859**	80.8989	83.8951	**84.762**	75.6554	78.6517	0.1
survival	73.529	**73.5294**	70.5882	73.529	**73.5294**	70.915	73.529	**73.5294**	72.8758	0.1
ta_eval	**52.965**	44.3709	41.0596	**48.298**	43.7086	37.0861	**43.798**	39.7351	32.4503	0.01
tic-tac-toe	**100**	84.3424	98.1211	**100**	78.81	96.2422	**98.747**	72.8601	78.7056	0.05
vehicle	**65.323**	64.0662	N/A^{mem}	**65.441**	64.0662	N/A^{mem}	64.02	**64.539**	N/A^{mem}	0.01
voting	**94.258**	91.7241	90.5747	**94.708**	93.3333	91.0345	**95.174**	91.954	89.1954	0.25
wine	**53.641**	39.8876	43.2584	**53.602**	39.8876	39.8876	**51.635**	42.1348	41.573	0.01
zoo	**93.954**	92.0792	88.1188	**92.074**	90.099	90.099	**93.045**	92.0792	89.1089	0.1
Average	**78.9537**	74.6577	74.6287	**78.6073**	72.5642	72.5460	**77.8352**	69.9157	68.2066	

Table 4: Accuracy (in %) Comparison of U{10,20}@{1,2,4}



(a) breast (b) wine

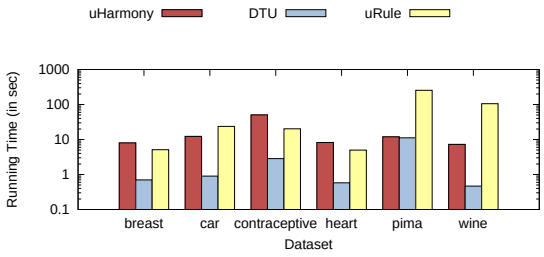
Figure 4: Accuracy Evaluation of U10@1 w.r.t. Minimum Support



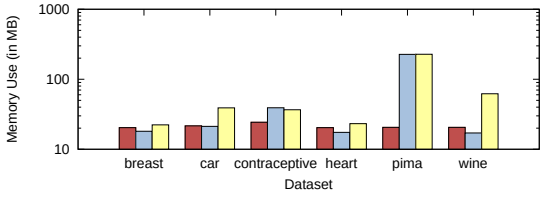
(a) breast (b) wine

Figure 5: Accuracy Evaluation of U10@1 w.r.t. Minimum Cover Prob.

age. sup_{min} values are the same as those in Table 4. Note that the running time includes both the classifier construction time and classifier classification time. For example for uHARMONY, it includes the time of mining patterns, converting to SVM input, SVM training and SVM classifying. All values are measured on all 10-fold cross validations. We could see that DTU is the fastest algorithm in all cases . Note that for uHARMONY, SVM training and classifying take a great amount of running time for more than half. Well for memory usage, uHARMONY consumes almost the fewest and the stablest in most cases, while uRule always consumes the most. uRule even ran out of the available memory on several datasets.



(a) Running Time (in sec)

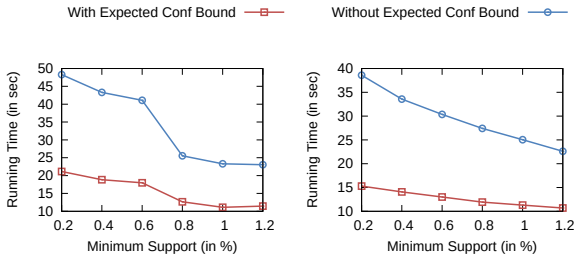


(b) Memory Use (in MB)

Figure 6: Classification Efficiency Evaluation of U10@1

6.3.2 *Effectiveness of the Expected Confidence Upper Bound*

We evaluated the effectiveness of the expected confidence upper bound. Figure 7 shows the results on two of the most popular datasets, *car* and *heart*, either with the expected confidence upper bound or without the upper bound. The effectiveness of adopting the expected confidence upper bound could be seen easily.

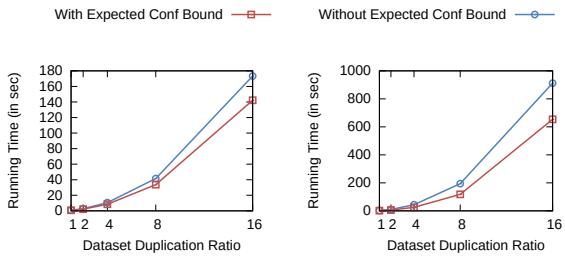


(a) car (b) heart

Figure 7: Running Time Evaluation of U10@4

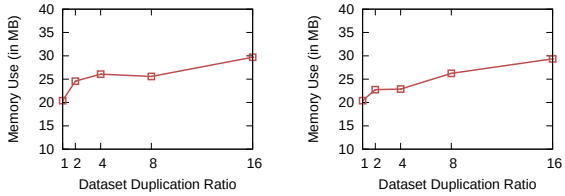
6.3.3 *Scalability Test*

Finally, we evaluated the scalability of our algorithm. Results in terms of running time are listed in Figure 8. It is obvious that using the expected confidence bound offers better efficiency in running time. Figure 9 also shows the results in terms of memory usage. We could see that the increase of memory usage is smaller than 10 MB even when the size of dataset increases 16 times. Hence, our algorithm is also efficient in terms of memory usage. Note that since using the upper bound on computation of expected confidence and not using the upper bound consume nearly the same amount of memory, only the results of using the upper bound are shown.



(a) car (sup_{min} = 0.01) (b) heart (sup_{min} = 0.01)

Figure 8: Scalability Evaluation (U10@1, Running Time)



(a) car (sup_{min} = 0.01) (b) heart (sup_{min} = 0.01)

Figure 9: Scalability Evaluation (U10@1, Memory Usage)

7. CONCLUSIONS

In this paper we propose a novel algorithm to solve the classification problem on uncertain categorical data. To achieve both high classification accuracy and efficiency, we try to mine frequent patterns directly from uncertain data using expected confidence as discrimination measure. The costly feature selection is avoided, and effective method for calculation of expected confidence is also devised. Empirical results show that our algorithm outperforms the state-of-the-art algorithms significantly with 4% to 10% improvements on average in terms of accuracy on 30 datasets under varying uncertain degrees and uncertain attribute numbers.

8. ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under grant No. 60873171, National Basic Research Program of China under Grant No. 2006CB303103, and the Program for New Century Excellent Talents in University under Grant No. NCET-07-0491, State Education Ministry of China.

9. REFERENCES

[1] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38, Paris, France, 2009. ACM.

[2] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128, Paris, France, 2009. ACM.

[3] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 716–725, Istanbul, Turkey, 2007. IEEE.

[4] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *Proceedings of the 24th International Conference on Data Engineering*, pages 169–178, Cancún, México, 2008. IEEE.

[5] C. K. Chui and B. Kao. A decremental approach for mining frequent itemsets from uncertain data. In *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2008*, pages 64–75, Osaka, Japan, 2008. Springer.

[6] C. K. Chui, B. Kao, and E. Hung. Mining frequent itemsets from uncertain data. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2007*, pages 47–58, Nanjing, China, 2007. Springer.

[7] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 115–123, Tahoe City, California, USA, 1995.

[8] G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 670–681, Baltimore, Maryland, USA, 2005. ACM.

[9] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 230–238, Las Vegas, Nevada, USA, 2008. ACM.

[10] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI-93, Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 1022–1029, Chambery, France, 1993.

[11] C. Gao and J. Wang. Efficient itemset generator discovery over a stream sliding window. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009*, pages 355–364, Hong Kong, China, 2009. ACM.

[12] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[13] C. K.-S. Leung, C. L. Carmichael, and B. Hao. Efficient mining of frequent patterns from uncertain data. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pages 489–494, Omaha, Nebraska, USA, 2007. IEEE Computer Society.

[14] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, San Jose, California, USA, 2001. IEEE Computer Society.

[15] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourteen ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York City, New York, USA, 1998.

[16] B. Qin, Y. Xia, and F. Li. Dtu: A decision tree for uncertain data. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2009*, pages 4–15, Bangkok, Thailand, 2009. Springer.

[17] B. Qin, Y. Xia, S. Prabhakar, and Y.-C. Tu. A rule-based classification algorithm for uncertain data. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009*, pages 1633–1640, Shanghai, China, 2009. IEEE.

[18] J. R. Quinlan. *C4.5: Programs for Machine Learning, Morgan Kaufmann: 1 ed.* 1993.

[19] J. R. Quinlan and R. M. Cameron-Jones. Foil: A midterm report. In *Machine Learning: ECML-93, European Conference on Machine Learning*, pages 3–20, Vienna, Austria, 1993. Springer.

[20] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee. Decision trees for uncertain data. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009*, pages 441–444, Shanghai, China, 2009. IEEE.

[21] J. Wang and G. Karypis. On mining instance-centric classification rules. *IEEE Trans. Knowl. Data Eng.*, 18(11):1497–1511, 2006.

[22] X. Yin and J. Han. Cpar: Classification based on predictive association rules. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA, 2003. SIAM.