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.

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 [4], C4.5 [18], FOIL [19], and CPAR [22] use heuristics like information-gain or gain 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 [6], 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 DDP-Mine [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 [4]. 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 \( \mathcal{A} \). For each attribute \( A_i \in \mathcal{A} \) with \( 1 \leq i \leq |\mathcal{A}| \), if it contains values which are uncertain it is called an uncertain attribute. We also denote by \( \mathcal{A}^u \) or else a certain attribute which is denoted by \( \mathcal{A}_c \). The set of all certain and the set of all certain attributes are denoted by \( \mathcal{A}_c \) and \( \mathcal{A}^u \), respectively. The value of attribute \( A_i \) in the \( j \)th instance is denoted by \( a_{i,j} \).

For an uncertain attribute \( A_i \), 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 \( A_i, dom_{A_i} = \{v_1, \cdots, v_n\}, pdf_{i,j} \) could be represented using a probability vector \( P_{i,j} = \{p_{1,i}, \cdots, p_{k,i}, \cdots, p_{n,i}\} \) 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 \( j \)th instance. In this paper, the class label attribute \( C \) is not included in the set of attributes \( \mathcal{A} \).

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

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Price</th>
<th>Looking</th>
<th>Tech. Spec.</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unacceptable</td>
<td>+</td>
<td>-</td>
<td>/</td>
<td>{0.8, 0.1, 0.1}</td>
</tr>
<tr>
<td>Acceptable</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>{0.1, 0.8, 0.1}</td>
</tr>
<tr>
<td>Good</td>
<td>-</td>
<td>+</td>
<td>/</td>
<td>{0.1, 0.8, 0.1}</td>
</tr>
<tr>
<td>Very Good</td>
<td>+</td>
<td>+</td>
<td>/</td>
<td>{0.1, 0.1, 0.8}</td>
</tr>
</tbody>
</table>

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 \( I \) in input database \( input_{db} \), an itemset \( x \) is defined as a subset of \( I \). A transaction \( t_i \) is defined as the set of values on each attribute \( A_i \) in the \( j \)th 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 \( x \subseteq t_i \). \( \{|t_i| \in T \land x \subseteq t_i\} \) is called the absolute support of \( x \) with respect to \( db \), denoted by \( sup_{db} \) or \( sup_x \) in clear context, while \( sup_{db} / |input_{db}| \) is called the relative support. When it is clear, absolute support and relative support could be used interchangeably. We also use \( sup_{db}^0 \) or \( sup_x^c \) to denote \( \{|t_i| \in T \land x \subseteq t_i\} \), 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 \( E(sup_x) = \sum_{t_i \in T} P(x \subseteq t_i) \). 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.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{A} ) ( (\mathcal{A}^u / \mathcal{A}_c) )</td>
<td>Set of (certain/uncertain) attributes</td>
</tr>
<tr>
<td>( A_i (A_i^u / A_i^c) )</td>
<td>( i )th attribute in ( \mathcal{A} ) (which is certain/uncertain)</td>
</tr>
<tr>
<td>( C )</td>
<td>Class attribute</td>
</tr>
<tr>
<td>( c_j )</td>
<td>Class label of the ( j )th instance</td>
</tr>
<tr>
<td>( dom_{A_i} )</td>
<td>Set of possible values on ( A_i )</td>
</tr>
<tr>
<td>( t )</td>
<td>Set of items on the whole input database</td>
</tr>
<tr>
<td>( T )</td>
<td>Set of transactions in current database</td>
</tr>
<tr>
<td>( sup_x (sup_x^c) )</td>
<td>Minimum support threshold</td>
</tr>
<tr>
<td>( con_x ) ( con_x^c )</td>
<td>( {} /@Looking, -@Quality )</td>
</tr>
<tr>
<td>( sup_{min} )</td>
<td>( {} /@Price, +@Quality )</td>
</tr>
<tr>
<td>( IS )</td>
<td>Set of discovered candidate itemsets through the algorithm</td>
</tr>
<tr>
<td>( U \times @y )</td>
<td>( x ) with uncertain attribute(s) under uncertain degree of ( x % )</td>
</tr>
</tbody>
</table>

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 guaranteed 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 \( E(con_x^c) = E(sup_x^c/sup_x) \) of itemset \( x \) on class \( c \) is not simply equal to \( E(sup_x^c)/E(sup_x) \), although we have \( con_x^c = sup_x^c/sup_x \) on certain database. For example in Table 1 for itemset \( x = \{\} /@Looking, -@Quality \) and class \( c = Unacceptable \) we have \( E(con_x^c) = 0 \) while \( E(sup_x^c) = 0.76 \) if \( sup_x = 0.8 \).

expected confidence 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.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{A} ) ( (\mathcal{A}^u / \mathcal{A}_c) )</td>
<td>Set of (certain/uncertain) attributes</td>
</tr>
<tr>
<td>( A_i (A_i^u / A_i^c) )</td>
<td>( i )th attribute in ( \mathcal{A} ) (which is certain/uncertain)</td>
</tr>
<tr>
<td>( C )</td>
<td>Class attribute</td>
</tr>
<tr>
<td>( c_j )</td>
<td>Class label of the ( j )th instance</td>
</tr>
<tr>
<td>( dom_{A_i} )</td>
<td>Set of possible values on ( A_i )</td>
</tr>
<tr>
<td>( t )</td>
<td>Set of items on the whole input database</td>
</tr>
<tr>
<td>( T )</td>
<td>Set of transactions in current database</td>
</tr>
<tr>
<td>( sup_x (sup_x^c) )</td>
<td>Minimum support threshold</td>
</tr>
<tr>
<td>( con_x ) ( con_x^c )</td>
<td>( {} /@Looking, -@Quality )</td>
</tr>
<tr>
<td>( sup_{min} )</td>
<td>( {} /@Price, +@Quality )</td>
</tr>
<tr>
<td>( IS )</td>
<td>Set of discovered candidate itemsets through the algorithm</td>
</tr>
<tr>
<td>( U \times @y )</td>
<td>( x ) with uncertain attribute(s) under uncertain degree of ( x % )</td>
</tr>
</tbody>
</table>

Table 2: Summary of Notations

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 \( E(con_x^c) = \sum_{w_i \in W} con_{x,w_i}^c \times P(w_i) = \sum_{w_i \in W} sup_{x,w_i}^c \times P(w_i) \) where \( P(w_i) \) is the probability of world \( w_i \). \( con_{x,w_i}^c \) is the expected confidence of \( x \) on class \( c \) in world \( w_i \), while \( sup_{x,w_i}^c \) is the expected 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 \( O(1 + \prod_{A_i \in \mathcal{A}^u} |dom_{A_i}|)^{|T|} \)
possible worlds, where 1 stands for not taking the transaction while \( \prod_{A_k \in A^n \{ \text{dom} A_k \} } \) 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 \( 0 \leq \text{sup}_c \leq \text{sup}_e \leq |T| \), we have:

\[
E(\text{conf}_x) = \sum_{w_i \in W} \text{conf}_x(w_i) \times P(w_i)
\]

\[
= \sum_{i=0}^{|T|} \sum_{j=0}^i \frac{j}{i} \times P(\text{sup}_x = i \land \text{sup}_e = j)
\]

\[
= \sum_{i=0}^{|T|} E_i(\text{sup}_x) \times \sum_{i=0}^{|T|} E_i(\text{conf}_x)
\]

where \( E_i(\text{sup}_x) \) and \( E_i(\text{conf}_x) \) denote the part of expected support and confidence of itemset \( x \) on class \( c \) when \( \text{sup}_x = i \).

Given \( 0 \leq n \leq |T| \), we define \( E_n(\text{sup}_c) = \sum_{i=0}^n E_i(\text{sup}_x) \) as the expected support of \( x \) on class \( c \) on the first \( n \) transactions of \( T \), and \( E_n(\text{sup}_e) \) as the part of expected support of \( x \) on class \( e \) 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

\[
E_i,n(\text{sup}_c) = p_n \times E_{i-1,n-1}(\text{sup}_c) + (1-p_n) \times E_{i,n-1}(\text{sup}_c)
\]

when \( c_n \neq c \), and

\[
E_i,n(\text{sup}_c) = p_n \times E_{i-1,n-1}(\text{sup}_c + 1) + (1-p_n) \times E_{i,n-1}(\text{sup}_e)
\]

when \( c_n = c \), where \( 1 \leq i \leq n \leq |T| \).

\[
E_n(\text{sup}_c) = 0
\]

for \( \forall n \) where \( i = 0 \), or where \( n < i \).

**Proof.** If \( x \not\subseteq t_n \), we have \( E_{n}(\text{sup}_c) = E_{n-1}(\text{sup}_c) + (1-p_n) \times E_{n-1}(\text{sup}_c) \) since both \( \text{sup}_c \) and \( \text{sup}_e \) in each possible world remains the same. If \( x \subseteq t_n \), there exist two situations:

When \( c_n \neq c \), \( \text{sup}_e = \text{sup}_c \) in each possible world remains the same while \( \text{sup}_c = \text{sup}_c + 1 \) and we have \( E_{n}(\text{sup}_c) = E_{n-1}(\text{sup}_c) + p_n \times E_{n-1}(\text{sup}_c) \).

When \( c_n = c \), \( \text{sup}_e = \text{sup}_e + 1 \) and \( \text{sup}_c = \text{sup}_c + 1 \) in each possible world. Hence we have \( E_{n}(\text{sup}_c) = E_{n-1}(\text{sup}_c) + p_n \times E_{n-1}(\text{sup}_c + 1) \) similarly.

Thus, the theorem is proved. \( \square \)

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

\[
E_{i,n}(\text{sup}_c) = p_n \times (E_{i-1,n-1}(\text{sup}_c + 1)) + (1-p_n) \times E_{i,n-1}(\text{sup}_c)
\]

\[
= p_n \times (E_{i-1,n-1}(\text{sup}_c) + P_{i-1,n-1}) + (1-p_n) \times E_{i,n-1}(\text{sup}_c)
\]

when \( c_n = c \), since we have:

\[
E_{i-1,n-1}(\text{sup}_c + 1) = E_{i-1,n-1}(\text{sup}_c) + P_{i-1,n-1}
\]

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

\[
P_{i,n} = \begin{cases} 
1 & \text{for } n = 0 \\
1 - p_n \times P_{i,n-1} & \text{for } 1 \leq n \leq |T| 
\end{cases}
\]

where \( i = 0 \).

\[
P_{i,n} = 0
\]

where \( n < i \).

**Proof.** The proof is similar to that of Theorem 1. \( \square \)

Now we could compute the expected confidence of itemset \( x \) on class \( c \), since \( E(\text{conf}_x) = E(\text{conf}_x) = \sum_{i=0}^n E_i(\text{conf}_c) \).

The whole computation is divided into \( |T| + 1 \) steps with \( E_i(\text{conf}_c) = E_{i-1}(\text{conf}_c) \) computed in \( i \)th step. Figure 1 shows the computation process.

**Figure 1: Computation Process of Expected Confidence**

Finally, we prove the computation complexity of the expected confidence \( E(\text{conf}_x) \) in terms of time and space in Theorem 3.

**Theorem 3.** The computation of the expected confidence \( E(\text{conf}_x) \) 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. \( \square \)

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 \( x < y \), the computation could be stopped since \( x \) would never be provided as a more discriminative pattern with higher confidence value.

**Theorem 4.** Given \( 1 \leq i \leq |T| \),

\[
\text{bound}_i(\text{conf}_x) = \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x) \times \left( \frac{1}{k} - \frac{1}{i} \right) + \frac{E(\text{sup}_x)}{i}
\]

is an upper bound of \( E(\text{conf}_x) \).

**Proof.** For \( \forall i (1 \leq i \leq |T|) \), we have

\[
E(\text{conf}_x) = E(\text{conf}_x)
\]

\[
= \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x) + \sum_{k=i}^{T} E_k|T|(\text{sup}_x)
\]

\[
\leq \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x) + \sum_{k=i}^{T} E_k|T|(\text{sup}_x)
\]

\[
= \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x) + \sum_{k=i}^{T} E_k|T|(\text{sup}_x) - \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x)
\]

\[
= \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x) + \frac{1}{k} - \frac{1}{i} + \frac{E(\text{sup}_x)}{i}
\]

\[
= \text{bound}_i(\text{conf}_x)
\]

Hence \( \text{bound}_i(\text{conf}_x) \) is an upper bound of \( E(\text{conf}_x) \). \( \Box \)

**Corollary 1.** For \( 1 \leq i \leq |T| \), we have:

\[
E(\text{sup}_x) = \text{bound}_1(\text{conf}_x)
\]

\[
\geq \cdots \geq \text{bound}_i(\text{conf}_x) \geq \cdots
\]

\[
\geq \text{bound}_{|T|}(\text{conf}_x) = E(\text{conf}_x)
\]

**Proof.** It is easy to get that \( \text{bound}_1(\text{conf}_x) = E(\text{conf}_x) \) and \( \text{bound}_{|T|}(\text{conf}_x) = E(\text{conf}_x) \) using the definition. Since

\[
\text{bound}_{i-1}(\text{conf}_x) - \text{bound}_i(\text{conf}_x)
\]

\[
= \left( \frac{1}{i-1} - \frac{1}{i} \right) \times (E(\text{sup}_x) - \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x)) \geq 0
\]

for \( 1 < i \leq |T| \), the corollary is proved. \( \Box \)

For \( i \)th step in computing \( E_i|T|(\text{conf}_x) \), we could compute the respective upper bound \( \text{bound}_{i|T}(\text{conf}_x) \) using \( E_k|T|(\text{sup}_x) \) \( (0 \leq k \leq i - 1 < i \leq |T|) \) which all have been computed previously. Actually since

\[
\text{bound}_i(\text{conf}_x) = \text{bound}_{i-1}(\text{conf}_x)
\]

\[
- \left( \frac{1}{i-1} - \frac{1}{i} \right) \times (E(\text{sup}_x) - \sum_{k=0}^{i-1} E_k|T|(\text{sup}_x))
\]

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 2 conducted on one of the evaluation datasets. (Details of the evaluation parameters could be found in Section 6).

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.
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 \( \text{coverProb}_{\text{min}} \). We try to assure that the probability of each instance not covered by anyone itemset is less than \( 1 - \text{coverProb}_{\text{min}} \). 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 \( \text{IS}_t \). Then only the first \( k \) itemsets \( \text{IS}_t[k] \subseteq \text{IS}_t \) with \( \prod_{c \in \text{IS}_t[k]}(1 - P(x \subseteq t)) < 1 - \text{coverProb}_{\text{min}} \) (1 \( \leq k \leq |\text{IS}_t|\)) are selected to remain in the list. For each removed itemsets in \( \text{IS}_t - \text{IS}_t[k] \), we decrease its total covered number on all instances, and remove it from the candidate itemset set \( \text{IS} \) 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\(^5\)light and svm\(^6\)multiclass are used for 2-class and multi-class situations, respectively.

\(^3\)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/svmlight/ For svm\(^6\)multiclass, the parameter of trade-off ("-c") is set to 1000.
6.2.1 Accuracy Comparison

Table 3 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%. $U_x@y$ denotes the datasets with $y$ uncertain attribute(s) under uncertain degree $x$%. N/A$^{unc}$ 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, which validates the effectiveness of our algorithm. Experimental results with other uncertain attributes and under other uncertain degrees also show similar results.

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. However, a too low minimum cover probability (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 uHARMONY.

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-
<table>
<thead>
<tr>
<th>Dataset</th>
<th>uHARMONY</th>
<th>DTU</th>
<th>uRule</th>
<th>uHARMONY</th>
<th>DTU</th>
<th>uRule</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U10@1</td>
<td>U10@2</td>
<td>U10@4</td>
<td></td>
<td>U10@1</td>
<td>U10@2</td>
<td>U10@4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian</td>
<td>86.542</td>
<td>85.942</td>
<td>84.2029</td>
<td>86.109</td>
<td>86.087</td>
<td>85.2174</td>
<td>86.821</td>
</tr>
<tr>
<td>Balance</td>
<td>90.577</td>
<td>65.12</td>
<td>69.92</td>
<td>90.736</td>
<td>66.08</td>
<td>71.2</td>
<td>90.736</td>
</tr>
<tr>
<td>Bands</td>
<td>69.939</td>
<td>65.126</td>
<td>N/Aemon</td>
<td>69.939</td>
<td>N/Aemon</td>
<td>N/Aemon</td>
<td>68.089</td>
</tr>
<tr>
<td>Breast</td>
<td>98.998</td>
<td>91.302</td>
<td>93.053</td>
<td>94.569</td>
<td>91.2732</td>
<td>93.5622</td>
<td>93.999</td>
</tr>
<tr>
<td>Bridges-V1</td>
<td>65.068</td>
<td>51.868</td>
<td>59.434</td>
<td>65.068</td>
<td>53.7376</td>
<td>60.3744</td>
<td>67.012</td>
</tr>
<tr>
<td>Bridges-V2</td>
<td>64.143</td>
<td>65.0943</td>
<td>60.3774</td>
<td>63.032</td>
<td>67.9245</td>
<td>55.6604</td>
<td>64.144</td>
</tr>
<tr>
<td>Car</td>
<td>88.66</td>
<td>91.1458</td>
<td>85.5324</td>
<td>89.82</td>
<td>72.9745</td>
<td>69.0394</td>
<td>87.907</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>51.797</td>
<td>50.1018</td>
<td>44.2634</td>
<td>49.895</td>
<td>51.5954</td>
<td>43.9919</td>
<td>49.426</td>
</tr>
<tr>
<td>Credit</td>
<td>85.517</td>
<td>87.3913</td>
<td>84.2029</td>
<td>85.517</td>
<td>86.9565</td>
<td>86.5217</td>
<td>86.382</td>
</tr>
<tr>
<td>Echocardiogram</td>
<td>93.289</td>
<td>92.3664</td>
<td>92.3664</td>
<td>92.52</td>
<td>92.3664</td>
<td>90.8397</td>
<td>92.52</td>
</tr>
<tr>
<td>Flag</td>
<td>65.667</td>
<td>67.0103</td>
<td>62.3711</td>
<td>62.544</td>
<td>65.9794</td>
<td>59.7938</td>
<td>59.868</td>
</tr>
<tr>
<td>German</td>
<td>72.8</td>
<td>69</td>
<td>70.6</td>
<td>72.7</td>
<td>70.7</td>
<td>70.5</td>
<td>72.9</td>
</tr>
<tr>
<td>Heart</td>
<td>57.943</td>
<td>54.1304</td>
<td>50</td>
<td>58.166</td>
<td>54.2391</td>
<td>50.9783</td>
<td>57.854</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>83.772</td>
<td>79.3548</td>
<td>78.0645</td>
<td>83.772</td>
<td>79.3548</td>
<td>77.4194</td>
<td>79.264</td>
</tr>
<tr>
<td>Horse</td>
<td>86.108</td>
<td>85.3261</td>
<td>87.5</td>
<td>86.108</td>
<td>85.3261</td>
<td>87.2283</td>
<td>86.1</td>
</tr>
<tr>
<td>Monks-1</td>
<td>100</td>
<td>97.8417</td>
<td>95.6835</td>
<td>100</td>
<td>74.6403</td>
<td>97.1223</td>
<td>100</td>
</tr>
<tr>
<td>Monks-2</td>
<td>69.554</td>
<td>65.7238</td>
<td>64.2263</td>
<td>72.187</td>
<td>65.7238</td>
<td>63.0616</td>
<td>76.7</td>
</tr>
<tr>
<td>Monks-3</td>
<td>96.402</td>
<td>98.917</td>
<td>98.1949</td>
<td>96.402</td>
<td>79.9639</td>
<td>79.9639</td>
<td>96.402</td>
</tr>
<tr>
<td>Mushroom</td>
<td>99.618</td>
<td>100</td>
<td>100</td>
<td>99.766</td>
<td>100</td>
<td>100</td>
<td>99.717</td>
</tr>
<tr>
<td>Pima</td>
<td>68.106</td>
<td>65.1042</td>
<td>68.099</td>
<td>68.106</td>
<td>65.1042</td>
<td>67.4479</td>
<td>68.106</td>
</tr>
<tr>
<td>Postoperative</td>
<td>69.614</td>
<td>68.8889</td>
<td>68.8889</td>
<td>69.614</td>
<td>70</td>
<td>70</td>
<td>69.198</td>
</tr>
<tr>
<td>Promoters</td>
<td>87.166</td>
<td>76.4151</td>
<td>71.6981</td>
<td>86</td>
<td>73.8494</td>
<td>69.8113</td>
<td>77.667</td>
</tr>
<tr>
<td>Spect</td>
<td>85.846</td>
<td>79.7753</td>
<td>83.951</td>
<td>86.956</td>
<td>79.7753</td>
<td>83.5206</td>
<td>85.474</td>
</tr>
<tr>
<td>Survival</td>
<td>73.529</td>
<td>73.5294</td>
<td>70.2614</td>
<td>73.529</td>
<td>73.5294</td>
<td>70.415</td>
<td>73.529</td>
</tr>
<tr>
<td>Ta_eval</td>
<td>54.298</td>
<td>44.3709</td>
<td>40.3974</td>
<td>52.263</td>
<td>48.3444</td>
<td>44.3471</td>
<td>45.746</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>100</td>
<td>85.6994</td>
<td>79.0705</td>
<td>100</td>
<td>78.81</td>
<td>69.6597</td>
<td>99.895</td>
</tr>
<tr>
<td>Vehicle</td>
<td>65.323</td>
<td>64.0662</td>
<td>N/Aemon</td>
<td>65.441</td>
<td>64.0662</td>
<td>N/Aemon</td>
<td>64.02</td>
</tr>
<tr>
<td>Voting</td>
<td>96.099</td>
<td>93.7931</td>
<td>91.954</td>
<td>94.475</td>
<td>93.5632</td>
<td>90.8046</td>
<td>95.406</td>
</tr>
<tr>
<td>Wine</td>
<td>53.086</td>
<td>39.8876</td>
<td>43.3584</td>
<td>53.601</td>
<td>39.8876</td>
<td>39.3258</td>
<td>50.525</td>
</tr>
<tr>
<td>Zoo</td>
<td>93.954</td>
<td>92.0792</td>
<td>89.1089</td>
<td>92.074</td>
<td>90.099</td>
<td>90.099</td>
<td>93.045</td>
</tr>
<tr>
<td>Average</td>
<td>79.0138</td>
<td>74.8738</td>
<td>75.2111</td>
<td>78.6970</td>
<td>73.1629</td>
<td>73.4107</td>
<td>77.9657</td>
</tr>
</tbody>
</table>

Table 4: Accuracy (in %) Comparison of U[10,20]@{1,2,4}
6.3.2 Effectiveness of the Expected Confidence Upper Bound

We evaluated the effectiveness of the expected confidence upper bound. Figure 4 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.

![Figure 4: Accuracy Evaluation of U10@1 w.r.t. Minimum Support](image)

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.

![Figure 7: Running Time Evaluation of U10@4](image)

![Figure 8: Scalability Evaluation (U10@1, Running Time)](image)

![Figure 9: Scalability Evaluation (U10@1, Memory Usage)](image)
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