Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security.

– John Allen Paulos

University of Alberta

CLASSIFICATION AND SEQUENTIAL PATTERN MINING FROM UNCERTAIN DATASETS

by

Metanat Hooshsadat

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2011
Edmonton, Alberta

Abstract

Several research projects explore the application of uncertain databases which contain probabilistic attributes. Uncertainty in data can be caused by inherent randomness, imprecision in measuring equipment, ambiguity, information extraction from unstructured data, etc.

The classification and Sequential Pattern Mining (SPM) of uncertain datasets both play a vital role in decision making systems and have recently attracted significant attention. In this study, we propose two novel algorithms for the aforementioned problems. Our novel associative classifier for uncertain data, UAC, has an effective rule pruning strategy. Using a general model for uncertainty, our experiments show that in many cases, UAC reaches higher accuracies than the existing algorithms.

In SPM for uncertain data, other studies aimed to solve the problem for specific uncertainty models. We introduce UAprioriAll which conducts SPM from datasets with general attribute level uncertainty. Our experiments show that this method scales linearly when increasing the number of transactions.

Table of Contents

List of Tables

List of Figures

List of Papers

- Metanat Hooshsadat, Osmar R. Zaïane, An Associative Classifier For Uncertain Data, submitted to ICDM 2011.
- Metanat Hooshsadat, Samaneh Bayat, Parisa Naeimi, Mahdieh S. Mirian, Osmar R. Zaïane, Finding Sequential Patterns in Probabilistic Data, submitted to ICDM 2011.
- Metanat HooshSadat, Hamman Samuel, Sonal Patel, Osmar R. Zaïane. Fastest association rule mining algorithm predictor - farm-ap. In Fourth International C* Conference on Computer Science and Software Engineering., pages 43–50, Montreal, QC, Canada, 2011.

Chapter 1

Introduction

1.1 Motivation

Typical relational databases or databases in general hold collections of records representing facts. These facts are observations with confident values stored in the fields of each tuple of the database. In other words, the observation represented by a record is assumed to have taken place and the attribute values are assumed to be true. For example, a customer database contains records of customers that exist and in each tuple, the recorded attribute values are considered to be true observations such as the address, the amount purchased, etc. We call these databases "certain database" because we are certain about the recorded data and their values. Sophisticated techniques for storing, managing, querying, sorting and mining "certain" databases have been developed for decades and are nowadays ubiquitous.

In contrast to "certain" data there is also "uncertain data"; data for which we may not be sure about the observation whether it really took place or not, or data for which the attribute values are not ascertained with 100% probability.

Many applications deal with such uncertain data. A typical example is data collected from sensor networks. Sensors may collect data such as temperature, pressure or other with some uncertainty specified by the manufacturers or uncertainty due to the age of the sensor, its location, the condition it is in or simply the energy source available. Another application example is the tracking of moving objects where the position of an object is never exact but an approximation bound by an interval or associated with a probability. Another example deals with medical

records that are populated from text-based patients' records. The information extraction methods used have confidence levels attached on the extracted information, that we use as uncertainty.

There are many models proposed in the literature to represent uncertainty and often these models are specifically fitted to the application at hand. However, in general, there are models that focus on uncertainty attached to a whole record centering on what is referred to as possible worlds, while other models put the uncertainty on the attribute values by either attaching a probability on an observed value for an attribute or a set of probabilities for all possible values an attribute could have (see Figure 1.1).

a_1	a_2	a_3	c	p_r
$v_{1,i}$	$v_{2,i}$	$v_{3,i}$	c_i	

(a)

a_1	a_2	a_3	c
$v_{1,i} : p_{1,i}$	$v_{2,i} : p_{2,i}$	$v_{3,i} : p_{3,i}$	c_i

(b)

a_1	$\{v_{1,i,1} : p_{1,i,1}, v_{1,i,2} : p_{1,i,2}, .., v_{1,i,n_1} : p_{1,i,n_1}\}$
a_2	$\{v_{2,i,1} : p_{2,i,1}, v_{2,i,2} : p_{2,i,2}, .., v_{2,i,n_2} : p_{2,i,n_2}\}$
a_3	$\{v_{3,i,1} : p_{3,i,1}, v_{3,i,2} : p_{3,i,2}, .., v_{3,i,n_3} : p_{3,i,n_3}\}$
c	c_i

(c)

a_1	$\{v_{1,i,1} : p_{1,i,1}, v_{1,i,2} : p_{1,i,2}, .., v_{1,i,k_1} : p_{1,i,k_1}\}$
a_2	$\{v_{2,i,1} : p_{2,i,1}, v_{2,i,2} : p_{2,i,2}, .., v_{2,i,k_2} : p_{2,i,k_2}\}$
a_3	$\{v_{3,i,1} : p_{3,i,1}, v_{3,i,2} : p_{3,i,2}, .., v_{3,i,k_3} : p_{3,i,k_3}\}$
c	c_i

(d)

Figure 1.1: Some of the possible models for uncertainty.(a) An uncertain tuple with three items and the class label. Probability p_r is set on the whole record. (b) An uncertain tuple in which a probability is attached to each observed value. (c) An uncertain tuple in which each item is denoted by a vector of probabilities on the whole attribute domain, where the size of the domain is n_j. (d) An uncertain tuple in which only k_j elements of the probability vector are known ($k_j \leq n_j$). One may also consider class labels uncertain.

Querying probabilistic data, particularly computing aggregations, ranking uncertain data or discovering patterns in probabilistic data is a challenging feat. Many researchers have focused on uncertain databases, also called probabilistic databases, for managing uncertain data [38, 22], top-k ranking uncertain data [18, 49], querying uncertain data [29, 10], clustering [9] or mining uncertain data [2, 24, 21].

In this study, we use frequent pattern mining to classify and mine the frequent

sequences from uncertain databases. Frequent pattern mining has been the focus of many studies for years. Numerous algorithms have been designed for addressing different problems in this area, ranging from frequent itemset mining in transaction databases, sequential pattern mining, structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering, to applications of these algorithms.

A transactional database, such as the one in Table 1.1, is a database containing sets of items correlated to id numbers. Frequent patterns are defined to be the set of items which appear in at least t transactions. The number of transactions containing an itemset is called *support*. To find frequent patterns, usually a user defined lower bound called *Minimum Support* is set on the support values of the itemsets, which adjusts the sensibility of the mining procedure to the noise. For example, with the minimum support of 2, itemset a, b, which appears in transactions $1, 2, 4$ and thus has a support of 3, is frequent. However, a, b, c only appears in 1 and thus has the support of 1, which implies that it is not frequent.

ID	transactions
1	a,b,c,e,f
2	a,b,f
3	e,f
4	a,b,e

Table 1.1: A sample transactional database

Using the concept of frequent pattern mining from uncertain data, we are interested in building associative classifiers on medical data in the context of health care decision support systems. There are two main challenges to this endeavor. The first challenge is that users of these decision support systems are medical practitioners who prefer to understand the classification models used and be able to inject new domain knowledge within the learned model. Therefore, rule-based classifiers are preferred. The second challenge is the fact that existing classification techniques are based on "certain" datasets. Very little work has been done so far on building a classifier on uncertain training data. The medical data we need to use is indeed uncertain. Most medical records are largely in free-text form and thus require ad-

vanced text mining techniques to extract structured information [50, 23]. Extracting patient information from free-text to populate a structured database is inherently uncertain. The machine learning techniques that identify specific information such as diseases, treatments, treatments for which disease, blood pressure, drug dosage, etc. do these identifications and extractions with some measured confidence. Therefore the acquired database is an uncertain database with probabilities attached to each attribute value representing this confidence obtained by the machine learning techniques used to extract the variety of patient information.

The other problem which is of our interest is mining sequential patterns from uncertain datasets. SPM is an important problem defined on the sequential datasets. In these datasets, each sequence is associated with a unique id and contains multiple transactions. Each transaction in a sequence is stamped with a particular order identifier, e.g. time. In this chapter, we often use the terms time stamp and order identifier interchangeably for simplicity. However, while time stamp is the most commonly used, there are many other order identifiers.

Definition 1. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified minimum support threshold, sequential pattern mining is to find all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than the minimum support.

There are many applications for SPM from uncertain datasets. For example, assume the problem of mining from medical records again. We are interested in studying the side effects of the medicine taken by patients on their health. To study this, we need to mine sequences from this inherently uncertain datasets to find chronological correlations between medicines and symptoms.

UAprioriAll presented in Chapter 4 is designed based on the well known and efficient AprioriAll to enable mining frequent sequences in datasets with existential uncertainty. One can assume a variety of models to capture uncertainty in sequential datasets. The uncertainty can be assumed on the transaction level (analogous to record level uncertainty) or on the items (analogous to attribute level un-

certainty). Possible models of uncertainty for datasets used for classification are presented in Figure 1.2. Item-level uncertainty, which is considered more general than transaction-level uncertainty, is our choice of model in Chapter 4. This is the first work that considers this broad model, while all of the previous related studies have assumed limitations on the dataset, such as assumptions on length of transactions. Indeed, there are numerous cases where the length of transactions differ in variations of the possible worlds (i.e. all possible databases). Our data model in Chapter 4 is more general and does not assume any restriction of this sort on the items or transactions.

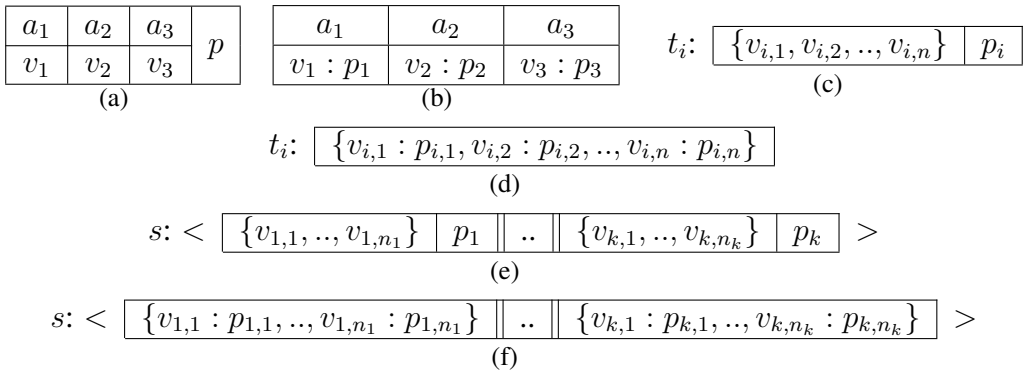


Figure 1.2: Some of the possible models for uncertainty in databases and sequential datasets: (a) A tuple with record-level uncertainty; (b) A tuple with attribute level uncertainty; (c) A transaction with transaction-level uncertainty; (d) A transactions with item-level uncertainty; (e) A sequence with transaction-level uncertainty; (f) A sequence with item-level uncertainty.

1.2 Hypothesis

In this manuscript, we are to prove the following hypotheses:

1. *A dataset with general attribute level uncertainty (model specified in Section 3.1) can be classified by an associative classifier where the accuracies are improved upon the existing rule based classifiers, including uHARMONY, DTU, uRule, UCBA.*

2. *Sequential Patterns can be extracted from a dataset with attribute level uncertainty.*

Hypothesis 1 can be elaborated as follows:

- The associative classifier is accurate on both traditional uncertainty model (Figure 1.1(c)) and the new more general uncertainty model introduced by this study (Figure 1.1(d)). To prove this, we apply our proposed method, UAC, to real data with randomly added uncertainty.

- The associative classifier is more accurate than the state of the art uncertain data classifiers. We report the accuracies of our proposed method and existing methods on both uncertainty models which shows that UAC reaches higher accuracies on most cases and on the average.

- The strategy used for handling uncertainty reaches better accuracy, more compact set of rules and needs less time to build the model, than the existing strategy. We compare our method to the existing method UCBA [33], an existing method which is similar to our method except for the uncertainty handling technique.

Hypothesis 2 can be elaborated as follows:

- Using expected support as a mathematically sound measure for frequentness, frequent sequential patterns can be extracted from uncertain datasets, using an algorithm for which the runtime grows linearly when increasing the number of sequences.

1.3 Contributions

- In this study, we introduce a new more general model for capturing uncertainty in classification problems. The space of this model contains the space of the model addressed by the existing uncertain data classifiers. Our model is able to capture more realistic cases of uncertainty.

- We present a probabilistic data associative classifier that effectively handles uncertainty. UAC is more accurate than the state of the art rule based algorithms. It is more accurate than UCBA, another CBA based algorithm, under

the new more general model. Additionally, the number of rules and the running time is far less than UCBA.

- To handle uncertainty in the classification without filtering out any information from the dataset, we introduce the concepts of coverage, probability inclusion and applicability.

- We design UAprioriAll, that is the first sequential pattern mining algorithm for uncertain datasets that handles a general form of attribute level uncertainty. We provide soundness and completeness proof for this algorithm, and results that show that the runtime of UAprioriAll grows linearly when increasing the number of transactions.

1.4 Outline

- Chapter 2 reviews the most important studies in the area of uncertainty modeling, frequent pattern mining, sequential pattern mining and classification. Section 2.1 mentions and briefly describes the different studies published on uncertainty modeling. Section 2.2 describes the problem of frequent pattern mining and addresses some of the important methods to solve the problem. Algorithm Apriori which is adopted by our algorithms is described and the concepts of expected support and probabilistic support are compared in this section. Section 2.3 briefly overviews the existing associative classifiers. This section also gives a clear explanation on the problem of coverage and compares the different probabilistic data classifiers based on the way they address this issue. Section 2.4 gives a summary of the most important sequential pattern mining algorithms for both traditional and probabilistic datasets, and clarifies the difference between our new method and the existing methods.

- Chapter 3 addresses the problem of devising an accurate rule-based classifier on uncertain training data after the information extraction. In this chapter, Section 3.1 explains our data model. Section 3.2 provides the elaborated description of our novel classification method: UAC. Section 3.3 contains the

experiments description and the results.

- Chapter 4 addresses the problem of mining sequential patterns from data with attribute level uncertainty and presents algorithm UAprioriAll. In Section 4.1 the data model used and the algorithm are explained. Algorithm correctness proof is provided in this section as well. Section 4.2 describes the experiments and the figures that show the runtime of the algorithm. The work presented in this chapter originated from an initial study made in the Winter of 2010 in the graduate course on Electronic Health Records and Data Analysis (CMPUT 690) by Samaneh Bayat, Parisa Naeimi and Mahdieh S. Mirian. In this chapter, I introduce their original contributions and further enhance their research.

- Chapter 5 summarizes the important conclusions and the contributions. In addition, we list the important future work that can be done regarding the problems discussed in this study.

Chapter 2

Related Works

In this chapter we review some of the important studies related to uncertain datasets, frequent pattern mining and classification.

2.1 Modeling Uncertain Data

The possible worlds model [1] represents the whole set of databases that are consistent with a schema and is used to display incomplete information (uncertain) databases. If N is the set of all finite n-ary relations over D, where D is a fixed countably infinite domain, an incomplete information database I is a subset of N [19]. With this definition, the usual (certain) databases are the single membered subsets, and the no-information databases is N which contains all possible databases. In other words, an uncertain relation represents a set of possible relation instances, rather than a single one. The possible world model is *complete* meaning that any possible set of relation instances is representable by an uncertain relation in the model. Consequently, the possible world model is closed under relational operations, meaning that applying any relational operator on databases represented by the possible world model results in a database that is also representable in the possible world model.

An inherent tension in the possible world model is the high complexity. Complete models are usually excessive for some applications and they are less readable than the incomplete models. Usually the intuitive models that can describe data are not complete. Sarma et al. [37] addressed this problem and introduced incomplete

models including or-sets and existential uncertainty.

In addition, the possible world model is not usually pragmatically usable in applications, since there are many different possible states for the database and the problem of mining and classification of such data is computationally intractable. Studies on uncertain data often use simplifications over the possible worlds model that do not reduce the generality of the problem [3]. The simplifications lead to famous practical models of uncertain databases, including record-level and attribute-level uncertainty. Record level uncertainty assigns a confidence for each tuple. Attribute level probability (or existential uncertainty) is another case in which attributes are associated with probabilities, probability density functions, or other statistical parameters such as variance. The probabilities assigned to the items are often assumed to be independent to simplify the calculations.

In the real world applications, Studies use simpler models that are as close as possible to reality, for example record level and attribute level uncertainty models. The continuous attributes are usually modeled by intervals where the minimum and maximum of an attribute value is known. Usually the interval is descritized to transform this data to categorical attributes. A probability density function may also be considered for the continuous attribute [31].

2.2 Frequent Pattern Mining

Many frequent pattern mining algorithms have been proposed over the past 20 years. Apriori [4] was the pioneer algorithm in which the frequent itemsets are mined from the data using a multilevel approach. This approach is based on *downward closure* property, that is if itemset x is a subset of y and x is not frequent, y is consequently impossible to be frequent. Other algorithms including FP-growth [20] and eclat [46] also mine the frequent patterns using data structures. From the performance point of view, none of the frequent pattern mining algorithms is superior to the others all the time. Some studies have captured the significant properties of the datasets that influence the running time of the different mining algorithms [27].

In this study, we propose two Apriori based algorithms to solve classification

and sequential pattern mining problems for probabilistic databases. The reason for this choice is that Apriori is a fast, simple and well known algorithm. Algorithm 1 shows the pseudocode of Apriori. In line 1 of this algorithm, function "freq-items" finds all of the single items that have a greater support than the minimum support. In line 3, function "generate" generates the level candidates based on the frequent itemsets of the previous level, that is generating the cartesian product $L_{k-1} \times L_{k-1}$ and eliminating any itemset that has subitemsets that are not frequent.

Algorithm 1 Apriori

```
1:  input: D is the dataset, minsup is the minimum support
2:  L_1= freq-items(D);
3:  for all (k= 2; L_{k-1} ≠ ∅; k++) do
4:      C_k= generate(L_{k-1}).
5:      for all transaction t in D do
6:          for all c ∈ C_k do
7:              if c ∈ t then
8:                  c.count++
9:              end if
10:         end for
11:     end for
12:     L_k = ∀ candidates c in C_k with c.count ≥ minsup
13: end for
14: return ⋃_k L_k
```

A probabilistic transactional database with item level uncertainty contains records that have probabilities attached to each item. Table 2.1 is a probabilistic database, where $a : p$ shows that a transaction contains item a with the probability p attached to it. To find Frequent patterns in an uncertain dataset, most studies use a user defined threshold on *expected support*. The expected support of itemset x in dataset D is defined by Equation 1 where $P(x \in T)$ denotes the probability of x existing in T. Based on this definition an item is frequent if $E(s(x)) \geq \sigma$ where σ is the minimum support threshold defined by the user.

$$\begin{array}{lcl} P(x \in T) & = & \prod_{i \in x} P(i \in T) \\ E(s(x)) & = & \Sigma_{T \in D} P(x \in T) \end{array} \quad (1)$$

Aggarwal *et al.* have studied frequent pattern mining on uncertain data by extending well-known deterministic algorithms of different classes using expected

ID	transactions
1	a:0.4,b:0.6,c:0.8,e:1,f:0.1
2	a:0.3,b:0.4,f:0.7
3	e:0.5,f:0.5
4	a:0.4,b:0.3,e:0.5

Table 2.1: A sample uncertain transactional database

support. These include Apriori, which belongs to the class of candidate generate-and-test algorithms, H-mine and FP-Growth which are pattern growth-based algorithms. In the deterministic case, FP-Growth is well-known for its efficiency especially in dense datasets. However, according to their study, an extension of deterministic algorithms to uncertain data shows completely counter-intuitive behavior. Apriori and H-mine demonstrate more efficiency in dealing with uncertainty while FP-Growth, because of its compressed structure, is not efficient enough in the uncertain case. Moreover, their evaluation shows that pruning techniques which are useful in cases of low uncertainty probability, do not work well in high uncertainty probability conditions [2, 44, 3].

Probabilistic support [7, 48] is another measure defined based on the probability that the itemset x in dataset D has a support more than the minimum support threshold σ. The frequent itemsets based on this definition, are the ones that match the inequality presented in Equation 2 where τ is the user defined probability threshold.

$$P(s(x) \geq \sigma \quad) = 1 - \Sigma_{S \in D:|S|<\sigma} (\Pi_{t \in S} P(x \in t)). \\ (\Pi_{t \in (T-S)} (1 - P(x \in t))) \geq \tau \tag{2}$$

PFIM [48] is one of the algorithms that uses probabilistic support as its frequency measure. Using dynamic programming to speed up computation time, the probabilistic support of each item is calculated and k most frequent items are computed. Another way of using the algorithm is to set a threshold for the probabilistic support of itemsets where the minimum support is also defined by the user.

Expected support-based algorithms like UApriori [3] grow linearly when the number of transactions is increased and the time complexity is $O(|T|)$ in the worst case [3], while probabilistic support-based algorithms like PFIM [7] are of $O(\sigma.|T|)$ time complexity. Usually σ is chosen relatively to the size or as a percentage of the

number of transactions, PFIM is of $O(|T|^2)$ time complexity in its average case [7]. In this study we use expected support, not only because of its strong theoretical basis, but also because of its linear time complexity.

2.3 Classification

Classification or supervised learning is one of the most active fields in machine learning which involves predicting the category of a given piece of data based on the previously observed information. In this category of learning algorithms, required information is extracted from an available database and then used for predicting the unknown attributes of other objects. For example, having enough information about symptoms and diseases in a number of patients, we can diagnose the disease of a new patient based on his/her symptoms. Various algorithms have been designed for this purpose, including SVM (Support Vector Machine), rule based classifiers, K nearest neighbors, neural networks, etc [40].

Recently, a considerable amount of studies in machine learning are directed toward the uncertain data classification, including: TSVC [8] (inspired by SVM), DTU [30] (decision tree), UNN [17] (based on Neural Network), Bayesian classifier [31], uRule [32] (rule based), uHARMONY [14] (based on HARMONY [41]) and UCBA [33] (based on CBA [26]). However, models suggested by the previous work do not capture some possible types of uncertainty. In previous studies, numerical attributes are only modeled by intervals, while they may exist in other forms such as probability vectors. Categorical attributes are modeled by a probability distribution vector over their domain where the vector is unrealistically assumed to be completely known. Section 3.1 explains this issue in detail and introduces our more general model for uncertainty.

High accuracy and strong flexibility are some of the advantageous characteristics of the rule based classifiers. Another important advantage is their high understandability by human experts, even those specialized in other fields of knowledge than computing science. For example, health experts can revise the rules of a rule based classifiers applied in health sciences to increase the accuracy. This is of

course only true if the number of rules is reasonably low. Given these significant advantages, investigating rule based uncertain data classifiers has been the theme of many studies. One of these studies is uRule [32], that is an uncertain data classifier based on RIPPER [11]. uRule defines the information gain metric in the presence of uncertainty. The probability of each rule classifying the instance is computed based on the weighting system introduced by uRule.

Associative classification is a large category of rule based classification in which the rule induction procedure is based on the association rule mining technique. Association rule mining is to extract all rules in the database that satisfy particular minimum support and minimum confidence constraints. Some of the prominent associative classifiers that use candidate generation are CBA [26], ARC [45], and CMAR [25]. Much work has been done in improving associative classifiers such as [35] and [36]. In Chapter 3, we introduce an associative classifier for uncertain datasets, which is based on CBA. CBA is highly accurate, flexible and efficient both in time and memory [26].

CBA [26] directly adopts Apriori [4] to mine the potential classification rules or strong *ruleitems* from the data. Ruleitems are those association rules of form $a \rightarrow c$, where the consequence (c) is a class label and the antecedent (a) is a set of *attribute assignments*. Each *attribute assignment* consists of an attribute and a value which belongs to the domain of that attribute. For example, if A_1 and A_2 are two attributes and c is a class label, $r = (A_1 : u_1, A_2 : u_2 \rightarrow c)$ is a ruleitem. r implies that if A_1 and A_2 have values of u_1 and u_2 respectively, the class label should be c. A ruleitem is strong if its support and confidence are above the predefined thresholds.

After mining the strong ruleitems, a large number of them are eliminated by applying the *database coverage* approach. This method is a filtering technique commonly used by associative classifiers. The procedure involves removing the rules that do not classify any instance correctly, as well as filtering the ruleitems that have negative effect on the accuracy. Database coverage increases both accuracy and understandability of the rule set. The database coverage method of CBA makes slightly more than one pass over the dataset. This makes CBA an efficient algorithm when the database is large and can only be partially loaded into main memory.

UCBA [33] is an uncertain data associative classifier which is inspired by CBA. This study uses the same weighting system as uRule and select multiple rules for classification of each instance. Another study in uncertain data classification is uHARMONY [14]. uHARMONY is mostly inspired by HARMONY [41], that is an associative classifier for "certain" data. HARMONY does not directly apply an association rule mining algorithm. It combines the mining and the construction of the final set of rules to improve the efficiency. uHARMONY defines and uses the concept of expected confidence.

An important problem addressed by all rule based uncertain data classifiers is the coverage problem. Rule based classifiers often need to evaluate various rules to pick the best ones. This level is critical in maintaining a high accuracy. The evaluation often involves the answer to the following question: *To which training instances can a rule be applied?*. Of course, in "certain" data classifiers the answer is obvious, a rule can only be applied to those instances that satisfy its antecedent. However, the answer is not obvious for uncertain datasets. Many uncertain dataset instances may satisfy the antecedent of a rule, each with a different probability. We call this the coverage problem. Existing uncertain data rule based classifiers have suggested various answers to this problem.

uHARMONY suggested a lower bound on the probability by which the instance satisfies the rule antecedent. This approach is simple and fast, but the difficulty or even impossibility of setting the threshold is a problem. This is explained in more detail in Section 3.2.2. uRule suggested that a rule can only be applied to a part of an instance. In contrast to uHARMONY, this method uses the whole dataset but it may cause sensitivity to noise which is undesirable. UCBA does not include the uncertainty in the rule selection process; they select as many rules as possible. This method does not filter enough rule; so may decrease the accuracy and understandability.

2.4 Sequential Pattern Mining

Many sequential pattern mining algorithms and approaches exist for "certain" sequential datasets. Pei *et al.* [28] presented an algorithm called Prefix Span which uses a prefix tree to speed up the SPM process on standard data. AprioriAll [5] is an apriori based algorithm which finds all of the frequent sequential patterns in a database. Algorithm SPIRIT [16] aims at finding the patterns that are specified by the user using regular expressions. SPADE [47] is another sequential pattern mining algorithm that uses lattice search and simple joins to speed up the procedure. GSP [39] discovers the sequential patterns in a database, assuming that the distance between the items in a sequence must be less than a user specified threshold.

Yang *et al.* [43] proposed an algorithm for SPM in uncertain datasets. Their problem involves finding the sequential patterns in presence of noise. Noise may cause an item to be misinterpreted as another item in the process of data gathering. This phenomenon happens in many cases including biomedical studies and consumer behavior analysis. They used a matrix of probabilities in which each cell shows the probability by which an item such as x is misread as another item such as y. Using this matrix, they calculated the probabilities for each sequence to be in the dataset and extracted the frequent sequences.

TrajPattern [42] is another study on frequent sequential pattern mining of uncertain databases that is an algorithm for finding the trajectory of mobile objects. In applications dealing with mobile objects, data is imprecise because it is only possible to obtain the imprecise locations at a given time due to the mobility of devices and unreliable communication links. The uncertainty modeling of this study included an interval of uncertainty for the location of each object .

Chapter 3

UAC: Uncertainty-handling Associative Classifier

Classifying uncertain datasets is an interesting problem that has different real world applications. Data collected from sensor networks, or data measured by imprecise measuring devices are always subject to uncertainty. These data can be collected in databases with attribute level uncertainty and used for prediction purposes.

We are interested in building classifiers on medical data in the context of health care decision support systems. There are two main challenges to this endeavor. The first challenge is that users of these decision support systems are medical practitioners who prefer to understand the classification models used and be able to inject new domain knowledge within the learned model. Therefore, rule-based classifiers are preferred. The second challenge is the fact that existing classification techniques are based on "certain" datasets. Very little work has been done so far on building a classifier on uncertain training data. The medical data we need to use is indeed uncertain. Most medical records are largely in free-text form and thus require advanced text mining techniques to extract structured information [50, 23]. Extracting patient information from free-text to populate a structured database is inherently uncertain. The machine learning techniques that identify specific information such as diseases, treatments, relationships: treatment for each disease, blood pressure, drug dosage, etc. make these identifications and extractions with some measured confidence. Therefore the acquired database is an uncertain database with probabilities attached to each attribute value representing this confidence obtained by the

machine learning techniques used to extract the variety of patient information.

This chapter addresses the problem of devising an accurate rule-based classifier on uncertain training data after the information extraction. There are many classification paradigms but the classifiers of interest to our study are rule-based because the practitioners want a transparent model as opposed to a black box. We opted for associative classifiers, classifiers using a model based on association rules, as they were shown to be highly accurate and competitive with other approaches [6].

First, in Section 3.1, we introduce our model for representing uncertain data. In Section 3.2, we introduce our novel classification method UAC. Finally in Section 3.3 we present empirical evaluations comparing UAC with other published works.

3.1 Uncertainty Modeling

In this study, we address the attribute level uncertainty. Expression of attribute value uncertainty depends upon whether an attribute is categorical or numerical.

3.1.1 Categorical Attributes

Categorical attributes have finite domains. The domain of uncertain categorical attribute A_j is denoted by $A_j.D$= $\{A_j.v_1,\ A_j.v_2,\ ...,\ A_j.v_n\}$. The value of this attribute for the i-th instance is a set of value-probability pairs denoted by $A_{j,i}.L$ and is presented by Equation 1. An example of uncertain categorical attributes is $color$ with domain of $\{black,\ white,\ blue\}$. Previous work on uncertain data classifiers assumed that the probability vector $[p_{j,i,1}, p_{j,i,2}, .., p_{j,i,k}]$ is always known (Figure 1.1(c)). Thus, the sum of the elements in the vector is always 1. Our model rejects that assumption. For example, for an attribute such as color, if value $black$ has a probability of p_{black}, p_{white} and p_{blue} are not identified (Figure 1.1(d)). This is Especially observed when the size of the domain is large.

$$\begin{aligned} &A_{j,i}.L = \{(u_{j,i,1} : p_{j,i,1}), (u_{j,i,2} : p_{j,i,2}), .., (u_{j,i,k} : p_{j,i,k})\} \\ &\forall l \leq k\,;\, u_{j,i,l} \in A_j, i.L\, \Sigma_{l=1}^{k}\, p_{j,i,l} \leq 1. \end{aligned} \tag{1}$$

3.1.2 Numerical Attributes

Numerical attributes have infinite sets as their domains. An example is attribute $time$ expressed in milliseconds. The domain of $time$ is all numbers greater than 0. Based on the method of extraction and the source of data, numerical attributes may have two different forms in an uncertain dataset: Sampled and Interval.

Sampled Numerical Attributes

Uncertainty in numerical attributes may be expressed by a set of possible values each associated with a probability. For example, an instance of attribute time may have values of 10 or 12 associated with probabilities 0.5 and 0.4, respectively. Equation 2 presents the formal definition of this model. $A_{i,j}.L$ is the value list associated with $A_{i,j}$, that is the i-th instance of sampled numerical attribute A_j and $A_j.D = [A_j.l, A_j.u]$. Similar to categorical attributes, the sum of probabilities for each probability vector is not necessarily 1, but at most 1.

$$\begin{aligned} &A_{j,i}.L = \{(x_{j,i,1} : p_{j,i,1}), (x_{j,i,2} : p_{j,i,2}), .., (x_{j,i,k} : p_{j,i,k})\} \\ &\forall q \leq k \, ; A_j.l \leq x_{j,i,q} \leq A_j.u \, \Sigma_{q=1}^{k} p_{j,i,q} \leq 1. \end{aligned} \tag{2}$$

Interval Numerical Attribute

In the real world, sometimes we are not able to identify the exact value of an attribute, but we have a lower and upper bound (interval). Besides the interval, a *Probabilistic Density Function* or *pdf* may also be available in some cases. For example, assume a network of sensors, where each sensor reports the time by which a car passes by it. In this system, the time consumed by processing, sending the information, and other undetermined errors lead to a value that is a close estimation of the actual time. However, there are methods to calculate the upper bound and the lower bound of this error which lead to a better representation of the information: an interval which contains the actual value of attribute *Time*. Equation 3 presents the formal definition of an interval based numerical attribute. $A_{i,j}.I$ is the interval associated with $A_{i,j}$, that is the i-th instance of interval numerical attribute A_j and $A_j.D = [A_j.l, A_j.u]$

$$\begin{aligned} &A_{j,i}.I = [A_{j,i}.l, A_{j,i}.u] \\ &A_j.l \leq A_{j,i}.l, \; A_{j,i}.u \leq A_j.u. \end{aligned} \tag{3}$$

3.2 Algorithm

In this section, we present our novel algorithm, *UAC*. UAC can be applied to all datasets modeled with the general notion introduced in 3.1. Before applying UAC to uncertain numerical attributes, they are first transformed into uncertain categorical attributes using U-CAIM [31], which is an uncertain data discretizer. This is further explained in Section 3.3.1.

Building an associative classifier consists of two distinct steps: 1- Rule Extraction, 2- Rule Filtering. In this section each step of UAC is explained. Later, the procedure of classifying a new test instance is described.

3.2.1 Rule Extraction

In uncertain datasets, an association rule is considered strong if it is frequent and its *confidence (Conf)* is above a user defined threshold called *minimum confidence*. A ruleitem is frequent if its *Expected Support (ES)* is above a user defined threshold called *minimum expected support*. The definitions of the expected support and the confidence are as follows.

Definition If a is an itemset and c is a class label, expected support (ES) and confidence (Conf) of a ruleitem are calculated by Equation 4. Here, the ruleitem is denoted by $r = a \rightarrow c$ and T is the set of all transactions.

$$\begin{array}{lcl} ES(a) & = & \Sigma_{\forall t \in T} \Pi_{\forall i \in a} P(i \in t) \\ ES(a \rightarrow c) & = & \Sigma_{\forall t \in T, t.class=c} \Pi_{\forall i \in a} P(i \in t). \\ Conf(a \rightarrow c) & = & \frac{ES(a \rightarrow c)}{ES(a)}. \end{array} \tag{4}$$

Some studies have criticized expected support and defined another measure which is called probabilistic support [48] [7]. Probabilistic support is defined as the probability of an itemset to be frequent with respect to a certain minimum expected support. For example assume that in a dataset with two itemsets $a : x : 0.8, y : 0.5$ and $b : x : 0.2, y : 0.5$, we are to find the items with minimum expected support of 1. Using the expected support, both x and y are frequent, but the probability of x being frequent is 0.84. This probability for y is 0.75. In this case, probabilistic support is a better measure for measuring frequentness. However, probabilistic support increases the time complexity significantly. In addition, since we apply a rule

filtering step, we will set the minimum support to a very small value. So the cases mentioned above are not significant in the resulting classifier. Therefore, UAC uses the expected support.

uHARMONY defines another measure instead of confidence which is called *expected confidence*. The computation of this measure takes $O(|T|^2)$ time where $|T|$ is the number of instances. Computing confidence is only $O(1)$, thus we use confidence for efficiency reasons. Our experimental results in Section 3.3 empirically proves that our confidence based method can reach high accuracies.

Our rule extraction method is based on UApriori [2]. The candidate set is first initialized by all rules of form $a \rightarrow c$ where a is a single *attribute assignment* and c is a class label. After removing all infrequent ruleitems, the set of candidates is pruned by the pessimistic error rate method [34]. Each two frequent ruleitems with the same class label are then joined together to form the next level candidate set. The procedure is repeated until the generated candidate set is empty, meaning all the frequent ruleitems have been found. Those ruleitems that are strong (their confidence is above the predefined threshold) are the potential classification rules. In the next section, the potential ruleitems are filtered and the final set of rules is formed.

3.2.2 Rule Filtering

The outcome of the rule extraction is a set of rules called *rawSet*. Usually the number of ruleitems in rawSet is excessive. Excessive rules may have negative impact on the accuracy and the understandability of the classification model. To prevent this, CBA uses the database coverage method to reduce the set of rules. UAC benefits from the database coverage too, but can handle uncertainty. The purpose in UAC is similar to CBA, which is to remove the rules that do not increase the accuracy. The initial step of the database coverage method in UAC is to sort rules based on their absolute precedence to accelerate the algorithm. Absolute precedence in the context of uncertain data is defined as follows:

Definition: Rule r_i has absolute precedence over rule r_j or $r_i \succ r_j$, if *a*) r_i has higher confidence than r_j; *b*) r_i and r_j have the same confidence but r_i has

higher expected support than r_j; *c*) r_i and r_j have the same confidence and the same expected support but r_i have less items in its antecedent than r_j.

When data is "certain", confidence is a good and sufficient measure to examine whether a rule is the best classifier for an instance. But when uncertainty is present, there is an additional parameter in effect. To illustrate this issue, assume rules $r_1 : [m, t \rightarrow c_1]$ and $r_2 : [n \rightarrow c_2]$ having confidences of 0.8 and 0.7, respectively. It is evident that $r_1 \succ r_2$. However, for a test instance like $I_1 : [(m : 0.4), (n : 0.6), (t, 0.3) \rightarrow x]$ where x is to be predicted, which rule should be used? According to CBA, r_1 should be used because its confidence is higher than that of r_2. However, the probability that I_1 satisfies the antecedent of r_1 is small, so r_1 is probably not a reliable rule. We solve this problem by including another measure called PI. PI or *probability of inclusion*, denoted by $\pi(r_i, I_k)$, is described as the probability by which rule r_i can classify instance I_k. PI is defined in Equation 5. In the example above $\pi(r_1, I_1)$ is only $0.3 \times 0.4 = 0.12$, while $\pi(r_2, I_1)$ is 0.6.

$$\pi(r_i, I_k) = \Pi_{w \in r_i} P(w \in I_k). \tag{5}$$

Next, we define *applicability*, denoted by $\alpha(r_i, I_k)$ in Equation 6. Applicability is the probability by which rule r_i correctly classifies instance I_k and is used as one of the main metrics in UAC. For the previous example, $\alpha(r_1, I_1) = 0.096$ and $\alpha(r_2, I_1) = 0.42$. Thus, it is more probable that I_1 is correctly classified by r_2 than r_1.

$$\alpha(r_i, I_k) = r_i.Conf \times \pi(r_i, I_k). \tag{6}$$

Now based on the applicability, we define the concept of relative precedence of rule r_i over rule r_j with respect to I_k. This is denoted by $r_i \succ_{[I_k]} r_j$ and is defined as follows:

Definition: Rule r_i has relative precedence over rule r_j with respect to instance I_k denoted by $r_i \succ_{[I_k]} r_j$, if: *a*) $\alpha(r_i, I_k) > \alpha(r_j, I_k)$ *b*) r_i and r_j have the same applicability with respect to I_k but r_i has absolute precedence over r_j.

Having $r_i \succ_{[I_k]} r_j$ implies that r_i is "more reliable" than r_j in classifying I_k. It is evident from the definition, that the concept of "more reliable" rule in an uncertain data classifier is relative. One rule can be more reliable than the other when dealing

with an instance, and the opposite may be true for another instance. In the previous example, r_2 has relative precedence over r_1, even though r_1 has absolute precedence over r_2.

UAC uses the relative precedence as well as the absolute precedence to filter rawSet. The database coverage algorithm of UAC has 3 stages that are explained below.

Stage 1: Finding ucRules and uwRules

After sorting rawSet based on the absolute precedence, we make one pass over the dataset. On this pass we link each instance in the dataset to two rules in rawSet: $ucRule$ and $uwRule$. $ucRule$ is the rule with the highest relative precedence that correctly classifies i. In contrast, $uwRule$ is the rule with the highest relative precedence that wrongly classifies i. The pseudocode for the first stage is presented in Algorithm 2.

In Algorithm 2, three sets are declared. U contains all the rules that classify at least one training instance correctly. Q is the set of all $ucRule$s which have relative precedence over their corresponding $uwRule$s with respect to the associated instances. If $i.uwRule$ has relative and absolute precedence over the corresponding $ucRule$, a record of form $< i.id, i.class, ucRule, uwRule >$ is put in A. Here, $i.id$ is the unique identifier of the instance and $i.class$ represents the class label.

To find the corresponding $ucRule$ and $uwRule$ for each instance, the procedure starts at the first rule of the sorted rawSet and descends. For example, if there is a rule that correctly classifies the target instance and has applicability of α, we pass this rule and look for the rules with higher applicabilities to assign as $ucRule$. Searching continues only until we reach a rule that has a confidence of less than α. Clearly, this rule and rules after it (with less confidence) have no chance of being $ucRule$. The same applies to $uwRule$. Also as shown in Algorithm 2 lines 4 and 6, the applicability values of $ucRule$ and $uwRule$ are stored to expedite the process for the next stages.

The purpose of the database coverage in UAC is to find the best classifying rule (coverage) for each instance in the dataset. The covering rules are then contained

in the final set of rules and others are filtered out. The best rule, that is the covering rule, in CBA is the highest precedence rule that classifies an instance. This definition is not sufficient for UAC because the highest precedence rule may have a small PI. To prevent this situation, we set a lower bound on the applicability of the covering rule. This lower bound is determined by the algorithm itself, as we explain in the following sections, and each individual instance has its specific lower bound. Setting a lower bound on PI decreases the effect of noise by excluding the unlikely predictors from the decision making process.

uHARMONY also takes a similar approach by setting a predefined lower bound on PI value of the covering rule. We do not predefine the lower bound, because of its disadvantages. Clearly, not only estimating the suitable lower bound is critical, but it is also intricate, and even in many cases impossible. When predicting a label for an instance, rules that have higher PI than the lower bound are treated alike. To ameliorate this effect, it is necessary to set the lower bound high enough to avoid low probability rules covering the instances. But then, it is possible that the only classifying rules for some of the instances are not above that lower bound and removed. Additionally, setting a predefined lower bound filters out usable information, while the purpose of the uncertain data classifiers is to use all of the available information. Moreover, having a single bound for all of the cases is not desirable. Different instances may need different lower bounds.

Given all the above reasons, we need to evaluate the suitable lower bound for each instance. Definition of the covering rule in UAC is as follows, where we use the applicability of $i.ucRule$ as our lower bound for covering i.

Definition: Rule r covers instance i if: *a*) r classifies at least one instance correctly; *b*) $\pi(r, i) > 0$; *c*) $\alpha(r, i) > \alpha(i.ucRule, i) = cApplic$. *d*) $r \succ i.ucRule$

$cApplic$ represents the maximum rule applicability to classify an instance correctly. Thus, it is the suitable lower bound for the applicability of the covering rules. This will ensure that each instance is covered with the best classifying rule ($ucRule$) or a rule with higher relative and absolute precedence than $ucRule$. In the next two stages, we remove the rules that do not cover any instance from rawSet.

Algorithm 2 UAC Rule Filtering: Stage 1

1: $Q = \emptyset; U = \emptyset; A = \emptyset$
2: **for all** $i \in Dataset$ **do**
3: $i.ucRule = firstCorrect(i)$
4: $i.cApplic = \alpha(i.ucRule, i)$
5: $i.uwRule = firstWrong(i)$
6: $i.wApplic = \alpha(i.uwRule, i)$
7: $U.add(ucRule)$
8: $ucRule.covered[i.class] + +$
9: **if** $(ucRule \succ_{[i]} uwRule)$ and $ucRule \succ uwRule$ **then**
10: $Q.add(ucRule)$
11: $flag(ucRule)$
12: **else**
13: $A.add(< i.id, i.class, ucRule, uwRule >)$
14: **end if**
15: **end for**

Stage 2: Managing Replacements

In this stage, cases that were stored in A at Stage 1 are managed. Algorithm 3 displays the pseudocode for Stage 2. A contains all cases where $i.uwRule$ has relative and absolute precedence over $i.ucRule$, thus $i.ucRule$ may not cover i. If $i.uwRule$ is flagged in Stage 1, i is covered by $i.uwRule$ (lines 3, 4, and 5). Otherwise based on the definition of the covering rule in Stage 1, i may get the coverage by the other rules such as w which have the following characteristics: *a*) w classifies i incorrectly; *b*) w has relative precedence over $i.ucRule$ with respect to i; *c*) w has absolute precedence over $i.ucRule$.

Function $allCoverRules$ (line 7) finds all such rules as w within U, which are called the replacements of $i.ucRule$. The replacement relation is stored in a DAG (directed acyclic graph) called $RepDAG$. In RepDAG, each parent node has a pointer to each child node via the $replace$ set (line 12). The number of incoming edges is stored in $incom$ (line 14). Each node represents a rule and each edge represents a replacement relation.

Each rule has a $covered$ array in UAC where $r.covered[c]$ is used to store the total number of instances covered by r and labeled by class c. If $r.covered[r.class] = 0$, then r does not classify any training instance correctly and is filtered out. Starting

from line 22, we traverse RepDAG in its topologically sorted order to update the $covered$ array of each rule. Rule r_i comes before r_j in the sorted order, if $r_i \succ r_j$ and there is no instance such as I_k where $r_j \succ_{[I_k]} r_i$. If a rule fails to cover any instance correctly (line 26), it does not have any effect on the $covered$ array of the rules in its replace set. At the end of stage 1, enough information has been gathered to start the next stage, which finalizes the set of rules.

Stage 3: Finalizing Rules

At Stage 3, the set of rules is finalized. Algorithm 4 presents the pseudocode of Stage 3. In this stage, UAC filters the rules based on a greedy method of error reduction. Function $computeError$ counts the number of instances that are covered by rule r but have a different class label than $r.class$. The covered instances are then removed from the dataset. Function $addDefaultClass$ finds the most frequent class label among the remaining instances (line 6). In line 8, the number of instances correctly classified by the default class is calculated. $totalError$ is the total errors made by the current rule r and the default class. In fact, each rule with positive coverage over its class, is associated with a particular $totalError$, $defClass$, and $defAcc$ (line 10). After processing the rules, we break the set of rules from the minimum error and assign $default$ and $defApplic$. $defApplic$ is used in rule selection as an estimate of applicability of the default class.

Our rule filtering algorithm has a runtime of $O(|T| \times |R|)$ in the worst case scenario, where $|T|$ is the number of instances in the dataset and $|R|$ is the size of rawSet. This is considering the assumption that the number of attributes is negligible compared to the number of instances, which is a fact in most of the real world problems. The worst case scenario is when at Stage 1, at least one $ucRule$ or $uwRule$ is the last rule in the sorted rawSet. This case rarely happens because the rules are sorted based on their absolute precedence. UAC also makes slightly more than one pass over the dataset in the rule filtering step. Passes are made in Stage 1 and 2. Note that array A is usually small, given that most of the instances are usually classified by the highest ranked rules. The number of passes is an important point, because the dataset may be very large. Especially for datasets that can not

Algorithm 3 UAC Rule Filtering: Stage 2

```
1: RepDAG = ∅
2: for all < i.id, y, ucRule, uwRule >∈ A do
3:    if flagged(uwRule) then
4:       ucRule.covered[y] − −
5:       uwRule.covered[y] + +
6:    else
7:       wSet = allCoverRules(U, i.id, ucRule)
8:       if !RepDAG.contains(ucRule) then
9:          RepDAG.add(ucRule)
10:      end if
11:      for all w ∈ wSet do
12:         w.replace.add(< ucRule, i.id, y >)
13:         w.covered + +
14:         ucRule.incom + +
15:         if !w ∈ RepDAG then
16:            RepDAG.add(w)
17:         end if
18:      end for
19:      Q = Q.add(wSet)
20:   end if
21: end for
22: S ← set of all nodes with no incoming edges
23: while S ≠ ∅ do
24:   r = S.next() {next removes a rule from the set}
25:   for all < ucRule, id, y >∈ r.replace do
26:      if (r.covered[r.class] > 0) then
27:         if id is covered then
28:            r.covered[y] − −
29:         else
30:            ucRule.covered[y] − −
31:            Mark id as covered.
32:         end if
33:      end if
34:      ucRule.incom − −
35:      if ucRule.incom = 0 then
36:         S.add(ucRule)
37:      end if
38:   end for
39: end while
```

be loaded into memory at once, it is not efficient to make multiple pases. This is an advantage for UAC over UCBA, which passes over the dataset once for each

rule in rawSet. The next section explains the rule selection that is the procedure of classifying test instances based on the set of rules.

Algorithm 4 UAC Rule Filtering: Stage 3

1: $C = \emptyset$
2: **for all** $r \in Q$ **do**
3: **if** $r.covered[r.class] > 0$ **then**
4: $finalSet.add(r)$
5: $ruleErrors+ = computeError(r)$
6: $defClass = addDefaultClass()$
7: $defErrors = computeDefErr(defClass)$
8: $defAcc = addDefAcc(uncovered(D) - defErrors)$
9: $totalError = defErrors + ruleErrors$
10: $C.add(r, totalError, defClass, defAcc)$
11: **end if**
12: **end for**
13: Break C from the rule with minimum error
14: C contains the final set of rules
15: $default = defClass.get(C.size)$
16: $defApplic = \frac{defAcc.get(C.size)}{|T|}$

3.2.3 Rule Selection

Rule selection is the procedure of classifying a test instance. In the previous sections, excessive rules were filtered out from rawSet. The remaining set of rules is called finalSet, and classifies the test instances. UAC selects one classifying rule for each instance. The selected classifying rule has the highest relative precedence with respect to the test instance.

The role of the default class ($default$ in Algorithm 4 line 15) is to reduce the number of rules. The default class predicts the labels of those instances that are not classified by the rules in the finalSet. So the best predicting label for some of the test instances may be default class. But UAC may prefer rules with small PI values to the default class if we follow the procedure of "certain" data classifiers. To prevent this, $defApplic$ is used as an estimate for applicability of the default rule. This value shows the number of training instances that were expected to be classified by the default rule. For example, when two classes, such as a and b, have the same population in the dataset but no rule labeled b exists, default rule has a

very important role. Consequently, the value of the default applicability is high. As a result, if the highest precedence rule with respect to a test instance has less applicability than the default rule, the default rule will predict the label for that.

3.3 Experiments and Results

We use an empirical study to compare UAC against the existing rule based methods. In all of the reported experiments on UAC, the minimum support is set to 1%, the minimum confidence to 50% and the maximum number of mined association rules to $80,000$. Each reported number is an average over 10 repetitions of 10-fold cross validations. The value of the minimum support is set to a very small number so no pattern is left out. Later in the filtering step we filter the unwanted patterns. The minimum confidence is set to 50% so we are sure that the rules we pick are more accurate than blind chance.

Since there is no known public repository of uncertain datasets, we synthetically added uncertainty to 30 well known UCI datasets. This method was employed by all the studies in the field including uHARMONY, DTU and uRule, uncertain svm, UCBA, etc., and gives a close estimation of the classifier performance in the real world problems. Although we cannot compare the algorithms applied to the real world problems, using synthetically added probabilities we are enabled to limit the uncertainty level and perform a valuable comparison. We selected the same datasets as in [14] to compare our method with the results reported in their paper for uHARMONY, uRule and DTU. This also ensures that we did not choose only the datasets on which our method performs better. For convenience, we use the same set of databases to compare UAC to UCBA.

3.3.1 Accuracy Comparisons

To compare our method against other classifiers, we employ averaging technique and case by case comparison [12]. The same method was employed by many other studies including CBA, uHARMONY, DTU and uRule to prove the better performance of their algorithms. Tables 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6 provide a com-

parison between UAC and other existing rule based methods in terms of accuracy for various levels of uncertainty. The reported accuracies for uHARMONY (#3), DTU (#4) and uRule (#5) are reproduced using the provided implementation from [15]. We applied UAC (#2) to the same datasets generated by the same procedure of adding uncertainty as [14] to make the comparison meaningful. Value $*$ shows that the classifier has run out of memory resources in the experiments. In Tables 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6 uncertainty level is Ux@y meaning that datasets have x percent uncertainty, where y of the attributes with the highest information gain are uncertain. To add a level x uncertainty to an attribute, its is attached with a $1 - x/100$ probability and the remaining $x/100$ is distributed randomly among the other values present in the domain. The accuracies in this table are reported on already discretized versions of dataset that are available online and referenced in [14]. The uncertainty model is set to be complete, as the algorithms are explicitly mentioned to work with this model.

In each table, *Average* shows the average accuracy over all cases where the classifier has reached an answer. Averaging on all datasets excluding those where at least one classifier has run out of resources, we get *AverageToCompare*. *Number of wins* shows the number of cases where the classifier has reached the maximum accuracy among all four classifiers. Bold values in the table show the maximum accuracy reached in that case.

The accuracies reported show that in most cases UAC has reached higher accuracies. This is evident from comparing the number of wins among the classifiers. In addition, UAC performs more accurately on average too.

Table 3.7 presents the accuracies reported by UAC, using a different method of adding uncertainty. In this experiment, we added uncertainty to the original versions of the datasets from UCI using our model explained in Section 3.1. UCBA is selected as our baseline because it is also inspired by CBA. The purpose of this is to measure the accuracy of UAC in more realistic cases, as well as to study the effectiveness of the adaptations made by the two algorithms. Both methods were implemented in Java and the experiments were performed on a system with a $2.2GHz$ processor and $4G$ of memory.

Table 3.1: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U10@1				U10@2			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	**85.0**	82.8	82.3	81.6	**85.0**	85.0	82.2	82.0
balance	84.5	**87.6**	59.3	67.3	84.8	**87.4**	64.4	69.0
bands	**78.6**	67.8	60.9	*	**78.5**	65.7	*	*
breast	91.1	**92.2**	84.3	92.5	91.0	90.2	90.1	**92.1**
car	**89.0**	86.5	87.5	83.2	**89.0**	88.4	69.9	67.4
cmc	41.8	**50.1**	45.1	41.3	41.1	46.9	**47.9**	40.7
credit	**85.0**	81.8	81.5	82.2	**85.0**	82.1	84.0	82.7
echocardiogram	**94.3**	92.0	89.9	89.4	**93.8**	89.1	89.8	88.5
flag	62.3	**63.0**	61.3	59.8	61.9	59.4	**62.8**	56.4
german	**74.0**	71.4	66.4	69.4	**74.0**	68.6	67.7	67.4
heart	**80.0**	56.6	48.3	47.9	**78.7**	53.3	52.1	49.4
hepatitis	**84.4**	80.8	75.7	77.0	**84.4**	78.8	76.0	76.1
horse-colic	80.2	82.5	79.4	**85.0**	80.3	82.9	82.6	**84.5**
monks-1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	73.2	**100.0**
monks-2	**77.5**	67.6	61.5	62.5	**76.7**	69.3	63.6	60.6
monks-3	**97.9**	92.7	94.0	95.6	**97.7**	92.8	78.8	78.9
mushroom	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
pima	**73.2**	64.9	59.3	66.8	**73.1**	64.6	61.9	65.8
postoperative	**69.0**	67.2	64.7	66.4	68.7	64.9	66.1	**68.8**
promoters	**84.2**	83.6	70.6	70.3	**83.1**	81.7	70.5	67.8
spect	**84.0**	83.2	74.4	81.2	**85.0**	84.4	77.6	80.1
survival	**73.7**	71.6	69.0	68.7	**73.7**	69.5	72.1	68.6
tae	**55.0**	51.0	37.8	38.1	**54.0**	49.1	45.1	30.7
tic-tac-toe	**100.0**	**100.0**	79.5	**100.0**	**100.0**	**100.0**	77.2	**100.0**
vehicle	**71.2**	61.9	59.1	*	**70.8**	61.4	62.9	*
voting	**94.1**	94.0	89.9	90.7	**94.1**	91.6	90.1	87.7
wine	**92.0**	51.0	36.5	40.8	**91.5**	51.9	38.8	38.2
zoo	**95.5**	92.8	85.6	86.9	**95.0**	90.3	87.2	86.9
Average	**82.1**	77.7	71.6	74.8	**81.8**	76.8	71.6	72.7
AverageToCompare	**81.2**	77.2	70.8	73.2	**81.1**	76.1	70.9	71.1
Number of Wins	**23.0**	7.0	2.0	4.0	**22.0**	4.0	3.0	6.0

Table 3.2: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U10@4				U10@8			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	80.2	**85.4**	83.6	84.3	80.0	81.7	83.1	**83.2**
balance	84.9	**89.3**	56.3	62.9	84.8	**88.4**	54.3	59.2
bands	**78.4**	58.6	*	*	**76.8**	64.5	*	*
breast	**94.3**	65.5	91.3	94.6	**94.3**	88.9	90.1	91.0
car	**89.3**	77.7	70.0	70.0	**83.8**	81.8	66.1	68.8
cmc	43.6	47.6	**50.1**	44.3	41.5	**46.1**	43.6	41.1
credit	78.1	**86.0**	84.3	74.3	78.0	83.8	82.9	69.8
echocardiogram	92.0	**93.3**	92.4	87.0	**93.5**	92.2	91.2	86.2
flag	45.7	52.4	**59.3**	44.8	42.5	51.9	**56.4**	49.7
german	71.9	69.6	**72.3**	70.1	**73.2**	70.8	66.1	67.4
heart	**77.3**	56.6	53.0	52.4	**77.2**	52.9	51.3	47.2
hepatitis	81.5	**82.5**	80.0	79.4	**80.8**	75.5	78.0	78.7
horse-colic	72.4	82.9	**85.3**	*	72.3	**82.2**	81.4	*
monks-1	**99.0**	91.4	74.6	*	**99.0**	98.0	71.0	73.8
monks-2	**75.5**	65.7	65.7	65.7	**75.4**	66.4	62.1	61.9
monks-3	**98.1**	96.4	80.0	68.1	**98.1**	93.6	77.2	63.0
mushroom	**100.0**	97.5	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	98.8
pima	**73.8**	65.1	65.1	67.3	**72.7**	63.5	*	61.1
postoperative	58.0	69.8	**70.0**	**70.0**	58.0	65.5	**67.8**	66.2
promoters	66.0	69.0	**71.7**	61.3	67.2	**76.4**	73.8	63.8
spect	**81.8**	80.2	79.0	81.6	80.0	**81.3**	76.4	76.4
survival	**74.0**	73.5	73.5	72.5	**73.9**	70.9	71.8	70.5
tae	**52.0**	45.0	48.3	33.8	**50.8**	41.0	46.6	36.5
tic-tac-toe	**90.8**	76.2	72.7	81.5	**91.0**	74.0	66.8	72.6
vehicle	**69.8**	62.4	64.8	*	**67.7**	57.4	*	*
voting	91.1	92.9	94.5	**94.9**	92.1	**92.8**	89.8	88.6
wine	**87.9**	51.1	42.1	41.6	**86.0**	44.1	49.4	36.7
zoo	**92.3**	88.8	92.1	89.1	**85.0**	79.6	76.1	74.0
Average	**78.6**	74.0	73.0	70.5	**77.7**	73.8	67.7	67.4
AverageToCompare	**77.3**	73.1	72.0	69.4	**77.5**	74.2	67.7	66.8
Number of Wins	**16.0**	5.0	7.0	3.0	**18.0**	7.0	3.0	1.0

Table 3.3: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U20@1				U20@2			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	**82.7**	82.0	80.6	75.6	82.3	**84.4**	81.3	77.8
balance	82.3	**86.8**	62.0	67.1	84.1	**86.3**	63.8	68.7
bands	**77.7**	65.4	62.3	*	**77.7**	66.3	*	*
breast	90.1	**91.8**	86.8	90.0	90.0	**93.0**	87.8	90.2
bridges-v1	**63.4**	61.2	49.9	57.0	63.3	**63.4**	51.6	58.2
bridges-v2	**64.2**	63.0	60.7	56.9	**64.3**	59.1	64.1	51.8
car	**89.9**	86.5	88.7	77.7	**88.4**	86.7	67.0	67.0
cmc	41.1	**48.9**	48.2	42.0	41.1	45.7	**49.7**	42.3
credit	**85.0**	82.1	79.4	77.9	**83.2**	83.0	82.4	80.1
echocardiogram	**94.0**	90.6	88.8	90.6	**93.8**	90.9	82.6	75.7
flag	62.5	**63.2**	62.1	61.0	61.6	59.7	**64.4**	58.2
german	**74.2**	68.3	65.7	67.0	**74.0**	70.3	68.5	67.1
heart	**78.3**	53.2	50.3	47.1	**76.5**	56.4	50.8	47.4
hepatitis	**84.8**	80.9	77.8	75.1	**84.2**	79.6	78.0	76.1
horse-colic	80.3	**85.0**	83.0	84.7	80.7	84.7	83.3	**85.4**
monks-1	**100.0**	95.4	92.2	92.5	**100.0**	**100.0**	72.4	88.6
monks-2	**77.0**	71.4	63.2	61.6	**77.0**	71.8	61.9	59.3
monks-3	**97.3**	92.0	94.6	94.8	**96.4**	93.0	76.3	76.3
mushroom	**100.0**	96.2	**100.0**	96.4	98.0	97.6	97.9	97.9
pima	**73.4**	64.7	64.3	66.2	**73.4**	65.6	62.9	65.3
postoperative	**69.4**	68.3	67.5	65.9	**68.1**	67.6	67.4	67.4
promoters	**83.6**	82.3	75.4	68.0	**82.7**	82.4	71.9	67.2
spect	81.7	**82.1**	77.6	81.7	80.4	**83.0**	77.3	80.3
survival	**73.7**	68.8	72.4	69.1	**73.7**	71.0	72.2	69.6
tae	**54.3**	49.2	42.7	37.5	**54.3**	46.9	42.4	35.7
tic-tac-toe	**100.0**	**100.0**	82.7	**100.0**	**100.0**	100.0	75.9	100.0
vehicle	**71.2**	62.6	61.9	*	**70.6**	61.2	62.4	*
voting	**94.0**	93.2	90.2	88.2	**93.9**	90.1	92.2	89.9
wine	**90.8**	50.8	35.7	41.1	**91.0**	49.2	36.3	36.3
zoo	**96.0**	92.3	89.3	85.7	**94.5**	89.7	86.4	86.4
Average	**80.4**	76.0	71.9	72.1	**80.0**	76.0	70.0	70.2
AverageToCompare	**80.4**	75.7	71.6	72.0	**79.9**	75.7	69.6	69.9
Number of Wins	**22.0**	7.0	1.0	1.0	**19.0**	6.0	2.0	1.0

Table 3.4: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U20@4				U20@8			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	80.1	**86.4**	75.9	75.5	81.2	**84.7**	81.9	75.1
balance	83.8	**90.1**	54.9	61.4	80.0	**87.8**	54.2	61.0
bands	**77.7**	68.7	*	*	**75.0**	65.3	*	*
breast	90.0	**92.5**	89.2	91.5	88.6	**93.7**	84.5	85.7
car	**89.2**	78.0	65.6	67.0	**88.9**	73.2	66.9	66.9
cmc	40.8	**48.3**	45.9	41.7	40.8	**45.2**	44.7	42.4
credit	78.1	**82.2**	78.6	72.2	77.8	**84.6**	82.2	61.9
echocardiogram	**93.4**	87.7	73.1	85.0	**93.5**	90.9	90.0	77.8
flag	43.4	**53.8**	45.6	36.5	42.6	55.1	**59.0**	45.6
german	**71.8**	70.9	69.3	64.9	**71.0**	68.1	66.5	66.5
heart	**76.6**	57.1	52.0	47.9	**77.0**	53.0	46.4	41.7
hepatitis	**80.9**	79.7	77.8	77.6	**80.5**	75.5	77.3	76.1
horse-colic	80.9	**84.5**	83.2	*	78.6	80.1	**81.2**	*
monks-1	**100.0**	**100.0**	72.5	67.0	**100.0**	**100.0**	71.9	69.4
monks-2	**76.5**	70.2	62.8	63.1	**74.2**	62.5	62.7	62.7
monks-3	**96.4**	93.1	76.7	68.1	**96.4**	93.3	77.3	57.6
mushroom	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	94.2	98.0	98.7
pima	**73.5**	66.8	63.8	64.9	**73.9**	65.3	*	61.5
postoperative	68.1	65.4	65.8	**68.5**	**71.0**	66.7	67.3	67.3
promoters	66.0	**77.6**	55.7	53.4	66.9	**79.2**	72.2	56.1
spect	81.0	**83.6**	74.5	77.6	**81.0**	77.0	77.1	77.4
survival	**73.7**	70.0	70.0	70.9	**73.6**	71.5	71.6	71.0
tae	**53.2**	42.5	38.5	29.9	**50.1**	43.8	37.9	35.2
tic-tac-toe	**99.0**	96.4	70.5	75.2	**98.3**	94.6	63.1	67.6
vehicle	**69.7**	62.3	62.8	*	**63.2**	56.2	*	*
voting	**92.0**	91.7	88.5	85.6	91.9	**92.3**	84.3	85.9
wine	**87.2**	47.6	38.1	38.1	**87.1**	44.1	48.8	35.4
zoo	**93.6**	88.2	87.2	86.9	**93.0**	84.5	70.1	71.1
Average	**79.2**	76.3	68.1	66.8	**78.4**	74.4	69.5	64.7
AverageToCompare	**78.4**	75.5	66.7	65.8	**77.8**	74.8	67.7	63.7
Number of Wins	**18.0**	11.0	1.0	2.0	**19.0**	8.0	2.0	0.0

Table 3.5: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U40@1				U40@2			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	**81.0**	77.4	73.0	74.1	**80.7**	79.4	73.2	73.1
balance	82.0	**84.3**	63.2	62.8	**82.0**	82.5	62.8	60.3
bands	**75.2**	68.9	64.1	*	**75.2**	68.8	*	*
breast	89.1	91.4	90.0	**92.6**	88.8	89.9	92.1	**91.4**
car	**80.8**	73.3	70.4	66.7	**75.0**	68.8	67.4	67.4
cmc	41.1	**49.5**	46.4	41.5	41.0	46.0	**47.0**	42.0
credit	**83.8**	79.3	71.7	71.0	**83.0**	82.6	73.7	74.8
echocardiogram	**93.7**	92.2	91.3	89.4	**91.8**	89.3	70.5	65.1
flag	62.5	60.8	62.2	**62.6**	62.4	61.5	**64.0**	57.3
german	**74.2**	70.4	69.0	68.5	**74.2**	70.0	67.9	66.4
heart	**76.9**	55.3	52.2	47.7	**76.3**	56.6	52.6	49.2
hepatitis	**84.8**	79.0	74.6	74.7	**84.8**	79.1	75.8	73.9
horse-colic	80.3	82.9	82.1	**86.4**	80.3	82.5	83.6	**85.7**
monks-1	**100.0**	**100.0**	81.9	82.4	**100.0**	**100.0**	71.8	82.6
monks-2	74.7	66.0	61.0	60.8	**73.7**	63.0	62.4	59.6
monks-3	97.0	93.9	77.4	78.6	**97.0**	93.3	70.9	74.5
mushroom	**100.0**	**100.0**	**100.0**	**100.0**	**99.9**	96.0	98.4	98.4
pima	**73.4**	66.1	63.9	65.4	**73.4**	65.8	64.5	63.3
postoperative	**69.4**	66.0	66.5	65.7	67.0	**69.3**	68.3	68.3
promoters	**82.0**	79.3	71.7	75.3	**82.0**	77.0	74.3	71.4
spect	**81.5**	82.1	77.5	79.5	81.7	**82.7**	80.6	81.7
survival	**73.7**	71.1	71.1	66.9	**73.7**	68.7	71.5	69.5
tae	**54.2**	50.0	43.3	39.9	**53.3**	46.8	40.1	33.5
tic-tac-toe	**99.4**	94.7	68.1	85.0	**98.0**	92.2	66.7	72.6
vehicle	**71.1**	62.7	61.4	*	**68.7**	62.5	60.8	*
voting	**93.0**	88.9	86.4	87.1	**92.5**	88.5	86.8	85.2
wine	**89.3**	48.8	36.2	39.5	**85.5**	48.9	38.8	38.8
zoo	**94.4**	89.5	87.6	85.9	**94.5**	90.0	87.6	87.6
Average	**80.7**	75.9	70.2	71.1	**79.9**	75.1	69.4	69.0
AverageToCompare	**80.0**	75.4	69.6	70.2	**79.4**	74.8	68.9	68.1
Number of Wins	**21.0**	4.0	1.0	4.0	**22.0**	3.0	2.0	2.0

Table 3.6: %Accuracy, reported by rule based classifiers on datasets modeled based on [14]

	U40@4				U40@8			
dataset	UAC	uHarmony	DTU	uRule	UAC	uHarmony	DTU	uRule
australian	**79.8**	79.6	69.5	69.2	79.5	**83.2**	61.0	59.0
balance	83.1	**87.3**	52.1	64.9	82.0	**88.1**	53.5	64.2
bands	**76.0**	66.0	*	*	**74.9**	66.5	*	*
breast	86.5	89.4	85.7	**90.3**	87.2	**91.0**	85.1	76.6
car	**74.8**	68.8	68.8	68.8	**74.8**	68.6	67.6	67.6
cmc	40.5	**46.0**	40.2	40.1	43.3	**46.5**	40.0	40.6
credit	76.2	**82.4**	73.9	72.1	77.6	**81.7**	62.3	60.2
echocardiogram	**93.4**	89.6	68.8	65.0	**94.4**	89.5	62.6	61.8
flag	**58.0**	56.1	46.5	37.4	**61.5**	58.5	36.1	41.3
german	**74.0**	69.4	65.3	67.9	**71.0**	70.0	66.9	66.4
heart	**76.1**	53.0	48.5	49.8	**75.5**	49.8	48.2	44.3
hepatitis	**80.7**	75.8	76.5	78.3	**82.3**	80.7	76.5	77.2
horse-colic	79.6	**84.4**	82.8	*	78.0	79.9	**83.2**	*
monks-1	**78.9**	72.2	72.2	64.4	**99.9**	94.8	73.2	69.6
monks-2	**73.9**	61.8	62.1	63.0	**76.9**	63.2	62.5	62.5
monks-3	**96.9**	94.6	73.1	73.7	**96.9**	93.3	73.7	59.0
mushroom	**100.0**	**100.0**	**100.0**	**100.0**	97.2	96.0	**98.9**	98.9
pima	**71.1**	65.6	63.3	63.8	**69.9**	61.6	*	63.6
postoperative	64.1	**68.9**	66.9	66.9	64.1	66.0	**67.8**	**67.8**
promoters	64.3	**67.4**	49.0	52.7	69.5	**72.1**	50.6	55.3
spect	**81.8**	77.5	74.8	73.0	**79.9**	77.5	77.0	75.5
survival	**73.7**	70.8	72.3	72.3	**73.6**	69.9	70.5	70.5
tae	**49.3**	39.1	33.0	33.0	**46.9**	40.7	30.9	30.9
tic-tac-toe	**95.3**	87.4	63.6	62.0	**89.0**	81.2	64.1	64.1
vehicle	**68.0**	60.3	60.1	*	**63.0**	58.7	*	*
voting	**91.9**	87.4	85.2	81.0	**91.9**	88.8	80.2	76.6
wine	**80.3**	46.5	41.1	41.1	**64.8**	40.6	47.0	34.1
zoo	**95.0**	89.9	90.7	87.8	**81.6**	73.3	68.3	68.3
Average	**77.3**	72.7	66.1	65.5	**76.7**	72.6	64.3	62.2
AverageToCompare	**76.7**	72.1	65.0	64.6	**76.3**	72.9	62.1	60.8
Number of Wins	**21**	7	1	2	**19**	6	3	1

In the columns titled as **CBA** in Table 3.7, the accuracy reported by CBA on the original dataset (with no uncertainty) is presented. The columns titled as **ulev** show the uncertainty level and columns named as **UAC** and **UCBA** contain the accuracies reported by UAC and UCBA, respectively. For each algorithm, the accuracy on both interval based and sampled based numerical attributes are reported under columns named **Intrv** and **Sam**, respectively. Value $*$ means that the 10 repetitions did not finish after 3 days of time.

In Table 3.7, *Average* shows the average accuracy over all cases where the classifier has reached an answer. Averaging on all datasets excluding those where UCBA classifier has run out of resources, we get *AverageToCompare*. *Number of wins* shows the number of cases where the classifier has reached the maximum accuracy among the two classifiers. Bold values show the maximum accuracy reached for that case. We report accuracies for three uncertainty levels to explore the effect of the uncertainty on the performance. The uncertainty is added to all of the attributes present in the dataset. The process of generating an attribute with a particular uncertainty level or $ulev$ is as follows.

Instances of categorical uncertain attributes contain a list of $< value : probability >$ pairs. Each of the other values in the domain is false. To generate an uncertain instance, we randomly decide whether to omit the actual true value with a probability of $ulev$. Indeed, this method would cause algorithms to achieve lower accuracies, but it generates more realistic datasets. If the true value is decided to be included, the probability associated to it is randomly generated between $[1 - ulevel, 1]$. Next, a set of false values, called *false-set*, are selected randomly from the domain. The size of false-set is selected randomly from a uniform distribution and all of the false values have equal chances to be included. A randomly generated number less than $ulev$ is then attached to each false value to make a <value:probability> pair, so that the sum of probabilities is less than 1. If all of the possible values from the domain are added to the value list, then the probabilities add up to 1.

A slightly different method is adopted for the sampled numerical attributes. The true value may be omitted with a chance of $ulev$ and the false values are selected from the domain, which in this case is an interval. Since the size of false-set is infi-

Dataset	CBA	ulev	UAC		UCBA		Dataset	CBA	ulev	UAC		UCBA	
			intrv	Sam	intrv	Sam				intrv	Sam	intrv	Sam
australian	85	0.1	**83.5**	**84**	78.5	73.5	monks-1	100	0.1	**92.8**	**92.8**	70	70
		0.2	**81.5**	**81.8**	68	72.8			0.2	**86**	**86**	63.3	63.3
		0.3	**73.8**	**79.8**	63.8	70.8			0.3	**71.6**	**71.6**	61.8	61.8
balance	85.5	0.1	**82.3**	**82.3**	77.5	77.5	monks-2	77.5	0.1	63.5	63.5	**66**	**66**
		0.2	**76**	**76**	71.3	71.3			0.2	**66**	**66**	**66**	**66**
		0.3	**68**	**68**	65.3	65.3			0.3	**66**	**66**	**66**	**66**
bands	78.3	0.1	**74.5**	**75**	**74.5**	**75**	monks-3	98	0.1	**92.5**	**92.5**	78.8	78.8
		0.2	**68.8**	**72**	**68.8**	**72**			0.2	**89**	**89**	72.3	72.3
		0.3	**67.5**	**66**	**67.5**	**66**			0.3	**84.2**	**84.2**	66.8	66.8
breast	94.8	0.1	**92.5**	**92.5**	87.3	87.3	mushroom	100	0.1	98.5	98.5	*	*
		0.2	**87**	**87**	82	82			0.2	95	95	*	*
		0.3	**85**	**85**	77	77			0.3	92	92	*	*
horse	81.3	0.1	74.5	**76.3**	**74.8**	73.3	pima	74.8	0.1	**72.3**	**71.3**	71	65.8
		0.2	**77.5**	**78**	72.5	71.5			0.2	**66**	**71.8**	65	68.8
		0.3	**72.5**	**74.3**	69.8	72			0.3	**65**	66.3	**65**	**66.8**
zoo	95	0.1	**90.5**	**90.5**	71.3	71.3	post_oper	68.8	0.1	**70.3**	**68.3**	70	64.5
		0.2	**82.3**	**82.3**	65.3	65.3			0.2	69	**71.3**	**70.3**	69
		0.3	**75.3**	**75.3**	48.3	48.3			0.3	70	67.3	**70.3**	**70.8**
car	89.8	0.1	**71**	**71.3**	70	70	promote	84.8	0.1	**72.3**	**72.3**	60	60
		0.2	**70**	**70**	**70**	**70**			0.2	72.5	72.5	*	*
		0.3	**70**	**70**	**70**	**70**			0.3	66.5	66.5	*	*
contracep	44	0.1	**43.5**	**43.8**	**43.5**	43	spect	82	0.1	**79**	**79**	**79**	**79**
		0.2	42.8	**42**	**43**	**42**			0.2	**79**	**81.8**	**79**	79
		0.3	**43.3**	**44**	42.3	43			0.3	**79**	**79**	**79**	**79**
credit	85	0.1	**83.5**	**83.8**	77	77.3	survival	73.5	0.1	73.8	**74**	**74**	**74**
		0.2	**79.3**	**81**	65.8	74.5			0.2	**74**	**74**	**74**	**74**
		0.3	**72.3**	**78**	61.3	71.8			0.3	**74**	73	**74**	**73.3**
echo	93.5	0.1	**98.8**	**90.8**	70.5	70.5	ta_eval	49.8	0.1	**45.8**	**42**	39.5	41
		0.2	**86.3**	**85.3**	65.3	66			0.2	**46.3**	**44.5**	43	41.8
		0.3	**85.5**	**85.5**	65	63			0.3	**41.8**	**42.3**	38	40.5
flag	63.5	0.1	**56.3**	**58.3**	43.8	48.3	tic-tac-toe	100	0.1	**83.5**	**83.5**	72.3	72.3
		0.2	**52.8**	**51.8**	41	43.3			0.2	**69.8**	**69.8**	67	67
		0.3	**50.8**	**49.8**	42	39			0.3	**65.3**	**65.3**	65.5	65.5
german	74	0.1	**70.5**	**70**	70	**70**	vehicle	72	0.1	**59.3**	**64.8**	53.5	53.5
		0.2	**70**	**70**	**70**	**70**			0.2	52.5	**64.8**	**55.3**	55.3
		0.3	**70**	**70**	**70**	**70**			0.3	46.3	**61**	**51.6**	51.6
heart	80.8	0.1	**82.5**	**80.3**	76	74.5	voting	94.3	0.1	**92.5**	**92.5**	78.3	78.3
		0.2	**79.5**	**77.5**	69.5	72.8			0.2	**89.8**	**89.8**	75	75
		0.3	**70.8**	**72**	66	70.8			0.3	**87.8**	**87.8**	72.5	72.5
hepatitis	84.3	0.1	**83.8**	**82**	79.5	79.3	wine	92	0.1	**92**	**91.8**	**92**	84
		0.2	**80.8**	**80.3**	79.3	79.5			0.2	**89.5**	**92.5**	82.8	81.8
		0.3	**79.3**	**79**	79	**79**			0.3	**86**	**88.5**	78	81.3
Average on all			**74.2**	**74.7**	67.3	67.7							
AverageToCompare			**73.5**	**74.1**	67.3	67.7							
Number of wins			**71**	**75**	25	20							

Table 3.7: comparison between UAC and UCBA

nite in this case, the size is limited by considering a maximum. In our experiments, we set the maximum size to 10.

An instance of a numerical attribute modeled by intervals is an interval with a length of $ulength = |A_j.u - A_j.l| \times ulev$, where $[A_j.l, A_j.u]$ is the domain. The centre of this interval is selected randomly, in a way that the actual value always belongs to this interval. Clearly, the actual value is placed in a random position in the interval, as opposed to the methods introduced in previous studies [32, 31] that always put the true value at the center. Our randomized generation process creates more realistic models of uncertain real datasets.

We discretize all numerical attributes before classification using U-CAIM [17]. U-CAIM covers both sampled and interval based data models and their modeling is compatible with ours. U-CAIM needs the cdf (cumulative distribution function) within the intervals to be specified. cdf is the anti-derivative function of the probability density function. Based on the characteristics of data, one can use any distribution with known cumulative density function. We assume normal distribution. Since U-CAIM is a supervised method, we apply it only on training sets within the folds and then use the discretization set to categorize the test data.

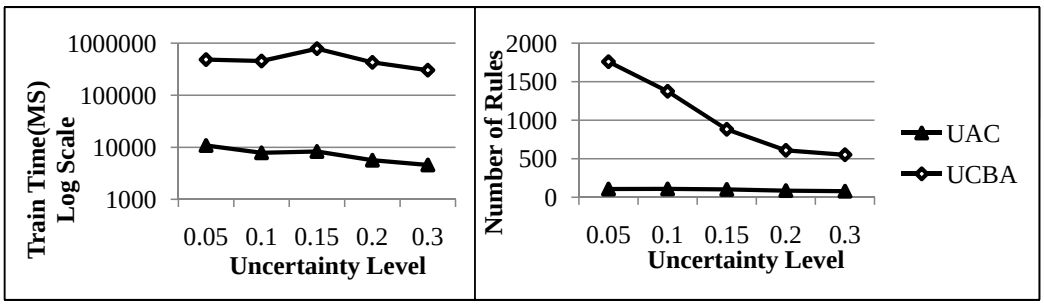
It is evident from the table that UAC outperforms UCBA in terms of accuracy, Especially in *monks-1*, *monks-3*, *voting*, *wine*, *echo* and *zoo* datasets. On average, UAC improves significantly upon the accuracy of UCBA. As uncertainty increases, the meaningful patterns in the data start to blur, which leads to accuracy drop in both algorithms. However based on the results reported in Table 3.7, accuracy decreases more slowly in UAC than UCBA.

3.3.2 Training Time and Number of Rules

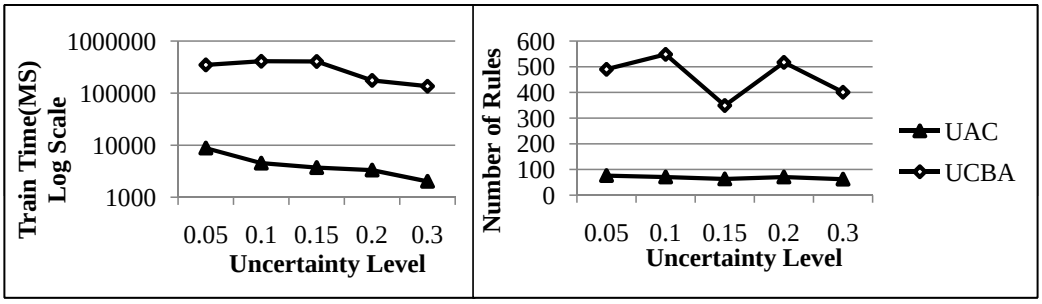
We compare the training time of UAC to UCBA and the number of rules in finalSet in Figure 3.1. The purpose of the comparison is to study the cost of the adaptations made by both algorithms in order to handle the uncertainty. We report the results of the two algorithms on two datasets, one containing only categorical attributes (*zoo*) and the other containing numerical attributes as well (*hepatitis*). The uncertainty level changes from 0.05 to 0.3. It is evident from the figures that UAC outperforms

UCBA with respect to both number of the rules in finalSet and training time. UAC has the highest number of rules in the finalSet for the two datasets selected. We chose these datasets to be able to show the difference between the number of rules found by each classifier.

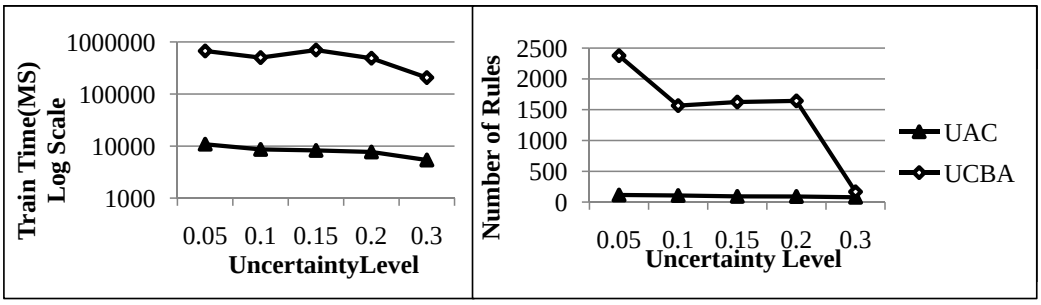
As uncertainty level increases, the training time and the number of the rules in finalSet for UAC usually decrease. This is because in our model, uncertainty decreases the meaningful patterns in the data, and consequently the number of the rules in rawSet (association rules mined by UApriori). Having less rules in rawSet has a direct effect on the time consumed by both rule extraction and rule filtering steps, As well as the number of the rules in rawSet. In Figure 3.1(c), UCBA has the same number of rules as UAC when $ulev$ is 0.3. However, based on Table 3.7, UAC has higher accuracy than UCBA in that case. Thus in this case we can claim the rules kept by UAC are better than UCBA.



(a) Results reported on hepatitis dataset with uncertain interval based numerical data and uncertain categorical attributes



(b) Results reported on hepatitis dataset with uncertain sampled based numerical data and uncertain categorical attributes



(c) Results reported on zoo dataset with uncertain categorical attributes

Figure 3.1: A comparison between UAC and UCBA in number of rules of finalSet and the training time elapsed.

Chapter 4

UAprioriAll: Mining Sequential Patterns from Uncertain Data

Sequential Pattern Mining or *SPM* is a well known problem and has been addressed by several studies [28, 43, 47, 5]. SPM in context of uncertain datasets is a recently raised problem with important applications. In this chapter, we propose a solution for the aforementioned problem by introducing a new algorithms: *UAprioriAll*

Sequential Pattern Mining or *SPM* is a well known and important problem in data mining and it has been addressed by many studies [28, 43, 47, 5]. However in spite of several applications, mining frequent sequences from uncertain datasets is still an open problem. In this paper, we propose a solution for the aforementioned problem by introducing a new algorithm called UAprioriAll.

The proposed algorithm uses *expected support* as the measure of frequentness of transactions and sequences. Expected support is a metric that measures the expected frequency of an itemset/sequence in uncertain datasets. It is memory efficient and very fast to compute.

In this chapter, we introduce and describe our proposed algorithm in Section 4.1 and discuss the experiments designed to evaluate the proposed algorithm in Section 4.2.

4.1 UAprioriAll

Mining sequential patterns from datasets with so called attribute level uncertainty has various applications. For example, consider a sequential database containing the

transactions for consecutive days. The items in the transactions may be a result of inaccurate measurements such as weather conditions which produces an uncertain dataset. UAprioriAll adopts two well known algorithms of UApriori and AprioriAll in its structure. UApriori has been described in the previous section. Therefore, we first give a brief description of AprioriAll, that is an SPM algorithm devised for "certain" datasets.

4.1.1 Data Modeling

The data model that we propose for a realistic capture of uncertainty in sequential datasets is the attribute level probabilistic dataset. In such datasets, each item exists within a transaction with a probability. Equation 1 shows the general form of our datasets. Each dataset (D) with size $|D|$ is a set of $|D|$ sequences (S_i), where each sequence of size $|S_i|$ contains $|S_i|$ transactions ($t_{i,j}$). Each transaction like $t_{i,j}$ is a set of pairs, where each pair includes an item and an existence probability. The pair $(it_{i,j,k}, pr_{i,j,k})$ implies that there is $1 - pr_{i,j,k}$ chance that item $it_{i,j,k}$ is not included in $t_{i,j}$. Sequence $< a : 0.4, b : 0.3 >$ is a simple example of a probabilistic sequence of size 2 with two transactions of $\{a : 0.4\}$ and $\{b : 0.3\}$ each of size 1.

$$\begin{array}{lcl} D & = & \{S_i : i = 1..|D|\}. \\ S_i & = & < t_{i,j} : j = 1..|S_i| > . \\ t_{i,j} & = & \{(it_{i,j,k}, pr_{i,j,k}) : k = 1..|t, j|)\}. \end{array} \quad (1)$$

This already covers many real world problems without setting unwanted restrictions and assumptions. The only assumption made here is the independence of the probabilities , which is a common simplifying assumption.

Uncertainty in the order may also happen within the sequential datasets. For example, as a result of the anonymization process, the dates of the medical documents are randomly changed to ensure the preservation of privacy. This causes the place of each transaction within each sequence to be not certainly known, which leads to multiple possible permutations of each sequence. With some domain based knowledge, we can assign a probability to each possible permutation. For instance, $< a : 0.4, b : 0.3 >$ is probable with $p = 0.6$ and $< b : 0.3, a : 0.4 >$ with $p = 0.4$.

We do not directly address the uncertainty in the order because of the very large

complexity of the solution, but UAprioriAll can also be applied to this problem with some compromises about the probability independence. If we assume that the probabilities are independent, we can write the above sequence as $< \{a : 0.24, b : 0.18\}, \{b : 0.12, a : 0.16\} >$ which is a sequence of size 2 consisting of two transactions of size 2. Note that this model generates some unwanted sequences of form $< a, a >$ as well. This does not affect the resulting frequent sequences because the sequences, within which two or more transactions are the same, do not effect the results of our algorithm. Of course, the independence assumption in the order uncertainty problem is not necessarily true, however, enables us to convert the complex problem to our simplified and tractable version of Equation 1 and solve it. However in this study we do not discuss further the uncertainty in the ordering.

4.1.2 AprioriAll

AprioriAll [5] is an SPM algorithm to mine frequent sequential patterns from "certain" datasets. It is efficient and has a high performance, which makes it suitable for large databases. AprioriAll consists of four major phases to find the frequent sequences:

- *Sort phase* in which transactions are sorted according to their time stamps. Sequential transactions are formed in this stage.
- *Litemset phase* in which frequent itemsets are found by applying Apriori;
- *Transformation phase* in which the original dataset is transformed to a new dataset by omitting infrequent itemsets and mapping the frequent ones to integer values for easy comparison;
- *Sequence phase* in which the candidate sequences are generated and the infrequent sequences are filtered out. Candidates set is initialized with the candidate sequences of size one which were found by *Litemset phase*.

4.1.3 UAprioriAll

In this section the novel algorithm UAprioriAll is explained. This algorithm is based on AprioriAll and has four phases: *a*) Sort; *b*) U-Litemset; *c*) U-Transformation; *d*) U-Sequence.

Sort: Sorting Based on Timestamps

The first step of UAprioriAll is to sort transactions according to their occurrence time. During this process the probabilities of the items are preserved.

U-Litemset: Mining Single Sequences

In this phase, the sequences of size 1 (containing only one transaction) are evaluated. Each single sequence (itemsets) is marked as a candidate, if its expected support is above the minimum threshold. When the dataset is not uncertain, support of an itemset is defined as the number of sequences of which at least one transaction contains that itemset. We know that in the probabilistic datasets D the expected support is computed by Equation 2 [3]. The value of $P(x \in S)$, that denotes the probability by which itemset x existing in the sequence S, is calculated by Equation 4. This equation computes the probability by which x exists in at least one of the transactions in S. Evaluating $p(x \in T)$, that denotes the probability by which itemset x existing in the transaction T, is performed by Equation 3.

$$E(s(x)) \quad = \quad \Sigma_{S \in D} \, P(x \in S) \tag{2}$$

$$P(x \in T) \quad = \quad \prod_{i \in x} P(i \in T) \tag{3}$$

$$P(x \in S) \quad = \quad 1 - \Pi_{T \in S} \, (1 - p(x \in T)) \tag{4}$$

We mine the *probabilistic frequent patterns* using a UApriori based technique. By the definition of the expected support from Equation 2, U-Litemset phase extracts itemsets that have higher expected support than the minimum support. These itemsets are put in a set called L_1. Next, each of the patterns in set L_1 is mapped to

a unique integer number and L_1 is transformed using this map. Set L_1 is the output of this phase. This phase was designed in collaboration with Samaneh Bayat, Parisa Naeimi and Mahdieh S. Mirian.

U-Transformation: Simplifying the Dataset

In this phase we transform the sequential dataset based on set L_1. The transformed dataset has two major differences with the original dataset. First, all the infrequent itemsets of the original dataset, that is the ones that are not contained in L_1, are removed from the transformed dataset. Second, the frequent itemsets are mapped into integer numbers. Algorithm 5 shows the pseudocode for this phase. Note that the pseudocodes are high level and do not involve the implementation details. For example, the list of the transactions that contain a frequent itemset is computed in U-Litemset phase and stored in the memory which can speed up the process here. This is not referenced in the provided pseudocode.

In a dataset which is not uncertain, we do not need to assign probabilities to the mapped frequent itemsets. However, the problem at hand deals with the uncertain itemsets and the probability associated with each itemset should be computed and attached. The probability, by which the frequent itemset x is included in the current transaction, is computed in line 9 of Algorithm 5 by Equation 4. At the end of U-Transformation Phase, we have a transformed dataset which contains sequences of transactions of uncertain *frequent itemsets* where each itemset is shown as an integer. This phase was designed in collaboration with Samaneh Bayat, Parisa Naeimi and Mahdieh S. Mirian.

UC-Sequence: Mining the Sequences

In this phase, we mine the frequent sequences from the transformed dataset (output of U-Transformation). The frequent sequences are computed by a UApriori-like scheme, that is a multi-level algorithm in which the candidate set for each level is formed based on the set of frequent sequences of the previous level. The initial set of frequent sequences is L_1 that contains the sequences of size 1, also called 1-sequences. Set L_1 is in fact the output of U-Litemset.

Algorithm 5 UAlrioriAll: U-Transformation Phase

1: D is the dataset.
2: $newD = \emptyset$ {#the transformed dataset}
3: L_1 is the output of U-Litemset
4: **for all** Sequence $seq \in D$ **do**
5: **for all** Transaction $t_1 \in seq$ **do**
6: $newseq = \emptyset$
7: **for all** Transaction $x \in L_1$ **do**
8: **if** $x \in t_1$ **then**
9: $prob = p(x \in t_1) = \prod_{i \in x} P(i \in t_1)$
10: Add $(x.id, x, prob)$ to $newseq$
11: **end if**
12: Add $newseq$ to $newD$
13: **end for**
14: **end for**
15: **end for**
16: return $newD$

Each level of this algorithm includes forming the candidate set, computing the expected support, and removing the infrequent sequences. The pseudocode of this phase is presented in Algorithm 6. Function gen-candidates (line 3) fills up the new candidate set (C_k) based on the frequent sequences of the previous level (L_{k-1}). This function evaluates $L_{k-1} \bowtie L_{k-1}$ for all two tuples that have $k-2$ items in common and then removes those that have infrequent subsets.

Algorithm 6 UAprioriAll: UC-Sequence phase

1: L_1 : output of U-Litemset.
2: **for** $k = 2$..maximum sequence length **do**
3: C_k = gen-candidates(L_{k-1});
4: **if** $C_k == \emptyset$ **then**
5: go to line 14
6: **end if**
7: **for all** candidate c from C_k **do**
8: Compute the expected support $ExpSup(c)$ {#by Equation 2}
9: **if** $Expsup(c) \geq$ minimum support **then**
10: add c to L_k
11: **end if**
12: **end for**
13: **end for**
14: return $\bigcup_k L_k$

Function $Expsup(c)$ calculates the expected support of a single candidate c. As

before, we define the expected support as the sum of the probabilities by which the candidate sequence is contained in the sequences of the dataset (Equation 2). Then we need to calculate the value of $P(x \in S)$ based on our definition of the support. The support of a candidate can be computed by counting the number of sequences containing it when there is no uncertainty involved. When probabilities are present, the process is more complex. The probability by which the first k items of candidate $c \in C_m$ ($m > k$) appears at least once within the first j items of the sequence $s \in D$ is denoted by $P_{j,k}(c, s)$ and computed by a recursive approach presented in Equation 5. The probability of c being a subsequence of s ($P(c \subseteq s)$) is then equal to $P_{|c|,|s|}(c, s,)$ where $|c|$ is the number of transaction is c and $|s|$ is the number of transactions is s. In Equation 5, $s[j]$ is the $j-th$ transaction in the sequence and $p(s[j])$ is the probability associated with $s[j]$ and $c[k]$ is the $k-th$ transaction of the candidate sequence. Value of $P_{j,1}(c \subseteq s)$ is computed based on Equation 4 and the case of $k > j$ results in 0 because a larger sequence cannot be contained within a smaller one. The recursive equation allows us to benefit from the dynamic programming scheme.

$$\begin{array}{lcl} Pc \subseteq s & = & P_{|c|,|s|}(c \subseteq s) \\ P_{j,k}(c \subseteq s) & = & P_{j-1,k-1}(c \subseteq s) * p(c[k] \subseteq s[j]) + \\ & & P_{j-1,k}(c \subseteq s) * (1 - p(c[k] \subseteq s[j])) \\ P_{j,k}(c \subseteq s) & = & 0, \text{if } k > j \\ P_{j,1}(c \subseteq s) & = & 1 - \Pi_{t_l \subseteq s}^{l=1..j} (1 - p(c[1] \subseteq t_l)) \end{array} \quad (5)$$

The recursive formula is achieved by dividing the problem into two mutually exclusive states. State a is when $s[j]$ contains $c[k]$ and state b is otherwise. The probability value is the addition of the probabilities of the two states. State a requires two events to happen, both $c[k] \subset s[j]$ and $s[1..j-1]$ (the $j-1$ first elements of s) should contain at least one appearance of $c[1..k-1]$ (the first $k-1$ elements of c). State b also requires two events, $c[k] \nsubseteq s[j]$ and c being a subset of $s[1..j-1]$. It is evident that states a and b are mutually exclusive.

To illustrate this method, assume that we have a candidate sequence $c =< x, y, z >$ and a dataset sequence $s =< x : 0.1, y : 0.5, x : 0.9, y : 0.8, t : 0.9, z : 0.5 >$. Table 4.1 shows the computed values associated with the dynamic programming method that employs Equation 5. Note that there are some *don't care* values

in this table that are denoted as 'DC'. The *don't care* values are not significant in the computation of the final value and cover around a half of the table. General set of *don't care* values for computing $P(c \subseteq s)$, denoted as $DC_{(c,s)}$, is formally stated in Equation 6. The total number of computations required can be assessed by Equation 7, based on the number of *don't care* values and zeros.

$$DC_{(c,s)} = \{P_{j,k}(c,s) : |c| - k > |s| - j\} \tag{6}$$

$$\begin{aligned} \text{Total Computations}[P(c \subseteq s)] &= |s|.|c| - \tfrac{|c|^2-|c|}{2} - \tfrac{|c|^2-|c|}{2} \\ &= |c|.(|s| - |c| + 1) \end{aligned} \tag{7}$$

	j	1	2	3	4	5	6
k	$s[1..j]$ / $c[1..k]$	$< x >$	$< x, y >$	$< x, y, x >$	$< x, y, x, y >$	$< x, y, x, y, t >$	$< x, y, x, y, t, z >$
1	$< x >$	0.1	0.1	0.91	0.91	DC	DC
2	$< x, y >$	0	0.05	0.05	0.738	0.738	DC
3	$< x, y, z >$	0	0	0	0	0	0.369

Table 4.1: Computation table for $p(c \subseteq s)$ where $c =< x, y, z >$ and $s =< x : 0.1, y : 0.5, x : 0.9, y : 0.8, t : 0.9, z : 0.5 >$

Mining Sequential Patterns from An Example Table by UAprioriAll

Below, we illustrate by an example how UAprioriAll proceeds to mine sequential patterns from an uncertain sequential dataset.

1. Sort: Table 4.2 is sorted into Table 4.3 based on the transaction times provided.

2. U-Litemset: Table 4.4 shows the output of U-Litemset phase for the dataset presented in Table 4.3, when the minimum support is $1/3$.

3. U-Transformation: According to L_1 presented in Table 4.4, Table 4.5 is the transformed form of Table 4.3.

4. UC-Sequence: Different level sets acquired by UC-Sequential on the transformed dataset shown in Table 4.5 are presented in Table 4.6. Level 1 is

Costumer ID	Transaction Time	Items Bought(item:probability)
S1	14 April 2011	a:0.7 , b:0.6 , d:0.3
S1	24 April 2011	d:0.9
S1	18 April 2011	b:0.8
S1	15 April 2011	c:0.6
S2	26 April 2011	b:0.9
S2	30 April 2011	b:0.8 , e:0.1
S2	25 April 2011	a:0.6 , c:0.2
S2	1 May 2011	d:1.0
S3	26 April 2011	d:0.5 , e:0.4, f:1.0
S3	18 April 2011	a:0.8 , b:0.8

Table 4.2: A sample probabilistic sequential dataset with timestamps

Sequence no.	Order (time)	Itemset (item:probability)
S1	1	a:0.7 , b:0.6 , d:0.3
	2	c:0.6
	3	b:0.8
	4	d:0.9
S2	1	a:0.6 , c:0.2
	2	b:0.9
	3	b:0.8 , e:0.1
	4	d:1.0
S3	1	a:0.8 , b:0.8
	2	d:0.5 , e:0.4, f:1.0

Table 4.3: Sorted form of Table 4.2

Frequent Single Sequence	Expected Support	Mapped To
a	2.1	1
b	2.7	2
d	2.43	3
f	1	4
a,b	1.06	5

Table 4.4: Output of U-Litemset: frequent single sequences mined from Table 4.3, also called L_1

Sequence no.	Order (time)	Mapped itemset:probability
S1	1	1:0.7 , 2:0.6 , 3:0.3, 5:0.42
	2	2:0.8
	3	3:0.9
S2	1	1:0.6
	2	2:0.9
	3	2:0.8
	4	3:1.0
S3	1	1:0.8, 2:0.8, 5:0.64
	2	3:0.5, 4:1.0

Table 4.5: Transformed form of Table 4.3

the output of U-Litemset phase and is shown in Table 4.4. The minimum expected support is set to $1/3$ and the expected supports are computed by Equations 2 and 5. After computing all patterns, we can apply the reverse mapping from Table 4.4 to get the actual patterns in the column named as "decoded patterns".

Level	Sequential Patterns	Expected Support	Decoded Patterns
L_2	2,3	2.21	b,d
	1,3	1.63	a,d
	1,2	1.15	a,b
L_3	1,2,3	1.09	a,b,d

Table 4.6: Frequent Sequential Patterns mined from 4.5

4.1.4 Algorithm Correctness

Relying on the mathematical meaningfulness of the expected support, the purpose of UAprioriAll is to find all sequences that have higher expected support than the predefined threshold. It is easily verifiable that our algorithm successfully fulfills its purpose. Line 9 of Algorithm 6 suffices to show that UAprioriAll only finds the sequences with higher expected support than the threshold (soundness). The following theorem proves the completeness of the algorithm by induction.

Theorem: For all sequences like $s =< i_1, .., i_n >$, where $ExpSup(s) > minSup$ and $minSup$ denotes the minimum support, $s \in L_n$ is true.

Proof: By induction.

- **Basis,** $n = 1$**:** This targets the completeness of U-Litemset phase. The correctness of U-Litemset is presumed, as UApriori algorithm is complete.

- **Hypothesis,** $n = k$**:** For all sequences like $s_k =< i_1, .., i_k >$, where $ExpSup(s) > minSup$, $s_k \in L_k$ is true.

- **Step,** $n = k + 1$**:** For all sequences like $s_{k+1}s =< i_1, .., i_{k+1} >$, where $ExpSup(s) > minSup$, $s_{k+1} \in L_{k+1}$ is true. Based on lines $7 - 12$ of Algorithm 6, we need to show $s_{k+1} \in C_{k+1}$ (defined in line 3) to prove $s_{k+1} \in L_{k+1}$. For $s_{k+1} \in C_{k+1}$ to be true, all sequences like s_k which are of size k and $s_k \subset s_{k+1}$ need to be in L_k so that function "gen-candidates" can

produce s_{k+1}. According to downward closure lemma for expected support, s_k is also frequent. The hypothesis is now applicable to s_k, meaning that $s_k \in L_k$. So $s_{k+1} \in L_{k+1}$.■

Based on the proof above, the algorithm is sound and complete. Also the algorithm terminates when a level set is empty. If the maximum length of the sequences is K_{max}, level $L_{K_{max}}$ is empty which results in the algorithm termination. Therefore, total correctness is proven for UAprioriAll.

The following observations are useful for analyzing the behaviour of UAprioriAll.

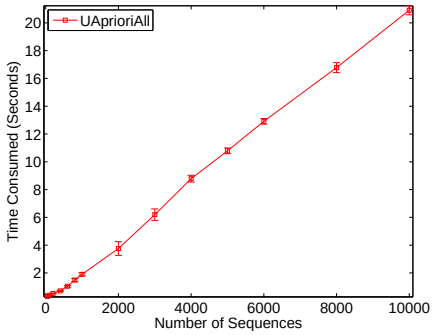
- When the dataset is certain, the results of UAprioriAll and AppriorAll are identical.
- When there are only two types of items in the dataset, the certain ones and ultimately improbable ones (with very low probabilities), the resulting set of frequent sequential patterns remains unchanged in the presence or absence of the improbable itemsets.
- Expected support of itemset a is higher than itemset b, if they occur with the same frequency in the sequences, but a is always more probable than b.

4.2 Experiments

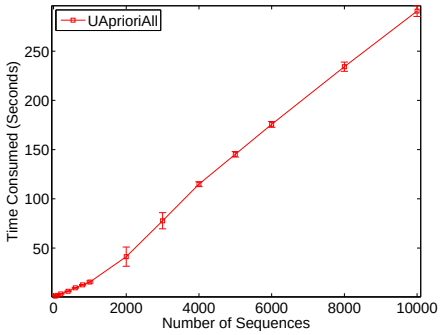
In this section, we present the empirical experiments showing the scalability of our algorithm in time consumption.

4.2.1 Datasets

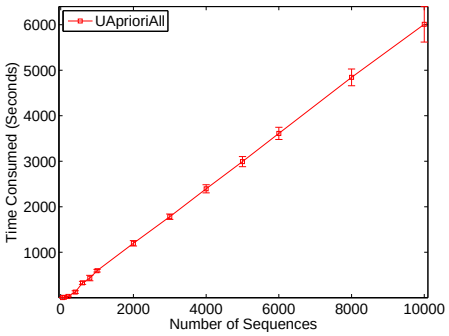
The medical data that we are investigating is not completely available as it is being constructed and moreover, for privacy reasons we would not be able to report the results of this dataset. As there are no uncertain sequential datasets available to the public and synthetic data is more useful in scalability measurements, similar to other studies on uncertain data (such as in [48]), we used synthetic datasets in our experiments. Each generated synthetic data is characterized by two parameters: the



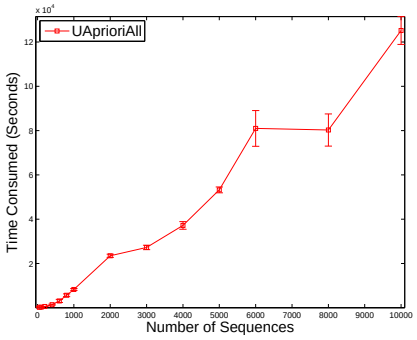
(a) Time consumption of *UAprioriAll* when minimum support = 0.5



(b) Time consumption of *UAprioriAll* when minimum support = 0.33



(c) Time consumption of *UAprioriAll* when minimum support = 0.25



(d) Time consumption of *UAprioriAll* when minimum support = 0.10

Figure 4.1: Time consumption of *UAprioriAll* with different support values

number of sequences (L) and the total number of items (I). Our goal is to investigate the effect of L on the time consumption, so we set I to 20 and L is varied from 50 to 10000. The number of transactions in each sequence is a monotonically distributed pseudo-random number from 1 to 20. We randomly choose the items within the transaction, where all items have equal chances. The existence probability attached to each item within the transaction is a randomly generated number between 0 to 1.

To increase the reliability of the results, for each value of L, 5 datasets were generated and the method was applied 5 times to each dataset. In this process, we used the *Scaling Method*. This method scales down a large dataset by randomly eliminating some transactions, to get a smaller dataset with lower number of sequences. Using the scaling method increases the accuracy of the results, as these values depend significantly on the candidate set size which itself depends on the trends and structure of the dataset as much as on the size. The scaling method fixes the trends and structure of the datasets to assess the effect of the size.

4.2.2 Experiments and Discussion

To measure the efficiency of our SPM algorithm, *UAprioriAll* was applied to the synthetic data explained in Section 4.2.1. The experiments were carried out on a machine with 2.66 GHz clock speed and 8 GB of RAM. In the implementation of the algorithm we adopted a free online Java implementation of Apriori [13]. Samaneh Bayat, Parisa Naeimi and Mahdieh S. Mirian have contributed to the implementation of UAprioriAll.

The experiments include the time consumption of the algorithm. The minimum support is varied from 0.1 to 0.5 for each dataset to assess the impact of the minimum support on the performance of UAprioriAll. The minimum support is important because decreasing it may cause a dramatic drop in performance. Also it is possible that the behaviour of the algorithm and the scalability vary significantly between the cases with small and large minimum supports. We have used 4 different values of support including: $0.1, 0.25, 0.33, 0.5$ to investigate the consistency in the behaviour of UAprioriAll. Setting minimum support in real world applications

depends greatly on the domain.

For each value of minimum support and dataset size, 5 experiments were carried out on each of the 5 different datasets. The presented results are the average on the 25 experiments. Figures 4.1(a), 4.1(b), 4.1(c), and 4.1(d) show the time scalability of UAprioriAll with different values of minimum support. From the provided figures, it is evident that the time consumption of UAprioriAll grows linearly based on the number of sequences. The trend remains unchanged with different values of the minimum support which shows that the scalability of UAprioriAll does not depend on the minimum support value. The results for Figure 4.1(d) are confirmed and we have no explanation for the plateau when the transactions are 6000. This might be due to the shared resources and changes in the resource usage behavior of other processes running on our server.

Chapter 5

Conclusions

In this study, we present two new algorithms for managing uncertainty in datasets. Uncertainty is common in many applications in today's world. One example is extracting information from unstructured health data. We address the problems of classification and sequential pattern mining and devise algorithms to solve these problems in uncertain datasets. Experiment results show that our algorithms effectively handle uncertainty.

In Chapter 2, a review over the related research shows the impact of the uncertainty management. Research related to modeling uncertain data, frequent pattern mining, classification, sequential pattern mining, etc. is reviewed in this chapter.

In Chapter 3, we state the problem of classifying uncertain datasets and introduce algorithm UAC in section 3.2 to solve this problem. Two experiments in section 3.3 show that UAC is more accurate than the state of the art rule based classifiers.

The first experiment compares UAC to other rule based classifiers including uHarmony, DTU and uRule. All of the rule based classifiers mentioned in this experiment are designed for the same uncertainty model which is less general than the model we use in this study. To be able to meaningfully compare the accuracies, we limit the space of the uncertainty model to their model using their method of adding uncertainty. We perform the experiment on the same UCI datasets as uHarmony to show that we do not pick special datasets where we reach better accuracies. Section 3.3.1, summarizes the experiments which shows UAC is more accurate than other rule based classifiers in most cases and on the average.

The second experiment aims to measure the accuracy under our more general uncertainty model. Section 3.3.1, compares UAC against another CBA based probabilistic associative classifier, UCBA. Since UCBA has no limitation over the model it can be used as the baseline for this experiment. The experiment shows that in most cases and on the average, UAC has better accuracy than UCBA. Also as mentioned in section 3.2.2, the number of rules is an important factor to the rule based classifiers. To see the effect of our uncertainty management techniques on the number of rules as well as the time needed for building the model, we perform a comparison between UAC and UCBA. This comparison in meaningful since both algorithms are CBA based. The experiment shows that UAC is faster and leads to far less classification rules than UCBA.

In Chapter 4, we state the problem of mining sequential patterns from uncertain datasets. Section 4.1, presents UAprioriAll, an Apriori based algorithm that effectively mines sequential patterns from uncertain datasets. Section 4.1.4 discusses the completeness and soundness of this algorithm and section 4.2 gives experiment results that show UAprioriAll scales linearly by increasing the number of transactions.

5.1 Summary of Contributions

- Introducing a new more general model for capturing uncertainty in classification problems.
- Presenting UAC, the probabilistic data associative classifier that effectively handles uncertainty. UAC is more accurate than the state of the art rule based algorithms under their model, and is more accurate than UCBA, another CBA based algorithm, under the new more general model. Additionally, the number of rules and the running time is far less than UCBA in many cases.
- Introducing the concept of applicability in probabilistic data classification, which is a new and more effective way of addressing the coverage problem as stated in section 3.2.2.

- Presenting UAprioriAll, the first sequential pattern mining algorithm for uncertain datasets that handles a general form of attribute level uncertainty.

5.2 Future Work

Uncertainty management is finding more applications every day in today's world. Based on the application, uncertainty modeling may vary in the studies. More general uncertainty model may be necessary for some domains. For example, adding uncertainty to the labels is one unaddressed problem in this area. One possible solution for this is to use an alternative accuracy measuring technique. For example, if a record like $a0.1, b0.9, c_1 0.8$, where c_1 is the class label, where classified as c_1, accuracy is added by 0.8. Otherwise, we add 0.8 to error.

Devising a data structure for the rules, as the one used in some traditional algorithms including CMAR, and applying a separation technique, as used by ARC, are other open problems that can be addressed in future research on uncertain data classification.

An open problem in uncertain data sequential pattern mining is order uncertainty. In health related applications, the process of annonymization may lead to records with uncertain dates. There are various ways to model this uncertainty. One way is to have intervals as the dates. For example, $P_1 : M_1 p_1 [d_1 - d_2]$ means that patient P_1 took medicine M_1 with a probability of p_1 between dates d_1 to d_2. Mining sequential patterns from this type of datasets is an open problem.

Another way to capture uncertainty in sequential datasets with probabilistic ordering is to have a probability assigned to orders. For example, we may have information like: medicine M_1 was either the first medicine patient P_1 took (with p_1 probability) or the second one (with p_2 probability) or the fourth one (with p_4 probability). Some restrictions are applied to the probability values including the fact that the sum of the probabilities is less than 1. The difficulty with this problem is to design a technique that with a reasonable running time, measures the probability that a sequence is a subsequence of another one. In our own problem where order is not uncertain, we solve this problem in section 4.1.3 using dynamic programming.

Although here, since the probabilities are not independent, one cannot use the same method.

Bibliography

[1] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *SIGMOD Record*, 16:34–48, December 1987.

[2] 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, June 2009.

[3] C. C. Aggarwal and P.S. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(5):609–623, May 2009.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499, Santiago, Chile, September 1994.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, pages 3–14, Taipei, Taiwan, Mar. 1995.

[6] M. L. Antonie, O. R. Zaiane, and R. C. Holte. Learning to use a learned model: A two-stage approach to classification. In *Proceedings of the 6th IEEE conference on data mining (ICDM)*, pages 33–42, Hong Kong, December 2006.

[7] T. Bernecker, H. p. Kriegel, M. Renz, F. Verhein, and A. Zuefle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, June 2009.

[8] J. Bi and T. Zhang. Support vector classification with input data uncertainty. In *Proceedings of the 18th Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 161–168, Vancouver, Canada, December 2004.

[9] M. Chau, R. Cheng, and B. Kao. Uncertain data mining: A new research direction. In *In Proc. Workshop on the Sciences of the Artificial*, Hualien, Taiwan, December 2005.

[10] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22:550–564, April 2010.

[11] W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, USA, July 1995.

[12] J. Demsar. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research 6 (JMLR)*, 7:1–30, 2006.

[13] P. Fournier-Viger. Algorithms/frequent itemset mining algorithms. url: http://www.philippe-fournier-viger.com/spmf/index.php, June 2010.

[14] C. Gao and J. Wang. Direct mining of discriminative patterns for classifying uncertain data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, July 2010.

[15] C. Gao and J. Wang. uharmony for kdd'10 paper. http://dbgroup.cs.tsinghua.edu.cn/chuancong/uharmony/, June 2010.

[16] M. N. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, VLDB '99, pages 223–234, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[17] J. Ge, Y. Xia, and C. Nadungodage. Unn: A neural network for uncertain data classification. In *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 449–460, Hyderabad, India, June 2010.

[18] T. Ge, S. Zdonik, and S. Madden. Top-k queries on uncertain data: on score distribution and typical answers. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 375–388, Providence, Rhode Island, USA, July 2009.

[19] T. Green and V. Tannen. Models for incomplete and probabilistic information. *Data Engineering Bullitan*, 29(1):17–24, 2006.

[20] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Record*, 29:1–12, May 2000.

[21] B. Jiang and J. Pei. Outlier detection on uncertain data: Objects, instances, and inference. In *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, Hannover,Germany, April 2011.

[22] B. Kanagal and A. Deshpande. Indexing correlated probabilistic databases. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 455–468, Providence, Rhode Island, USA, July 2009.

[23] M.-Y. Kim, Q. Dou, O.R. Zaiane, and R. Goebel. Unsupervised mapping of sentences to biomedical concepts based on integrated information retrieval model and clustering. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 322–329, Niagara Falls, New York, August 2010.

[24] C. K-S Leung, C. L. Carmichael, and B. Hao. Efficient mining of frequent patterns from uncertain data. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 489–494, Washington, DC, USA, October 2007.

[25] 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 (ICDM)*, pages 369–376, Phuket, Thailand, December 2001.

[26] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. In *Proceedings of Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD)*, pages 80–86, 1998.

[27] S. Patel O. R. Zaiane M. HooshSadat, H. Samuel. Fastest association rule mining algorithm predictor - farm-ap. In *Fourth International C* Conference on Computer Science and Software Engineering.*, pages 43–50, Montreal, QC, Canada, 2011.

[28] J. Pei, J. Han, B. Mortazavi Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. Prefixspan mining sequential patterns efficiently by prefix projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 215–226, Heidelberg, Germany, April 2001.

[29] Jian Pei, Ming Hua, Yufei Tao, and Xuemin Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1357–1364, Vancouver, Canada, June 2008.

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

[31] B. Qin, Y. Xia, and F. Li. A bayesian classifier for uncertain data. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC)*, pages 1010–1014, Sierre, Switzerland, March 2010.

[32] B. Qin, Y. Xia, S. Prabhakar, and Y. Tu. A rule-based classification algorithm for uncertain data. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE)*, Shanghai, China, March 2009.

[33] X. Qin, Y. Zhang, X. Li, and Y. Wang. Associative classifier for uncertain data. In *Proceedings of the 11th international conference on Web-age information management (WAIM)*, pages 692–703, Jiuzhaigou, China, July 2010.

[34] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[35] R. Rak, L. Kurgan, and M. Reformat. A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. *Data and Knowledge Engineering*, 64:171–197, January 2008.

[36] R. Rak, W. Stach, O. R. Zaane, and M. L. Antonie. Considering re-occurring features in associative classifiers. In *Proceedings of the Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD)*, pages 240–248, 2005.

[37] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. Technical Report 2005-3, Stanford InfoLab, 2005.

[38] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 596–605, Istanbul, Turkey, April 2007.

[39] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th international Conference on Extending Database Technology: Advances in Database Technology*, pages 3–17, London, March 1996.

[40] J. Friedman T. Hastie, R. Tibshirani. *The elements of statistical learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2 edition, 2008.

[41] J. Wang and G. Karypis. On mining instance-centric classification rules. *IEEE Transactions on Knowledge and Data Engineering*, 18:1497–1511, November 2006.

[42] J. Yang and M. Hu. Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology*, March 2006.

[43] J. Yang, T. J. Watson, W. Wang, P. S. Yu, and J. Han. Mining long sequential patterns in a noisy environment. In *Proceedings of the ACM SIGMOD international conference on Management of data*, Madison, Wisconsin, USA, June 2002.

[44] X. Yin and Han J. Cpar: Classification based on predictive association rule. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 369–376, San Fransisco, CA, USA, May 2003.

[45] Osmar R. Zaiane and Maria-Luiza Antonie. Classifying text documents by associating terms with text categories. *Proceedings of the 13th Australasian database conference*.

[46] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12:372–390, May 2000.

[47] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Journal of Machine Learning Research*, 42(1-2):31–60, 2001.

[48] Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 819–832, Vancouver, Canada, June 2008.

[49] W. Zhang, X. Lin, Y. Zhang, J. Pei, and W. Wang. Threshold-based probabilistic top-k dominating queries. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 19:283–305, April 2010.

[50] X. Zhou, H. Han, I. Chankai, A. Prestrud, and A. Brooks. Approaches to text mining for clinical medical records. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 235–239, Dijon, France, April 2006.