2010-01-13

DS-ARM: An Association Rule Based Predictor that Can Learn from Imperfect Data

Kasun Jayamal Sooriyaarachchi Wickramaratna
University of Miami, k.wickramaratna@umiami.edu

UNIVERSITY OF MIAMI

DS-ARM: AN ASSOCIATION RULE BASED PREDICTOR THAT CAN LEARN FROM IMPERFECT DATA

By

Kasun J. S. Wickramaratna

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2010

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

DS-ARM: AN ASSOCIATION RULE BASED PREDICTOR THAT CAN LEARN FROM IMPERFECT DATA

Kasun J. S. Wickramaratna

Approved:

Kubat Miroslav, Ph.D. Associate Professor of Electrical and Computer Engineering	Terri A. Scandura, Ph.D. Dean of the Graduate School
Kamal Premaratne, Ph.D. Professor of Electrical and Computer Engineering	Akmal Younis, Ph.D. Associate Professor of Electrical and Computer Engineering
Nigel John, Ph.D. Lecturer of Electrical and Computer Engineering	Dushyantha T. Jayaweera , M.D., M.R.C.O.G. (UK), F.A.C.P. Professor of Clinical Medicine

S. WICKRAMARATNA, KASUN J. (Ph.D., Electrical and Computer Engineering)

DS-ARM: An Association Rule Based Predictor that Can Learn from Imperfect Data (May 2010)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Kubat Miroslav.
No. of pages in text. (118)

Over the past decades, many industries have heavily spent on computerizing their work environments with the intention to simplify and expedite access to information and its processing. Typical of real-world data are various types of imperfections, uncertainties, ambiguities, that have complicated attempts at automated knowledge discovery. Indeed, it soon became obvious that adequate methods to deal with these problems were critically needed. Simple methods such as "interpolating" or just ignoring data imperfections being found often to lead to inferences of dubious practical value, the search for appropriate modification of knowledge-induction techniques began. Sometimes, rather non-standard approaches turned out to be necessary. For instance, the probabilistic approaches by earlier works are not sufficiently capable of handling the wider range of data imperfections that appear in many new applications (e.g., medical data). Dempster-Shafer theory provides a much stronger framework, and this is why it has been chosen as the fundamental paradigm exploited in this dissertation.

The task of association rule mining is to detect frequently co-occurring groups of items in transactional databases. The majority of the papers in this field concentrate on how to expedite the search. Less attention has been devoted to how to employ the identified frequent itemsets for prediction purposes; worse still, methods to tailor association-mining techniques so that they can handle data imperfections are virtually nonexistent.

This dissertation proposes a technique referred to by the acronym DS-ARM (Dempster-Shafer based Association Rule Mining) where the DS-theoretic framework is used to enhance a more traditional association-mining mechanism. Of particular interest is here a method to employ the knowledge of partial contents of a "shopping cart" for the prediction of what else the customer is likely to add to it. This formalized problem has many applications in the analysis of medical databases.

A recently-proposed data structure, an itemset tree (IT-tree), is used to extract association rules in a computationally efficient manner, thus addressing the scalability problem that has disqualified more traditional techniques from real-world applications. The proposed algorithm is based on the Dempster-Shafer theory of evidence combination. Extensive experiments explore the algorithm's behavior; some of them use synthetically generated data, others relied on data obtained from a machine-learning repository, yet others use a movie ratings dataset or a HIV/AIDS patient dataset.

To my wife & parents

Acknowledgements

I extend my sincere gratitude and appreciation to my dissertation advisor and chairman of the committee, Professor Miroslav Kubat, for his guidance, support, suggestions and encouragement throughout the period this research was being conducted. I am also thankful to Professors Kamal Premaratne, Akmal Younis, Nigel John of the Department of Electrical and Computer Engineering, and Professor Dushyantha T. Jayaweera of the Department of Medicine, for accepting the appointment to the dissertation committee and for their suggestions and support.

The financial assistance through U.S. National Science Foundation (NSF) Grants IIS-0513702 is gratefully acknowledged. The financial assistance from the Department of Electrical and Computer Engineering, and Professor Peter Minnett, UM Rosenstiel Schools Division of Meteorology and Physical Oceanography is also acknowledged.

This dissertation would not have been possible without the help of many people in so many ways. I would like to thank my many friends and colleagues at the University of Miami with whom I have had the pleasure of working over the years. My special thanks goes to Rohitha, Chamara, Shaminda and Indika for always being a helpful and encouraging backstage presence. I would like to extend my utmost gratitude to my parents for providing moral and material support during the long years of my education, and also for their faith in me and allowing me to be as ambitious as I wanted. Finally, and most importantly, I would like to thank my wife Bhagya for her support, encouragement, quiet patience and unwavering love, which made this work possible. Over the course of this four-year journey that has now culminated, she has been my enduring source of strength.

Kasun J. S. Wickramaratna

University of Miami

May 2010

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Advances in computational power and storage capabilities have led to computerization of working environments, resulting in huge collections of data. These data collections are analyzed with the intention to discover or induce valuable new information or knowledge. From the many techniques developed to this end, the work reported in this introduction focusses on the detection of association rules from frequently co-occurring groups of items in transactional databases.

The knowledge of itemsets frequently co-occurring in a transactional database can be exploited for prediction. For instance, if `bread`, `butter`, and `milk` often appear in the same transactions, the presence of `butter` and `milk` in a shopping cart suggests the customer may also buy `bread`. More generally, knowing which items a shopping cart contains, we want to predict other items that the customer is likely to add before proceeding to the checkout counter.

Although this motivation behind association mining was mentioned as early as in the pioneering paper by Agrawal et al [AIS93], the prediction task has not been investigated as deeply as it could. This is partly due to the extremely high computational costs of itemset discovery in large databases. The scientific community simply felt that the attempts to expedite this search have much more urgency than the ways to exploit its results (see the survey in [KHR$^+$03b]). Other authors found it

important to investigate such special aspects of association mining as the discovery of time-varying patterns [GGR99, GGR00, RK05] or the identification of localized patterns [RH00, APY02].

This does not mean that no work has been done on the classification front at all. One of the earliest attempts to convert frequent itemsets to rules was reported in [BA99]. And some scientists looked into what would happen if one of the items is treated as a binary class whose value (absence = 0; presence = 1) is to be predicted. For instance, they asked: based on the current contents of the customer's shopping cart, can we predict whether or not the customer will buy `bread`? If yes, with what reliability? The results of early attempts were encouraging to the degree where some scientists even observed that the classification performance of association-mining systems may compare favorably with the more traditional machine learning techniques [ZSP+04a, LHM98, LHP01, LSW97].

The work reported here wanted to go one step further, assuming that *any* item in the shopping cart (not just `bread`) can be the treated as the "class label." The question is: given an incomplete list of the items in the shopping cart, can we guess the remaining items? An example will clarify the point. Suppose the shopping cart of a customer at the checkout counter contains `bread, butter, milk, cheese` and `pudding`. Can someone who met the same customer ten minutes ago, while the cart contained only `bread, butter` and `milk`, predict that the person will add also `cheese` and `pudding`? Let us remind the reader that—implicitly or explicitly—this task stood at the cradle of this field in the 1990s.

Although the basic association mining task is for historical reasons cast in the department-store paradigm (and the vast majority of papers rely on synthetic data with controllable parameters), the problem is typical of many real-world domains.

Thus in the application domain discussed in [NRC01], each "shopping cart" contained a set of hyperlinks that pointed to a given web page [NRC01]; in a medical domain, the shopping cart may contain a patient's symptoms and diagnoses; and in a financial domain it may contain companies held in the same portfolio. One study even suggested a framework for using frequent itemsets in information retrieval [BSHR01].

In all these domains, the prediction of currently missing items can be of great importance. For instance, medical doctors know that a patient's symptoms are rarely due to just a single cause; two or more diseases usually conspire. Having identified one of them, the physician tends to focus on how to treat this single disorder, ignoring the others that he or she is not aware of. This unintentional negligence can have unfortunate consequences, and yet, the situation is in a sense inevitable: the number of laboratory tests that a patient is to undergo is limited by many practical (e.g., economical) factors. Under these circumstances, a decision support system advising the physician of what other diseases usually accompany the one already diagnosed can help the physician decide which other tests should be prescribed.

In practice, however, this is not so simple. Knowing the presence or absence of a disease is not enough. What matters is also the severity as quantified, say, by a value from $\Theta = \{Critical, Medium, Normal\}$. When assigning the value, a physician relies on his/her experience and/or the experience of colleagues. Such ratings are inevitably ambiguous and not easily expressed in terms of probabilities—for instance, it would be mistaken to assume that the statement, "the symptom is *Critical* with a 70% confidence" implies a 30% confidence in the complement of *Critical*. On top of that, different experts' opinion on the same matter could be contrastingly different. The lack of mechanisms to accommodate such subjectivity often necessitates various

unwarranted "interpolations." Their inadequacy motivates our research: knowing some (ambiguous) ratings a user has given, we want to predict his or her other ratings.

The scope of applications of such a framework is broader than it seems. Suppose we have a database of customer ratings of a line of products. If a group of users tend to rate products $\{p_1, p_2, p_3\}$ in a similar way, $\{\langle p_1, \mathfrak{r}_1\rangle, \langle p_2, \mathfrak{r}_2\rangle, \langle p_3, \mathfrak{r}_3\rangle\}$, then a new user's ratings $\{\langle p_1, \mathfrak{r}_1\rangle, \langle p_2, \mathfrak{r}_2\rangle\}$ lead us to expect that this user, too, will rate p_3 as $\mathfrak{r}_3$. By considering a $\langle product, rating\rangle$ pair as an item, we can convert the problem to the department-store paradigm and ultimately predict the user rating.

1.1 Motivation

The idea of knowledge discovery from health-care databases is not novel to the data mining community. However, it has turned out to be much more complicated than originally thought, mainly due to data imperfections, to typical of medical databases. The author of this dissertation believes that the knowledge discovery from data sources of this kind is not feasible without adequate methods to accommodate and/or account for such data imperfections, and the need for such algorithms and techniques was actually one of the original motivations for the research reported here. Subjective information plays such a critical role in medical domains that simplistic methods such as "interpolating" or simply ignoring data imperfections often lead to inferences of dubious practical value. In this sense, it is crucial to modify existing knowledge induction techniques by effective strategies so as to reflect data imperfections and to better quantify the uncertainty implicit in the induced knowledge.

Specific Problem

The World Health Organization (WHO) estimates that 40 million people worldwide have been infected with the human immunodeficiency virus (HIV), with millions of new infections being added to this count each year. Treatment of HIV with highly active antiretroviral therapy (HAART) can effectively control HIV infection; however, non-adherence to the medication significantly limits the success outside clinical research settings. Once the effectiveness of the prescribed HAART regime is diminished, a new regime has to be prescribed after a careful study of the patient's HARRT regime response history and other related factors. This is one place where the physician's own experience really matters.

HIV damages the immune system, and thus makes the patients vulnerable to opportunistic infections and tumors. Opportunistic diseases usually tend to conspire with one another. Having identified one disease, the physician has to anticipate other diseases yet to come. In doing so, the physician relies on his or her own experience and on published data from controlled clinical trials. Hence, the decision making would be greatly enhanced by tools that extract information from databases and provide information to resolve some of the involved uncertainties.

Unfortunately, existing databases are marred with imperfections ranging from missing information to data entry errors and subjective evaluations. In machine learning, data imperfections have received inadequate attention, which is surprising in view of the abundance of mathematical frameworks developed by the decades of research in the field of uncertainty processing.

In the proposed work, we try to address the aforementioned issues. In particular, we explore the use of frequently co-occurring patterns, for prediction purposes. We emphasize association mining in domains with ambiguities and uncertainties.

1.2 Related Work

Most of the earlier association-mining systems were proposed with the focus of building a *classifier*. Given a set of data instances together with their corresponding class-labels, the task was to induce a "classifier." The given data set is called a *training set* and a *data instance* is identified by a set of attribute value pairs. The purpose of the *classifier* is to predict the class label (i.e., value of the *class-attribute*) of future data instances of which the class label is unknown. The approach used to build the classifier is often dubbed as *classification rule mining*. Classification rule mining aims to discover a small set of rules in the database that forms an accurate classifier. Unlike in association rule mining where the target of discovery is not predetermined, classification rule mining is done with one and only one predetermined target (the class label).

We believe that it is worth spending some time here to emphasize the differences in the "classification" task and the task at our hands. Let us formulate the notions for our discussion. We call each $\langle attribute, value \rangle$ pair an *item*. Each data instance is called a *transaction*. Thus, a *transaction* is identified by a set of *items*. Given a **partially observed** transaction, our task is to *predict* the rest of the items that might appear in the transaction. One may perceive this as a shopping cart completion task. Unlike in the above classification task, we cannot identify an explicit single class-label. In fact, any attribute (one or many) could take the place of class attribute depending on the given testing instance. In fact, the previous classification task can be seen as a special (and more simpler) case of this general task.

In principle, at least, we could adopt any of the classification methodologies to our paradigm; however, the problem is that they were designed primarily for the classification task, and not for the shopping cart completion task described earlier.

Specifically, the number of times such classifiers have to be invoked to address the issue at hand would be equal to the number of all distinct items in the database (i.e., n) minus the number of those already present in the shopping cart. For instance, we can use the training data set to build as many classifiers as the number of attributes, each time considering a different attribute as the class-attribute. Once a partially observed transaction is presented, we can select a subset of classifiers, such that each selected classifier predicts the value of one unobserved attribute. The subset of classifiers, thus, collectively predicts all the unobserved attribute values (or missing items). It should be noted that this is viable only if the classifiers are capable of accommodating total 'missingness' of attribute values. Obviously, this rather naive approach is feasible for situations where the number of attributes is limited to few (usually one or two) dozens. Classifiers proposed in previous studies thus lack the predictive abilities we look for in our paradigm. We would rather look for an efficient *predictor* which can predict all the items that would be added to the transaction. This is why we have sought to develop an efficient predictor that would predict all 'missing' items in the course of a single procedure.

With this being said, we shall review the earlier researches on building association-rule based classifiers, as the challenges faced in our paradigm is just an extension of the challenges faced by those researchers.

1.2.1 Association-Rule Based Classifiers

Association rules explore highly confident associations among multiple variables. Some researchers [LHP01] have suggested that this may overcome some constraints induced by a decision-tree induction methods which examine one attribute at a time. Effective classifiers have been built by careful selection of association rules [LHM98,

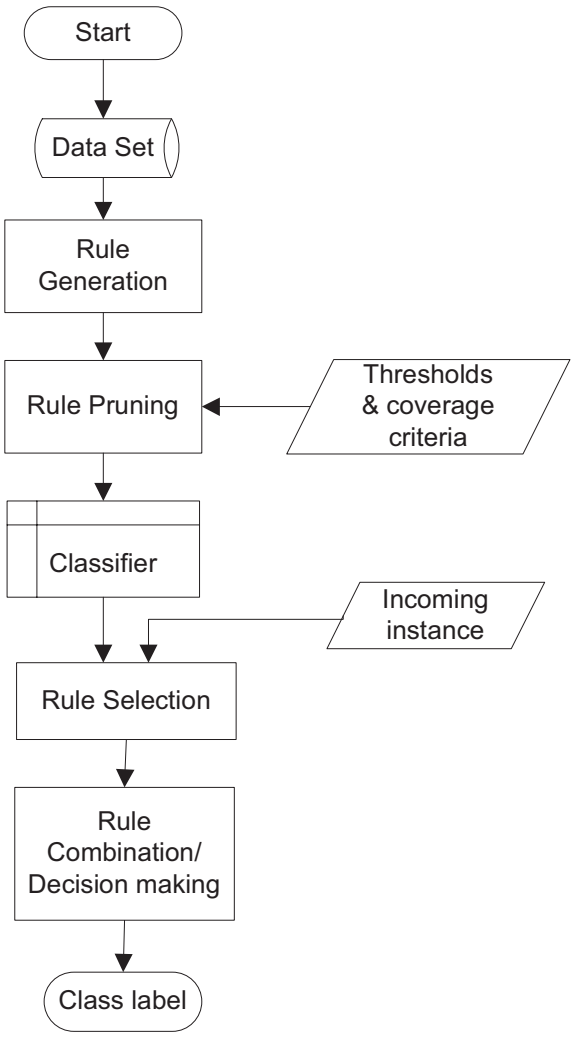


Figure 1.1: Building an association-rule based classifier

LHP01, HL05, ZSP+04a]. Extensive performance studies in above-cited papers have shown that association rule based classification may have better accuracy in general.

Building an association-rule based classifier usually entails four stepped procedure. 1.) Rule Generation, 2.) Rule Pruning, 3.) Rule Selection, and 4.) Reaching the Decision. Basic flowchart of the procedure is given in the Figure 1.1.

1. Rule Generation

The focus of rule generation is on a special subset of association rules whose right-hand-side is restricted to the classification class attribute. This might be challenging mainly due to the requirement of lower minimum support thresholds in order to extract rules from the rare class and the huge number of associations that result from this setting. The famous apriori algorithm [AIS93] could fail under these circumstances due to combinatorial explosion. Many researchers, thus, have adapted apriori algorithm to develop "apriori-like" algorithms [LHM98, ZSP+04a, HPS07] whereas some others have developed different methodologies such as FP-tree [HPY00].

Apriori-like methods are based on Apriori heuristic: *if a k length pattern is not frequent in a database, then any of its super-patterns of $(k+1)$ length can never be frequent.* The idea is to iteratively generate candidate frequent item sets of higher length thus reducing the size of candidate set. Apriori-like algorithms, however, are also known to suffer form performance deterioration when the "size" of the mining problem is large. The mining problem "size" could be determined by four parameters: number of transactions, number of distinct items, average transaction length and the minimum support threshold. For example, the problem size increases when the number of distinct *frequent items* in the dataset is increased (i.e., when a lower minimum support threshold is used). This is mainly due to the costs associated with handling a large number of candidate item sets generated under the above conditions that result in combinatorial explosion and repeated parses across the database.

Raising the minimum support threshold to reduce the computational costs could result in losing very important rules from the rare class. Studies suggest many "tricks" to limit the number of rules, yet, extracting rules from the rare class. In one such trick, a data set is partitioned according to the class label so that each partition contains

data instances from only one class. Association rule mining is then employed on each partition separately to extract rules. This can avoid the domination of a majority class in the generated rule set and ensure extraction of rules from a rare class. This approach is used for rule generation in [ZSP+04a].

The use of *multiple minimum support thresholds*, instead of a single minimum support threshold, is suggested in [LMW00]. The idea is to assign each class, c_i, a different minimum support threshold. The user only gives a total minimum support value, denoted by t_minsup, which is distributed to each class by:

$$minsup_i = t\_minsup \times sup\ (c_i)$$

The formula gives frequent classes higher minimum supports and infrequent classes lower minimum supports. This ensures that it will generate sufficient rules for infrequent classes and also it will not produce too many over-fitting rules for frequent classes.

We propose an alternative selective rule generation approach to overcome these difficulties.

2. Rule Pruning

As discussed above, many studies have emphasized the requirement of using a lower support threshold to include cases from the rare class. This requirement has led to the generation of huge numbers of frequent itemsets and hence even larger number of association rules[AS94]. Thus, to achieve high accuracy, a classifier may have to 'handle' a large number of rules, which includes storing the generated rules, retrieving the matching rules and sorting the retrieved rules. It is challenging to store, and effectively identify most effective rule(s) to classify a new instance. Thus, the generated rule set is often pruned using some criteria, aiming to retain a smaller set of rules that is still capable of forming an accurate classifier.

Studies have proposed several approaches to prune the rule set. We classify them under three categories— 1) threshold based pruning, 2) coverage Based Pruning, and 3) coverage with redundancy.

• Threshold Based Pruning

Researchers have come-up with many metrics to measure the "quality" (or "goodness" or "interestingness") of the rules. Those rules that do not satisfy the "quality" requirements are then pruned out. Only those "high-quality" rules are retained to form the classifier.

Most widely used quality metrics are *support* and *confidence* [AIS93]. The support of an item set A is equal to the fraction of the records in the data set that contains A. We shall denote this by $sup(A)$. Support of a rule $A \Rightarrow C$ is equal to $sup(A \cup C)$. The *confidence* of a rule is usually defined with the help of supports (relative frequencies) of the antecedent and consequent. More specifically, the confidence in $A \Rightarrow C$ is defined as the percentage of transactions that contain C among those transactions that contain A:

$$confidence = \frac{sup(A \cup C)}{sup(A)}. \tag{1.1}$$

The confidence, thus, can be considered as a measure of the "correctness" of a rule as well. Only those rules that have support and confidence values above certain thresholds are kept to form the classifier. These thresholds are often called $minsup$ and $minconf$ respectively [AIS93].

Among many other metrics used to measure the "quality" of a rule are gain [FMMT96], variance and chi-squared value [Mor98], entropy gain [MFM+98], gini index [MFM+98], and Laplace accuracy [HL05]. It is not our interest to discuss all these approaches here.

• **Coverage Based Pruning**

Coverage is used to further prune out the rule set. In this method the discovered rule set is sorted in accordance with their corresponding "priority". The best rule is then taken out from this collection. The training records "covered" by this rule are also taken out from the training set. Then the second best rule is selected from the rule collection, and data instances covered by this rule are removed from the training data set. This procedure is carried out until no data instances are left in the training data set. The selected rule set is then used to form the classifier and rest of the rules are pruned out.

A classical example of this approach can be found in [LHM98], CBA classifier. *Priority* of the rules is assessed in the course of a three-set process. Suppose we want to compare two rules, r_i and r_j. We will say that r_i *precedes* r_j, and write $r_i > r_j$, if (a) the confidence of r_i is greater than the confidence of r_j; or (b) their confidences are the same, but the support of r_i is greater than the support of r_j; or (c) both the confidence and support have the same values for both rules, but r_i has been generated earlier than r_j. The rule set is then pruned to select the minimal set of high priority rules that covers the training set.

The set of generated rules R is then sorted according to the priority. Rules for the classifier are then selected from R following the sorted sequence. This will ensure that the highest precedence rules are selected for the classifier C. For each rule r, we go through D to find those cases covered by r (they satisfy the conditions of r). We mark r if it correctly classifies a case $d \in D$. If r can correctly classify at least one case (i.e., if r is marked), it will be a potential rule in the classifier. Those cases it covers are then removed from D. A default class is also selected (the majority class in the remaining data). After selection of each rule, the number of total errors made

by the current classifier C, on the training data is recorded. Rules are added to the classifier until no rules are left or no data instances are left. Finally, those rules that do not improve the accuracy of the classifier are discarded. The first rule at which there is the least number of errors recorded on D is the cut-off rule. Hence, all the rules below that point will be discarded. Finally, the selected rule set and the default class label make up the CBA classifier.

- **Coverage with Redundancy**

Some researchers argue [LHM98] that the use of a smaller rule set helps improve the "understandability" as opposed to classification systems having many rules. However, others argue [HL05] that smaller classifiers are too sensitive to the missing values in unseen test data. The latter group proposes to add redundant rules to the classifier to make it more robust. The idea is to make use of alternative (redundant) rules, when some rules are paralyzed by missing values. An example would clarify this point further. Assume two rules ($TestA$ = *positive* $\Rightarrow$ *diabets*) and ($Test\ B$ = *positive and Symptom* = $C \Rightarrow$ *diabetes*), account for a group of patients. Assume that the covering algorithm used to prune the rule set discards the second rule. Now, a patient who has not taken the $Test\ A$ but has tested positive on $Test\ B$ and symptom C will miss the matching of the first rule, and may be miss-classified as normal by the default prediction. This could be crucial in medical applications as there is common case of "missingness" in test data, often due to new patients coming in with low medical history.

The classifier proposed in [HL05] addresses this issue by adding at least two rules to the classifier for every record in the training data. In this process, [HL05] also make sure that the redundant rule added does not have a complete overlap with a

previous rule. For instance, if the first selected rule is $a, b \Rightarrow z$ the second rule will never include both a and b in the antecedent. Experiment results have shown the approach is more robust and accurate.

We propose a completely different and novel approach firstly to avoid the situation of having no applicable rules for a given test case.

3. Rule Selection

Once a test case is presented to the classifier, the classifier has to select the "matching" rule(s) to classify the given instance. Numerous different approaches have been proposed in the literature for the "matching" rule selection. Selection of "Best" Rule, selection of K-nearest rules, selection of all rules that are "close" to the test case etc., are few prevalent techniques.

- **"Best" Rule:**

Some approaches use the "best" rule that covers the test case to classify the instance (e.g [LHM98, DZWL99, HL05]). The best rule is selected based on some evaluation metric discussed above. Such a simple pick may, however, adversely affect the accuracy of the classification since it predicts the consequent based on the "testimony" provided by a single rule, ignoring the simple fact that many rules can have "matching" antecedents (or even the same antecedent), while implying different consequents. The system may also be sensitive to the rather subjective value of the user-specified cut-off thresholds. Hence, collective decision of a set of rules is preferred by many researchers. The CMAR classifier proposed in [LHP01] makes use of all the "matching" rules (selected from the pruned rule set) to make the classification. Most of the others use a selected subset of the matching rules.

• **KNN Method:**

Presented a test instance, the KNN method selects the "nearest" or "best" k rules to make the classification. The "closeness" is measured by some distance metric. This approach is used in [ZSP+04a].

• **Distance Threshold:**

This is the approach used in [HPS07]. In this approach, all the rules that are "closer" than a user defined "distance threshold" participate in the classification of a given test case.

4. Reaching a Decision

Once the subset of rules that matches the incoming test case is selected, they collectively reach the final decision. If all the selected rules suggest the same class label, we can simply assign that label to the test case. However, often, this is not the case.

• **"Best" rule's decision:**

The classifiers that select only the best rule obviously assign the test case to the class suggested by the selected rule.

• **Majority Voting:**

When rules are not consistent in class label, some studies suggest the use of majority voting. Weighted majority makes more sense in this case. CMAR [LHP01] proposes a weighted majority system. When the rules are not consistent in class labels, CMAR divides the rules into groups according to class labels. All rules in a group share the same class label and each group has a distinct label. CMAR compares the strengths

of the groups and yields to the strongest group. Strength of the group is measured in terms of a measure called, weighted X^2. Support values of the rules in the groups are used to compute this measure. Main weakness of this approach is that it ignores the possibility of having many overlapping rules in a group.

• **Evidence Combination:**

Evidence provided by multiple rules can be combined to get the final decision. Dempster's combination rule (DCR) [Sha76] is used to combine the selected rule set in [HPS07, ZSP$^+$04a]. In combining the rules, above-cited papers devise a method to account for confidence and uncertainties associated with rule's opinion. The most attractive aspect about this approach is that the classifier can present the result/ decision as a fused BBA (basic belief assignment) showing how much belief it places on each proposition. A knowledgeable expert may look at this result and make an informed decision rather than accepting a hard decision given by a classifier. We prefer this approach because of the many more reasons to be discussed in later sections.

1.2.2 Accommodating Data Imperfections

Data collected from real world applications such as medical databases are rarely perfect and often consist of missing, inaccurate or inconsistent data. To make better inferences from imperfect data we strongly believe that it is required to develop methods which are capable of accommodating imperfections, and then propagating these throughout the decision-making process. Even though, many association rule based classifiers have been developed, surprisingly few studies have addressed this issue.

A problem we often come across is class label ambiguities. Class label ambiguities are typically generated due to inability of an expert to decide between class labels that are 'close' to each other. For instance, in HIV therapy HAART (highly active antiretroviral therapy) regimes can effectively control the infection. If the prescribed HAART regime has reduced plasma HIV-1 RNA levels of a patient to less than 50 copies/mL it is believed to be a "highly-effective" regime. How about if the regime has reduced the plasma HIV-1 RNA level to 51 or 52 copies/mL? Do you still categorize the regime as "highly-effective"? How far can you vary this boundary? Experts may waver in their decision between "highly-effective" and "effective".

Accommodating such class label ambiguities in an association rule based classifier has been discussed in [ZSP+04a]. They adopt a classifier that is based on belief theoretic notions. Hence, in addition to the improvement in classification performance when class label ambiguities are present, the classifier is also able to provide quantitative confidence information regarding the classification decision it makes. This study, however, did not address the issue of ambiguities associated with other attributes. A belief theoretic based association rule mining framework, in which the rules can accommodate general imperfections (both in class label and attributes) in data has been proposed in [HPS07]. They developed a probabilistic relational data model that allows probabilistic information to be associated with attributes. We strongly, believe that such a framework is essential for knowledge discovery from medical databases which often are rife with data imperfections. Authors in [HPS07], however, came across a major difficulty: the over whelming computational burden. To reduce the computational burden, a data structure referred to as the *belief itemset tree* is used. Yet, they observed that the cost of their association rule mining algorithm is 2 to 3 times higher than the regular apriori algorithm.

1.3 Proposed Work

Our task is made even more complicated by the fact that any (one or more) attribute can take the place of class-attribute. To the best of our knowledge this problem has not been studied in earlier work. In the previous sections we learned that even if we focused on predicting a single class label it could generate a huge number of rules. It becomes even worse when the classifier has to be capable of predicting any attribute. Hence, we have to devise a novel rule generation methodology to limit the number of generated rules, while still ensuring that all interesting rules are discovered. As discussed in [HL05], the weakness of classifiers with small rule sets is that they rely significantly on the default prediction. Default prediction is used when there is no matching rule in the classifier's rule collection to classify a new incoming test case. Predictions based on the default prediction may be misleading. For example, consider a patient data base having 5% of patients suffering from diabetes and 95% patients testing negative on diabetes. Say, we use the default class label, in an effort to classify diabetics. Even though this classifier would pick up 95% accuracy it is meaningless for a doctor. Thus, in the effort to reduce the number of rules used in the classifier we have to avoid the situation of having to use the default class label to classify a new test case.

The usual approach to building a classifier is to discover *all* possible rules from the test database that passes some thresholds, and then to retrieve "matching" rules from this collection to classify a test case. The proposed selective rule generation methodology limits the generated rule set using simple guidelines, and it also avoids the situation of having no applicable rules for a given test case. In the proposed approach, we query the database to extract rules only after a test case is given. We are not interested in a complete list of itemsets, but prefer to constrain the search for

"matching" rules. Thus, we avoid the need of searching and storing "all" rules. Unlike apriori-like algorithms which need repeated searches over the database our algorithm requires just one parse over the database to generate the rule set. Querying the database is made even faster by rearranging the database using the IT-tree data structure proposed in [KHR+03a]. This becomes handy especially in the batch mode prediction (i.e., when you have to predict 'missing items' for several 'shopping carts').

Another fact that is evident in all of above approaches is the enormous amount of effort/cost it takes to obtain a tangible and meaningful set of rules. Costs associated with rule pruning and rule selection can be effectively minimized by better organization of rules. We propose a novel data structure to facilitate the storing and retrieval(selection) of relevant rules.

We also present a novel rule combination methodology based on Dempster's combination rule. The method uses the support, confidence and 'detailedness' (i.e., how specific the rule is) of the rules to assess the strength of the rules.

As discussed in the previous section, it is very important to develop methods capable of accommodating data imperfections and then propagating them throughout the decision making process. Computational burdens associated with such approaches have made it difficult to use them in situations with a large number of attributes. We believe that this novel rule generation and rule organization methodology will be the first step of removing that impediment. With the encouraging results observed on "crisp" databases in the experimental evaluations, we extend this methodology to Dempster-Shafer belief theoretic relational databases (DS-DB)[HPS07] that can conveniently represent a wider class of data imperfections.

1.4 Formal Problem Statement

Let $I = \{i_1, \ldots, i_n\}$ be a set of distinct *items* and let a database consist of transactions $T_1, \ldots, T_N$, such that $T_i \subseteq I, \forall i$. An *itemset,* X, is a group of items, i.e., $X \subseteq I$. The *support* of itemset X is the number, or the percentage, of transactions that subsume X. An itemset that satisfies a user-specified minimum support value is referred to as a *frequent itemset* or a *high support itemset.*

An *association rule* has the form $r^{(a)} \Rightarrow r^{(c)}$, where $r^{(a)}$ and $r^{(c)}$ are itemsets. The former, $r^{(a)}$, is the rule's *antecedent*, and the latter, $r^{(c)}$, is its *consequent.* The rule reads: if all items from $r^{(a)}$ are present in a transaction, then all items from $r^{(c)}$ are also present in the same transaction. The rule does not have to be absolutely reliable. The probabilistic *confidence* in the rule $r^{(a)} \Rightarrow r^{(c)}$ can be defined with the help of supports (relative frequencies) of the antecedent and consequent as the percentage of transactions that contain $r^{(c)}$ among those transactions that contain $r^{(a)}$:

$$conf = \frac{support\left(r^{(a)} \cup r^{(c)}\right)}{support\left(r^{(a)}\right)}. \tag{1.2}$$

Let us assume that an association mining program has already discovered all high support itemsets. For each such itemset, X, any pair of subsets, $r^{(a)}$ and $r^{(c)}$, such that $r^{(a)} \cup r^{(c)} = X$ and $r^{(a)} \cap r^{(c)} = \emptyset$, we can define an association rule: $r : r^{(a)} \Rightarrow r^{(c)}$. The number of such rules generated grows exponentially in the number of items in X, but we usually consider only rules derived from high support itemsets and with confidence satisfying a user-specified minimum.

Given an itemset s, an algorithm developed by [KHR+03a] generates, in a computationally feasible manner, *all* rules $s \Rightarrow \ell$ that satisfy the user-supplied minimum support and confidence values θ_s and θ_c, respectively. Of course, if no frequent itemset subsumes s, no rules will be generated.

However, we are also interested in rules with antecedents that are subsumed by s. As an example, suppose no frequent item set subsumes $s = (\texttt{wine}, \texttt{milk}, \texttt{bread})$. To claim that s does not imply any items, as one may infer from the algorithm in [KHR$^+$03a], would mean to ignore associations implied by *subsets* of s. For instance, the antecedent of $(\texttt{milk}, \texttt{bread}) \Rightarrow \texttt{butter}$ is subsumed by s; we will say that $(\texttt{milk}, \texttt{bread}) \Rightarrow \texttt{butter}$ "matches" s. If we have other such matching rules, for instance, $(\texttt{wine}, \texttt{bread}) \Rightarrow \texttt{egg}$ or $\texttt{wine} \Rightarrow \texttt{beer}$, we would like to consider them as well. We thus need a mechanism that not only generates, but also combines, rules that match s.

Furthermore, we need to be aware of the circumstance that the presence of an item might suggest the *absence* of other items. For instance, if the shopping cart contains `chips`, `cookies`, `cashews`, the customer may *not* buy `nuts`. We are therefore interested in rules such as $(\texttt{chips}, \texttt{cookies}, \texttt{cashews}) \Rightarrow \neg\texttt{nut}$, where $\neg\texttt{nuts}$ means that *no* nuts will be added to the cart. Classical association mining usually ignores this aspect, perhaps because negated items tend to increase significantly the total number of rules to be considered; another reason can be that rules with mutually contradicting consequents are not so easy to combine.

With the issues discussed above in mind, we narrow down the space of association rules by using following guidelines:

(1) For a given itemset s, rule antecedents should be subsumed by s.

(2) The rule consequent is limited to only one 'unseen' item (presence or absence of the unseen item).

In essence, the tasks being addressed in this paper are the following: Given a transaction with the itemset $s \subseteq I$, find the set of matching rules that exist in the training dataset that are of the form $r^{(a)} \Rightarrow i_j$, $j = \overline{1,n}$, such that $r^{(a)} \subseteq s$ and

$i_j \notin s$, and pass the user-supplied minimum support, θ_s, and minimum confidence, θ_c, thresholds. Then, devise a method to combine the matching rules with mutually contradicting consequents and reach a decision on what other items would be added to the transaction.

CHAPTER 2

Preliminaries

Probability theory does not provide a satisfactory way to model ignorance or lack of information. This is often illustrated using the famous coin tossing example. Consider two pieces of evidence provided by two players tossing a coin.

- Payer A – knows that the coin being tossed is fair.
- Player B – does not know the coin being tossed is fair.

In either case, a probability theory based model would assign $P(heads) = P(tails) = 0.5$. This assignment may be satisfactory as far as the player A is concerned; but, as far as B is concerned, it is based on lack of knowledge and the coin may have a bias towards heads or tails.

To address such situations, one typically relaxes the additivity axiom of probability theory, i.e., we do not impose the condition $P(X) + P(\overline{X}) = 1$. One approach that uses this strategy is Dempster-Shafer (DS) belief theory [Sha76].

Belief or evidence theory was first introduced by Dempster [Dem67] and further developed by Shafer [Sha76]. Since then it has become a very popular research topic and many developments and implementations have taken place. Apart from the obvious advantage of having a method for representing a wider class of imperfections, Dempster-Shafer (DS) evidence theory has other advantages as well. In the Bayesian

model, probabilities have to be assigned to all the propositions of interest, and as new information is obtained these measures are updated. In DS theory, depending on the evidence that is available, masses are assigned to only those propositions that are supported by the evidence. As new evidence becomes available, masses are assigned to another set of propositions that are supported by the new evidence and these two sets of masses can be combined to give a new set of propositions that are supported by the combined evidence. After masses have been assigned to the propositions supported by the evidence, the surplus mass is associated with ignorance. As more evidence is received this belief is spread out into different propositions, reducing the ignorance. DS theory is a powerful tool in handling imperfections in databases. Therefore, we will base our discussion on this formalism from here onwards.

2.1 DS Theory

Consider a set of mutually exclusive and exhaustive propositions, $\Theta = \{\theta_1, \ldots, \theta_K\}$, referred to as the *frame of discernment (FoD)*. A proposition θ_i, referred to as a *singleton*, represents the lowest level of discernible information. In our context, θ_i states that the "value of attribute is equal to θ_i." Elements in 2^Θ, the power set of Θ, form all propositions of interest. Any proposition that is not a singleton, e.g. (θ_1, θ_2), is referred to as *composite*. Table 2.1 summarizes the commonly used terminology in DS-theory. To keep the discussion simple we assume that $|\Theta|$ is finite.

DS theory assigns to any set, $A \subseteq \Theta$, a numeric value $m(A) \in [0, 1]$, called a *basic belief assignment (BBA)* or *mass* that quantifies the evidence one has towards the proposition that the given attribute value is A and only A. The mass function has to

Table 2.1: Notations

$\Theta = \{\theta_i\}$	Frame of discernment (FoD) or frame or sample space; set of all possible mutually exclusive and exhaustive objects.
θ_i	Singletons or elements of Θ; represents the lowest level of discernible information.
2^Θ	Powerset of Θ; its elements form all the propositions of interest in DS theory.

satisfy the following conditions [Sha76]:

$$\begin{aligned} &i.)\ m(\emptyset) = 0; \\ &ii.)\ \sum_{A \subseteq \Theta} m(A) = 1. \end{aligned} \tag{2.1}$$

Note that, if $\bar{A}$ is the complement of A, then $m(A) + m(\bar{A}) \leq 1$.

Objects for which there is no information are not assigned an a-priori mass. Hence committing support for an event does not necessarily imply that the remaining support is committed to its negation; the lack of support for any particular event simply implies support for all other events. In this manner, the additivity axiom in the probability formalism is relaxed in DS theory.

Any proposition A that possesses a non-zero mass, i.e., $m(A) > 0$, is called a *focal element;* the set of focal elements, $\mathfrak{F}$, is referred to as the *core.*

$$\mathfrak{F}(\Theta) = \{A \subseteq \Theta : m(A) > 0\}.$$

The triple $\{\Theta, \mathfrak{F}, m(\bullet)\}$ is called the *body of evidence (BoE).*

Complete ignorance of the state of nature can be modeled via the *vacuous bba*:

$$m(A) = \begin{cases} 1, & \text{for } A = \Theta; \\ 0, & \text{for } A \subset \Theta. \end{cases} \tag{2.2}$$

An indication of the evidence one has towards all propositions that may themselves imply a given proposition $A \subseteq \Theta$ is quantified via the *belief,* $Bel(A) \in [0,1]$, defined as:

$$Bel(A) = \sum_{B \subseteq A} m(B). \tag{2.3}$$

Note that $Bel(A) = m(A)$ if A is a singleton. *Plausibility* of A is defined as $Pl(A) = 1 - Bel(\overline{A})$; it represents the extent to which one finds A plausible.

2.1.1 Belief to Probability Transformation

Many techniques have been developed for converting DS theoretic information into the probabilistic domain; see [CS03] for a detailed discussion on various belief to probability transformations. The most commonly used transformation method is the pignistic transformation [Sme90].

A probability distribution $Pr(\cdot)$ satisfying $Bel(A) \leq Pr(A) \leq Pl(A)$, $\forall A \subseteq \Theta$, is said to be compatible with the underlying BBA $m(\bullet)$. As mentioned before, an example of such a probability distribution is the *pignistic probability distribution* $BetP(\bullet)$ defined for each singleton $\theta_i \in \Theta$ as follows [Sme99]:

$$BetP(\theta^{(i)}) = \sum_{\theta_i \in A \subseteq \Theta} \frac{m(A)}{|A|}. \tag{2.4}$$

Here $|A|$ denotes the cardinality of set A.

2.1.2 Belief Combination

Dempster's rule of combination (DRC) makes it possible to arrive at a new BoE by fusing the information from several BoEs that span the same FoD. Consider the two BoEs, $\{\Theta, \mathfrak{F}_1, m_1(\bullet)\}$ and $\{\Theta, \mathfrak{F}_2, m_2(\bullet)\}$. Then,

$$K_{12} = \sum_{B_i \cap C_j = \emptyset} m_1(B_i)\, m_2(C_j) \tag{2.5}$$

is referred to as the *conflict* because it indicates how much the evidence of the two BoEs are in conflict. If $K_{12} < 1$, then the two BoEs are compatible, and the two BoEs can be combined to obtain the overall BoE $\{\Theta, \mathfrak{F}, m(\bullet)\}$ as follows: for all $A \subseteq \Theta$,

$$\begin{aligned} m(A) &\equiv (m_1 \oplus m_2)(A) \\ &= \frac{\sum\limits_{B_i \cap C_j = A} m_1(B_i)\, m_2(C_j)}{1 - K_{12}}. \end{aligned} \tag{2.6}$$

A variation of the DRC that can be used to address the reliability of the evidence provided by each contributing BoE is to incorporate a *discounting factor* d_i, $d_i \leq 1$, to each BoE [Sha76]. The BBA thus generated is

$$m(A) = (\hat{m}_1 \oplus \hat{m}_2)(A), \text{ where, for } i = 1, 2,$$

$$\hat{m}_i(A) = \begin{cases} d_i m_i, & \text{for } A \subset \Theta; \\ (1 - d_i) + d_i m_i(\Theta), & \text{for } A = \Theta. \end{cases} \tag{2.7}$$

2.1.3 Decision making

In our context, an attribute can take one out of many (finite) values. One singleton denotes the true value associated with an attribute x. In the effort to predict the value of attribute x, our predictor presents the decision in the form of a BoE, $\{\Theta_{L,x}, \mathfrak{F}_{L,x}, m_{L,x}\}$ where $\Theta_{L,x}$ is the set of values x shall take. If we want a 'hard' decision on a *singleton* value, one may use the pignistic probability, BetP, to pick a singleton.

Now that we established the basic notions of DS-theory, it is time to establish the basic notions of the IT-tree concept. The rest of this chapter will present the basics if IT-trees.

2.2 IT-tree

The Itemset-tree or IT-tree is developed by Kubat et. al. [KHR+03a] in 2003 with the intention of expediting targeted queries in market basket type transactional datasets. IT-tree is a compact and easily updatable representation of the transactional dataset. Experiments in [KHR+03a] indicate that the targeted queries are answered in a time that is roughly linear in the number of market baskets, N. Also, the construction of the itemset tree has $O(N)$ space and time requirements. We use this data structure to speed up the proposed predictor.

As outlined in section 1.4 we formulate our problem as a selective rule generation problem, where we examine the database by submitting specialized queries. Specifically, we want to search for rules having matching antecedents for a given test case. An example taken from the department-store paradigm would illustrate it further. Say a partially observed market basket contains `bread, milk`. We want to predict all items that can be added to the market basket. Query of this kind seek for rule of the form [`bread, milk`]$\Rightarrow i$, [`milk`]$\Rightarrow i$ or [`bread`]$\Rightarrow i$(where i is an item). Search for all rules would waste computational resources.

2.2.1 Building itemset trees

Let D be the dataset and p be the number of distinct items in the dataset. Each item is identified with an integer from $[1, p]$, so that the items in an itemset $\{I_1, I_2, \ldots, I_m\}$ can be ordered according to their corresponding integer value — $I_i < I_j$ for $i < j$, where I_i and I_j are integers identifying ith and jth items, respectively.

Let us first establish the relationships between nodes in the tree–ancestor, largest common ancestor, and child; each represents an itemset.

Definition 1 (Ancestor, Largest common ancestor, Child) *Let the symbols s, c and l denote itemsets.*

(1) s is an ancestor of c and write $s \sqsubseteq c$, iff $s = \{I_1, I_2, \ldots, I_m\}, c = \{I_1, I_2, \ldots, I_n\}$, and$m \leq n$.

(2) l is the largest common ancestor of s and write $l = s \sqcap c$, iff $l \sqsubseteq s, l \sqsubseteq c$, and there is no l' such that $l' \sqsubseteq s, l' \sqsubseteq c$, and $l \sqsubseteq l', l \neq l'$.

(3) c is a child of s iff $s \sqsubseteq c$ and there is no l, different from s and c, such that $s \sqsubseteq l \sqsubseteq c$. ■

Note that an ancestor of c is an uninterrupted sequence of the smallest items in c. For instance, $[1, 2]$ is an ancestor of $[1, 2, 3, 4]$. Itemsets $[1, 2]$ and $[2, 5]$ cannot have a common ancestor because any ancestor of $[1, 2]$ has to begin with item 1 that is not contained in $[2, 5]$. This case is dealt with by having a root node whose itemset is empty. By definition, an empty set is a (trivial) ancestor of any nonempty itemset. Note that, if s is an ancestor of c, then s is also the largest common ancestor of s and c.

Definition 2 (Itemset-Tree) *An itemset tree, T, consists of a root and a (possibly empty) set, $\{T_1, \ldots, T_k\}$, each element of which is an itemset tree. The root is a pair $[s, f(s)]$, where s is an itemset and $f(s)$ is a frequency. If s_i denotes the itemset associated with the root of the ith subtree, then $s \sqsubseteq s_i$, $s \neq s_i$, must be satisfied for all i.* ■

An IT-tree is a partially ordered set of pairs, $[itemset, f]$, where the f-value tells us how many occurrences of the itemset the node represents. An algorithm that builds the IT-tree in the course of a single pass through the database is also presented in [KHR$^+$03a]. The paper also proves some properties of the algorithm. The number

of nodes in the IT-tree is upper-bounded by twice the number of transactions in the original database (experiments indicate that, in practical applications, the size of the IT-tree rarely exceeds the size of the database). Moreover, each distinct transaction database is represented by one and only one distinct IT-tree and the original transaction can be reproduced from the IT-tree.

Example 1 (An IT-tree) *Figure 2.1 shows an itemset tree created from the dataset D={[1,5], [2,3], [1,2,3,4], [1,2,4], [2,3], [2,4]}.* ■

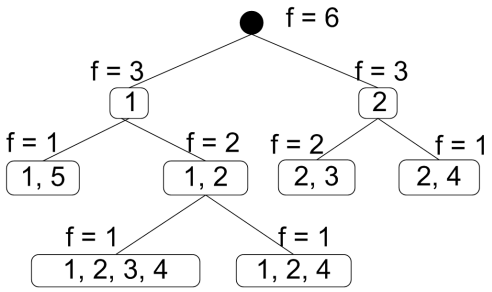


Figure 2.1: Itemset-tree constructed from the database, D={[1,5], [2,3], [1,2,3,4], [1,2,4], [2,3], [2,4]}.

2.2.2 Flagged IT-trees

In [LK06], Li, et al, made a useful modification to the algorithm. Note that some of the itemsets in IT-tree (e.g., $[1, 2, 4]$ in Fig. 2.1) are identical to at least one of the transactions contained in the original database, whereas others (e.g., $[1, 2]$) were created during the process of tree building where they came into being as common ancestors of transactions from lower levels. [LK06] modified the original tree-building algorithm so that it flags each node that is identical to at least one transaction. In Fig. 2.2, the flags are indicated by black dots. We use this flagged tree as the base of our rule generation algorithm.

Example 2 (A Flagged IT-tree) *Consider the database D={[1,5], [2,3], [1,2,3,4], [1,2,4], [2,3], [2,4]}. Fig. 2.2 shows the flagged IT-tree created for this dataset.* ■

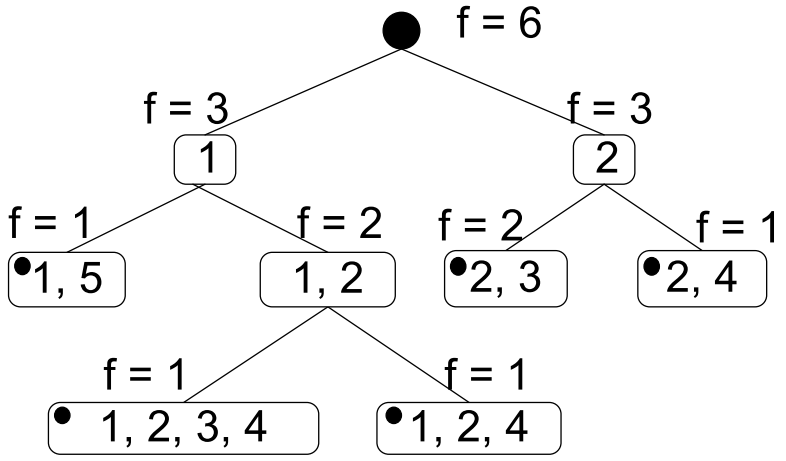


Figure 2.2: Flagged Itemset-tree constructed from the database, D={[1,5], [2,3], [1,2,3,4], [1,2,4], [2,3], [2,4]}.

CHAPTER 3

Building the Predictor

As explained in the section 1.4, association rule mining is defined in the domain of binary attributes $I = \{i_1, \ldots, i_n\}$, called *items*. Binary attributes themselves are a special case of 'categorical' attributes. The domain of categorical attributes is not limited to simply True and False values, but could be any arbitrary finite set of values. An example of a categorical attribute is `color` whose domain, for instance, may include the values *brown*, *black*, and *white*. To generate association rules in case of categorical attributes, common practice is to consider $\langle attribute,\ value\rangle$ pair as an *item*. For instance, $\{\langle color,\ brown\rangle, \langle color,\ black\rangle, \langle color,\ white\rangle\}$ are considered as three items, each could be True or False for a given data instance.

In the prediction phase, if we are to predict the value of the attribute `color`, we consider all the rules having any one of the above three *items*, $\{\langle color,\ brown\rangle$, $\langle color,\ black\rangle$, $\langle color,\ white\rangle\}$, as the rule consequent. The final decision on the value of the `color` attribute is reached via an evidence combination approach which treats the each rule as a piece of evidence. The following example will further clarify this approach.

Example 3 (A prediction example) *Let T be a small training set (see table 3.1). It is composed of five objects (persons) characterized by four attributes defined as follows: Eyes =$\{$Brown, Blue, Black$\}$; Hair = $\{$Dark, Blond$\}$; Skin = $\{$Black, Brown, Fair$\}$, Race=$\{$Asian, African, Indian, White$\}$.*

Table 3.1: An Illustrative Data set of five persons

	Eyes	Hair	Skin	Race
1	Blue	Blond	Fair	White
2	Black	Dark	Fair	Asian
3	Black	Dark	Black	African
4	Brown	Dark	Brown	Indian
5	Brown	Dark	Fair	White

This dataset can be converted from the above table format to a transactional format by considering each $\langle attribute,\ value \rangle$ pair as an item. Each item can be identified with an integer as shown in Table 3.2. Finally, the dataset in table 3.1, is converted to the transaction dataset shown in table 3.3.

Table 3.2: Integer notation for $\langle attribute,\ value \rangle$ pairs, considering each pair as an item

Item–value pair	Identifying Integer	Item–value pair	Identifying Integer
(Eyes, Brown)	1	(Skin, Brown)	7
(Eyes, Blue)	2	(Skin,Fair)	8
(Eyes, Black)	3	(Race, Asian)	9
(Hair,Dark)	4	(Race,African)	10
(Hair, Blond)	5	(Race,Indian)	11
(Skin, Black)	6	(Race,White)	12

Table 3.3: The dataset in the table 3.1 as a transaction dataset using the integer notation

Tx. #	Item set
1	2,5,8,12
2	3,4,8,9
3	3,4,6,10
4	1,4,7,11
5	1,4,8,12

Given a test case: {(Eyes=Blue), (Hair=Blonde)}, we want to predict the race and skin-complexion. We are now interested in rules having (Eyes=Blue) and/or (Hair=Blonde) in the antecedent and race or skin-complexion in the consequent. In

other words, we want those rules having item(s) 2 and/or 5 in the antecedent and one of the items from {6,7,8,9,10,11,12} in the consequent. To reach a decision on the skin-complexion, for instance, we will apply evidence combination to those rules having item 6, 7, or 8 as the consequent. Final decision of the predictor will be presented as a 'soft' label in the form of a BoE, which shows the predictor's belief on each proposition 6, 7 or 8.

For instance, the output could be:(all values are illustrative)

$$\begin{array}{rcrrcr} m(6) = & 0.01 & \Rightarrow & m(Skin = Black) = & 0.01 \\ m(7) = & 0.14 & \Rightarrow & m(Skin = Brown) = & 0.14 \\ m(8) = & 0.80 & \Rightarrow & m(Skin = Fair) = & 0.80 \\ m(\Theta) = & 0.05 & \Rightarrow & m(\Theta) = & 0.05 \end{array}$$

If a 'hard' decision is required we can pick the proposition having the highest pignistic probability as the correct label. In the above case: (Skin=Fair).

Even though the attribute values in the above example are all "crisp", it is worth mentioning here that the DS-ARM predictor is capable of accommodating probabilistic, possibilistic or ambiguous values. A detailed discussion is given in Chapter 5. ■

For the sake of easy understanding we will ignore the notion of *attributes, values* in this chapter and only consider *items* which take only two values 1(True) or 0(False). Under this formulation, our task is to predict the missing items in a partially observed transaction.

Although we are not aware of any other study that applies to our task of "predicting missing items in a shopping cart," the Bayesian approach that has been around since before World War II [NP28, Fis36] can be adopted to serve the purpose, as explained below.

3.1 Bayesian Approach

The mathematically "clean" version is known to be computationally expensive in domains where many independent variables are involved. Fortunately, this difficulty can be side-stepped by the so-called Naive Bayes assumption that assumes that all variables are mutually pairwise conditionally independent [Goo65]. Although this assumption is rarely strictly satisfied, decades of machine learning research have shown that, most of the time, the Naive Bayes assumption can be invoked nevertheless—conditional interdependence of variables affects classification behavior of the resulting formulas only marginally.

Suppose we want to establish whether the presence of item set $s = \{i_1^{(s)}, \ldots, i_k^{(s)}\}$ increases the chance that item $i_j \notin s$ is also present. Bayes' rule yields

$$P(i_j | i_1^{(s)}, \ldots, i_k^{(s)}) = \frac{P(i_1^{(s)}, \ldots, i_k^{(s)} | i_j)\, P(i_j)}{P(i_1^{(s)}, \ldots, i_k^{(s)})}. \tag{3.1}$$

The item that yields the highest value for this probability is then selected. Since the denominator is the same for any index k, it is enough if the classifier chooses the item that leads to the highest value of the numerator.

With the Naive Bayes assumption, pairwise independence of $i_1^{(s)}, \ldots, i_k^{(s)}$ conditional to i_j yields the following expression for the numerator of (3.1):

$$P(i_1^{(s)}, \ldots, i_k^{(s)} | i_j) = \prod_j P(i_j^{(s)} | i_j). \tag{3.2}$$

So, i_j is chosen if it maximizes $P(i_j) \prod_j P(i_j^{(s)} | i_j)$.

One practical problem with this formula is that its value is zero if $P(i_\ell^{(s)} | i_j) = 0$ for some $\ell = \overline{1, k}$, a situation that will occur quite often, given the sparseness of the data in association mining applications. This difficulty is easily rectified if we estimate the conditional probabilities by the $\mathfrak{m}$-estimate (originally proposed for machine learning

applications by [CB91]) that makes it possible to bias the probabilities toward user-set a priori values as follows. Let us constrain ourselves to a binary domain where a set of trials has resulted either in $\oplus$ or $\ominus$. Let us denote by $\aleph_{\oplus}$ and $\aleph_{\ominus}$ the number of occurrences of the respective outcomes, and let $\aleph_{all} = \aleph_{\oplus} + \aleph_{\ominus}$. If, before the trials, the user's prior expectation of the probability of $\oplus$ was $p_{\oplus}$, after the trials, the probability of $\oplus$ is estimated as

$$P_{\oplus} = \frac{\aleph_{\oplus} + \mathfrak{m}\, p_{\oplus}}{\aleph_{all} + \mathfrak{m}}. \tag{3.3}$$

The parameter $\mathfrak{m}$ quantifies the experimenter's confidence in this estimate. Note that, for $\aleph_{all} = 0$ (which implies $\aleph_{\oplus} = 0$), (3.3) degenerates to the prior expectation, $p_{\oplus}$. Conversely, the equation converges to relative frequency if $\aleph_{all}$, is so large that the terms $\mathfrak{m}p_{\oplus}$ and $\mathfrak{m}$ can be neglected. Generally speaking, for low values of $\mathfrak{m}$, even small evidence will affect the prior estimate; the higher the value of $\mathfrak{m}$, the more evidence is needed to overturn the prior estimate.

In the prediction context, we want to compare the probability estimates of the presence/absence of one item based on another item. For $\mathfrak{m} = 2$, the $\mathfrak{m}$-estimate will be calculated as

$$P(i_{\ell}^{(s)}|i_j) = \frac{count(i_{\ell}^{(s)}|i_j) + 1}{count(i_j) + 2}. \tag{3.4}$$

For a given i_j, this formula is used to calculate the values $P(i_{\ell}^{(s)}|i_j)$ for all items in the antecedent. This makes it possible to evaluate the probabilities in (3.2), which in turn are used to calculate the posteriors in (3.1). The following rule is then used to predict the "missing items":

Finding $s = \{i_1^{(s)}, \ldots, i_k^{(s)}\}$ in the shopping cart, predict all i_j such that $P(i_j|i_1^{(s)}, \ldots, i_k^{(s)}) > P(\neg i_j|i_1^{(s)}, \ldots, i_k^{(s)})$.

Returning back to the notion of attributes and values, if a_k is a categorical attribute that can take any value from $\{y_j |\ j = 1 \ldots p\}$, we predict $a_k = y_c$ if $P(a_k = y_c|i_1, \ldots, i_n) >= P(a_k = y_j|i_1, \ldots, i_n),\ \forall\ j$. Value of all such "unknown" a_k' are to be predicted.

3.2 DS-ARM Predictor

Association rule mining in its original form finds all the rules existing in the data base that satisfy some minimum support and minimum confidence constraints. The target of discovery is not pre-determined. Many later works tried to integrate classification and association rule mining. The objective was to build a classifier using a special set of association rules, so-called *class association rules*. The difference in classification rule mining is that there is one and only one pre-determined target, the class label. Classification rule mining, most of the time, is applied to databases in table format, where you have a set of attributes and a class label. Even though missing values are allowed for attributes, they usually, take a value out of a finite set of values.

Some previous work, including [HPS07] and [ZSP+04b], has shown encouraging results by incorporation of DS theoretic notions with class association rules. However, most of these methods are designed for datasets with limited number of attributes (or datasets with small number of distinct items) and one class label. We propose a novel approach to predict all missing items in a shopping cart. The algorithm uses the strengths of IT-trees to speed up the rule generation in batch mode predictions. It uses DS theoretic notions to combine the generated rules and present the prediction decision. Figure 3.1 shows a basic flowchart of DS-ARM predictor.

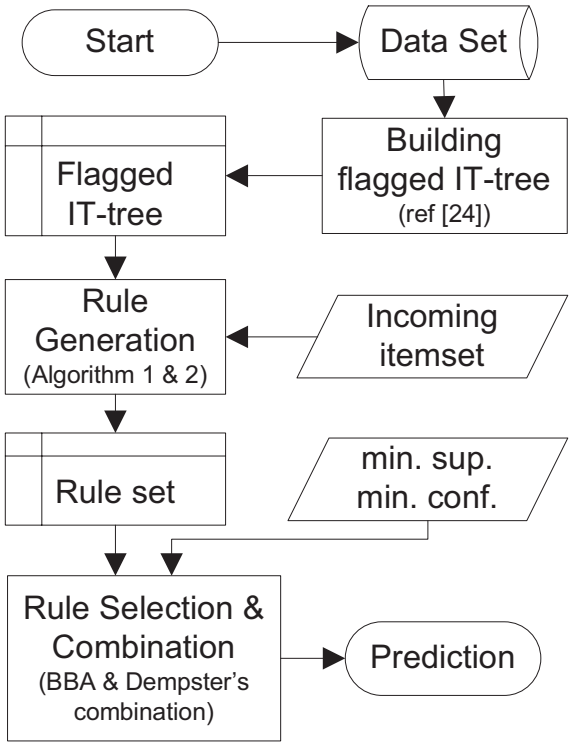


Figure 3.1: Flowchart of the predictor

3.2.1 Rule Generation

Let $I = \{I_1, \ldots, I_n\}$ be the set of items in the dataset. Given a record with item set s our objective is to find all rules of the form $r_i^{(a)} \Rightarrow I_j$, $j = \overline{1,p}$, where $r_i^{(a)} \subseteq s$ and $I_j \notin s$, that pass minimum support and minimum confidence thresholds.

It should be noticed that the consequent is always a single unseen item for the given shopping cart. Also, note that the prediction (consequent) could be $\langle item = present \rangle$ or $\langle item = absent \rangle$. Then, for each of the unseen item, the corresponding rule set is selected and a DS theoretic approach is used to combine the rules. Prediction is given as a mass structure, with focal elements being $I_j = present$ and $I_j = absent$. If no rule consequent in the generated ruleset contains a particular unseen item, no prediction is made regarding that item (vacuous BoE).

Proposed Solution

The proposed algorithm takes an incoming itemset as the input and returns a graph that illustrates the association rules that would be fired by the given incoming itemset. The graph structure is used to properly store and efficiently retrieve the large number of classification rules.

The graph consists of two lists: the antecedents list $\Re^{(a)}$ and the consequents list $\Re^{(c)}$. Each node $r_i^{(a)}$ in the antecedents list keeps its corresponding frequency count $f(r_i^{(a)})$. As shown in Fig. 3.2, a line $\ell_{i,j}$, between the two lists links an antecedent $r_i^{(a)}$ with a consequent I_j. The cardinality of the link, $f(\ell_{i,j})$ represents the support count of the rule $r_i^{(a)} \Rightarrow I_j$. The frequency counts denoted by $f_o(\bullet)$ are used in the building of the graph. If the incoming itemset is s and T_i represents a transaction in the database, $f_o(r^{(a)})$ records the number of times $s \cap T_i = r^{(a)}$. $f_o(\ell_{i,j})$ records the number of times where $s \cap T_i = r^{(a)}$ and $I_j \in T_i$. All the frequency counts are initialized to zero at the beginning of the algorithm and are updated as we scan through the database.

We propose a novel rule generation algorithm to generate the *matching* set of association rules hidden in the training set and that would be fired by the incoming itemset. The required rule set can be generated by using algorithm 1, in combination with algorithm 2 using only a single parse over the data set. Properties of the rule set will be discussed later.

The basic idea of the Update_Graph() algorithm is to update the frequency counts of all the rules that already exist in the rule graph and of the form $r_i^{(a)} \Rightarrow I_j$ where $r_i^{(a)} \subseteq r^{(a)}$, for every new candidate rule $r^{(a)} \Rightarrow I_j$ where $I_j \in \{T_i \setminus r^{(a)}\}$. If the new rule does not exist in the rule graph, it has to be added to the graph and frequency count has to be updated using all the rules, of the form $r_i^{(a)} \Rightarrow I_j$ where $r^{(a)} \subseteq r_i^{(a)}$,

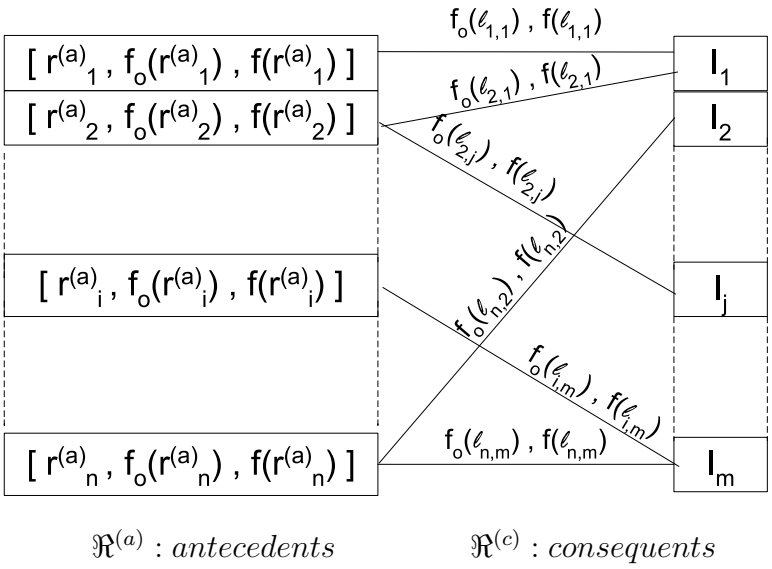


Figure 3.2: The Rule Graph, G. $f(r_i^{(a)})$= frequency count of antecedent, $f(\ell_{i,j})$=support count of rule $r_i^{(a)} \Rightarrow I_j$

Algorithm 1 Algorithm that processes the dataset D and returns the rule graph G that predicts 'missing' items in a partially observed transaction s.

Let T_i denote a transaction in D
To invoke *Rule_mining* use: $G = Rule\_mining(s, D, \{\})$.

1: **Rule_mining**(s,D,G):
2: **for all** $T_i \in D$ **do**
3: **if** $r^{(a)} \equiv s \cap T_i \neq \emptyset$ **then**
4: $G \leftarrow Update\_Graph(G, r^{(a)}, T_i, 1)$;
5: **end if**
6: **end for**
7: **return** G;

in the graph. A simplified version of the algorithm is given in Algorithm 2. A more detailed and complete algorithm is given in algorithm 3. Note that this algorithm is not intended to generate all the rules, but the *matching* set of rules that truly exist in the training dataset that would build an effective classifier. We will elaborate on this in a later example.

Algorithm 2 Simplified algorithm to update the rule graph.

Let G denote the current rule graph.
Let $\mathfrak{c}_i$ denote an itemset from a node and f_o denote the number of appearances of $\mathfrak{c}_i$ in the database.
Let $r^{(a)} = \mathfrak{c}_i \cap s$ where s is the incoming itemset.
To invoke *Update_Graph* use: $G = Update\_Graph(G, r^{(a)}, \mathfrak{c}_i, f_o)$.

Update_Graph$(G, r^{(a)}, \mathfrak{c}_i, f_o)$:
for all $r_i^{(a)}$ in $\Re^{(a)}$ **do**
  if $r_i^{(a)} \subseteq r^{(a)}$ **then**
    update the frequency count of $r_i^{(a)}$, $f(r_i^{(a)}) \leftarrow f(r_i^{(a)}) + f_o$;
    for all $r_i^{(a)} \Rightarrow I_j$ where $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
      update frequency count of rule $r_i^{(a)} \Rightarrow I_j$, $f(\ell_{i,j}) \leftarrow f(\ell_{i,j}) + f_o$;
    end for
    if $r_i^{(a)} = r^{(a)}$ **then**
      update frequency count record, $f_o(r_i^{(a)}) \leftarrow f_o(r_i^{(a)}) + f_o$;
      for all $r_i^{(a)} \Rightarrow I_j$ where $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
        update frequency count record of rule,$f_o(\ell_{i,j}) \leftarrow f_o(\ell_{i,j}) + f_o$;
      end for
    end if
  else if $r^{(a)} \subset r_i^{(a)}$ **then**
    update the frequency count of new rule antecedent $f(r^{(a)}) \leftarrow f(r^{(a)}) + f_o(r_i^{(a)})$;
    for all $r_i^{(a)} \Rightarrow I_j$ in G where $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
      update the frequency count of new rule $r^{(a)} \Rightarrow I_j$, (add $f_o(\ell_{i,j})$) ;
    end for
  end if
end for
for all $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
  if $\nexists (r^{(a)} \Rightarrow I_j) \in G$ **then**
    add the rule to the graph with updated frequency counts;
  end if
end for
return G;

Algorithm 3 Detailed algorithm to update the rule graph.

Let G denote the current rule graph.
Let $\mathfrak{c}_i$ denote an itemset from a node and f_o denote the number of appearances of $\mathfrak{c}_i$ in the database.
Let $r^{(a)} = \mathfrak{c}_i \cap s$ where s is the incoming itemset.
To invoke *Update_Graph* use: $G = Update\_Graph(G, r^{(a)}, \mathfrak{c}_i, f_o)$.

Update_Graph$(G, r^{(a)}, \mathfrak{c}_i, f_o)$:
$L \equiv [r^{(a)}, f_o]$; ▷ L is a temporary list to hold frequency counts
for all $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
    $L \leftarrow L \cup [I_j, f_o]$; ▷ initialize the list
end for
for all $[r_i^{(a)}, f_o(r_i^{(a)}), f(r_i^{(a)})]$ in $\Re^{(a)}$ **do**
    if $r_i^{(a)} \subset r^{(a)}$ **then**
        $f(r_i^{(a)}) \leftarrow f(r_i^{(a)}) + f_o$;
        for all $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
            if $\exists$ a link $\ell_{i,j}$ from $r_i^{(a)}$ in $\Re^{(a)}$ to I_j in $\Re^{(c)}$ **then**
                $f(\ell_{i,j}) \leftarrow f(\ell_{i,j}) + f_o$;
            end if
        end for
    else if $r^{(a)} \subset r_i^{(a)}$ **then**
        $f^{(L)}(r^{(a)}) \leftarrow f^{(L)}(r^{(a)}) + f_o(r_i^{(a)})$; ▷ $f^{(L)}(r^{(a)})$ is the frequency count of $r^{(a)}$ in L
        for all $\ell_{i,j}$ leading from $r_i^{(a)}$ to I_j in $\Re^{(c)}$ **do**
            $f^{(L)}(I_j) \leftarrow f^{(L)}(I_j) + f_o(\ell_{i,j})$;
        end for
    end if
end for
if $\exists\ r_i^{(a)} \in \Re^{(a)}$ such that $r_i^{(a)} = r^{(a)}$ **then**
    $f(r_i^{(a)}) \leftarrow f(r_i^{(a)}) + f_o$;
    $f_o(r_i^{(a)}) \leftarrow f_o(r_i^{(a)}) + f_o$;
    for all $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
        if $I_j \notin \Re^{(c)}$ **then** add I_j to $\Re^{(c)}$;
        end if
        if $\nexists$ a link $\ell_{i,j}$ from $r_i^{(a)}$ to I_j in $\Re^{(c)}$ **then**
            add link $\ell_{i,j}$ and set $f(\ell_{i,j}) \leftarrow f^{(L)}(I_j)$;
            $f_o(\ell_{i,j}) \leftarrow f_o$;
        else
            set $f(\ell_{i,j}) \leftarrow f_o(\ell_{i,j}) + f^{(L)}(I_j)$;
            $f_o(\ell_{i,j}) \leftarrow f_o(\ell_{i,j}) + f_o$;
        end if
    end for
else
    add $[r^{(a)}, f_o, f^{(L)}(r^{(a)})]$ to $\Re^{(a)}$;
    for all $I_j \in (\mathfrak{c}_i \setminus r^{(a)})$ **do**
        if $I_j \notin \Re^{(c)}$ **then**
            add I_j to $\Re^{(c)}$;
        end if
        add link $\ell_{i,j}$ from $r^{(a)} \in \Re^{(a)}$ to $I_j \in \Re^{(c)}$;
        set $f(\ell_{i,j}) \leftarrow f^{(L)}(I_j)$;
        $f_o(\ell_{i,j}) \leftarrow f_o$;
    end for
end if
return G;

3.2.2 Speeding-up the Computations by Itemset-Trees

To expedite the rule generation process in batch mode, we take the assistance of 'flagged itemset tree' concept proposed in [LK06]. It has been shown that it is much faster to scan the compact data structure, *IT-tree*, rather than scanning the actual data set that resides in the disc. The rule generation algorithm proposed in [KHR$^+$03a] is modified for two reasons. Firstly, the algorithm proposed in [KHR$^+$03a] addresses a slightly different task. It generates all the rules of the form, $s \Rightarrow \ell$ where s and ℓ are itemsets and $s \cap \ell = \emptyset$. As explained in the beginning of this section, we are interested in a slightly different set of rules. Secondly, our objective is not to generate all the association rules, but to build a predictor using a set of effective association rules.

Algorithm 4 does a depth-first search in the IT-tree to find nodes that possess non-empty intersections with the incoming itemset s. Steps of the algorithm can be summarized as below. Let $R = [s_R, f(s_R)]$ denote the root, let c_i denote the children of R, and let s denote the incoming itemset. If the first item in c_i is greater than the last item in s, it is guaranteed that no node in the tree rooted at c_i will contain items in the incoming itemset s. If $c_i \cap s = \emptyset$ and last item in c_i is greater than last item in s, that also guarantees no nodes in the subsequent trees possess non-empty interactions with s. However, if the first item of c_i is less than the last item of s, the subtree T_i with the root $[c_i, f(c_i)]$ may contain one or more items in s. The algorithm thus starts the search for rules in subtrees rooted at the children of c_i. If $c_i \cap s \neq \emptyset$, the intersection (say $r^{(a)}$) is a candidate for a rule antecedent. However, if the node $[c_i, f(c_i)]$ is not flagged (i.e. itemset c_i does not exist in the actual dataset), the candidate antecedent loses the candidacy. Now, the nodes in the subtrees starting from children of c_i posses equal or larger intersections with s. The algorithm thus

continues the search for rules in subtrees rooted at the children of c_i. If c_i is flagged, $f(c_i) - \sum frequency\ of\ children$ gives the number of occurrences of c_i in the dataset. $r^{(a)}$ then becomes a rule antecedent and each item in $c_i \setminus r^{(a)}$ becomes a consequent. These new rules of the form $r^{(a)} \Rightarrow I_j$ where $I_j \in (c_i \setminus r^{(a)})$ are added to the rule graph.

Algorithm 4 Algorithm that processes the itemset tree T and returns the rule graph G that predicts unseen items in a user-specified itemset s.

Let R denote the root of T and let $[\mathfrak{c}_i, f(\mathfrak{c}_i)]$ be R's children.
Let T_i denote the subtree whose root is $[\mathfrak{c}_i, f(\mathfrak{c}_i)]$.
To invoke *Rule_mining* use: G = *Rule_mining*$(s, T, \{\})$.

1: **Rule_mining**(s,T,G):
2: **for all** $\mathfrak{c}_i$ such that *first_item*($\mathfrak{c}_i$) $\leq$ *last_item(s)* **do**
3: **if** $s \cap \mathfrak{c}_i = \emptyset$ and *last_item*($\mathfrak{c}_i$) $<$ *last_tem(s)* **then**
4: $G \leftarrow$ *Rule_mining*(s, T_i, G);
5: **else if** $r^{(a)} \equiv s \cap \mathfrak{c}_i \neq \emptyset$ **then**
6: **if** $\mathfrak{c}_i$ is not flagged **then**
7: $G \leftarrow$ *Rule_mining*(s, T_i, G);
8: **else**
9: **if** $\mathfrak{c}_i$ does not have children **then** $f_o \leftarrow f(\mathfrak{c}_i)$;
10: **else** $f_o \leftarrow f(\mathfrak{c}_i) - \sum f(\mathfrak{c}_i$ *'s children*$)$; ▷ f_o is the frequency of $\mathfrak{c}_i$ in the database
11: **end if**
12: $G \leftarrow Update\_Graph(G, r^{(a)}, \mathfrak{c}_i, f_o)$;
13: $G \leftarrow$ *Rule_mining*(s, T_i, G);
14: **end if**
15: **end if**
16: **end for**
17: **return** G;

Each non-empty intersection of s with a flagged node of the tree generates set of association rules of the form $r^{(a)} \Rightarrow I_j$ where $r^{(a)}$ is the intersection of s with the node, and I_j is an item in node s.t. $I_i \notin r^{(a)}$. These rules are added to the rule graph using Algorithm 2(or algorithm 3).

Example 4 (A Rule Generation Example) *We consider the same dataset that was considered in the previous example: D={[1,4], [2,5], [1,2,3,4,5], [1,2,4], [2,5],*

[2,4]}. Assume the incoming itemset $s = [2, 3]$. *Fig. 3.3 shows the step by step building of the rule graph, according to the algorithm. The itemset s possesses non-empty intersections with 4* flagged *nodes,* $[1, 2, 3, 4, 5]$, $[1, 2, 4]$, $[2, 4]$, *and* $[2, 5]$. *A set of rules is added to the rule graph with each non-empty intersection.* ■

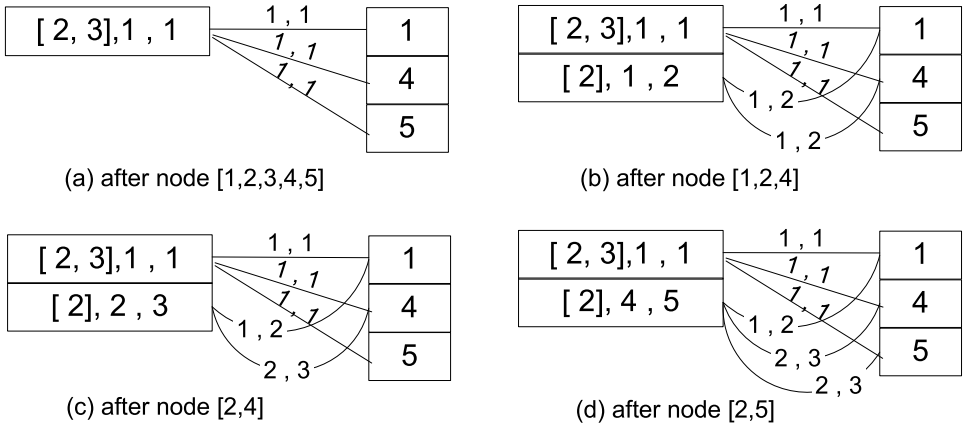


Figure 3.3: Rule graph construction for the testing itemset [2,3] using IT-tree in Fig. 2.1 (Testing itemset possesses non-empty intersections with only four nodes of the tree). 3.3(d) shows the final rule graph, G.

The ruleset that resides in G (figure 3.3(d)) is given in Table 3.4. The rule $[2, 3] \Rightarrow 1$ suggests that, if the itemset $[2, 3]$ is present in a shopping cart, item 1 is likely to be added to the cart. Support of this rule is 1/6 and the confidence is 1.

Table 3.4: The ruleset that resides in rule graph G (figure 3.3(d))

Rule	support	confidence
$[2, 3] \Rightarrow 1$	1/6	1
$[2, 3] \Rightarrow 4$	1/6	1
$[2, 3] \Rightarrow 5$	1/6	1
$[2] \Rightarrow 1$	2/6	2/5
$[2] \Rightarrow 4$	3/6	3/5
$[2] \Rightarrow 5$	3/6	3/5

Note that the ruleset in Table 3.4 consists of only two distinct antecedents, [2] and $[2, 3]$. Since no minimum support or confidence threshold is applied yet, one may expect another ruleset with the antecedent [3]. However, our algorithm does

not generate rules having antecedent [3]. This is not unintentional. Note that no transaction T_i in the dataset D provides an intersection $T_i \cap s \equiv [3]$, i.e., whenever item 3 appears in a transaction one or more of other items in s, also, happen to appear in T_i. So, item 3 alone does not provide us any additional evidence. Our rule generation algorithm, thus, ignores such rules.

It is important to note here that one might be interested in rules that suggest the absence of items, e.g., $[2, 3] \Rightarrow \langle 1 = absent \rangle$, i.e., when items 2 and 3 are already present in the cart, item 1 is unlikely to be added to the cart in the future. In such a case, the IT-tree has to be constructed considering $\langle item, value \rangle$ pair as an *item*. For instance, $\langle 1 = present \rangle$ is one item and $\langle 1 = absent \rangle$ is another item. The generated ruleset will eventually be composed of rules suggesting both the presence and absence of items. These rules could, then, be combined to reach a final decision.

Overlapping Rules

We noted that some rules have overlapping antecedents. For instance, $a \Rightarrow z$ and $a, b \Rightarrow z$ are two overlapping rules (i.e., antecedent of the first rule is a subset of the second rule). The process of updating the rule graph can be made faster if we can keep track of subset/superset relationship of the rule antecedent itemsets.

To ease the process of identification of overlapping rules, we made a simple, yet useful, change to the rule graph. We added a pointer to make each node in the rule antecedent list $\mathfrak{R}_a$ point to its super/sub set. Figure 3.4 shows the new rule graph.

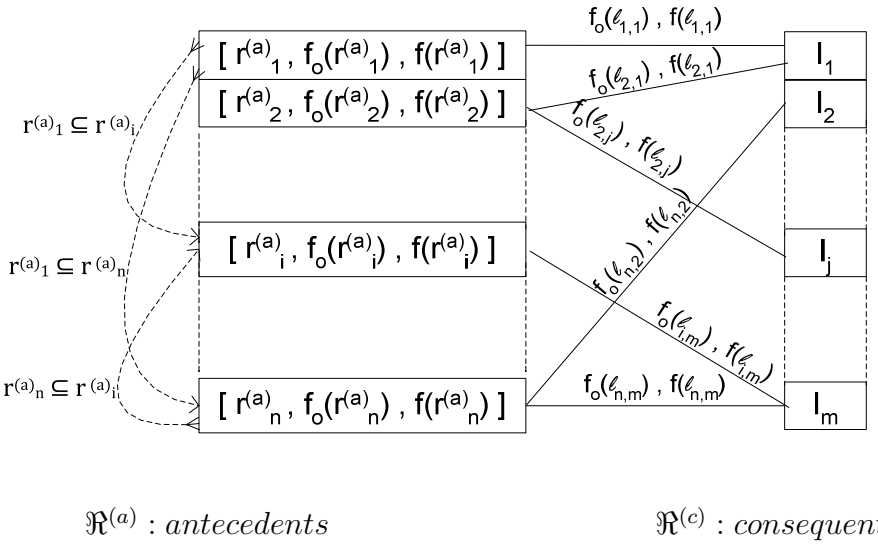


Figure 3.4: The Modified Rule Graph, G. Dotted links make it easy to find super/sub sets of items set $r_i^{(a)}$s

3.3 Employing Dempster-Shafer Theory

Suppose we want to establish whether the presence of items $i_1, \ldots, i_n$ increases the chance that also i_y is present. Many rules containing $i_1, \ldots, i_n$ in their antecedents may differ in their consequents—some of these consequents contain i_y, others do not. The question is how to quantify and combine the evidence behind all these rules. One way to handle this is to rely on the principles from the Dempster-Shafer theory of evidence combination. Let us now describe our technique that we refer to by the acronym DS-ARM predictor (Dempster-Shafer based Association Rule Mining).

Concrete Application to Our Task:

Basic Belief Assignment

In ARM approaches, a minimum support threshold is defined to select the high support rules. Once the rules are selected, they are all treated the same irrespective of how high or how low the support count is. Decisions are then made solely based

on the confidence value of the rule. A more intuitive approach would be to give a higher weight to those rules with higher support. Following this intuition, we propose a novel method to assign masses to the rules. Mass assignment is done considering both the confidence and support of the rule. However, the support value should have less impact on the mass.

In most practical situations, the training data set is highly skewed. For instance, in a supermarket scenario, the percentage of shopping carts containing a specific item, say `canned fish`, could be 5%. The remaining 95% of the shopping carts do not contain this item. Hence, the rules suggesting the presence of `canned fish` would have very low support while rules suggesting the absence of `canned fish` have a higher support. Unless specifically compensated for, a classifier or a predictor built from such a skewed training set typically tends to favor the 'majority' classes at the expense of 'minority' classes. In many scenarios, especially high risk domains, such a situation must be avoided at all costs because ignoring the minority class could have devastating consequences.

To account for this data set skewness, we propose to adopt a modified support value as follows:

Definition 3 (Partitioned-Support) *The* partitioned-support *p_supp of the rule,* $r^{(a)} \Rightarrow r^{(c)}$ *is the percentage of transactions that contain* $r^{(a)}$ *among those transactions that contain* $r^{(c)}$*, i.e.,*

$$p\_supp = \frac{support(r^{(a)} \cup r^{(c)})}{support(r^{(c)})}.$$

With Definition 3 in place, we take inspiration from the traditional F_α-measure [vR79] and use the weighted harmonic mean of support and confidence to assign the following BBA to the rule $r^{(a)} \Rightarrow i_j$.

$$m(r^{(c)}|r^{(a)}) = \begin{cases} \beta, & \text{for } i_j = present; \\ 1-\beta, & \text{for } i_j = \Theta; \\ 0, & \text{otherwise,} \end{cases} \tag{3.5}$$

where

$$\beta = \frac{(1+\alpha^2) \times conf \times p\_supp}{\alpha^2 \times conf + p\_supp}; \;\; \alpha \in [0,1]. \tag{3.6}$$

Note that, for the task at hand, $|i_j| = 1$ and hence, $\Theta = \{(i_j = present), (i_j = absent)\}$. Note that, as α decreases, the emphasis placed upon the partitioned-support measure in $m(\bullet)$ decreases as well.

With this mass allocation, the effectiveness of a rule is essentially tied to both its confidence and partitioned-support. Moreover, just as the F_α-measure enables one to 'trade' the precision and recall measures of an algorithm, the mass allocation above allows one to trade the effectiveness of the confidence and partitioned-support of a rule. Indeed, the parameter α can be thought of as a way to quantify the user's willingness to trade an increment in confidence for an equal loss in partitioned-support.

Discounting Factor

Following the work in [HPS07], the reliability of the evidence provided by each contributing BoE is addressed by incorporating the following discounting factor:

$$d = [1+Ent]^{-1}\left[1+\ln\left(n-|r^{(a)}|\right)\right]^{-1}, \text{ with}$$

$$Ent = -\sum_{i_j \subseteq \Theta} m(i_j|r^{(a)}) \ln [m(i_j|r^{(a)})]. \tag{3.7}$$

Recall that n denotes the number of items in the database. The term $1/(1+Ent)$ accounts for the *uncertainty* of the rule about its consequent. The term $1/(1+\ln[n-$

$|r^{(a)}|]$) accounts for the *non-specificity* in the rule antecedent. Note that d increases as *Ent* decreases and length of rule antecedent increases. As dictated by (2.7), the BBA then gets accordingly modified. The DRC is then used on the modified BoEs to combine the evidence.

Example 5 (Workout a Prediction Example) *Consider a supermarket dataset having five distinct items* {`egg, bread, butter, milk, wheat bread`}*. The training data set has details of six historic transactions. Given a partially observed transaction with* {`bread, milk`} *our task is to predict rest of the items that would be added to the cart.*

The training data set is shown in table 3.5. It is worth mentioning here, that the given data set is totally illustrative and may not reflect the actual behavior of a supermarket dataset. Let's use the integers from 1 – 5 to denote the presence of items and 6 – 10 to denote absence of items. For instance, $(egg = present) \equiv 1, (bread = present) \equiv 2, (egg = absent) \equiv 6$ etc. The converted dataset and the generated IT-tree are shown in table 3.6 and figure 3.5 respectively. Frequency count of each item is shown in table 3.7.

Table 3.5: The training dataset(illustrative)

Tx.#	Item set
1	bread, butter
2	egg, wheat bread
3	egg, bread, butter, milk
4	egg, break, milk
5	bread, butter
6	bread, milk

The rule graph generated for the incoming itemset {2,4} is given in figure 3.6. The rule set that resides in this graph is given in table 3.8.

Table 3.6: The dataset in the table 3.5 using integer notation

Tx.#	Item set
1	2,3,6,9,10
2	1,5,7,8,9
3	1,2,3,4,10
4	1,2,4,8,10
5	2,3,6,9,10
6	2,4,6,8,10

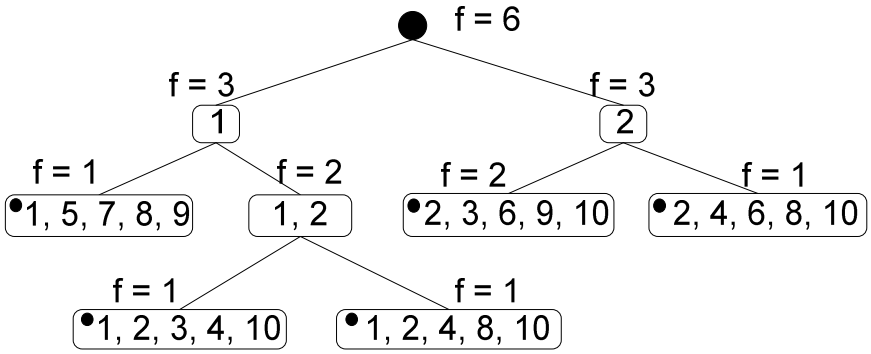


Figure 3.5: The itemset tree for the dataset in table 3.6

In this example (table 3.8, mass assignment is done using 3 times less weight for *support* compared to *confidence* (i.e., $\alpha = 0.33$). The last two columns show the computed BBA and d values of the rules. In the next step we eliminate the overlapping rules. For instance, rules 2 and 3 both suggest `'no egg'`, and the antecedent of the second rule is a subset of the first rule. However, rule 2 has lower confidence compared to rule 3. To make this illustration and computations simple, we ignore the less confident rule to make the rule set small and as independent as possible. If two overlapping rules have same confidence, the rule that has the lower support is dropped.

The rule set after pruning overlapping rules, is given in table 3.9. In order to reach a decision, rules are combined using Dempsters rule of combination. For instance, to predict if `eggs` are present or not, the first and second rules are combined. Using

Table 3.7: Frequency count for each item in data set in table 3.5

item	1	2	3	4	5	6	7	8	9	10
freq. count	3	5	3	3	5	3	1	3	3	5

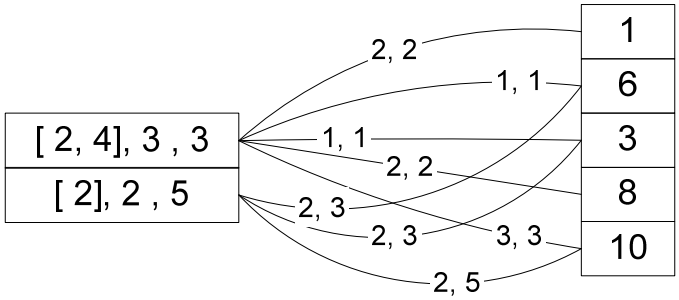


Figure 3.6: The rule graph for the partially observer shopping cart {2,4} and training dataset in table 3.6

pignistic probability, we can finally predict that only egg will be added to the cart (no butter, no wheat bread). However, the belief of prediction:'no wheat bread' is high ($BetP(no\ wheatbread) = 0.71$) compared to the other predictions ($BetP(egg) = BetP(no\ butter) = 0.52$).

3.4 Space and Time Complexity

It is important to analyze the space and time complexity of the proposed framework. Space complexity of the proposed framework is mainly governed by the space complexity of the itemset tree. Additionally, it takes space to store the rule graph. The "size" of an itemset tree is given by the number of nodes (including the root) and by the number of layers (root being the first layer). Processing a single transaction or market-basket will never give rise to more than two new nodes and, thus, cannot increase the depth of the tree by more than one layer. Let us use T to denote the itemset tree generated from a database of N distinct market baskets. The number of nodes in T is upper bounded by $2N$. The number of layers in T is upper bounded

Table 3.8: Rule Set that Resides in the Rule Graph G in Fig. 3.6

		support count					p-	$m(\bullet\|r_a)$		
	rule	ante.	rule	cons.	conf.	supp.	supp	proposition	BBA	d
1	$[2,4] \Rightarrow 1$	3	2	3	0.67	0.33	0.67	$\langle egg \rangle$	0.67	0.29
2	$[2,4] \Rightarrow 6$	3	1	3	0.33	0.17	0.33	$\langle No\ egg \rangle$	0.33	0.29
3	$[2] \Rightarrow 6$	5	3	3	0.60	0.50	1.00	$\langle No\ egg \rangle$	0.62	0.25
4	$[2,4] \Rightarrow 3$	3	1	3	0.33	0.17	0.33	$\langle butter \rangle$	0.33	0.29
5	$[2] \Rightarrow 3$	5	3	3	0.60	0.50	1.00	$\langle butter \rangle$	0.62	0.25
6	$[2,4] \Rightarrow 8$	3	2	3	0.67	0.33	0.67	$\langle No\ butter \rangle$	0.67	0.29
7	$[2,4] \Rightarrow 10$	3	3	5	1.00	0.50	0.60	$\langle No\ w.bread \rangle$	0.94	0.39
8	$[2] \Rightarrow 10$	5	5	5	1.00	0.83	1.00	$\langle No\ w.bread \rangle$	1.00	0.42

Table 3.9: Rule Set after pruning the overlapping rules in table 3.8

			$\hat{m}(\bullet\|r_a)$			combined BBA $M(\bullet\|r_a)$		
	rule	prediction	present	absent	Θ	present	absent	Θ
1	$[2,4] \Rightarrow 1$	egg	0.20	0.00	0.80	0.18	0.14	0.67
2	$[2] \Rightarrow 6$		0.00	0.16	0.84			
3	$[2] \Rightarrow 3$	butter	0.16	0.00	0.84	0.14	0.18	0.67
4	$[2,4] \Rightarrow 8$		0.00	0.20	0.80			
5	$[2] \Rightarrow 10$	w. bread	0.00	0.42	0.58	0.00	0.42	0.58

by N. Thus, the worst case space requirement of the resulting itemset tree is comparable with the size of the original database and has $O(N)$ space requirements. This is acceptable for many real-world datasets of interest, since, with the advancements in technology users have the sophistication of using larger memories. However, this would still be a bottleneck for larger datasets.

Apriory based association rule mining techniques require repeated parses over the dataset to count the frequent itemsets. If the number of distinct items in the dataset is m, there are 2^m frequent itemsets in worst case. The cost of frequent item generation for the apriory algorithm is $O(2^m \times N)$. Then to generate the association rules of the form $X \rightarrow Y; X \cap Y \neq \emptyset$ for these frequent itemsets, one needs to check

$supp(X \cup Y) \geq supp_threshold$ and $supp(X \cup Y)/supp(X) \geq conf_threshold$ for every frequent itemset pair X and Y. Since, in worst case, there are $(2^m - 1)2^{m-1}$ of such X, Y pairs, rule generation operation also takes considerable computational power.

DS-ARM significantly reduces these costs by relying on IT-trees; the whole database is permanently organized in a data structure that makes it possible to obtain relevant rules in time that does not grow faster than linearly in the number of transactions. Time complexity analysis, thus, can be divided into two parts—the time taken to build the itemset tree and time taken to generate the rule graph and to combine the rules. It has been shown by [KHR+03a] that the IT-tree does not grow prohibitively fast with size of the shopping carts. Also, the construction of the itemset tree has $O(N)$ space and time requirements. Maintenance of the IT-tree does not entail any additional costs compared to the case when the shopping carts are stored in a conventional transactional database. The only additional cost is the need to modify the IT-tree after the arrival of a new set of data; however, [KHR+03a] shows that this can be done very efficiently.

Once the IT-tree is generated only one scan of the IT-tree is required to generate the 'matching' ruleset for a given incoming itemset. For every non-empty intersection of the incoming itemset with an itemset in a 'flagged' tree-node, one parse over the rule graph is required to update the frequency counts. The maximum number of flagged nodes the IT-tree can have is upper bounded by N. Thus, in the worst case it requires N parses over the rule graph. The rule graph starts to grow from 0 and length is upper bounded again by N ($O(N)$ space complexity). Even though this results in a worst case time complexity of $O(N^2)$ the average time complexity is much less in real-world situations.

CHAPTER 4

Empirical Evaluation of DS-ARM on "Crisp" Data

In our experiments, we relied on two sources of data. First, we followed the common practice of experimenting with the synthetic data obtained from the IBM-generator[1]. This gave us the chance to control critical data parameters and to explore our programs under diverse circumstances. Moreover, relying on well-known software available on the web, we made it easy for other scientists to replicate our experiments.

The generator employs various user-set parameters to create an initial set of frequent itemsets whose sizes are obtained by a random sampling of the Poisson distribution. Then, transactions ("shopping carts") are created in a manner that guarantees that they contain these itemsets or their fractions. The lengths of the transactions are established by a random number generator that follows the Poisson distribution. For a more detailed description, see [AS94].

Table 4.1 summarizes the user-set parameters. In the data we worked with, we varied the average transaction length and the average size of the "artificially added" itemsets. The specific values of these parameters are indicated in the names of the generated files. For instance, T10.I4 means that the average transaction contained 10 items, and that the artificially implanted itemsets contained 4 items on average.

[1] http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html

Table 4.1: IBM-Generator Parameters

Parameter	Description
$\|D\|$	Number of transactions
$\|T\|$	Average size of the transactions
$\|I\|$	Average size of the maximal potentially large itemsets
$\|L\|$	Number of maximal potentially large itemsets
N	Number of items

Apart from using the IBM generator, we experimented with two benchmark domains from the UCI repository.[2] which is popular in machine learning research. These, too, are broadly known, and their characteristics are well understood by the community. To be more specific, we used the congressional-vote domain and the SPECT-Heart domain.

The *congressional vote* dataset has the form of a table where each row represents one congressman or congresswoman, and each column represents one bill. The individual fields contain "1" if the person voted in favor of the bill, "0" if he or she voted against, and "?" otherwise. We numbered the bills sequentially as they appear in the table (from left to right), and then converted the data by creating for each politician a "shopping cart" that contains the numbers of those (and only those) bills that he or she voted for. For instance, the shopping cart containing $[1, 4, 8]$ indicates that the politician has voted for bills represented by the first, fourth, and eighth column in the table. In our experiments, we ignored the information about party affiliations (the class label in the original data).

Our research questions can be formulated as follows: being told about a subset of a politician's voting history, can we tell (ignoring party affiliation) how he or she has

[2]http://archive.ics.uci.edu/ml/

voted on the other bills? How much does the correctness of these predictions depend on the number of known voting decisions? How will alternative algorithms handle this task?

To find out, we removed, in the shopping-cart version of the data, a certain percentage of items. For instance, the original cart containing $[1, 3, 12, 25, 26]$ may now contain only $[3, 25, 26]$. From this incomplete information, we induced the vote predictor and measured its accuracy against the known values of the votes. This means that, in this particular cart, we would ideally want our program to tell us that the politician has also voted for bills 1 and 12. The evaluation of such predictions of course calls for appropriate performance criteria—these will be discussed in the next section.

Apart from the congressional vote, we used another binary dataset, the SPECT Heart domain, where 267 instances are characterized by 23 boolean attributes. Again, we converted these data into the shopping-carts paradigm, following the same methodology as described above, seeking to predict items that have been hidden.

4.1 Performance Criteria

Literature on traditional classification tasks often measures a classifier's performance in terms of the percentage of correctly predicted classes, or, conversely, as percentage of incorrect classifications. In our domains, though, this criterion is not so appropriate. For one thing, we need to evaluate the prediction accuracy in a situation where several "class attributes" have to be predicted at the same time. For another, plain error-rate related criteria fail to characterize the predictor's performance in terms of the two essential types of error: "false negative," where the

predictor incorrectly labels a positive example of the class as negative, and a "false positive," where the predictor incorrectly labels a negative example as positive. For these reasons, we preferred to rely on the *precision* and *recall* criteria borrowed from the information-retrieval literature.

To start with, suppose, for simplicity, that we are interested only in one specific class. Let us denote by TP the number of true positives (correctly labeled positive instances); by FN the number of false positives; by FP the number of false positives; and by TN the number of true negatives (correctly labeled negative instances). These quantities are used to define *precision* (Pr) and *recall* (Re) in the following way:

$$Pr = \frac{TP}{TP + FP}, \quad Re = \frac{TP}{TP + FN}$$

For the needs of combining these two in a single metric, [VCS04] proposed a so-called F_1-measure:

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re}$$

Thus armed, we can now proceed to the multi-class case where each example can belong to two or more classes at the same time, and we thus need to average the prediction performance over all classes. To this end, [GS04] proposed two alternative approaches: 1) *macro averaging*, where the above metrics are computed for each item and then averaged; and 2) *micro averaging*, where they are computed by summing over all individual decisions. The requisite formulas are summarized by Table 4.2 where Pr_i and Re_i stand for the precision and recall as measured on item i. TP_i, TN_i, FP_i, and FN_i denote the values of the four basic variables for the item i.

Table 4.2: Macro averaging and micro averaging of precision and recall. N denotes the number of different classes (items).

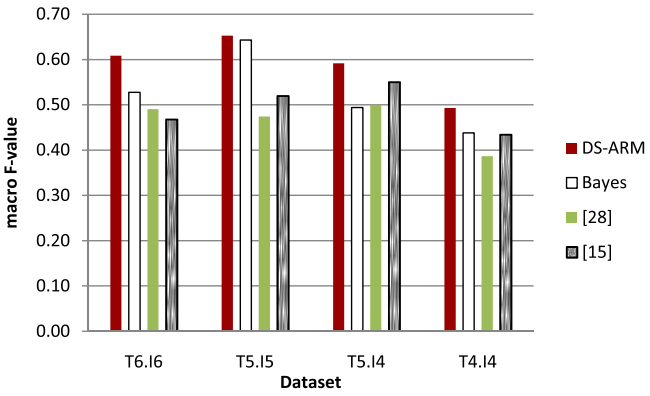
	Macro (M)	Micro (μ)
Precision	$\frac{\sum_{i=1}^{N} Pr_i}{N}$	$\frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i)}$
Recall	$\frac{\sum_{i=1}^{N} Re_i}{N}$	$\frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FN_i)}$
F_1	$\frac{2 \times Pr^M \times Re^M}{Pr^M + Re^M}$	$\frac{2 \times Pr^\mu \times Re^\mu}{Pr^\mu + Re^\mu}$

4.2 Experiments

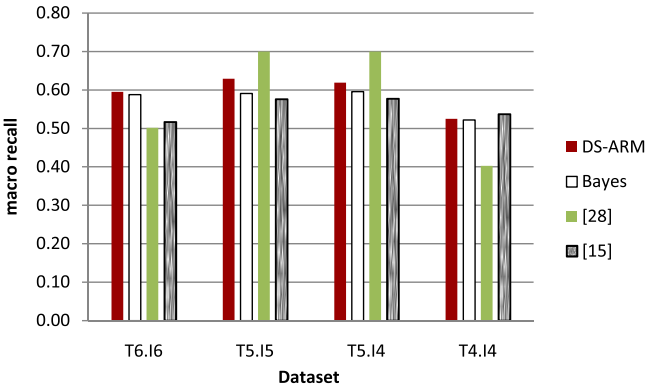
4.2.1 Experiment 1: Simple Synthetic Data

The first obvious question is how the performance of the technique DS-ARM, proposed in Section 3.3, compares to that of Bayesian classifiers and to that of the older techniques from [HPS07, LHM98]. For the sake of fairness, we have to respect that the older techniques (with the exception of Bayes) were developed for domains with small numbers of items and with only one "class attribute." This is why we generated, for the first round of experiments, synthetic domains with only 10 distinct items. We used transaction lengths from 4 to 6 and equally sized artificial itemsets: (T6.I6), (T5.I5), (T5.I4), and (T4.I4). Note that these domains are simple enough for all the algorithms to be computationally affordable.

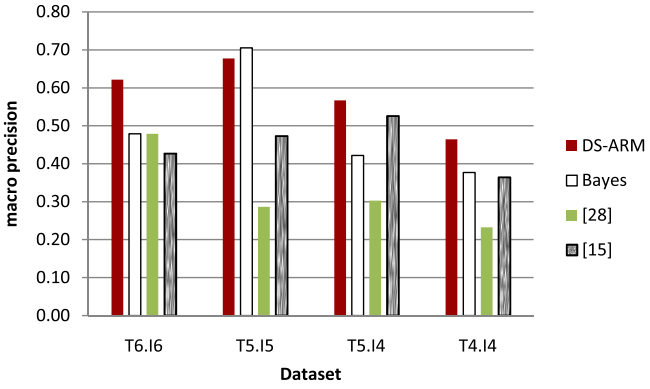
In each of these synthetic domains, we prepared the prediction task in the following manner. For each item in each shopping cart, we generated a random number (uniform distribution) from the interval $[0, 1]$. If the number was greater than 0.7, we removed the item from the shopping cart. In other words, each item had 30% chance of being removed. From these incomplete shopping carts, we induced association rules to be used to "guess" which items we removed. We evaluated the prediction performance by comparing this prediction with the known lists of removed items.



(a) Macro F-value.



(b) Macro recall.



(c) Macro precision.

Figure 4.1: Macro F-value, macro recall, and macro precision of missing-item prediction in the synthetic domains.

The results are summarized in Figures 4.1(a), 4.1(b), and 4.1(c) in terms of macro recall, precision, and F_1 (for the micro versions, the results were similar). The reader can see that, in terms of macro F_1, our technique rather convincingly outperformed the other techniques in each domain. A closer look at the bar charts reveals that its performance edge is particularly well pronounced in precision, whereas recall is not much better that of the other techniques. We regarded these results as encouraging in view of the fact that we used domains that were intentionally made suitable for the other techniques.

4.2.2 Experiment 2: UCI Benchmark Domains

In the next experiment, we asked whether DS-ARM would fare equally well in more realistic domains. It should be noted that, here, the shopping carts contained on average many more items than in the synthetic domains from Experiment 1; this rendered the older two DST-based techniques computationally so inefficient that we decided to compare DS-ARM only with the Bayesian classifier. We wanted to ascertain whether the latter would "beat" DS-ARM at least along some criteria; also, we wanted to know how the prediction performance depended on the amount of available information (or, conversely, on how many of the items in the shopping carts have been removed).

For the congressional vote domain, the results are summarized in the graphs in Figures 4.2 and 4.3, and for the SPECT Heart domain, the results are summarized Figures 4.4 and 4.5. The graphs show the average values of the three fold cross validation. In both domains, whether along the macro metrics or the micro metrics, we always observed the same behavior. First, DS-ARM outperforms the Bayesian approach along the more general F_1 metric. Second, DS-ARM has almost perfect

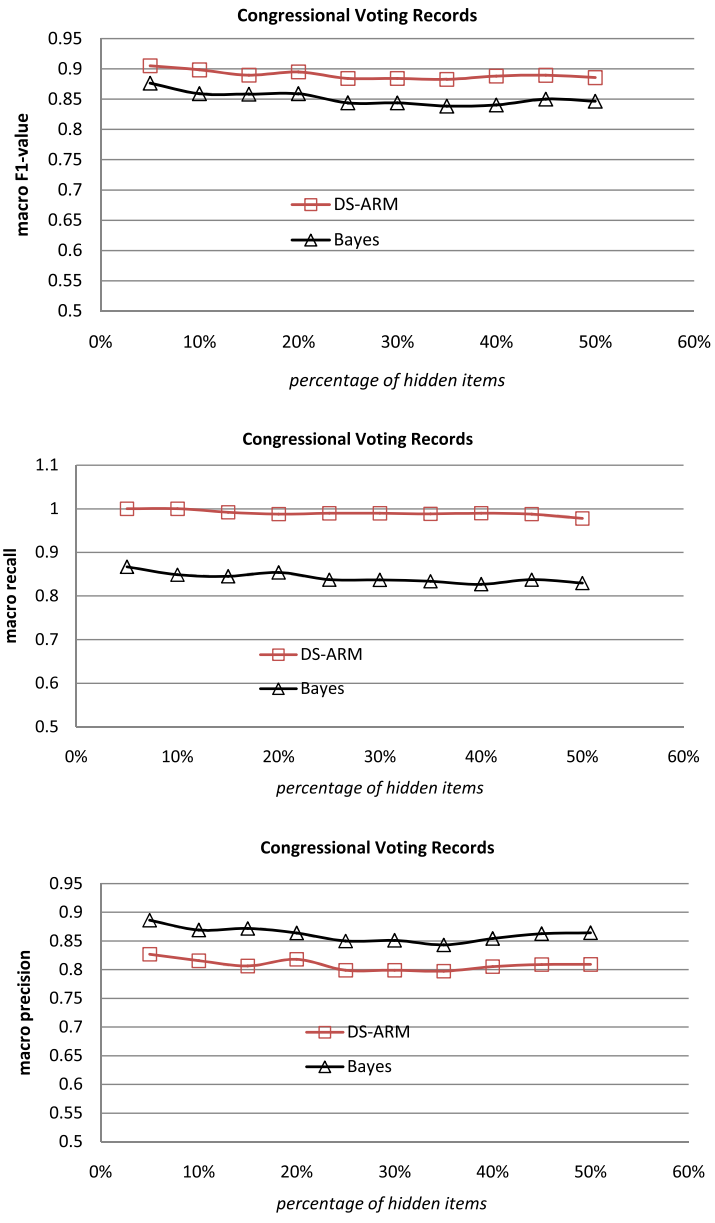


Figure 4.2: "Macro" performance in the congressional-vote domain.

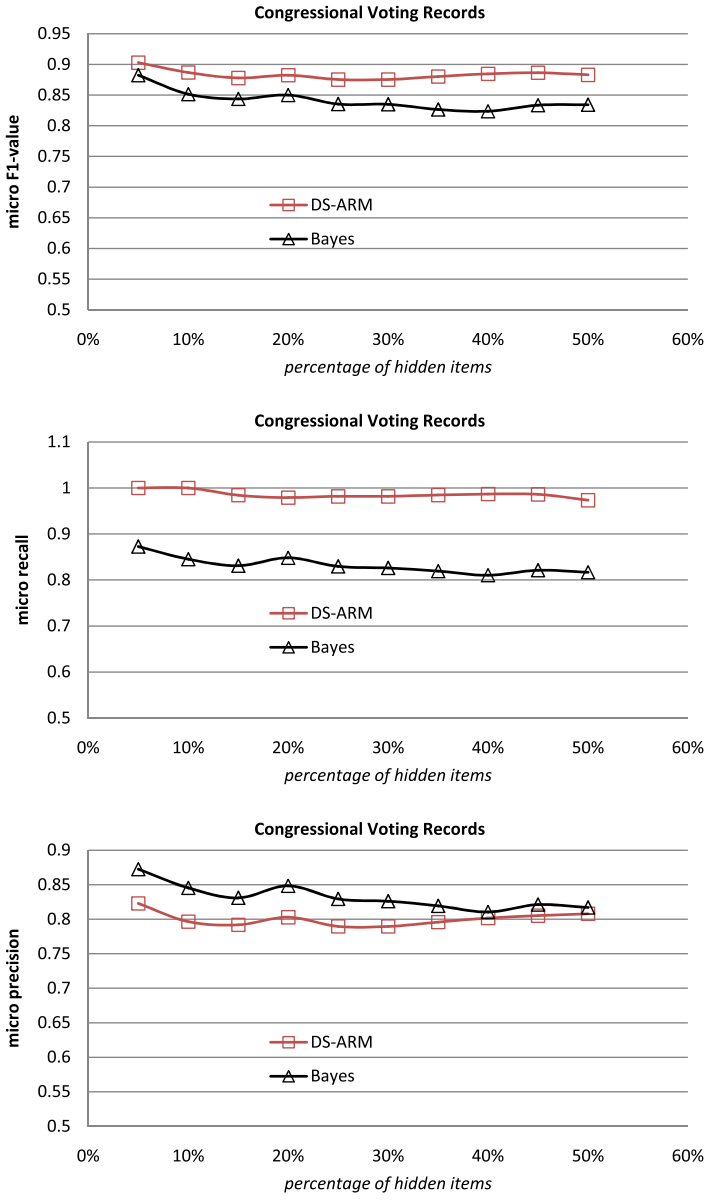


Figure 4.3: "Micro" performance in the congressional-vote domain.

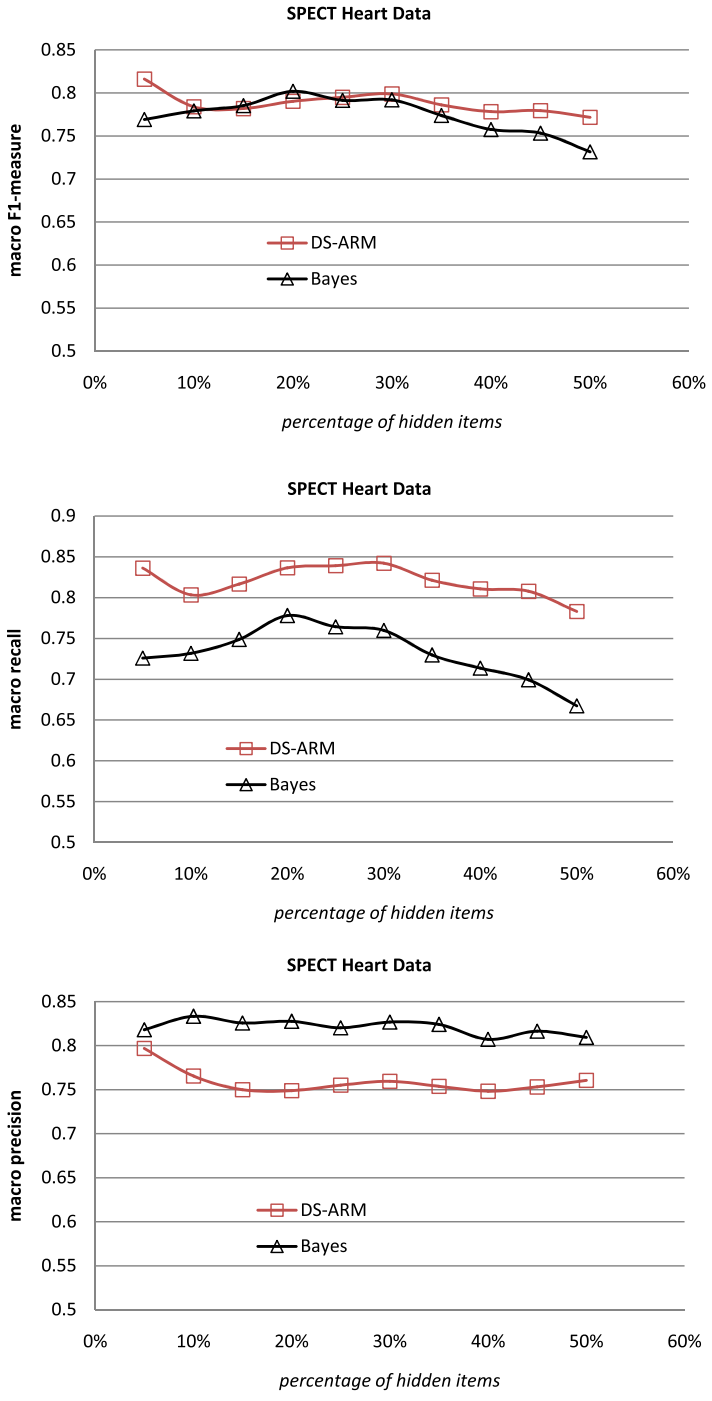


Figure 4.4: "Macro" performance in the SPECT Heart domain.

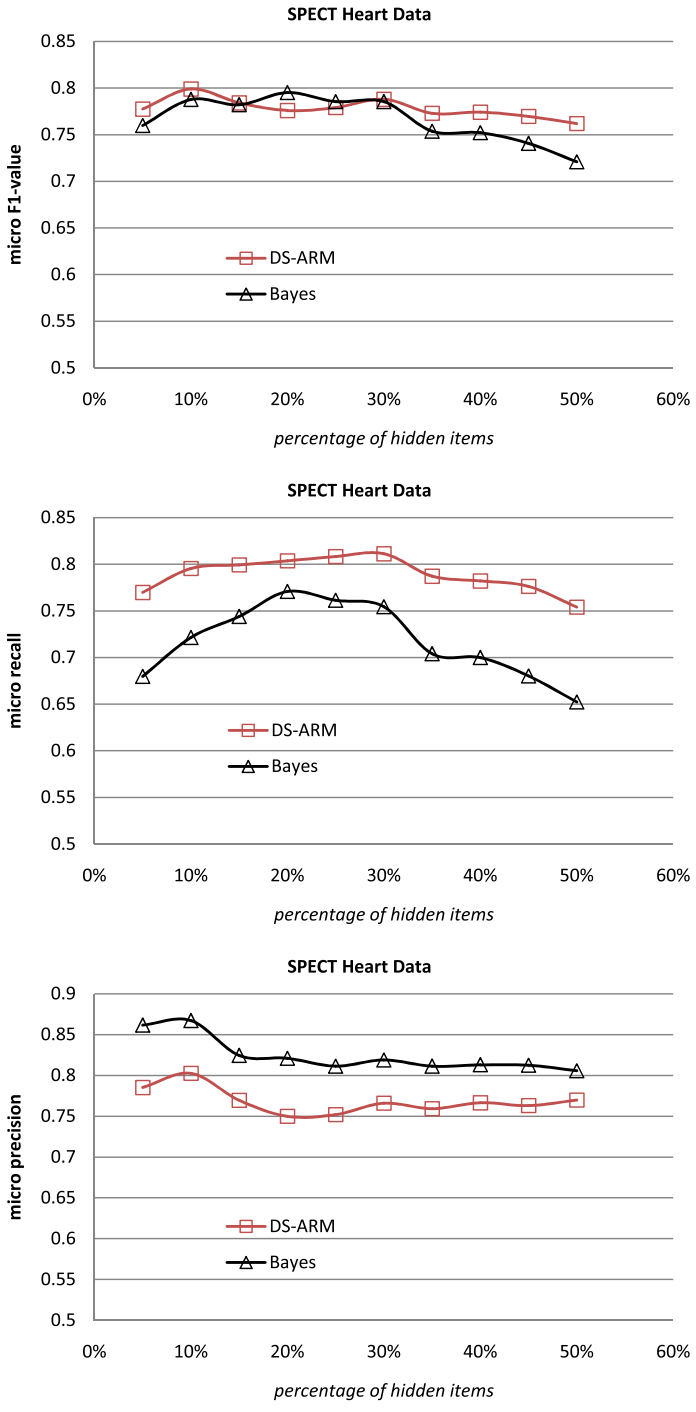


Figure 4.5: "Micro" performance in the SPECT Heart domain.

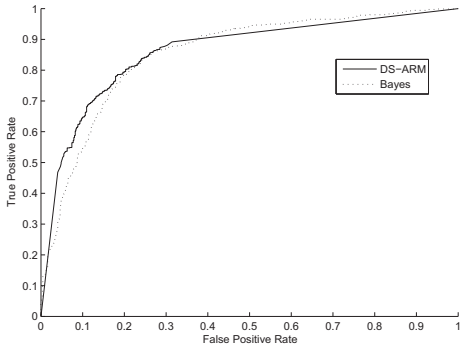
recall. This means, for instance, that if a politician voted for a certain bill, the system will almost always correctly predict this vote based on the available voting record. Third, the Bayesian approach has a slightly better precision—which means, for instance, that when the system says a politician voted for a bill, the Bayesian approach is somewhat less frequently mistaken. Fourth, the prediction performance dropped only very slightly even when as many as 50% of the items in the shopping carts were removed. Fifth, the behavior was about the same whether the macro or micro criteria were used (this was perhaps due to the relative balance in the representation of 1s and 0s in these two domains).

4.2.3 Experiment 3: ROC Curves

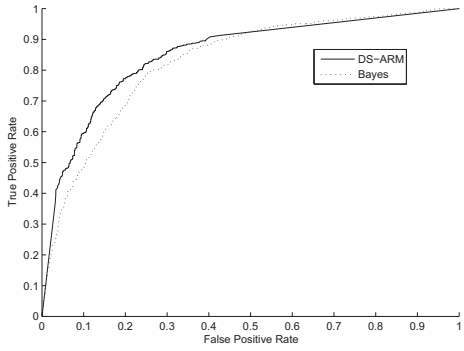
In previous experiments, we chose a "cutpoint" above which we considered the item to be present and below which we considered the item absent. The position of the cutpoint will determine the number of true positive, true negatives, false positives and false negatives. We may wish to use different cutpoints for different situations if we wish to minimize one of the erroneous types of test results.

Receiver operating characteristic (ROC) curves were initially developed in the 1950s for signal detection theory to analyze noisy signals. ROC curves characterize the trade-off between positive hits and false alarms. ROC curves plot true positive rate (y-axis) against false positive rate (x-axis). Performance of the predictor/classifier is represented as a point on the ROC curve. Changing the threshold or the cutpoint changes the location of the point.

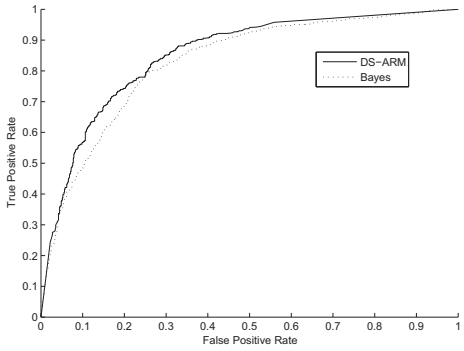
True positive rate is defined by $TP/(TP + FN)$ and the false positive rate by $FP/(FP + TN)$. A diagonal line in the ROC curve indicates random guessing. The ROC curve of a good predictor should be well above the diagonal line. In an ideal case, the area under the curve is equal to 1.



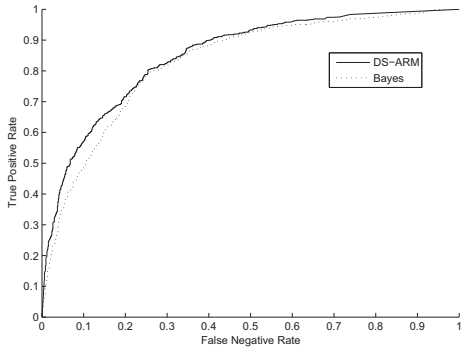
(a) 5% items hidden.



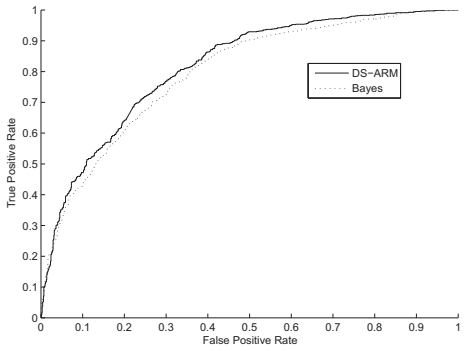
(b) 15% items hidden.



(c) 30% items hidden.



(d) 45% items hidden.



(e) 60% items hidden.

Figure 4.6: ROC curves of DS-ARM and Bayes predictors on congressional voting dataset

Figure 4.6 shows sample ROC curves for the DS-ARM and Bayes predictors on the congressional voting datasets with different percentages of hidden items. Plots are done for only one randomly selected cross validation dataset. The DS-ARM ROC curve lies above the Bayes curve in most cases.

4.2.4 Experiment 4: DS-ARM Predictor for Classification

The predictor can easily be used for classification tasks. In that case, the item to predict is the class label. We used the SPECT Heart Data Set from the UCI data repository. The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The data set has 267 instances that are described by 23 binary attributes. It is already divided into training and testing. The training dataset has 80 instances, out of which 40 instances are categorized as abnormal and 40 are normal. The testing dataset has 187 instances out of which 172 instances are from class 1 and 15 from class 0. Authors in [KCT+01] have reported that their CLIP3 and CLIP4 machine learning algorithms have achieved 84.0% and 86.1% accuracy, respectively. Table 4.3 shows the comparison of results. We were unable to run classifiers proposed in [HPS07] and [ZSP+04a] since they were unable to handle the number of attributes. Our predictor was able to make 170 *hard-predictions* and 159 of those predictions were correct. The rest of the 17 testing instances were classified based on the pignistic probability. For the results shown in the table we computed the pignistic probability of each class, and the testing instance is assigned to the class that has the higher pignistic probability. Experiment results indicate that the DS-ARM predictor can comfortably outperform the other classifiers.

Table 4.3: Classification Accuracy on SPECT Dataset

	CLIP3	CLIP4	Bayes	Proposed DS-ARM
Accuracy	84.0%	86.1%	74.9%	89.8%

4.2.5 Experiment 5: Computational Costs on Larger Data

In domains of more realistic size, and more realistic numbers of distinct items, earlier techniques are too expensive due to the costs associated with the need to generate *all* association rules with the given antecedents. DS-ARM significantly reduces these costs by relying on the recently proposed technique of IT-trees; the whole database is permanently organized in a data structure that makes it possible to obtain relevant rules in time that do not grow faster than linearly in the number of transactions, and has been shown by [KHR+03a] not to grow prohibitively fast with size of the shopping carts. The maintenance of the IT-tree does not entail any additional costs compared to the case when the shopping carts are stored in a conventional transactional database—the only additional cost is the need to modify the IT-tree after the arrival of a new set of data. However, [KHR+03a] shows that this can be done very efficiently and—importantly—off-line.

Our next experiment investigates the computational costs of DS-ARM on a few synthetic datasets with the same parameter settings as in [AS94]. We considered $|D| = 10,000$ shopping carts, $N = 100$ distinct items, instructing the data generator that the number of frequent itemsets should be about $|L| = 2,000$. We generated three datasets differing in the average size of the shopping cart and in the average length of the frequent itemsets as summarized by Table 4.4.

The earlier algorithms are all very sensitive to the value of the user-set minimum support—as its value decreases, the number of frequent itemsets from which the classification rules are obtained grows very fast; this adds to the computations needed

Table 4.4: Parameter Settings

Dataset	T5.I2	T10.I2	T10.I4
$\|T\|$	5	10	10
$\|I\|$	2	2	4

to select the matching rules and to combine them. Figure 4.7 illustrates, for three different synthetic domains, the situation in the case of DS-ARM. Since the same IT-tree is always used, it is fair to ignore the costs of its building, particularly so because it can be built off-line. The vertical axis shows the time needed to complete all 1,000 incomplete shopping carts in the case where 20% of the shopping-cart contents were hidden. The measured time includes the costs of finding all relevant rules for the given incomplete cart, and of combining them when making the final decision about which items to predict. The time is plotted against the varying values of the minimum support. The reader can see that, expectedly, the computation time significantly drops with growing minimum-support threshold, which implies the obvious trade-off between completeness and computational demands.

Figure 4.8 then shows how DS-ARM scales up as the average transaction length is increased. For this experiment, the number of distinct items was 100, and the number of shopping carts in the dataset was 1,000. When generating the data, we used the following parameter settings: T5.I2, T10.I2, T15.I2, and T20.I2. The costs grow very rapidly with transaction lengths. This is to be attributed to the fact that increasing size of shopping carts means that only a small portion of all shopping carts will have empty intersections; the number of generated rules then grows exponentially. Still, we deem the costs affordable in view of the fact that many real-world applications rarely have more than a few dozen items in each shopping cart.

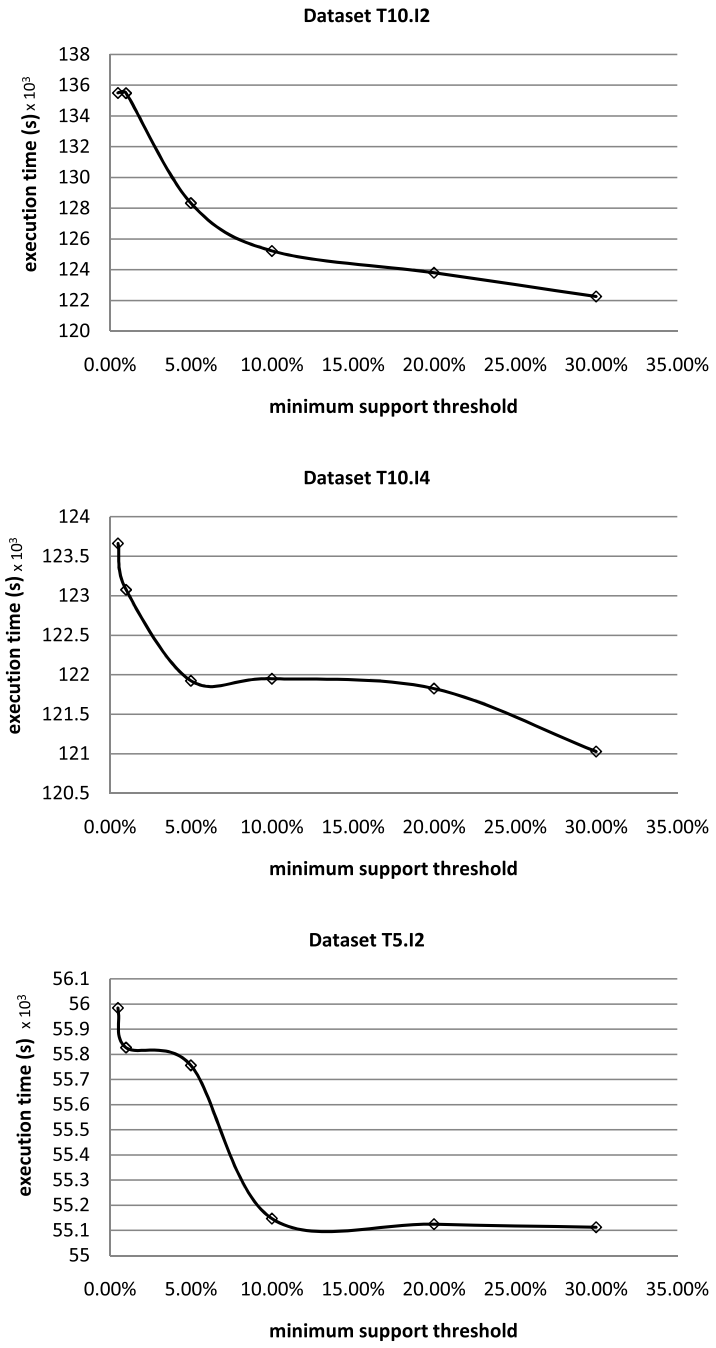


Figure 4.7: Execution time vs. minimum support: Even though the rule generation algorithm is not sensitive to minimum support threshold, rule combination costs reduce with increasing minimum support due to reduction in number of rules

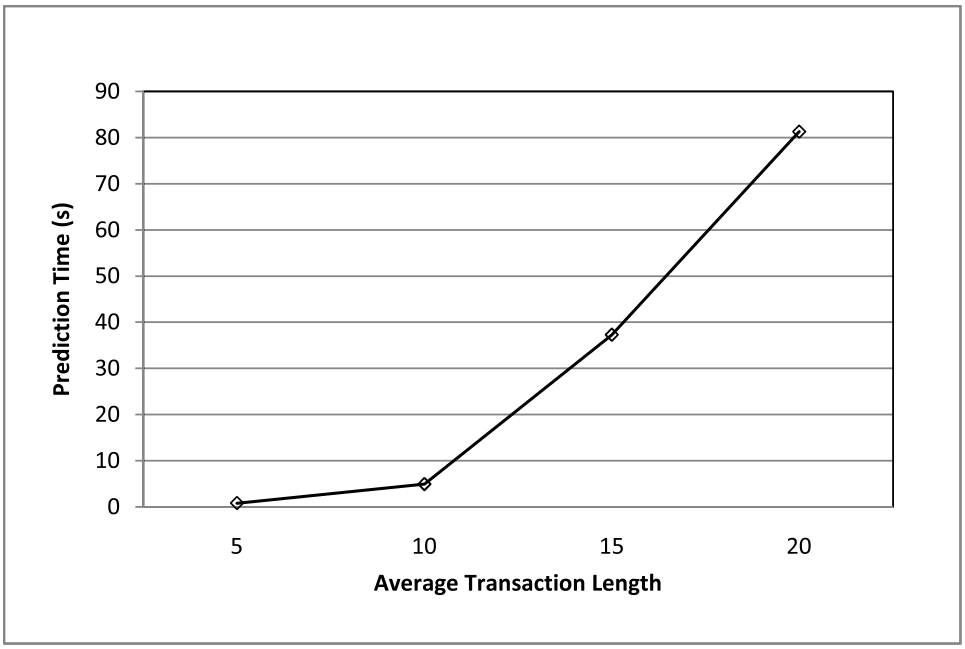


Figure 4.8: Average time per prediction vs. average transaction length

4.2.6 Experiment 6: Impact of the Number of Distinct Items

Finally, we wanted to know how DS-ARM would react to the growing number of distinct items. This was the task of the next round of experiments. Here, we fixed the number of shopping carts at 10,000, and generated synthetic data with the following parameter settings: T5.I2, T10.I4, and T20.I6. For the relatively low (and, hence, expensive) minimum support of 1%, and for varying the number of distinct items from 100 to 1,000, we obtained the results shown in Figure 4.9. The curves indicate that, as the number of items increases, DS-ARM computational costs grow exponentially. This tells us that, although this system can handle more "difficult" domains than the older approaches, much more work remains to be done.

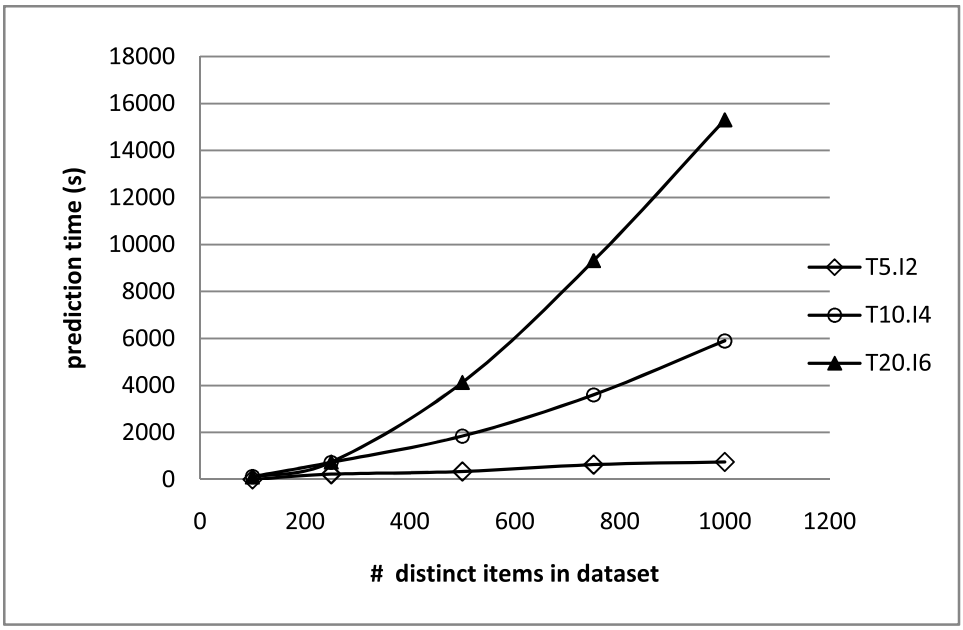


Figure 4.9: Average time per prediction vs. the number of items in three synthetic domains.

4.2.7 Experiment 7: Comparison of Computational Costs with Bayes Method

So far, we were testing the DS-ARM on 'crisp' datasets. We expected DS-ARM results to be at least comparable with other existing methods on 'crisp' datasets. Results were more than promising. However, the main objective of developing DS-ARM was to come-up with a knowledge discovery framework that is capable of effectively accommodating and accounting for imperfections in data. The DS-theoretic framework that we used to model the data imperfections, however, resulted in higher computational costs. The capability of handling data imperfections is, thus, accomplished at the cost of computational burden. To complete this study, we felt the need to compare the computational costs of DS-ARM with the Bayes method. Here, we fixed the number of shopping carts to 10,000, and generated synthetic data with parameter settings for average transaction length 5 and the average length of maximal potentially large itemsets 4 (T5.I4). The number of distinct items was varied from 100 to

Table 4.5: Comparison of computation time between Bayes and DS-ARM: Average time take per prediction (values are given in seconds)

	number of distinct items				
Predictor	100	250	500	750	1000
Bayes	0.02	0.06	0.16	0.22	0.30
DS-ARM	1.01	4.21	19.20	38.01	59.61

1000. We employed both Bayes and DS-ARM to make 1000 predictions. A relatively low minimum support of 1%, was used for DS-ARM. The average time to make a prediction was computed and the obtained results are shown in Table 4.5 and Figure 4.10. The computation time taken by both predictors rapidly increases with the number of distinct items. However, the time taken by Bayes predictor is much less than the DS-ARM.

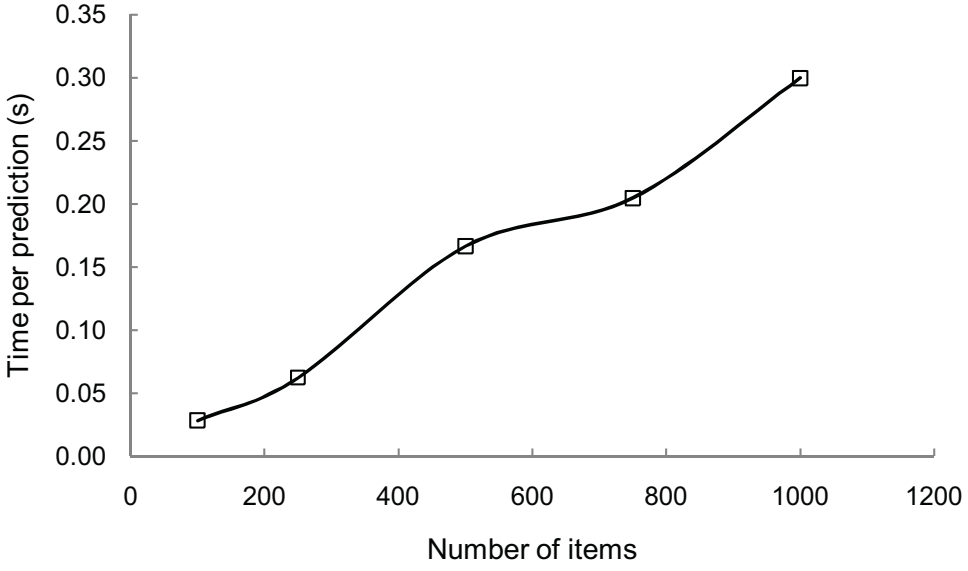


Figure 4.10: Average time per prediction of Bayes predictor vs. the number of items dataset.

CHAPTER 5

Representation of Imperfect Data

So far we have been working with "crisp" datasets: the observed data vales are considered as hundred percent reliable. However, the prime objective of developing DS-ARM framework is to work with imperfect data sets in which the probabilistic approaches fail to yield effective results.

The probabilistic approach for modeling data imperfections does not have provisions to address issues like (i) reliability in data where the observed values are associated with uncertainty factors, e.g., value of the variable X is x_1 with a reliability of 0.7; and (ii) ambiguity where a single value cannot be discerned for a particular variable but only a composite value can be allocated, e.g., the value of X could be one of $\{x_1, x_2\}$ with a certainty of 0.7 and no other information is available to discern between the two values. Probabilistic approaches need to make various assumptions in such instances to match the data to the probability model. For example, in (i), 0.3 is distributed equally or according to some distribution among other variables; and in (ii), 0.7 is divided equally (assuming equi-probability) between x_1 and x_2.

In this chapter, we discuss how we use DS evidence theory to model such situations giving specific examples from a real-world medical dataset.

5.1 Attribute Value Ambiguities

Suppose we have a database of customer ratings of a line of products. In most practical situations users are allowed to pick, say, one out of five rating values. However, there are certain situations where you simply cannot pick a single value. It may be due to lack of evidence or other inevitable uncertainties associated with the rating process. Such difficulties are so common in medical applications such as rating drug responses, or severity of a disease. If the severity is quantified, say, by a value from $\Theta = \{Critical, Medium, Normal\}$, when assigning the value, a physician relies on his/her experience and/or the experience of colleagues. Such ratings are inevitably ambiguous and not easily expressed in terms of probabilities. For instance, it would be mistaken to assume that the statement, "the symptom is *Critical* with a 70% confidence" implies a 30% confidence in the complement of *Critical*.

A Specific Problem: Rating HAART Regime Responses

Highly active antiretroviral therapy (HAART) is used for the treatment of HIV patients. HAART will not eradicate HIV, and the current goal of therapy is to inhibit viral replication over a long-term period so that immune responses to most common pathogens are restored. HAART has dramatically improved the prognosis of patients with HIV. The initial virological response to HAART, by reducing viral load to below the limit of detection, is essential for reducing the risk of drug resistance, which in the longer term may lead to a deterioration in immune function and an increased risk of clinical disease progression. There are a number of problems associated with HAART, including the development of drug resistance, the difficulty of maintaining long-term adherence, and drug-related toxicities, all of which may lead to virological failure, which in turn leads to immunological failure and clinical progression.

In general, patients who start HAART have a rapid decrease in HIV viraemia to undetectable levels within months of starting HAART and a gradual increase in CD4 cell count to levels approaching those seen among uninfected patients. There are several factors commonly reported to be associated with initial virological response to HAART. One of the most important factors related to virological response to HAART is prior treatment; treatment-experienced patients have a poorer response to HAART and are less likely to achieve a viral load of below the limit of detection. Treatment-experienced patients may have accumulated drug resistance, which may result in a poorer virological response. Adherence also plays a key role in virological response. Results from both clinical trials and observational studies report an increase in virological response amongst patients who are more adherent, although it is often difficult to capture accurate data on adherence. Failure to adhere to HAART results in low drug levels, which can rapidly lead to the selection of virus with decreased susceptibility. In addition, as different antiretrovirals within the same drug-class are cross-resistant, the number of potential regimens rapidly decreases for the non-adherent patient.

It takes longer for patients with a high viral load when starting HAART to reduce their viral load to below the limit of detection; some studies also suggest that patients with a higher viral load when starting HAART are less likely to respond with a viral load below the limit of detection. Thus, one log_{10} drop of the viral load count is considered an excellent response in such cases. Assessing the reasons for a lack of early response to HAART and addressing problems probably by change in treatment may improve the clinical outcome. The physician, thus, has to switch the HAART regimen and pick a proper new regimen to treat the patient. In doing so, the physician assess the responsiveness of the patient for earlier treatments. Previous treatments have to

be deeply analyzed since some of the antiretrovirals are cross-resistant. Under these circumstances, a decision support system advising the physician of what HAART regimens might work on this patient would help the physician to make a decision.

Assessing the responsiveness of a HAART regimen can be done by looking at the viral load or CD4 count of the patient. Figure 5.1 shows arbitrary viral load traces for three patients. The x-axis shows the number of weeks after the start of HAART regimen and the y-axis shows the viral load/count. By looking at the viral load trace an expert may be able to give a rating to the virological response.

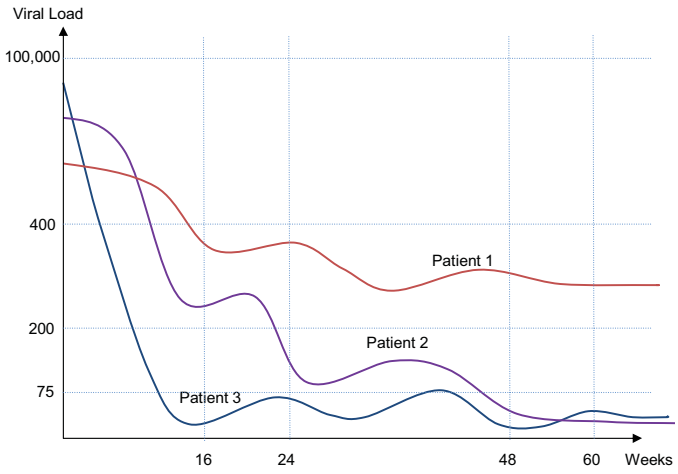


Figure 5.1: Arbitrary viral load traces

Rating the HAART Responsiveness and Differences in Opinion

Initial virological response, the tail of the viral load trace and how fast the viral count become undetectable are the main factors to consider in assigning the rating. Different experts have a different opinion on the relative importance of each of these factors.

We defined three experts to assign ratings to the virological response. Each individual expert assigns a rating based on only one of the above three factors. Rating values are drawn from the set $\{Excellent, Good, Fair, Poor\}$.

- Expert 1 (profile expert): Analyze the viral load variation within 60 weeks of HAART start date.
- Expert 2 (initial gradient): Assign a rating based on the gradient of viral load drop within the first 16 weeks of treatment.
- Expert 3 (tail of viral load curve): Rating is assigned based on the final viral load level at the end of 60 weeks of treatment.

The expert one, who analyzes the whole viral load profile, makes the decision based on the viral load profiles shown in the Figure 5.2.

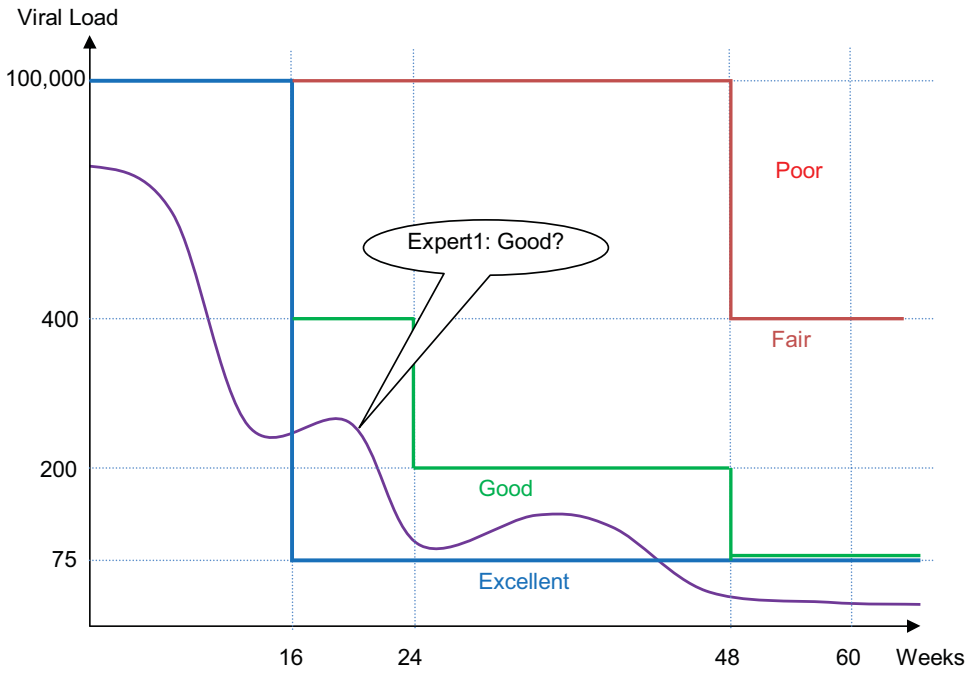


Figure 5.2: Viral load variation profiles

Lack of Evidence and Rating Assignment

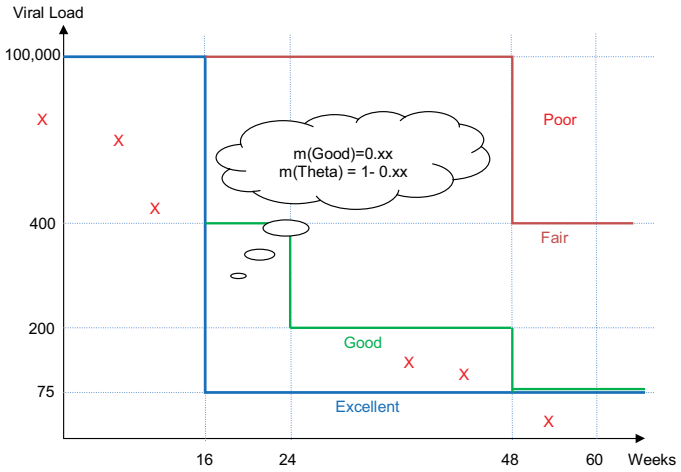
The viral load of the patient is taken from the available lab test results. Even though HIV patients often check their viral count, it is quite common to find long periods where no lab tests are done. When the viral counts are not available it

becomes difficult to assess the virological response. Thus, "crisp" ratings are not expected from the experts. Whenever there is lack of evidence or if the available evidence does not support a specific rating the expert may give a rating as a BBA on the frame of discernment. Figure 5.3, shows examples for lack of evidence. The red crosses indicate the observed vial load counts.

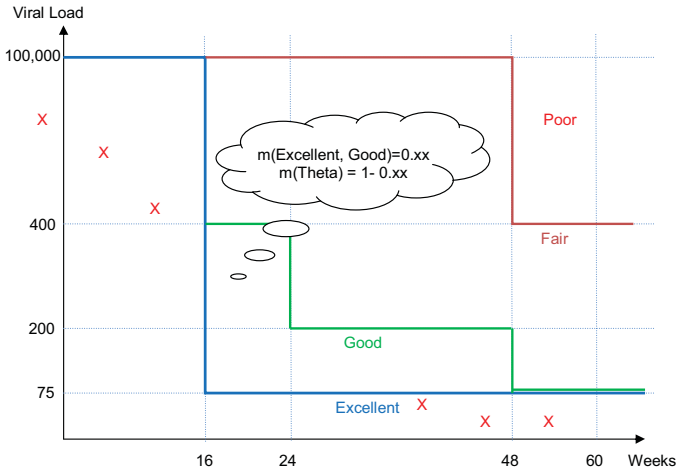
Each of the three expert thus assign ratings for HAART regimen based on the available information regarding the virological response. These individual ratings may be combined to arrive at a final rating for the HAART regimen.

Given a patient with the ratings for his/her previous HAART regimen treatments, our task is to suggest what might be the rating for a new HAART regimen. For better understanding, let us reformulate the problem in more familiar domain–'movie ratings'. Companies such as NetFlix let you rate the movies you watched. You are allowed to pick a one value out of five, five being the best and 1 being the worst. These ratings are then used to suggest/recommend new movies to you. The principle behind this is that if many other users with a 'taste' similar to your taste have given a higher rating to, say, 'sound of music', then it is possible that you also might like that movie. The difference in our HAART regimen rating is that the ratings are not 'crisp'–they contain many ambiguities. Thus, we need an efficient and effective framework that is capable of accommodating and propagating those uncertainties to the decision level without making any unwarranted assumptions.

Recent work has studied this issue in the framework of classifier induction: for class-label ambiguities [SZP$^+$08] as well as for attribute-value imperfections [HPS07]. But the problem studied here is more general. Whereas classification usually seeks to predict a single preselected class attribute, we are concerned with the case where *any* attribute can be the "class label." We want to predict all unknown *items* based on



(a) Lack of evidence is accounted for by assigning a mass to the complete ambiguity Θ



(b) The rating could be 'Excellent' or 'Good' depending on the viral load counts in the 'unobserved' period. Belief is assigned to the composite proposition$\{Excellent, Good\}$. Uncertainty is captured by assigning a mass to Θ.

Figure 5.3: Rating assignment in the face of lack of information

the partial knowledge of the presence of other items (note that classification is only a special case of this task). A collaborative-filtering-based approach to this task has been recently proposed by [Wic08], but to use association mining to this end is new. We propose a novel technique, DS-ARM—Dempster-Shafer based Association Rule Mining and report experiments illustrating its behavior.

5.2 Formal Problem Statement

We use p_j, $j = \overline{1, N_p}$, to denote products (or attributes) in the dataset. Let $\Theta = \{\theta_1, \ldots, \theta_K\}$ be the set of mutually exclusive and exhaustive ratings. Rating values that can be assigned to a product are thus drawn from the power set 2^Θ of Θ and $\mathfrak{r}_\ell$, $\ell = \overline{1, N_\mathfrak{r}}$, where $N_\mathfrak{r} = |2^\Theta|$, is used to denote user assigned ratings. We refer to each pair $\langle product, rating \rangle$ or $\langle attribute, value \rangle$ as an *item* and the item vector of a single user as a *transaction*. More formally, let $I = \{i_{j\ell} | j = \overline{1, N_p}, \ell = \overline{1, N_\mathfrak{r}}\}$ be a set of distinct items where $i_{j\ell} = \langle p_j, \mathfrak{r}_\ell \rangle$. Let a database consist of N transactions, $T_1, \ldots, T_N$, such that $T_k \subseteq I$, $\forall k$. An *itemset*, X, is a group of items, i.e., $X \subseteq I$. The *support* of itemset X is the number, or the percentage, of transactions that subsume X. An itemset that satisfies a user-specified minimum support value is called a *frequent itemset* or a *high support itemset*.

Let us assume that an association mining program has already discovered all high support itemsets. For each such itemset, X, any pair of subsets, $r^{(a)}$ and $r^{(c)}$, such that $r^{(a)} \cup r^{(c)} = X$ and $r^{(a)} \cap r^{(c)} = \emptyset$, we can define an association rule: $r : r^{(a)} \Rightarrow r^{(c)}$; $r^{(a)}$ is the rule's *antecedent* and $r^{(c)}$ is the *consequent*. The rule reads: if all items from $r^{(a)}$ are present in a transaction, then all items from $r^{(c)}$ are also present in the

same transaction. The rule does not have to be absolutely reliable. The probabilistic *confidence* in the rule $r^{(a)} \Rightarrow r^{(c)}$ can be defined with the help of the support (relative frequency) of the antecedent and consequent as the percentage of transactions that contain $r^{(c)}$ among those transactions that contain $r^{(a)}$:

$$conf = support\,(r^{(a)} \cup r^{(c)})/support\,(r^{(a)}). \tag{5.1}$$

The number of rules implied by X grows exponentially in the number of items; it is thus practical to consider only high-confidence rules derived from high-support itemsets.

Given an itemset s in a transaction, we want to predict the remaining items of this transaction. The association rules we generate for this purpose must satisfy the following: (1) The rule antecedents should be sufficiently similar to s. (2) The rule consequent is limited to *any* single item $\langle p_j, \bullet \rangle \notin s$.

In summary: Given the itemset $s \subseteq I$, find the matching rules of the form $r^{(a)} \Rightarrow \langle p_j, \bullet \rangle$, such that $r^{(a)}$ is "close" (we will formalize this later) to s and $\langle p_j, \bullet \rangle \notin s$, that exceed the user-set minimum support, θ_s, and minimum confidence, θ_c. Then find a method to combine rules with mutually contradicting consequents to predict unknown items. In this way, we are ultimately predicting the $\langle product, rating \rangle$ values of unrated products.

Handling Imperfections

The FoD of rating of product p_j, is taken to be finite and is denoted by Θ_{pref}. For instance, in a "five-star" rating system $\Theta_{pref} = \{1, 2, 3, 4, 5\};$. The number of possible singleton values a product rating may assume is $|\Theta_{pref}|$ and $\mathfrak{r}_\ell \in 2^{\Theta_{pref}}$. The "intra-attribute BBA" or the BBA of rating of product p_j is a BBA $m_j : 2^{\Theta_{pref}} \mapsto [0, 1]$ defined on the FoD $\Theta_{pref}; \{\Theta_{pref}; \mathfrak{F}_j; m_j\}$ (*intra-BoE*) [HPS07]. Note that, $\mathfrak{F}_j = \emptyset$

denotes that the product p_j is "not rated". We assume that a $\langle product, rating \rangle$ vector whose ratings are all "not rated" is non-existent (i.e., in our context, each user has rated at least one product).

The intra-attribute BBA captures the uncertainty among the ratings each product may take. An intra-BBA allows several types of common data imperfections to be conveniently modeled. For example, for $\Theta_{pref} = \{\theta_1, \theta_2, \theta_3\}$, Table 5.1 shows the types of data imperfections that the intra-BBA $m_j(\bullet)$ can capture. These DS theoretic notions allow one to represent a wide variety of data imperfections with ease. For example, in our HAART therapy scenario, the BBAs $\{m(Good); m(\Theta_{pref})\} = \{0.7; 0.3\}$ and $m(Excellent; Good; Fair) = 1.0$ would elegantly capture the ratings "Good with a 70% level of confidence" and "definitely not Poor but more evidence is needed to discern further," respectively; an unrated item can be captured via the vacuous BBA $m(\Theta_{pref}) = 1.0$.

Table 5.1: Types Of Imperfections That Can Be Captured By An Intra-BBA

Type of imperfection	Intra-BBA	
	Proposition	$m_{p_j}(\bullet)$
Hard (perfect)	θ_2	1.0
Probabilistic	θ_1	0.2
	θ_2	0.3
(focal elements are singletons)	θ_3	0.5
Possibilistic	θ_1	0.7
	(θ_1, θ_3)	0.2
	Θ_{pref}	0.1
Ambiguous (Unable to discern)	(θ_1, θ_2)	1.0
Missing/Unknown	Θ_{pref}	1.0
Belief theoretic	$\sum_{A \subseteq \Theta_{pref}} m_{p_j}(A) = 1.0$	

The "inter-attribute BBA" can capture the interrelationships among different attributes [HPS07]. Clearly, the inter-FoD Θ_T of each attribute vector T is the cross-product of the intra-FoD of each attribute. The inter-BBA of a given record is referred to as Data Record BBA (DR-BBA).

Table 5.2 shows a toy domain with four distinct products and the ratings (some ambiguous, others "crisp") given by two users. An empty field indicates the user has not rated the product. For instance, the user u_1 has given a rating of 4 for the product p_1 and for the product p_2, u_1 has given a possibilistic rating. The product p_4 has not been rated by u_1. DR-BBAs generated from Table 5.2 are shown in Table 5.3. The user rating vector u_1 is converted into four data-records in the cross-product space (that grows exponentially in the number of ambiguous ratings). So, the proposed method becomes expensive in highly ambiguous domains. Our toy domain can be seen as a transaction database where each $\langle product, rating \rangle$ represents an item and each row represents a transaction. This database is then used for frequent-itemset detection and for association rule generation.

Table 5.2: Intra-BBAs of Two Data Records

product	p_1		p_2		p_3		p_4	
user	$\mathfrak{F}_1$	m_1	$\mathfrak{F}_2$	m_2	$\mathfrak{F}_3$	m_3	$\mathfrak{F}_4$	m_4
u_1	4	1.0	4	0.8	3	0.6		
			4,5	0.2	3,4	0.4		
u_2	1	1.0	5	1.0			3	0.8
							2,3	0.2

5.3 Making Predictions

Given a user's ratings with ambiguities and asked to predict unrated products, the first step is to get the cross-product of intra-BBAs and find the DR-BBAs of the given rating vector. For instance, if the given user is u_2 (Table 5.2), and we are asked

Table 5.3: DR-BBAs of the Data Records in Table 5.2

Data Rec.	DR-BBA	Itemset
$u_1^{(1)}$	0.48	$\langle p_1, 4\rangle, \langle p_2, 4\rangle, \langle p_3, 3\rangle$
$u_1^{(2)}$	0.32	$\langle p_1, 4\rangle, \langle p_2, 4\rangle, \langle p_3, (3, 4)\rangle$
$u_1^{(3)}$	0.12	$\langle p_1, 4\rangle, \langle p_2, (4, 5)\rangle, \langle p_3, 3\rangle$
$u_1^{(4)}$	0.08	$\langle p_1, 4\rangle, \langle p_2, (4, 5)\rangle, \langle p_3, (3, 4)\rangle$
$u_2^{(1)}$	0.80	$\langle p_1, 1\rangle, \langle p_2, 5\rangle, \langle p_4, 3\rangle$
$u_2^{(2)}$	0.20	$\langle p_1, 1\rangle, \langle p_2, 5\rangle, \langle p_4, (2, 3)\rangle$

to predict the rating for product p_3, we get two "data records" $u_2^{(1)}$, $u_2^{(2)}$ (Table 5.3). The matching rule set is generated for each of the records, and the prediction is made by the combination of the rules. Each prediction is discounted based on the corresponding DR-BBA; the discounted BBAs are then combined in making the final prediction.

For a given itemset $s \subseteq I$, we want to find all rules of the form $r^{(a)} \Rightarrow i_{j\ell}$, where $r^{(a)}$ "matches" s and $\langle p_j, \bullet\rangle \notin s$, that exceed minimum support and minimum confidence. Note that the consequent $i_{j\ell}$ is a single item, i.e., $\langle p_j, \mathfrak{r}_\ell\rangle$. For each unrated product p_j, the corresponding ruleset—all the matching rules having a consequent of the form $\langle p_j, \mathfrak{r}_\ell\rangle; \mathfrak{r}_\ell \subseteq 2^{\Theta_{pref}}$—is selected and a DS theoretic approach is used to combine the rules. This prediction is given as a DS theoretic mass structure over the set of singletons or the frame of discernment. If no rule consequent in the generated ruleset has $\langle p_j, \bullet\rangle$, no prediction is made for p_j.

5.3.1 Distance Metrics

We define a rule $r^{(a)} \Rightarrow i_{j\ell}$ and given itemset s is "matching" iff (a) $\forall\langle p_j, \bullet\rangle \in r^{(a)} \rightarrow \langle p_j, \bullet\rangle \in s$, and (b) $\forall\langle p_j, \mathfrak{r}_j^{(r^{(a)})}\rangle \in r^{(a)}$ and $\langle p_j, \mathfrak{r}_j^{(s)}\rangle \in s$: $d_j \equiv |\mathfrak{r}_j^{(r^{(a)})} - \mathfrak{r}_j^{(s)}| \leq d_t$; where d_t is a user-set distance threshold. $\mathfrak{r}_j^{(r^{(a)})}$ and $\mathfrak{r}_j^{(s)}$ are the ratings given for the

product p_j in the rule antecedent $r^{(a)}$ and given itemset s respectively. If the rating $\mathfrak{r}_j$ is not a singleton we take the mean value to calculate the distance.

The distance between a matching rule antecedent and incoming itemset s is denoted by $\mathfrak{d}_{s,r^{(a)}}$; where $\mathfrak{d}_{s,r^{(a)}} = \sum_j d_j/|r^{(a)}|$.

5.4 Performance Criteria for "Soft" Predictions

When the user preference ratings are soft, we must determine how well the predicted BoE's ($\hat{\gamma}_j$) approximate the ground truths (γ_j). $BetP_{\gamma_j}$ denotes the pignistic probabilities drawn from the BoE γ_j. Taking inspiration from [JGB01], we evaluate the soft result via two metrics DS_PE1 and DS_PE2. In short, DS_PE1 measures the error or distance between two mass distributions, and DE_PE2 measures the distance between two probability distributions. Hence, in order to compute DS_PE2 we convert the mass distribution to a probability distribution via pignistic conversion. The two metrics DS_PE1 and DS_PE2 are defined below.

DS_PE1**:**

$$DS\_PE1 = \sum_{j=1}^{\mathfrak{N}_p} JGB(\gamma_j, \hat{\gamma}_j) \div \mathfrak{N}_p \tag{5.2}$$

where $JGB(\gamma_j, \hat{\gamma}_j) = \sqrt{0.5(m_{\gamma_j} - m_{\hat{\gamma}_j})D(m_{\gamma_j} - m_{\hat{\gamma}_j})^T}$. Here, $m_{\gamma_j}, m_{\hat{\gamma}_j}$ are each a size $2^{\Theta_{pref}}$ vectors containing the masses allocated to each subset of Θ_{pref} by γ_j and $\hat{\gamma}_j$, respectively; $D = \{d_{k,\ell}\}$ is a size $2^{\Theta_{pref}} \times 2^{\Theta_{pref}}$ matrix with $d_{k,\ell} = |A^{(k)} \cap A^{(\ell)}|/|A^{(k)} \cup A^{(\ell)}|$; $A^{(k)}, A^{(\ell)} \in 2^{\Theta_{pref}}$; $|\emptyset \cap \emptyset|/|\emptyset \cup \emptyset| \equiv 0$.

DS_PE2:

$$DS\_PE2 = \sum_{j=1}^{\mathfrak{N}_p} \frac{1}{\sqrt{2}} ||BetP_{\hat{\gamma}_j} - BetP_{\gamma_j}|| \div \mathfrak{N}_p, \tag{5.3}$$

where $\mathfrak{N}_p$ is the number of predictions made and $||\bullet||$ denotes the Euclidean norm.

Note that *DS_PE*'s are error measures and take values from $[0, 1]$: $DS\_PE = 0$ means the prediction is exactly same as that of the ground truth. We could also have used the KL-divergence instead of the Euclidean norm, but the error then would not be bounded by the closed interval $[0, 1]$. Moreover, KL-divergence requires the pignistic distributions corresponding to the true and predicted BPAs to have identical supports.

5.5 Experiments

We experimented with MovieLens, a movie recommendation domain widely used for benchmarking [Res07]. The dataset consists of 100,000 ratings provided by 943 users for 1682 movies. The ratings are integers from the interval [1,5], with 5 being best and 1 being the worst. Users were allowed to assign crisp ratings. To demonstrate our technique's full functionality, we needed soft ratings that were not available in MovieLens. We thus created the dataset DS-MovieLens by artificially introducing soft ratings: we relied on different user profiles obtained from "partial probability models," a widely used methodology to convert data with diverse types of imperfections into the DS theoretic framework [BP99], [HPS07].

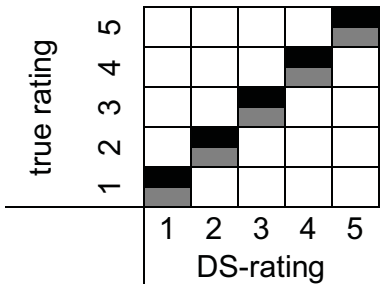
For generating the DS MovieLens dataset, we take the following viewpoint regarding the MovieLens ratings. Suppose the users considered "soft" ratings. Given that the MovieLens domain only shows hard ratings, the following question nat-

urally arises: what soft rating could have generated the observed hard rating (in MovieLens)? Or, conversely, what mechanism might transform a soft rating from DS MovieLens into a hard rating in MovieLens?

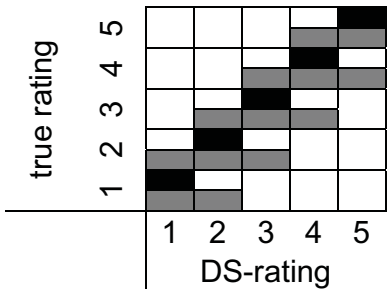
To be more specific, we used—as in [Wic08]—three user profiles: zero tolerance, ± 1 tolerance, and end-weighted ± 1 tolerance. The partial probability models for each profile are shown in Figure 5.4. The horizontal axis (lighter shading) always represents the user rating as it appears in the DS-MovieLens dataset; the vertical axis (dark shading) represents the true rating a movie should receive. A power-set approach enables us to account for user rating imperfections without resorting to various "assumptions" and "interpolations" that may be hard to justify.

We then built DS-MovieLens by the following steps: (a) Select a user rating that has been rated as $\mathfrak{r}_k$. (b) Randomly, with the probabilities $\{p, (1-p)/2, (1-p)/2\}$, select one user profile from Figs 5.4(a), 5.4(b), and 5.4(c), respectively. (c) Obtain the corresponding feasible true ratings and DS theoretic basic probability assignment (BPA) $\mathfrak{r}_k^{(DS)}$ via the procedure in [BP99]. (d) Replace $\mathfrak{r}_k$ with $\mathfrak{r}_k^{(DS)}$. (e) Repeat for all rated entries in MovieLens dataset.

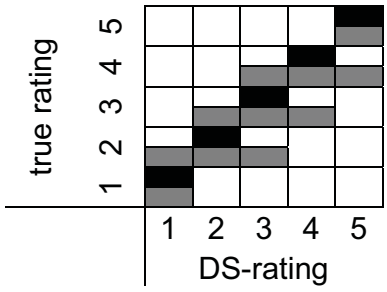
To see how we use user profiles to generate DS MovieLens dataset, consider a movie with True Rating = 2. A ± 1 tolerance user may, with equal probability, allocate either True Rating = 2 or a rating from the set (1, 2, 3) (see Fig. 5.4(b). One may also interpret this as follows: if MovieLens would allow it, a ± 1 tolerance user may sometimes rather give the interval valued rating (1, 2, 3) to this same movie instead of the rating 2 he/she was forced to allocate. We capture these two possibilities by saying that we are using either the black or gray distribution, the former representing the hard rating = 2 and the latter representing the interval valued rating (1, 2, 3). In a similar manner, for the same movie, a zero tolerance user would allocate MovieLens



(a) 0 tolerance



(b) ±1 tol.



(c) ±1 tol. end-weight

Figure 5.4: Partial probability models of user profiles.

Rating = 2 (see Fig. 5.4(a)); an end-weighted ±1 tolerance user behaves similar to a ±1 tolerance user (see Fig. 5.4(c)). Clearly, these three user profiles enable us to represent a relatively broad spectrum of users.

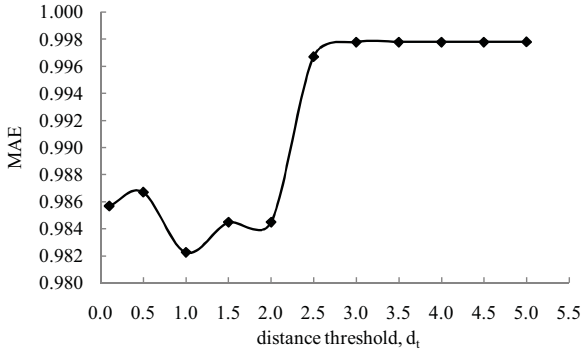
Experiment Setup

For consistency with previous work, we followed the methodology from [HKBR99]: We randomly selected 10% of users and, for each of them, we withheld 5 randomly selected ratings, i.e., we "hid" 5 non-empty fields in the ratings matrix and prevented them from being used for training. We then used these withheld ratings as an independent testing set. The remaining ratings represented the training set. We repeated this process for 10 different random splits into training and testing sets. Results shown here are the average results obtained from the 10 splits.

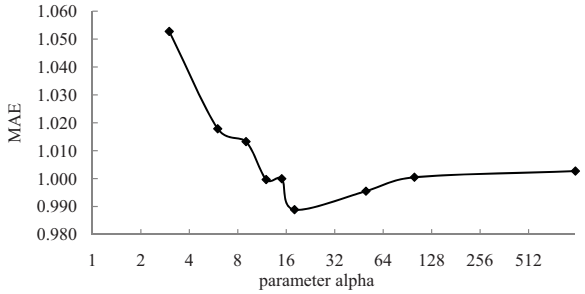
Experiment 1. DS-ARM Performance and parameter settings

Let us first investigate DS-ARM's behavior under diverse parameter settings. The technique performance is likely to depend on the distance threshold d_t, the minimum *p_supp* threshold, and the parameter α in Eq. (3.6). For the time being, let us focus on mean absolute error, MAE. Throughout the experiments, we will keep two parameters fixed at "baseline values," while varying the third parameter. The baseline values are $p\_supp = 0.01, \alpha = 10, d_t = 1.5$.

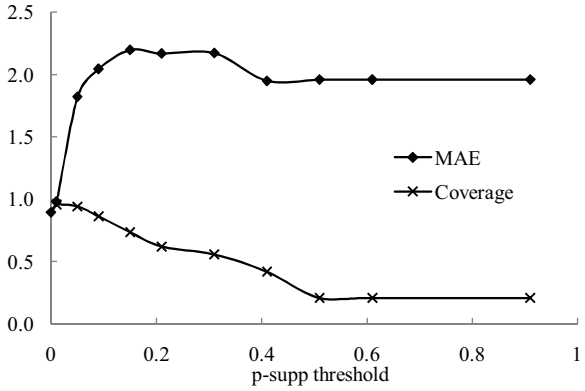
Figure 5.5(a) shows how the performance varies with growing d_t, with the other parameters fixed. The minimum error was achieved when $d_t = 1.0$. With high d_t, the error increases due to the contributions from too "dissimilar" rules. When the distance threshold is tight, few rules are involved and the lack of diverse opinions seems to cause errors. Figure 5.5(b) shows how MAE varies with changing α. The



(a) Overall MAE versus d_t



(b) Overall MAE versus α



(c) MAE and Coverage Vs *p_supp*

Figure 5.5: Behavior of DS-ARM

minimum is reached around $\alpha = 20$, which supports our use of the partitioned-support value in mass allocation. Figure 5.5(c) shows how MAE varies with minimum partitioned-support threshold of selecting rules. Best performance is obtained by keeping the threshold very low. We have to remember that the computational costs of rule combination are high if many rules are combined. We observe that as the *p_supp* threshold increases, the coverage decreases (i.e., no rules are selected to predict certain preferences).

Experiment 2. DS-ARM Performance on Hard Data:

Although the main strength of DS-ARM is its ability to deal with ambiguous ratings, we still wanted to see how it compares to older techniques when "crisp" values are used. To this end, we compared DS-ARM with one of the most widely used mechanisms of Automated Collaborative Filtering (ACF): the correlation analysis based approach from [HKBR99]. We will refer to our re-implementation of this algorithm by the acronym CORR.

The parameters of both systems were set to maintain at least 95% level of "coverage," calculated as the percentage of predictions made by the predictor out of the total number of predictions. Predictors sometimes fail to achieve 100% coverage due to lack of evidence. The parameter settings are: for DS-ARM, $d_t = 1.0$, $\alpha = 20$, $p\_supp\ thres = 0.01$; and for CORR, similarity threshold = 0.1.

The mean absolute error (MAE) is the most popular performance criterion to evaluate user ratings [HKTR04]. Since our algorithm presents the prediction as a mass structure over the FoD $\Theta_{pref} = \{1, 2, 3, 4, 5\}$, to compute the MAE, these DS theoretic predictions are converted to hard predictions via the pignistic transformation. Pignistic transformation converts the DS-theoretic "soft" decision to a hard decision.

Note that even though we did this transformation to directly compare our results with other available methods, this approach is somewhat unfair to the proposed DS-ARM whose strength lies in its ability to generate soft decisions. In addition to MAE, other standard performance metrics—such as precision, recall, and F_1—were used in the results.

Table 5.4 summarizes the results, with boldface values indicating the best performance. Although the difference is on average only marginal, our method consistently out-performs CORR in predicting high user ratings "3-5". For ratings "1-2", CORR is better. Note that the frequencies of ratings "1-2" are low (the ratings distribution is "1": 6%, "2": 11%, "3": 27%, "4": 34%, "5": 21%). Based on the results, we conclude that the two methods provide comparable performance even in the case of "crisp" data.

Table 5.4: Performance Comparison: Hard Decisions

algo.		True Rating					mean
	Metric	1	2	3	4	5	MAE
DS-ARM	MAE	2.31	1.62	**0.68**	**0.39**	**1.06**	0.89
	Pr	0.38	0.13	**0.38**	**0.39**	**0.36**	
	Re	0.08	0.04	**0.39**	**0.63**	**0.23**	
	F_1	0.14	0.06	**0.39**	**0.48**	**0.27**	
CORR	MAE	**1.80**	**1.38**	0.71	0.57	1.18	0.91
	Pr	**0.40**	**0.19**	0.33	0.38	0.32	
	Re	**0.16**	**0.16**	0.38	0.51	0.19	
	F_1	**0.23**	**0.18**	0.36	0.44	0.24	

Experiment 3. DS-ARM Performance on Soft Data:

As we have said, we were not aware of any other system that can predict ratings based on the "soft" data such as those in DS-MovieLens. Still, we felt that some comparison with previous work is needed. This is why we decided to use the CORR approach we worked with in Experiment 2 and to interpret the hard decisions made

by CORR predictor as soft decisions. Employing the CORR on the soft data is made easier by the fact that it can work with real values ratings. In order to apply CORR, soft ratings in the dataset are converted to probabilistic ratings by applying pignistic conversion and then a real valued rating is obtained by taking the expected value of the resulted probability distribution.

The comparison of CORR and DS-ARM is made simpler by the fact that it is in the nature of correlation analysis that the predictions of CORR are not necessarily integer-valued. To be able to interpret a CORR prediction, $\hat{\mathfrak{r}}_k$, as soft, we relied on the following DS-theoretic BPA:

$$\hat{m}_k(A) = \begin{cases} \lceil\hat{\mathfrak{r}}_k - \hat{\mathfrak{r}}_k, & \text{for } A = \lfloor\hat{\mathfrak{r}}_k \text{ when } \hat{\mathfrak{r}}_k \notin \Theta; \\ \hat{\mathfrak{r}}_k - \lfloor\hat{\mathfrak{r}}_k, & \text{for } A = \lceil\hat{\mathfrak{r}}_k \text{ when } \hat{\mathfrak{r}}_k \notin \Theta; \\ 1, & \text{for } A = \hat{\mathfrak{r}}_k \text{ when } \hat{\mathfrak{r}}_k \in \Theta; \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

where $\lceil\hat{\mathfrak{r}}_k$ and $\lfloor\hat{\mathfrak{r}}_k$ denote the lowest integer rating that does not fall below and the highest integer rating that does not exceed the CORR prediction $\hat{\mathfrak{r}}_k$, respectively. For instance, with $\Theta = \{1, 2, 3, 4, 5\}$, the CORR prediction 3.3 is interpreted as the Bayesian statement, "The rating is 3 with 70% confidence, and 4 with 30% confidence"; (5.4) corresponds well with this interpretation.

Table 5.5 compares the results of CORR and DS-ARM, using the performance metric DS_PE1 and DS_PE2 defined by (5.2 and 5.3). The probability of selecting the "zero tolerance user" varies from 1 (no ambiguity) to 0.8 (20% ambiguity). The results indicate that DS-ARM indeed comfortably outperforms CORR on these data.

Table 5.5: Performance Comparison: Soft Data

algo.	metric	Zero tolerance user selection probability, p				
		1.00	0.95	0.90	0.85	0.80
DS-ARM	DS_PE1	**0.60**	**0.59**	**0.58**	**0.57**	**0.57**
	DS_PE2	**0.57**	**0.55**	**0.53**	**0.52**	**0.51**
CORR	DS_PE1	0.62	0.62	0.61	0.59	0.59
	DS_PE2	0.58	0.57	0.56	0.55	0.54

5.5.1 Interpretation of results

At first glance we can see that the DS_PE values decreases with the increasing ambiguity. This does not mean that the predictions are improving. The quality of the prediction cannot be determined by simply observing the DS_PE values.

For simplicity of understanding, we can explain this by taking an analogy from mean absolute error (MAE) measure. Assume a 5-star rating system. If the actual user rating is always 3, the maximum error of any prediction is 2. However, if the actual user rating is 1 or 5, then the error of a prediction could go up to 4. Thus, just by looking at the MAE value one cannot judge the quality of prediction: MAE of 2 and MAE of 4 could be equally bad. It all depends on the actual ratings in the dataset. In order to get a better grasp of the meaning of the DS_PE measures, the following experiments were carried out. In this experiment, the actual rating vector (mass distribution or pignistic probability distribution) is modified by adding a random number (selected from a normal distribution having 0 mean and x standard deviation) to each mass/probability value in the vector. Then the modified rating vector is normalized.

For instance, if the rating of an item is 4 with probability 0.8 and 3 with probability 0.2 (probabilities are 0 for other ratings), these probability values are modified by adding random noise taken from a normal distribution with mean 0 and standard deviation x. (value of x is varied from 0 to 2)

The DS_PE measures are then computed to measure the distance between the actual and the modified rating vectors. The resulting DS_PE values are plotted against the selected standard deviation values of noise. The DS_PE distances between complete ambiguity and the noise added rating vector is also plotted on the same graph. In figure 5.7, $D(actual, noise\ added\ actual)$ is the DS_PE1 distance between actual rating and the noise added rating. $D(theta, noise\ added\ actual)$ is the distance between complete ambiguity and the noise added rating. The 'saturation' points of these curves depend on the actual ratings. If the actual ratings are too ambiguous the curves 'saturate' at lower distances. The DS_PE2 measure also showed a similar variation.

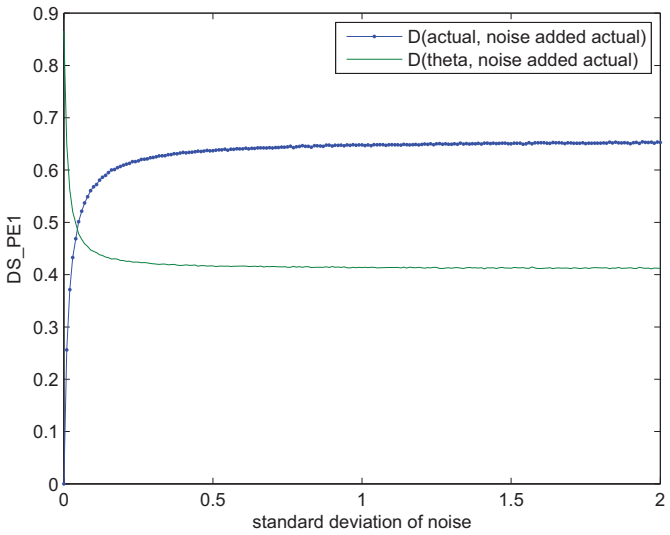
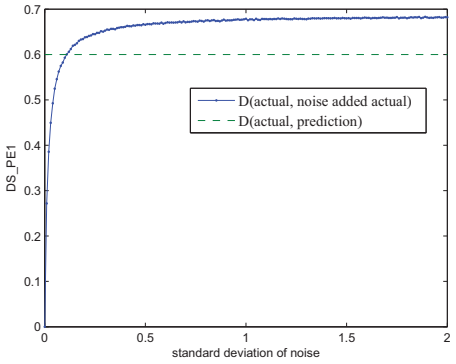
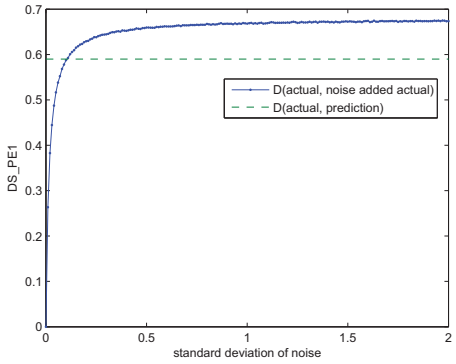


Figure 5.6: Variation of DS_PE1 distance

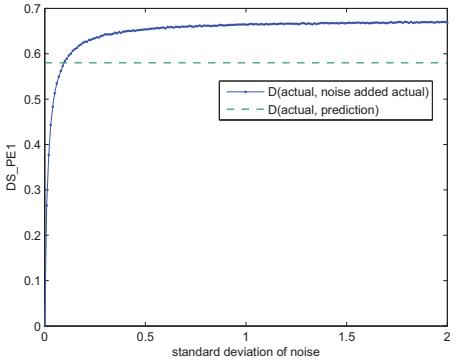
In general we expect a lower average DS_PE distance between predictions and actual ratings. The average DS_PE distance for complete random predictions would be close to the saturation points of the curves. If the average DS_PE distance between predictions and actual ratings is below the 'turning point' of the curve, it can be considered as a reasonable prediction.



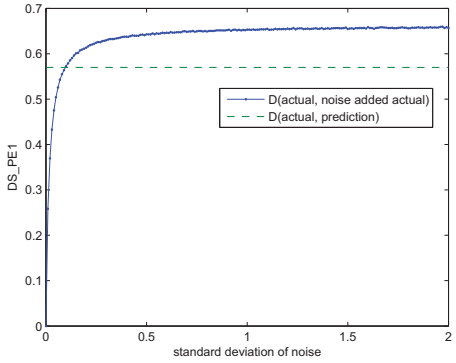
(a) DS_PE1 curves for $p = 1.00$.



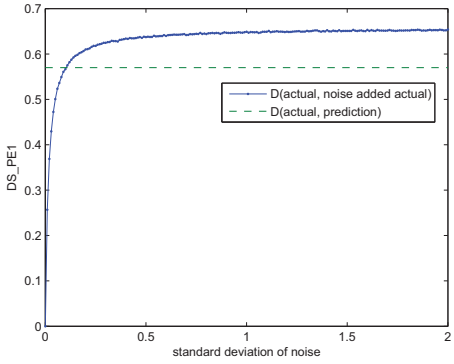
(b) DS_PE1 curves for $p = 0.95$.



(c) DS_PE1 curves for $p = 0.90$.



(d) DS_PE1 curves for $p = 0.85$.



(e) DS_PE1 curves for $p = 0.80$.

Figure 5.7: Quality of prediction of DS_ARM on MovieLens dataset in terms of DS_PE1 metric

Figure 5.7(a) through 5.7(e) may be used to visualize the quality of the DS-ARM predictions. The dotted line shows the DS_PE1 distance between the prediction and the actual rating. It being well below the 'saturation' level of the curve indicates that the prediction is much better than random guessing. As data become more and more ambiguous we can see that the 'saturation' level continues to decrease. Hence, the reduction in the DS_PE1 distance between the prediction and the actual rating when the data become more ambiguous does not indicate better prediction accuracy. Similar observations were made for the DS_PE2 measure as well.

5.5.2 Experiments on HIV patient Dataset

Now that DS-ARM has shown its ability to learn from ambiguous data and make comparably better predictions, we employed the DS-ARM predictor on the HIV dataset to predict the drug regimen effectiveness on HIV patients. HIV dataset consisted of 222 patient records, and 49 distinct drug regimens. On average, each patient is treated with 3 to 4 drug regimens.

As described in the beginning of this chapter, virological responsiveness of the HAART regimens on patients was rated by three individual experts, initial gradient expert, profile expert, and the 'tail' expert. We tested the DS-ARM's ability to predict each individual expert's ratings.

In order to evaluate the whether the DS-ARM predictor can perform at least better than a totally random prediction, we developed a random predictor that always gives a random BoE as the prediction. Table 5.6 shows the DS_PE distance between the prediction and the actual ratings for both the DS-ARM and the random predictor. Average values of 5-fold cross validation results are shown in the table. The table indicates that the DS_ARM can do better than a random predictor. Even though,

the DS_ARM always beats the random predictor, in the case of a profile expert and the combined expert predictions, the performance difference is marginal, especially in terms of DS_PE1 metric.

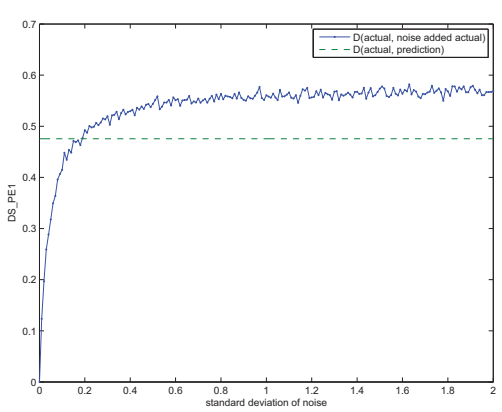
Table 5.6: Performance comparison of DS_ARM with a random predictor on the HAART regimen virological responsiveness prediction.

Predictor	Metric	Expert			
		Gradient	Profile	Tail	All Combined
DS_ARM	DS_PE1	0.482	0.369	0.496	0.498
	DS_PE2	0.421	0.279	0.450	0.439
Random predictor	DS_PE1	0.576	0.382	0.574	0.518
	DS_PE2	0.595	0.390	0.590	0.524

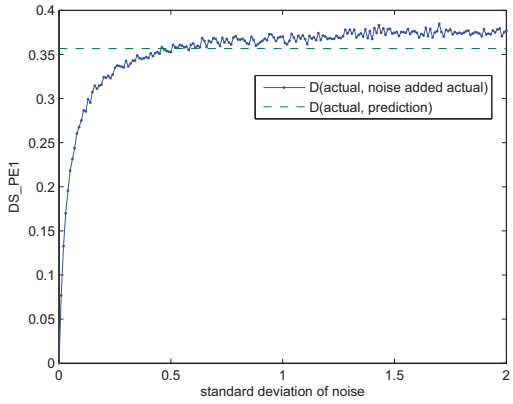
In order to further evaluate the results, the $DS\_PE(actual, noise\ added\ actual)$ and the $DS\_PE(actual, prediction)$ were plotted on the same graph as described in the previous section. As can be seen from the figure 5.8(b) the results for the profile expert predictions are not that impressive in terms of DS_PE1 metric. However, it performs much better in terms of the DS_PE2 metric (Fig. 5.9(b)). This is because the DS-ARM prediction assigns a considerably high mass to the total ambiguity. In other words, the predictor is highly uncertain about its decision. This is a result of the lack of strong (i.e., rules with high confidence and support) association rules to make a strong prediction.

However, in the case of combined prediction, DS-ARM fails in terms of both metrics. This is mainly due to the conflict of opinion between the three experts and the inherent weaknesses of Dempster's combination to handle the conflict. In addition, there are many other reasons for this result.

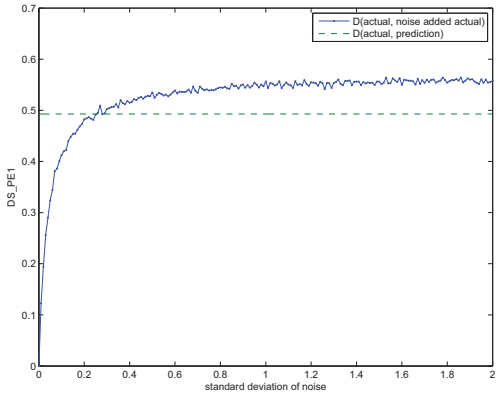
- As opposed to the MovieLens data set where we have 100,000 data records, and each user has rated at least 20 movies—the patient data set is too small and the available patient history(only a few regimens) is too short to learn patterns.



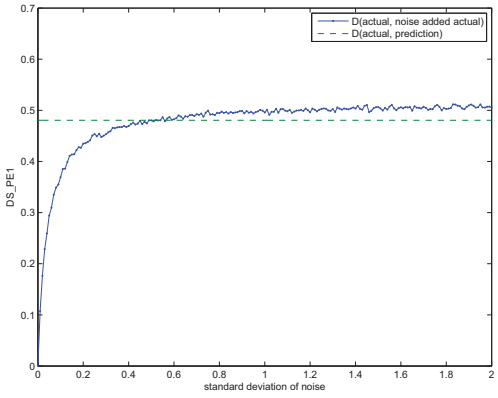
(a) Gradient-expert predictions.



(b) Profile-expert predictions.

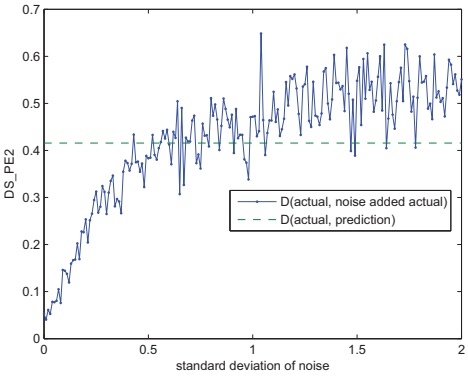


(c) Tail-expert predictions.

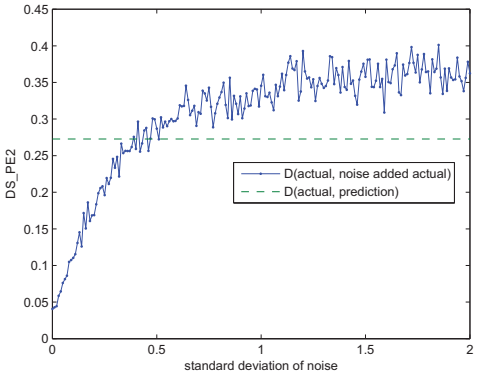


(d) Combined (all three experts) predictions.

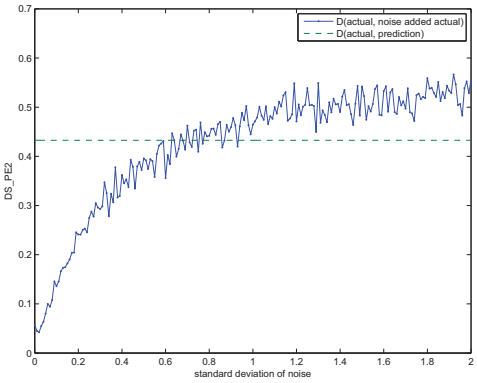
Figure 5.8: Quality of DS_ARM prediction on HAART regimen virological responsiveness dataset in terms of DS_PE1 metric



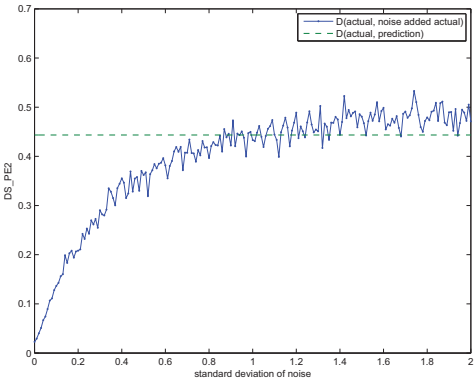
(a) Gradient-expert predictions.



(b) Profile-expert predictions.



(c) Tail-expert predictions.



(d) Combined (all three experts) predictions.

Figure 5.9: Quality of DS_ARM prediction on HAART regimen virological responsiveness dataset in terms of DS_PE2 metric

- Other background information plays a huge role in the medical domain. For instance, a HAART regimen's virological response could be vastly different on a 'young-Asian-male' patient as compared to an 'old-African-American-female'. Even the patient's body-mass-index could play a major role. However, we do not have that background information in our dataset.

CHAPTER 6

Using Taxonomy Information to Improve the Predictions

The experimental results of the proposed DS-ARM on the MovieLens data set were promising, and gave acceptable results on the HAART regimen virological responsiveness dataset. We applied the same methodology in an effort to predict possible opportunistic infection on HIV infected patients.

Opportunistic infections in HIV infected patients usually conspire with other two or more diseases. Having identified one of them, the physician may want to know what other diseases could conspire with the one already identified and treat all of them, rather than focusing on how to treat one single disorder. However, the number of laboratory tests that a patient is to undergo is limited by many practical (e.g., economical, patient's discomfort) factors. Under these circumstances, a decision support system advising the physician of what other diseases usually accompany the one already diagnosed can help the physician decide which other tests should be prescribed. In practice, however, this is not so simple. Knowing the presence or absence of a disease is not enough. What matters is also the severity as quantified, say, by a value from $\Theta = \{Critical, Medium, Normal\}$. When assigning the value, a physician relies on his/her experience and/or the experience of colleagues. Such ratings are inevitably ambiguous and not easily expressed in terms of probabilities—for instance, it would

be mistaken to assume that the statement, "the symptom is *Critical* with a 70% confidence" implies a 30% confidence in the complement of *Critical*. On top of that, a different expert's opinion on the same matter could be contrastingly different.

Since DS-ARM has proven to work well in a such situation, we expected encouraging results in this situation as well. However, it failed to show acceptable performance on the actual HIV patient dataset, in the case of predicting possible opportunistic infections. Our first feeling was that these diseases are too specific, so that the associations among them are weak. Taking an example from the market basket domain, besides finding the *percentage of people who buy 'wheat bread' if they buy '2 percent milk'* it could also be important to see the *percentage of people who buy 'bread' if they buy 'milk'* (i.e., more general rules). Can we improve our results by combining results from association rules generated at higher level of abstraction? Hereafter, the discussion is focused on developing an efficient method for mining association rules at multiple taxonomy levels.

6.1 Taxonomy Lattice

The necessity for mining multiple level association rules or using taxonomy information at mining association rules has already been observed by researchers, e.g.,[AS94]. The most prevailing approach is to create a generalization tree according to taxonomy information as shown in figure 6.1 and to mine the associations at different levels of the tree. This requires non-overlapping categorization of items in the dataset (i.e. an item in lower level belongs to only one immediately higher layer category). However, this is not the case for many datasets. Figure 6.2 shows a generalization lattice from the movies domain, in which one lower level item may belong

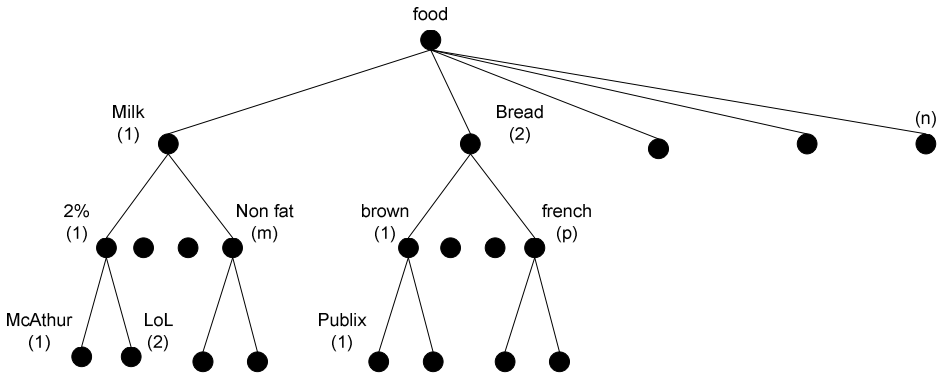


Figure 6.1: A taxonomy for market basket dataset. (non-overlapping categorization of items)

to many higher level items. This, in fact, is the case for the HIV patient dataset as well. If we categorize the opportunistic diseases into more abstract categories, the lower level, more specific diseases may belong to one or more higher level categories. For the ease of understanding, we will continue the discussion in the movies domain.

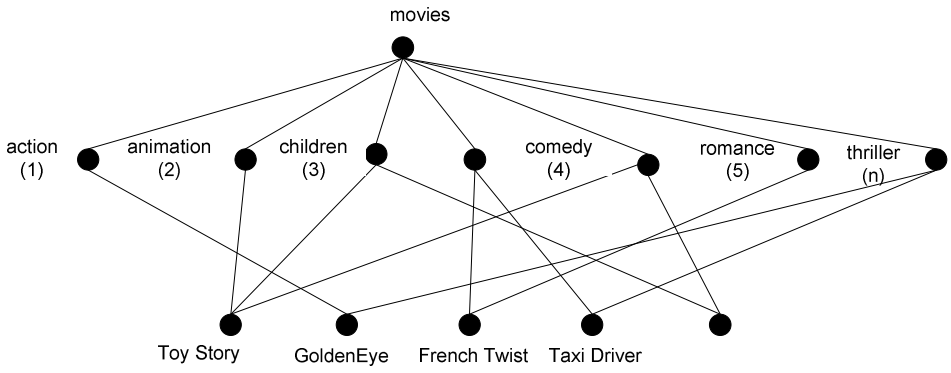


Figure 6.2: A generalization lattice for movies dataset (first level shows the genre information)

In general, users have a tendency to select the movies from a genre of their preference. For instance, certain users like 'action' movies, certain other users like 'drama' etc. If a user has given higher ratings for 'action' movies, he/she is likely to give higher ratings for new movies from the same genre. Thus, we can get an initial esti-

mate or hunch on the rating of a new movie by looking at his/her ratings for other movies from the same genre. We also can depend on the association rules from higher level of abstraction. For instance, if most of the users who like both {`action`} and {`action, romance`}, hate {`action, comedy`} we can expect the same from others. However, an important thing to notice is that the users do not assign ratings for the genres. The rating is actually assigned for a movie that belongs to one or more genres. If a user has given a higher rating for a movie that belongs to {`action, romance`} category, that does not necessarily mean that he/she likes both individual genres. We thus use the ability of evidence theoretic frameworks to assign masses to composite propositions to model such situations.

6.1.1 Propagating the ratings in the taxonomy lattice

We use $p_i^{(\ell)}$ to denote items, where, ℓ indicates the hierarchical level of the item and $i = \overline{1, n_\ell}$, n_ℓ is the number of distinct items in each level (Fig. 6.3). $\mathbb{P}^{(\ell)} = \{p_i^{(\ell)}; i = \overline{1, n_\ell}\}$ is used to denote the set of items in level ℓ.

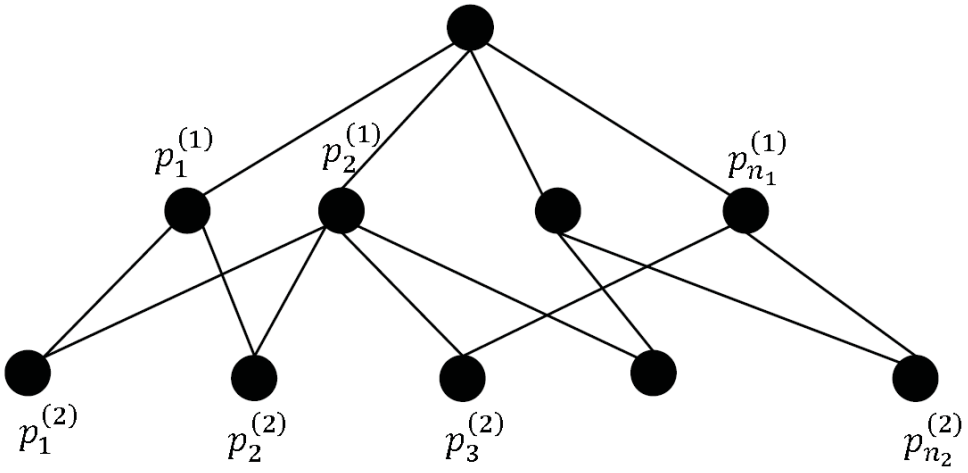


Figure 6.3: A generalization lattice

Users assign ratings for the items in the lowest level of the hierarchy. When the ratings are propagated to a higher layer, those ratings may propagate to composite items. When a user gives a rating of 4 to the 'Toy Story' (Fig. 6.2), we consider that he/she assigns that rating to the composite item {`animation, children, comedy`} in the upper level. We use $P_i^{(\ell)}$ to denote composite items from level ℓ. Note that $P_i^{(\ell)} \in 2^{\mathbb{P}^{(\ell)}}$ and $i = \overline{1, 2^{n_\ell}}$.

It is common to have many lower level items belonging to the same set of higher level items. For instance, there are many movies that belong to {`romance, drama`} category. When a user has rated two or more lower level items that belong to the same category, say, $P_k^{(j)}$, the rating for $P_k^{(j)}$ is obtained by combining the user assigned ratings using Dempster's rule of combination.

6.1.2 Use of taxonomy information to assigning an initial estimate for the rating

Assume that you want to predict the rating for item $p_k^{(2)}$, and it belongs to the higher level itemset $P_j^{(1)}$. Assignment of the initial rating estimate to $p_k^{(2)}$ is broken into three cases.

- If the user has rated one or more item that belongs to $P_j^{(1)}$, we propagate those ratings to $P_j^{(1)}$ and use the result as an initial estimate to $p_k^{(2)}$.

- If no item belongs to $P_j^{(1)}$, has been rated before, we generate association rules at level 1 of the hierarchy and use those rules to make the initial estimate.

- If no matching association rules are found, we use the vacuous BoE as the initial estimate.

Mining association rules at higher level of the hierarchy

Once the user ratings are propagated in the taxonomy lattice, we have $\langle item, rating \rangle$ vectors for each user at each level of the hierarchy. These $\langle item, rating \rangle$ vectors are then used to generate association rules at higher levels. Rule generation and combination are done by using the same procedures discussed in the previous chapters.

6.2 Experiments

The same experimental setup and performance criterion described in Experiment 3 of Section 5.5 is used to evaluate the improvements in predictions. In this experiment, an initial estimate for the user ratings was obtained using the taxonomy information and higher level association rules, as described above. These results were compared with the earlier predictions. The comparison is carried out in Table 6.1 for several different values of p, the probability with which the zero tolerance user was selected; the other 3 user profiles were selected with equal probability. The reader can see that DS-ARM consistently shows better performance when taxonomy information is used to get an initial estimate.

Table 6.1: Performance Comparison: Soft Data

algo.	metric	Zero tolerance user selection probability, p				
		1.00	0.95	0.90	0.85	0.80
DS-ARM(before	DS_PE1	0.60	0.59	0.58	0.57	0.57
using taxonomy information)	DS_PE2	0.57	0.55	0.53	0.52	0.51
DS-ARM(after	DS_PE1	**0.59**	**0.57**	**0.57**	**0.56**	**0.55**
using taxonomy information)	DS_PE2	**0.56**	**0.53**	**0.52**	**0.50**	**0.50**

It is not so simple to use taxonomy information in the case of opportunistic infections. First, we need to identify the most effective taxonomy lattice as there are so many ways to categorize one disease. Further research and heavy involvement of domain experts is necessary to carry out the work in the medical domain. In addition, rating the severity of diseases also remains a massive task to tackle.

CHAPTER 7

Conclusions

The technique DS-ARM reported in this work addresses one of the oldest tasks in association mining: using partial knowledge of a shopping cart contents, can we predict which other items the shopping cart may contain? Intuitively, the answer is "yes": very often, the presence of some items indicates the presence of others that are known to be frequently associated with the known ones. For instance, the presence of milk is an indication of milk products.

Although experiments with techniques addressing association mining usually rely on the "supermarket paradigm," it is well known that the application scope of the paradigm is much broader, including medical domains, finance, even web research, as briefly discussed in the Introduction. The literature survey that follows indicates that, while some of the recently published systems can, at least in principle, be used to this end, their practical utility is severely constrained, by the high computational costs, to domains with limited numbers of distinct items. Bayesian classifiers can be used, too, but the author is not aware of any systematic study of how this paradigm might operate under the diverse circumstances commonly encountered in association mining.

By way of seeking to overcome these limitations, the dissertation proposed a technique referred to by the acronym DS-ARM (Dempster-Shafer based Association Rule

Mining). The basic idea is relatively simple: being presented with an incomplete list, s, of items in a shopping cart, the program first identifies all high-support and high-confidence rules whose antecedents include subsets of s. A subsequent step then combines the consequents of all these (sometimes conflicting) rules and generates a set of items most likely to complete the shopping cart.

During the development of the proposed framework, three major hindrances had to be overcome. First, how to find all relevant rules in a computationally efficient manner. DS-ARM's solution capitalized on the recently proposed data structure of IT-trees that demonstrably reduced the costs of the attendant calculations to acceptable levels. The second problem was to find a mechanism to accommodate imperfections in data and the third problem was the question how to combine, and quantify, the evidence of many conflicting rules. Here, some simple ideas from the Dempster-Shafer theory of evidence fusion were used.

The results of the experiments reported in this dissertation were very promising. They show that DS-ARM is indeed capable of outperforming an approach based on Bayesian decision theory; it also outperforms older approaches, and does so even in domains designed in a manner meant to be "tailored" to these older approaches. Moreover, DS-ARM can be used in domains where the older approaches incur intractable computational costs (e.g., in domains with more than just a few distinct items). In this sense, DS-ARM clearly solved some long-standing obstacles precluding the use of association mining in real-world problems.

In spite of the encouraging results, the experiments have also shown that there is ample room for further improvements to be addressed by follow-up research. As indicated by Experiments 5 and 6, the computational costs of DS-ARM still tend to grow very fast with the average length of the transactions and with the number of

distinct items found in the database. These circumstances can become an issue in certain applications.

In addition to this limitation, it is really important to give more attention to methods capable of accommodating data imperfections and to way propagating these imperfections throughout the decision-making process. The computational costs associated with the approach addressed here have made the technique somewhat impractical in domains with large numbers of attributes. However, the author believes that the proposed ways of rule generation, rule organization, and rule combination methodology can go a long way towards mitigating this issue. With the encouraging results observed on "crisp" databases (in the experimental evaluations), subsequent work extended the methodology to Dempster-Shafer belief theoretic relational databases (DS-DB)[HPS07] that represent a much wider class of data imperfections.

Although the experimental results on a synthetically generated domain 'soft'-MovieLens and on a real-world HAART regimen virological responsiveness were encouraging, preliminary studies on a HIV patients' opportunistic infections dataset were less successful—the associations among items turned out to be weaker than expected. Intensive discussions with field experts led the author to the conclusion that the items in the dataset (diseases or medications) are too specific. Again, the problem is perhaps best illustrated using the market-basket paradigm. Apart from identifying the percentage of cases where a customer buying "2% milk" will also buy "wheat bread" (rather specific), it might be no less interesting to establish the percentage of cases where "milk" might imply "'bread" (more general). The reader will agree that the more general case can actually prove much more useful. Therefore, rather than mining association rules at a single concept level, detecting them at multiple levels and combining the results could ultimately contribute to the utility of the predictions

that can be made with them. MovieLens data results corroborate the expectation that the predictions may indeed improve. However, the effectiveness of the idea in the HIV domain is yet to be ascertained.

Future research should perhaps also look at the possibilities of grouping the patients based on background information, such as age, gender, ethnicity and use of group specific rules to fine-tune the predictions. Experimental results on the HAART data indicated that the use of DECF (Dempster's evidence combination function) to combine conflicting opinions tends to lead to a highly ambiguous rules whose practical utility was rather limited.

As a final comment, the author believes that existing methods for evidence combination are somewhat suboptimal when it comes to combining conflicting pieces of evidence. Also some recent paper did attempt to pioneer research in this direction, much more remains to be done.

Bibliography

[AIS93] R. Agrawal, T. Imielinski, and A. Swami, *Mining association rules between sets of items in large databases*, Proceedings of ACM SIGMOD, 1993, pp. 207–216.

[APY02] C.C. Aggarwal, C. Procopius, and P.S. Yu, *Finding localized associations in market basket data*, IEEE Transactions on Knowledge and Data Engineering **14** (2002), 51 – 62.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant, *Fast algorithms for mining association rules*, Proc. 20th Int. Conf. Very Large Data Bases, VLDB (Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, eds.), Morgan Kaufmann, 12–15 1994, pp. 487–499.

[BA99] R.J. Bayardo and R. Agrawal, *Mining the most interesting rules*, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Diego, California), 1999, pp. 145–154.

[BP99] S. Blackman and R. Popoli, *Design and analysis of modern tracking systems*, Artech House, 1999.

[BSHR01] P. Bollmann-Sdorra, A. Hafez, and V.V. Raghavan, *Data warehousing and knowledge discovery: 3rd international conference, (dawak-01)*, ch. A Theoretical Framework for Association Mining based on the Boolean Retrieval Model, pp. 21–30, Springer, Munich, Germany, September 2001.

[CB91] B. Cestnik and I Bratko, *On estimating probabilities in tree pruning*, Proceedings of the European Workshop on Machine Learning (Porto, Portugal), 1991, pp. 138–150.

[CS03] B. R. Cobb and P. Shenoy, *A comparison of methods for transforming belief function models to probability models*, Proc. European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU03) (Berlin, Germany) (T. D. Nielsen and N. L. Zhang, eds.), Lecture Notes in Computer Science, vol. 2711, Springer-Verlag, 2003, p. 255266.

[Dem67] A. P. Dempster, *Upper and lower probabilities induced by a multivalued mapping*, Annals of Mathematical Statistics 38 (1967), no. 2, 325339.

[DZWL99] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li, *Caep: Classification by aggregating emerging patterns*, DS '99: Proceedings of the Second International Conference on Discovery Science (London, UK), Springer-Verlag, 1999, pp. 30–42.

[Fis36] R.A. Fisher, *The use of multiple measurement in taxonomic problems*, Annals of Eugenics **7** (1936), 111–132.

[FMMT96] Takeshi Fukuda, Yasukiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama, *Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization*, SIGMOD Rec. **25** (1996), no. 2, 13–23.

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan, *Demon: Mining and monitoring evolving data*, Proceedings of the 16th International Conference on Data Engineering (Philadelphia, Pennsylvania), 1999.

[GGR00] J Gehrke, V Ganti, and R Ramakrishnan, *Detecting change in categorical data: Mining contrast sets*, Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2000, pp. 126–137.

[Goo65] I.J. Good, *The estimation of probabilities: An essay on modern bayesian methods*, MIT Press, 1965.

[GS04] S. Godbole and S. Sarawagi, *Discriminative methods for multi-labeled classification*, Proc. Pacific-Asia Conference (PAKDD'04), 2004, pp. 22–30.

[HKBR99] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl, *An algorithmic framework for performing collaborative filtering*, SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1999, pp. 230–237.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, *Evaluating collaborative filtering recommender systems*, ACM Trans. Inf. Syst. **22** (2004), no. 1, 5–53.

[HL05] Hong Hu and Jiuyong Li, *Using association rules to make rule-based classifiers robust*, ADC '05: Proceedings of the 16th Australasian database conference (Darlinghurst, Australia, Australia), Australian Computer Society, Inc., 2005, pp. 47–54.

[HPS07] K. K. R. G. K. Hewawasam, Kamal Premaratne, and M.-L. Shyu, *Rule mining and classification in a situation assesment application: A belief theoretic approach for handling data imperfections*, IEEE Transactions on Systems, Man and Cybernetics, Part B Cybernetics **37** (2007), no. 6, 1446–1459.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin, *Mining frequent patterns without candidate generation*, 2000 ACM SIGMOD Intl. Conference on Management of Data, ACM Press, May 2000, pp. 1–12.

[JGB01] A.L. Jousselme, D. Grenier, and E. Bosse, *A new distance between two bodies of evidence*, Information Fusion **2** (2001), 91–101.

[KCT+01] L.A. Kurgan, K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L.S. Goodenday, *Knowledge discovery approach to automated cardiac SPECT diagnosis*, Artificial Intelligence in Medicine **23** (2001), no. 2, 149–169.

[KHR+03a] M. Kubat, A. Hafez, V. V. Raghavan, J. R. Lekkala, and Wei Kian Chen, *Itemset trees for targeted association querying*, IEEE Transactions on Knowledge and Data Engineering **15** (2003), no. 6, 1522 – 1534.

[KHR+03b] M. Kubat, A. Hafez, V.V. Raghavan, J. Lekkala, and W.K. Chen, *Itemset trees for on-line association mining*, IEEE Transactions on Data and Knowledge Engineering **15** (2003), 1522–1534.

[LHM98] B. Liu, W. Hsu, and Y.M. Ma, *Intergraing classification and association rule mining*, Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'98) (New York, NY), August 1998, pp. 80–86.

[LHP01] W. Li, J. Han, and J. Pei, *CMAR: Accurate and efficient classification based on multiple class-association rules*, Proceedings IEEE International Conference on Data Mining (ICDM'01) (San Jose, CA), Nov./Dec 2001, pp. 369–376.

[LK06] Yu Li and Miroslav Kubat, *Searching for high-support itemsets in itemset trees*, Intell. Data Anal. **10** (2006), no. 2, 105–120.

[LMW00] Bing Liu, Yiming Ma, and Ching Kian Wong, *Improving an association rule based classifier*, Principles of Data Mining and Knowledge Discovery, 2000, pp. 504–509.

[LSW97] B. Lent, A. Swami, and J. Widom, *Clustering association rules*, Proceedings 13th International Conference on Data Engineering, 1997, April 1997, pp. 220–231.

[MFM+98] Yasuhiko Morimoto, Takeshi Fukuda, Hirofumi Matsuzawa, Takeshi Tokuyama, and Kunikazu Yoda, *Algorithms for mining association rules for binary segmentations of huge categorical databases*, VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1998, pp. 380–391.

[Mor98] Shinichi Morishita, *On classification and regression*, Discovery Science, 1998, pp. 40–57.

[NP28] J. Neyman and E.S. Pearson, *On the use and interpretation of certain test criteria for purposes of statistical inference*, Biometrica **20A** (1928), 175–240.

[NRC01] S. Noel, V.V. Raghavan, and C.H. Chu, *Visualizing association mining results through hierarchical clusters*, Proceedings of the 2001 International Conf. on Data Mining (ICDM-01) (San Jose, CA), Nov. - Dec. 2001, pp. 425–432.

[Res07] GroupLens Research, *Movielens data sets*, 2007.

[RH00] V.V. Raghavan and A. Hafez, *Dynamic data mining*, Proceedings 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE (New Orleans, Louisiana), June 2000, pp. 220–229.

[RK05] A. Rozsypal and M. Kubat, *Association mining in time-varying domains*, Intelligent Data Analysis **9** (2005), 273–288.

[Sha76] G. Shafer, *A mathematical theory of evidence*, Princeton University Press, 1976.

[Sme90] Philippe Smets, *Constructing the pignistic probability function in a context of uncertainty*, UAI 89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (Amsterdam, The Netherlands), North-Holland Publishing Co., 1990, p. 2940.

[Sme99] P. Smets, *Practical uses of belief functions*, Proc. Conference on Uncertainty in Artificial Intelligence (UAI'99) (K. B. Laskey and H. Prade, eds.), Morgan Kaufmann, 1999, pp. 612–621.

[SZP+08] S.P. Subasingha, J. Zhang, K. Premaratne, M.-L. Shyu, M. Kubat, and K.K.R.G.K. Hewawasam, *Using association rules for classification from databases having class label ambiguities: Belief theoretic method (to appear)*, 2008.

[VCS04] D. Vilar, M. J. Castro, and E. Sanchis, *Multilabel text classification using multinomial models*, Proc. Espan a for Natural Language Processing (EsTAL'04) (Alicante, Spain), Oct 2004, pp. 220–230.

[vR79] C. J. van Rijsbergen, *Information retireval*, Butterworths, London, UK, 1979.

[Wic08] T. L. Wickramaratna, *A belief theoretic approach for automated collaborative filtering*, Master's thesis, Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL, 2008.

[ZSP+04a] J. Zhang., S.P. Subasingha, K. Premaratne, M.-L. Shyu, M. Kubat, and K.K.R.G.K. Hewawasam, *A novel belief theoretic association rule mining based classifier for handling class label ambiguities*, Workshop on Foundations of Data Mining (FDM'04), International Conference on Data Mining, (ICDM'04) (Brighton, UK), November 2004.

[ZSP+04b] J. Zhang, S.P. Subasingha, K. Premaratne, Mei-Ling Shyu, M. Kubat, and KKRGK Hewawasam, *A novel belief theoretic association rule mining based classifier for handling class label ambiguities*, the Third Workshop in Foundations of Data Mining (FDM04), in conjunction with the Fourth IEEE International Conference on Data Mining (ICDM04) (Brighton, UK), November 2004, pp. 213–222.