HANDLING MISSING ATTRIBUTE VALUES IN DECISION TABLES USING VALUED TOLERANCE APPROACH

Supriya Vasudevan

BTech., Information Technology
Anna University, India, 2006

Submitted to the Department of Electrical Engineering & Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Thesis Committee:

_____
Dr. Jerzy Grzymala-Busse: Chairperson

_____
Dr. Jun Huan: Member

_____
Dr. Prasad Kulkarni: Member

_____
Date Defended

The Thesis Committee for Supriya Vasudevan certifies that this is the approved version of the following thesis:

Handling missing attribute values in decision tables
using valued tolerance approach

Thesis Committee:

_____
Dr. Jerzy Grzymala-Busse: Chairperson

_____
Dr. Jun Huan: Member

_____
Dr. Prasad Kulkarni: Member

_____
Date Defended

ABSTRACT

Rule induction is one of the key areas in data mining as it is applied to a large number of real life data. However, in such real life data, the information is incompletely specified most of the time. To induce rules from these incomplete data, more powerful algorithms are necessary. This research work mainly focuses on a probabilistic approach based on the valued tolerance relation.

This thesis is divided into two parts. The first part describes the implementation of the valued tolerance relation. The induced rules are then evaluated based on the error rate due to incorrectly classified and unclassified examples. The second part of this research work shows a comparison of the rules induced by the MLEM2 algorithm that has been implemented before, with the rules induced by the valued tolerance based approach which was implemented as part of this research. Hence, through this thesis, the error rate for the MLEM2 algorithm and the valued tolerance based approach are compared and the results are documented.

ACKNOWLEDGEMENTS:

I would like to express my heartfelt thanks to my advisor and thesis committee chair, Dr.Jerzy Grzymala Busse for his continued support and encouragement. The knowledge he imparted on me through his data mining lectures and during research discussions has been invaluable and helped me complete this thesis.

I would like to thank Dr. Jun Huan and Dr.Prasad Kulkarni for consenting to be members of this thesis committee. I extend my thanks to my parents for their constant support and encouragement in every aspect of my graduate education.

I would also like to thank Sathya for helping me during the documentation and presentation areas of this thesis. Not to forget, I am also thankful to all my friends and peers of The University of Kansas.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

In data mining different algorithms exist to process large amounts of data. However, real life data are characterized by missing values which renders the classification process difficult.

1.1 Motivation

The data mining algorithms are used for classifying critical and sensitive real life data. Hence, accurate classification becomes very essential. Many different algorithms were proposed to address this issue. The technique of learning from inconsistent examples, which is based on the concept of rough set theory, is one such example.

Rule induction algorithms deal with missing examples in two ways. One is by preprocessing the incomplete decision table to make it complete and then performing rule induction on it. The other is by performing rule induction on incomplete data directly. MLEM2 algorithm that is based on rough set theory is one such rule induction algorithm which acts on missing examples directly. When an incomplete decision table is provided, rough set theory handles it by approximating the inconsistent or missing data with two approximations, lower and upper. Similar to MLEM2, a valued tolerance relation, as introduced by Jerzy Stefanowski and Alexis Tsoukias, works on the incomplete decision tables directly. The valued tolerance approach is based on a probabilistic relation and captures the similarity between objects more intuitively [2].

Results on experiments published so far on the valued tolerance approach have no mention of the accuracy of the induced decision rules. This research work mainly focuses on implementing the valued tolerance approach for rule induction and evaluating the induced decision rules based the error rate. These results are then compared with the existing results of the MLEM2 algorithm.

1.2 Organization of the document

This thesis is organized into six chapters that include a detailed background of rule induction, a description of the MLEM2 and valued tolerance approaches, the related experimentation and results.

Chapter1 gives an overview of the motivation for the thesis. An overall idea about the topic and the main focus of the thesis is explained.

Chapter 2 provides the background information related to the thesis. It is divided into two parts. The first part explains some basic concepts related to rule induction. It also explains the concept of missing attribute values in decision tables and the different approaches to induce rules from these incomplete decision tables. The second part describes rough set approach and the related concepts in detail. Finally, it describes the MLEM2 algorithm with an example on how to induce rules from incomplete decision tables.

Chapter 3 introduces the tolerance and valued tolerance relations. It outlines the rule induction process using the valued tolerance approach. This chapter also describes the implementation of the valued tolerance approach.

Chapter 4 deals with the experiments done for both approaches. It explains the tools used for evaluating the decision rules and also provides the results of a comparative study between the valued tolerance algorithm and the MLEM2 algorithm.

Chapter 5 contains the conclusions of the thesis and also gives suggestions for future areas of exploration.

CHAPTER 2: BACKGROUND

This chapter gives an overview of the fundamental concepts of rough set theory and the MLEM2 algorithm.

2.1 Decision tables and decision rules

In the field of data mining the information available about real world data is represented in the form of a decision table. A decision table is a powerful and clear way of representing data.

2.1.1 Information representation in decision tables

A decision table describes a set of examples. Each example is presented in the form of a set of attribute values and a decision value. The attribute values provide information about the object and the decision value provides a way of categorizing the object. Also, each example belongs to a class, known as the concept. Such a class is represented by a set of examples having the same value for a decision. A decision table describing cars is shown in Table 2.1.

Table 2.1: An example of the decision table

Case	Year	Mileage	Decision
1	1990	300000	Least Reliable
2	2000	150000	Less Reliable
3	2008	6000	Very Reliable

In the above decision table, Least Reliable, Less Reliable and Very Reliable are concepts about the car. Mileage and Year are attributes that describe the car.

Decision tables are a precise yet compact way to model complicated situations. In mathematical terms, a decision table 'DT' is an information table of the form (U, C ∪ {d}) where d is a distinguished attribute called a decision, U is a set of objects and elements of C are called condition attributes.

2.1.2 Decision rules

Decision rules classify the data in the decision table into different concepts. Usually decision rules are expressed in the form:

If (attribute1 = value1) & (attribute2 = value2) - > (decision = value)

The left hand side represents the attribute-value pairs and the right hand side represents the concept. A case x is covered by a rule r if and only if every condition or attribute-value pair of r is satisfied by the corresponding attribute value for x. For example, from Table 2.1, the following rule can be extracted

If (Year = 2000) -> (Decision = Less Reliable)

Since case 2 with the concept "Less Reliable" satisfies this rule, it is said to be covered by the rule.

Decision rules can be described using two important terms – completeness and consistency. The basic idea is to compute rules which are both complete and consistent. Such a rule set R is called discriminant. Consider a concept C. This is completely covered by a rule set R if for every case x from C there exists a rule such that r covers x. A rule set R is complete if every concept from the data set is completely covered by R. It is consistent with the data set, if every case x is

covered by R, where x is a member of the concept C indicated by R. In other words, a rule set R is consistent if and only if every rule $r \in R$ is consistent with the data set.

Decision rule induction is the process by which rules are induced from the decision tables. It involves extraction of high level information from low level data and is the most fundamental data mining technique. Examples for rule induction algorithms are LEM1 and LEM2 [6].

2.1.3 Handling missing attribute values in decision tables

Until recently, data represented in decision tables were assumed to be complete and consistent. In recent times, the need for new algorithms to deal with uncertain input, lost input or erroneous information has risen. Hence, handling missing attribute values is an active topic of research in data mining. One of the general assumptions made in all these rule induction algorithms is that, in a particular case at least one value should be present which means we cannot induce a rule from a case which has all of its attributes missing [2].

In general, it is assumed that there are three specific types of missing data:

(1) The attribute value might be lost, i.e., either erased or missed out. These are called "lost conditions", represented by '?'.

(2) The attribute value might be irrelevant to the current classification. In other words, the value is unnecessary. So, we do not care if they are missing or not. Hence, they are called "do not care" conditions. These missing attribute values are represented by '*'.

(3) The missing attribute values are assumed to belong to a particular concept. These are called "attribute-concept" values, represented by '-'.

The theories that provide solution to the missing attribute value problem follow two main approaches.

1) The input data is preprocessed, that is the missing attribute values are filled up using some heuristics that the application demands. Some examples are: replacing the missing attribute value with the most frequently occurring value, replacing the missing attribute value with the attribute average, replacing the missing attribute value with the most frequently occurring value in that concept etc. Hence, the data becomes complete after preprocessing. Now, the rule induction becomes the same as the traditional approach [5].

2) Inducing rules with the incompletely specified table itself. Examples for this approach are MLEM2 and C4.5 algorithms [2].

Table2.2 illustrates the three types of missing data using an example.

Table 2.2: An incomplete decision table: example 1

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	High	-	No	Yes
2	Very_high	Yes	Yes	Yes
3	?	No	No	No
4	High	Yes	Yes	Yes
5	High	?	Yes	No
6	Normal	Yes	No	No
7	Normal	No	Yes	No
8	*	Yes	*	Yes

In the above table, there are three attributes namely 'Temperature', 'Headache', 'Nausea' and one decision 'Flu'. It has eight cases of which four have missing attribute values. Cases 3 and 5 have "lost conditions", case 8 has two "do not care" conditions and case 1 has an "attribute-concept" value. There are two concepts 'Yes' and 'No' for classification.

2.2 Rough set theory

Rough set theory, introduced by Z.Pawlak, brings a mathematical approach to data mining. It has evolved into a very powerful tool for analyzing the ambiguity of data elements and has served as the foundation for innumerable theories. It has also proved its usefulness in many real life applications.

2.2.1 Preliminary concepts of rough set theory

The main idea behind rough set theory is the indiscernibility relation between objects which can be explained as follows:

Any two objects in the universe can be assumed to be indiscernible if the available or known facts about them are similar. In other words, they are categorized based on the available information about them. The unknown information might show them to be different. But according to the facts available, they are classified to be similar [11, 12].

In mathematical terms, the indiscernibility relation can be explained as: A decision table IT is a pair (U, C) where U is a non empty set of objects and C is a non empty set of attributes [3, 14, 15]. For each subset of attributes $B \subseteq C$, the indiscernibility relation IND (B) can be defined as:

$$IND(B) = \{(x, y) \in U \times U : \forall c_i \in B, c_i(x) = c_i(y)\}$$

where x and y are two objects and $c_i(x)$ and $c_i(y)$ are the values of attribute c_i for x and y respectively. According to rough set theory, with each set, a pair of precise sets called lower and upper approximation is associated. For any element x of U, the equivalence class of IND containing x will be denoted by $[x]_{IND}$. A lower approximation of X in (U, IND), denoted by $\underline{R}X$, is the set

$$\{x \in U \mid [x]_{IND} \subseteq X\}.$$

An upper approximation of X in (U, IND), denoted by $\overline{R}X$, is the set

$$\{x \in U \mid [x]_{IND} \cap X \neq \varnothing\}.$$

Lower approximation represents data that certainly belong to the set and hence the rules extracted using lower approximation are called certain rules. Upper

approximation represents data that may possibly belong to the set and hence the rules induced this way are called possible rules [3, 16].

Let U be a non-empty set called the universe and let IND be the indiscernibility relation. An ordered pair (U, IND) is called an approximation space. Let X be a subset of U. X is defined in terms of the set in (U, IND). Any finite union of elementary sets in (U, IND) is called the definable set in (U, R). The lower approximation of X in (U, R) is the greatest definable set in (U, IND), contained in X. The upper approximation of X in (U, IND) is the least definable set in (U, IND), containing X [1].

The main idea behind the approximations is that if it looks impossible to completely categorize a set of objects using the available information, it can be approximated using the lower and upper approximations. The set that separates the lower and the upper approximation is the boundary region for the rough set. The main advantage of rough set theory is that it does not require any additional information about data, unlike a lot of statistical approaches. Thus, it is possible to find hidden information about data more efficiently [2].

2.2.2 Rule induction algorithms based on rough set theory

Rule induction is one of the most important techniques of data mining that extracts regularities from the real life data. The critical task of rule induction is to induce a rule set R that is consistent and complete. That said, rule induction has its own challenges. For example, the input data can be affected by errors, may

contain numeric attributes (which need to be converted to symbolic attributes before or during rule induction), may be incomplete due to missing values or inconsistent due to two cases having the same attribute values but a different decision value. There are two important kinds of rule induction algorithms, global and local. In global algorithms all the attribute values are considered where as in local algorithms the concept of attribute-value pairs come into picture. The two important rule induction algorithms based on rough set theory are LEM1 (Learning by Examples) and LEM2 [6].

Let B be a nonempty subset of a set C of all attributes. LEM1 is a global algorithm that is based on rough set theory and uses the indiscernibility concept IND (B). The family of all B-elementary sets will be denoted by B*. According to the definition of LEM1, for a decision *d* we say that *{d}* depends on *B* if and only if *B** <= *{d}** where {d}* is the family of elementary sets of d. A global covering of *{d}* is a subset *B* of C such that *{d}* depends on *B* and *B* is minimal in C. The main idea behind LEM1 is to compute a single global covering. The computed rules then undergo a process called dropping conditions. This is done by scanning through all the computed rules, dropping conditions one by one and then, checking if the consistency of the rule set is affected by dropping the condition. If the rules with dropped conditions are consistent and cover the same number of cases, then the condition can be termed as a redundant condition and dropped forever.

On the other hand, LEM2 is a local algorithm, which takes into account attribute-value pairs and then converts them into a rule set. LEM2 works better than

LEM1 because it works on lower and upper approximations separately and hence the input files for LEM2 are always consistent.

For an attribute-value pair (a, v) = t, a block of t, denoted by [t], is a set of all cases from the universe U which have attribute a and value v. Let B be a nonempty lower or upper approximation of a concept represented by a decision-value pair (d, w). Set B depends on a set T of attribute-value pairs t = (a, v) if and only if no proper subset T′ of T exists such that B depends on that subset T′.

$$\varnothing \neq [T] = \cap_{t \in T} [t] \subseteq B$$

Thus in LEM2, the first step is to compute a set of all attribute-value blocks [a, v]. After that, an iterative process is followed to identify the minimal complex. A detailed description of the LEM2 procedure is given in Appendix A [6].

2.2.3 Classification system in LERS

LERS (Learning from Examples based on Rough Sets) is a well known data mining system that induces rules based on rough set theory and uses the already induced rules to further classify more data. LERS associates three basic parameters to evaluate the decision rules namely strength, specificity and support.

- *Strength* is defined as the total number of examples correctly classified by the rule during training. It is a measure of how well the rule performed.

- *Specificity* is the total number of conditions or attribute value pairs on the left hand side of the rule. Thus, rules with more attribute-value pairs are considered to be more specific.
- *Support* is defined as the sum of scores of all matching rules from the concept [8, 9].

2.2.4 Characteristic sets and characteristic relations for incomplete decision tables

For decision tables with missing attribute values, modified definitions are needed to determine to which attribute-value block the cases with missing attribute values can be added. The following heuristics are followed:

- If for an attribute a there exists a case x such that the value of a for x is '?', i.e., the corresponding value is lost, then the case x should not be included in any block $[(a, v)]$ for all values v of attribute a.
- If for an attribute a there exists a case x such that the value of a for x is '*' i.e., the corresponding value is a do not care condition, then the case x should be included in blocks $[(a, v)]$ for all specified values v of attribute a.
- If for an attribute a there exists a case x such that the value is a '-', i.e., the corresponding value is an attribute concept value then the case x should be included in blocks $[(a, v)]$ for all specified values v of attribute a belonging to that concept [2].

For example, the attribute-value block for Table 2.2 can be defined as:

[(Temperature, High)] = {1, 4, 5, 8},

[(Temperature, Very_high)] = {2, 8},

[(Temperature, Normal)] = {6, 7},

[(Headache, Yes)] = {1, 2, 4, 6, 8},

[(Headache, No)] = {3, 7},

[(Nausea, No)] = {1, 3, 6, 8},

[(Nausea, Yes)] = {2, 4, 5, 7, 8}.

A few important terms that need to be defined in rough set theory are characteristic sets and characteristic relations. The characteristic set $K_B(x)$ for a set B, is the intersection of blocks of attribute-value pairs (a, v) for all attributes a from B for which $\rho(x, a)$ is specified and $\rho(x, a) = v$. For Table 2.2, the characteristic set can be defined for $B = A$ as:

For Case 1: $K_A(1) = \{1, 4, 5, 8\} \cap \{1, 3, 6, 8\} = \{1, 8\}$,

For Case 2: $K_A(2) = \{2, 8\} \cap \{1, 2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}$,

For Case 3: $K_A(3) = \{1, 2, 3, 4, 5, 6, 7, 8\} \cap \{3, 7\} \cap \{1, 3, 6, 8\} = \{3\}$,

For Case 4: $K_A(4) = \{1, 4, 5, 8\} \cap \{1, 2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 8\}$,

For Case 5: $K_A(5) = \{1, 4, 5, 8\} \cap U \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,

For Case 6: $K_A(6) = \{6, 7\} \cap \{1, 2, 4, 6, 8\} \cap \{1, 3, 6, 8\} = \{6\}$,

For Case 7: $K_A(7) = \{6, 7, 8\} \cap \{3, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}$,

For Case 8: $K_A(8) = \{1, 4, 8\} \cap \{2,8\} \cap \{1,2,4,6,8\} = \{1,2,4,8\}$.

Thus, for a given interpretation of missing attribute values, the characteristic set $K_B(x)$ may be interpreted as the smallest set of cases that are indistinguishable from x [2, 10].

The characteristic relation R (B) is a relation on a set of all objects U defined for objects x, $y \in$ U as follows

(x, y) $\in$ R (B) if and only if $y \in K_B(x)$

The characteristic relation is known if we know all the characteristic sets x $\in$ U. For Table 2.2, the characteristic relation can be calculated as:

$R(A)$ = {(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (6, 8), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)}.

2.2.5 Rough set theory in incomplete decision tables

In rough set theory incomplete decision tables are described using characteristic relations, in the same way complete decision tables are described using the indiscernibility relation. For a complete decision table, once the indiscernibility relation is fixed and the concept is given, the lower and upper approximations are unique. For an incomplete decision table, for a given characteristic relation and concept there are three different ways to define lower and upper approximations namely singleton, subset and concept.

In the case of singleton method, the singleton B-lower and B-upper approximations of X are defined as follows:

$$\underline{B}X = \{x \in U \mid K_B(x) \subseteq X\}$$

$$\overline{B}X = \{x \in U \mid K_B(x) \cap X \neq \emptyset\}$$

where $K_B(x)$ is the characteristic set and U is the set of all objects.

For example, in Table 2.2 the singleton *A*-lower and *A*-upper approximations of the two concepts {1, 2, 4, 8} and {3, 5, 6, 7} are:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 5, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 5, 6, 7}.

Singleton approximations are not very useful since they cannot be represented as union of attribute value blocks. So, the other approach uses characteristic relations instead of elementary sets. There are two different ways to do this. The first method is called subset approximations.

A subset *B*-lower approximation of *X* is defined as:

$$\underline{B}X = \cup\{K_B(x) \mid x \in U, K_B(x) \subseteq X\}$$

A subset *B*-upper approximation of *X* is defined as:

$$\overline{B}X = \cup\{K_B(x) \mid x \in U, K_B(x) \cap X \neq \emptyset\}$$

For example, for Table 2.2, the subset lower and upper approximations are given as follows:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 5, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 4, 5, 6, 7, 8}.

The second method is by changing the subset definition by replacing the universe U with the concept C. A concept *B*-lower approximation of the concept *X* is defined as:

$$\underline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \subseteq X\}$$

A concept *B*-upper approximation of the concept *X* is defined as:

$$\overline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \cap X \neq \emptyset\} = \cup\{K_B(x) \mid x \in X\}$$

For example, in Table 2.2, the concept lower approximations for B = A are:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 4, 5, 6, 7, 8}.

For completely defined decision tables, singleton, subset and concept lower and upper approximations are the same. As explained above, they are different for incomplete decision tables [2, 4, 10].

2.2.6 MLEM2 algorithm

Most of the data mining algorithms are not designed to deal with numerical attributes. They take only symbolic (alphanumeric) attributes as inputs. So, to

classify data containing numerical attributes, they have to be converted to symbolic attributes as a preprocessing step. Most of the data mining rule induction algorithms take the numerical data and preprocess it by a discretization process and then conduct the rule induction process. However, this approach doubles the processing time. So, the currently developed algorithms conduct rule induction and discretization at the same time. Examples for the latter approach are Modified LEM2 (MLEM2), C 4.5 etc [5, 9, 10].

The key focus in this thesis is the modified LEM2 (MLEM2) which is based on the LEM2 algorithm. MLEM2 classifies all attributes into two major types namely numerical and symbolic. Approximations are computed in different ways for numerical and symbolic attributes. The procedure is as follows: the first step is to sort all values of a numerical attribute. The next step is to compute the average of any two consecutive values of the sorted list. For each cut point x, MLEM2 works by creating two blocks, the first block containing the values smaller than x and the second block containing the values larger than x. The search space of MLEM2 includes both the blocks computed from symbolic attributes as well as the blocks computed from numerical attributes. From this point, the rule induction follows the same procedure as the LEM2 algorithm [9, 10].

With the background literature reviewed, the next chapter deals with the main theoretical aspects of the thesis.

CHAPTER 3: VALUED TOLERANCE RELATION

This chapter gives an overview of rule induction using the valued tolerance relation.

3.1 Introduction to valued tolerance relation

Before we describe the valued tolerance relation, we need to define the tolerance relation. Given a decision table IT = (U, C), where U is a set of objects, C is a set of attributes and a subset of attributes $B \subseteq C$, tolerance relation $T_B(x, y)$ for can be described as the following binary relation,

$$T_B(x, y) = \forall c_j \in B, c_j(x) = c_j(y) \text{ or } c_j(x) = * \text{ or } c_j(y) = *$$

where x and y are two objects and $c_j(x)$ and $c_j(x)$ are the values of attribute c_j for x and y respectively. The tolerance relation is the characteristic relation for "do not care" conditions which are represented by '*'. Let us denote $I_B(x)$ to be the set of objects y for which $T_B(x, y)$ holds. The set $I_B(x)$ is called the tolerance class of x.

Based on the tolerance relation, B-lower and B-upper approximations of X can be defined as:

$$\text{Lower approximation } \underline{B}X = \{x \in U | I_B(x) \subseteq X\}$$

$$\text{Upper approximation } \overline{B}X = \{x \in U | I_B(x) \cap X = \varnothing\}$$

We can see that the definitions of the lower and upper approximations are the same as singleton lower and upper approximations with the characteristic relation T_B.

Table 3.1: An incomplete decision table: example 2

Case number	Attributes				Decision
a1	3	2	1	0	Φ
a2	2	3	2	0	Φ
a3	2	3	2	0	Ψ
a4	*	2	*	1	Φ
a5	*	2	*	1	Ψ
a6	2	3	2	1	Ψ
a7	3	*	*	3	Φ
a8	*	0	0	*	Ψ
a9	3	2	1	3	Ψ
a10	1	*	*	*	Φ
a11	*	2	*	*	Ψ
a12	3	2	1	*	Φ

Table 3.1 shows an incompletely specified decision table. It has twelve examples and four attributes with only do not care conditions in missing attribute values. Based on Table 3.1**,** we can deduce the tolerance relation as:

$I_B(a1)$ = {a1, a11, a12},

$I_B(a2)$ = {a2, a3},

$I_B(a3)$ = {a2, a3},

$I_B(a4)$ = {a4, a5, a10, a11, a12},

$I_B(a5)$ = {a4, a5, a10, a11, a12},

$I_B(a6)$ = {a6},

$I_B(a7)$ = {a7, a8, a9, a11, a12},

$I_B(a8)$ = {a7, a8, a10},

$I_B(a9)$ = {a7, a9, a11, a12},

$I_B(a10)$ = {a4, a5, a8, a10, a11},

$I_B(a11)$ = {a1, a4, a5, a7, a9, a10, a11, a12},

$I_B(a12)$ = {a1, a4, a5, a7, a9, a11, a12}.

From this, the lower and upper approximations can be deduced as

Lower approximation $\underline{A}\{a1, a2, a4, a7, a10, a12\} = \varnothing$,

Upper approximation

$\overline{A}\{a1, a2, a3, a4, a7, a10, a12\} = \{a1, a2, a3, a4, a5, a7, a8, a9, a10, a11, 12\}$,

Lower approximation $\underline{A}\{a3, a5, a6, a8, a9, a11\} = \{a6\}$,

Upper approximation

$\overline{A}\{a3, a5, a6, a8, a9, a11\} = \{a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, 12\}$.

With the tolerance relation explained, we now describe the valued tolerance relation. Consider three objects A, B and C. Suppose B has two attribute values missing and C has only one attribute value missing. The valued tolerance helps capture the fact that C is more similar to A than B. Thus the valued tolerance relation is more intuitive than the tolerance relation.

3.1.1 Lower and upper approximations based on valued tolerance relation

Consider Table 3.1 with missing attributes values. The valued tolerance approach deals with only the '*' values, that is, the "do not care" conditions. It is a probability-based relation. A general assumption made by the valued tolerance approach is that there is a uniform probability distribution among the set of all attributes and the set of values that each attribute has is discrete. Consider a decision table IT (U, C). For every attribute c_j, we can associate a set E_j such that it contains all the possible values of the attribute $E_j = \{e^1{}_j, e^2{}_j, e^3{}_j \ldots\ldots\ldots e^n{}_j\}$.

For a given object $x \in U$, the probability that $c_j(x)$ is equal to $e^1{}_j$ is $\frac{1}{|E_j|}$. Consider two objects $x, y \in U$ and an attribute $c_j \in C$. There are four different situations:

(a) If the values of x and y are known and $c_j(x) = c_j(y)$, then the probability that both are same in terms of c_j is

$$R_{c_j}(x, y) = 1$$

(b) If the values of x and y are known and $c_j(x) \neq c_j(y)$, then the probability that both are same in terms of c_j is

$$R_{c_j}(x, y) = 0$$

(c) For one of the objects, the value of c_j is known, say $c_j(x) = e^i{}_j$, then the probability that x and y are same in terms of c_j is

$$R_{c_j}(x, y) = \frac{1}{|E_J|}$$

(d) If for both the objects, the value of c_j is unknown, then the probability that x and y are same in terms of c_j is

$$R_{c_j}(x, y) = \frac{1}{|E_J|^2}$$

The joint probability for all the attributes $C = \{c^1{}_j, c^2{}_j, c^3{}_j \ldots\ldots\ldots c^n{}_j\}$, that an object x is the same as another object y can be computed as:

$$R_C(x, y) = \prod_{cj \in C} R_{c_j}(x, y)$$

The joint probability $R_C(x, y)$ is the valued tolerance relation for the decision table. We know that if two objects have at least one attribute value different, then the joint probability is going to be 0. Thus, the first step in rule induction using valued tolerance approach is to calculate the valued tolerance relation table. For example, for Table 3.1, the set of all possible values for all the attributes are given by:

a1= {1, 2, 3},

a2= {0, 2, 3},

a3= {0, 1, 2},

a4= {0, 1, 3}.

1. Probability that a1 is the same as a2 is given by 0 (an example of case {b}).
2. Probability that a2 is the same as a3 is given by 1 (an example of case {a}).
3. Probability that a4 is the same as a11 is given by

   (1/3*1/3) * 1 * (1/3*1/3) * (1/3) = (1/243)

Thus, the valued tolerance relation for the entire decision table is computed in Table 3.2. It is obvious to note that the relation is symmetric**.** We can see that the values in the shaded region and the non shaded region are the same.

Table 3.2: The valued tolerance relation table

	a1	**A2**	**a3**	**a4**	**a5**	**A6**	**a7**	**a8**	**a9**	**a10**	**a11**	**a12**
a1	1	0	0	0	0	0	0	0	0	0	0.037	0.33
a2	0	1	1	0	0	0	0	0	0	0	0	0
a3	0	1	1	0	0	0	0	0	0	0	0	0
a4	0	0	0	1	0.012	0	0	0	0	0.004	0.004	0.03
a5	0	0	0	0.012	1	0	0	0	0	0.004	0.004	0.037
a6	0	0	0	0	0	1	0	0	0	0	0	0
a7	0	0	0	0	0	0	1	0.012	0.11	0	0.004	0.037
a8	0	0	0	0	0	0	0.012	1	0	0.004	0	0
a9	0	0	0	0	0	0	0.11	0	1	0	0.037	0.33
a10	0	0	0	0.004	0.004	0	0	0.004	0	1	0.001	0
a11	0.037	0	0	0.004	0.004	0	0.004	0	0.037	0.001	1	0.12
a12	0.33	0	0	0.03	0.037	0	0.037	0	0.33	0	0.12	1

Given a set of objects U, a subset of attributes B $\subseteq$ C, and an object a1, the lower and upper approximations for a set X are given as:

Lower approximation => $\mu_{\underline{BX}}(a1) = \prod_{x \in IND(a1)} (1 - R_c(a1, x) + R_c(a1, x)\hat{x})$

Upper approximation => $\mu_{\overline{BX}}(a1) = 1 - \prod_{x \in IND(a1)} (1 - R_c(a1, x)\hat{x})$

where IND (a1) is the tolerance class of object a1, Rc (a1, x) is the joint probability that a1 is the same as x and $\hat{x}$ refers to the implication to the set Φ. In other

words, $\hat{x}$ defines the membership to sets Φ and Ψ, $\hat{x} \in \{0, 1\}$. For the Table 3.1, $\hat{x}$ takes the values (1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1) for the set Φ and the values (0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0) for the set Ψ. Thus, based on the formula defined above the lower and upper approximations for Table 3.1 are given in Table 3.3

Table 3.3: Lower and upper approximation for valued tolerance

Attributes	Lower approximation for set Φ	Upper approximation for the set Φ	Lower approximation for the set Ψ	Upper approximation for the set Ψ
a1	0.97	0	1	0.03
a2	0	0	1	1
a3	0	0	1	1
a4	0.98	0	1	0.01
a5	0	0.947	0.05	1
a6	0	1	0	1
a7	0.87	0	1	0.12
a8	0	0.98	0.016	1
a9	0	0.59	0.4	1
a10	0.99	0	1	0.01
a11	0	0.94	0.05	1
a12	0.63	0	1	0.3

3.1.2 Decision rule induction based on the valued tolerance relation

In this approach, the decision rules are accepted only when they are above a certain threshold called the credibility degree. This is based on the fact that objects may be similar to the condition part of the rule only to a certain degree. Also, the decision part in the rule is considered to be uncertain.

Thus, we can define rule ρ_i such that an element 'x' supports the rule ρ_i .Here, x is similar to the condition part of the rule to a certain extent. Thus, we have a credibility degree associated with each rule which can be defined as:

$$\mu(\rho_i) = \prod_{x \in S} (1 - S_B(x, \rho_i) + S_B(x, \rho_i)\mu_{\underline{B}X}) \text{ for certain rules,}$$

$$\mu(\rho_i) = \prod_{x \in S} (1 - S_B(x, \rho_i) + S_B(x, \rho_i)\mu_{\overline{B}X}) \text{ for possible rules.}$$

Here $S_B(x, \rho_i)$ represents the support degree of element x to the rule ρ_i. In other words, it is the tolerance degree of each element with respect to the condition part of the rule. The $\mu_{\underline{B}X}$ and $\mu_{\overline{B}X}$ values represent the lower and upper approximation values calculated for a particular concept for the set X.

Further, the rules have to be non-redundant and minimal. So the rules are filtered based on the credibility degree threshold. Those that are higher than the credibility degree threshold undergo a process called dropping conditions. Rules can be generated with shorter condition parts, provided the credibility is still above the threshold.

An example of decision rule induction from Table 3.1 is given below:

Consider the cases belonging to the set Φ. Among these we consider only the cases with no missing attribute values. Thus, if we consider the values for the case a1, the rule candidate ρ_i is

$$\rho_i = (a1 = 3) \wedge (a2 = 2) \wedge (a3 = 1) \wedge (a4 = 0) -> (d = \Phi)$$

The support and the approximation values for this rule are given in Table 3.4 Therefore, using the above formula we can calculate the credibility degree for the rule ρ_i to be $\mu(\rho_i) = 0.8$.

Table 3.4: Credibility degree table

	$S_B(x, \rho_i)$	$\mu_{\underline{B}X}$
a1	1	0.97
a11	0.0156	0
a12	0.25	0.63

If this $\mu(\rho_i)$ is above the user specified threshold, then the rule is further processed by dropping conditions. Similarly, rule induction should be done for all the rules in Ψ and Φ.

3.1.3 Dropping conditions

The simplification process called dropping the redundant conditions is explained below:

Only those rules which have credibility degree $\mu(\rho_i)$ above the threshold undergo the dropping conditions process. Other rules which do not meet the credibility degree criteria are discarded. For example, for the Table 3.1, for the rule

$$\rho_i = (a1 = 3) \wedge (a2 = 2) \wedge (a3 = 1) \wedge (a4 = 0) -> (d = \Phi)$$

The condition a1=3 is dropped and the whole process of calculating lower and upper approximations and rule induction is repeated and the credibility degree is calculated for the rule

$$(a2 = 2) \wedge (a3 = 1) \wedge (a4 = 0) -> (d = \Phi)$$

If this credibility degree is below the threshold, then the condition a1=3 cannot be discarded and is put back into the rule. If this reduced rule qualifies for further simplification, then the whole process is repeated again by eliminating the second condition (a2=2) which further reduces the rule to

$$(a3 = 1) \wedge (a4 = 0) -> (d = \Phi)$$

This rule is again reduced to remove the third condition a3=1 (if credibility degree is above the threshold) which further reduces the rule to

$$(a4 = 0) -> (d = \Phi)$$

At any point in this reduction process, if the credibility threshold is not satisfied by removing a condition, the condition is put back in the rule. This process is done for all candidate rules. The same method is carried out using lower and upper approximations to calculate the certain rules and possible rules respectively.

3.2 Implementing the valued tolerance approach

The implementation of the valued tolerance approach is the primary focus of this thesis. This implementation is carried out using C/C++. The program uses vectors to store and access the large input data file. An example input file will have the following format:

< a a a a d >
<!this is a comment>
[a1 a2 a3 a4 d]
3 2 1 0 low
2 3 2 0 low

……..
……

The first line indicates that there are four attributes followed by a decision. The second line is a comment and is ignored. The third line shows the names of the attributes and the decision. This kind of input file adheres to the LERS input format.

The valued tolerance program takes four inputs namely

1) The name of the input file,

2) Whether the probability table needs to be written to a file,

3) Whether the lower and upper approximations have to be written to a file,

4) The credibility threshold that the user chooses.

This program has two main classes- the decision table class and the rule class. The decision table class has member variables and functions which are used mainly for the storage of the input decision table. Also, this class has functions that perform the preliminary steps in the valued tolerance approach namely, the calculation of probability table, the calculation of lower and upper approximations etc. The rule class has members associated with the rule induction process and the process of dropping conditions. The member variables are used for storage and optimization of rules and the functions are used for inducing the rules and removing redundant rules and dropping conditions. A detailed explanation of the components of the classes is given in Appendix B.

CHAPTER 4: EXPERIMENTS

This chapter explains the experiments done on the rules induced by the valued tolerance relation.

4.1 Tools used for evaluation of the decision rules

The main parameter that is used for evaluating the decision rules is the error rate. The validation of the decision rules is done by a process called ten-fold cross validation. In this process, two sets of input data are used namely training data and testing data. Training data is used for initial analysis, from which rules are induced by either MLEM2 or the valued tolerance approach. Testing data is used for confirming and validating the rules induced by the algorithm. In a k-fold cross validation algorithm, the initial input data is divided into k sets of training and testing data. For this thesis, the value of k is taken to be 10. Hence this process is called ten-fold cross validation. This method is done using the following set of tools:

(a) Sample

Sample is the first program used that processes the input file to give the files that can be used for ten-fold cross validation. This program shuffles the input file and provides ten large files for training and 10 smaller files for testing purposes. Sample can work for any n-fold cross validation.

The way that sample is used for this thesis is

sample –f T-E-M-P.N-foldD -m pctg -c

where -f is to specify the input file,

T-E-M-P.N-foldD is the input file,

-m option is to specify that multiple files will be generated with some percentage split,

pctg is the value of the percentage split for generating the testing and training files; for this thesis, the percentage split is taken to be 90-10,

–c option means that the program uses a constant seed in the random number generation.

(b) Rule checker

The rule checker program takes the rules induced by the algorithm (MLEM2 or valued tolerance) and computes the error rate. Two rule checkers are used namely, 'srch' and 'chkrul'. The rule checker 'srch' is used to convert the rules according to the LERS format since it does not handle missing attribute values. The 'srch' rule checker takes the following inputs:

(a) Input rule file –which is the rule file generated by the algorithm,

(b) Input data file – the file which is given as the input to the algorithm,

(c) Name of the report file. This report file gives a list of attributes describing the rules. One such example report file is:

This report was created from: valuedresult and from: bank-5.d

The total number of examples is: 66
The total number of attributes is: 5
The total number of rules is: 25
The total number of conditions is: 100
The total number of examples that are not classified: 41
The total number of examples that are incorrectly classified: 0
The total number of examples that are not classified or are incorrectly classified: 41

Error rate: 62.12 percent

Concept(d, 1):
The total number of examples that are not classified: 8
The total number of examples that are incorrectly classified: 0
The total number of examples that are correctly classified: 25
The total number of examples in the concept: 33

Concept(d, 2):
The total number of examples that are not classified: 33
The total number of examples that are incorrectly classified: 0
The total number of examples that are correctly classified: 0
The total number of examples in the concept: 33

(d) Name of the new rule set file. This gives the rule file according to the LERS format. One such example rule file is:

```
! This rule file was created from: input.txt.r and from: input.txt
! ------------------------------------------------------

2, 1, 1
(a3, 1) & (a4, 0) -> (d, low)
3, 1, 1
(a1, 2) & (a3, 2) & (a4, 1) -> (d, high)
3, 1, 1
(a2, 2) & (a3, 1) & (a4, 3) -> (d, high)
```

Where the 3 numbers before the rule specify the Strength, Specificity and Support respectively described in Section 2.2.6. In addition to the above input files, the rule checker needs to be specified if the error rate has to be computed for the certain or possible rules.

The second rule checker used is 'chkrul' which takes the rule file generated by srch and gives the error rate for the rules induced. Input rule file for this program is the rule file generated by srch according to LERS format. The 'chkrul' rule checker takes the following inputs:

(a) Input data file which is the file from which rules are induced,

(b) If the rules are to be evaluated using conditional probability or strength,

(c) If support is to be used or not,

(d) If specificity is to be used or not,

(e) If the concept statistics need to be printed,

(g) If the case classifications need to be printed.

A typical example report generated by chkrul with the concept statistics is given below:

General Statistics

This report was created from: ru and from: input.txt
The total number of cases: 12
The total number of attributes: 4
The total number of rules: 3
The total number of conditions: 8
The total number of cases that are not classified: 0
PARTIAL MATCHING
The total number of cases that are incorrectly classified: 1
The total number of cases that are correctly classified: 2
COMPLETE MATCHING
The total number of cases that are incorrectly classified: 4
The total number of cases that are correctly classified: 5
PARTIAL AND COMPLETE MATCHING
The total number of cases that are not classified or incorrectly classified: 5
Error rate: 41.67%

Concept ("d","low")
The total number of cases that are not classified: 0

PARTIAL MATCHING
The total number of cases that are incorrectly classified: 1
The total number of cases that are correctly classified: 0

COMPLETE MATCHING
The total number of cases that are incorrectly classified: 4
The total number of cases that are correctly classified: 1
The total number of cases in the concept: 6

Concept ("d","high")
The total number of cases that are not classified: 0

PARTIAL MATCHING
The total number of cases that are incorrectly classified: 0
The total number of cases that are correctly classified: 2

COMPLETE MATCHING
The total number of cases that are incorrectly classified: 0
The total number of cases that are correctly classified: 4
The total number of cases in the concept: 6

(c) Error rate calculator

This program 'nf-sum' computes the final error rate of all the ten runs of cross validation. It computes the ratio between the number of incorrectly classified or not classified cases and the total number of cases and gives the error rate in terms of percentage.

(d) MLEM2

This program generates rules from the input file based on the MLEM2 algorithm. Rule induction can be done based on singleton, subset and concept approximations (described in Section 2.2.3). This program generates the rule files separately for certain and possible rules. A sample execution is given below:

[svasudev@cycle1 ~]$ ./mlem2
MLEM2 Modified version 2.0
Credits: Qiang Zhang - Original version
Sachin S. Siddhaye – Modified v1.0
Steven Santoso - Modified v2.0

Please enter input file name: input.txt
NOTE: Output file names will be automatically generated
Certain rules for all concepts will be inside <input_filename>.r.c
Possible rules will be inside <input_filename>.r.p
NOTE:
The tables will be generated in □eparate files:
<input_filename>.<concept_name>.l – all lower approximation of every concept
<input_filename>.<concept_name>.u – all upper approximation of every concept

What approximation definition would you use: Singleton (1), Subset (2), Concept (3)?
1
.....
Time for execution is 0 seconds.

(e) Valued tolerance

This program implements the valued tolerance approach and induces rules based on the valued tolerance relation.

This program takes the following inputs:

(i) The input file,

(ii) The credibility threshold supplied by the user,

(iii) If the probability table needs to be stored in a file,

(iv) If the approximations need to be stored in a file.

A sample execution is given below:

```
[svasudev@cycle1 wine20try]$ ./valued_tolerance –f valued.txt
valued tolerance Program:
Credits: Jerzy Grzymala Busse
         Supriya Vasudevan
Welcome to the valued tolerance program
Output Rule file will be generated in the name of INPUTFILE.r
NOTE: Output file names will be automatically generated
The execution time is 9 seconds
 Thank you for using this program!!
```

Thus, with an easy-to-use and efficient set of tools, the decision rules induced by the MLEM2 and the valued tolerance approach are compared and analyzed based on the error rate.

4.2 Comparison of valued tolerance approach and MLEM2

In this thesis, the emphasis is mainly put on comparing the valued tolerance approach with the MLEM2 approach based on the error rate which can be calculated as:

$$\text{Error Rate} = \frac{\text{No of incorrectly classified values or unclassified values}}{\text{Total number of Values in the Decision Table}}$$

The input data sets taken into consideration are the following:

(a) **m-hepatitis.d**: This data set contains 155 examples, 19 attributes and 167 missing attribute values (5% missing values)

(b) **m-wine-aca-5.d**: This data set has 178 examples , 13 attributes and 116 missing values (5% missing values)

(c) **m-wine-aca-20.d**: This data set has 178 examples, 13 attributes and 461 missing values (20% missing values).

(d) **m-lymph-symb.d**: This data set contains 148 examples and 18 attributes and 133 missing values (5% missing values).

(e) **m-image-aca-5.d**: This data set contains 210 examples, 19 attributes and 200 missing values (5% missing values).

Rules were induced from these data sets using MLEM2 and valued tolerance approaches. A ten-fold cross validation was then performed. The valued tolerance approach was used with different credibility thresholds 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0. The results obtained using the valued tolerance approach with different credibility threshold values and the MLEM2 approach for the above mentioned data sets are shown in Table 4.1 and Table 4.2.

Table 4.1: Error rate for certain rules in MLEM2 and valued tolerance

Certain rules							
Input files	Valued tolerance approach						MLEM2
	Credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	21.29	21.29	21.94	22.58	22.58	22.58	21.62
m-lymph-symb.d	27.19	26.62	25.95	25.95	25.95	31.11	18.06
m-wine-aca-5.d	11.18	10.67	9.55	8.43	11.24	12.36	5.45
m-image-aca-5.d	11.24	11.24	9.55	8.43	11.24	12.36	26.97
m-wine-aca-20.d	27.04	26.84	26.84	25.92	19.66	24.07	38.2

Table 4.2: Error rate for possible rules in MLEM2 and valued tolerance

Possible rules							
Input files	Valued tolerance approach						MLEM2
	Credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	20.65	20.65	20.65	20.65	20.33	20.33	21.62
m-lymph-symb.d	27.15	31.76	32.43	32.43	33.11	35.21	17.42
m-wine-aca-5.d	20.65	20.65	20.65	20.65	20.33	20.33	21.62
m-image-aca-5.d	30.31	31.69	31.69	32.35	32.25	26.63	9.55
m-wine-aca-20.d	21.69	21.64	21.64	22.25	22.25	26.63	20

4.3 Observations

The following comparisons were made for both certain and possible rules:

(a) Comparison of the error rate for MLEM2 and for different values of credibility degree for valued tolerance approach using 10-fold cross validation,

(b) Comparison of the number of rules induced by both the approaches,

(c) Comparison of the number of conditions in the rules induced by both the approaches.

From the results for study (a), as shown in Tables 4.1 and 4.2, it can be inferred that for certain rules as well as possible rules, MLEM2 usually works better than the valued tolerance algorithm. Different credibility degree thresholds are being used to filter the rules. It is found that for a majority of the data sets, MLEM2 usually works better than any credibility degree threshold in the valued tolerance approach. The results for study (b) are shown in the Tables 4.3 and 4.4. Based on the results obtained, it can be inferred that MLEM2 usually induces smaller number of rules with smaller number of conditions than the valued tolerance approach. Since the error rate for MLEM2 is also lesser we can conclude that MLEM2 algorithm usually induces minimal and more accurate rule sets. Tables 4.5 and 4.6 show the results of study (C) based on the number of conditions for MLEM2 and valued tolerance.

Table 4.3: Comparison of the number of certain rules

Number of certain rules							
Input files	Valued tolerance approach						MLEM2
	credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	91	91	91	91	91	90	24
m-lymph-symb.d	45	45	45	45	45	50	31
m-wine-aca-20.d	119	120	111	119	109	69	49
m-wine-aca-5.d	46	47	47	47	40	38	47
m-image-aca-5.d	99	96	97	94	98	76	21

Table 4.4: Comparison of the number of possible rules

Number of possible rules							
Input files	Valued tolerance approach						MLEM2
	Credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	125	125	125	125	125	125	21
m-lymph-symb.d	57	55	53	53	50	30	31
m-wine-aca-20.d	22	22	22	22	22	21	29
m-wine-aca-5.d	29	29	29	29	29	20	21
m-image-aca-5.d	95	95	95	95	95	91	21

Table 4.5: Comparison of the number of conditions in certain rules

Number of conditions in certain rules							
Input files	Valued tolerance approach						MLEM2
	Credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	522	522	522	522	522	511	76
m-lymph-symb.d	436	436	436	435	435	455	105
m-wine-aca-20.d	337	348	348	349	323	311	265
m-image-aca-5.d	430	475	413	444	434	422	228
m-wine-aca-5.d	337	348	348	349	323	311	265

Table 4.6: Comparison of the number of conditions in possible rules

Number of conditions in possible rules							
Input files	valued tolerance approach						MLEM2
	Credibility degree threshold values						
	0.5	0.6	0.7	0.8	0.9	1.0	
m-hepatitis.d	250	250	250	250	238	238	89
m-lymph-symb.d	186	185	182	186	178	103	105
m-wine-aca-5.d	195	195	195	195	195	160	86
m-image-aca-5.d	175	171	175	171	171	139	189
m-wine-aca-20.d	122	122	122	122	122	113	98

CHAPTER 5: CONCLUSIONS

In this research the valued tolerance approach to induce decision rules from incompletely specified decision tables was implemented. This implementation was evaluated by conducting a series of experiments with the induced rules using a set of available tools. A ten-fold cross validation was performed with the MLEM2 algorithm and the valued tolerance approach. The error rate of the rules, the number of conditions and the number of rules were calculated for both the algorithms and a comparative analysis was done. Based on the experiments done on five different data sets and the available results, it can be concluded that the MLEM2 algorithm, based on rough set theory usually works better than the valued tolerance approach for handling missing attribute values. This is because the rules induced by MLEM2 algorithm usually have smaller error rate and smaller number of rules and conditions. This makes the rule set minimal as opposed to the valued tolerance approach. Thus, for missing attribute values, the MLEM2 algorithm usually produces a more accurate classification. Also, the MLEM2 algorithm works with the do-not-care conditions (*), lost values (?) as well as attribute-concept values (-). On the other hand, the valued tolerance approach works with only the do-not-care conditions. This makes the MLEM2 algorithm typically a candidate for a wider application domain than the valued tolerance approach.

5.1 Recommendations and future extensions

The current design and implementation of the valued tolerance approach deals with only the do-not-care type of missing attribute values. Future work can include the lost values and the attribute-concept values. Further, this research is based only on the probability-based valued tolerance relation. Other kinds of valued tolerance relations can be explored and rule induction algorithms can be proposed. Analysis of the induced rules can be done based on various other parameters besides the error rate and the number of rules. Also, a comparative study can be made based on different kinds of n-fold cross validations besides the ten-fold cross validation done in this thesis. Thus, the valued tolerance relation proves to be a promising area of research in dealing with missing attribute values in decision tables.

REFERENCES

[1] Jerzy Stefanowski, Alexis Tsoukias, *Incomplete Information tables and Rough Classification*, Computational Intelligence Journal, ed. By Randy Goebel and Russell Greiner, Volume 17, Number 3, Aug 2001, 545-566.

[2] Jerzy W. Grzymala-Busse, *Rough set Theory with Applications to Data Mining,* Real World Applications of Computational Intelligence, ed. by M. G. Negoita and B. Reusch, Springer Verlag, 2005, 221-244.

[3] Jerzy W. Grzymala-Busse, *On the unknown attribute values in Learning from Examples*, Proceedings of the ISMIS-91, 6th International Symposium on Methodologies for Intelligent Systems, Charlotte, North Carolina, Oct 16-19, 1991, 368-377.

[4] Jerzy W. Grzymala-Busse, *Three Approaches to Missing Attribute Values- A Rough Set Perspective,* Proceedings of the Workshop on Foundation of Data Mining, associated with the Fourth IEEE International Conference on Data Mining, Brighton, UK, November 1-4, 2004, 55–62.

[5] Jerzy W. Grzymala-Busse, Witold J. Grzymala-Busse, *Handling Missing Attribute Values,* The Data Mining and Knowledge Discovery Handbook 2005, ed. By Oded Miamon and Lior Rokach, Springer-Verlag, 2005, 37-57.

[6] Jerzy W. Grzymala-Busse, Witold J.Grzymala-Busse, *Rule Induction,* The Data Mining and Knowledge Discovery Handbook, ed. By Oded Maimon and Lior Rokach, Springer-Verlag, 2005, 277-294.

[7] Jerzy W. Grzymala-Busse, *EECS 837 Lecture Notes*, taken during Fall 2006.

[8] Temidayo.B.Ajayi, *Incomplete Data Mining, A Rough Set Approach,* Thesis document submitted to the Department of Electrical Engineering and Computer Science, University of Kansas, under the guidance of Jerzy W. Grzymala-Busse.

[9] Jerzy W. Grzymala-Busse, *A Comparison of Three Strategies to Rule Induction from Data with Numerical Attributes,* Proceedings of the International Workshop on Rough Sets in Knowledge Discovery (RSKD2003), associated with the European Joint Conferences on Theory and Practice of Software 2003, Warsaw, Poland, April 2003, 132-140.

[10] Jerzy W. Grzymala-Busse, Sachin Siddhaye, *Rough Set Approaches to Rule Induction from Incomplete Data,* Proceedings of the IPMU'2004, the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy, July , 2004, Volume 2, 923–930.

[11] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski, and Wojciech Ziarko, *Rough Sets,* Communications of the ACM November 1995, Volume 38, Issue 11 , 88 - 95.

[12] Zdzislaw Pawlak, *Rough Sets – Theoretical aspects of reasoning about data,* Boston, MA, Kluwer Academic Publishers, 1991, Boston Dordrecht, London.

[13] Jerzy Stefanowski and Alexis Tsoukias *On the Extension of Rough Sets under Incomplete Information,* ed by N. Zhong, A. Skowron, S. Ohsuga, RSFDGCrC, Springer-Verlag Berlin Heidelberg, 1999, 73-82.

[14] Jerzy Grzymala-Busse, *Rough Set Approach to Incomplete Data,* Artificial Intelligence and Soft Computing - ICAISC 2004, vol. 3070, 50-55.

[15] Zdzislaw Pawlak, *Inference Rules and Decision Rules,* Artificial Intelligence and Soft Computing - ICAISC 2004 7th International Conference, Zakopane, Poland, June 2004, 102-108.

[16] Zdzislaw Pawlak, *Rough sets and data analysis,* Fuzzy Systems Symposium, Soft Computing in Intelligent Systems and Information Processing, Dec 11-14, 1996, 1-6.

LIST OF TABLES

APPENDIX A

LEM2 algorithm

The procedure for LEM2 algorithm is given below: [6]

Procedure LEM2
(**input:** a set B,
output: a single local covering $\mathbb{T}$ of set B);
begin
G := B;
$\mathbb{T}$:= Ø;
while G ≠ Ø
begin
T := Ø;
T(G) := {t | [t] ∩ G ≠ Ø};
while T = Ø **or** [T] ⊊ B
begin
select a pair t ∈ T(G) with the highest attribute priority,
if a tie occurs, select a pair t ∈ T(G) such that |[t] ∩ G| is maximum;
if another tie occurs, select a pair t∈ T(G) with the smallest cardinality of [t]; if a further tie occurs, select first pair;
T := T ∪ {t};
G := [t] ∩ G;
T(G) := {t | [t] ∩ G ≠ Ø};
T(G) := T(G) – T;
end {while}
for each t in T **do**
if [T – {t}] ⊆ B **then** T: = T – {t};
$\mathbb{T}$:= $\mathbb{T}$ ∪ {T};
G := B – $\cup_{T \in \mathbb{T}}[T]$
end {while};
for each T in $\mathbb{T}$ **do**
if $\cup_{S \in \mathbb{T}-T}[S] = B$ then $\mathbb{T}$:= $\mathbb{T}$ – {T}
end {procedure}.

APPENDIX B

Implementation details:

The following are the components of the decision table class. After processing the input file, in addition to storing it in a vector, the number of attributes and the number of cases are also calculated. After this, the module for calculating the probabilities is executed. This module takes as inputs the input data vector and the number of attributes and has the following structure:

void calc_prob_table(vector<string> inputdata,int attr_count);

This function calculates the probability table and stores that in a two-dimensional array. This data structure is chosen since only half the probability table is calculated and accessing the values from a 2D array is more efficient. Since the probability table is a symmetric structure, it is sufficient to calculate only half of the table for further processing. This is one of the code optimization techniques used in the program. This probability function is being used multiple times in the program while dropping the conditions in the decision rules where the entire process is reiterated. If the user had given the option to store the probabilities in a file, the module to write the table in a file is called.

After calculating the probabilities and storing them in proper data structures, the lower and upper approximations are calculated using the function,

void calc_valuedtolerance_table();

If the user had chosen the option to store the lower and upper approximations, then the module to store them in a file is called.

In addition to these main modules, there are also supplementary functions in the decision table class. These are helper functions which perform the tasks of accessing the data row wise, calculating the unique concept values etc.

After this, the job is taken over by the members of the rule class. The module that is executed next is the ***void calculate_I_S_values(decision_table dt).*** This function does a series of function calls to get started with the process of rule induction. First it calculates the rows with no missing attribute values using a helper function. Taking these rows as rule candidates or templates, the rows which support the rule are calculated. Then the values of I and S are calculated by calling the functions ***get_I_values () and get_S_values ()*** which calculate the I values and the S values for the decision rule induction process. The S value indicates the support degree that the rule has to the particular element. The I value corresponds to the lower and upper approximations.

After calculating the I and S values, the value of the credibility degree is calculated using the function ***calculate_credibility_degree ()***. This credibility degree is then verified with the threshold and the appropriate ***dropping_conditions ()*** function is called for the simplification process. Also, the rules are stored in appropriate vectors. Thus this module acts as the key module for the entire rule induction process.

Also, in this module, the coverage of rules is calculated, ie, the set of cases covered by each rule. The module ***dropping_conditions ()*** then takes care of the simplification process, by removing redundancies. Here, two kinds of redundancies are removed. By dropping conditions, the redundant conditions are removed. Also one more level of redundancy is removed, that is, the rule level redundancy. The rules which cover the same set of cases or a subset of cases covered by another rule are eliminated, which is also essential to get a set of minimal and complete rules.

Thus, the dropping conditions module takes the biggest credibility degree value and then calls the entire process from the calculation of probabilities to rule induction again in a loop until a set of minimal rules are obtained. After the entire rule set is obtained and the condition-level redundancy is eliminated, the redundant rules are removed using the two-dimensional array storing the coverage of all the rules mentioned above thus completing the coverage-level redundancy also.

Thus, this program performs the entire process of rule induction using the valued tolerance approach in an efficient way, without any recursion, by a series of function calls and loop data structures. Thus the valued tolerance approach for decision rule induction is implemented in the C/C++ programming language.