2010

Two new approaches to evaluate association rules

Delpisheh, Elnaz

Lethbridge, Alta. : University of Lethbridge, Dept. of Mathematics and Computer Science, c2010

TWO NEW APPROACHES TO EVALUATE ASSOCIATION RULES

ELNAZ DELPISHEH
Bachelor of Science, Alzahra University, Tehran, Iran, 2005

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

To my beloved parents and aunt.

Abstract

Data mining aims to discover interesting and unknown patterns in large-volume data. Association rule mining is one of the major data mining tasks, which attempts to find inherent relationships among data items in an application domain, such as supermarket basket analysis. An essential post-process in an association rule mining task is the evaluation of association rules by measures for their interestingness. Different interestingness measures have been proposed and studied. Given an association rule mining task, measures are assessed against a set of user-specified properties. However, in practice, given the subjectivity and inconsistencies in property specifications, it is a non-trivial task to make appropriate measure selections.

In this work, we propose two novel approaches to assess interestingness measures. Our first approach utilizes the analytic hierarchy process to capture quantitatively domain-dependent requirements on properties, which are later used in assessing measures. This approach not only eliminates any inconsistencies in an end user's property specifications through consistency checking but also is invariant to the number of association rules. Our second approach dynamically evaluates association rules according to a composite and collective effect of multiple measures. It interactively snapshots the end user's domain-dependent requirements in evaluating association rules. In essence, our approach uses neural networks along with back-propagation learning to capture the relative importance of measures in evaluating association rules.

Case studies and simulations have been conducted to show the effectiveness of our two approaches.

Acknowledgments

Writing this thesis has been a process which involved many individuals whom I appreciate. First, I would like to thank my supervisor, Dr. John Z. Zhang, for his endless guidance in every single step of this work. Second, I would like to thank Dr. Hadi Kharaghani, my co-supervisor, for supporting this project and providing intellectual inputs. I would also like to thank my committee members, Dr. Robert Benkoczi, Dr. Paul Hazendonk, and Dr. Howard Cheng, for taking the time to read my thesis and for their valuable feedbacks on this work.

I am extensively indebted everything in my life to my family and friends. I would like to thank my parents who taught me faith, kindness, and hard work; my sisters and brother Narjes, Marjan, and Amirali, who taught me love. I would also like to thank my aunt for her endless care, love, and support; and my grandmother who taught me endurance by sharing her life experiences. I would also like to thank my best friend Maede whom it was with her support that my stay in Lethbridge became more pleasant.

Contents

Approval/Signature Page ii

Dedication iii

Abstract iv

Acknowledgments v

Table of Contents vi

List of Tables viii

List of Figures ix

1 Introduction 1
- 1.1 What is data mining? . . . 1
- 1.2 Data mining tasks . . . 2
- 1.3 Association rule mining . . . 4
  - 1.3.1 Finding frequent itemsets . . . 5
  - 1.3.2 Generating association rules . . . 6
- 1.4 Evaluating association rules . . . 8
- 1.5 Interestingness criteria of association rules . . . 9
- 1.6 Main contributions . . . 11
- 1.7 Thesis outline . . . 12

2 Background 13
- 2.1 Interestingness measures . . . 13
  - 2.1.1 Subjective measures . . . 14
  - 2.1.2 Objective measures . . . 16
- 2.2 Properties for assessing measures . . . 18
- 2.3 Selecting interestingness measures . . . 20

3 An AHP-based Measure Assessment Method 23
- 3.1 A revised analytic hierarchy process . . . 23
  - 3.1.1 AHP details . . . 24
  - 3.1.2 Measure of consistency . . . 27
- 3.2 AHP-based measure assessment method . . . 28
  - 3.2.1 Presenting properties . . . 28
  - 3.2.2 Rating properties . . . 28
  - 3.2.3 Assessing and prioritizing properties . . . 29
  - 3.2.4 Assessing property fulfillment . . . 29
- 3.3 Case studies . . . 29

3.3.1 Case one . . . 30
3.3.2 Case two . . . 33
3.4 Different measure assessment methods . . . 38
3.5 Summary . . . 39

4 A Dynamic Composite Approach for Evaluating Association Rules 41
4.1 Neural networks . . . 42
4.1.1 Definition and structure of neural networks . . . 42
4.1.2 Issues in neural networks . . . 46
4.2 A dynamic and composite association evaluation . . . 49
4.2.1 Approach details . . . 50
4.2.2 Training process . . . 56
4.2.3 Testing process . . . 57
4.2.4 Other implementation details . . . 57
4.3 Simulations . . . 58
4.3.1 Results on car data set . . . 58
4.3.2 Results on the solar data set . . . 66
4.4 Discussions . . . 70
4.4.1 The neural networks . . . 71
4.4.2 The effectiveness of our approach . . . 73
4.5 Summary . . . 74

5 Conclusion and Future Work 76
5.1 Our approaches . . . 77
5.2 Future work . . . 77

Bibliography 79

Appendix-A 83

Appendix-B 85

List of Tables

1.1 A transactional database 5

2.2 Objective measures for association rules 17
2.1 A 2×2 contingency table 18
2.3 Interestingness properties fulfilled by measures 20
2.4 Groups of objective measures with similar properties 21

3.1 The typical importance values in CPC 24
3.2 The CPC algorithm 25
3.3 Case one consistent property specifications 31
3.4 Case one composite weights 33
3.5 Case two inconsistent property specifications 34
3.6 Case two consistent property specifications 35
3.7 Case two composite weights 37
3.8 Measure assessment approaches and their characteristics 38

4.1 Interestingness measure scores for R_1, R_2, and R_3 59
4.2 Absolute residual values ($\eta = 0.2$) 63
4.3 Absolute residual values ($\eta = 0.3$) 63
4.4 Absolute residual values ($\eta = 0.4$) 63
4.5 Absolute residual values ($\eta = 0.5$) 63
4.6 Absolute residual values ($\eta = 0.6$) 63
4.7 Absolute residual values ($\eta = 0.7$) 63
4.8 Absolute residual values (number of hidden neurons = 17) 64
4.9 Absolute residual values (number of hidden layers = 0) 65
4.10 Absolute residual values (number of hidden layers = 1) 65
4.11 Absolute residual values ($\alpha = 0.2$) 66
4.12 Absolute residual values ($\alpha = 0.4$) 66
4.14 Absolute residual values (solar data set) 70
4.13 The results of the solar flare empirical studies 70
4.15 Absolute residual values (a new training set) 72
4.16 Absolute residual values (new training and measure sets) 72
4.17 Absolute residual values (random inconsistent training and testing set) . . . 73

List of Figures

1.1 The basket analysis 4
1.2 Frequent itemsets generation 7

4.1 A neuron 43
4.2 The sigmoid function 44
4.3 Multi-layer neural network 45
4.4 The overview of an association mining system 48
4.5 The high-level view of our approach 48
4.6 A hidden neuron receiving back-propagated errors 50
4.7 The first 50 rounds of the training process ($\eta = 0.2$) 62
4.8 The last 50 rounds of the training process ($\eta = 0.2$) 62
4.9 The entire training process ($\eta = 0.2$) 62
4.10 The entire training process ($\eta = 0.3$) 62
4.11 The entire training process ($\eta = 0.4$) 62
4.12 The entire training process ($\eta = 0.5$) 62
4.13 The entire training process ($\eta = 0.6$) 62
4.14 The entire training process ($\eta = 0.7$) 62
4.15 The entire training process (number of hidden neurons = 17) 64
4.16 The entire training process (number of hidden layers = 0) 65
4.17 The entire training process (number of hidden layers = 1) 65
4.18 The entire training process (solar data set) 70
4.19 The entire training process (a new training set) for car 72
4.20 The entire training process (new training and measure sets) for car 72

Chapter 1

Introduction

People have always been confronting with a growing amount of data, which in turn demands more on their ability to interpret and comprehend them. Previously, data used to be collected and stored in large repositories that were seldom revisited, because there were no novel technologies to extract the valuable information and knowledge hidden in them. Consequently, in many situations, decisions were only made based on decision makers' intuitions and instincts, rather than on educated rationales from data itself [14].

During the past two decades, the concepts of *data mining* have emerged to remedy the situation. Various data mining techniques and tools have been proposed, designed, and implemented to perform data analysis to uncover interesting patterns from data, with the aim of providing help in business strategies, knowledge bases, scientific research, etc. [14]. Data mining has evolved from a number of different disciplines, including *statistics*, *machine learning*, *artificial intelligence*, *database technologies*, *information retrieval*, *high-performance computing*, etc. [39]. In the following, we provide a brief introduction to data mining.

1.1 What is data mining?

There exist various definitions of data mining from different perspectives. Some define data mining as a tool that connects users to huge amount of data, while others consider its critical role in mining data. We show some of these definitions in the following.

- Data mining is a process of automatically analyzing data in large data repositories to extract interesting patterns, such as relationships, from data [14].
- Data mining is a process of selection, exploration, and modeling of large-volume

data to discover relationships that are previously unknown, aiming at obtaining clear and useful results for the owner of the data [12].

- Data mining is a process of data-driven extraction of implicit, previously unknown, and potentially useful (or actionable) information from large databases [39].

- Data mining is a process of discovering various models, summaries, and derived values from a given collection of data [21].

In our work, we refer to data mining as *an automatic process to analyze and extract (mine) interesting information from large-volume data, thus facilitating making critical decisions*.

1.2 Data mining tasks

In the literature, there are several major data mining tasks. We briefly discuss them in the following.

- *Classification mining*. Classification is one of the most common data mining tasks. It is the act of assigning a class label to an unknown data item. The data item is represented by a set of attributes, one of which is for the class label. The classification task finds a model that can predict, for an unknown data item, its class label based on its other attributes. The classification process includes two main steps. In the first step, a *classifier* is built to describe predetermined sets of data classes. This step is called *learning step* or *training phase*. Essentially, it finds a function, $y = f(x)$, that can predict the class label y of an unknown data item x based on the information of other attributes of x. The second step is to test the classifier. It estimates the prediction accuracy of the classifier by applying it on a set of testing data items and their associated class labels. If the accuracy is acceptable, it will be then put into use to classify future data items with unknown class labels [14, 29].

Some classification examples are as follows:

  - Classifying loan applicants as *risky* or *safe* for banks. The attributes of an applicant include, *salary level*, *education level*, *past credit history*, etc.
  - Classifying tomorrow's weather, represented by *precipitation*, *humidity level*, *temperature*, etc., as *sunny*, *cloudy*, or *rainy*.
  - Classifying vertebrates, with attributes such as *fur*, *tail*, *wings*, etc., into *mammals*, *birds*, *reptiles*, *amphibians*, or *fish*.

  The algorithms that are widely used for classifications include *neural network learning algorithms*, *support vector machines*, *K-nearest neighbors*, *Gaussian mixture model*, *decision trees*, etc. [14, 39]

- *Clustering mining*. Clustering, also called *segmentation*, is similar to classification. But classes of data items are not predefined. Given a set of data items, a clustering process groups them into clusters that are observed similar. The data items that are close to one another are put in one cluster. For example, a bank can cluster its customers into several clusters based on the similarities of their *age*, *income*, *residence*, etc. The characteristics shared by customers in a cluster can be used to describe that cluster of customers. These clusters help the bank understand its customers better and provide more suitable customized products and services. *Partitioning-based*, *hierarchical-based*, *partial-based*, *density-based*, *model-based*, *grid-based* are some common clustering techniques, which are discussed in [14, 29, 39].

- *Association rule mining*. Association rule mining discovers the inherent relationships among data items in an application domain. One such application domain could be the basket analysis in supermarkets, where one tries to discover the relationships among commodities in baskets. In association mining, interesting patterns are represented as *association rules*. For example, in the basket analysis,

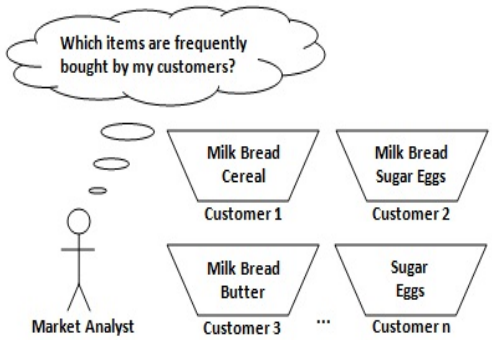

Figure 1.1: The basket analysis (adapted from [14]).

$\{milk, eggs\} \rightarrow \{bread\}$ is an association rule, implying that if *milk* and *eggs* are bought together by a customer, then *bread* is likely to be bought as well [3, 12].

In addition to the above three major data mining tasks, there are also many others, including *regressional analysis*, *web mining*, *data stream mining*, *text mining*, etc. For a comprehensive review on them, see [14].

In our work, we focus on the association rule mining. The following section discusses association rules and their mining algorithms.

1.3 Association rule mining

Let $I = \{i_1, i_2, i_3, \cdots, i_n\}$ be a set of n *items* and $D = \{t_1, t_2, t_3, \cdots, t_m\}$ be a *transactional database*, where each transaction $t_i \in D$ is a nonempty subset of I. An association rule is of the form $A \rightarrow B$, where A and B are called the *itemsets* and $A \subset I$ and $B \subset I$. It is required that $A \cap B = \phi$. A is called the *antecedence* of the rule while B is called its *consequence*. We say that the rule $A \rightarrow B$ holds for the database D with *support* s, where s is the percentage of transactions in D that contain both A and B, and with *confidence* c, where c is the percentage of transactions containing A that also contain B.

TID	List of item IDs
$T100$	I1, I2, I5
$T200$	I2, I4
$T300$	I2, I3
$T400$	I1, I2, I4
$T500$	I1, I3
$T600$	I2, I3
$T700$	I1, I3
$T800$	I1, I2, I3, I5
$T900$	I1, I2, I3

Table 1.1: A transactional database [14].

1.3.1 Finding frequent itemsets

A typical example of association rule mining, as previously mentioned, is the basket analysis. It analyzes customers' shopping habits by finding associations among the different items that customers place in their shopping baskets, as shown in Figure 1.1. The discovery of such associations can help retailers develop marketing strategies, i.e., selective marketing, arranging shelf spaces, etc., by gaining insight into which items are frequently purchased together by customers [14].

The first step for generating association rules for a transactional database is to find the so-called *frequent itemsets*, whose supports are more than a user-specified *minimum support*.

In the literature, many algorithms for finding different frequent itemsets have been proposed and studied [18]. The one we have chosen for our work is the basic *Apriori* algorithm. Other versions of the algorithm, for instance, the ones in [26, 45], have also been developed to further improve the efficiency of the algorithm.

The Apriori algorithm is first proposed by Agrawal and Srikant [2]. It uses the *prior knowledge* of itemsets to find future itemsets. To improve the efficiency of the algorithm, the so-called *Apriori property* is used, which states that all nonempty subsets of a frequent itemset must also be frequent. Thus, if an itemset is found not frequent, it will not be used

to generate larger itemsets, i.e., any larger itemsets generated from it will definitely fail to satisfy the minimum support.

The algorithm is explained informally by applying it to the transactional database example shown in Table 1.1. The minimum support for finding frequent itemsets is set to 22%. This is equivalent to requiring that a frequent itemset must appear in at least $\lceil 22\% \times 9 \rceil = 2$ transactions, where 9 is the total number of transactions in Table 1.1.

First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted as L_1. Next, L_1 is joined[1] by itself to generate the set of candidate 2-itemsets, denoted as C_2. Given the Apriori property, we only select those frequent 2-itemsets, denoted together as L_2, from C_2 for further consideration. Then, using the join operation again, we generate C_3, from which we select the frequent 3-itemsets. We repeat this process, each time generating larger frequent itemsets, until no more frequent itemsets can be found. Figure 1.2 illustrates the whole process for finding the frequent itemsets with the specified minimum support from the transactional database in Table 1.1.

1.3.2 Generating association rules

Association rules are generated from frequent itemsets that satisfy *the minimum confidence* requirements. For each frequent k-itemset l, where $k \geq 2$, we check every nonempty proper subset s of l as whether the rule $s \rightarrow (l-s)$ satisfies $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold and $support\_count(x)$ is the occurrence

[1]The *join* operation, denoted as $\bowtie$, combines items from two or more joinable itemsets in a candidate set. Let l_1 and l_2 be two itemsets in L_{k-1}, where $k \geq 2$. Let $l_i[j]$ refer to the jth item in itemset l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1). Assuming that the items in l_1 and l_2 are in a lexical order, the two itemsets l_1 and l_2 are joinable if their first $k-2$ items are in common. That is, l_1 and l_2 are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The resultant itemset by joining l_1 and l_2 is $\{l_1[1], l_1[2], \cdots, l_1[k-2], l_1[k-1], l_2[k-1]\}$. This way, we generate the candidate set C_k by $L_{k-1} \bowtie L_{k-1}$. For example, consider $L = \{\{I1,I2\},\{I1,I3\},\{I1,I5\},\{I2,I3\},\{I2,I4\},\{I2,I5\}\}$. $L \bowtie L = \{\{I1,I2\},\{I1,I3\},\{I1,I5\},\{I2,I3\},\{I2,I4\},\{I2,I5\}\} \bowtie \{\{I1,I2\},\{I1,I3\},\{I1,I5\},\{I2,I3\},\{I2,I4\},\{I2,I5\}\} = \{\{I1,I2,I3\},\{I1,I2,I5\},\{I1,I3,I5\},\{I2,I3,I4\},\{I2,I3,I5\},\{I2,I4,I5\}\}\}$.

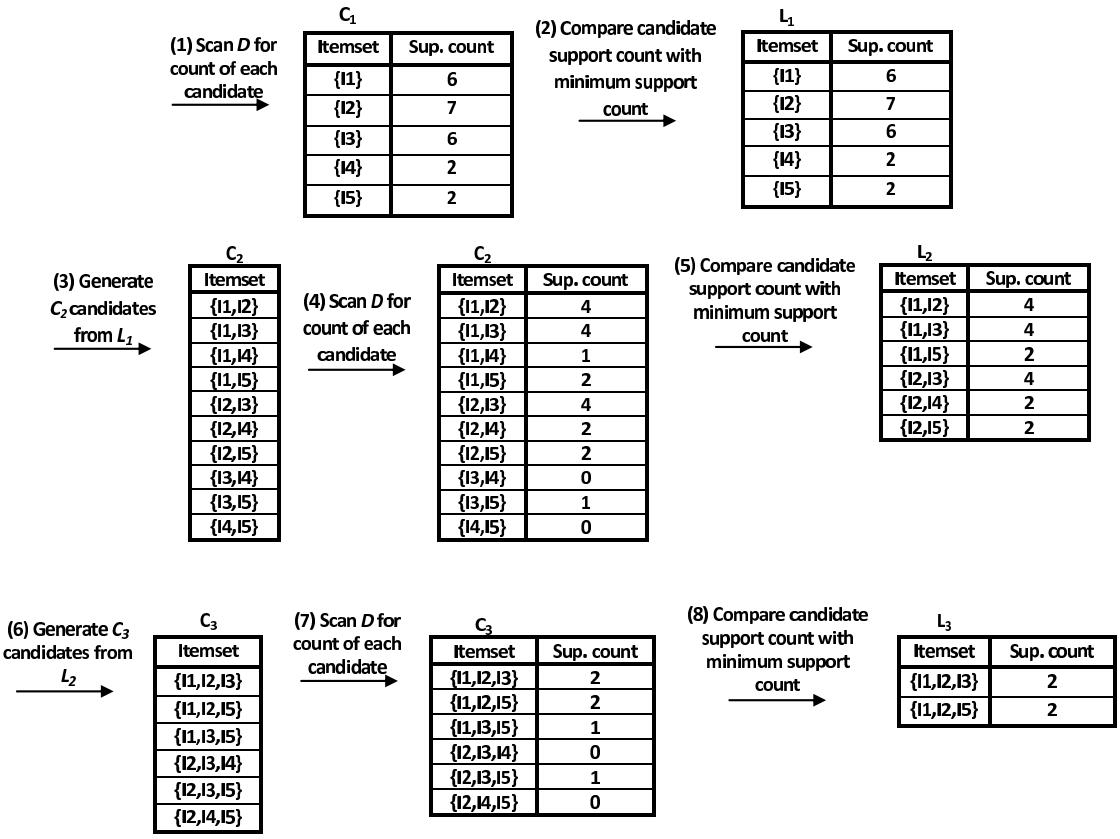

Figure 1.2: Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2. (1) and (2) The algorithm simply scans all of the transactions in order to count the number of occurrences of each item. L_1 corresponds to the frequent 1-itemsets satisfying the minimum support; (3) C_2 is generated using $L_1 \bowtie L_1$; (4) The transactions are scanned and the support count of each candidate itemset in C_2 is accumulated; (5) The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate set of 2-itemsets in C_2 satisfying the minimum support count; (6) C_3 is generated using $L_2 \bowtie L_2$; (7) By scanning the transactions again, collect the support count for each 3-itemset in C_3; (8) L_3 is obtained. Afterward, L_3 is used to generate the candidate set of 4-itemsets, $C_4 = \{\{I1, I2, I3, I5\}\}$. It is easy to check that $\{I1, I2, I3, I5\}$ is not frequent. Thus, $L_4 = \phi$. The Apriori algorithm terminates, after having found all of the frequent itemsets in $L_1 \cup L_2 \cup L_3$ [14].

frequency of itemset x in a transactional database.

For example, for the frequent 3-itemset $l = \{I1, I2, I5\}$, obtained from Table 1.1, the nonempty subsets of l are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$. The resultant association rules are shown as below, each with its confidence:

$R_1 : \{I1, I2\} \rightarrow \{I5\}$ $\quad 2/4 = 50\%$

$R_2 : \{I1, I5\} \rightarrow \{I2\}$ $\quad 2/2 = 100\%$

$R_3 : \{I2, I5\} \rightarrow \{I1\}$ $\quad 2/2 = 100\%$

$R_4 : \{I1\} \rightarrow \{I2, I5\}$ $\quad 2/6 = 33\%$

$R_5 : \{I2\} \rightarrow \{I1, I5\}$ $\quad 2/7 = 29\%$

$R_6 : \{I5\} \rightarrow \{I1, I2\}$ $\quad 2/2 = 100\%$

If the minimum confidence is taken to be 40%, then only the association rules R_1, R_2, R_3, and R_6 are returned.

1.4 Evaluating association rules

In the literature, two types of users are considered to be involved in association mining process. An *end user* is an expert of the data to be mined in a particular application domain, while an *analyst* is a specialist in data mining. The end user should provide the analyst with sufficient domain knowledge, represented as requirements, in the domain, in order to extract interesting association rules [25]. Throughout this thesis, the term *user* is meant to be end user.

One of the non-trivial issues in association rule mining is to deal with the large number of association rules discovered [17, 24, 27, 30, 33, 38, 43]. While many of them convey non-trivial and interesting information, some are trivial or even irrelevant [11, 27, 33].

One solution to identify trivial or irrelevant information is to evaluate association rules in an association mining process, as how beneficial or interesting they are. Usually this

takes place in the post stage of the mining process. Doing so not only tackles the situation where a mining process usually would return a large number of association rules but also distinguishes the truly interesting association rules from those that are not.

1.5 Interestingness criteria of association rules

In the literature, people have proposed to evaluate association rules based on their *interestingness*. Obviously, interestingness is a rather subjective concept, since an association rule that may sound interesting to one user sounds uninteresting to another.

Therefore, to make interestingness as objective as possible, it is measured against a set of *criteria* as follows. *Conciseness* concerns about whether an association rule contains a certain number of attribute-value pairs. *Generality* asks whether the relationship the association rule reveals occurs frequently. *Peculiarity* checks whether an association rule largely differs from other discovered association rules. Other criteria include *diversity*, *novelty*, *surprisingness*, *utility*, *comprehensibility*, etc. [10, 11, 30, 43]. It is easy to see that some of the above criteria, such as novelty and surprisingness, are dependent on each other, while some others, such as peculiarity and generality, are contradictory to each other [11].

Each criterion is made concrete by one or more so-called *measures* and a measure may serve more than one criteria. Depending on the extent of the user's involvement and the criteria to pursue, measures are divided into two groups - *objective* and *subjective*. An objective measure depends only on the structural nature, such as the statistical properties of association rules and the underlying data, whereas a subjective measure considers more of the views of the user on data, such as the user's subjective requirements. Measures, such as conciseness, generality, peculiarity, diversity, etc., are considered objective, while others, such as novelty, utility, surprisingness, comprehensibility, etc., are considered subjective.

For instance, given an association rule from a transactional database, *Recall r* is an objective measure assessing its conciseness, which is defined as the ratio between the num-

ber of correctly returned rules to the total number of correct rules that should have been returned. This measure states that $r\%$ of relevant association rules are retrieved. These concepts will be discussed in detail in Chapter 2.

However, as another example, some novel association rules may not be frequent enough. In addition, frequent interesting rules may be redundant or may correspond to prior knowledge that may not be interesting any longer. Thus, only considering (subjective) novelty of an association rule or its (objective) frequency is not sufficient to evaluate its interestingness. As a result, it is desirable for an appropriate interestingness measure to include a combination of objective and subjective measures [36].

Furthermore, given the overwhelming number of interestingness measures and their variety in pursuing interestingness criteria, they highly differ in evaluating association rules. For example, *support*, an objective measure, assesses the generality of association rules, while *specificity*, another objective measure, focuses on the peculiarity of association rules. Thus, an association rule may be considered the best based on the first measure but the worst based on the other. Consequently, evaluating association rules is highly domain-dependent since each domain has its own characteristics and accordingly its own requirements. As a result, it has always been a challenge to select a measure that well suits a given application domain.

To select the best measure given an application domain, the current practice is to consider a set of properties from that domain in order to make a choice against those properties. In the literature, three general selection approaches have been studied: *clustering methods*, *ranking methods*, and *multiple criteria decision aid methods*. These methods will be discussed in detail in Chapter 2.

1.6 Main contributions

Based on our understanding and experience in evaluating association rules, we propose that the following considerations be taken into account and be desirable for assessing and selecting appropriate measures for a given association rule mining task. (1) The measure assessment and selection are highly subjective. They depend on the user's requirements, their domain knowledge, and the particular application domain; (2) The measure assessment and selection are driven by the user, who might have various contradictory and inconsistent requirements; (3) The user's requirements may change over time, and accordingly, the result of assessing and selecting measures may change as well; and (4) The selection process should be computationally efficient.

Toward these goals, we first use a statistical-based approach to assess and select appropriate measures for a given application. A multiple-criteria decision making method is proposed to capture the user's desired requirements on properties quantitatively and uses them in assessing measures. The decision making technique in our work is a revised version of the *analytic hierarchy process* (*AHP*, for short) that makes use of chain-wise paired comparisons to prioritize user-specified properties. This approach is implemented and experimented using two case studies. It takes into account the user's requirements, avoids inconsistencies in her/his property specifications, and is invariant to the number of association rules.

We then propose a novel approach to dynamically evaluate association rules according to a collective effect of multiple measures, while taking the user's feedbacks into consideration. It utilizes the concepts and techniques of neural networks. The neural network implemented in our work has a multi-layered structure, along with a back-propagation learning algorithm. Circumventing property specifications, our approach learns the relative importance of measures for evaluating association rules in an iterative process. It interactively obtains the user's domain-dependent requirements and further uses them in the rule

evaluation process. A set of simulations have been carried out on two real-life data sets.

1.7 Thesis outline

Chapter 2 discusses the concepts of interestingness measures as well as various properties against which measures are assessed. It shows various approaches proposed in the literature to select an appropriate measure and explains the characteristics they aim to pursue as well as their advantages and drawbacks.

Chapter 3 introduces our first proposed measure assessment approach, which makes use of a revised version of AHP, analytic hierarchy process, to capture the user's subjective requirements on measure selection through chain-wise paired comparisons of properties. It also compares our approach to the others' on how they pursue various measure selection characteristics.

Chapter 4 presents our approach to evaluate association rules collectively using multiple measures. It first briefly discusses neural networks, employed in our approach, and their various models and algorithms. It then explains the reasons of choosing neural networks for the purpose of our approach. Furthermore, it details our proposed method and the process in which the learning and testing take place. The effectiveness of our approach is demonstrated through simulations, experiments, and discussions.

Finally, Chapter 5 summarizes our thesis with remarks and future work.

Chapter 2

Background

In order to handle the overwhelming number of association rules in an association mining process, various interestingness measures have been proposed to evaluate them, as how interesting or beneficial they are. These measures evaluate association rules according to criteria - conciseness, generality, utility, etc., which are specified by a user in an application domain.

In this chapter, interestingness measure concepts and their selection strategies are discussed. Section 2.1 discusses interestingness measures. Section 2.2 provides various criteria and properties against which interestingness measures are assessed. Section 2.3 presents methods proposed so far in the literature to select an appropriate interestingness measure.

2.1 Interestingness measures

Since there is no widespread agreement on a formal definition of interestingness, it is treated as a broad concept pursuing criteria such as *comprehensibility*, *conciseness*, *diversity*, *generality*, *novelty*, *peculiarity*, *surprisingness*, and *utility*. These eight criteria are used to determine interestingness and are thoroughly described in [10, 11, 24, 27, 30, 38, 43]. Brief descriptions of them are presented below:

- *Comprehensibility*. An association rule is comprehensible if it is easily understandable by a user.

- *Conciseness*. An association rule is concise if it contains a few attribute-value pairs[1]. In other words, an association rule is concise if it covers a small number of transactions in a transactional database.

[1] Attribute-value pair is a data model expressed as a collection of tuples (attribute name, value).

- *Diversity*. An association rule is diverse if the classes its items belong to differ significantly from each other.

- *Generality/Reliability*. An association rule is general/reliable if it covers a large number of transactions in a transactional database.

- *Novelty*. An association rule is novel if it is new to the user and it is not possible to infer it from other known association rules.

- *Peculiarity*. An association rule is peculiar if it largely differs from other discovered association rules.

- *Surprisingness/Unexpectedness*. An association rule is surprising if it contradicts the user's existing knowledge or expectations.

- *Utility/Actionability*. An association rule is of utility/actionability if its use contributes to reaching a goal.

Some of the above criteria, such as novelty and surprisingness, are dependent on each other, while some others, such as peculiarity and generality, are contradictory [11]. Given these, for instance, an association rule may be considered the best based on one criterion but the worst based on another [40, 43]. Moreover, the aforementioned criteria differ in their objectives. For example, comprehensibility, surprisingness, novelty, and utility consider the user's expectations, while conciseness, generality, diversity, and peculiarity consider the structure of rules to assess interestingness. Consequently, measures meeting these criteria fall into two groups: *subjective* and *objective*, which are discussed in the sequel.

2.1.1 Subjective measures

Subjective measures assess association rules by taking both the data and the user's prior knowledge into account. They reflect the views of the user on the data. Criteria such as

surprisingness, novelty, and utility are considered subjective, since they involve the user's prior knowledge and interests. These measures have been thoroughly studied throughout the literature [10, 11, 27, 30, 33, 36, 38, 49].

Klemettinen [22] demands the user to classify items to class hierarchies. Then, some *templates* are provided to describe a set of rules by specifying what class attributes occur in the antecedents and what occur in the consequences. After that, the user explicitly specifies whether a template is interesting or not. Accordingly, the approach retrieves rules matching the templates that are marked interesting.

Silberschatz and Tuzhilin [38] examine rule interestingness based on its actionability and unexpectedness. They classify the user's beliefs into two groups of *hard beliefs* and *soft beliefs*. Hard beliefs are constant and cannot be changed while soft beliefs are the ones that users are willing to change with new evidence. It is pointed out that the more an association rule contradicts the belief system, the more interesting it is. If an association rule contradicts hard beliefs, it is always interesting since it represents that the data used to derive that rule must be wrong. On the other hand, if an association rule contradicts soft beliefs, the stronger[2] the belief is, the more interesting the rule is, and accordingly, the more the belief needs to change.

To discover more general rules, Padmanabhan and Tuzhilin [33] propose a strategy that generates unexpected association rules with respect to the user's intuition by eliciting their beliefs about a domain and searching for rules contradicting them. Their proposed algorithm, called *ZoomUAR*[3], consists of two parts: *ZoominUAR* and *ZoomoutUAR*. Given a belief $X \rightarrow Y$, ZoominUAR first discovers all significant rules, $X, A \rightarrow \bar{Y}$, which are refinements to the beliefs such that the beliefs are contradicted. Then, ZoomoutUAR returns other more general rules of the form $X', A \rightarrow \bar{Y}$, where $X' \subset X$.

[2]Importance of an association rule is assigned through a weight. Approaches such as, *Bayesian networks*, *Dampster-Shafer*, etc., are used to set this weight. These approaches are beyond the scope of our discussions; however, interested readers are referred to [38]

[3]UAR stands for *Unexpected Association Rules*.

	B	$\bar{B}$	
A	AB	$A\bar{B}$	$Total\ A$
$\bar{A}$	$\bar{A}B$	$\bar{A}\bar{B}$	$Total\ \bar{A}$
	$Total\ B$	$Total\ \bar{B}$	N

Table 2.1: A 2×2 contingency table.

Sahar [36] provides the user with a subset of association rules and asks them to classify those rules as true/false and interesting/uninteresting and, accordingly, filters out the ones that are not interesting.

Liu *et al.* [27] propose an intuitive specification language to help the user specify their existing knowledge, beliefs, or concerns. Then, rules on the contrary are concluded unexpected, thus, interesting and novel. The work also proposes an interesting analysis system that asks the user to explicitly specify what types of association rules are interesting/uninteresting and only generates interesting rules that satisfy them, thus, avoiding generating a large number of rules.

Xin *et al.* [46] propose to discover interesting association rules through the user's interactive feedback. A model of the user's prior knowledge is built via asking them to rank a small subset of sample association rules according to their interests and is further used to rank association rules. This model is constantly updated and refined through the user's newly arrived feedbacks.

In addition to subjective measures, objective measures are proposed to assess interestingness based on the structure of association rules, as discussed in the following section.

2.1.2 Objective measures

Objective measures evaluate the interestingness of an association rule in terms of its structure, such as its statistical properties, its predictive performance, the underlying data, etc. Criteria such as conciseness, generality, peculiarity, and diversity are considered objective.

Table 2.2 lists a few common objective measures and their respective formula.

Interestingness Measures	Formula
Support	$P(AB)$
Confidence/Precision	$P(B\|A)$
Coverage	$P(A)$
Prevalence	$P(B)$
Recall	$P(A\|B)$
Specificity	$(\bar{B}\|\bar{A})$
Accuracy	$P(AB)+P(\bar{A}\bar{B})$
Lift	$P(B\|A)/P(B)$ or $P(AB)/P(A)P(B)$
Leverage	$P(B\|A)-P(A)P(B)$

Table 2.2: Objective measures for association rules [11, 41].

In the table, A and B represent the antecedence and the consequence of a rule. $N(A)$, $P(A)$, $P(\bar{A})$, $P(A|B)$, and $P(AB)$ respectively denote the total number of itemsets containing A, the probability of the occurrence of A in transactions in a transactional database, the complement of $P(A)$, the conditional probability of the occurrence of A given B, and the probability of the occurrence of A and B together in transactions in a transactional database. These measures are defined in terms of the frequency counts tabulated in a 2×2 contingency table as shown in Table 2.1, where N is the total number of transactions in a transactional database. They come from areas, such as *statistics*, *information theory*, *information retrieval*, etc. [11]. For more explanations of these measures, interested readers are referred to Appendix-A and [16, 32].

These objective measures, depending on the criteria they pursue, highly differ in ranking association rules. In order to assess objective interestingness measures and select an appropriate one for a given application domain, a set of properties is proposed. These properties represent the requirements of an application domain. The appropriate measure for a given application domain is selected based on how it fulfills the properties. The next section explains these properties and illustrates the measures pursuing them.

2.2 Properties for assessing measures

Given an application domain, appropriate objective measures[4] are selected to evaluate association rules for an association mining task based on a set of properties as follows.

P_1. This property represents that a rule occurred by chance has no interestingness value [11, 24].

P_2. This property states that, when the support for A and B are fixed, the larger the value of support for A and B, the larger their interestingness value is [11, 24].

P_3. This property states that, when the support for AB and B (A, respectively) are fixed, the smaller the support for A (B, respectively), the more interesting the association rule is [11, 24].

P_4. This property makes a distinction between measures evaluating rules $A \rightarrow B$ from those evaluating $B \rightarrow A$, resulting in different interestingness values [11, 24, 30, 36, 40].

P_5. This property states that, if there is no counterexample to the rule, interestingness stays constant.

P_6. This property states that the growth of a database does not influence interestingness value if the rate of A, $A \rightarrow B$, or B is constant [11, 24, 30, 36, 40].

[4]These properties are not considered for subjective measures, which are more concerned about the subjective views of the user on association rules.

P_7. This property states that having a slow decrease in the neighborhood of a logical rule rather than a fast or even linear decrease is desirable. It reflects the tolerability of a few counterexamples without the loss of interest [11, 24].

P_8. This property states that the threshold to identify interesting measures from uninteresting ones should be easy to choose and change [11, 24].

P_9. This property states that a measure is able to express a comprehensive idea of rule interestingness according to three following factors: easiness in its definition, value interpretation, and comprehensibility for the user [11, 24, 30, 40].

P_{10}. This property refers to the interestingness value that changes its sign if either the rows or the columns of the contingency table of a rule are permuted [11, 30, 36, 40].

P_{11}. This property states that the interestingness value remains the same if both the rows and columns of the contingency table of the rule are permuted. In fact, this property is a special case of P_{10} because permuting the rows (columns) causes the sign to change once and permuting the columns (rows) causes it to change back [11, 30, 36].

P_{12}. This property states that interestingness has no relationship with the number of the records that do not contain A and B [11].

Interestingness measures differ in pursuing the aforementioned properties, as illustrated in Table 2.3. Value 1 or 0 in each cell in the table is to show if an interestingness measure fulfills the respective property or not. For a complete table, see Appendix-B.

Interestingness Measures	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Support	0	1	0	1	0	0	1	1	1	0	0	0
Confidence/Precision	0	1	0	0	1	0	1	1	1	0	0	0
Coverage	0	0	0	0	0	0	0	1	1	0	0	0
Prevalence	0	0	0	0	1	0	1	1	0	0	0	0
Recall	0	1	0	0	1	0	0	1	1	0	0	1

Continued on next page

Table 2.3 – continued from previous page

Interestingness Measures	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Specificity	0	0	0	0	0	0	0	1	1	0	0	0
Accuracy	0	1	1	1	0	0	1	1	1	0	1	0
Lift/Interest	0	1	1	1	0	0	0	1	1	0	0	0
Leverage	0	1	1	0	0	0	1	1	0	0	0	1

Table 2.3: Interestingness properties fulfilled by measures [11, 24, 41, 48].

On the one hand, interestingness measures differ in pursuing the various properties, as illustrated in Table 2.3. On the other hand, each application domain requires its own characteristics and accordingly its own requirements on properties. Thus, it has been always a challenge to select a measure that well suits a given application domain [20]. The following section discusses the strategies proposed in the literature to select appropriate interestingness measures.

2.3 Selecting interestingness measures

The techniques proposed to select an appropriate interestingness measure fall into three categories based on the approach applied, namely, *clustering methods*, *ranking methods*, and *multiple criteria decision aid methods*.

- *Clustering methods*. Clustering methods (*CM*, for short) [25, 43] cluster measures based on their similar properties. For instance, Tan *et al.* [40, 41] state that the higher the correlation between the properties of two measures, the stronger the consistency between the measures. As a result, they are classified into one single class. For example, Table 2.4 illustrates a categorization of some interestingness measures based

Group	Interestingness Measures
1	$Odds\ ratio$, $Yule's\ Q$, $Yule's\ Y$
2	$Cosine$ and $Jaccard$
3	$Support$ and $Laplace$
4	$\phi - Coefficient$, $Collective\ Strength$, $Piatetsky - Shapiro$
5	$Gini\ index$
6	$Added - Value$, $Klosgen$
7	$Mutual\ Information$, $Certainty\ Factor$

Table 2.4: Groups of objective measures with similar properties [41].

on these methods.

Clustering methods are objective since they categorize association rules based on their properties. However, there exist a few shortcomings about them. These methods do not take the user's requirements into account and thus do not capture all the complexities that exist in her/his views in assessing measures. For example, although two measures might be clustered into a same class, the user may consider that one measure is of more importance than the other.

- *Ranking methods*. Ranking methods (*RM*) proposed by Sahar [36] and Tan *et al.* [40], provide the user with a small subset of discovered rules for ranking. Then, this subset is assessed by objective measures. Finally, a measure, whose ranking is the most similar to the user's ranking is selected as the appropriate measure.

  These methods perform well on taking the user's requirements into account but they fail to avoid the user's incomplete and growing domain knowledge that may cause some inconsistencies in evaluating association rules. In addition, if the number of association rules is large, selecting a suitable measure for a small subset may not be generalizable to the entire set of association rules.

- *Multiple-criteria decision aid methods*. Multiple-criteria decision aid methods (*MCDA*), proposed by Lenca *et al.* [25], select appropriate interestingness measures based on

the user's objectives. To take important criteria into account, a few MCDA procedures have been used. A preference function is presented to rank the desired properties and some weights are decided and assigned to them, which are further used to assess the relative measures.

In this method, the consistency and dynamic behavior of the user's requirements are not checked.

In the next two chapters, we will present our two approaches that attempt to remedy some deficiencies in the current interestingness measure selection methods in terms of their efficiency and effectiveness.

Chapter 3

An AHP-based Measure Assessment Method

The general approach to select appropriate interesting measures is to assess them against interestingness properties, as shown in Chapter 2. However, the approach has some deficiencies. (1) Many properties are incompatible to each other; (2) Properties are sometimes driven by users, who might have various contradictory rational requirements, resulting in inconsistencies in property specifications; and (3) The user's requirements or preferences may change over time, and accordingly, the required properties may change as well.

In this chapter, we propose an assessment approach that not only considers a user's [1] requirements represented by interestingness properties but also checks their consistency in property specifications. Moreover, the approach is invariant to the growth of data volume and the number of association rules to be evaluated, as shall be seen shortly.

The approach captures the user's desired requirements on properties quantitatively by utilizing so-called the *analytic hierarchy process*. The structure of this chapter is as follows. Section 3.1 introduces a revised version of the Analytic Hierarchy Process that prioritizes the user's subjective requirements. After that, Section 3.2 presents our approach to assess interestingness measures. Section 3.3 applies our proposed approach on two cases. Section 3.4 discusses the characteristics of measure selection approaches and compares our approach to the previous ones. Finally, Section 3.5 summarizes our discussions.

3.1 A revised analytic hierarchy process

The *analytic hierarchy process* (*AHP*, for short) is a multi-criteria decision making technique that was originally developed by Saaty [35]. Given an application domain, to achieve

[1] As previously mentioned in Chapter 1, it is supposed that an end user is using our approach whose specifications on desired properties are utilized to assess measures.

Scale	Importance
1	Equal importance
3	Moderate importance
5	Essential or strong importance
7	Very strong importance
9	Extreme importance
2,4,6	Intermediate values

Table 3.1: The typical importance values in CPC [35].

maximally a goal that has multiple properties, the user needs to prioritize, i.e., assigns importance to properties, under a set of requirements. This is essentially an optimization problem. In practice, the assignment process of preferences may introduce inconsistencies in property importance, given, for instance, incomplete domain knowledge, inconsistent requirements, etc. Moreover, it is usually hard to conduct the importance assignment to individual properties given the requirements. AHP provides a systematic and effective way to tackle this complicated prioritization by quantifying subjective requirements.

3.1.1 AHP details

The first step in AHP is to conduct pair-wise comparisons among properties. For a set of n properties, $\{P_1, P_2, P_3, \cdots, P_n\}$, of a given domain, there are $\binom{n}{2}$ comparisons. For each comparison between two properties, the user chooses a quantitative ratio value between them based on their qualitative requirements. This way, we encode qualitative domain-dependent requirements on the two properties into a quantitative value.

Since it is costly to conduct as many as $\binom{n}{2}$ comparisons, our work chooses a revised version of AHP, in which properties are engaged in a *chain-wise paired comparison* (*CPC*, for short) (requiring only $n-1$ property comparisons) [34].

The revised AHP starts from the user-specified importance value (denoted as *weight* (W_i)) for each property P_i, where $i = 1, 2, \cdots, n$. The user is invited, based on her/his

i	R_i	D_i	I_i	$\tilde{R}_i$	M_i	V_i
1	W_1/W_2	D_1	$\frac{D_1}{\prod D_i}$	$\frac{D_1}{\sqrt[n]{\prod D_j}}$	$\prod_{i=1}^{n-1} \tilde{R}_i$	$\frac{M_1}{\sum M_j}$
2	W_2/W_3	D_2	$\frac{D_2}{\prod D_i}$	$\frac{D_2}{\sqrt[n]{\prod D_j}}$	$\prod_{i=2}^{n-1} \tilde{R}_i$	$\frac{M_2}{\sum M_j}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$n-1$	W_{n-1}/W_n	D_{n-1}	$\frac{D_{n-1}}{\prod D_i}$	$\frac{D_{n-1}}{\sqrt[n]{\prod D_j}}$	$\tilde{R_{n-1}}$	$\frac{M_{n-1}}{\sum M_j}$
n	W_n/W_1	D_n	$\frac{D_n}{\prod D_i}$	$\frac{D_n}{\sqrt[n]{\prod D_j}}$	1	$\frac{M_n}{\sum M_j}$

Table 3.2: The CPC algorithm.

requirements on these properties, to compare them in pairs and assign them importance ratio values. A set of typical ratio values is illustrated in Table 3.1.

Let R_i denote the ratio of the ith property to its successor. We close the comparison chain with the ratio of the nth property to the 1st property. This is represented in the following equation.

$$R_i = \begin{cases} \frac{W_i}{W_{i+1}} & i = 1 \ldots n-1 \\ \frac{W_n}{W_1} & i = n \end{cases} \tag{3.1}$$

The value of R_i between two properties is specified by the user based on her/his domain requirements. It is easy to see that, under perfect consistency of the specifications, the product of R_i for all i should equal 1.

$$\prod_{i=1}^{n} R_i = \left(\frac{W_1}{W_2}\right)\left(\frac{W_2}{W_3}\right)\cdots\left(\frac{W_n}{W_1}\right) = 1 \tag{3.2}$$

Otherwise, there is a certain degree of inconsistency. This is exactly the situation where our approach needs to capture when the user tries to specify which properties are important and which are not based on her/his requirements for the given association mining task.

In order to do this, two different values of R_i are obtained: *direct* or *indirect*. The direct

value, D_i, is the value of W_i that is directly expressed by the user, whereas the indirect value, I_i, is the value computed in relative to the other direct values, i.e., the reciprocal of the product of other D_is.

$$I_i = \frac{D_i}{\prod D_i} \tag{3.3}$$

Obviously, I_i is equal to $\frac{W_i}{W_{i+1}}$ if a perfect consistency is achieved, i.e., $\prod D_i = 1$.

A better estimation of R_i, denoted by $\tilde{R}_i$, using the two different values D_i and I_i, is proposed [34] using the weighted geometric mean of D_i and I_i. Calculation for $\tilde{R}_i$ is shown by Equation 3.4.

$$\tilde{R}_i = D_i^{(n-1)/n} \cdot I_i^{1/n} = D_i^{(n-1)/n} \cdot (\frac{D_i}{\prod D_i})^{1/n} = D_i \cdot \frac{1}{\sqrt[n]{\prod D_i}} \tag{3.4}$$

The rationale behind Equation 3.4 is that, since direct values are more important than indirect values, each D_i weighs $(n-1)$ times more than I_i. Also the geometric mean is used since we expect that the ratio changes occur in a relative way. Note that we compare a pair of properties relatively, i.e., the importance of the property on the numerator relative to the one of the property on the denominator (we can consider this ratio as a percentage).

After introducing $\tilde{R}_i$, every direct value D_i is treated equally in the derivation process. It is noted that, while R_is are inconsistent in practice, $\tilde{R}_i$s are perfectly consistent, i.e., $\prod_{i=1}^{n} \tilde{R}_i = 1$.

Since $\tilde{R}_i$s are perfectly consistent, the weight for each interestingness property relative to the nth property, $M_i = W_i/M_n$, can be determined from $\tilde{R}_i$. The weight of the nth property relative to itself is 1, i.e., $M_n = 1$. The following recursion represents this calculation.

$$M_i = \begin{cases} \prod_{i}^{n-1} \tilde{R}_i & i = n-1 \cdots 1 \\ 1 & i = n \end{cases} \tag{3.5}$$

Finally, the normalized relative weights, V_i, for each of the interestingness properties P_i, are obtained by Equation 3.6. The values of V_is represent the priority of the relative property.

$$V_i = \frac{M_i}{\sum_{j=1}^{n} M_j} \tag{3.6}$$

The whole CPC algorithm is summarized in Table 3.2 [34, 44].

3.1.2 Measure of consistency

Consistency for chain-wise comparisons can be measured by the quantity, $\prod R_i$, which, when not equal to 1, shows a certain degree of inconsistency. In practice, some degree of inconsistency is unavoidable.

As pointed in [34], there are two types of consistencies: *ordinal* and *cardinal*. Ordinal consistency is maintained as long as the correct order of the interestingness properties is maintained. Cardinal consistency is more strict and requires that the importance values for all comparisons, assigned by the user, be correct [34]. We will discuss more on the cardinal consistency after the ordinal consistency.

The ordinal consistency is easily checked by comparing the magnitude of I_i to its corresponding D_i, i.e., the pair (I_i, D_i). The ordinal consistency is achieved if both I_i and D_i are of the same magnitude, i.e., both I_i and D_i are larger than or equal to 1 or both are smaller or equal to 1.

The cardinal consistency of the estimations is checked using the following equation, as introduced before.

$$\prod_{i=1}^{n} R_i = 1 \tag{3.7}$$

Full consistency is achieved if Equation 3.7 is satisfied. However, it is hard to achieve full consistency given the uncertainty in the user's requirements for a given association rule

mining task. Practically, a consistency level more than 90% is acceptable[2] [34].

During the above process, if any consistency violation occurs, the user should be notified to reevaluate the entire property pair-wise comparisons.

3.2 AHP-based measure assessment method

Our *AHP-based measure assessment* (*AHPMA*, for short) approach quantitatively prioritizes the user's subjective requirements. It includes the following four basic steps: (1) Present properties to the user; (2) Invite the user to rate the properties; (3) Assess the ratings and prioritize the properties; (4) Assess property fulfillment by each interestingness measure.

3.2.1 Presenting properties

The set of properties, P_1, P_2, $\cdots$, P_n are presented to the user. As discussed in Chapter 2, we consider a total number of twelve (12) interestingness properties for selecting measures to consider.

3.2.2 Rating properties

Using chain-wise paired comparisons, discussed in Section 3.1, the user assigns weights to the above properties in pairs based on their importance in her/his requirements for the given association mining task. For instance, based on the requirements, if the ratio of P_1 to P_2 is $4/3$, it states that P_1 is $4/3$ more important than P_2.

[2]The degree of acceptable cardinal consistency is determined through simulation experiments for a given application domain [34].

3.2.3 *Assessing and prioritizing properties*

The algorithm in Table 3.2 is employed to calculate the relative weight vector. This vector represents the prioritized properties. The most important property gains the maximum weight and the least important property gains the minimum weight. Since some properties gain extremely small weights in comparison to other properties, the effect of leaving them out (setting them to zero) from further consideration is negligible. Thus, the weights, V_j, of the remaining properties are adjusted to obtain the *adjusted weights*, denoted as $AW(V_j)$.

$$AW(V_j) = \frac{V_j}{\sum_{i=1}^{n} V_i}, 1 \leq j \leq n \tag{3.8}$$

where n is the number of interestingness properties.

3.2.4 *Assessing property fulfillment*

Considering the prioritized properties, the measures are assessed. For each measure (M_i), its overall *composite weight*, denoted as $CW(M_i)$, is computed based on the adjusted weights, $AW(P_j)$, discussed in the previous step, using $CW(M_i) = \sum_j AW(P_j) \cdot C(i,j)$, where $C(i,j)$ represents whether measure i fulfills property j (as indicated by 1) or not (as indicated by 0) For a complete table of measures fulfilling properties, see Chapter 2.

3.3 Case studies

To assess interestingness measures for a given application via our method, two case studies are discussed in this section.

3.3.1 Case one

This case is presented to illustrate the application of our approach in assessing interestingness measures, where the user provides consistent specifications on interestingness properties. The following scenarios are required in this case.

Scenario 1: Reliability of results is required. Thus, less counterexamples to an association rule are desired.

Scenario 2: A selected measure should be easily comprehensible by the user.

Scenario 3: The number of the transactions including neither A nor B should not affect the interestingness of an association rule.

Scenario 4: Given the reliability requirement, association rules happening by accident should be avoided.

Scenario 5: Rules of the form $A \rightarrow B$ and $B \rightarrow A$ should differ in their interestingness values.

Not all the interestingness measures are useful to pursue these requirements. Using our method, the 12 properties are prioritized and weighed based on the above scenarios. For example, the ratio of P_1 to P_2 is set to be 9 to 2, given the importance of P_1 based on *Scenario* 4. Moreover, the ratio of P_7 to P_8 is set to be 8 to 3, given the importance of P_7 based on *Scenario* 1. The rest of the weights are assigned according to the above scenarios. Table 3.3 illustrates the results. Column D_i represents the ratio values specified by the user according to the scenarios.

Both the ordinal and cardinal consistency, as explained in Section 3.1.2, for this prioritization are calculated. Ordinal consistency is checked by pairs (I_i, D_i). It is seen that the two corresponding columns all have the same magnitude. Thus, ordinal consistency is achieved. Then cardinal consistency is checked using Equation 3.7 and the result is 0.933, which is acceptable.

The last column shows the priority vector of the 12 properties. Since the weights for P_2,

i	R_i	D_i	I_i	$\tilde{R}_i$	M_i	V_i
1	W_1/W_2	$9/2 = 4.500$	4.821	4.527	0.623	0.136
2	W_2/W_3	$3/2 = 1.500$	1.608	1.509	0.138	0.030
3	W_3/W_4	$2/8 = 0.250$	0.268	0.252	0.091	0.020
4	W_4/W_5	$8/3 = 2.667$	2.858	2.683	0.363	0.079
5	W_5/W_6	$2/2 = 1.000$	1.072	1.006	0.135	0.030
6	W_6/W_7	$2/8 = 0.250$	0.268	0.252	0.134	0.029
7	W_7/W_8	$8/3 = 2.667$	2.858	2.683	0.534	0.117
8	W_8/W_9	$2/8 = 0.250$	0.268	0.252	0.199	0.044
9	W_9/W_{10}	$7/2 = 3.500$	3.751	3.521	0.792	0.173
10	W_{10}/W_{11}	$2/3 = 0.667$	0.715	0.671	0.225	0.049
11	W_{11}/W_{12}	$3/9 = 0.333$	0.357	0.335	0.335	0.073
12	W_{12}/W_1	$8/5 = 1.600$	1.715	1.610	1.000	0.219

Table 3.3: Consistent property specifications for the scenarios of case one.

P_3, P_5, P_6, P_8, and P_{10} are very small, we will not include them for further calculation. Thus, only the properties P_1, P_4, P_7, P_9, P_{11}, and P_{12} are considered and, using Equation 3.8, their corresponding weights are adjusted. $\sum_{i=1}^{12} V_i = 0.136 + 0.079 + 0.117 + 0.173 + 0.073 + 0.219 = 0.798$. As a result, $AW(P_1) = \frac{0.138}{0.798} = 0.171$, $AW(P_4) = \frac{0.08}{0.798} = 0.099$, $AW(P_7) = \frac{0.117}{0.798} = 0.147$, $AW(P_9) = \frac{0.173}{0.798} = 0.217$, $AW(P_{11}) = \frac{0.073}{0.798} = 0.092$, and $AW(P_{12}) = \frac{0.217}{0.798} = 0.274$,

Under the seven selected properties with their adjusted weights, the objective measures are examined with the aim to choose an appropriate measure, illustrated in Table 3.4. The rows represent the candidate measures while the columns represent the selected properties. For instance, the composite weight for *Support* is computed as follows: $CW(Support) = (0.171 \times 0) + (0.099 \times 1) + (0.147 \times 1) + (0.217 \times 1) + (0.092 \times 0) + (0.274 \times 0) = 0.463$.

Interestingness Measures	P_1	P_4	P_7	P_9	P_{11}	P_{12}	CW
	0.171	0.099	0.147	0.217	0.092	0.274	
Support	0	1	1	1	0	0	0.463

Continued on next page

Table 3.4 – continued from previous page

Interestingness Measures	P_1	P_4	P_7	P_9	P_{11}	P_{12}	CW
	0.171	0.099	0.147	0.217	0.092	0.274	
Confidence/Percision	0	0	1	1	0	0	0.364
Coverage	0	0	0	1	0	0	0.217
Prevalence	0	0	1	0	0	0	0.147
Recall	0	0	0	1	0	1	0.491
Specificity	0	0	0	1	0	0	0.217
Accuracy	0	1	1	1	1	0	0.555
Lift/Interest	0	1	0	1	0	0	0.316
Leverage	0	0	1	0	0	1	0.421
Added Value	1	0	1	1	0	0	0.535
Relative Risk	0	0	1	0	0	0	0.147
Jaccard	0	1	1	0	0	1	0.520
Certainty Factor	1	0	0	1	1	0	0.480
Odds ratio	0	1	0	0	1	0	0.191
Yule's Q	1	1	0	1	1	0	0.579
Yule's Y	1	1	0	1	1	0	0.579
Klosgen	1	0	0	1	0	0	0.388
Conviction	0	0	0	1	1	0	0.309
Collective Strength	0	1	0	1	1	0	0.408
Laplace Correction	0	0	1	0	0	0	0.147
Gini Index	1	0	0	0	1	0	0.263
Goodman and Kruskal	1	1	0	1	1	0	0.579
Normalized Mutual Information	1	0	1	0	1	0	0.410
J-Measure	1	0	0	1	0	0	0.388

Continued on next page

Table 3.4 – continued from previous page

Interestingness Measures	P_1	P_4	P_7	P_9	P_{11}	P_{12}	CW
	0.171	0.099	0.147	0.217	0.092	0.274	
One-Way Support	1	0	0	1	0	1	0.662
Two-Way Support	1	1	0	1	0	1	0.761
ϕ-Coefficient	1	1	0	1	1	1	0.853
Piatetsky-Shapiro	1	1	1	1	1	1	1.000
Cosine	0	1	0	1	0	1	0.590
Loevinger	1	0	0	1	0	1	0.662
Information Gain	1	1	0	0	0	1	0.544
Sebag-Schoenauer	0	0	0	1	0	1	0.491
Least Contradiction	0	0	0	1	0	1	0.491
Odd Multiplier	0	0	0	1	0	0	0.217
Example and Counterexample Rate	0	0	0	1	0	1	0.491
Zhang	1	0	0	0	0	0	0.171

Table 3.4: Composite weights for measures based on property importance for case one.

As shown in Table 3.4, the measure $Piatetsky-Shapiro$ is chosen with the highest composite weight.

3.3.2 Case two

This case is presented aiming at explaining occurrence of inconsistency. Taking the following scenarios into account, the user is asked to rank the properties.

Scenario 1: Accidental rules have no interestingness value.

i	R_i	D_i	I_i
1	W_1/W_2	$4/2 = 2.000$	0.056
2	W_2/W_3	$8/2 = 4.000$	0.112
3	W_3/W_4	$5/2 = 2.500$	0.070
4	W_4/W_5	$4/6 = 0.667$	0.019
5	W_5/W_6	$3/2 = 1.500$	0.042
6	W_6/W_7	$8/7 = 1.143$	0.032
7	W_7/W_8	$4/7 = 0.571$	0.016
8	W_8/W_9	$7/3 = 2.333$	0.066
9	W_9/W_{10}	$1/2 = 0.500$	0.014
10	W_{10}/W_{11}	$3/2 = 1.500$	0.042
11	W_{11}/W_{12}	$6/3 = 2.000$	0.056
12	W_{12}/W_1	$7/9 = 0.778$	0.022

Table 3.5: Inconsistent property specifications for the scenarios of case two.

Scenario 2: Since novelty is an important property, association rules with less frequent items in the antecedent and consequent are considered more interesting.

Scenario 3: The size of the database may enlarge over time. Interestingness of a rule should be invariant to this change.

Scenario 4: Interestingness criteria may change over time, thus interestingness threshold should easily change considering new criteria.

Scenario 5: Interestingness value changes upon row or column permutations.

Scenario 6: Interestingness value stays the same upon both row and column permutation.

Considering the ratios provided by the user in Table 3.5, for $i = 1,2,3,5,6,8,10$ and 11, one finds that the magnitude of I_i differs from the one of the corresponding D_i. Thus, the ordinal consistency is violated. Moreover, the value of cardinal consistency is calculated using Equation 3.7 with a value of 35.56, which is not acceptable. As a result, the ranking is cardinally inconsistent too. This inconsistency has occurred due to violating the transitivity in "is greater than relation"[3] in the user's property preferences. The specifications state that

[3]A relation is called transitive whenever for given properties, P_i, P_j, and P_k, the property P_i is more

i	R_i	D_i	I_i	$\bar{R}_i$	M_i	V_i
1	W_1/W_2	$8/3 = 2.667$	2.788	2.678	3.496	0.161
2	W_2/W_3	$3/8 = 0.375$	0.392	0.377	1.305	0.060
3	W_3/W_4	$7/2 = 3.500$	3.657	3.514	3.467	0.159
4	W_4/W_5	$2/3 = 0.667$	0.697	0.670	0.987	0.045
5	W_5/W_6	$4/6 = 0.667$	0.697	0.670	1.473	0.068
6	W_6/W_7	$8/3 = 2.667$	2.787	2.678	2.200	0.101
7	W_7/W_8	$3/7 = 0.429$	0.448	0.431	0.822	0.038
8	W_8/W_9	$7/4 = 1.750$	1.829	1.757	1.908	0.088
9	W_9/W_{10}	$3/7 = 0.429$	0.448	0.431	1.086	0.050
10	W_{10}/W_{11}	$5/3 = 1.667$	1.742	1.674	2.521	0.116
11	W_{11}/W_{12}	$3/2 = 1.500$	1.567	1.506	1.506	0.069
12	W_{12}/W_1	$2/7 = 0.286$	0.299	0.287	1.000	0.046

Table 3.6: Consistent property specifications for the scenarios of case two.

P_6 is more preferred than P_7, P_7 is more preferred than P_5, and P_5 is more important than P_6, which contradicts the transitivity in the user's preferences.

Consequently, the user is asked to reevaluate the properties for their importance values. It should be noted that this process may need to repeat several times until both types of consistency are passed. We show one set of consistent specifications on the twelve properties in Table 3.6.

Given the same magnitudes of all the pairs (I_i, D_i), the ordinal consistency is met. In addition, the cardinal consistency, which is equal to 0.957 according to Equation 3.7, is obtained. Column V_i shows the priority of the properties. Only P_1, P_2, P_3, P_5, P_6, P_8, P_{10}, and P_{11} are taken into account in this case. Since the weights for P_4, P_7, P_9 and P_{12} are very small, they are neglected for further considerations. The adjusted weights of the other properties are computed, as follows. $\sum_{i=1}^{12} V_i = 0.161 + 0.060 + 0.159 + 0.068 + 0.101 + 0.088 + 0.116 + 0.069 = 0.822$, $AW(P_1) = \frac{0.161}{0.822} = 0.196$, $AW(P_2) = \frac{0.060}{0.822} = 0.073$, $AW(P_3) = \frac{0.159}{0.822} = 0.193$, $AW(P_5) = \frac{0.068}{0.822} = 0.083$, $AW(P_6) = \frac{0.101}{0.822} = 0.123$, $AW(P_8) =$

preferred than the property P_j and P_j is more important than the property P_k, P_k is less preferred than P_i; otherwise, it is intransitive.

$\frac{0.088}{0.822} = 0.107$, $AW(P_{10}) = \frac{0.116}{0.822} = 0.141$, and $AW(P_{11}) = \frac{0.069}{0.822} = 0.084$.

Under the eight selected properties with their adjusted weights, the objective measures and their properties are examined.

Interestingness Measures	P_1	P_2	P_3	P_5	P_6	P_8	P_{10}	$P11$	CW
	0.196	0.073	0.193	0.083	0.123	0.107	0.141	0.084	
Support	0	1	0	0	0	1	0	0	0.180
Confidence	0	1	0	1	0	1	0	0	0.263
Coverage	0	0	0	0	0	1	0	0	0.107
Prevalence	0	0	0	1	0	1	0	0	0.190
Recall	0	1	0	1	0	1	0	0	0.263
Specificity	0	0	0	0	0	1	0	0	0.107
Accuracy	0	1	1	0	0	1	0	1	0.457
Lift	0	1	1	0	0	1	0	0	0.373
Leverage	0	1	1	0	0	1	0	0	0.373
Added Value	1	1	1	0	0	1	0	0	0.569
Relative Risk	0	1	1	0	0	1	0	0	0.373
Jaccard	0	1	1	0	0	1	0	0	0.373
Certainty Factor	1	1	1	0	0	1	0	1	0.653
Odds ratio	0	1	1	1	1	1	1	1	0.804
Yule's Q	1	1	1	1	1	1	1	1	1.000
Yule's Y	1	1	1	1	1	1	1	1	1.000
Klosgen	1	1	1	0	0	1	0	0	0.569
Conviction	0	1	0	1	0	1	0	1	0.347
Collective Strength	0	1	1	0	0	1	1	1	0.598
Laplace Correction	0	1	0	0	0	1	0	0	0.180

Continued on next page

Table 3.7 – continued from previous page

Interestingness Measures	P_1	P_2	P_3	P_5	P_6	P_8	P_{10}	$P11$	CW
	0.196	0.073	0.193	0.083	0.123	0.107	0.141	0.084	
Gini Index	1	0	0	0	0	1	0	1	0.387
Goodman- and Kruskal	1	0	0	0	0	1	0	1	0.387
Normalized- Mutual Information	1	1	1	0	0	1	0	1	0.652
J-Measure	1	0	0	1	0	1	0	0	0.386
One-Way Support	1	1	1	0	0	1	0	0	0.569
Two-Way Support	1	1	1	0	0	1	0	0	0.569
ϕ-Coefficient	1	1	1	0	0	1	1	1	0.794
Piatetsky-Shapiro	1	1	1	0	0	1	1	1	0.794
Cosine	0	1	1	0	0	1	0	0	0.373
Loevinger	1	1	0	1	0	1	0	0	0.459
Information Gain	1	1	1	0	0	1	0	0	0.569
Sebag-Schoenauer	0	1	1	1	0	1	0	0	0.457
Least Contradiction	0	1	1	0	0	1	0	0	0.375
Odd Multiplier	0	1	1	1	0	1	0	0	0.457
Example and- Counter-example Rate	0	1	1	1	0	1	0	0	0.457
Zhang	1	0	0	0	0	0	0	0	0.196

Table 3.7: Composite weights for measures based on property importance of case two.

Characteristics	CM	RM	MCDA	AHPMA
C_1		✓	✓	✓
C_2				✓
C_3				
C_4	✓		✓	✓

Table 3.8: Measure assessment approaches and their characteristics.

As shown in Table 3.7, the measures *Yule's Q* and *Yule's Y* are chosen, since they fulfill the majority of the properties with the highest composite weight.

3.4 Different measure assessment methods

The other measure selection approaches proposed so far, namely the clustering methods, the ranking methods, and the multiple-criteria decision aid methods, as discussed in Chapter 2, aim at pursuing the following characteristics [24, 25, 36, 40, 43]:

C_1*: Taking the user's preferences into account* [11, 25, 30, 33, 36, 38]. The user should provide enough information on what rules are considered interesting.

C_2*: Avoiding inconsistencies in the user's decisions* [35]. The user may be inconsistent in providing inputs. A selection strategy should deal with those inconsistencies.

C_3*: Adapting to the varying user's requirements* [25, 27, 38]. A selection strategy should be able to adapt itself with the user's dynamic requirements.

C_4*: Invariant to the number of association rules* [25, 43]. An appropriate selection strategy should not be affected by the number of association rules.

These approaches differ in fulfilling the above characteristics, as shown in Table 3.8. A symbol (✓) is used to represent the satisfaction of the characteristics by the respective measures.

Since clustering methods (CM) classify measures based on either their properties or association rules, they do not take the user's preferences into account and accordingly do

not conduct consistency checking. Furthermore, because they focus on properties, they are invariant to the number of association rules [25, 43]. Thus, they are only marked on C_4.

Ranking methods (RM) provide the user with a small subset of discovered rules for ranking. Thus, it is marked on C_1. However, if the number of association rules is large, selecting a suitable interestingness measure for them may be problematic. To make it worse, given the user's incomplete and growing domain knowledge, there might be some inconsistencies in ranking association rules [36, 40].

Multiple-criteria decision aid methods (MCDA) perform the selection based on the user's objectives and a few MCDA procedures. The consistency of the user's requirements is not checked. Moreover, it does not pay attention to dynamic changes in the user's requirements [24, 25].

Our proposed method, (AHPMA), takes the user's requirements into account and detects any inconsistencies in their specifications. In addition, it is not influenced by the number of association rules, i.e., the time complexity of our method only depends on the number of properties under consideration and is invariant to the size of the data set and the number of association rules mined. But, like the other three methods, it is not able to adapt itself to the user's dynamic requirements.

3.5 Summary

Our main contribution in this chapter is a novel approach assessing interestingness measures for association rules, based on a set of user's requirements. We attribute this to the Analytic Hierarchical Process (AHP). Some initial case studies are presented to show the effectiveness of our approach. It is also noted that none of the measure assessment methods so far deals with the dynamic behavior of the user's requirements. In other words, these methods must be reapplied in case of changes in the user's behavior. To remedy this problem, in the next chapter, an approach is proposed to dynamically evaluate association

rules according to a collective effect of multiple measures while taking the user's dynamic requirements and feedbacks into consideration.

Chapter 4

A Dynamic Composite Approach for Evaluating Association Rules

As previously discussed, the following considerations are desirable for assessing and selecting appropriate measures for a given association mining task. (1) The measure assessment and selection strategies are highly subjective. They depend on a user's requirements, their domain knowledge, and the application domain; (2) The assessment and selection are driven by the user, who might have various inconsistent contradictory rational requirements; and (3) The user's requirements may change over time, and accordingly, the result of assessing and selecting measures may change as well.

In this chapter, a novel approach is proposed that takes all the above issues into consideration. It dynamically evaluates association rules according to a composite and collective effect of multiple measures, while involving feedbacks from the user.

In a nutshell, in our approach, instead of striving for one best measure for a given association mining task, we propose to include multiple measures in the evaluation process of association rules. Thus, the ranking score of an association rule no longer depends only on one measure. Rather, it is a composite and collective effect of multiple measures.

In addition, our approach has a merit of circumventing the step which asks the user to specify her/his requirements. It invites the user to engage in a training process, during which their requirements are embedded and any inconsistencies are detected.

Given an association rule r and a set of n measures $\{M_1, M_2, \cdots, M_n\}$, let $M_i(r)$ denote the score of measure M_i applied to the rule r. The *Composite Score* (*CS*, for short) of the rule r is defined as follows.

$$CS(r) = F(\sum_{i=1}^{n} W_i \cdot M_i(r)) \tag{4.1}$$

where W_i ($\in [0,1]$) encodes the relative importance of measure M_i by which we evaluate the rule r and $\sum_{i=1}^{n} W_i = 1$. $F(\cdot)$ is a strictly monotonically increasing function. One simple example is $F(x) = x$. However, as shall be seen in the sequel, in our implementation of Equation 4.1, the score of each measure is normalized into a certain range, such as $(0,1)$, and function $F(\cdot)$ is realized using a *neural network* with *back-propagation* learning ability.

If the user desires to ignore some particular measure, its corresponding importance weight will eventually become zero. A more "appropriate" measure will be assigned a higher importance value. Obtaining importance weights, W_is, in Equation 4.1 highly depends on the user's requirements in an application domain.

The structure of this chapter is as follows. Section 4.1 briefly discusses neural networks and their various models and algorithms. Moreover, it explains the application of neural networks for the purpose of our thesis. In addition, it discusses the influential parameters of the efficiency and accuracy of neural networks. Section 4.2 discusses our proposed approach - a neural-network-based evaluation process with back-propagation learning ability. Section 4.3 and Section 4.4 demonstrate the effectiveness of our approach through simulations, experiments, and discussions. Finally, Section 4.5 summarizes the chapter.

4.1 Neural networks

4.1.1 Definition and structure of neural networks

Neural networks are originally developed in the field of machine learning to imitate the neurophysiology of human brains through the combination of simple connected computational elements, known as *neurons*, in a highly interconnected system [12]. The interconnection strength between the neurons stores domain knowledge that is acquired through a learning process [5, 37]. Neural networks have been applied for a broad spectrum of applications, such as aerospace, automotive, electronics, entertainments, food industry, insurance, mar-

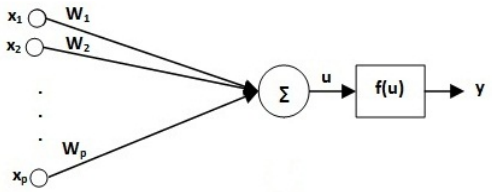

Figure 4.1: A neuron.

keting, manufacturing, medical, real estate, robotics, securities, etc. [19], performing a variety of tasks, including prediction, function approximation, pattern classification, clustering, etc. [12]. We present below a few definitions of a neural network.

1. A neural network is a massively distributed system including a collection of interconnected neurons that incrementally learn from their environment to capture essential linear and nonlinear trends in complex data and to provide reliable predictions for new situations containing even partial information [37].

2. A neural network is a machine capable of learning as a result of adjustment of the state of itself in response to the random data generated by the environment [4].

3. A neural network is a massive parallel distributed processor made up of simple processing units, which store prior knowledge and make it available for use [15].

4. A neural network is a massively parallel system, inspired by the structure of the brain, used to address problems that are cumbersome with traditional methods [8].

Inspired from the above definitions, we define a neural network as *a collection of interconnected neurons that incrementally learn from its environment to solve a complex problems*. Some features of this definition will be further explained below.

- *A collection of interconnected neurons*. *Neurons*, also known as *perceptrons*, are basic elements of a neural network. They are first proposed by Frank Rosenblatt in 1958 [8, 19]. As shown in Figure 4.1, a neuron receives its inputs as $\{x_1, x_2, \cdots, x_n\}$

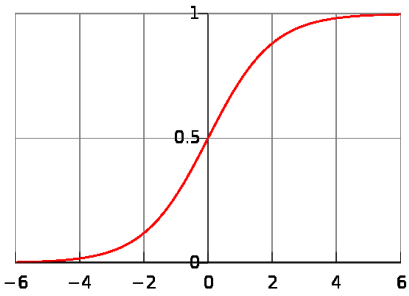

Figure 4.2: The sigmoid function.

and their relative weights as $\{w_1, w_2, \cdots, w_n\}$ and calculates the weighted summation of the inputs into a single-value output.

$$u = \sum_{i=1}^{n} w_i x_i \tag{4.2}$$

This output is then input to the *transfer function*, also called the *activation function*, f, which calculates the final output of the neuron. Many functions can be utilized to be a transfer function. However, it is highly recommended to use the functions that are more commonly used, such as the *sigmoid function*[1] given its simple derivative and automatic gain control, i.e., small inputs are allowed to pass through without excessive attenuation while large inputs are accommodated without saturation.

The sigmoid function is an s-shaped function, as shown in Figure 4.2, which produces values over the range of $(0, 1)$. It is mathematically stated as Equation 4.3 [12].

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.3}$$

Neurons are interconnected in *layers* to form the *neural network model*, as illustrated in Figure 4.3. The first and the last layer are the *input* and *output* layers that are

[1]Transfer functions could be of other forms, such as *linear functions*, *hyperbolic tangent function*, etc. [5, 9, 12, 19].

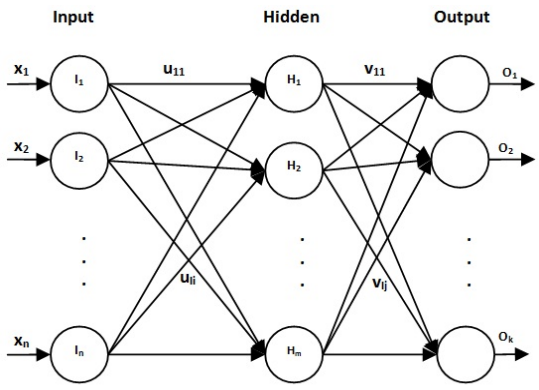

Figure 4.3: Multi-layer neural network [9].

respectively used for input and output purposes. There may exist some layers between them, called the *hidden layers*. Neural networks vary based on the *number of layers* and *neurons*. These features [19] are selected depending on the application domain. For example, *Multiple adaptive linear neuron*, a single layer neural network based on the *perceptron*, is used for problems with multiple inputs; *Feed-forward neural networks* are widely used in predictions; *Radial basis function networks* are used for modeling uncertain and nonlinear functions; *Hopfields networks* are used as associative memories; *Self-organizing neural networks* are used for finding unknown clusters in data; *Probabilistic neural networks* are used to predict the probability distribution of classes using the distribution of the training sets; and *Time-varying neural networks* are used when the input and output are dynamic, mostly in speech recognition, motor control, signal processing, and vision.

- *Incrementally learning from its environment*. This feature reflects the iterative nature of the learning process of neural networks. Basically, the neural network is exposed to and stimulated by an environment. In other words, the set of input data, $\{x_1, x_2, \cdots, x_n\}$, called the *training set*, are given, shown in Figure 4.3. Accordingly, the structure and parameters of the neural network are adjusted. Then, its response to the environment, $\{O_1, O_2, \cdots, O_k\}$ in Figure 4.3, is obtained. If the response is

satisfactory, compared to the expected response provided by the user, which means the difference *(error)* between the output of the neural network and the expected results is less than a specified threshold, the training is over, the weights, u_{ij} and v_{lj} in Figure 4.3, are fixed, and the model is built; otherwise, this process is repeated until the value of error is minimized to be less than a defined threshold.

Many learning algorithms could be applied depending on the model of the neural network. Some of these learning algorithms are *Hebbian learning algorithm* used for self-organizing networks, *unsupervised/competitive learning algorithm* used to respond correctly without the involvement of an external agent, and *back-propagation learning algorithm* mostly used in multi-layer neural networks. Interested readers are referred to [5, 12, 19, 23, 37] for more details.

- *Solving complex problems*. Although there exist other methods for prediction purposes, such as *regression analysis*, neural networks outperform them in many aspects. Neural networks are able to derive complex linear and nonlinear relationships among data without having them explicitly formalized to make predictions. Moreover, they include less formal statistical training to develop. They are less sensitive to inconsistent data. In addition, they can be developed using a variety of learning algorithms, which makes them flexible to work with [31, 42].

Apart from the models and learning algorithms utilized in neural networks, they are mainly affected by the parameters they apply in their learning processes and models. The following section describes these parameters.

4.1.2 Issues in neural networks

The accuracy and performance of neural networks depend on the parameters used in their learning algorithms and models. Some of these major parameters are as follows.

- *The size of training set*. The size of a training set should be large enough to cover a range of inputs with diverse features. It should be pointed out that the size should not be too large to cause the network to be *over-trained*, in which the network memorizes the training set rather than generalizing from it, resulting in performing well on the training set but poorly on the datasets when in use [5, 23].

- *The range of training set*. For the learning purposes, it is better to use a range that behaves in a linear way. In addition, due to the linear behaviour of the activation functions near zero, for instance the *sigmoid function*, it is better to make use of the range $[-1, 1]$. *Convertors* may be applied to convert the value of inputs to this particular range [5, 9, 28]. Moreover, for the neural network to learn very well and work efficiently and accurately in real life situations, the input is diversely taken from the entire range without any redundancy [13, 47].

- *The number of the units in hidden layers*. Units stand for neurons in hidden layers, $\{H_1, H_2, \cdots, H_m\}$, in Figure 4.3. The more the number of the units, the more patterns can be accurately recognized. However, this may lead to over-training. To avoid that problem and obtain a desirable number of hidden layers, one should start with one layer and increment it gradually through simulations [5, 23, 31].

- *The learning algorithms*. The learning algorithms differ on how they minimize the error and are chosen based on an application domain [23].

- *Learning rate*. Learning rate controls how fast weights, u_{ij} and v_{lj} in Figure 4.3, adjust. The larger the learning rate, the faster the weights change. The best approach for the learning rate is to be large at the beginning to get in the vicinity of the best weights. Then, when the network gets closer to the desired solution, the learning rate should decrease so the network can fine tune to the most optimal weights [5, 23, 31].

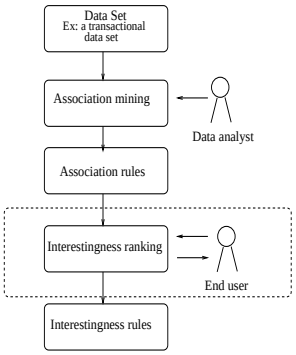

Figure 4.4: The overview of an association mining system.

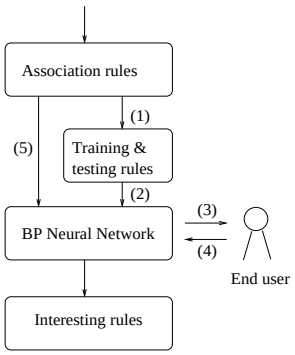

Figure 4.5: The high-level view of our approach.

- *Momentum*. Momentum keeps the direction the weight adjustments is heading in unchanged. Thus, it speeds up the neural network convergence, and is in conjunction with the magnitude of the previous weight adjustments. Without this term, it takes a long time for a network to reach its optimal value. A network with a high momentum allows the weights to oscillate more freely. Thus, reaching the minimum error becomes a slower process. It is advisable to start with a high momentum value and gradually decrease it during the training process [5, 9, 23].

- *The number of iterations*. An iteration is completed once an output of the network is obtained through a learning process. However, minimizing the value of the error requires the learning process to iterate. The number of iterations is chosen in advance depending on the error threshold and on the user's domain knowledge and experiences. If it is large, it results in the over-training problem. Thus, it should be chosen in a way to avoid this problem [23, 31].

Taking these issues into account, in the next section, we present our novel approach to evaluate association rules collectively by multiple measures.

4.2 A dynamic and composite association evaluation

Our proposed approach aims at developing a method for assessing interestingness measures that not only takes the user's requirements into account while avoiding inconsistencies, but also captures changes in the user's requirements. After incorporating our approach, an association mining system is depicted in Figure 4.4. Any preference changes in evaluating association rules will be fedback into the system by the user who is interacting with the system and accordingly the system will trigger its learning process and adjust itself.

At the heart of our interestingness evaluation approach lies a multi-layer neural network along with the back-propagation learning ability. Figure 4.5 shows the high-level view of our approach. After the required data sets, such as transactional records, are processed by some association mining algorithm, such as the *Apriori* algorithm[2], a set of association rules are obtained. A subset of the rules will be carefully selected (as indicated by (1) in Figure 4.5) by the user to make the training and testing rule set. First, the training rule set is input into the back-propagation neural network to learn the connection weights among the neurons (as indicated by (2)) in the network structure. In this process, the output from the network is presented to the user (as indicated by (3)) and the feedbacks are collected (as indicated by (4)). Once the training process is finished, the testing process starts using the testing rule set. This training-and-testing process may need to go through several rounds[3], until some criteria are satisfied (to be discussed in detail shortly). We call this the *offline* mode of our approach.

Once the neural network is ready, it can be used to evaluate association rules (as indicated by (5)). It should be noted that the use of the neural network is constantly monitored by the user. Once a discrepancy between the neural network output and the desired target

[2]See Chapter 1.

[3]A round is an interval during which the training set is in the process of learning. It consists of a number of iterations equal to the number of rules in the training set. Once all the rules in the training set are gone through the training process but the finishing criteria are not met, the next round of the training process is triggered.

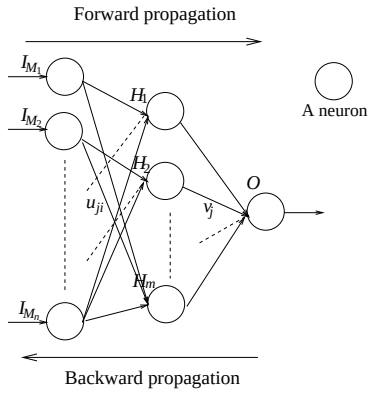

Figure 4.6: A hidden neuron receiving back-propagated errors from other neurons.

output is found for a given association rule, the learning process is triggered again. This corresponds to the situation where the user changes their requirements in evaluating rules. These changes are gradually incorporated into the neural network and reflected in the future outputs. In contrast to the offline mode, we call this the *online* mode of our approach. Toward this end, it is reasonable for us to require that the user's requirements be stable for a certain period of time before next changes.

4.2.1 Approach details

Due to the ability of multi-layer feed-forward neural networks and the back-propagation learning algorithm to learn complex relationships among data and make good predictions in an application domain, they are chosen for calculating the composite score of each association rule.

Neural network structure

Multi-layer feed-forward neural networks include an input layer and an output layer with no interconnections between the neurons within a layer. But, different from single layer forward neural networks, it has a number of intermediate or hidden layers existing between

the input and the output layer.

The neural network structure adopted in our approach has one or more hidden layers. Figure 4.6 shows the structure consisting of an input layer, a hidden layer and an output layer. This structure is utilized to illustrate scoring and learning process of a given rule. Our discussions can be easily generalized into the situation where more than one hidden layers are involved, which are discussed in our simulations and empirical studies in Section 4.3.

For an association mining task, the set of n measures $\{M_1, M_2, \cdots, M_n\}$ are used, shown in Figure 4.6, as n input neurons, each of which is denoted by I_{M_i}, where $i = 1, 2, \cdots, n$. The hidden layer consists of m neurons, each of which is denoted as H_j, where $j = 1, 2, \cdots, m$. In the literature, the selection of the number of hidden neurons in a multi-layer neural network is the state-of-the-art and highly depends on the application domain. We will empirically study it in Section 4.3. There is only one neuron, denoted as O, at the output layer, whose output corresponds to the final output score of a given association rule.

For each connection between I_{M_i} and H_j, there is a weight u_{ji} associated with it. Similarly, there is a weight v_j associated with the connection between H_j and O. The entire purpose of learning weights, W_is in Equation 4.1, is to assign appropriate values to weights u_{ji} and v_j such that each association rule is scored as desired.

Scoring an association rule

An association rule is scored adapting the formulas in [5, 9, 23]. Given an association rule r, measure M_i is applied and its measure scores, denoted by $M_i(r)$, where $i = 1, 2, \cdots, n$ are obtained. These measure scores will be used as the inputs to neurons I_{M_i} at the input layer, respectively. By abusing the notation, let I_{M_i} also be the value it obtains from $M_i(r)$.

For each neuron H_j at the hidden layer, its weighted input, H_j^i, is collected from the neurons at the input layer by Equation 4.4,

$$H_j{}^i = \sum_{i=1}^{n} u_{ji} \cot I_{M_i}, \tag{4.4}$$

while its output, denoted by H_j^o, is calculated as

$$H_j{}^o = 1/(1 + exp(-H_j^i)), \tag{4.5}$$

where $j = 1, 2, \cdots, m$. We choose the sigmoid function as the activation function. Once the outputs of the hidden neurons are calculated, the input to the output neuron O, denoted as O^i, is obtained through

$$O^i = \sum_{j=1}^{m} v_j \cdot H_j^o. \tag{4.6}$$

Using the sigmoid function again, the output from O, called the *output score* from the neural network for the rule r, denoted as $OS(r)$, is calculated as

$$OS(r) = 1/(1 + exp(-O^i)). \tag{4.7}$$

It is easy to see that $OS(r) \in (0, 1)$. The above process is known as the *forward propagation* [9] in neural networks, as shown in Figure 4.6.

Learning an association rule

Once rule r is scored, in the training process, its output score from the neural network is compared with its *target score*, denoted as $TS(r)$, specified by the user. Any discrepancy between them is used to adjust the weights u_{ji} and v_j, through the back-propagation learning process. In the learning process, the error in the neural network's approximation of the score of the rule r is calculated by Equation 4.8.

$$E(r) = \frac{1}{2}(TS(r) - OS(r))^2 \tag{4.8}$$

In the spirit of the *gradient descent*[4], the partial derivative of $E(r)$ with respect to each weight in the neural network is required to be computed, as shown below.

For each weight v_j, its adjustment is proportional to the corresponding derivative, calculated as

$$\Delta v_j^{new} = -\eta \frac{\partial E}{\partial v_j} + \alpha \Delta v_j^{old}, \tag{4.9}$$

where terms η and α are respectively the learning rate and momentum, as discussed in Section 4.1.2, and Δv_j^{new} is the current adjustment on v_j while Δv_j^{old} is the previous one. We can write

$$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial O^i} \frac{\partial O^i}{\partial v_j}. \tag{4.10}$$

Considering Equation 4.6 we see

$$\frac{\partial O^i}{\partial v_j} = H_j^o. \tag{4.11}$$

To compute $\frac{\partial E}{\partial O^i}$ we apply the chain rule to write this partial derivatives as the product of two factors, one factor reflecting the change in error as a function of the output of the unit and the other reflecting the change in the output as a function of changes in the input. Thus, we have

$$\frac{\partial E}{\partial O^i} = \frac{\partial E}{\partial OS(r)} \frac{\partial OS(r)}{\partial O^i}. \tag{4.12}$$

The second factor is computed by Equation 4.7.

[4]Gradient descent is a first-order optimization algorithm. It finds a local minimum of a function by taking steps proportional to the negative of the gradient of the function at the current point.

$$\frac{\partial OS(r)}{\partial O^i} = \frac{\partial}{\partial O^i}\left(\frac{1}{1+exp(-O^i)}\right) \tag{4.13}$$

$$= -\frac{\frac{\partial(1+exp(-O^i))}{\partial O^i}}{(1+exp(-O^i))^2} \tag{4.14}$$

$$= \frac{1}{(1+exp(-O^i))^2}(exp(-O^i))$$

$$= \left(\frac{1}{1+exp(-O^i)}\right)\left(\frac{exp(-O^i)}{1+exp(-O^i)}\right)$$

$$= OS(r)(1-OS(r)).$$

To compute the first factor of Equation 4.12, we follow the definition of E,

$$\frac{\partial E}{\partial OS(r)} = -(TS(r) - OS(r)). \tag{4.15}$$

Thus, the error value from the output neuron can be written as

$$\frac{\partial E}{\partial v_j} = -(TS(r) - OS(r))OS(r)(1-OS(r))H_j^o. \tag{4.16}$$

Consequently, the new value of v_j is calculated as

$$\Delta v_j^{new} = \eta(TS(r) - OS(r))OS(r)(1-OS(r))H_j^o + \alpha\Delta v_j^{old}, \tag{4.17}$$

and the new weight v_j is obtained as $v_j^{new} = v_j^{old} + \Delta v_j^{new}$ [5, 9, 23].

Similarly, as for each weight u_{ji}, its adjustment is proportional to the corresponding derivatives. Thus, the error value from this unit is calculated as

$$\frac{\partial E}{\partial u_{ji}} = \frac{\partial E}{\partial H_j^i}\frac{\partial H_j^i}{\partial u_{ji}}. \tag{4.18}$$

Considering Equation 4.4, the second factor is

$$\frac{\partial H_j^i}{\partial u_{ji}} = I_{M_i}. \tag{4.19}$$

Applying the chain derivatives, we have

$$\frac{\partial E}{\partial H_j^i} = \frac{\partial E}{\partial H_j^o} \frac{\partial H_j^o}{\partial H_j^i}. \tag{4.20}$$

The second factor is computed by Equation 4.5,

$$\begin{aligned} \frac{\partial H_j^o}{\partial H_j^i} &= \frac{\partial}{\partial H_j^i} \left(\frac{1}{1+exp(-H_j^i)} \right) \\ &= \frac{1}{(1+exp(-H_j^i))^2} (exp(-H_j^i)) \\ &= \left(\frac{1}{1+exp(-H_j^i)} \right) \left(\frac{exp(-H_j^i)}{1+exp(-H_j^i)} \right) \\ &= H_j^o (1 - H_j^o). \end{aligned} \tag{4.21}$$

To compute the first factor, chain derivatives are applied,

$$\frac{\partial E}{\partial H_j^o} = \frac{\partial E}{\partial O^i} \frac{\partial O^i}{\partial H_j^o}. \tag{4.22}$$

The second factor is computed using Equation 4.6,

$$\begin{aligned} \frac{\partial O^i}{\partial H_j^o} &= F^{'}(H_j^o) \\ &= v_j. \end{aligned} \tag{4.23}$$

Applying chain derivatives again, we have

$$\frac{\partial E}{\partial O^i} = \frac{\partial E}{\partial OS(r)} \frac{\partial OS(r)}{\partial O^i}. \tag{4.24}$$

The second factor is computed using Equations 4.13 and 4.13

$$\frac{\partial OS(r)}{\partial O^i} = OS(r)(1 - OS(r)). \tag{4.25}$$

According to the definition of E, the first factor is calculated as

$$\frac{\partial E}{\partial OS(r)} = -(TS(r) - OS(r)). \tag{4.26}$$

Consequently, the error value can be written as

$$\frac{\partial E}{\partial u_{ji}} = -(TS(r) - OS(r))OS(r)(1 - OS(r))v_j H_j^o(1 - H_j^o)I_{M_i}. \tag{4.27}$$

The new value of u_{ji} is calculated as

$$\Delta u_{ji}^{new} = \eta(TS(r) - OS(r))OS(r)(1 - OS(r))v_j H_j^o(1 - H_j^o)I_{M_i} + \alpha \Delta u_{ji}^{old}. \tag{4.28}$$

Thus, the new weight u_{ji} is obtained as $u_{ji}^{new} = \Delta u_{ji}^{new} + u_{ji}^{old}$ [5, 9, 23].

4.2.2 Training process

In our simulations, the *cross validation* is used to perform training and testing processes. In this validation method, the available samples are divided into P disjoint subsets, with $P-1$ subsets to be used for training and the remaining subset for testing. In our simulations, P is

selected to be 2 and could be generalized to other numbers with ease.

For each association rule r_i in the training rule set, it is input into the neural network and the steps above are followed to adjust the weights in the network. In one round, this process goes through all the association rules in the training set. The *Mean Square Error* (*MSE* for short) for the round is calculated as $MSE = \frac{1}{N}\sum_{i=1}^{N}(TS(r_i) - OS(r_i))^2$, where N is the number of association rules in the training rule set. The training process continues until *MSE* is no more than a user's defined threshold or the user's defined maximum number of rounds has been reached.

4.2.3 Testing process

Given the trained network and the testing rule set, the network is further assessed on the testing rule set. The quality of the neural network is assessed by calculating the difference between the target and the output scores, also called the *residual value* for each association rule, r, denoted as $RV(r)$, in the testing rule set. Obviously, the less the residual value, the more precise the predictions of the neural network are, and consequently, the better the neural network is [21, 31].

4.2.4 Other implementation details

Since our neural network takes inputs in the range $[-1,1]$, each measure score s of a given association rule is normalized as $s' = 2 \times \frac{s-a}{b-a} - 1$, where $[a,b]$ is the original range of score s for the measure and $s' \in [-1,1]$. For instance, for the range $[-7,-1]$, score $s = -5$ is normalized as $s' = 2 \times \frac{-5-(-7)}{6} - 1$, resulting in $s' = -\frac{1}{3}$.

In some implementations of back-propagation neural networks, one can add the so-called *bias neurons* to offset the origin of the activation function, which allows faster convergence of training process. A bias neuron provides a constant input in Equations 4.4

and 4.6 while its weights on its connections with other neurons are engaged in the same back-propagation learning process. In our implementation, since we do not have such a need, bias neurons are not included.

4.3 Simulations

The simulations are carried out on two real-life data sets - the *car evaluation* data set and *solar flare* data set, retrieved from the *UCI Repository* [6, 7].

4.3.1 Results on car data set

Preparing data sets

The car evaluation data set includes 1728 records with 6 input attributes, such as *buying-price* (*bp* for short), *maintenance-price* (*mp*), *number-of-doors* (*dn*), *number-of-people* (*pn*), *size-of-lug-boot* (*lb*), and *safety* (*sf*). In order to make use of the *Apriori* algorithm [2] to extract association rules, the attributes and their respective values are combined and converted into 21 binary attributes. For instance, attribute *bp* has four values, namely *vhigh*, *high*, *med*, and *low*. Correspondingly, we have four binary attributes as *bp_vhigh*, *bp_high*, *bp_med*, and *bp_low*.

The Apriori algorithm[5] is applied on the car data set with different minimum supports. For the sake of space, the results for the setting where the minimum support is 3.5%, where there are 529 resultant association rules, are reported. The minimum confidence is set to be 0%. The reason for this, according to our use of the car data set, is to produce more association rules.

[5] See Chapter 1.

Interestingness Measure	R_1	R_2	R_3
Laplace Correction (LAP)	0.112	0.335	0.335
Sebag $-$ *Schoenauer* (SEB)	0.125	0.500	0.500
$Confidence\ (CONF)$	0.112	0.333	0.333
Example and Counterexample Rate (ECR)	0.484	0.871	0.871
Coverage (COV)	0.333	0.111	0.111
Support (SUP)	0.037	0.037	0.037
Least Contradiction (LC)	-0.555	0.555	0.926
GINI Index (GI)	0.500	-0.500	-0.500
$Specificity$	-0.900	0.667	0.667
Odds Ratio	-0.889	0.600	0.600
Jaccard	0.091	0.091	0.091
Goodman & Kruskal	-0.250	-0.250	-0.250
Collective Strength	0.588	-0.901	-0.901
Leverage	0.074	0.296	0.296
Accuracy	0.630	0.630	0.630
Prevalence	0.111	0.333	0.333
Yule's Q	0.500	0.500	0.500
Recall	0.333	0.111	0.111

Table 4.1: Interestingness measure scores for R_1, R_2, and R_3.

Creating training and testing rule sets

From the resultant 529 rules, our work, for the purpose of simulations, has attempted various numbers of association rules for creating the training, ranging from three (3) to six (6). It is found that increasing further the number of training rules does not actually improve the performance of the neural network. The results reported here are for the training set consisting of six (6) rules.

Objective interestingness measures in our simulations, as discussed in Chapter 2, include *Recall*, *Specifity*, *Accuracy*, *Lift*, *Support*, *Confidence*, *Coverage*, *Prevalence*, *Leverage*, *Added Value*, *Relative Risk*, *Jaccard*, *Certainty Factor*, *Odds Ratio*, *Yule's Q*, *Yule's Y*, *Klosgen*, *Conviction*, *Collective Strength*, *Laplace Correction*, *Gini Index*, *Goodman and Kruskal*, *Normalized Mutual Information*, *J-Measure*, *One-Way Support*, *Two-Way Sup-*

port, *Linear Correlation Coefficient*, *Piatetsky-Shapiro*, *Cosine*, *Loevinger*, *Information Gain*, *Sebag-Schoenauer*, *Least Contradiction*, *Odd Multiplier*, *Example and Counterexample Rate*, and *Zhang*[11].

After thorough investigation of the resultant scores, it is found that some measures, such as *Lift*, *Added Value*, *Klosgen*, *Normalized Mutual Information*, etc., return the same scores[6] for each association rule chosen. Consequently, they are not included in our simulations.

As one example, for the three association rules, $R_1 : \{pn\_More\} \rightarrow \{lb\_Big, sf\_High\}$, $R_2 : \{pn\_4, lb\_Small\} \rightarrow \{sf\_Low\}$, and $R_3 = \{bp\_vhigh \rightarrow lb\_small\}$, their respective measure scores are illustrated in Table 4.1.[7]

The target scores of the rules are simulated by assigning higher weights for some particular measures. For instance, one may consider the measures, *LAP*, *SEB*, *CONF*, *ECR*, more important, while paying less attention to the other measures. Therefore, for rule R_1 in Table 4.1, its target score can be calculated as $[x/4(0.112+0.125+0.112+0.484)+(1-x)/14(0.333+0.037+(-0.555)+0.5+0.9+(-0.9)+0.091+(-0.25)+0.588+0.074+0.63+0.111+0.5+0.333)]$, where x could be chosen as 0.8 to simulate the user's requirements and be averaged over the four chosen measures. It should be noted that in using our approach in real applications, the assignment of rule target scores in a training rule set should be conducted by the user carefully and may not be so uniform, as shown in this example, among the chosen measures.

A similar procedure is followed to create the testing rule set using the rules from the association mining process. Seven (7) association rules are chosen for the testing rule set.

The training process commences by inputting each rule in turn in the training rule set to the neural network, following the scoring and learning steps as outlined in Section 4.2.1.

[6]This situation is generally not true.

[7]For simplicity and clarity, only three training rules are shown. The other rules in the training rule set have similar situations.

This process continues until either the maximum number of rounds or the minimum *MSE* is met. Our simulations use 300000 for the maximum number of rounds and 0.0000000001 for *MSE*.

Our network structure starts on with four layers with 18, 18, 18, and 1 neurons in the input layer, the first hidden layer, the second hidden layer and the output layer, respectively. Given 18 inputs (measures) and 1 output, 18 and 1 are chosen for the number of neurons in the first and the last layer. The number of neurons on the hidden layers is empirically attempted in our simulations.

Different neural network structures, in terms of the involved number of input neurons, number of hidden layers, number of hidden neurons in the hidden layers, learning rate, and momentum are used. At the end, one network is chosen and is further evaluated for its structure and performance.

Learning rates

A series of experiments are conducted, aiming to find an appropriate learning rate on neural networks with four layers, including one input layer, two hidden layers, and an output layer, with 18, 18, 18, and 1 neurons, respectively. The value of momentum is constant and set to be 0.3. The learning rate is firstly assigned to be 0.2 and is gradually increased to 0.3, 0.4, 0.5, 0.6, and 0.7. For the sake of space, a few results are illustrated in Figures 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14. It is easy to see that all the training processes follow a similar trend, i.e., when progressing with more rounds, the *MSE* becomes smaller and smaller. In the figures, the X-axis represents the number of rounds whereas the Y-axis represents the absolute residual value. In order to help understand the figures better, we magnify the first 50 rounds, the last 50 rounds, and the entire training process for the settings where $\eta = 0.2$ in Figures 4.7, 4.8, and 4.9, respectively. In all of our simulations using the above settings, the convergence is demonstrated in training the

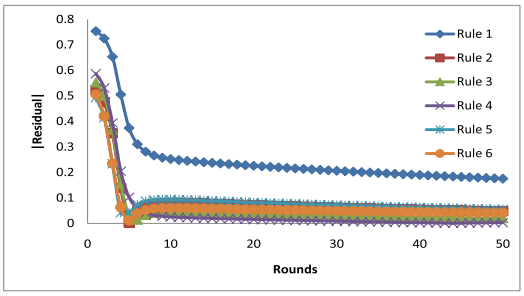

Figure 4.7: The first 50 rounds of the training process (η = 0.2).

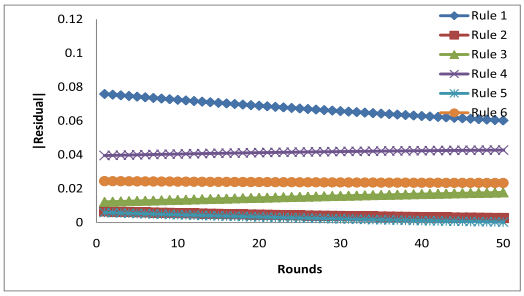

Figure 4.8: The last 50 rounds of the training process (η = 0.2).

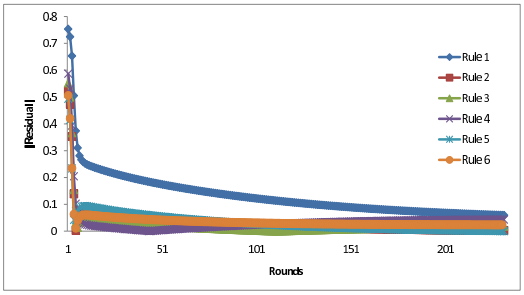

Figure 4.9: The entire training process (η = 0.2).

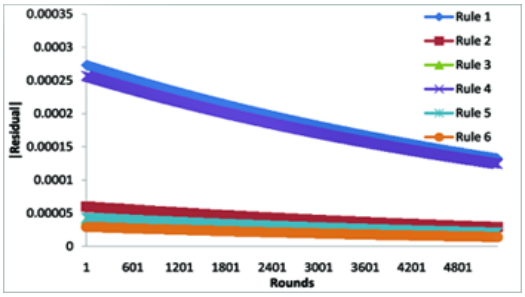

Figure 4.10: The entire training process (η = 0.3).

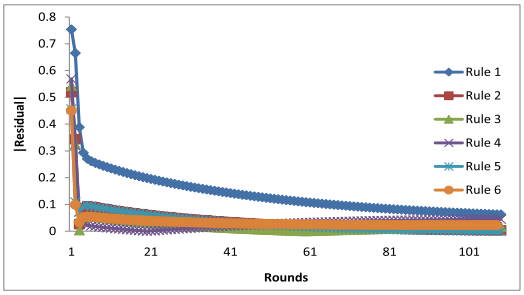

Figure 4.11: The entire training process (η = 0.4).

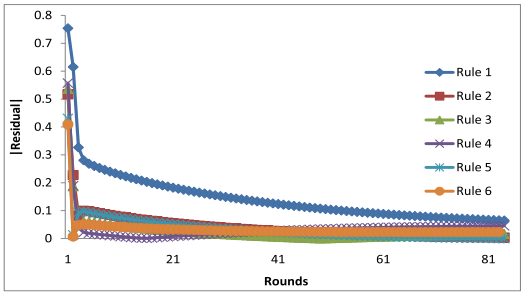

Figure 4.12: The entire training process (η = 0.5).

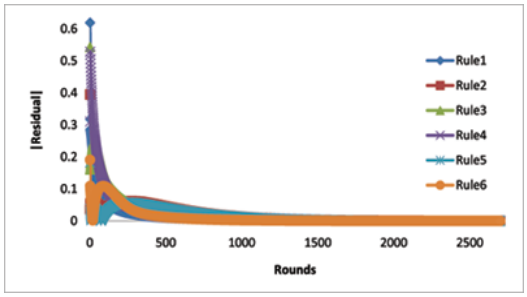

Figure 4.13: The entire training process (η = 0.6).

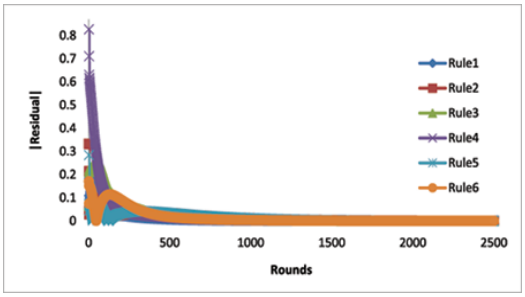

Figure 4.14: The entire training process (η = 0.7).

neural networks.

After the training, the networks are tested using the testing rule set, which is composed of seven association rules, as aforementioned. The respective residual results are shown in Tables 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7.

TS	OS	$\|RV\|$
0.197	0.254	0.057
0.418	0.413	0.005
0.324	0.377	0.053
0.386	0.378	0.007
0.360	0.395	0.035
0.380	0.401	0.021
0.254	0.254	0.000

Table 4.2: Absolute residual values ($\eta = 0.2$).

TS	OS	$\|RV\|$
0.197	0.197	0.000
0.418	0.419	0.001
0.324	0.378	0.054
0.386	0.390	0.005
0.360	0.374	0.014
0.380	0.377	0.003
0.254	0.254	0.000

Table 4.3: Absolute residual values ($\eta = 0.3$).

TS	OS	$\|RV\|$
0.197	0.255	0.059
0.418	0.414	0.004
0.324	0.377	0.053
0.386	0.378	0.007
0.360	0.397	0.037
0.380	0.403	0.023
0.254	0.255	0.001

Table 4.4: Absolute residual values ($\eta = 0.4$).

TS	OS	$\|RV\|$
0.197	0.256	0.060
0.418	0.415	0.003
0.324	0.377	0.053
0.386	0.378	0.007
0.360	0.398	0.038
0.380	0.404	0.024
0.254	0.256	0.002

Table 4.5: Absolute residual values ($\eta = 0.5$).

TS	OS	$\|RV\|$
0.197	0.224	0.028
0.418	0.417	0.001
0.324	0.372	0.048
0.386	0.375	0.010
0.360	0.408	0.048
0.380	0.414	0.034
0.254	0.224	0.029

Table 4.6: Absolute residual values ($\eta = 0.6$).

TS	OS	$\|RV\|$
0.197	0.254	0.057
0.418	0.412	0.005
0.324	0.377	0.053
0.386	0.378	0.007
0.360	0.394	0.034
0.380	0.400	0.020
0.254	0.254	0.000

Table 4.7: Absolute residual values ($\eta = 0.7$).

In general, increasing the value of the learning rate leads to large changes that increases the speed of convergence. However, oscillations may occur, i.e., no matter how long one attempts to train a neural network, it does not converge such that the error does not meet the specified MSE. For instance, although learning rate increases in Figure 4.10, it results in increasing the number of rounds that is due to the oscillations occurred while minimizing the value of MSE. The network with learning rate of 0.2 performs better with a smaller MSE value in a less number of rounds.

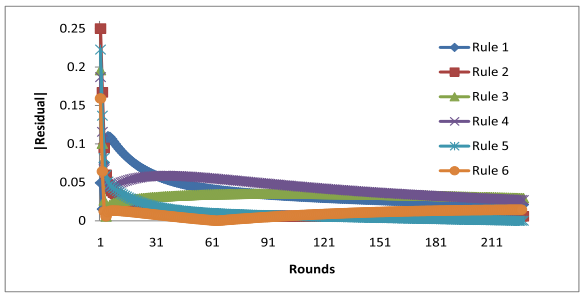

Figure 4.15: The entire training process (number of hidden neurons = 17).

Hidden neurons

To achieve an "appropriate" number of neurons in the hidden layers, a variety of numbers in these layers are attempted. Due to the space limitations, the simulation results on a neural network with 17 hidden neurons, learning rate and momentum of 0.2 are illustrated in Figure 4.15 and Table 4.8. It should be noted that other simulations follow the same trend. As a result, decreasing the number of neurons in the hidden layer leads to increasing the number of rounds as well as the total error.

TS	OS	$\|RV\|$
0.197	0.293	0.096
0.418	0.383	0.035
0.324	0.395	0.070
0.386	0.407	0.021
0.360	0.382	0.022
0.380	0.378	0.002
0.254	0.293	0.096

Table 4.8: Absolute residual values (number of hidden neurons = 17).

Hidden layers

The number of hidden layers in the previous network is set to be zero and one with 0.2 for both the learning rate and momentum. Figures 4.16 and 4.17 present the results. Furthermore, the absolute residual values are shown in Tables 4.9, and 4.10.

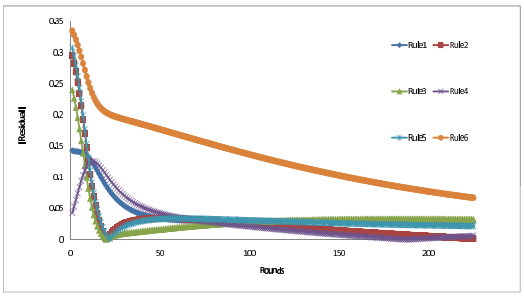

Figure 4.16: The entire training process (number of hidden layers = 0).

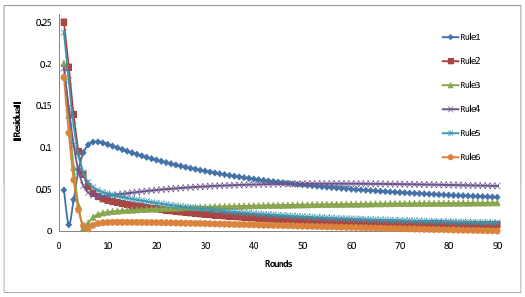

Figure 4.17: The entire training process (number of hidden layers = 1).

TS	OS	$\lvert RV \rvert$
0.197	0.225	0.028
0.418	0.426	0.008
0.324	0.370	0.046
0.386	0.384	0.001
0.360	0.443	0.083
0.380	0.420	0.040
0.254	0.225	0.029

Table 4.9: Absolute residual values (number of hidden layers = 0).

TS	OS	$\lvert RV \rvert$
0.197	0.237	0.040
0.418	0.392	0.026
0.324	0.407	0.082
0.386	0.430	0.045
0.360	0.435	0.075
0.380	0.414	0.034
0.254	0.237	0.017

Table 4.10: Absolute residual values (number of hidden layers = 1).

Consequently, neural networks with the number of hidden layers other than two reduce the number of rounds but perform poorly on predicting the score of the unknown rules. Thus, due to the performance and accuracy of the network with two hidden layers, it is chosen for our objective and is further tested on the value of momentum.

Momenta

Three neural networks with momenta of 0, 0.2, and 0.4 are simulated. When momentum is assigned to 0 and 0.2, the number of rounds drastically increases over 400000, which is unacceptable based on our goal to decrease the time cost. The user senses the delay in terms of training process time and individual round response time. Moreover, the network performs poorly when the momentum is 0.4 when it is applied on the testing rule set. The

results for the momenta of 0.2 and 0.4 are displayed in Tables 4.11 and 4.12.

TS	OS	$\|RV\|$
0.197	0.255	0.059
0.418	0.412	0.005
0.324	0.377	0.053
0.386	0.378	0.007
0.360	0.394	0.034
0.380	0.400	0.020
0.254	0.255	0.002

Table 4.11: Absolute residual values ($\alpha = 0.2$).

TS	OS	$\|RV\|$
0.197	0.197	0.000
0.418	0.419	0.001
0.324	0.378	0.054
0.386	0.390	0.005
0.360	0.374	0.014
0.380	0.377	0.003
0.254	0.197	0.057

Table 4.12: Absolute residual values ($\alpha = 0.4$).

Result analysis

Considering the performance and accuracy of the aforementioned neural networks in evaluating association rules, a four-layer neural network consisting of 18 neurons in the first and the two hidden layers and one neuron in the output layer, with the values of 0.3 for momentum and 0.2 for learning rate, is selected for car data set. In Section 4.4, its accuracy and performance are evaluated.

4.3.2 Results on the solar data set

The second simmulation has been conducted on the solar flare data set, which consists of 1389 records with 10 attributes, including *largest spot size*, *spot distribution*, *activity*, etc. Similar to our empirical studies on the car data set, the association rule set is obtained using the Apriori algorithm. Then, following our approach further simulations have been conducted, shown in Table 4.13, and consequently, the network with three layers, including 18, 16, and 1 neurons in the input, hidden, and output layer is selected as the appropriate network for this data set. We have chosen 17 association rules to train the network and five

rules to test it.

Moreover, due to the accuracy and performance of this neural network and its convergence in a less number of rounds, the value of 0.5 is chosen for both the learning rate and momentum. The training and testing for this network are shown in Figure 4.18 and Table 4.14. As can be seen, it bears a similar trend to the results for the simulations on the car data set.

Layers	*Neurons*	*Rounds*	η	TS	OS	$\lvert RV \rvert$
4	$18, 18, 18, 1$	642	0.1	0.049	0.052	0.003
				0.236	0.202	0.034
				0.374	0.348	0.027
				0.148	0.118	0.030
				0.946	0.808	0.138
4	$18, 18, 18, 1$	1365	0.3	0.049	0.208	0.029
				0.236	0.002	0.234
				0.374	0.346	0.028
				0.148	0.122	0.025
				0.946	0.799	0.146
4	$18, 18, 18, 1$	909	0.5	0.049	0.051	0.002
				0.236	0.202	0.034
				0.375	0.336	0.038
				0.148	0.120	0.028
				0.946	0.804	0.141
4	$18, 18, 18, 1$	69	0.7	0.049	0.049	0.000
				0.236	0.173	0.063
				0.374	0.298	0.076
				0.148	0.101	0.047

Continued on next page

Table 4.13 – continued from previous page

Layers	*Neurons*	*Rounds*	η	TS	OS	$\lvert RV \rvert$
				0.946	0.810	0.135
3	18, 18, 1	128	0.1	0.049	0.096	0.047
				0.236	0.266	0.029
				0.375	0.298	0.076
				0.148	0.161	0.013
				0.946	0.680	0.266
3	18, 18, 1	92	0.3	0.049	0.061	0.012
				0.236	0.238	0.001
				0.375	0.375	0.000
				0.148	0.122	0.025
				0.946	0.774	0.172
3	18, 18, 1	79	0.5	0.049	0.049	0.000
				0.236	0.207	0.030
				0.375	0.351	0.023
				0.148	0.102	0.046
				0.946	0.773	0.173
3	18, 18, 1[8]	9174	0.5	0.049	0.054	0.005
				0.236	0.233	0.004
				0.375	0.376	0.001
				0.148	0.115	0.033
				0.946	0.809	0.136
				0.049	0.046	0.003
				0.236	0.205	0.031

Continued on next page

[8]From this study to the rest of the studies in the table, the value of momentum is set to *zero*.

Table 4.13 – continued from previous page

Layers	*Neurons*	*Rounds*	η	TS	OS	$\lvert RV \rvert$
3	18, 18, 1	285	0.3	0.375	0.357	0.017
				0.148	0.098	0.049
				0.946	0.776	0.170
3	18, 18, 1	488	0.5	0.049	0.054	0.005
				0.236	0.229	0.008
				0.375	0.377	0.002
				0.148	0.114	0.034
				0.946	0.811	0.134
3	18, 17, 1	201	0.5	0.049	0.058	0.009
				0.236	0.229	0.008
				0.375	0.376	0.002
				0.148	0.126	0.022
				0.946	0.877	0.069
3	18, 16, 1	541	0.5	0.049	0.053	0.004
				0.236	0.222	0.014
				0.375	0.377	0.002
				0.148	0.117	0.030
				0.946	0.902	0.044
3	18, 14, 1	281	0.5	0.049	0.049	0.000
				0.236	0.196	0.040
				0.375	0.353	0.021
				0.148	0.109	0.039
				0.946	0.920	0.025
				0.049	0.063	0.014

Continued on next page

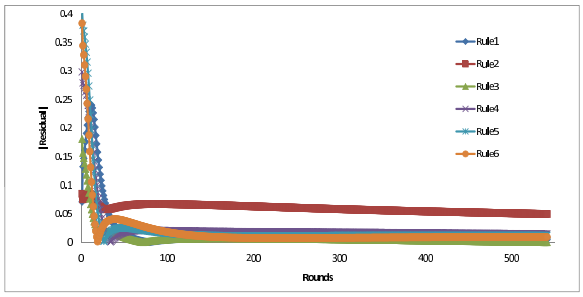

Figure 4.18: The entire training process (solar data set).

TS	OS	$\lvert RV \rvert$
0.049	0.053	0.004
0.222	0.236	0.014
0.375	0.377	0.002
0.148	0.117	0.030
0.946	0.902	0.044

Table 4.14: Absolute residual values (solar data set).

Table 4.13 – continued from previous page

Layers	*Neurons*	*Rounds*	η	TS	OS	$\lvert RV \rvert$
3	$18, 15, 1$	51	0.5	0.236	0.231	0.006
				0.375	0.376	0.001
				0.148	0.135	0.013
				0.946	0.876	0.070

Table 4.13: The results of the solar flare empirical studies.

4.4 Discussions

Our approach is discussed regarding two main aspects: firstly, the design and performance of the neural networks in our approach and secondly, its effectiveness of evaluating interestingness of association rules.

4.4.1 The neural networks

We mainly discuss the neural networks in our approach according to the following criteria [1]:

- *Performance*. The results from the neural network should match the results the user expects. The user is required to assign target scores in its training and testing rule sets. All of our experiments as discussed in Section 4.3 support that the neural networks we have designed perform as expected in the training process. Moreover, the neural network accuracy in terms of small residual values in the testing process is high, resulting in negligible errors.

- *Convergence*. Convergence is concerned with the capabilities of the network in the learning procedure and minimizing the value of the error to obtain an optimal solution in a reasonable period of time. Also, as shown in our simulations, all the experiments converge as expected, given the consistent specifications of target scores in the training and testing process. However, we will also discuss shortly the situation where inconsistent target scores are brought into the neural networks.

- *Generalization*. Generalization is the ability of the neural network to perform well on the data set outside the training set. In our work, this criterion requires that the rules used for testing predictive capabilities of the model be different from the ones used to develop and train the model. If the accuracy of the testing result meets the threshold expected, the neural network is considered to have learned well. In our experiments, we follow the process of cross validation. As shown in the experiment results in Section 4.3, the neural networks we have designed perform as expected in this regard.

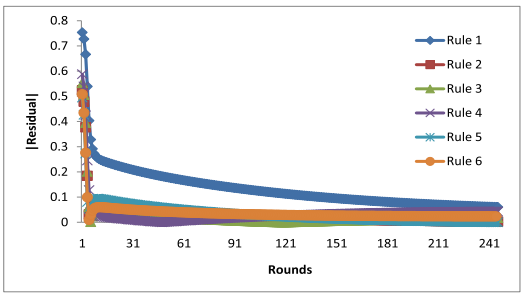

Figure 4.19: The entire training process (a new training set) for car.

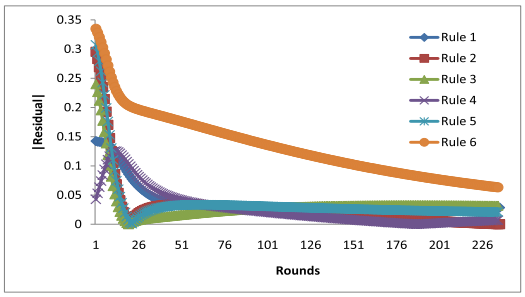

Figure 4.20: The entire training process (new training and measure sets) for car.

- *Stability*. Stability is the consistency of the results, during the validation phase using different samples of data. Regarding this criterion, our selected neural network was trained and tested using two new sample sets of data. The first sample focuses on a new training set, as explained in Section 4.3.1, and the result is illustrated in Figure 4.19 and Table 4.15. Then, the second simulation focuses on another set of measures that includes *simplicity*. The corresponding results are presented in Figure 4.20 and Table 4.16. We also apply our approach to the solar data set and have obtained similar results. We believe that our approach should be applicable to other data sets and in real applications.

TS	OS	$\|RV\|$
0.344	0.344	0.000
0.841	0.834	0.007
0.677	0.709	0.032
0.814	0.719	0.095
0.340	0.812	0.028

Table 4.15: Absolute residual values (a new training set).

TS	OS	$\|RV\|$
0.663	0.659	0.004
0.541	0.527	0.014
0.523	0.536	0.013
0.567	0.540	0.027
0.663	0.659	0.004

Table 4.16: Absolute residual values (new training and measure sets).

TS	OS	$\lvert RV \rvert$
0.344	0.947	0.603
0.841	0.607	0.234
0.677	0.636	0.041
0.814	0.619	0.194
0.840	0.215	0.625

Table 4.17: Absolute residual values (random inconsistent training and testing set).

4.4.2 The effectiveness of our approach

As discussed in Chapter 2, there is a set of main objectives of interestingness measures to evaluate association rules. We discuss the effectiveness of our approach according to them as follows.

Taking the user's requirements into account [25, 36, 38]. The decision on interestingness of a rule should be made based on the user's feedbacks. To meet this criterion, the type of learning algorithm used in this chapter is *supervised learning*, where the user's ideas are fed into the network for training and learning.

Avoiding inconsistencies in the user's requirements [35]. Any inconsistencies given the user's incomplete knowledge should be calculated and removed. One strength of neural networks is their insensitivity toward inconsistency. In other words, if the number of inconsistent data is a few, they are ignored and ineffective in the learning process. If all the data are driven randomly and inconsistently, the final result of the neural network is largely different from what the user expects. Toward this purpose, we generate a pair of training and testing rule sets by assigning random target scores to the training and testing rules. It is observed that, although the training process bears the similar trends but with much more rounds, the testing results, i.e, the quality of the related neural network, are far away from what are specified for the testing rules. The results are illustrated in Table 4.17. This confirms our requirement that the user be consistent on creating training and testing rule sets and their requirements be stable for a certain time before next changes.

Adapting to varying requirements. [25, 27]. A measure selection should be able to adapt itself to the user's dynamic requirements. To pursue this requirement, the expiry date of the neural network, which happens upon the change in the requirements, should be considered. Thus, the training process should repeat. Accordingly, a new training set is fed to the neural network and the parameters are subsequently adjusted, and the model is finally rebuilt.

Invariant to the number of association rules. [25, 43]. Interestingness measures should not be affected by the number of association rules. The parameters involved in our proposed approach are invariant to the number of association rules. Once the neural network model is built using diverse samples of association rules, it works for various numbers of data.

Fast training time and individual response time. The training time is measured for our neural networks. Typically, it takes 0.001 seconds to finish one round. For example, if our approach is used for offline learning, the training time for an average number of 300 rounds will be 0.3, which is fast. Online processing time, triggered by the user for the interestingness score of a single association rule, is also observed to be 0.0001 seconds. Thus, the user is able to use our approach for online learning, as discussed in Section 4.2.

4.5 Summary

This chapter proposes a novel approach to dynamically evaluate association rules according to a collective effect of multiple measures, while taking the user's feedbacks into consideration. It utilizes multi-layered feed-forward neural networks along with back-propagation learning algorithm, which learns the relative importance of measures for evaluating rules in an iterative process.

Since neural networks differ in their structures, including the number of layers, the number of input/output/hidden neurons, the direction of the flow for the computation, etc., they are selected depending on the application domain. In our simulations, a number of networks with different structures have been studied, on two real-life datasets retrieved from

the UCI repository, namely car evaluation data set and the solar flare data set. The results are compared and, finally, one network is chosen and further evaluated for its structure and performance.

For both data sets, the selected network performs well on taking the user's requirements into account while avoiding inconsistencies in their ideas given the supervised back-propagation learning algorithm. Our approach also has the ability to adapt itself to user's varying requirements.

Chapter 5

Conclusion and Future Work

Data mining aims at discovering interesting patterns from large-volume data. Association mining, one of the data mining tasks, discovers the inherent relationships among data objects in an application domain. One of the non-trivial problems in association mining is to deal with the number of association rules discovered, which is usually too large to be easily interpreted and comprehended. While many of the association rules convey non-trivial and interesting information, some of them are trivial or even irrelevant. It is desirable to distinguish the interesting ones from those uninteresting. Toward these problems, evaluating association rules as a post-process has become an integral part of an association mining task.

A large variety of interestingness measures to evaluate association rules have been proposed and studied. They evaluate association rules according to a set of criteria, such as conciseness, coverage, reliability, etc. Some of these criteria, such as novelty and surprisingness, are dependent on each other, while some others, such as peculiarity and coverage, are contradictory. Given these facts, an association rule may, for instance, be considered the best based on one measure but the worst based on the other.

In the literature, a set of interestingness properties representing domain-dependent requirements have been proposed to select an appropriate measure for an association mining task. Since each application domain has its own characteristics and accordingly its own requirements, making such a selection usually will cause incomplete and contradictory specifications of properties. In many situations, it is even harder to represent the requirements by interestingness properties. Therefore, assessing measures for an association mining task is an involved process.

Furthermore, a user's requirements may be changing with time. These changes should

be reflected in the future measure selection process. Also, as discussed before, two measures may totally differ in evaluating an association rule. Rather than using one measure for an association mining task, it would be desirable to consider multiple measures together. So far, to our knowledge, there exist no approaches that could handle these situations.

5.1 Our approaches

In our work, we propose two approaches to tackle the situations aforementioned. Our first proposed approach, an AHP-based measure assessment method (AHPMA), takes a user's requirements into account and represents them as pair-wise interestingness property comparisons. It prioritizes them in assessing interestingness measures. This method performs very well, as supported by case studies, and is invariant to the number of association rules. However, it is not able to adapt itself to the user's dynamic requirements.

Our second approach, a neural-network-based evaluation approach, is an approach that dynamically evaluates association rules according to a composite and collective effect of multiple measures, while involving the user's feedbacks into consideration. It makes use of a multi-layer neural network along with the back-propagation learning ability. To evaluate the effectiveness of our method, a set of empirical studies have been conducted. Our approach outperforms other selection methods in many aspects. It adapts itself to the user's dynamic requirements, avoids any inconsistencies in the user's domain-dependent requirements, and captures any requirement changes instantly in its use.

5.2 Future work

For our future work, we definitely will do more case studies for our AHPMA. We plan to use it for some real-life applications. In addition, for our neural-network-based evaluation approach, we plan to apply it to more application domains, attempting to obtain domain-

dependent neural network structures and the related parameters, such as learning rates, the number of hidden layers, etc.

Besides, we are also considering the following possible extensions of our work. Following the same spirit of our AHPMA, we could apply the AHP technique directly to the set of interestingness measures, including both objective and subjective measures, rather than their properties. The user could conduct an interestingness measure pair-wise comparison to select a measure that well suits her/his requirements.

In order to reduce the processing time of an association rule evaluation process, we are considering whether we could reuse some similar idea in our neural-network-based approach. Rather than applying each measure to each association rule and inputting it into the selected neural network, we could create a new neural network that outputs the most appropriate interestingness measure for a given association rule mining task. To do so, a new neural network that connects interestingness properties and interestingness measures could be built. In this network, interestingness properties are used in the input layer, while the interestingness measures are in the output layer. Weights attached to the connections between measures and properties represent how strong their fulfillments are. The input values of the interestingness properties to this network would be obtained from an AHP process, as shown in our AHPMA. As a result, the most important interestingness measure would gain the highest score, which will be output for further association rule evaluations.

Bibliography

[1] M. Adya and F. Collopy. How effective are neural networks at forecasting and prediction? a review and evaluation. *Journal of Forecasting*, 17:481–495, 1998.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the International Conference on Management of Data*, pages 207–216, New York, NY, USA, 1993. ACM.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, The United Kingdom, first edition, 1999.

[5] M. J. A. Berry and G. S. Linoff. *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management*. John Wiley and Sons, Inc., USA, second edition, 2004.

[6] M. Bohanec and B. Zupan. UCI machine learning repository: Car evaluation data set, http://archive.ics.uci.edu/ml/datasets/car+evaluation. 1997.

[7] G. Bradshaw. UCI machine learning repository: Solar flare data set, http://archive.ics.uci.edu/ml/datasets/solar+flare. 1989.

[8] J. E. Dayhoff. *Neural Network Architectures an Introduction*. Van Nostrand Reinhold, Australia, 1990.

[9] J. P. Marques de Sá. *Pattern Recognition, Concepts, Methods, and Applications*. Springer Berlin Heidelberg, Germany, 2001.

[10] A. A. Freitas. On rule interestingness measures. *Elsevier Science*, (3):309–315, 1999.

[11] L. Geng and H. J. Hamilton. Choosing the right lens: finding what is interesting in data mining. In *Quality Measures in Data Mining*, volume 43, pages 3–24. Springer Berlin Heidelberg, 2007.

[12] P. Giudici. *Applied Data Mining Statistical Methods for Business and Industry*. John Wiley and Sons, Inc., England, first edition, 2003.

[13] M. H. Hammond, C. J. Riedel, S. L. Rose-Pehrsson, and F. W. Williams. Training set optimization methods for a probabilistic neural network. *Chemometrics and Intelligent Laboratory Systems*, 71(1):73–78, 2004.

[14] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, second edition, 2006.

[15] S. Haykin. *Neural Networks A Comprehensive Foundation*. Prentice-Hall, Inc., The United States of America, second edition, 1999.

[16] R. Hilderman and H. Hamilton. Knowledge discovery and interestingness measures: A survey. Technical report, 1999.

[17] R. J. Hilderman and H. J. Hamilton. Evaluation of interestingness measures for ranking discovered knowledge. In *Lecture Notes in Computer Science*, pages 247–259. Springer Berlin Heidelberg, 2001.

[18] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *SIGKDD Exploration Newsletter*, 2(1):58–64, 2000.

[19] S. Huang, K. K. Tan, and K. Z. Tang. *Neural Network Control: Theory and Applications*. Research Study Press Ltd., Great Britain, 2004.

[20] R. J. Bayardo Jr. and R. Agrawal. Mining the most interesting rules. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 145–154. ACM, 1999.

[21] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley and Sons, Inc., New York, USA, second edition, 2003.

[22] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *The third international conference on Information and knowledge management*, pages 401–407, New York, NY, USA, 1994. ACM.

[23] B. Krose and P. Smagt. *An Introduction to Neural Network*. The University of Amsterdam, Amsterdam, eighth edition, 1996.

[24] P. Lenca, P. Meyer, B. Vaillant, and S. Lallich. On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European Journal of Operational Research*, 184(2):610–626, 2008.

[25] P. Lenca, B. Vaillant, P. Meyer, and S. Lallich. *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, chapter Association Rule Interestingness Measures: Experimental and Theoretical Studies, pages 51–76. Springer Berlin Heidelberg, 2007.

[26] B. Liao. An improved algorithm of apriori. *Computational intelligence and intelligent systems: fourth international symposium, ISICA*, pages 427–432, 2009.

[27] B. Liu, W. Hsu, S. Chen, and Y. Ma. Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems and their Applications*, 15(5):47–55, 2000.

[28] Z. Lu, A. I. Rughani, B. I. Tranmer, and J. Bongard. Informative sampling for large unbalanced data sets. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2047–2054, New York, NY, USA, 2008. ACM.

[29] J. MacLennan, Z. Tang, and B. Crivat. *Data Mining with SQL Server 2008*. John Wiley and Sons, Inc., Indiana, 2007.

[30] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[31] G. J. Myatt. *Making Sense of Data, A Practical Guide to Exploratory Data Analysis and Data Mining*. John Wiley and Sons, Inc., 2007.

[32] D. L. Olson and D. Delen. *Advanced Data Mining Techniques*. Springer Berlin Heidelberg, first edition, 2008.

[33] B. Padmanabhan and A. Tuzhilin. Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems*, 27(3):303–318, 1999.

[34] J. W. Ra. Chainwise paired comparisons. *Decision Sciences*, 30:581–599, 1999.

[35] T. L. Saaty. *The analytic hierarchy process: Planning, priority setting, resource allocation*. McGraw-Hill International Book Co., New York and London, 1980.

[36] S. Sahar. Interestingness via what is not interesting. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 332–336, New York, NY, USA, 1999. ACM.

[37] S. Samarasinghe. *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*. Taylor and Francis Group, LLC., USA, 2007.

[38] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, 1996.

[39] S. Sumathi and S. N. Sivanandam. *Introduction to Data Mining and its Applications (Studies in Computational Intelligence)*, volume 29. Springer-Verlag New York, Inc., Secaucus, NJ, USA, second edition, 2006.

[40] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41, Edmonton, Alberta, Canada, 2002. ACM.

[41] P. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Information System*, 29(4):293–313, 2004.

[42] J. V. Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231, 1996.

[43] B. Vaillant, P. Lenca, and S. Lallich. A clustering of interestingness measures. In *Lecture Notes in Computer Science*, pages 290–297. Springer Berlin Heidelberg, 2004.

[44] P. Wallin, J. Froberg, and J. Axelssonand. Making decisions in integration of automotive software and electronics: A method based on atam and ahp. In *SEAS '07: Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*, page 5, Washington, DC, USA, 2007. IEEE Computer Society.

[45] H. Wu, Z. Lu, L. Pan, R. Xu, and W. Jiang. An improved apriori-based algorithm for association rules mining. In *FSKD'09: Proceedings of the 6th international conference on Fuzzy systems and knowledge discovery*, pages 51–55, Piscataway, NJ, USA, 2009. IEEE Press.

[46] D. Xin, X. Shen, Q. Mei, and J. Han. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 773–778, 2006.

[47] K. Yamasaki and H. Ogawa. Methods for choosing a training set which prevents overlearning. In *Systems and Computers in Japan*, volume 25, pages 88–98. Wiley amd Sons, Inc., 1994.

[48] Y. Zhang, L. Zhang, G. Nie, and Y. Shi. A survey of interestingness measures for association rules. *Business Intelligence and Financial Engineering, International Conference on*, 0:460–463, 2009.

[49] Y. Zhao, C. Zhang, and S. Zhang. *Discovering Interesting Association Rules by Clustering*. Springer Berlin Heidelberg, Secaucus, NJ, USA, 2004.

Appendix-A

This table completes Table 2.2. Objective interestingness measures for association rules [11, 41]:

Interestingness Measures	Formula
Support	$P(AB)$
Confidence/Precision	$P(B\|A)$
Coverage	$P(A)$
Prevalence	$P(B)$
Recall	$P(A\|B)$
Specificity	$(\bar{B}\|\bar{A})$
Accuracy	$P(AB)+P(\bar{A}\bar{B})$
Lift	$P(B\|A)/P(B)$ or $P(AB)/P(A)P(B)$
Leverage	$P(B\|A)-P(A)P(B)$
Added Value	$P(B\|A)-P(B)$
Relative Risk	$P(B\|A)/P(B\|\bar{A})$
Jaccard	$P(AB)/(P(A)+P(B)-P(AB))$
Certainty Factor	$(P(B\|A)-P(B))/(1-P(B))$
Odds Ratio	$\frac{P(AB)P(\bar{A}\bar{B})}{P(A\bar{B})P(\bar{B}A)}$
Yule's Q	$\frac{P(AB)P(\bar{A}\bar{B})-P(A\bar{B})P(\bar{A}B)}{P(AB)P(\bar{A}\bar{B})+P(A\bar{B})P(\bar{A}B)}$
Yule's Y	$\frac{\sqrt{P(AB)P(\bar{A}\bar{B})}-\sqrt{P(A\bar{B})P(\bar{A}B)}}{\sqrt{P(AB)P(\bar{A}\bar{B})}+\sqrt{P(A\bar{B})P(\bar{A}B)}}$
Klosgen	$\sqrt{P(AB)}(P(B\|A)-P(B))$
Conviction	$\frac{P(A)P(\bar{B})}{P(A\bar{B})}$
Collective Strength	$\frac{P(AB)+P(\bar{B}\|\bar{A})}{P(A)P(B)+P(\bar{A})P(\bar{B})}\cdot\frac{1-P(A)P(B)-P(\bar{A})P(\bar{B})}{1-P(AB)-P(\bar{B}\|\bar{A})}$
Laplace Correction	$\frac{N(AB)+1}{N(A)+2}$
Gini Index	$P(A)(P(B\|A)^2+P(\bar{B}\|A)^2)+P(\bar{A})(P(B\|\bar{A})^2$ $+P(\bar{B}\|\bar{A})^2)-P(B)^2-P(\bar{B})^2$
Goodman- and Kuskal	$\frac{\sum_i max_j P(A_iB_j)+\sum_j max_i P(A_iB_j)}{2-max_i P(A_i)-max_i P(B_j)}$ $-\frac{max_i P(A_i)+max_i P(B_j)}{2-max_i P(A_i)-max_i P(B_j)}$

Continued on next page

Table 5.1 – continued from previous page

Interestingness Measures	Formula
Normalized- Mutual Information	$\sum_i \sum_j P(A_i B_j) \log_2 \frac{P(A_i B_j)}{P(A_i)P(B_j)} / (-\sum_i P(A_i) \log_2 P(A_i))$
J-Measure	$P(AB) \log \frac{P(B\|A)}{P(B)} + P(A\bar{B}) \log (\frac{P(\bar{B}\|A)}{P(\bar{B})})$
One-Way Support	$P(B\|A) \log_2 \frac{P(AB)}{P(A)P(B)}$
Two-Way Support	$P(AB) \log_2 \frac{P(AB)}{P(A)P(B)}$
ϕ-Coefficient	$\frac{P(AB) - P(A)P(B)}{\sqrt{P(A)P(B)P(\bar{A})P(\bar{B})}}$
Piatetsky-Shapiro	$P(AB) - P(A)P(B)$
Cosine	$\frac{P(AB)}{\sqrt{P(A)P(B)}}$
Loevinger	$1 - \frac{P(A)P(\bar{B})}{P(A\bar{B})}$
Information Gain	$\log \frac{P(AB)}{P(A)P(B)}$
Sebag-Schoenauer	$\frac{P(AB)}{P(A\bar{B})}$
Least Contradtiction	$\frac{P(AB) - P(A\bar{B})}{P(B)}$
Odd Multiplier	$\frac{P(AB)P(\bar{B})}{P(B)P(A\bar{B})}$
Example and - Counterexample Rate	$1 - \frac{P(A\bar{B})}{P(AB)}$
Zhang	$\frac{P(AB) - P(A)P(B)}{max(P(AB)P(\bar{B}), P(B)P(A\bar{B}))}$

Appendix-B

This table completes Table 2.3. Interestingness properties fulfilled by measures [11, 24, 41, 48]:

Interestingness Measures	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Support	0	1	0	1	0	0	1	1	1	0	0	0
Confidence/Precision	0	1	0	0	1	0	1	1	1	0	0	0
Coverage	0	0	0	0	0	0	0	1	1	0	0	0
Prevalence	0	0	0	0	1	0	1	1	0	0	0	0
Recall	0	1	0	0	1	0	0	1	1	0	0	1
Specificity	0	0	0	0	0	0	0	1	1	0	0	0
Accuracy	0	1	1	1	0	0	1	1	1	0	1	0
Lift/Interest	0	1	1	1	0	0	0	1	1	0	0	0
Leverage	0	1	1	0	0	0	1	1	0	0	0	1
Added Value	1	1	1	0	0	0	1	1	1	0	0	0
Relative Risk	0	1	1	0	0	0	1	1	0	0	0	0
Jaccard	0	1	1	1	0	0	1	1	0	0	0	1
Certainty Factor	1	1	1	0	0	0	0	1	1	0	1	0
Odds ratio	0	1	1	1	1	1	0	1	0	1	1	0
Yule's Q	1	1	1	1	1	1	0	1	1	1	1	0
Yule's Y	1	1	1	1	1	1	0	1	1	1	1	0
Klosgen	1	1	1	0	0	0	0	1	1	0	0	0
Conviction	0	1	0	0	1	0	0	1	1	0	1	0
Collective Strength	0	1	1	1	0	0	0	1	1	1	1	0
Laplace Correction	0	1	0	0	0	0	1	1	0	0	0	0
Gini Index	1	0	0	0	0	0	0	1	0	0	1	0
Goodman and Kruskal	1	0	0	1	0	0	0	1	1	0	1	0
Normalized- Mutual Information	1	1	1	0	0	0	1	1	0	0	1	0
J-Measure	1	0	0	0	1	0	0	1	1	0	0	0
One-Way Support	1	1	1	0	0	0	0	1	1	0	0	1
Two-Way Support	1	1	1	1	0	0	0	1	1	0	0	1
ϕ-Coefficient	1	1	1	1	0	0	0	1	1	1	1	1
Piatetsky-Shapiro	1	1	1	1	0	0	1	1	1	1	1	1
Cosine	0	1	1	1	0	0	0	1	1	0	0	1
Loevinger	1	1	0	0	1	0	0	1	1	0	0	1
Information Gain	1	1	1	1	0	0	0	1	0	0	0	1
Sebag-Schoenauer	0	1	1	0	1	0	0	1	1	0	0	1
Least Contradiction	0	1	1	0	0	0	0	1	1	0	0	1
Odd Multiplier	0	1	1	0	1	0	0	1	1	0	0	0
Example and- Counterexample Rate	0	1	1	0	1	0	0	1	1	0	0	1
Zhang	1	0	0	0	0	0	0	0	0	0	0	0