High-Order Pattern Discovery and Analysis of Discrete-Valued Data Sets

by

Yang Wang

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 1997

©Yang Wang 1997

National Library of Canada
Bibliothèque nationale du Canada

Acquisitions and Bibliographic Services
Acquisitions et services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-22245-4

Canada

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

To

my parents and sister on the other side of this planet

and

my dear wife, Xuelan, who is always with me

Abstract

Automatic pattern discovery from data collections and the analysis of the patterns for useful information are common and important in both science and engineering today. This discovery is especially demanding in challenging industrial and business applications where the explosive volume of data makes manual analysis virtually impossible. The problems of pattern discovery and analysis that this research addresses include: 1) the discovery of polythetic patterns; 2) the discovery of patterns in the presence of noise and uncertainties; 3) schema for representing different order patterns; 4) the inference process for flexible pattern prediction; and 5) the application of pattern discovery to large database analysis and data mining.

In this thesis, the design and development of a system for pattern discovery and analysis of categorical or discrete-valued data is presented. The system starts with detecting the event association patterns of different orders and provides a probabilistic inference mechanism to achieve flexible classification and prediction. Here a pattern is defined as a significant event association in a problem domain. To detect significant event associations, residual analysis in statistics is used. The insights gained from the analysis of the event associations of different orders and the properties of the residuals lead to a general pattern discovery paradigm which detects patterns according to the deviations of the observed patterns from a default model. Along with the paradigm, techniques are developed to avoid exhaustive search in the process of discovering high order patterns from a large data set. An attribute hypergraph is proposed to represent and to operate on the discovered patterns which can be of different orders. The pattern discovery process can be viewed as a hypergraph generation process. The attributed hypergraph acts as a bridge linking the pattern discovery process with the inference process. For

pattern analysis and inference, a generalized reasoning process based on the weight of evidence is introduced. With this paradigm, flexible prediction becomes possible.

This thesis covers also the implementation of the major ideas outlined in the pattern discovery framework in an integrated software system. It ends with discussions on the experimental results of pattern discovery and analysis on data obtained from various sources (including synthetic and real-world data). Compared with the existing systems, the new methodology this thesis presents stands out, possessing significant and superior characteristics in both pattern discovery and pattern analysis.

Acknowledgements

First of all, my heartfelt gratitude goes to my supervisor, Prof. Andrew K. C. Wong, for his continuous enthusiasm, encouragement, and support during the whole course of the program. Not only did he guide my research but also influenced the way I think over the not-too-short four and half years.

I would like to thank my external examiner, Prof. Nick Cercone from the University of Regina and other members of my defence committee, Prof. Jiahua Chen of Statistics and Actuarial Sciences, Prof. D. Stashuk and Prof. F. Karray of Systems Design Engineering, for their helpful comments on my thesis.

Hay, my PAMI friends, it is your turn. Some of you have already graduated, but your friendship will be with me for long. Without your friendship and discussions, the four and half years would sure have felt much much longer. A special thank-you goes to Stephen Bay, who proof-read my thesis when he was busy writing his own. Thanks also go to Tom Chau, Li Rong, Dr. Fenton Ho and Florence Kong.

I would also like to thank my parents in China. Without their love and encouragement, I would not have had the opportunity to pursue a research career in Canada. Above all, thanks go to my dear wife and best friend, Xuelan. Her endless love and invaluable support made all these happen.

Contents

1 Introduction **1**

1.1 The Problem of Pattern Discovery and Analysis 2

1.2 Motivation and Objectives 5

1.2.1 Research Motivation 5

1.2.2 Thesis Objectives 9

1.3 Research Outline 9

1.3.1 Pattern Discovery and Representation 10

1.3.2 Pattern Analysis and Inference 11

1.4 Organization of the Thesis 13

2 Review of Related Works **15**

2.1 An Overview of Pattern Discovery and Analysis 15

2.2 Algorithms for Pattern Discovery and Analysis 19

2.2.1 Tree-Based Algorithms 19

2.2.2 Rule-Based Algorithms 22

2.2.3 Probabilistic Graphical Approaches 25

2.2.4 Other Pattern Discovery Approaches 28

2.3 Dependence Tree, Event-Covering and PIT 32

2.4 Summary 35

3 Pattern Discovery and Representation **37**

3.1 Terminology and Definitions 38

3.2 Pattern as Significant Event Association 40

3.3 Pattern Representation Using Attributed Hypergraphs (AHG) . . . 43

3.3.1 Other Representation Schemes 43

3.3.2 Representing Patterns in AHG 45

3.3.3 Summary of the AHG Representation 49

3.4 Different Order Patterns in a Data Set 53

3.5 Residual Analysis for Significant Associations 56

3.5.1 Detection of Deviation 56

3.5.2 Properties of Residuals 59

3.5.3 Parameter Estimation 63

3.6 Algorithm for Pattern Discovery 69

3.6.1 Overview of the Algorithm 70

3.6.2 Selection of Informative Candidates 73

3.6.3 Analysis of Complexity 84

4 Pattern Analysis and Inference **89**

4.1 The Best-N Problem and the Missing-Value Problem 90

4.1.1 The Best-N Problem 90

4.1.2 The Missing-Value Problem 92

4.2 Weight of Evidence 94

4.3 Finding the Best N Associations 96

4.4 Classifying a New Object 101

5 Experiments and System Performance **112**

5.1 System Implementation 113

5.2 Experiments on Pattern Discovery 116

5.2.1 XOR Problem 117

5.2.2 Synthetic Data Sets 122

5.2.3 Zoo Database 129

5.2.4 Injury Database 132

5.3 Experiments on Classification 138

5.3.1 Zoo Database 139

5.3.2 Wisconsin Breast-Cancer Database 142

5.3.3 Mushroom Database 144

5.3.4 The Monk's Problems 147

5.4 Summary 150

6 Conclusion and Future Research **154**

6.1 Summary of Contributions 155

6.2 Suggested Future Research 159

A Hierarchical Models **162**

Bibliography **171**

List of Tables

1.1 Example Instances in an Animal Database 3

5.1 Data Sets Selected for Testing Various Pattern Discovery Goals . . 117

5.2 Second-Order Pattern Detection from XOR Data Set : Unsupervised 119

5.3 Third-Order Pattern Detection from XOR Data Set : Unsupervised 120

5.4 Second-Order Pattern Detection from XOR Data Set : Supervised . 121

5.5 Patterns Detected from XOR Data Set : Supervised 121

5.6 Data sets and Execution Time 124

5.7 Candidates Generated with A6-D5-I10K 126

5.8 Attributes of Zoo Database 130

5.9 Fields in the Injury Database 133

5.10 Order, Candidates, Associations and Execution Time 134

5.11 Data Sets Selected for Testing Various Pattern Analysis Goals . . . 139

5.12 Classification Accuracy on Attribute *Type* of Zoo Data 141

5.13 Attributes of Breast-Cancer Database 142

5.14 Classification Accuracy of the Wisconsin Breast-Cancer Domain . . 144

5.15 Attributes of Mushroom Database 145

5.16 Classification Results for Mushroom Data 146

5.17 Robot Attributes in the Monk's Problem 148

5.18 Associations Detected from the First MONK's Problem 149

5.19 Classification Results for the Monk's Problems 150

List of Figures

3.1 A Hypergraph with Eight Vertices and Six Hyperedges 46

3.2 AHG Representation of XOR Patterns 49

3.3 Associations in XOR Data Set 53

3.4 Associations in Database 54

3.5 Different Orders of Significant Event Associations 55

3.6 A Third Order Compound Event 58

3.7 Candidate Selection in Pattern Discovery 84

5.1 Overview of the Prototype System 114

5.2 Data Structure of Attributed Hypergraph 115

5.3 Data Structure of a Node in the Linked List 115

5.4 *Class 1* of XOR 122

5.5 Computing Time vs Number of Instances 124

5.6 Computing Time vs Number of Attributes 125

5.7 Multi-valued Data Sets for Testing 126

5.8 Moving Patterns in the Data Set 128

5.9 Part of the AHG for Zoo Database 131

5.10 Association Examples Found in the Injury Database 135

5.11 Relative Numbers of Possible Combinations, Selected Candidates and Significant Associations: (a) Order 2 (b) Order 3 136

5.12 Patterns with Similar Frequencies But Different Levels of Confidence 137

5.13 Distribution of Associations vs Adjusted Residual 137

5.14 Part of the Significant Associations from Breast-Cancer Database . 143

Chapter 1

Introduction

The ability to automatically discover inherent patterns from data and to infer or reason with the acquired knowledge is central to human intelligence. Since artificial intelligence (AI) is concerned with the science of making machines behave intelligently, the implantation of pattern discovery and inference capabilities in machines has always been one of its ultimate goals. It has been an enticing topic of machine learning since the fast development of artificial intelligence in the mid-70's.

Research in pattern discovery and analysis was brought forward by statisticians and data analysts long before the emergence of AI. Most of their methods, however, are rather *ad hoc* and manual [90]. They also require extensive domain knowledge to obtain useful models from data. Furthermore, the patterns or models extracted from the data are normally very complicated and hence difficult for humans to interpret [30]. Later research in AI and machine learning attempts to discover the regularities or patterns inherent in the data automatically. Besides, the discovered patterns should be represented in a way that humans could easily understand [66]. Today, the combination of statistical techniques and AI methodologies has drawn

increasingly attention from researchers in both areas, especially when dealing with data containing noise and uncertainties. However, finding a good framework for effective pattern discovery and analysis is still a major endeavor of many researchers in AI and other areas today.

1.1 The Problem of Pattern Discovery and Analysis

This thesis deals with the problem of computer-based pattern discovery and analysis. The proposed system is capable of discovering the statistically significant patterns inherent in a given domain described by a set of data, forming concepts and organizing them along with their relations in a graphical representation, and providing a mechanism for flexible reasoning and inference. By patterns, we mean the regularities inherent in a given data collection but previously unknown to the outside world. By discovery, we mean that the existence of an external entity such as a teacher is not needed to guide the pattern detection process. By inference or pattern analysis, we mean an answer can be issued corresponding to an incoming query such that certain properties of the original data collection are preserved.

For example, we have an animal database, part of which is illustrated by Table 1.1. In such a database, patterns are the relationships among different characteristics of the animals. *Milk* = *yes* and *Type* = *mammal* are a pattern since they are correlated. Some patterns depict only pairwise relationships, while more complicated patterns involve more factors. Pattern [*Legs* = *four*, *Milk* = *no* and *Type* = *reptile*] has three propositions and it is a third order pattern. The ability of a system to discover high order patterns is important for the analysis of complex real-world

problems. Pattern discovery is a process of detecting the regularities from a given database. The discovered patterns are expected to give a better understanding of the problem domain. Later in the analysis and inference stage, these patterns will support decision making tasks such as classification. With the animal database, we expect the patterns can be used to determine the values of unobserved features of an animal which is not in the original database. We can predict the type of the animal or the number of its legs without re-learning the whole database.

Table 1.1: Example Instances in an Animal Database

Hair	*Feathers*	*Eggs*	*Milk*	···	*Legs*	*Tails*	*Type*
yes	no	no	yes	···	4	no	mammal
yes	no	no	yes	···	4	yes	mammal
no	no	yes	no	···	0	yes	fish
no	yes	yes	no	···	2	yes	bird
no	no	yes	no	···	0	no	invertebrate
yes	no	no	yes	···	4	yes	mammal
no	no	yes	no	···	4	no	amphibian
no	no	yes	no	···	6	no	insect
no	no	yes	no	···	4	yes	reptile
no	yes	yes	no	···	2	yes	bird
yes	no	no	yes	···	4	yes	mammal
no	no	yes	no	···	0	yes	fish
no	no	yes	no	···	0	yes	reptile
···	···	···	···	···	···	···	···

The above description partially accomplishes the definition of pattern discovery or data mining given by Fayyad, et al [30]. In their framework, *Data* is a set of facts F. *Pattern* is an expression E in a language L describing facts in a subset F_E of F. E is called a pattern if it is simpler than the enumeration of all facts in F_E. In addition to this, it is also maintained that E is consistent to a certain degree

with F_E though previously unknown. Hence, *Pattern discovery* is a process that, under some acceptable computational efficiency limitations, produces a particular enumeration $\mathbf{E}$ of patterns E_j over F. A *pattern analysis* or *inference* is a process that, provided with a set of patterns $\mathbf{E}$ discovered from F, generates an answer A to a given query Q such that A is consistent with F according to a measurement function $m()$. Note that the definitions here are by no means complete. Detailed definitions and discussions are later furnished in the corresponding chapters.

Particularly, this research focuses on the following aspects of pattern discovery and analysis:

1. discovering patterns from missing, noisy and incomplete data;
2. detecting high order patterns[1] without exhaustive search;
3. representation scheme for different order patterns and their relationships;
4. transparency of the discovery and representation, and
5. evaluating the discovered patterns and providing different kinds of decision support activities such as classification and rule induction.

Further, the following idealizations and assumptions have been imposed to narrow down the problem.

1. All the attributes (variables, features) describing the data assume categorical or discrete values.

[1]High order patterns are those patterns regarding the relationships among more than two factors (e.g. attribute-value pairs). For example, the association among propositions *Legs = four*, *Milk = no* and *Type = reptile* is a third order patterns. Detailed definition is given in Chapter 3.

2. The number of instances in the data set is fixed and does not change during the pattern discovery process.

3. The number of attributes describing the data set is finite. The domain of each attribute is also finite.

4. In prediction, only the value of one attribute is predicted each time. This assumption follows the formalism of a classification problem. The combinations of more than one attribute are not under consideration.

It is not assumed that the data set is noise-free, complete or correct, nor that the background knowledge of the domain and a discovering guide are available.

1.2 Motivation and Objectives

1.2.1 Research Motivation

Data is an extremely valuable asset,
but like a cash crop,
unless harvested, it is wasted.

– Sid Adelman

The importance of discovery in AI has been repeatedly emphasized by a number of researchers and will not be repeated here. Either by being-told or by self-discovery, pattern discovery is a process whose target is to gain the principles of behavior of the working domain in order to reason, infer and predict the behaviors in the same domain.

The original motivation of this research comes from the following practical problems. Consider a company which has a large database. A telecommunication company, for instant, might have logged hundreds of thousands of trouble reports. A financial service company might have a database of past loan applications and the credit histories of their customers. Suppose that these companies wish to develop a system for prediction, diagnosis, simulation and training purposes. Could they use their existing databases to automatically derive the patterns for diagnosis or prediction, the regularities that reflect the operations and performance of the companies, or the relationships among market demands and decision criteria? More specifically, for the financial service company, what advice should the system give when a new problem occurs or when a customer applies for a new loan? If the focus of interest is only on the cost of reparation or the risk of granting a loan, in what form should the above questions be formulated?

With the advent of inexpensive electronic and magnetic storage media and the ever broadening use of computers in a vast spectrum of business, such databases become quite commonplace. A 1994 MetaGroup survey revealed that by 1997 about 90% of Fortune 1000 companies will be pursuing data warehousing projects. The size of the databases will range from gigabytes to terabytes. Analyzing these databases and providing the users with useful knowledge is very challenging and difficult in the meantime. The huge volume of data virtually makes manual analysis impossible. The real-world characteristics of these databases such as noise, incompleteness, inconsistency, and redundancy are open questions posed to today's machine learning research. These demands and concerns create both a need and an opportunity for automatically extracting knowledge from databases. It is quite clear that if a company has an existing database of its business records, such a pattern discovery and analysis system would be very useful. Recently, data mining

has been ranked as one of the most promising topics by researchers in the areas of both database and machine learning [35] [87] [30].

Academically, this research targets some open problems in machine learning and automatic knowledge acquisition from large databases. These challenging problems motivate the research and form the objectives of the research. The following list gives a more detailed description to each of the research problems stated in Section 1.1.

1. *Discovering patterns from missing, noisy and incomplete data*
   Many existing learning systems were developed under the assumption of noise-free environments [62] [16]. Therefore they are deterministic rather than probabilistic [26] [16]. Such assumptions may hold only in certain domains such as logic based problems and game playing but rarely hold when dealing with real-world problems. Although this concern has been noted by many researchers, more research is still on the way to a general and integrated system.

2. *Discovery of complex relations among variables and variable values in data*
   Due to the complexity of real-world problems, the ability of the discovery algorithm to acquire *polythetic* rather than only *monothetic* patterns [34] is crucial. Existing approaches are either monothetic such as ID3, or polythetic but search-intensive, such as CN2. New versions of some monothetic approaches, such as C4 of ID3, use various pruning technologies to achieve polythetic pattern learning capability, but they introduce extra computational burdens. Some Bayesian approaches, such as the work of Pearl [77], are polythetic, but they are applicable only under certain restrictive conditions [90] and have difficulties in effective representation and inference. Polythetic assessments of

feature combinations (or higher order relationship detection) are imperative for robust learning. In the meantime, avoiding exhaustive searches is crucial to the feasibility of detecting polythetic patterns.

3. *Understandability of discovered patterns*

   This problem is two-fold. First, the discovered patterns should be interpretable by humans. Second, the discovering process should be transparent. They are related to one of the natural requirements of learning, *transparency* versus *blackbox* [39] [104]. With a transparent system, it is much easier to construct a meaningful explanation of the patterns and their relationships. Understandability is one of the most important aspects of a knowledge discovery and data mining system.

4. *Representation of complex patterns*

   Knowledge representation is always a central research topic in AI and knowledge discovery. Most of the existing schemes have various shortcomings in representing patterns in a flexible and generic manner. There is a need for a representation scheme that directly encodes polythetic patterns, supports probabilistic inference, and is easy to visualize.

5. *The ability to predict values of more than one attribute*

   As indicated by Fisher [34], *flexible prediction* is one of the goals of a learning/discovery system and is more general than classification. Traditional machine learning approaches pay more attention to classification where only the class label is predictable. A great deal of problems are worth studying for a system capable of predicting the value of more than one attribute of interest. In addition to flexible prediction, other forms of analysis, such as rule induction, should also be supported in the same system.

1.2.2 Thesis Objectives

The following are the objectives of this study:

- An integrated system which is able to detect inherent patterns from a given data set and make inferences based upon the discovered patterns;
- The ability to discover polythetic patterns from real-world data in the presence of noise and uncertainties;
- A suitable pattern representation language for patterns of different orders and for probabilistic inference;
- Flexible prediction; and
- Experimental demonstration and evaluation for analyzing the performance of the proposed method.

1.3 Research Outline

The research presented in this thesis can be subdivided into two sections. The first section focuses on the discovery of patterns from a data collection. The second section describes the inference process based on the discovered patterns.

In the first section, a statistical residual analysis based algorithm is proposed to detect significant event association patterns in the presence of uncertainties. This section also deals with a suitable representation of the discovered patterns. Attributed hypergraphs are used to represent patterns of different orders. In the second section, two general inference problems, namely the best-N problem and the

missing-value problem. are presented. Weight of evidence. an information measure for significant patterns. is engaged in solving the two problems.

1.3.1 Pattern Discovery and Representation

The first portion of the research attempts to formulate pattern discovery as a process of detecting significant event associations in the data set.

A pattern is defined as a statistically significant association among two or more events in a problem domain. The significance is determined by the difference between the actual occurrence in the data set and the expected occurrence according to a default model or *a priori* knowledge. The process of detecting inherent patterns involves statistically testing the significance of a pattern candidate. The statistical test is based on the residual analysis in which the adjusted residual of a pattern candidate is evaluated to determine if the candidate is significant. To give a theoretical background of this approach. the properties of residuals are studied. This method is believed to be robust in the presence of noise because of its statistical characteristics.

Because the patterns are considered as probabilistic rather than deterministic. high order patterns cannot be synthesized from the low order ones, which forces one to test every candidate. To avoid exhaustive search which is combinatorial. techniques are developed to truncate the search space by considering the validation of statistical tests and different situations of event associations. The reduction of search space not only speeds up the search process but also allows the algorithm to scale up to deal with large database.

To represent the detected patterns. an attributed hypergraph (AHG) representation scheme is defined. In the proposed pattern representation. each vertex in

the AHG is an event (an attribute-value pair) and each hyperedge depicts a significant relation (or association) among the events. The AHG representation is a simple structure that is general enough to encode information at many levels of abstraction. Its simplicity is desirable for quantifying the information content of the structure when an information theoretical approach is adopted for inference purposes. This thesis will give the definition of AHG representation and show its basic operations. The advantages of AHG representation are discussed along with the pattern discovery process.

1.3.2 Pattern Analysis and Inference

The task of inference is to quantitatively measure the evidence provided by interesting patterns in response to the query posed by an external agent (or user). Here it is formulated to tackle two problems: the best-N problem and the missing-value problem. The best-N problem deals with finding the most informative N rules that best describe the problem domain. The missing value problem attempts to predict the most plausible value of a missing attribute in an observation.

To solve these two problems, the weight of evidence which measures the amount of evidence provided by a set of events in support of, or against, an attribute taking on a particular value, will be defined. With the proposed weight of evidence, the best-N problem can be readily solved by recursively generating production rules and keeping the best-N rules based on their weights of evidence. The missing-value problem is more difficult, since in the observation, not all observed events are relevant to a value of the predicting attribute (the value of which will be determined). To deal with this problem, the weight of evidence measure is extended to higher order. First, the entire observation is decomposed into several disjoint subsets of

events. If the subset of events are associated with a value of the predicting attribute, their weights of evidence are calculated and summed up. Otherwise, it is concluded that the subsets are irrelevant to the predicting attribute. The value of the predicting attribute which gains the greatest positive weight of evidence from the same observation will be the most plausible value and thus the missing value problem is solved. More details regarding the calculation of weight of evidence of high order events and certain special cases of interpreting weight of evidence in the classification stages will be discussed later in the thesis.

Detailed algorithm pseudocodes which guided the implementation of the prototype system for experimentation and performance analysis will be presented.

The characteristics of the proposed system can be summarized as follows:

1. Different from building comprehensive model from data, this method detects patterns as associations deviated from a default model;
2. Patterns are detected directly from an incomplete and noisy data set by a statistical method;
3. High order patterns in addition to pairwise relationships are discovered without relying on exhaustive search;
4. New attributed hypergraph representation of patterns provides a transparent scheme for pattern analysis and understanding;
5. Two common problems, the best-N and the missing-value problems, are solved under the same framework through the information measure, weight of evidence; and
6. Flexible prediction is achieved so that the value of any attribute can be determined without re-learn the whole database.

1.4 Organization of the Thesis

There are six chapters in this thesis including this introduction.

To give a better understanding of the research field, a brief review of existing ideas relevant to pattern discovery and analysis is presented in Chapter 2. Discussions of individual approaches follow a general overview of pattern discovery and analysis. The advantages and the disadvantages of these methods are also examined with regard to the goals of this research.

Chapter 3 and Chapter 4 embody the major part of this research. In Chapter 3, a pattern discovery technique based on residual analysis is described in detail. It explains how this approach is able to handle noise and discover statistically significant event association patterns of different orders. An attributed hypergraph representing different order event associations is also introduced. Discussion regarding the algorithms developed for discovering patterns without relying on an exhaustive search is followed with supportive demonstrations.

To analyze the patterns discovered, an inference mechanism based on the weight of evidence of significant event associations is discussed in Chapter 4. The two categories of the inference problem are discussed: the best-N problem and the missing-value problem. Algorithms to solve the two problems are presented in detail. In the same chapter, the method to achieve flexible prediction is explained.

In Chapter 5, the proposed system is tested with sets of real-world and synthetic data. The experiments fall into two groups. In the first group, the performance of the pattern discovery algorithm is tested and evaluated. How well the system can handle large data sets is investigated. In the second group, four common data sets are used to test and evaluate the classification performance of the proposed inference methods. The results are then compared with some other well-known

classification algorithms.

Chapter 6 highlights the contributions of this study and suggests the direction of future research in this area.

Chapter 2

Review of Related Works

2.1 An Overview of Pattern Discovery and Analysis

Data analysis and pattern recognition have long been recognized as significant research areas by statisticians and more recently by researchers in artificial intelligence (AI). Pattern discovery and analysis came as an extension of data analysis and pattern recognition. It later became a part of the activities in the broader disciplines of machine learning and intelligent systems. In general, statisticians work on building models from data to characterize the system behavior, while AI researchers, at the same time, try to understand the system better by describing the discovered regularities in a way that humans can easily interpret. Numerous research papers and reports are now available. However, a comprehensive comparison of those methods is difficult since different methods address different goals and are based on different implicit assumptions. Still, there are a few broad dimensions along which one can categorize them from a more general viewpoint.

In the ordinary sense, "discovering regularities" from a system, or a data set, simply implies partitioning the instances observed from the system or data set into classes according to the similarity of those instances [66]. In a more formal language, it is the finding of clusters of instances in the instance space. Thus, it is not surprising to find that pattern discovering closely resembles statistical clustering and model fitting. In their paper, Michalski and Stepp [66] pointed out that the traditional distance-based statistical clustering techniques make no distinction between attributes that are more relevant and those that are less relevant or irrelevant. They do not render conceptual description of the clusters nor take into account how humans would describe a pattern. However, the AI approaches attempt to represent the detected patterns in terms of definitions, rules (e.g. the conjunction of attribute-value pairs) [66] or other languages (e.g. decision trees) [79] that may have natural interpretation for humans. These representations can be transformed into another that can support goal achievement [92] such as classifying a new object or predicting the missing value of an attribute.

With the demand from expert system applications for automatic knowledge acquisition, machine learning researchers in AI try to teach the machine to extract useful knowledge automatically from data. Unfortunately, traditional *ad hoc* and manual data analysis techniques cannot easily break the processing bottleneck to meet the challenges of the large amount of data and the fast growing demand of knowledge. Machine learning, be it referred to as conceptual clustering [66], object classification [79] or rule induction [89], is to find the relations among the attributes and/or among their values. The importance of learning in AI has been repeatedly and alternatively emphasized by a number of researchers [64] [88] [80] [14]. There are several forms of learning, ranging from *learning-by-being-told* to *learning-by-discovery* [11]. In the former type of learning, which is often referred

to as *supervised learning*, the learner is told explicitly what is to be learned. In learning-by-discovery, which is often referred to as *unsupervised learning*, the learner spontaneously discovers new concepts from unstructured observations and performs experiments in the environment of the problem domain. There is no teacher's environment that plays the role of an oracle. Traditionally, machine learning research pays more attention to supervised learning (also named as inductive learning and learning from examples) or the *classification* problem. The patterns that interest supervised learning are those related to a special class tag assigned by an external teacher. The desired performance of such a learning paradigm is apparent, i.e. to improve the prediction of class membership. For unsupervised learning, e.g. concept clustering [66] [32] [19] and rule induction [89], the performance is more difficult to measure. Fisher [31] observes that *flexible prediction*, or the ability of predicting any attribute's value, is an important aspect of unsupervised discovery. Thornton [92] points out from another viewpoint that, in a fully unsupervised setting, improvements in the behavior of the learner do not involve the evaluation of the actual outputs. Instead, the desired behavior is achieved via a direct, algorithmic process which usually can be described in terms of an objective function. As a general pattern discovery and analysis system, data is simply collected and stored without specific *a priori* knowledge of its class relationship [40]. In this sense, unsupervised learning is more useful than supervised strategies when there is no explicit classification information [103]. However, a pattern discovery and analysis system should be able to perform classification tasks when asked. A good system should be able to automatically discover patterns and reason with them.

The techniques of pattern discovery, or machine learning in specific, can roughly be subdivided into two distinct categories, namely the *symbolic* approaches and the *statistically-oriented* approaches [44]. The better known symbolic techniques in-

clude Mitchell's version space algorithm [67] and its later evolution, and the AQ family of algorithms of Michalski [63] including the concept clustering algorithm CLUSTER/2 [66]. Symbolic approaches always assume that the learning environment is deterministic [52] [78] [68] [16]. Hence, their application areas are rather restrictive and noise is not easily handled. More recently, to deal with imperfect data collections and real-world problems, statistically-oriented approaches gained more attentions. Some typical works include Breiman's CART [12], Quinlan's ID3 [79] and its varieties such as C4.5 [81] and $\mathcal{CDP}$ [2], Fisher's COBWEB [32], ITRULE [89] by Smyth and Goodman, and the Bayesian approaches [53]. With these algorithms, various statistical measures or hypothesis tests are applied to detect patterns and/or rules. Statistical approaches have been successfully applied to real-world situations. Even in noise-free game-playing problems, the performance of statistical approaches are comparable with that of the symbolic approaches.

In the last decade, the explosive growth in our abilities to generate and collect data provides us with huge amount of information. Such volumes of data not only overwhelm the traditional manual methods of data analysis, but also prevent the machine learning algorithms from being directly applied to such domains. Knowledge discovery from database (KDD) [35] or data mining [110] hence becomes a challenging topic for researchers in machine learning, statistics and data analysis [30]. Although KDD can be considered as a process from data selection to pattern interpretation/evaluation [30], pattern discovery and analysis is the core of the process. A number of statistical and machine learning methods have been adopted and integrated into a data mining system. Unlike traditional machine learning methods which are largely classification oriented, pattern discovery and analysis in KDD are more general and sensitive to computational complexity due to the large amount of data. Although pioneering works can be found in the literature [2] [3] [13] [50] [18]

[103] [108], it is a difficult task to analyze the algorithms as a whole since the goals of the systems are quite diverse. An interesting summary can be found in [30].

2.2 Algorithms for Pattern Discovery and Analysis

In this section, a subset of popular techniques in pattern discovery and analysis are reviewed and discussed. This review is brief and cursory, but it yields insight into the current status of pattern discovery and analysis research. The discussion will be focused on KDD and data mining.

2.2.1 Tree-Based Algorithms

Trees are perhaps the most popular vehicle in decision support tasks such as classification and concept generation. In classification tasks, trees generated from the training data for classifying (new) instances are sometimes referred to as *decision trees*. The basic idea of tree-based algorithms is to break up a complex decision into a union of several simpler decisions which can be organized into a tree or a forest, with the hope that the final solution obtained this way would resemble the desired solution. This group of algorithms are popularized with the development of ID3 [79] and CART [12].

The ID3 algorithm constructs a decision tree using a top-down divide-and conquer approach [79]. It is a simple and effective AI method for learning from examples and can be thought of as a good example of a hybrid statistical and symbolic technique [90]. At each node of the decision tree, the training objects are partitioned according to their values along a single attribute. An information theoretical measure

is used to select the attribute whose values would improve the prediction of class membership bringing it above the accuracy expected from a random guess. The training set is recursively decomposed in this manner until no remaining attribute could improve the prediction in a statistically significant manner as indicated by a user-defined parameter. The original version of ID3 [78] was designed to include all the positive training instances and to exclude all the negative ones. Since this may result in the problem of *over-fitting* with noisy data, many of the later ID3-based algorithms such as ASSISTANT [10], C4 [82] and C4.5 [81], adopt different strategies, such as *pre-pruning* [10] [15] and *post-pruning* [80] [73] to simplify the decision tree while retaining satisfactory accuracy on the training data. However, decision trees that use univariate splits restrict their applications to a small portion of the functional models [30]. Complicated patterns such as the XOR problem are difficult to discover when the single-attribute-split strategy is used.

Unlike ID3, CART [12] allows the tree to be split at a node by considering several attributes in a linear regression manner. This approach enables CART to detect more complicated patterns. In CART, a large tree with all the leaves pure or nearly pure is first generated without any stopping criterion. Then the tree is pruned by a multi-pass top-down algorithm. The disadvantage of CART is that it is very computationally expensive as it requires the generation of multiple auxiliary (sub)trees. Besides, CART selects the final pruned subtree from a parametric family of pruned trees, and this parametric family may not even include the optimal pruned subtree [84].

There are a good number of other tree-based classification algorithms such as [38] [43] [94]. A thorough and comprehensive survey of decision trees can be found in [84]. With respect to the pruning techniques, Chan concluded in [16] that pre-pruning suffers from the problem of using local information in determining whether

or not the learning process should be terminated and as a result, the "omission of more important global information" cannot be avoided. Pre-pruning in general is also rather slow. As for post-pruning strategies, many of them are quite slow and inefficient, and for some, a second training set is required. These shortcomings make it difficult to apply them to large sets of real-world data. Recent research efforts in KDD and data mining try to adopt decision tree based algorithms for large database analysis. Examples can be found in [1] [2].

From the literature, all the decision tree algorithms are intended for classification, and from a more general viewpoint of pattern discovery and analysis, they can only make predictions on a single attribute (the class). Flexible prediction is hard to achieve without a multiple learning process.

In unsupervised learning, trees also play an important role. Michalski and Stepp proposed CLUSTER/2 [66] as a *conceptual clustering* algorithm. Given a set of objects and a parameter K to specify the number of desired clusters, CLUSTER/2 constructs a tree-like structure which optimally partitions the set into K groups according to a pre-defined quality criterion known as LEF. With this representation, each node of the tree is a cluster at the leaf level. Thus, the resulting clusters are mutually disjoint and cover all of the objects. For each cluster, there is a description called a *logical complex* which is a logical product of one or more attribute-value pairs. CLUSTER/2 is an effective algorithm for analyzing a small set of objects containing no noise. However, for a large data set, it is computationally expensive even with its Hierarchy-Building Module. Another drawback of such a deterministic system is that it would not be able to give correct clustering results in the presence of noise.

To deal with noise, COBWEB was introduced by Fisher [32]. Different from its contemporaries, COBWEB is an incremental learning method. It evaluates classi-

fication schemes by how well they facilitate the prediction of unknown properties of new observations incrementally. Its basic idea is to incorporate a new object into the class that "best" matches the object according to *category utility*. By examining objects one by one and tentatively placing each in a class, it outputs a concept tree which encodes the relations between concepts. The attribute-value distribution of each class is tentatively updated. From the updated probabilities, the category utility of the class is computed. The class that maximizes category utility after adding the new object is chosen as the class for that object and distributions are updated. COBWEB is an "optimistic" learning strategy in that rules are exploited until evidence confirms that they are idiosyncratic. Optimistic learning has advantages in incremental learning [34] [33]. COBWEB is polythetic and is able to learn from data with noise [31]. Because of the nature of optimistic learning, the concept tree generated by COBWEB might be very large and post-pruning techniques have to be applied. For deterministic pattern discovery such as the MONK's problems, COBWEB does not work well compared with the other AI and connectionist approaches [93].

2.2.2 Rule-Based Algorithms

Rule-based approaches have been long studied with the development of expert systems. Smyth and Goodman [89] argue that (if-then) rules provide a much more flexible representation than the tree structures, especially from the viewpoint of expert systems. Furthermore, individual rules are more understandable for human beings than trees particularly when a tree grows too large. A (production) rule has the form of: **if** [*conditions*] **then** [*consequence*] **with** [*measure*]. Originally, rules are discovered to classify objects into classes at the consequence side. In some approaches, the consequences of the rules may deal with more than one attribute.

In such a case, the rules describe the characteristics of a domain rather than classify a new object.

AQ, with its later related extensions [64] [65] and CN2 [25] [26] are two typical rule-based classification algorithms. For example, AQ15 constructs classification rules in the form of disjunctive complexes by searching through a space of logical expressions to determine those that account for all positive examples and no negative ones. Then, a rule truncation technique is employed to reduce the complexity of these rules by removing the complexes that explain comparatively fewer training instances. To determine the class membership of a new instance, its description is matched against the truncated rules by a flexible rule matching technique. A disadvantage of AQ15 is that it requires users to introduce and manipulate tunable parameters in order to guide and control the search process [25]. Also, with AQ15, features that appear more often in the training instances may be mistaken as being more important. This may result in the ignorance of patterns that are statistically significant though they rarely occur [26]. Some of the more recent developments [9] of the classical AQ algorithms have incorporated constructive induction routines that can dynamically generate new attributes. AQ based algorithms perform very well with problems generated by deterministic and game-playing domains. However they are rather computationally expensive. Furthermore, their ability of handling large and noisy data is a problem of concern.

To avoid some of the above observed problems in AQ15, CN2 [25] [26] and similar GREEDY3 [75] were developed. CN2 is an algorithm that adopts a general-to-specific search strategy to generate rules in the form of disjunctive complexes. Unlike AQ15, instead of performing rule truncation after the construction of classification rules that are perfectly consistent with the training data, it employs a probabilistic measure to guide the process of specialization. A complex is added to

a logical expression that predicts a particular class only when it is consistent with a large number of examples of that class and few of others. The rule specialization process may terminate before all the training instances are correctly classified if, according to the likelihood ratio test, any further specialization is not supported by a significant number of training examples. CN2 is believed to be able to handle noisy data better than AQ bases algorithms. CN2 is also much better at learning XOR relations than the post-pruning versions of ID3 [33] because it introduces higher-order (or polythetic) relations. But CN2 and GREEDY3 are very search intensive. With complicated problems, these approaches require very long, sometimes impractical, learning time. Smyth [89] pointed out that the likelihood ratio constraint used by CN2 to restrict the algorithm's potentially combinatorially large search is an *ad hoc* one. Actually, to use the same likelihood ratio to test the goodness of different order rules is questionable, especially when the number of possible combinations is large. Neither of the AQ and the CN2 rule measures include an *a priori* probability term [89]. Incorporation of *a priori* probability is a necessary component of a rule dealing with a probabilistic domain.

It is worth noting that both AQ and CN2 algorithms detect patterns considering more than one attribute-value at a time. This approach avoids the shortcoming encountered in single-attribute-split tree approaches. Hence, more complex patterns can be discovered. Thus, it is plausible for real-world applications.

Instead of detecting rules with only class labels as consequences, Smyth and Goodman [44] [89] developed a system called ITRULE to solve the more general problem, namely *rule induction*. ITRULE recursively selects one attribute to be the right-hand result of a candidate rule and then searches the combinations of the propositions of the other attributes as the conditions at the left-hand side. A measure of information content, called *J-measure*, is defined to evaluate the

performance of the candidate rule. The m (pre-defined) rule candidates with the largest J-measures are then selected as the result. Hence, the problem ITRULE tries to solve is also referred to as the *Best-N* problem – to find the best-N rules in describing the problem domain. Because the rules are based on J-measures which are always positive and cannot be summed, it is hard to quantitatively evaluate the evidence of a set of observations during the inference process, especially for the classification of an object [103]. ITRULE is expected to achieve flexible prediction, but the rules generated restrict the prediction to a few right-hand side events when m is not large enough. If a particular value of an attribute which is not at the right-hand side of the rules is to be predicted, ITRULE will not be able to give an answer.

Unlike ITRULE, Agrawal, et al [2] [3] tried to find all the *association rules* from a database. The main concern of their method is to ensure its applicability to large data sets. Their methods use a simple user-defined confidence, which is basically the probability of an event, to determine if an association rule holds. They do not consider negative associations or missing items. In [3], they synthesize higher order associations with the lower order ones. In a probabilistic environment, the assumption of high order patterns being able to be synthesized from lower order patterns is not always true [102] [103].

2.2.3 Probabilistic Graphical Approaches

Graphical models specify the probabilistic dependencies which underlie a particular model using a graph structure [77]. In its simplest form, the model (pattern) specifies which variables are directly dependent (or independent) on each other. Most of these approaches are based on the Bayesian inference rules and are depicted

as probabilistic networks such as Markov networks or Bayesian networks [77] [53].

Because of its solid theoretical basis, Bayesian inference has always played an important role in reasoning in the presence of uncertainties. These methods provide a formalism for reasoning about partial beliefs under conditions of uncertainty. Once a probabilistic network is built, one can derive the probability of an event conditioned by a set of observations to compare that event with others for a classification. The basic limitation of the early research in Bayesian inference is manual estimation [90]: where multiple variables interact in the network, a large number of parameters have to be estimated in a subjective way. To overcome this shortcoming, a number of algorithms have been developed to automatically construct probabilistic networks from data.

CONSTRUCTOR [36] is proposed by Fung and Crawford to automatically induce a probabilistic model from data. More specifically, CONSTRUCTOR induces discrete Markov networks of arbitrary topology from data. These networks contain a quantitative (i.e., probabilistic) characterization, and a qualitative (i.e., structural) description, of the data. The basic idea of CONSTRUCTOR is to find the Markov boundary of each node (attribute, variable or factor) in the networks which will "shield" the node from being affected by other nodes outside the boundary. The independence between the current node and the nodes outside the boundary has to be tested. High dimensional contingency tables are used to determine whether or not an attribute (random variable) is independent from a set of attributes. To simplify this search-intensive work, CONSTRUCTOR considers only those distributions which are "composable" [36]. This simplification allows CONSTRUCTOR to avoid checking for high order dependency among the nodes. Even though CONSTRUCTOR is reported to work well when tested with training sets generated from probabilistic models and with real data in an information retrieval application [37],

it has several disadvantages. First, Markov networks have their shortcomings – not all probabilistic dependencies can be captured by undirected graphs [77]. Actually, neither undirected nor directed graphs are capable of representing all kinds of dependencies among variables. Second, when going to high-order, contingency tables introduce too much computational burden. The construction of high-dimensional contingency tables is expensive in both computation and memory. Furthermore, CONSTRUCTOR is a variable-oriented method. It is worth pointing out that systems dealing with event-based dependencies such as the rule-based and the tree-based algorithms are more efficient than variable based algorithms [22] [98] [16] [90]. From the inference point of view, Markov networks have to attach a matrix of joint probabilities to each edge of the network, otherwise, the original data will be used to estimate the joint probabilities for probabilistic inference. If the domains of the variables are large, the matrix of joint probabilities will be large. This makes the networks difficult to handle. On the other hand, since not all the joint events of two variables are significant, it is not necessary to store the information regarding these events, not only because of more required storage space but also more computational complexity.

The representative power of Markov networks is limited: they cannot represent induced and non-transitive dependencies [77]. To overcome this deficiency, Bayesian networks use the richer language of directed graphs, where the directions of the edges permit us to distinguish genuine dependencies from spurious dependencies induced by hypothetical observations. Heckerman [53] gives a good introduction to Bayesian networks. Individual works can be found in [77] [91] [27] [54]. Bayesian networks model the problem domain into directed graphs. In the inference process, conditional probabilities can be computed according to the structure of the graph and the partial conditional probabilities associated with the edges. It is be-

lieved that Bayesian networks are superior in handling noisy data and incorporating domain knowledge. Another advantage of the probabilistic (Markov or Bayesian) network approaches is that the inference can be made with respect to any variable in the network by deriving the conditional probability. However, not all dependencies that are representable by a Markov network can also be represented by a Bayesian network. Pearl [77] concluded that, in both undirected and directed graphs, separation between two sets of vertices is defined in terms of pairwise separation between their corresponding individual elements. According to the theory of probability, however, independence among elements does not imply independence among sets. It implies that any representation relying on graphs (including trees) is incapable of depicting all the dependencies induced by probabilistic models. Bayesian networks are also variable-oriented approaches, and therefore they have the same problems as the other variable-oriented methods.

2.2.4 Other Pattern Discovery Approaches

A considerable number of other methods for pattern discovery and analysis cannot be categorized into any of the previously mentioned leagues. Some of them are briefly discussed here.

One of the popular family of algorithms in pattern recognition and machine learning are neural networks, and more generally the algorithms from linear and nonlinear regression of basis functions (e.g. sigmoids, splines and polynomials) to combinations of the input variables. Typical examples are feed-forward neural networks [55] for classification and competitive neural networks for unsupervised learning [59]. Hitherto, intensive research has been conducted on the neural computing models, including training algorithms, network structures and coding techniques.

Recent research also deals with transferring a network structure into a set of production rules [6]. In terms of model evaluation, while networks of the appropriate size can universally approximate any smooth function to any desired degree of accuracy, relatively little is known about the representation properties of fixed size networks estimated from finite data sets [30]. Many of the functions used to train the neural networks can be found in regression and classification research done by statisticians [85]. For example, backpropagation [83] is a parameter search method which performs gradient descent in the parameter space to find a local maximum of the likelihood function starting from random initial conditions. Neural networks have shown strong potential in fields such as optimal control, pattern recognition and learning non-linear functions. However, for pattern discovery and analysis from categorical data, special coding techniques are needed to feed the categorical data into a network. Nonlinear regression methods, though powerful in representation, can be very difficult for humans to interpret [30]. One of the main goals of data mining and pattern/knowledge discovery in general is to ultimately understand the problem domain [30]. Because of that. Agrawal, et al [1] argue that neural networks are not quite suitable for data mining problems. Although research attempting to transfer neural network structures into rules is in progress [6] [72], promising results are still yet to come.

With the demand of KDD and data mining tools for large databases, a number of algorithms have been developed to cope with large numbers of data samples in the pattern discovery process. Some of the approaches adopt algorithms found in the machine learning literature. For example, $\mathcal{CDP}$ [2] modifies ID3 with an adaptive precision threshold to evaluate a path in the tree. However, unlike most machine learning algorithms which are classification oriented, pattern discovery from databases is sometimes concerned with extracting best rules to describe the

domain. One of the pioneer works is proposed by Cai, Cercone and Han in [13] [49] [50] as the DBLearn system. Given a set of concept hierarchies for the attributes in a database, DBLearn tries to generalize the large database into a much smaller number of tuples that better describe the domain. The tuples can be transformed into simple logical formula. The generalization is controlled by a user defined threshold of tuple numbers. The count of generalized tuples is kept to offer a statistical characteristics of the final concept. The idea is quite simple but effective for large relational databases. More recent development includes applying rough set theory to further generalize the concepts [57]. DBLearn relies heavily on concept hierarchies to provide new knowledge based on the existing data. If the hierarchies are short and simple, the data generalize very quickly, but few new and interesting discoveries will be made [48]. The patterns discovered by DBLearn are not directly applicable to the classification of new instances. For this purpose, DBLearn can be viewed as a pre-processing stage which performs a preliminary generalization using domain knowledge [51].

Similar to ITRULE [89], but not limiting the number of rules to be discovered, Apriori/AprioriTid algorithms are proposed by Agrawal, et al [3] to find "all" association rules from a database. They use two simple conditional probabilities $p(Y|X)$ (confidence) and $p(X.Y)$ (support) to test the significance of a rule $X \Rightarrow Y$. In their approach, negative or missing items are not considered as items of interest. One distinctive characteristic is that their rules allow a consequence to have more than one item. (Y may contain more than one event.) The basic idea of their approach is to detect all "large itemsets", which are the combinations of events that have support above a pre-defined minimum threshold, then transfer those item sets into the rule format. The search of large item sets is recursively done by adding a new item to the existing item set at one time. The initial sets are a seed set of large

itemsets, called candidate itemsets. The basic property used in their approach is that any subset of a large itemset must be large. Based on this assumption, the candidate itemsets having k items can be generated by joining large itemsets having $k-1$ items, and deleting those that contain any subset that is not large. This treatment results in a much smaller number of candidate itemsets, hence the search space is pruned rapidly. However, from the statistical significance point of view, there is no direct relation between higher order significance and lower order significance [102] [103]. Therefore, the subsets of a high order significant itemset may not be significant at all. Using their approach, some of the significant itemsets may not be discovered. The rules are also not directly applicable to classification tasks.

To deal with uncertainties of the data and the patterns, the theory of rough sets [76] is applied by some researchers to data analysis [69] [109] [106] [57]. The key idea of rough set approaches arose from the observation that the representations of noisy data help detect regularities inherent in the data sets. The quality of the information in data is measured in terms of rough sets by using lower and upper set approximation. With this representation, the "quality" of a concept or a rule can be determined and computed. The rough set theory has solid mathematical foundations and is strong in quantifying the data dependency relationships among attributes. However, application of rough set directly to large database analysis is not always feasible since the computational complexity is NP-hard [108] [57]. A pre-stage generalization scheme should applied to shrink the size of the original database. The performance of the system will depend also on the choice of this generalization scheme.

Many other systems which have been developed cannot be covered in this brief review. But most if not all of them can be found similar to one of the categories

discussed above. For a good discussion on the KDD algorithms, interested readers can go to [30] for more details.

2.3 Dependence Tree, Event-Covering and PIT

The research presented in this thesis is based on previous work done in the same lab. It is necessary to introduce the historical research so as to give a comprehensive perspective of the achievement and the significance of this study.

The use of *dependence trees* and *discriminant trees* in pattern recognition was first presented by Chow. et al [24] and Wong, et al [100] [101] respectively. By finding a set of $(n-1)$ first order statistical dependence relationships, a dependence tree is constructed to approximate an n-dimensional discrete probability distribution. First a complete graph is constructed with the vertices as the variables and the edge weights as the mutual information of the two connected variables. The dependence tree is a maximum weighted spanning tree of the graph that optimizes the sum of mutual information. From the dependence tree structure, the probability distribution $P(X)$ can be estimated by the product of conditional probabilities between tree nodes. In this way, when applied to empirical observations from an unknown distribution. the approximated distribution by the dependence tree is the maximum likelihood estimate of the true distribution [24] [100]. Since these methods are based on the dependencies between two variables, they suffer from the problem of being "variable-oriented" since they only consider if two variables are dependent but provide no information on which joint events contribute to the variable dependency. To approximate a probability distribution using only first order conditional probabilities is not always suitable for all applications. They cannot handle complicated problems in which only high order relationships are significant.

To overcome the "variable-oriented" problem, it was suggested that "value-to-variable" dependencies be analyzed instead of "variable-to-variable" dependencies so that information on whether or not an event is dependent upon a variable can be obtained. Based on this idea, the *Event-Covering* method was proposed by Chiu and Wong [22] [98]. In order to know if a variable X is dependent on a value y_i of Y, a statistical technique based on the χ^2 test is used. It is correspondent to calculate χ^2 of the i-th row or column in a two-way contingency table for X and Y. If y_i is dependent on X, it is called a *covered event*. After finding the covered event sets of X and Y, information measures can be employed to detect the statistical patterns of these subsets. The information measure used by event-covering is based on mutual information and Shannon's entropy. Event-covering can be applied to classification tasks as well as clustering problems. Theoretically, flexible prediction is achievable using event-covering techniques. However, several theoretical and conceptual problems still exist [16].

Chan [16] summarized the shortcomings of event-covering in five aspects, i.e., the limited distribution of the test statistic, the problems related to value-to-variable dependency, the omission of the identification of some important attribute values, the limitation of the interdependence redundancy measure, and, the limitation of the normalized surprisal measure. Besides these, it was also noticed that event-covering cannot yet go to higher order easily. This problem remains in the *Probabilistic Inference Technology* or *PIT* proposed by Wong and Chan [97] [16] [17] [18].

Probabilistic Inference Technology (PIT) uses two-way contingency tables and residual analysis to detect the dependencies between two values. One of the values is a known class. So PIT is a supervised inductive learning technology. Based on PIT, the *APACS* [17] inductive learning algorithm was developed. To judge if a

value x_i is relevant to a class label c_j, the two-way contingency table of a variable X and the class variable C is constructed. The residual of the cell (x_i, c_j) is calculated. If the absolute value of the residual is greater than a threshold, it is concluded that x_i is relevant to c_j, a rule $x_i \Rightarrow c_j$ is obtained. To determine the class membership of a new instance, only the relevant events in the instance are used. An information measure, known as weight of evidence, is calculated and compared with each class to decide the best classification.

PIT is quite effective for detecting the first-order relationships in data sets. Experiments show that PIT is superior to many other inductive learning algorithms in classification when dealing with noisy domains. The classification accuracy and the execution time of PIT are very satisfactory when compared to well-accepted methods such as ID3, a naive Bayesian classifier, AQ11 and AQ15 [16] [20]. But unfortunately, this method also has its shortcomings and deficiencies. The most important one is that, when higher-order relationships have to be detected, for example in the XOR data, PIT will not give satisfactory results[1] because PIT detects only first-order relations in the training data, and ignores higher-order relationships. Later in the inference process, PIT decomposes high order conditional probabilities into the product of first-order probabilities to calculate the weights of evidence. It is efficient only when first-order relationships are sufficient for describing the nature of the data set. PIT was developed under the assumption that the class information is known. In a more general case, it would be more desirable to see a system which can detect patterns relating to any attributes.

[1]PIT can be extended to solve problems with high order patterns by applying high-way contingency table analysis.

2.4 Summary

Pattern discovery and analysis is a broad research area. The above discussions are mainly from the viewpoint of artificial intelligence and machine learning. Major lessons given to the development of new systems from the preceding review are summarized as the following:

- The ability to handle noisy data is crucial for applications targeting real-world problems. Real-world data are always incomplete, inconsistent and they always contain noise. The assumptions of noise-free, complete or perfectly supervised, are not always realistic.

- Because of the complexity of real-world problems, high order (polythetic [34]) patterns have to be detected in addition to first order (monothetic) patterns. Patterns involving only two variables (values) and tree generation by single attribute splitting are two typical scenarios that ignore high order relationships.

- An event-oriented approach provides more detailed information on the problem domain than variable-oriented methods. In the inference process, they are also more effective.

- High order patterns need a new graphical representation to depict the relationship among events and sets of events. Such a representation scheme should offer a mechanism for easy knowledge re-organization or focus on a certain portion of the knowledge to meet the changing goal. The represented knowledge should be transparent and easy to visualize, understand, and handle. Probabilistic inference should be supported by the representation to cope with noisy domains.

- Flexible prediction is one of the important goals of a general pattern discovery and analysis system.
- To scale up a pattern discovery and analysis system for large databases is not a trivial problem. Expensive computation and excessive memory requirement should be avoided.

Chapter 3

Pattern Discovery and Representation

In order to discover the inherent patterns in a database in the presence of uncertainty, a statistics-based technique is proposed. Evolved from *event-covering* [22] [23] [98] and *APACS/PIT* [16] [17] [18] [97], this technique focuses on high order pattern discovery and analysis. This research can be roughly divided into two parts. In the first part, a discovery engine is developed to find the statistically significant patterns from a given data set; in the second part, an inference mechanism is developed to extract interesting associations and rules, or classify new observations.

The focus of this chapter is on the discovery and representation of significant patterns inherent in a database.

3.1 Terminology and Definitions

Consider that we have M observations or samples in a data set D. Each sample item is described in terms of N attributes or features, which can assume values in a corresponding set of N discrete alphabets. For example, the data set might be described in the form of 10-component binary vectors, where $N = 10$.

These N attributes comprise the attribute set of the training data, i.e., $\mathbf{X} = \{X_1, \cdots, X_N\}$. Each variable, X_i, $1 \leq i \leq N$, can take on values from the corresponding alphabet $\alpha_i = \{\alpha_i^1, \cdots, \alpha_i^{m_i}\}$, $1 \leq i \leq N$, where m_i is the cardinality of the i-th attribute alphabet. Thus, a realization of $\mathbf{X}$ can be denoted as $\mathbf{x}_j = \{x_{1j}, \cdots, x_{Nj}\}$, $1 \leq j < \infty$, where x_{ij} can take on any value from the correspondent alphabet α_i. In this manner, each sample in the data set D is a realization of $\mathbf{X}$. With the above notations, the following terms are defined.

Definition 3.1 *A* primary event *of a random variable X_i ($1 \leq i \leq N$) is a realization of X_i which takes on a value from α_i.*

The p-th ($1 \leq p \leq m_i$) primary event of X_i is denoted as

$$[X_i = \alpha_i^p]$$

or simply x_{ip}. For data set D, there are

$$\nu = \sum_{i=1}^{N} m_i \tag{3.1}$$

different primary events. It is assumed that two primary events, x_{ip} and x_{iq}, of the same variable X_i are mutually exclusive if $p \neq q$.

Let s be a subset of integers $\{1, \cdots, N\}$ containing k elements ($k \leq N$), and $\mathbf{X}^{\mathrm{s}}$ be a subset of $\mathbf{X}$ such that

$$\mathbf{X}^{\mathrm{s}} = \{X_i | i \in \mathrm{s}\}.$$

Then $\mathbf{x}^{\mathrm{s}}$ denotes the realization of $\mathbf{X}^{\mathrm{s}}$.

Definition 3.2 *A* compound event *associated with the variable set* $\mathbf{X}^{\mathrm{s}}$ *is a set of primary events instantiated by a realization* $\mathbf{x}^{\mathrm{s}}$. *The* order *of the compound event is* $|\mathrm{s}|$.

The j-th compound event associated with the variable set $\mathbf{X}^{\mathrm{s}}$ can be represented by

$$\mathbf{x}_j^{\mathrm{s}} = \{X_i = \alpha_i^{p_j} | i \in \mathrm{s},\ p_j \in \{1, \cdots, m_i\}\}$$

where p_j is a map of j to an i-th alphabet. $\mathbf{x}_j^{\mathrm{s}}$ is also called the j-th realization of $\mathbf{X}^{\mathrm{s}}$

A 1-compound event is a primary event. A k-compound event has k primary events of k distinctive attributes. Every instance in the data set is an N-compound event.

Definition 3.3 *A* sub-compound event *of* $\mathbf{x}_j^{\mathrm{s}}$ *is a compound event* $\mathbf{x}_j^{\mathrm{s}'}$ $\forall\ \mathrm{s}' \subset \mathrm{s}$ *and* $s' \neq \phi$.

Here, the XOR problem is used to illustrate both the primary and compound events. Consider that three binary attributes, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$. Each of them can take on two values, T or F. Thus, $[\mathbf{A} = T]$ is a primary event, $[\mathbf{A} = F, \mathbf{C} = T]$ is a 2-compound event, and, $[\mathbf{A} = T, \mathbf{B} = F, \mathbf{C} = T]$ is a 3-compound event. $[\mathbf{A} = T, \mathbf{B} = F]$ is a sub-compound event of $[\mathbf{A} = T, \mathbf{B} = F, \mathbf{C} = T]$.

3.2 Pattern as Significant Event Association

In general, a pattern can be any relation or regularity in the problem domain. In data analysis and data mining, such relations and regularities are described by models or rules which represent the relationship among the attributes and/or attribute values in the data set. Pattern discovery has long been studied by researchers in various fields. Unlike the traditional statistical methods which try to build probabilistic models of random variables, this thesis project attempts to discover the associations among the variable values or events. The reason for developing an event-based system is quite intuitive. In practice, it is desirable to know not only if random variables are related, but also how they are related. For example, if the standard χ^2 test shows two random variables are not independent, one would like to know which events lead to this conclusion. This "subtle" information provides more insights in the problem domain [47]. From the viewpoint of expert systems and artificial intelligence, systems dealing with event-based dependencies are more efficient than those dealing with variable based dependencies [16] [22] [90] [98] since event-based systems require less parameter estimates and memory.

Though a pattern discovery system can be applied to verify if the data set fits the available knowledge[1], this research deals mainly with discovering previously unknown patterns. In other words, the goal is to understand to what extent the present knowledge does not cover. Hence, what is interesting are the events that deviate from what is already known. In this sense, the term "significant" is introduced to refer to the occurrence of an association candidate event that "significantly" deviates from its expectation according to the knowledge that one has or what one could best guess. If the significance is verified from certain (statistical) tests on the

[1]Systems, such as SAS, can accomplish this task easily.

data set, the candidate that passes the tests becomes a significant event association; otherwise, it is not.

Since the associations in a real-world database are probabilistic rather than deterministic, for a certain candidate, some instances in the data set may suggest that an event is significant while others may not. Thus, a statistical significance test should be conducted on the data set with respect to the event. Such a test should have the common form of a statistical hypothesis testing:

1. H_0: Candidate c_i is not significant;
   H_1: Candidate c_i is *significant*;
2. The decision rule has the following form:
   $f(c_i) > t \Rightarrow H_1$

where $f(\cdot)$ can be defined according to the interest and the problem domain. Parameter t is a threshold normally related to a selected confidence level.

A popular way to define $f(\cdot)$ is to assign it as the probability of a candidate. That is, if the probability of an event's occurrence is greater than a user-defined threshold, then this event is significant. While simple, this approach is questionable. Since the marginal probabilities of an event are not likely to be the same, the actual probability of the event does not really reflect its significance in the data set. For example, suppose that the probability of event $[A, B]$ in a given data set is 0.75. Is it significant? If one chooses $f(\cdot)$ as the difference of the probability and the product of the two marginal probabilities (equivalent to the independent model), and if the marginal probabilities for A and B are 0.9 and 0.9 respectively, the event $[A, B]$ is unlikely to be significant when the *threshold* is 0.2. On the other hand, if the probability of $[A, B]$ is 0.2, according to the same criteria, it is significant (the

difference is 0.61 which is larger than 0.2). Hence, to use the absolute probability value of an event to judge its significance is biased. A better way is to compare the actual probability with its expected value. According to what is known or what is assumed, a probability model can be made to obtain the expected value of a certain test event. A statistical test regarding the actual and expected values will justify the significance of the event against what is taken for granted. This approach is more objective than merely using the probabilities from the data set. In view of this, the test design, the default model and the properties of the selected statistic are the main issues and will be discussed later.

Definition 3.4 *Given a probabilistic model $\mathcal{P}()$ on the variables $X_1, \cdots, X_N$, the* expected occurrence *of a compound event $\mathbf{x}_j^s$ is the expected total under the default model $\mathcal{P}()$.*

The expected occurrence of $\mathbf{x}_j^s$ is denoted as $e_{\mathbf{x}_j^s}$ and can be estimated as:

$$e_{\mathbf{x}_j^s} = M \cdot \mathcal{P}(\mathbf{x}_j^s). \tag{3.2}$$

Definition 3.5 *Let* T *be a statistical significance test. If the occurrence of a compound event $\mathbf{x}_j^s$ is significantly different from its expectation, we say that the primary events of $\mathbf{x}_j^s$ have a* statistically significant association according to T *or simply they are* associated.

Definition 3.6 *If a compound event $\mathbf{x}_j^s$ passes the statistical significance test* T, *we say that $\mathbf{x}_j^s$ is a* significant pattern, *or simply a* pattern, *of* order $|\mathbf{s}|$.

In the following discussions, the terms such as pattern, association, and significant event association are used interchangeably.

As an illustration, consider an XOR database containing 1,000 samples. Each primary event of this database occurs 500 times. The number of expected occurrences of the 3-compound event $[\mathbf{A} = T, \mathbf{B} = T, \mathbf{C} = F]$ under the independence model is $0.5 \times 0.5 \times 0.5 \times 1,000 = 125$. The *standardized residual* [45] [46] is used to test the significance of the occurrence of this compound event. Suppose that the actual occurrence of this compound event is 250. The standardized residual is 11.18, larger than 1.96, the value at the 95% significant level. Thus, one concludes that the compound event $[\mathbf{A} = T, \mathbf{B} = T, \mathbf{C} = F]$ is *significantly different* from its expectation or that the three primary events $[\mathbf{A} = T]$, $[\mathbf{B} = T]$ and $[\mathbf{C} = F]$ are *associated*. Therefore, this compound event is a 3rd-order *pattern*. It is also noticed that the primary events $[\mathbf{A} = T]$ and $[\mathbf{C} = F]$ are NOT associated, because the standardized residual of the compound event $[\mathbf{A} = T, \mathbf{C} = F]$ is less than 1.96 (the expected occurrences of this event is 250). Thus $[\mathbf{A} = T, \mathbf{C} = F]$ is not a pattern.

3.3 Pattern Representation Using Attributed Hypergraphs (AHG)

3.3.1 Other Representation Schemes

Over the years, numerous knowledge representation schemes have been reported. The most popular ones are decision trees, networks, production rules and logic.

Decision trees are a simple representation popularized by Quinlan's ID3 and successfully applied to inductive learning. Decision tree based systems are found in a wide range of application domains, mostly in the classification-oriented areas. A disadvantage of the decision tree is its difficulty for humans to interpret, especially

from the viewpoint of expert systems [89] and KDD systems [56]. Also, trees are not designed to deal with missing attribute information [89]. Moreover, since decision trees are mainly designed for classification purposes, they are not suitable for multi-attribute predictions [32].

Trees can be considered as a special case of graphs. Graph representations, such as Bayesian and Markov networks, usually provide more general methods to represent patterns. They directly represent the first order associations between two nodes by links. However, as observed by Pearl [77], graph-based representation, including trees and networks, cannot distinguish between set connectivity and connectivity among their elements. Hence, they are not general enough for representing different order patterns.

Production rules (if-then statements) are another scheme widely used in expert systems and classification oriented tasks. It explicitly presents the association between a set of observations (left-hand conditions) and one attribute value (right-hand consequent). Rules are considered easier to understand than trees. However, in KDD applications, with changing interest, the values of many different attributes have to be predicted. As well, a huge number of rules have to be obtained. This is sometimes impractical in the real world [103]. In this case, a scheme which can easily re-organize the represented knowledge for different goals of the system is needed.

In addition to attribute (proposition) based representations, relational representations such as Horn clause (see [60] for an overview) and First Order Logic (see [70] for an overview) are used in the learning systems. They are very powerful and expressive formalisms. Since they were originally designed to formalize mathematical reasoning and later used in logic programming, patterns in them are deterministic rather than probabilistic. To do probabilistic reasoning, special adap-

tations have to be made. This problem also exists in the structured representations such as semantic networks. Besides, logic based representations are considered less comprehensible and harder to visualize than graph based representations.

3.3.2 Representing Patterns in AHG

Let us first give a formal definition of hypergraph.

Definition 3.7 (Berge 1989 [7]) *Let $Y = \{y_1, y_2, \cdots, y_n\}$ be a finite set ($n < \infty$). A* hypergraph *on Y is a family $H = (E_1, E_2, \cdots, E_m)$ ($m < \infty$) of subsets of Y such that*

1. $E_i \neq \phi \quad (i = 1, 2, \cdots, m)$, *and*
2. $\bigcup_{i=1}^{m} E_i = Y$.

The elements $y_1, y_2, \cdots, y_n$ of Y are called vertices, *and the sets $E_1, E_2, \cdots, E_m$ are the* edges *of the hypergraph, or simply,* hyperedges.

Definition 3.8 *A* simple hypergraph *is a hypergraph H with hyperedges $(E_1, E_2, \cdots, E_m)$ such that*

$$E_i = E_j \Rightarrow i = j.$$

Unless otherwise indicated, *hypergraph* here is referred to as *simple hypergraph.*

Fig. 3.1 [7] shows a hypergraph with 8 vertices and 6 hyperedges.

The *order* of a hypergraph H, denoted by $n(H)$, is the number of vertices. The *number of edges* will be denoted by $m(H)$. Further, the *rank* of H is the maximum number of vertices in a hyperedge and $r(H) = \max_j |E_j|$; the *anti-rank* is the minimum number of vertices in a hyperedge and $s(H) = \min_j |E_j|$.

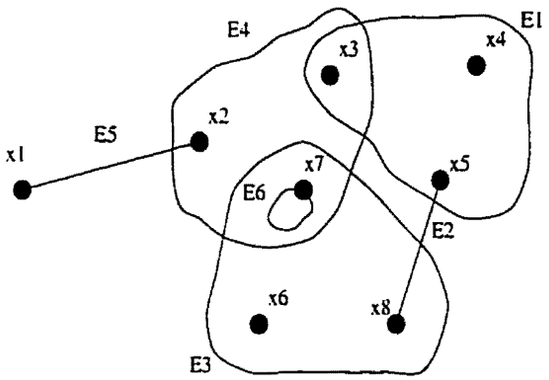

Figure 3.1: A Hypergraph with Eight Vertices and Six Hyperedges

For a set $J \subset \{1, 2, \cdots, m\}$, the family

$$H' = (E_j | j \in J)$$

is called the *partial hypergraph generated by the set J*. The set of vertices of H' is a non-empty subset of X.

For a set $A \subset X$, the family

$$H_A = (E_j \bigcap A | 1 \leq j \leq m, E_j \bigcap A \neq \phi)$$

is called the *sub-hypergraph induced by the set A*.

Definition 3.9 *An* attribute *of a hypergraph is a data structure associated with a hyperedge or a vertex.*

Definition 3.10 *An* attributed hypergraph *or* AHG *is a hypergraph such that each of its hyperedges and vertices has an attribute.*

In *AHG* representation [95], each *vertex* represents a primary event. Each pattern or statistically significant association is represented by a *hyperedge*. The *rank*

(*anti-rank*) of a hypergraph is the highest (lowest) order of the patterns detected from the database. For an event e, the *star* $H(e)$ of hypergraph H with center e represents all the patterns related to the event e. Let A be a subset of all primary events, the *sub-hypergraph* of hypergraph H induced by A represents the event associations in A. The following list gives some hypergraph terminologies and their corresponding meanings in pattern representation:

- Each vertex of a hypergraph is a primary event of a data set;
- Each hyperedge represents a pattern (significant event association) in the data set;
- The order of a hypergraph is the number of primary events appearing in the data set;
- The rank of a hypergraph is the highest order of the patterns detected from the data set; similarly, the anti-rank is the lowest order of detected patterns;
- For a primary event x_{ij}, the *star* $H(x_{ij})$ of hypergraph H with center x_{ij} represents all the patterns related to the primary event x_{ij}.
- Let A be a subset of all primary events, the sub-hypergraph of hypergraph H induced by A represents the associations among the primary events in A.

The attributes of both the vertices and the hyperedges depend on the application and the pattern discovery algorithm applied. Necessary information for the later inference process should be included. For the system discussed in the following chapters, the attribute of each vertex is the marginal probability of the corresponding primary event. The attribute of each hyperedge contains the probability of the compound event, the expected probability of the compound event, and the

probabilities of sub-compound events one order lower. All of these attributes will be useful for the inference process. Therefore, hyperedges depict the qualitative relations among their elementary vertices, while the attributes associated with the hyperedges and the vertices quantify these relations [103] [95].

Within the AHG framework, to manipulate patterns is to operate on the hyperedges, vertices and their attributes. To re-organize knowledge is to select sub-hypergraphs according to the current system goal. If a new instance is to be classified against a field X_1, only the hyperedges containing an event of X_1 are interesting. If the system is later asked to find the patterns related to event $X_2 = True$, only the hyperedges containing this event are focused on. Because there are a good number of mature algorithms on graphs, these operations are expected to be computationally efficient. As indicated by Agrawal, Imielinski and Swami [2], most database mining problems can be classified into three categories: association, classification, and sequence. In the AHG framework, associations among events are represented as hyperedges. When class labels are considered as a special field, classification can always be treated as using patterns related to this special field to predict the class membership of a new object. The sequential problem is just a special case of association with a time tag attached.

Operations on an AHG are application dependent. Basic operators include *Construct()* which constructs an attributed hypergraph from a database, *HighestOrder()* and *LowestOrder()* which find the highest (lowest) order of detected relationships, *FindRelation()* which extracts all the patterns related to a specified event, and *FindSubEvent()* which extracts all the patterns that contain a given composite or its non-empty sub-composites. The last operation finds all the compound events which are considered relevant to the inference process from a set of facts.

Fig. 3.2 shows the AHG representation of the patterns in the XOR problem. In total, there are six vertices and four hyperedges. Each hyperedge represents a pattern. The attributes of both the vertices and the hyperedges are shown in brackets. Hyperedges qualitatively represent the associations among events while the attributes describe the numerical properties of these association patterns. The significance level of each hyperedge is calculated by its observed and expected occurrences. Notice that the AHG in Fig. 3.2 shows that there are only third order patterns in the XOR problem.

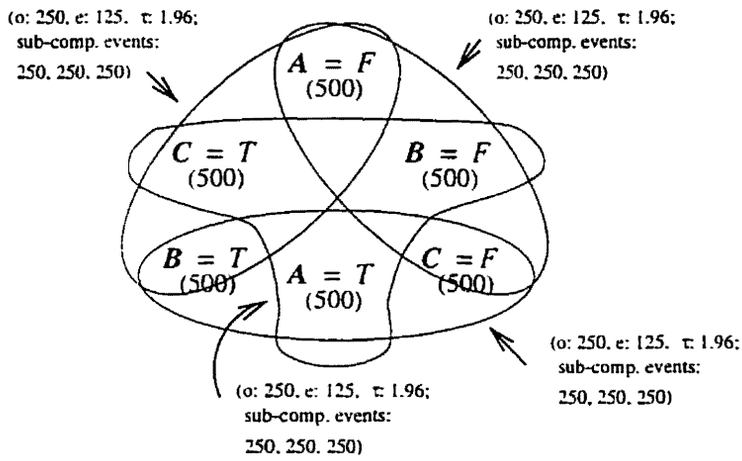

Figure 3.2: AHG Representation of XOR Patterns

Using the AHG representation, the pattern discovery process can be seen as an AHG generating process.

3.3.3 Summary of the AHG Representation

The attributed hypergraph is chosen as the proposed representation scheme for the following reasons.

First, because high order patterns which induce more than 2 events are the major

concern, a framework which is capable of representing the relationships among multiple events has to be used. Another reason is that, in probability inference and many other AI techniques, network representations are extensively used. Networks are graphs which can be considered a special case of a hypergraph. Networks explicitly show the relationships between two nodes. It is very difficult for networks to represent the relationship among 3 events, any two of which are not related. To illustrate the problem, let's consider an experiment [77] with two coins and a bell that rings whenever the outcomes of the two coins are the same. If one ignores the bell, the coin outcomes, say C_1 and C_2, are mutually independent, but if the bell (B) is noticed, then learning the outcome of one coin should change the opinion about the other coin, which means C_1 and C_2 are no longer independent. How then can one represent, using a graph (or networks), the simple dependencies between the coins and the bell, or between any two causes leading to a common consequence? If the naive approach is taken and links are assigned to (B, C_1) and (B, C_2), leaving C_1 and C_2 unlinked, the graph C_1-B-C_2 is obtained. This graph suggests C_1 and C_2 are independent given B. If a link is added between C_1 and C_2, the graph turns into a complete graph which no longer reflects the obvious fact that the two coins are genuinely independent.

In practice, these kinds of dependencies exist everywhere. Over the years, directed acyclic graphs have been introduced to represent such dependencies. Although the directed acyclic graph representation is more flexible than the undirected graph representation, and it captures a larger set of probabilistic independencies, there are still some important shortcomings. First, not all the dependencies that are representable by the undirected graph can also be represented by the directed acyclic graph. Second, computational and representational complexities would rise compared to undirected graph representations. Third, the directed acyclic graph

cannot represent the type of dependencies induced by probabilistic models [77]. Pearl [77] concludes:

> *.....no graphical representation can distinguish connectivity between set from connectivity among their elements. In other words, in both directed and undirected graphs, separation between two sets of vertices is defined in terms of pairwise separation between their corresponding individual elements. In probability theory, on the other hand, independence of elements does not imply independence of sets. .....*

In the proposed attributed hypergraph representation, however, higher order relations are not induced by lower order relations. This representation does not depend on pairwise links. The hyperedges are sets that show the associations among their elements which can also be sets. Yet, the basic element of the proposed hypergraph representation is not a variable but a primary event. That is, dependencies occur among events and not variables. In the bell-coin experiment, if the bell can make 3 kinds of sound, only the first kind of sound, say beep twice, indicates whether or not the outcomes of the two coins are the same. Other signals have nothing to do with the coins. (Perhaps they indicate situations of other events.) It is the event $[B = beep\ twice]$, not B, that relates the outcomes of the coins. In hypergraph representation, the hyperedges of $[B = beep\ twice, C_1 = head, C_2 = head]$ and $[B = beep\ twice, C_1 = tail, C_2 = tail]$ show the relationships among them.

Different sizes of the hyperedges reflect different levels of generalization. The larger the number of vertices in a hyperedge, the more details a concept (pattern) contains. The hyperedges of smaller sizes often represent more generalized concepts (or patterns). One advantage of hypergraph representation is that it allows easy

movement among different levels of generality, which cannot be done (or only with great difficulty) by graph or network representations.

The procedure of constructing an attributed hypergraph is totally "transparent" to the world. The patterns of various orders along with the intermediate states of this procedure are returned. Hence, the learning is "transparent" [39], and not a blackbox. The "transparent" property of AHG representation provides the ability of supporting acquisition of different levels of patterns and that of recording the intermediate states. The AHG representation is conceptually efficient. Just as with other graphical representations, a variety of mature algorithms can be directly applied to achieve goals such as searching, matching and transforming. The AHG representation is also computationally efficient.

In summary, the attributed hypergraph representation can directly reflect the nature of the data set. The patterns encoded in an AHG are of different orders according to how much detailed information they contain. Along with the attributes assigned to each vertex and hyperedges, AHG provides a framework for future reasoning and inference. The AHG representation permits the encoding of both conceptual and relational descriptions at many levels of abstraction to exist simultaneously within the framework. This property is extremely desirable when forming conceptual clustering algorithms [61]. At the event level, an AHG captures the basic associations among the events in the given data set and avoids many shortcomings of other graphical representations. Unlike the blackbox approaches, the AHG representation makes the patterns easy to understand and visualize at different abstraction levels corresponding to their orders.

A	B	C	
FALSE	FALSE	FALSE	[a=FALSE, b=FALSE, c =FALSE]
FALSE	TRUE	TRUE	[a=FALSE, b=TRUE, c =TRUE]
TRUE	TRUE	FALSE	[a=TRUE, b=TRUE, c =FALSE]
TRUE	FALSE	TRUE	[a=TRUE, b=FALSE, c =TRUE]

Figure 3.3: Associations in XOR Data Set

3.4 Different Order Patterns in a Data Set

A real-world database usually contains different order patterns. Due to the statistical nature of the patterns in a real-world database, the existence of higher order patterns does not mean the existence of lower order patterns and *vice versa*. In other words, high order patterns cannot be synthesized from their low order patterns; they can only be found by testing the candidates of that order [103]. This is why some learning methods which deal only with low order relationships do not work well with data embodying high order patterns. A typical example is the XOR problem, where only 3rd-order patterns but no 2nd-order ones exist (Fig. 3.3).

In a database, the significance of a compound event can be measured by its frequency of occurrence against its expectation. In Fig. 3.4, a database table is characterized by N fields. The rectangles show the frequencies of the primary events. Event $[X_1 = A_1]$ is associated with event $[X_2 = B_1]$ since the co-occurrence of these two events are significant. The same thing happens to event $[X_1 = A_1]$ and $[X_4 = D]$, but not $[X_1 = A_1]$ and $[X_3 = C_1]$ because of less significance.

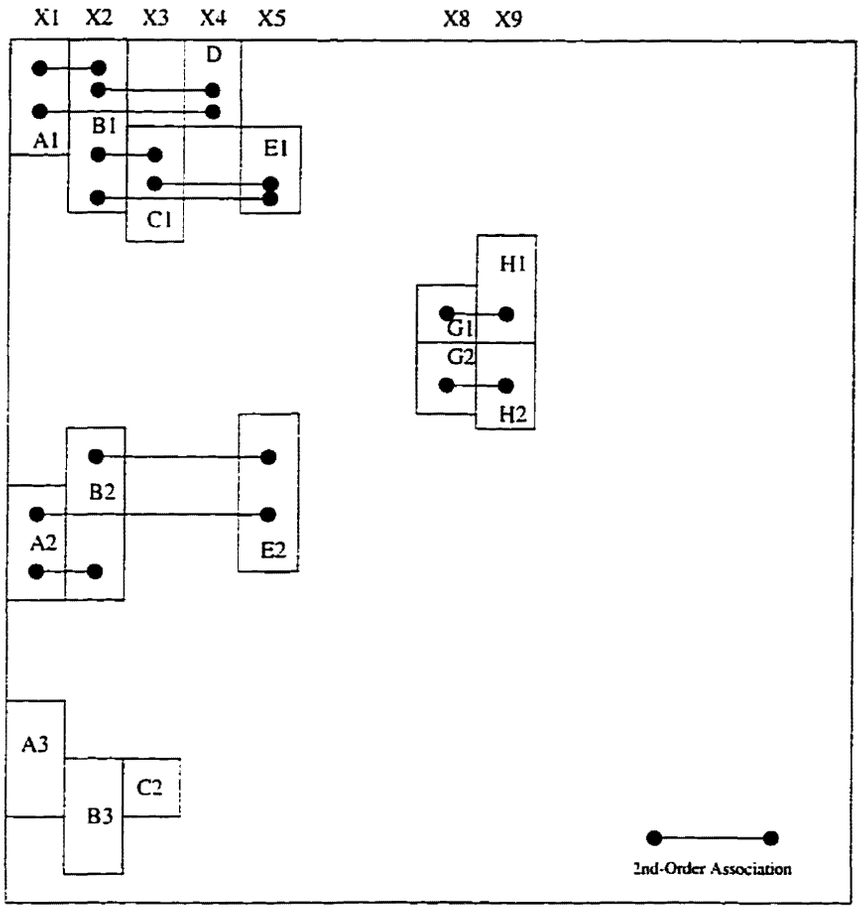

Figure 3.4: Associations in Database

$[X_1 = A_1, X_2 = B_1, X_4 = D]$ and $[X_1 = A_3, X_2 = B_3, X_3 = C_2]$ are all third order associations. The difference is that there are no second order associations in the compound event $[X_1 = A_3, X_2 = B_3, X_3 = C_2]$.

Fig. 3.5 shows some generalized cases of different order significant associations. The upper part of each case in this figure depicts the primary event occurrences and their pairwise associations in the data set, while the lower part furnishes the hypergraph representation of the associations among those primary events. In Case 1, there exist two second order patterns, $[A, B]$ and $[B, C]$, but no third order

pattern. In Case 2, the third order pattern $[A, B, C]$ and two second order patterns $[A, B]$, $[A, C]$ exist, but there is no significant association between B and C. Case 3 shows a situation where the third order pattern $[A, B, C]$ exists, but there is no second order association between A, B and C. Case 4 depicts a contrary instance where all of the three second order patterns, but not the third order pattern, exist. Case 5 presents the state in which the third order pattern $[A, B, C]$ and all of the three second order patterns $[A, B]$, $[B, C]$ and $[A, C]$ exist.

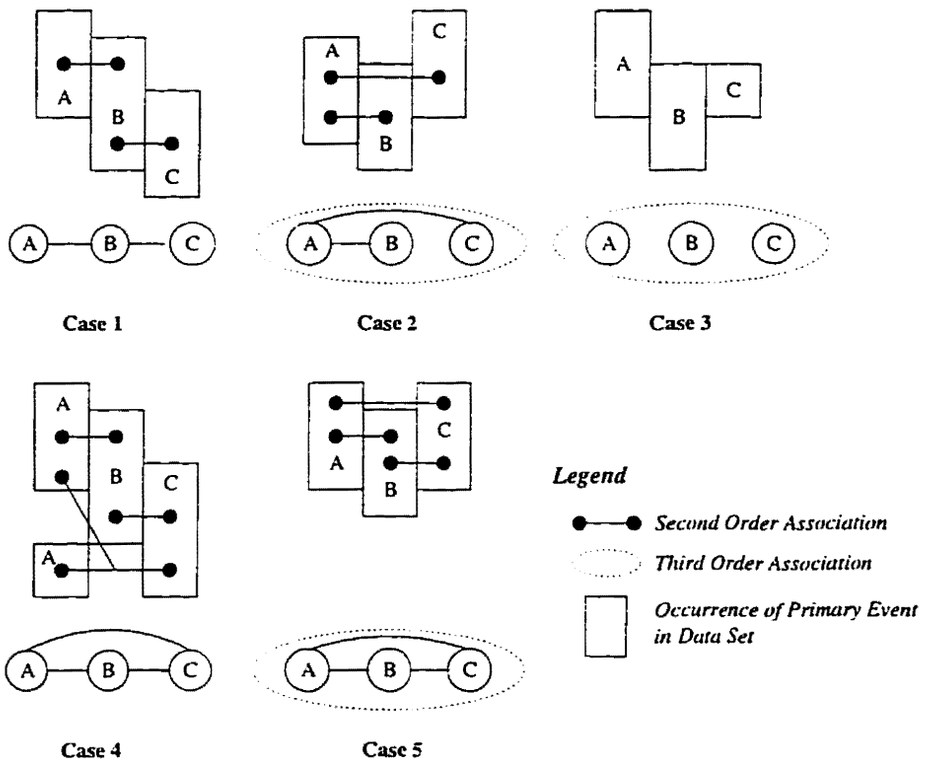

Figure 3.5: Different Orders of Significant Event Associations

In general, if $\mathbf{x}_j^s$ is a k-th order pattern in data set D, any l-th order ($l < k$) sub-compound event of $\mathbf{x}_j^s$ may not necessarily be a pattern in D [95] [103]. An example can be taken from the XOR problem. Note that $[\mathbf{A} = T, \mathbf{B} = T, \mathbf{C} = F]$ is a 3rd order pattern, but none of the sub-compound events $[\mathbf{A} = T, \mathbf{B} = T]$,

$[\mathbf{A} = T, \mathbf{C} = F]$ and $[\mathbf{B} = T, \mathbf{C} = F]$ are patterns. Case 3 of Fig. 3.5 shows such a situation. Conversely, even if all its $(k-1)$th order sub-compound events of a given k-th order compound event $\mathbf{x}_j^s$ are patterns, $\mathbf{x}_j^s$ itself may not necessarily be a pattern. Case 4 of Fig. 3.5 portrays such a situation. Hence, whether or not a compound event is a pattern cannot be determined by examining its sub-compound events and *vice versa*. This implies that, in general, higher order patterns cannot be synthesized from the lower order ones [95] [103].

3.5 Residual Analysis for Significant Associations

3.5.1 Detection of Deviation

As observed earlier, to discover patterns is to unearth the regularities previously unknown, or in other words, to find the regularities that significantly deviate from the current knowledge or what one can best guess. If the current knowledge is a pre-assumed model of the domain, patterns are those events which may not follow the model.

The most traditional approach to determine if a pre-assumed probability model is consistent with the given data set uses the Pearson χ-square (χ^2) statistic to provide an overall test of goodness-of-fit. It has the form of:

$$\chi^2 = \sum_i \frac{(o_i - e_i)^2}{e_i} \tag{3.3}$$

where o_i is the actual count of event i and e_i is the expected value of o_i under the proposed model. An alternative test is the likelihood-ratio χ-square (or L^2)

statistic:

$$L^2 = 2 \cdot \sum_i o_i \log\left(\frac{o_i}{e_i}\right). \tag{3.4}$$

where o_i and e_i bear the same meanings in Eqn. 3.3.

Other statistics can also be found to perform a similar overall test as χ^2 and L^2. The problem with these variable-based statistics is that none of them can give any specific information on the nature of deviations of the pre-assumed model from the actual data set [45].

Since this study concerns the events which are significantly different from the pre-assumed model, an event based statistic has to be applied to analyze the departure of an event from the model. In the feature space of the domain, each compound event can be illustrated as a hypercell in the space whose axes are the involved attributes. Fig. 3.6 shows a third order compound event $\mathbf{x}_1^s$ ($[x_{11}, x_{21}, x_{31}]$) of attribute X_1, X_2 and X_3. The three shaded squares represent its second order sub-compound events. Any instances in data set D satisfying $X_1 = x_{11}$, $X_2 = x_{21}$ and $X_3 = x_{31}$ fall into this cell. A statistical significance test on this cell will determine whether $\mathbf{x}_1^s$ is a pattern or not.

Considering a k-compound event $\mathbf{x}_j^s$, one would like to determine if it is a pattern by investigating whether or not the number of instances falling into this cell significantly deviates from what is expected based on a default model. In statistics, the residual is widely used to investigate how a model fails. Residuals provide indications of which individual frequency counts are larger or smaller than the expected under the pre-assumed probability model [47]. Residual analysis is an event based analysis that examines individual events instead of the random variables as a whole. Thus, subtle information of specific events can be provided. In practice, the relations among events are more interesting than those among variables [90] [102]

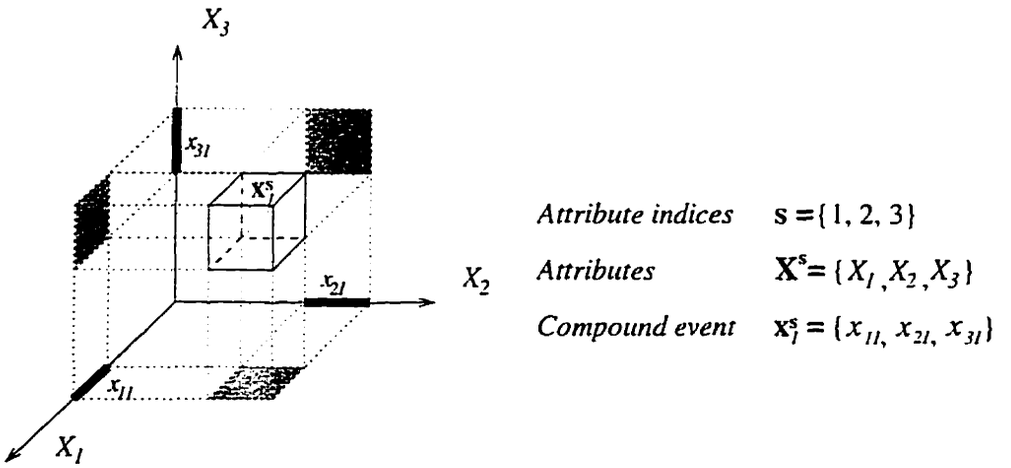

Figure 3.6: A Third Order Compound Event

[103]. For example, we would like to know if an injury in the upper back is associated with the Supply and Service department, more than just if that injury type is related to a certain department of a company. Much of such subtle information may have escaped the user's awareness. The discovered patterns could significantly enhance current system management and performance.

Definition 3.11 *The* residual *of an event* $\mathbf{x}_j^s$ *against the pre-assumed model in* D *is defined as the difference between the actual occurrence of the event and its expected occurrence. That is:*

$$d_{\mathbf{x}_j^s} = o_{\mathbf{x}_j^s} - e_{\mathbf{x}_j^s} \tag{3.5}$$

where $o_{\mathbf{x}_j^s}$ *is the actual count of event* $\mathbf{x}_j^s$ *in* D*, and* $e_{\mathbf{x}_j^s}$ *is the expected occurrence under the pre-assumed model.*

According to Wrigley [105], the absolute difference between the observed and the expected frequencies, $|o_{\mathbf{x}_j^s} - e_{\mathbf{x}_j^s}|$, cannot be employed for evaluating the relative size of the discrepancy between $o_{\mathbf{x}_j^s}$ and $e_{\mathbf{x}_j^s}$ because the absolute difference may be

affected by the marginal totals. In practice, the residual is first standardized before any analysis is conducted.

Definition 3.12 (Haberman, 1973 [45]) *The* standardized residual $z_{\mathbf{x}_j^s}$ *of event* $\mathbf{x}_j^s$ *is defined as the ratio of its residual and the square root of its expectation. That is:*

$$z_{\mathbf{x}_j^s} = \frac{d_{\mathbf{x}_j^s}}{\sqrt{e_{\mathbf{x}_j^s}}}. \tag{3.6}$$

With a little additional computation, one can use the *adjusted residual* other than the standardized residual in the analysis.

Definition 3.13 (Haberman, 1973 [45], 1978 [47]) *The* adjusted residual $r_{\mathbf{x}_j^s}$ *of event* $\mathbf{x}_j^s$ *is defined as:*

$$r_{\mathbf{x}_j^s} = \frac{d_{\mathbf{x}_j^s}}{\sqrt{c_{\mathbf{x}_j^s}}}, \tag{3.7}$$

where $c_{\mathbf{x}_j^s}$ *is the variance of* $d_{\mathbf{x}_j^s}$.

The difference of the statistical properties between z and r will be discussed in the following section.

Using residuals to analyze the occurrences of an event, the process can be divided into three steps: 1) a default probabilistic model is assumed; 2) the residual is calculated; and 3) a statistic test is conducted to draw the conclusion whether or not the event under test deviates from the default model.

3.5.2 Properties of Residuals

Residuals provide a quantitative measure to determine how an event deviates from the default probabilistic model. Yet how large is a residual when the deviation of

the event is considered significant? To answer this question, the distributions of standardized and adjusted residuals are studied.

The distributions of the residuals are related to the pre-assumed models. In statistics, the majority of interesting models for categorical data are *log-linear models.* A family of log-linear models is often referred to as an *exponential family.* Consider a random vector $\mathbf{X}$ containing N random variables. The random vector $\mathbf{X}$ has mean $\mathbf{e} = \{e_i | 1 \leq i \leq N\}$. We assume e_i is positive for each i. Thus for each i, $\log e_i$ is well defined. Therefore, if $\mu = \{\log e_i | 1 \leq i \leq N\}$, then $\mu \in \mathcal{R}^N$. In a log-linear model, it is assumed that $\mu \in \mathcal{L}$, where $\mathcal{L}$ is a p-dimensional linear manifold contained in $\mathcal{R}^N$ and $0 < p \leq q$, where $q \leq N$. In other words, the logarithm of a probability model $P(\mathbf{x}|\theta_1, \cdots, \theta_k)$, where $\theta_1, \cdots, \theta_k$ are k real valued parameters, can be represented as [5]:

$$\log P(\mathbf{x}|\theta_1, \cdots, \theta_k) = \sum_j^m g_j(\mathbf{x})\varphi_j(\theta_1, \cdots, \theta_k) + h(\mathbf{x}) - K(\theta_1, \cdots, \theta_k) \tag{3.8}$$

where $g_j()$, $\varphi_j()$ and $h()$ are all real valued functions of their arguments. The function $K()$ has an expression that makes $\sum_{\mathbf{x}} P(\mathbf{x}|\theta_1, \cdots, \theta_k) = 1$. In Eqn. 3.8, m, the smallest integer for which Eqn. 3.8 is possible, is called the *dimension* of the model.

Two frequently used log-linear models in analyzing categorical data are the Poisson model and the multinomial model. While the Poisson models are usually applied to data sets with variable numbers of instances, the multinomial models are generally used to investigate data sets with fixed numbers of samples, such as an existing database.

Let $X_1, \cdots, X_k$ be multinomially distributed with parameters M and $p_1, \cdots, p_k$,

i.e.,

$$P(x_1, \cdots, x_k | p_1, \cdots, p_k, M) = \begin{pmatrix} M \\ x_1, \cdots, x_k \end{pmatrix} p_1^{x_1} \cdots p_k^{x_k}. \tag{3.9}$$

Then

$$\begin{aligned} &\log P(x_1, \cdots, x_k | p_1, \cdots, p_k, M) \\ &= \sum_{j=1}^{k-1} x_j \cdot \log(\frac{p_j}{p_k}) + M \log p_k + \log \begin{pmatrix} M \\ x_1, \cdots, x_k \end{pmatrix} \end{aligned} \tag{3.10}$$

and Eqn. 3.10 suggests that the multinomial distribution Eqn. 3.9 is log-linear. The maximum likelihood estimates of the parameters $p_1, \cdots, p_k$ are given by the equations:

$$x_j = E[X_j] = Mp_j, \quad j = 1, \cdots, k-1. \tag{3.11}$$

The maximum likelihood estimates

$$\hat{p}_j = \frac{x_j}{M} \tag{3.12}$$

for the original parameters are thus found directly from Eqn. 3.11.

The general background of residual analysis can be found in [28] and [29]. Haberman [45] [46] discussed the properties of residuals for log-linear models. The results are borrowed and shown below without giving the proof.

The data set D can be represented as an N-dimensional contingency table where N is the number of attributes. The size of the contingency table is fixed, which means that the dimension and the size of the table does not change when the number of instances M in D changes. For a given data set, since we know exactly how many attributes we have and we assume the domain of each attribute is also

known, the size of the contingency table for the data set is always fixed. In this case, if the chosen model for the domain is log-linear and the model holds for the data set, the standardized residual in Eqn. 3.6 has an asymptotic normal distribution. The mean of the normal distribution of standardized residual is 0 and the variance is given by

$$\sigma = \frac{c_{\mathbf{x}_j^s}}{e_{\mathbf{x}_j^s}}. \tag{3.13}$$

If $c_{\mathbf{x}_j^s} > 0$ and $\hat{c}_{\mathbf{x}_j^s}$ is the maximum likelihood estimate of $c_{\mathbf{x}_j^s}$. The adjusted residual illustrated by Eqn. 3.7 has an approximate asymptotic normal distribution of zero mean and unit variance [46] [47]. If the marginal counts are much smaller than the total number of instances, the standardized residuals and the adjusted residuals are similar.

When the distribution of the adjusted/standardized residual of a compound event is known, it can be determined whether or not a candidate significantly deviates from the default model. Let us consider a k-compound event $\mathbf{x}_j^s$. Whether or not $\mathbf{x}_j^s$ is significant can be determined by the value of its residuals. If the absolute value of its adjusted residual $r_{\mathbf{x}_j^s}$ exceeds 1.96, then, by conventional criteria, it can be concluded with a confidence level of 95% that the difference between $o_{\mathbf{x}_j^s}$ and $e_{\mathbf{x}_j^s}$ is significant. The default model does not hold for this event. If the true value of $r_{\mathbf{x}_j^s}$ exceeds 1.96, it can be concluded that the primary events of $\mathbf{x}_j^s$ are "associated" and will likely occur together, with a confidence level of 95%. In this case, $\mathbf{x}_j^s$ is referred to as a *positive event association*. If the true value of $r_{\mathbf{x}_j^s}$ is less than -1.96, it can be concluded that the primary events in $\mathbf{x}_j^s$ are unlikely to co-occur. It is then referred to as a *negative event association*.

This analytical method applies to both standardized and adjusted residuals. If the marginal counts are much smaller than the total number of instances in the

data set, the standardized residual is suggested as the analytical parameter since it requires less computation. The variance will not be estimated. If this condition does not stand or there is no means to justify such criteria, adjusted residual should be used since it is more accurate.

Based on residual analysis, this association detecting process is qualitative in that it determines whether a compound event is a significant association or not. It is also quantitative in that the significant level along with the probabilities are calculated and recorded.

3.5.3 Parameter Estimation

According to Eqn. 3.7, there are two parameters to be estimated for a compound event given data set D before the value of the adjusted residual can be calculated. One is the expected occurrence of the candidate event and the other is the variance of its residual d.

To estimate both parameters, a default model has to be chosen first according to the problem domain and the available knowledge. If *a priori* knowledge about the domain is not available, normally a model assuming the independence of the random variables is selected. In practice, when the number of variables are large (greater than three), a large number of relationships among the variables may be considered. To conduct residual analysis in this high dimensional situation, multinomial hierarchical models are always engaged [47]. This is because hierarchical models provide a general tool for the examination of contingency tables in which two or more variables are cross-classified. Such models are based on a general method of parameterization commonly encountered in the analysis of factorial tables. With the hierarchical models, the independence situations of the variables can be easily

interpreted.

Here three variables are used to show the properties of hierarchical models and the parameter estimation of residuals. The results can be easily applied to domains with more variables. For a detailed discussion of hierarchical models, please refer to Appendix A or [47].

In a three-variable case, each third-order compound event is a hypercell in the feature space (see Fig. 3.6). The space can be represented as a three-way contingency table of three variables A, B and C with finite length L, width W and height H respectively. Let us consider a generic hypercell indexed as (i, j, k), $1 \leq i \leq L$, $1 \leq j \leq W$ and $1 \leq k \leq H$. If e_{ijk} is the expected value of the actual count o_{ijk}, the hierarchical models have the form:

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} + \lambda_{ik}^{AC} + \lambda_{jk}^{BC} + \lambda_{ijk}^{ABC} \tag{3.14}$$

where all the λ_+^+'s are unique parameters that satisfy:

$$\begin{aligned} \sum \lambda_i^A = \sum \lambda_j^B = \sum \lambda_k^C = \sum_i \lambda_{ij}^{AB} = \sum_j \lambda_{ij}^{AB} = \sum_i \lambda_{ik}^{AC} = \sum_k \lambda_{ik}^{AC} \\ = \sum_j \lambda_{jk}^{BC} = \sum_k \lambda_{jk}^{BC} = \sum_i \lambda_{ijk}^{ABC} = \sum_j \lambda_{ijk}^{ABC} = \sum_k \lambda_{ijk}^{ABC} = 0. \end{aligned} \tag{3.15}$$

As in the analysis of variance, the λ_i^A is called the main effect for variable A, the λ_{ij}^{AB} are $A \times B$ interaction, and the λ_{ijk}^{ABC} are $A \times B \times C$ interaction. The λ_{ij}^{AB}, λ_{ik}^{AC}, λ_{jk}^{BC} are the two-factor interactions, while the λ_{ijk}^{ABC} is the three-factor interaction. If no restrictions are imposed on the λ-parameters, Eqn. 3.14 and Eqn. 3.15 specify a saturated model [42].

In other hierarchical models, some λ-parameters are set to 0. The hierarchical restriction is followed such that, if any λ-parameter with superscript S is set to 0,

then any λ-parameter of the same or higher order is set to 0. Here a λ-parameter has the same or higher order if its superscript contains each letter of S. For example, one may assume λ_{ijk}^{ABC} is 0, which leads to a model of no three-factor interaction

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} + \lambda_{ik}^{AC} + \lambda_{jk}^{BC}.$$

If λ_{ij}^{AB} is set to 0, λ_{ijk}^{ABC} has to be 0, so that

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ik}^{AC} + \lambda_{jk}^{BC}.$$

On the other hand, the assumption

$$\log e_{ijk} = \lambda + \lambda_{ij}^{AB} + \lambda_k^C$$

does not yield a hierarchical log-linear model, since the λ_i^A is assumed to be 0, the λ_{ij}^{AB} must also be 0.

Hierarchical models are an attractive family of models for high-way contingency table analysis because of their simple interpretations. As Birch [8] and Goodman [42] note, in a three-way table, except for the model of no three-factor interaction, each hierarchical model is equivalent to one or more hypotheses of independence, conditional independence, or equiprobability (see Appendix A for details) and the model of no three-factor interaction can be interpreted in terms of comparison of cross-product ratios in several two-way tables. Some computational convenience is associated with hierarchical models. Maximum-likelihood estimates and adjusted residuals for hierarchical models can be found without recourse to iterative computation, except for the model of no three-factor interaction. These properties can be easily generalized for higher order analysis [47].

In practice, if there is no *a priori* knowledge on the problem domain, the best we can guess is an independent model which assumes that the random variables describing the domain are mutually independent[2]. In such a case, the log-linear hierarchical models in Eqn. 3.14 and Eqn. 3.15 degrade to

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C \tag{3.16}$$

so that all two-factor and three-factor interactions are assumed to be 0.

In analyzing a data set D, it can be assumed that the number of instances M in D is fixed, so that the hypercells in the table have a multinomial distribution with sample size M [3]. For convenience, the probability of the cell (i, j, k) is denoted as p_{ijk} and the marginal probabilities as p_i^A, p_j^B and p_k^C respectively.

The model in Eqn. 3.16 holds if and only if A, B and C are mutually independent (see [42] for proof), so that

$$p_{ijk} = p_i^A p_j^B p_k^C. \tag{3.17}$$

The maximum-likelihood estimate of e_{ijk} is

$$\hat{e}_{ijk} = \frac{o_i^A o_j^B o_k^C}{M^2} = M \hat{p}_i^A \hat{p}_j^B \hat{p}_k^C, \tag{3.18}$$

where o_i^A, o_j^B and o_k^C are marginal counts and $\hat{p}_i^A$, $\hat{p}_j^B$ and $\hat{p}_k^C$ are the maximum-likelihood estimates of the respective marginal probabilities. Since M is fixed, the

[2]If the domain knowledge leads to other models instead of the independence model, and the model is still log-linear, the properties of residuals discussed above still stand, but the following parameter estimations should be revised to fit the actual model.

[3]One may also consider the case in which M has a Poisson distribution with mean m. This is helpful when developing an incremental system with changing sample size.

table has a multinomial distribution. We have

$$\begin{aligned} \hat{p}_i^A &= \frac{o_i^A}{M} \\ \hat{p}_j^B &= \frac{o_j^B}{M} \\ \hat{p}_k^C &= \frac{o_k^C}{M} \end{aligned} \tag{3.19}$$

The maximum-likelihood estimate of the variance c_{ijk} of residual $o_{ijk} - e_{ijk}$ is more complicated. Fortunately, since the independence model is decomposable, there is an explicit solution from sufficient marginals to the maximum-likelihood estimate of the variance [46] [5]. Only the results are presented here without proofs. Under the independence model, the maximum-likelihood estimate of residual variance can be calculated as

$$\hat{c}_{ijk} = \hat{e}_{ijk}(1 - \hat{p}_i^A\hat{p}_j^B - \hat{p}_i^A\hat{p}_k^C - \hat{p}_j^B\hat{p}_k^C + 2\hat{p}_i^A\hat{p}_j^B\hat{p}_k^C) \tag{3.20}$$

$$= \frac{o_i^A o_j^B o_k^C}{M^2}\left(1 - \frac{o_i^A}{M}\frac{o_j^B}{M} - \frac{o_i^A}{M}\frac{o_k^C}{M} - \frac{o_j^B}{M}\frac{o_k^C}{M} + 2\frac{o_i^A}{M}\frac{o_j^B}{M}\frac{o_k^C}{M}\right) \tag{3.21}$$

From Eqn. 3.18, Eqn. 3.19 and Eqn. 3.21, parameters to calculate the adjusted residual of three variables can all be estimated based on marginal counts and total sample size. Since the model of mutual independence is always decomposable, there is always an explicit solution to the likelihood equations and the estimated expected numbers are direct functions of the sufficient marginals (by theorem stated by Goodman [41] and proved by Haberman [46]). The equations 3.18 and 3.21 for calculating expected occurrence and variance are generalized to higher order situations in the following.

For a compound event x_j^s of order $|s|$, the maximum-likelihood estimate of the

expected occurrence of $\mathbf{x}_j^s$ given D is

$$\hat{e}_{\mathbf{x}_j^s} = M \cdot \hat{Pr}(\mathbf{x}_j^s) = M \cdot \prod_{x_{ip} \in \mathbf{x}_j^s} \hat{Pr}(X_i = \alpha_i^p) \tag{3.22}$$

$$= M \cdot \prod_{x_{ip} \in \mathbf{x}_j^s} \frac{o_{x_{ip}}}{M}, \tag{3.23}$$

and the maximum likelihood estimate of the variance of residual $d_{\mathbf{x}_j^s}$ can be calculated as

$$\hat{c}_{\mathbf{x}_j^s} = \hat{e}_{\mathbf{x}_j^s} \left(1 - \sum_{x_{ip} \in \mathbf{x}_j^s} \prod_{\substack{x_{jq} \in \mathbf{x}_j^s \\ x_{jp} \neq x_{ip}}} \hat{Pr}(x_{jq}) + (|\mathbf{s}| - 1) \prod_{x_{ip} \in \mathbf{x}_j^s} \hat{Pr}(x_{ip}) \right)$$

$$= M \cdot \prod_{x_{ip} \in \mathbf{x}_j^s} \frac{o_{x_{ip}}}{M} \left(1 - \sum_{x_{ip} \in \mathbf{x}_j^s} \prod_{\substack{x_{jq} \in \mathbf{x}_j^s \\ x_{jp} \neq x_{ip}}} \frac{o_{x_{jp}}}{M} + (|\mathbf{s}| - 1) \prod_{x_{ip} \in \mathbf{x}_j^s} \frac{o_{x_{ip}}}{M} \right) \tag{3.24}$$

With Eqn. 3.23 and Eqn. 3.24, the adjusted residual of any order compound events can be obtained through marginal counts from a given data set D.

If only second order patterns are considered as in APACS [17] which uses a two-way contingency table, Eqn. 3.23 can be simplified to

$$\hat{e}_{x_{ij}} = \hat{e}_{ij} = \frac{o_{i+} \cdot o_{+j}}{M}$$

where o_{i+} is the row sum of the i-th row in the 2-way table and o_{+j} is the column sum. Physically, o_{i+} corresponds to the marginal counts of the primary event at

row i and o_{+j} to the marginal counts of the primary event at column j. At the same time, Eqn. 3.24 can be simplified to

$$\begin{aligned}\hat{c}_{x_{ij}} &= \hat{c}_{ij} \\ &= \hat{e}_{ij} \cdot \left(1 - \frac{o_{i+}}{M} - \frac{o_{+j}}{M} + \frac{o_{i+}}{M}\frac{o_{+j}}{M}\right) \\ &= \hat{e}_{ij} \cdot \left(1 - \frac{o_{i+}}{M}\right)\left(1 - \frac{o_{+j}}{M}\right).\end{aligned}$$

The two parameters $\hat{e}_{ij}$ and $\hat{c}_{ij}$ are exactly the same as their counterparts stated in [17]. This shows that the method proposed here is a general case of APACS/PIT by Chan [16] [17].

3.6 Algorithm for Pattern Discovery

The algorithm developed for discovering different order significant event associations from a data set is described in this section. This algorithm assumes:

1. Data set D has a finite number of instances. The instance number M is fixed and previously known. This is required by the maximum-likelihood estimates of the marginal probabilities and expected occurrence when calculating residuals. If M is changing, one may use Poisson models instead of multinomial models to conduct the estimation;

2. Data set D is described by N attributes (random variables). N is finite. Every attribute (variable) is categorical or discrete-valued. Continuous variables in the domain have been discretized first by discretization algorithms such as those in [99] and [21];

3. The domain of each attribute (variable) is available. The cardinality of each attribute (variable) is finite and known *a priori*. Together with the second assumption, this requirement implies a contingency table of fixed dimension, which is critical for the discussions in Section 3.5.2 and Section 3.5.3 to be correct.

3.6.1 Overview of the Algorithm

The objective of the pattern discovery algorithm is to find all the significant event associations inherent in a given data set and generate an attributed hypergraph to represent the discovered patterns. The discovered patterns should include both the positive and the negative event associations of different orders. The algorithm can be subdivided into two phases:

1. Find all the candidate compound events which are likely to be significant given a data set.

2. Test each candidate and determine whether or not it is significant by calculating the standardized/adjusted residual.

This process repeats until all possible orders are exhausted or no candidates can be generated.

One can easily design an algorithm that exhaustively generates all the possible combinations of the primary events at different orders and test their significance by calculating their adjusted residuals. In such an approach, high-way contingency tables of different orders are always generated to represent the data. Each cell of the tables is a hypothesis to be tested. When the order is low, the performance of this method is satisfactory. An example is APACS [17], in which, two-way

contingency tables are used to search for second order patterns. However, when the order is high (greater than 3), the contingency table approach is not feasible. For instance, if we have a problem with 8 variables, each of which can take on one of the 5 values, at order 3, there are totally 6,000 hypotheses to be tested with an exhaustive algorithm. At order 4, this number increases to 43,750. Basically, exhaustive approaches work well only with small problems containing just a few attributes of small domains. It is not realistic for real-world problems. Therefore, it is crucial to avoid exhaustive search in solving real-world problems.

The algorithm proposed here generates the pattern candidates on the fly which allows heuristics to be applied to eliminate uninformative candidates at an early stage and remove them from further consideration. It is outlined as follows:

Notation:

N	Number of attributes in D
P	Set of all primary events
α_i	primary events of attribute X_i, $1 \leq i \leq N$
$A(E, V)$	AHG representation of significant associations discovered
	E is the set of hyperedges and V is the set of vertices
r	Current working order
C_r	Set of association candidates of order r
c_{ri}	The i-th candidate in C_r
aj_i	Adjusted residual of c_{ri}
$conf$	confidence level (e.g. 2.58 for 99%)

1. **Begin Procedure** *detect_association*( )
2. $P := \{\alpha_i | 1 \leq i \leq N\}$ // primary event set

```
    A.E := φ                                          // hyperedges
    A.V := P                                          // vertices are the primary events
    r := 2                                            // initial order is 2
    C_r := φ                                          // reset candidate set
    C_r := gen_candiate(r, P, C_r)                    // generate candidates
    for ( C_r ≠ φ .AND. r ≤ N ) do begin
        forall candidate c_ri ∈ C_r do begin
            aj_i := adj_res(c_ri)                     // calculate adjusted residual
            if |aj_i| > conf
                A.E := A.E ∪ {c_ri}                   // add hyperedge to AHG
            endif
        end
        r++                                           // increment order
        C_r := gen_candidate(r − 1, P, C_{r−1})       // generate new candidates
    end
    Output A
End Procedure
```

In the algorithm *detect_association*(), the vertex set of the initial attributed hypergraph contains all the primary events, while no hyperedge exists. The pattern discovery process starts generating all the second order candidates by calling *gen_candidates*() with order 2, all the primary events and an empty set of current candidates as arguments. The adjusted residuals of all the second order candidates are calculated. If the absolute value of a candidate exceeds the threshold of the confidence level, a hyperedge corresponding to this compound event is added to the current attributed hypergraph. When all the second order candidates have been examined, the algorithm goes to one order higher. The higher order pattern candidates are generated by calling *gen_candidates*() again with order, primary events

and the lower order candidates as the arguments. This allows the algorithm to examine the lower order candidates to determine if higher order candidates containing these lower order candidates are likely to be informative. Such a mechanism makes it possible to eliminate uninformative compound events as early as possible. If and when no more candidates can be generated, or the algorithm has reached the highest order which is the number of variables, the process stops. The outcome of the algorithm is an attributed hypergraph representing all the discovered patterns. This algorithm guarantees that all the event associations in the attributed hypergraph are significant according to their residual with a user-specified confidence level.

3.6.2 Selection of Informative Candidates

Exhaustive search is avoided by eliminating the candidates which cannot be significant or to which the statistical hypothesis test based on adjusted residual is not applicable. This approach is called *candidate selection*. In this section, two general criteria for eliminating impossible pattern candidates are presented. One is based on the validation of the significance test, and the other is for testing higher order negative patterns.

Validation of the Significance Test

The residual analysis used in the discovery process is based on the asymptotic statistical properties. These properties stand only when the sample size is large. If the sample size is not large enough, the information provided by the data set cannot draw a reliable conclusion.

For a testing compound event, the requirement of a large sample for asymptotic analysis is reflected as the expected occurrence of that event. Large-sample approximations for the adjusted residual are probably adequate if the expected occurrence exceeds 25 for model validation with a conservative estimation [47]. To detect the significance of deviation from a model, this requirement can be relaxed. If the expected occurrences are smaller, the residuals can still provide information concerning the patterns of deviations [47].

Setting the threshold for expected occurrences for a valid statistical test is a trade-off between detecting random combinations as associations and missing some real associations. In practice, this threshold is set between 5 and 25 depending upon user's choice and common practice in hypothesis testing for contingency tables [103]. The noise level of the data set is also a factor to be considered. On the other hand, this threshold helps a learning system avoid the problem of overfitting. Higher order patterns are considered more specific than the lower order ones. At high order, expectations of compound events will decrease. If the expectations are less than the threshold, the statistical test is not meaningful. The continuity of generating higher order patterns from these events is stopped so as to keep the detected patterns at a certain simplicity level.

Those events for which statistical testing is no longer valid are not included in the formalization of higher order patterns. The threshold of expectation can then be applied to eliminate those ineligible compound events to avoid exhaustive search. That is to say, only when

$$e_{\mathbf{x}_j^*} \geq \omega_e \tag{3.25}$$

where ω_e is the requirement for a valid test, higher order compound events contain-

ing $\mathbf{x}_j^s$ are to be tested.

Proposition 3.1 *If the expected occurrence of a compound event $\mathbf{x}_j^s$ is $e_{\mathbf{x}_j^s}$, the expected occurrences of any higher order compound event $\mathbf{x}_i^r$ such that $\mathbf{x}_i^r \supset \mathbf{x}_j^s$ cannot be greater than $e_{\mathbf{x}_j^s}$.*

Proof:

Since $\mathbf{x}_i^r \supset \mathbf{x}_j^s$, $\mathbf{x}_i^r$ can be rewritten as :

$$\mathbf{x}_i^r = \mathbf{x}_j^s \bigcup \mathbf{x}_k^t$$

where $\mathbf{r} = \mathbf{s} \bigcup \mathbf{t}$ and $\mathbf{s} \bigcap \mathbf{t} = \phi$. Then, $\mathbf{x}_i^r$ is the union of two disjoint compound events $\mathbf{x}_j^s$ and $\mathbf{x}_k^t$. According to Eqn. 3.23, we have

$$\begin{aligned}
\hat{e}_{\mathbf{x}_i^r} &= M \cdot \prod_{x_{p+} \in \mathbf{x}_i^r} Pr(x_{p+}) \\
&= M \cdot \left(\prod_{x_{a+} \in \mathbf{x}_j^s} Pr(x_{a+}) \cdot \prod_{x_{b+} \in \mathbf{x}_k^t} Pr(x_{b+}) \right) \\
&= \hat{e}_{\mathbf{x}_j^s} \cdot \prod_{x_{b+} \in \mathbf{x}_k^t} Pr(x_{b+})
\end{aligned}$$

Since $0 \leq Pr(\cdot) \leq 1$, $0 \leq \prod_{x_{b+} \in \mathbf{x}_k^t} Pr(x_{b+}) \leq 1$. Hence:

$$\hat{e}_{\mathbf{x}_i^r} \leq \hat{e}_{\mathbf{x}_j^s}$$

The proposition is proved. □

Thus, when the expected occurrence of a compound event $\mathbf{x}_j^s$ is less than ω_e, all higher order compound events with $\mathbf{x}_j^s$ as one of their sub-compound events have an

expectation less than ω_e. Then, they are statistically untestable. These compound events should be eliminated from further consideration.

Let us consider a database of animals. The animals in the database belong to seven types, *mammal*, *bird*, *reptile*, *fish*, *insect*, *amphibian* and *invertebrate*. Sixteen other attributes, such as *hair*, *feathers*, *eggs*, *backbone* and *milk*, etc.[4], characterize each animal. When the criterion discussed earlier is applied in the discovery process, if we found a candidate $[Type = reptile, Feathers = yes]$ has an expectation of only 2.2, for example, and our threshold of expectation for valid test is 5, then, there is no need to calculate the residuals and all higher order compound events containing $[Type = reptile, Feathers = yes]$ such as $[Type = reptile, Feathers = yes, milk = no]$ and $[Type = reptile, Feathers = yes, eggs = yes]$ are truncated.

Negative Patterns

When determining whether or not a compound event $\mathbf{x}_j^s$ is a possible component of a higher order pattern, one should realize that there are some differences in handling negatively significant and positively significant associations. Negative event associations can be used to eliminate higher order candidates. If $\mathbf{x}_j^s$ is an overwhelming negative pattern, higher order compound events of $\mathbf{x}_j^s$ should not be included in the candidate set. Instead of the expected occurrence of a compound event, the actual occurrence must be examined to determine whether or not a negative pattern is overwhelming.

A negative pattern suggests that the primary events in this pattern are not likely to occur together. If primary events x_{ip_1} and x_{jp_2} never occur together (the occurrence of compound event $[x_{ip_1}, x_{jp_2}]$ is zero) in data set D, the primary events

[4]For a full list of the attributes in this database, please refer to Section 5.2.3

of any compound events containing x_{ip_1} and x_{jp_2} will not occur together either. Such a compound event is called an *overwhelming* negative pattern, since in the inference process, if anyone of x_{ip_1} and x_{jp_2} is observed, no more higher order evidence will be necessary to draw the conclusion that the other event will not happen together. To determine whether or not higher order compound events $\mathbf{x}_i^r$ containing negative pattern $\mathbf{x}_j^s$ should be tested, the occurrence of $\mathbf{x}_j^s$, $o_{\mathbf{x}_j^s}$, should be investigated.

Proposition 3.2 *Let $\mathbf{x}_j^s$ be a compound event whose occurrence is $o_{\mathbf{x}_j^s}$. The occurrences of any higher order compound event $\mathbf{x}_i^r$ such that $\mathbf{x}_i^r \supset \mathbf{x}_j^s$ cannot be greater than $o_{\mathbf{x}_j^s}$.*

Proof:

The proof is trivial. Without loss of generality, assume that $\mathbf{x}_i^r$ is only one order higher than $\mathbf{x}_j^s$. Then, $\mathbf{x}_i^r$ contains a primary event x_{pk} of attribute X_p which is not covered by $\mathbf{x}_j^s$. We can rewrite $\mathbf{x}_i^r$ as $\mathbf{x}_j^s + x_{pk}$. It is also assumed that the cardinality of the domain of attribute X_p is m_p and $m_p > 0$. Obviously $1 \leq k \leq m_p$.

Then the occurrence of $\mathbf{x}_j^s$ is:

$$o_{\mathbf{x}_j^s} = \sum_{a=1}^{m_p} o_{\mathbf{x}_j^s|x_{pa}}$$

where $o_{\mathbf{x}_j^s|x_{pa}}$ is the count of the event $\mathbf{x}_j^s$ given x_{pa} happens. Since $1 \leq k \leq m_p$, and $o_+ \geq 0$,

$$\begin{aligned} o_{\mathbf{x}_j^s} &= o_{\mathbf{x}_j^s|x_{pk}} + \sum_{a=1, a\neq k}^{m_p} o_{\mathbf{x}_j^s|x_{pa}} \\ &= o_{\mathbf{x}_i^r} + \sum_{a=1, a\neq k}^{m_p} o_{\mathbf{x}_j^s|x_{pa}} \end{aligned}$$

$$\geq o_{\mathbf{x}_i^r}$$

If $\mathbf{x}_i^r$ is more than one order higher than $\mathbf{x}_j^s$, the above step can be recursively applied to have the proposition proved. □

Given $e_{\mathbf{x}_i^r}$, to test whether or not adjusted residual $d_{\mathbf{x}_i^r}$ is greater (or less) than a threshold is equivalent to testing whether or not $o_{\mathbf{x}_i^r}$ is greater (or less) than a correspondent threshold. Defined as the requirement of $o_{\mathbf{x}_i^r}$ for $\mathbf{x}_i^r$ to be significant, $\bar{o}_{\mathbf{x}_i^r}$ is calculated by

$$\bar{o}_{\mathbf{x}_i^r} = \hat{e}_{\mathbf{x}_i^r} + K \cdot \sqrt{\hat{c}_{\mathbf{x}_i^r}}. \tag{3.26}$$

where, $\hat{c}_{\mathbf{x}_i^r}$ is calculated by Eqn. 3.24 and K is the constant according to a fixed confidence level (e.g. 1.96 for 95%).

Let $\mathbf{x}_j^s$ be a p-th order negatively significant pattern whose occurrence is $o_{\mathbf{x}_j^s}$. Of all the q-th order compound events ($q > p$) which contain $\mathbf{x}_j^s$ as a sub-compound event, according to Proposition 3.1, $\mathbf{x}_i^r$ has the minimum expectation among all of its sub-compound event. Hence, the requirement of $\bar{o}_{\mathbf{x}_i^r}$ to make $\mathbf{x}_i^r$ negatively significant is the smallest among all of its sub-compound events. Therefore, If $o_{\mathbf{x}_j^s}$ is less than $\bar{o}_{\mathbf{x}_i^r}$, according to Proposition 3.2, all the sub-compound events of $\mathbf{x}_i^r$ which contain $\mathbf{x}_j^s$ have occurrences less than $\bar{o}_{\mathbf{x}_i^r}$. They are all negatively significant event associations. Thus, it is not necessary to test those compound events whose orders are between p and q. If q is the highest order for the data set, all of the compound events containing $\mathbf{x}_j^s$ will be eliminated from consideration.

When $o_{\mathbf{x}_j^s}$ is very small compared with $e_{\mathbf{x}_j^s}$, it provides very strong evidence that the primary events in $\mathbf{x}_j^s$ are not likely to co-occur. An extreme case happens when $o_{\mathbf{x}_j^s}$ is zero. In this case, the occurrence of any higher order compound event $\mathbf{x}_i^r$

which contains $\mathbf{x}_j^s$ is also zero, since

$$0 \leq o_{\mathbf{x}_i^r} \leq o_{\mathbf{x}_j^s}, \qquad \mathbf{x}_i^r \supset \mathbf{x}_j^s. \tag{3.27}$$

That is to say, if it is sure (with high confidence level) that a pattern $\mathbf{x}_j^s$ is negative, all compound events containing $\mathbf{x}_j^s$ cannot be positive patterns. From the view point of inference, higher order negative pattern $\mathbf{x}_i^r$ provides no more information than $\mathbf{x}_j^s$ itself. The discovery of negatively significant pattern $\mathbf{x}_j^s$ makes the detection of its associated higher order patterns unnecessary. Hence, none of the compound events containing $\mathbf{x}_j^s$ will be examined. This is not suggesting that higher order negative patterns containing $\mathbf{x}_j^s$ can be synthesized from $\mathbf{x}_j^s$ nor that $\mathbf{x}_j^s$ can be determined as a negative pattern since there exists a higher order negative pattern containing $\mathbf{x}_j^s$. Therefore, using low order negative patterns to avoid exhaustive search does not conflict with the statement in the previous sections on the impossibility of synthesizing high order patterns from the lower order ones.

In practice, the absolute zero occurrence requirement is usually relaxed due to the existence of noise. Two common practices are proposed here. The first one is that, when we found a negative association whose occurrence is now less than the requirement of expectation for a valid statistical test, the higher order compound events will be eliminated. That is, if $\mathbf{x}_j^s$ is negatively significant and

$$o_{\mathbf{x}_j^s} \leq \omega_e, \tag{3.28}$$

the higher order compound events containing $\mathbf{x}_j^s$ will not be considered. This is because, if Eqn. 3.28 stands, all the compound events containing $\mathbf{x}_j^s$ will be either negatively significant or insignificant due to the lack of information.

The second possible criterion is more optimistic. Although there is a chance that

the higher order super-compound event of a negative pattern is positive[5], the chance is rather small. This situation happens only at high orders, and it is relatively rare. One can assume that any compound events containing a negative pattern as their sub-compound event will not be positively significant. They should be eliminated from the discovery process.

It is suggested that one makes a more conservative policy if it is known that the problem domain is almost deterministic. For a deterministic problem, events are either associated or not associated. The occurrence of an overwhelming negative association should be zero.

If $\mathbf{x}_j^s$ is positively significant, situations can be more complicated since the higher order compound events containing $\mathbf{x}_j^s$ may be positively significant, negatively significant, or insignificant. It is difficult to decide whether or not higher order events should be eliminated according to the occurrence of $\mathbf{x}_j^s$. In this case, only the requirement of expected occurrence applies.

It should be noted that, in a problem where all the lower order combinations of the primary events are uniformly distributed, this criterion will not take effect until the highest order is reached. The XOR problem is an example. Before the third order is reached, none of the patterns are detected and none of the candidates are deleted. For such problems, exhaustive search is not avoidable without domain knowledge.

An Algorithm for Candidate Selection

Combining the two techniques mentioned above, the algorithm of generating testing candidates can be summarized as follows:

[5]The occurrence of such a negative pattern has to be greater than ω_e

Notation:

N	Number of attributes in D
α_i	primary events of attribute X_i, $1 \leq i \leq N$
r	Current working order ($r \geq 2$)
C_r	Set of association candidates of order r
C'_r	Set of new candidates of order r
c_i	Any candidate in a specific candidate set
$conf$	Confidence level (e.g. 2.58 for 99%)
$threshold$	Threshold of expected occurrence
e_{c_i}	The expected occurrence of candidate c_i

Input:

r: current order

P: set of all primary events

C_r: set of candidates at order r

```
Begin Procedure gen_candidate(r, P, C_r)
    C'_{r+1} := φ                                   // new candidate set
    if C_r = φ                  // order 2, generate all pairwise combinations
        forall 1 ≤ i, j ≤ N, i ≠ j do
            forall 1 ≤ p ≤ m_i and 1 ≤ p ≤ m_j do begin
                c' := [α_i^p, α_j^q]
                C'_{r+1} := C'_{r+1} ∪ {c'}
            end
    else        // when order larger than 2, generate candidates on the fly
        forall c_i ∈ C_{r-1} do begin
            if e_{c_i} ≤ threshold          // occurrence less than threshold
```

```
                continue                    // exclude it from consideration
            else if adj_res(c_i) < -conf                // negative association
                continue                // exclude it from consideration as well
            else                                // generate higher order candidates
                forall 1 ≤ j ≤ N, j ≠ i do
                    forall 1 ≤ p ≤ m_j do begin
                        c' := c_i ∪ α_j^p
                        if c' ∉ C'_{r+1}
                            C'_{r+1} := C'_{r+1} ∪ {c'}
                        endif
                    end
            endif
        end
    endif
    Return C'_{r+1}
End Procedure
```

The algorithm *gen_candidate*() generates compound events as pattern candidates from all the primary events and a set of initial compound events which are one order lower. An empty initial candidate set indicates the beginning of the discovery process. All the possible combinations of two primary events of different attributes are generated and put into the new candidate set. When the order is greater than 2, the candidates are generated on the fly. In this case, old candidates are picked up from the initial candidate set at one time and new candidates are generated based on the picked event. For any compound event $\mathbf{x}_j^s$ (c_i in the pseudocode) of order r in the initial candidate set, it contains r primary events of r different attributes whose indices are in $\mathbf{s}$. If the expectation of c_i is less than the threshold for a valid statistical test, the expectations of all the higher order

compound event containing c_i will also be less than this threshold, hence, no higher order candidates will be generated from the compound event c_i. This heuristic corresponds to the first criterion discussed in this section. If c_i is a negatively significant compound event, no higher order candidates will be generated from c_i either. And this corresponds to the second criterion.

If none of the two criteria can be applied to candidate c_i, every primary event α_j^p which is not in c_i and j is not in s will be added to c_i to generate a new candidate one order higher. Such candidates will be put into a candidate set and output to the discovery process. Additional domain knowledge for candidate selection can also be applied before the generation of higher order candidates for more efficiency.

Fig. 3.7 shows the process of the discovery algorithm with candidate selection. In the figure, the black rectangles represent the compound events from which no more candidates will be generated. This process can be seen as a concept specialization procedure. At lower order, the significant patterns are more general compared with the higher order ones. With the order going higher, concepts (patterns) are more specific. Such a process stops when not enough samples verify the statistical significance or when further specification is not necessary. They are guarded by the two candidate selection techniques.

The criteria discussed here are general enough to be applied to any problem domain. For a particular application, however, when domain knowledge is available, more efficient heuristics may be found to truncate the searching space. Such heuristics can be applied together with those discussed earlier. For example, in a data set, if it is known that one attribute is correlated with another attribute (e.g. $A = \neg B$), we can just eliminate one of the two attributes from the discovery process. This will definitely speed up the discovery process. In the zoo database of animals, if the rule $Type = mammal \longrightarrow Backbone = yes$ applies, from a

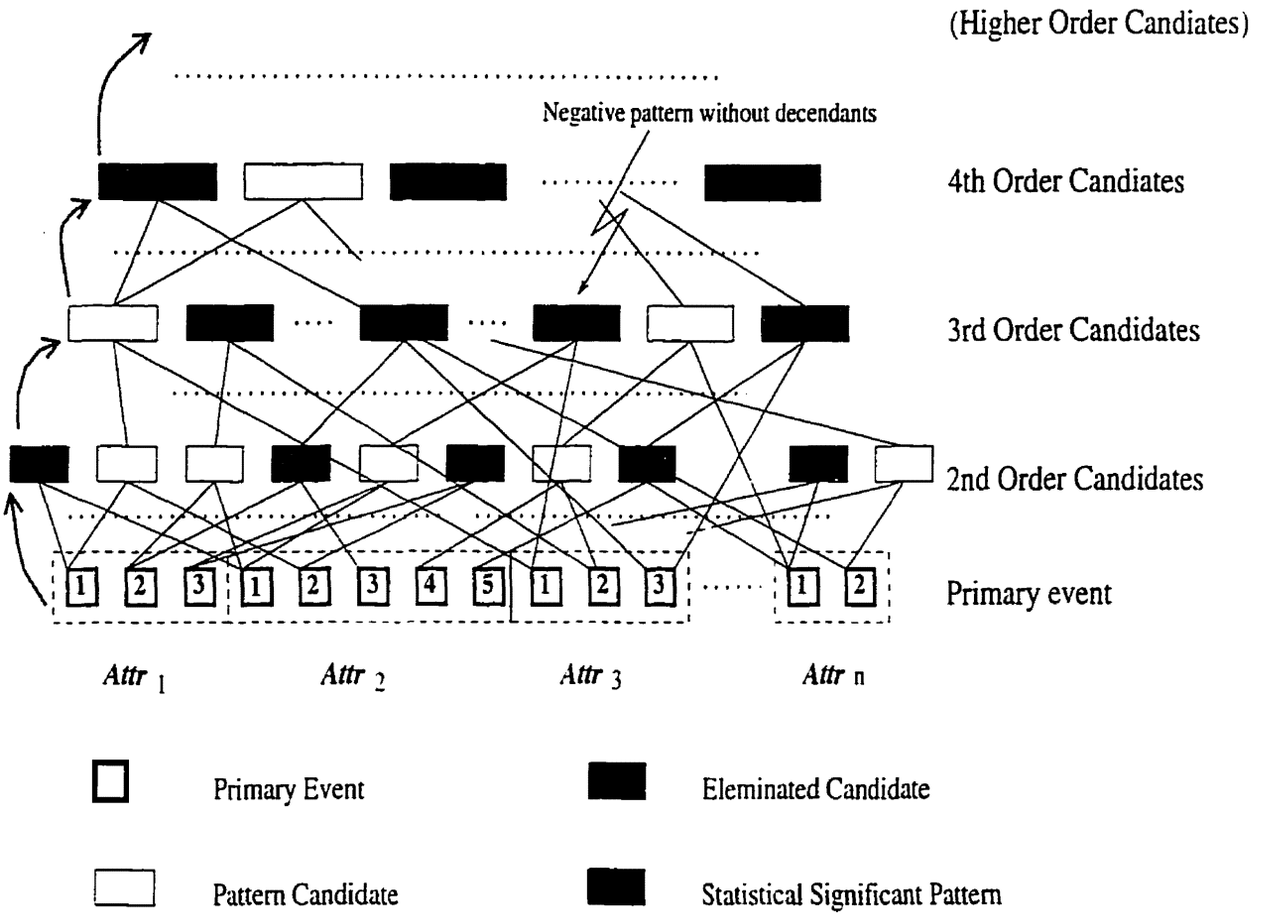

Figure 3.7: Candidate Selection in Pattern Discovery

compound event containing $Type = mammal$ as a primary event, no higher order candidate with $Backbone = yes$ will be generated. When such rules are available, the *gen_candidate*() function can accommodate this information for further truncation of the search space.

3.6.3 Analysis of Complexity

The previous section gives some practical methods for eliminating the impossible combinations at lower order before they are considered as higher order candidates.

Nevertheless, some comments on complexity in the general case are given here.

To investigate the complexity of the pattern discovery algorithm, let us consider a data set described by N m-ary attributes[6]. The total number of possible combinations is given by

$$\Theta = \sum_{k=2}^{N} \theta_k \tag{3.29}$$

where k indicates the order and θ_k denotes the total number of primary event combinations at order k. As an example, if we have 10 attributes each of which can take on one of 3 possible values, there are 346,710 possible combinations. From a practical point of view, it is quite impossible to have enough data or computational resources to manage them. Hence, techniques are developed to "prune" the set of possible pattern candidates considerably as early as possible.

At the k-th order, we have

$$\theta_k = (m)^k \cdot \begin{pmatrix} N \\ k \end{pmatrix}, \qquad 2 \leq k \leq N \tag{3.30}$$

since there are $\begin{pmatrix} N \\ k \end{pmatrix}$ sets of variables of size k and m^k possible primary events for each variable set. Let

$$\zeta_k = \frac{\theta_k}{\theta_{k-1}} = \frac{m \cdot (N - k + 1)}{k}. \tag{3.31}$$

Eqn. 3.31 gives the ratio of the number of pattern candidates of order k to the number of candidates of order k-1. When ζ_k becomes less than 1, the number of

[6]This analyzing idea was also used by [89].

higher order pattern candidates falls off rather than increases, i.e., when

$$k > \left(\frac{m}{m+1}\right) \cdot (N+1). \tag{3.32}$$

As the order of the patterns increases from $k = 2$ and upwards, the size of the search space increases. When k is small compared to N, the size increases geometrically (see Eqn. 3.31). Θ of Eqn. 3.29 can then be rewritten as

$$\Theta = \sum_{k=2}^{N} \zeta_k \cdot \theta_{k-1} = \sum_{k=2}^{N} \left(\prod_{j=1}^{k} \zeta_j\right) \tag{3.33}$$

where

$$\zeta_j = m \cdot \left(\frac{N-j+1}{j}\right)$$

The complexity of an algorithm which exhaustively searches high order patterns is $O(P) = N \cdot m^N$. If we "prune" the uninformative candidates as early as possible just as stated above, the algorithm will have complexity [89]

$$\Theta' = \sum_{k=2}^{N} \left(\prod_{j=1}^{k} \eta_j\right) \tag{3.34}$$

where $\eta_j < \zeta_j$. A practical algorithm will have $\eta_j < 1$ for k greater than a small fraction of N.

The complexity of the proposed method cannot be determined exactly since it highly depends upon the nature of the input data. The complexity analysis above is for the worst case of this method. For data sets in which only the lower order relations exist, the proposed method will stop searching for higher order candidates. Generally, exhaustive searching is not necessary. Only in the worst scenario where all the highest order patterns have to be detected is exhaustive search unavoidable.

For example, the worst case occurs for the pathological case of a set of N binary attributes whose N-th order joint distribution is entirely uniform [103], i.e. all transition probabilities are equal to 0.5. In this case, none of the pattern candidates are significant from their expectation. In the XOR problem, for example, before the third order is reached, no patterns can be found. The lack of quantitative results on the complexity of the proposed method reflects the well-known inherent difficulty in quantifying the complexity of "open-ended" learning problems [89].

A probabilistic analysis based on the average performance over all possible input data sets is too difficult to carry out without invoking unrealistic assumptions concerning the nature of the input data sets. The best that can be done is to invoke the argument that as pattern order increases, the simplicity of the hypothesis on a compound event to be a pattern decreases to the extent that their probability of occurrence is very small. With real-world problems, it is believed that significant associations are sparsely scattered rather than uniformly distributed in the hypothesis space, especially when the order is high. Since insignificant patterns are eliminated at early stages, only a subset of all the pattern candidates is under consideration. Thus, the smaller the set is, the more efficient the algorithm will be.

In practice, M, the number of instances in the database, also influences the computational complexity. M imposes an upper-bound on the order of candidate patterns to be detected, and therefore prevents the algorithm from going to high order cases where testing is no longer statistically valid [103]. Since the residual analysis used here is originally adopted from a contingency table, a bound of the order given a fixed sample size can be computed by considering the contingency table for the same variables which constitute a certain compound event. One should bear in mind that the sample size should be large enough to keep the assumption of normal distribution valid for the testing cell. A safe estimate of this sample size

is

$$M \geq \omega_e \cdot \prod_{i \in s} m_i. \tag{3.35}$$

where m_i is the alphabet cardinality of the variable X_i which is involved in the compound event $\mathbf{x}_j^s$. ω_e is the minimum requirement of the expected occurrence of a cell for a valid significant test. This implies that, for a reliable statistical test of a k-th order compound event, the sample size should be larger than the product of the total number of cells in the corresponding contingency table and the lower bound of the expected occurrence. Therefore, the complexity of the proposed algorithm turns out to be:

$$\Theta' = \sum_{k=2}^{K} \left(\prod_{j=1}^{k} \eta_j \right) \tag{3.36}$$

where η_j has the same definition as that in Eqn. 3.34, and K is the highest order bounded by M, $K \leq N$.

Basically, the search in the proposed method can be seen as a "*test-and-discard*" search. It searches the hypothesis space of patterns for statistically significant event association. If each possible compound event is a point in the searching space, the algorithm, in the worst case, visits each point once. When the elimination criteria are applied, the searching space is pruned. These techniques work well especially when the significant event associations are sparsely scattered in the hypothesis space.

Chapter 4

Pattern Analysis and Inference

The pattern discovery algorithm discussed in the previous chapter can detect numerous significant event associations inherent in a data set and render an attributed hypergraph as the structure of the discovered patterns. But when this system is used to analyze a specific problem domain, not all the patterns are interesting at a particular time. The discovery of interesting patterns of data is essentially an egocentric activity. What is interesting to one user may or may not be interesting to the others. For example, the user may be interested in the most important associations in the data set instead of all the patterns, or the user needs to determine the attribute value of a new observed instance. To undertake these tasks, the discovered patterns are sorted, grouped and/or picked up for further analysis according to the user's queries or constraints. Such queries and constraints come from the user's specific interest and knowledge of the domain. One example of domain knowledge is the concept hierarchies as used in DBLearn [13] [49] [50]. When such knowledge is not available, user's interest is usually represented as the selection of specific attributes to which the patterns are related.

In this chapter, two common tasks in data analysis and database mining are examined. They are the *Best-N* problem and the *Missing-Value* problem. The Best-N problem regards the selection of the most informative association rules from the patterns found, while the missing-value problem is a general case of classification – to predict the most plausible value of an attribute provided with a set of observations. The best N rules may be selected from all patterns regarding the entire domain according to a certain information measure, or they can be confined to a sub-domain characterized by a set of attributes selected by the user as his interest. In the missing-value problem, any attribute of the domain can be predicted. Thus, which attribute value is going to be predicted is determined by the current goal of the analysis. This goal may change from time to time.

To solve these two problems, an information measure for the discovered patterns is needed. In this chapter, the two problems are first defined. Then, an information measure – the weight of evidence – is proposed to evaluate the evidence provided by a set of primary events in support of, or against, another attribute taking on a certain value. Presented immediately after that are the algorithms to solve the two problems.

4.1 The Best-N Problem and the Missing-Value Problem

4.1.1 The Best-N Problem

The best-N problem is to extract N (association) rules which describe the problem domain the best. Sometimes, it is also referred to as rule induction [89]. The N

rules can be related to any attributes in the domain, or as required by the user, be related to a certain subset of the domain. The number of association rules to be extracted is also specified by the user.

The best-N problem is closely related to the pattern discovery process. Once all the significant associations inherent in a data set have been uncovered, the best-N problem is relatively easy to solve. With an attributed hypergraph $\mathbf{A}$ representing the discovered patterns, the *best-N* problem can be formalized as to define a decision function $f(\cdot)$ to select a set $\mathbf{R}$ of $N(N \geq 0)$ association rules from $\mathbf{A}$ such that they are the most informative. That is:

$$\mathbf{R} = f(\ \mathbf{A},\ N\ ). \tag{4.1}$$

where $\mathbf{R} = \{r_i | 0 \leq i \leq N\}$ is a set of association rules. According to Agrawal, et al [3], an association rule has an implication of the form:

$$r_i : X \Rightarrow Y \tag{4.2}$$

where X, the left-hand condition, is a set of primary events connected by logic AND; Y is a primary event, always called the right-hand consequence; and $X \bigcap Y = \phi$. Generally speaking, Y can contain more than one primary event, but we confine the discussion within the constraint that Y has only one primary event. In real-world problems, since an association rule is generally probabilistic, there are always probabilistic measures attached to the rule to show its strength.

To reflect the specific interest of a user, other constraints can be posed by the user in terms of the right-hand or left-hand attributes/events or the number of left-hand propositions are imposed by the user. For example, the marketing manager of a car manufacturer may only be interested in the best N sales patterns regarding

compact cars with automatic transmission. Then, all rules for him should contain the events *Size* = *compact* and *Transmission* = *auto*. By providing these kinds of constraints, the user will obtain the patterns that interest him the most. In this way, an egocentric analysis of data is achieved. If such constraints are posed, the decision function $f(\cdot)$ has one more argument:

$$\mathbf{R} = f(\ \mathbf{A},\ N,\ \mathbf{C}\ ). \tag{4.3}$$

where $\mathbf{C}$ is a set of constraints.

4.1.2 The Missing-Value Problem

The missing-value problem is a generalization of the classification problem. To classify an object is to determine the class label to which the object belongs. Basically, the class of an object is not previously defined by an external teacher. Any attribute can be considered as the class attribute, depending upon the current interest. If the class attribute is categorical, the problem is always called a classification problem. To determine the class label of an object is to find the most plausible value of an attribute (the class) for this object. The value of such an attribute is missing or unknown, but the domain of this attribute has already been defined within the problem or the data set. Hence, the classification (reasoning, prediction) task can be generalized as solving the *missing-value* problem - the problem of recovering missing values in a set of noisy discrete multivariate data.

The missing-value problem can be formalized as determining the value of a missing attribute, X_i, given a set of observations, $x_{1+}, \cdots, x_{(i-1)+}, x_{(i+1)+}, \cdots, x_{q+}$, where $x_{j+} \in \alpha_j$, $1 \leq j \leq q$ is any possible value of attribute X_j. Let $(\underline{\mathbf{X}}, Y)$ be jointly distributed random variables with q-dimensional vector $\underline{\mathbf{X}}$ denoting a feature

(observation) vector and Y denoting the attribute whose value is to be determined. The missing-value problem is to find a decision rule $d(\cdot)$ that maps R^q into the domain of Y such that certain properties of the original data set are preserved. The feature vector $\underline{\mathbf{X}}$ is called a new *observation* or a new *object* and Y its class label or predicting attribute. It is assumed *a priori* that attribute Y is discrete. In traditional classification, or supervised learning, Y is a special pre-defined attribute called "class" while in unsupervised learning or for flexible prediction, Y can be any attribute describing the problem domain.

Of all the significant associations discovered from the data set, only those which are associated with the predicting attribute Y and the observation $\underline{\mathbf{X}}$ are useful in determining a plausible value of Y. The insignificant compound events are considered statistically independent with the predicting attribute under the default independence model. The problem here is how to define the decision function $d(\cdot)$ given an observation $\underline{\mathbf{X}}$, a predicting variable Y and an attributed hypergraph $\mathbf{A}_Y$ (of Y), representing the set of significant associations related to Y, such that

$$Y = d(\ \underline{\mathbf{X}},\ \mathbf{A}_Y\ ). \tag{4.4}$$

The selection of attribute to be predicted is based on the interest of the user or the current goal of the system. With a changing goal, the attribute to be predicted changes, but there is no need to re-learn the domain as long as all the significant patterns have been discovered.

4.2 Weight of Evidence

The best-N problem and the missing-value problem can be generalized as a problem of finding a decision function. Such a decision function evaluates the strength provided by a set of primary events to another attribute taking on a certain value in its domain against the others. Since only the significant associations are interesting and other compound events are considered uninformative, there is no need to go back to the original data set. The attributed hypergraph generated from the original data in the discovery process acts as a knowledge base for the inference tasks.

Osteyee's uncertainty measure [74], weight of evidence, which was originally defined for applications involving two variables, is extended here. Weight of evidence evaluates the plausibility of Y taking on a value y_i against other values.

Suppose $\mathbf{x}_j^s$ is a significant association in $\mathbf{A}$, containing a primary y_i of Y. It can always be rewritten in the form of $(\underline{\mathbf{x}}, y_i)$, where $\underline{\mathbf{x}} = \mathbf{x}_j^s - y_i$ is a sub-compound event of $\mathbf{x}_j^s$, and $y_i \in \mathbf{x}_j^s$ is the i-th primary event of attribute Y. The amount of evidence provided by $\underline{\mathbf{x}}$ for y_i being a plausible value of Y can be quantitatively estimated by an evidence measure which is derived from an information-theoretic measure known as the *mutual information* [74] [101]:

$$I(Y = y_i : \underline{\mathbf{x}}) = \log \frac{Pr(Y = y_i \mid \underline{\mathbf{x}})}{Pr(Y = y_i)}. \tag{4.5}$$

The mutual information measure is positive if and only if $Pr(Y = y_i \mid \underline{\mathbf{x}}) > Pr(Y = y_i)$, otherwise it is either negative or has a value of zero. $I(Y = y_i : \underline{\mathbf{x}})$ intuitively measures the decrease (if positive) or increase (if negative) in uncertainty about Y taking on the value y_i given $\underline{\mathbf{x}}$.

Based on the mutual information, the difference in the gain of information when

Y takes on the value y_i and when it takes on some other values, given $\underline{\mathbf{x}}$, is a measure of evidence provided by $\underline{\mathbf{x}}$ in favor of y_i being a plausible value of Y as opposed to other values. This difference, denoted by $W(Y = y_i/Y \neq y_i|\underline{\mathbf{x}})$, is defined as the *weight of evidence*, which has the following form:

$$\begin{aligned} W(Y = y_i/Y \neq y_i|\underline{\mathbf{x}}) &= I(Y = y_i : \underline{\mathbf{x}}) - I(Y \neq y_i : \underline{\mathbf{x}}) && (4.6) \\ &= \log \frac{Pr(Y = y_i \mid \underline{\mathbf{x}})}{Pr(Y = y_i)} - \log \frac{Pr(Y \neq y_i \mid \underline{\mathbf{x}})}{Pr(Y \neq y_i)} && (4.7) \end{aligned}$$

The weight of evidence is positive if $\underline{\mathbf{x}}$ provides positive evidence supporting Y taking on y_i, otherwise, it is negative or zero. A negative weight of evidence implies that there is negative evidence provided by $\underline{\mathbf{x}}$ against Y taking on the value y_i. In other words, it is more likely for this attribute to take on another value. A zero weight of evidence suggests that $\underline{\mathbf{x}}$ is irrelevant to the prediction of Y.

By applying Bayes formula, Eqn. 4.6 can be rewritten equivalently as:

$$\begin{aligned} & W(Y = y_i/Y \neq y_i|\underline{\mathbf{x}}) \\ &= \log \frac{Pr(\underline{\mathbf{x}} \mid Y = y_i)}{Pr(\underline{\mathbf{x}})} - \log \frac{Pr(\underline{\mathbf{x}} \mid Y \neq y_i)}{Pr(\underline{\mathbf{x}})} \\ &= \log \frac{Pr(\underline{\mathbf{x}} \mid Y = y_i)}{Pr(\underline{\mathbf{x}} \mid Y \neq y_i)} && (4.8) \\ &= \log \frac{Pr(\underline{\mathbf{x}}, Y = y_i) \cdot (1 - Pr(Y = y_i))}{Pr(Y = y_i) \cdot (Pr(\underline{\mathbf{x}}) - Pr(\underline{\mathbf{x}}, Y = y_i))} && (4.9) \end{aligned}$$

where $Pr(\underline{\mathbf{x}}, Y = y_i) = Pr(\mathbf{x}_j^s)$, which is the probability of the significant association $\mathbf{x}_j^s$.

As an illustration, let us look at this example. Suppose in the attributed hypergraph derived from a zoo database there is a hyperedge $[Hair = yes,\ Feathers = no,\ Backbone = yes,\ Type = mammal]$. If we elect to decide on the evidence pro-

vided by observation $\{Hair = yes,\ Feathers = no,\ Backbone = yes\}$ in support of $Type$ taking on value $mammal$, the weight of evidence can be calculated. If the probabilities of the events $[Hair = yes,\ Feathers = no,\ Backbone = yes,\ Type = mammal]$, $[Hair = yes,\ Feathers = no,\ Backbone = yes]$ and $[Type = mammal]$ are 0.386, 0.426 and 0.406 respectively, the weight of evidence is:

$$
\begin{aligned}
& W(Type = mammal/Type \neq mammal | Hair = yes, Feathers = no, Backbone = yes) \\
= \ & \log \frac{0.386 \times (1 - 0.406)}{0.406 \times (0.426 - 0.386)} \\
= \ & 3.82
\end{aligned}
$$

This weight of evidence is positive. It means that this animal with hair, backbone and no feathers is likely to be a mammal. Weight of evidence is not always positive. From the same database, the hyperedge $[Hair = no,\ Backbone = yes,\ Venomous = no,\ Type = mammal]$ provides negative weight of evidence of -3.70 with respect to $[Type = mammal]$. It implies that it is unlikely for an animal which is hairless, not venomous and has a backbone to be a mammal.

4.3 Finding the Best N Associations

The best-N problem is relatively easy to solve when all the significant event associations in the data set have been discovered. The remaining problem is to transfer a hyperedge into an association rule and measure the weight of the rule for comparison with the other rules. Since the best-N problem does not deal with new observations, the attributed hypergraph generated in the pattern discovery process is the one to be analyzed.

Suppose there is a hyperedge $\mathbf{x} = [x_{1+}, x_{2+}, \cdots, x_{r+}]$ of order r. Putting each of

the primary events at the right-hand side of an association rule and the remaining at the left-hand side, r rules can be generated in the form of:

$$\bigwedge_{i=1,\, i\neq j}^{r} (X_i = x_{i+}) \;\Rightarrow\; (X_j = x_{j+}) \text{ with a quantitative measure, } 1 \leq j \leq r$$

The weight of evidence can be used to measure the strength of such a rule. The weight of evidence in support of X_j taking on the value x_{j+} given $\{x_{i+}|1 \leq i \leq r, i \neq j\}$, $W(X_j = x_{j+}/X_j \neq x_{j+}|x_{i+}, 1 \leq i \leq r, i \neq j)$ can be calculated by Eqn. 4.9. Recall that the weight of evidence is the gain of information when X_j takes on the value x_{j+} and when it takes on the other values, given $\{x_{i+}|1 \leq i \leq r, i \neq j\}$. It can be interpreted as a measure of distance between x_{j+} and other possible values of X_j, provided that $\{x_{i+}|1 \leq i \leq r, i \neq j\}$ happens. This distance shows the strength of the association between the right-hand side event and the left-hand side conditions. Unlike normal distance, the larger the weight of evidence, the shorter the distance is. A zero weight of evidence means an equal distance between x_{j+} and $\neg x_{j+}$ given the observations.

Two cases need special explanations. The first case is when the weight of evidence is positive infinity. In this case, the right-hand side event will certainly happen if the left-hand side events are observed. The other case is just the opposite. When the weight of evidence is equal to negative infinity, the right-hand side event will never happen if the left-hand side condition stands. In the case of a negative weight of evidence, the association rule can be re-written as:

$$\bigwedge_{i=1,\, i\neq j}^{r} (X_i = x_{i+}) \;\Rightarrow\; (X_j \neq x_{j+}) \text{ with } WOE = |W_j|, \;\; 1 \leq j \leq r$$

With this notation, the absolute values of the weights of evidence can be used to

show the strength of the rules.

Once an attributed hypergraph representing the significant event associations is available, the algorithm to solve the best-N problem can be defined. This algorithm goes through the hypergraph and generates N association rules, where N is a user-defined parameter. The set of N rules are the N most significant rules from the original data as measured by the weight of evidence. The probabilities required for calculating the weight of evidence are assumed to be stored as the attributes of the hyperedge.

The algorithm proceeds by processing one hyperedge at one time. For each hyperedge of order r, r rules are generated and the weights of evidence for each rule are calculated. The first N rules are sorted and put into a list. The smallest weight of evidence, that of the N-th element of the list, is then defined as the running minimum W_{min}. From that point onwards, new rules which are candidates for inclusion in the rule list have their weights of evidence compared with W_{min}. If their weights of evidence are greater than W_{min}, they are added to the list; the N-th rule is deleted, and the W_{min} is updated with the weight of evidence of the current N-th rule in the list. This process continues until no hyperedges are left in the attributed hypergraph. The algorithm is summarized as follows:

Notation:

$\mathcal{A}$	AHG representing all significant event associations
$\mathbf{x}$	A hyperedge (pattern) in $\mathcal{A}$
r	Order of a
x_{j+}	A primary event in $\mathbf{x}$, $1 \leq j \leq r$
W_j	Weight of evidence of $\mathbf{x} - x_{j+}$ w.r.t. x_{j+}

R_j	Rule $[\mathbf{x}\backslash x_{j+} \Rightarrow (\neq)x_{j+}$ with $WOE = \|W_j\|]$
N	User defined parameter
$\mathbf{R}$	Set of rules sorted by $\|W\|$, $0 \leq \|\mathbf{R}\| \leq N$
W_{min}	The absolute value of the minimum weight of evidence in $\mathbf{R}$
n	Current number of rules in $\mathbf{R}$

```
Begin Procedure best_N( A, N )
    R = φ                                          // initialize R
    n := 0                                         // initialize n
    forall x ∈ A do begin                          // go through every hyperedge
        for 1 ≤ j ≤ r do begin                     // r rules for x
            W_j := W(x_j+/ ≠ x_j+|x − x_j+)        // calculate WOE
            if n < N do begin
                R := R + R_j                       // insert new rule to the sorted list
                n++
                W_min = W_n                        // update W_min
            else if |W_j| > W_min                  // a new strong rule
                R := R − R_n                       // delete the lest strong rule
                R := R + R_j                       // insert new rule
            end if
        end for
    end forall
    output R                                       // the result
End Procedure
```

Basically, this algorithm will generate

$$H = \sum_{r=2}^{m} r \cdot H_r \tag{4.10}$$

rules, where H_r is the number of hyperedges of order r and m is the highest order

in the attribute hypergraph $\mathcal{A}$. N rules are selected according to the weight of evidence if $N \leq H$.

This general description of the algorithm given above is not intended as a definitive description of how the algorithm should be implemented. Actual implementation depends on the specific problem and the system, such as the choice of data structure, etc.

As stated before, in the applications of data analysis and data mining, certain constraints on the association rules are usually applied according to the user's interest. Such constraints are normally imposed to restrict the choice of the right-hand side and the left-hand side events. For example, the user may only want to look at the events that lead to a specific set of consequences. In this case, one can restrict the right-hand side consequence to that subset of interest. If a hyperedge does not contain a primary event in the interesting subset, no rules will be generated from that hyperedge. This situation is analogue to classification tasks, in which only the propositions in the event space of a single variable, the class label, are interesting. The rules generated in this way are always called classification rules. Another example is that the user is interested in the consequence of a set of known conditions, or the consequence of an interesting proposition when given a set of conditions. Forecasting tasks can be formulated in this way. One can make constraints on both the right-hand side and the left-hand side propositions when selecting the rules.

Once the significant event associations have been discovered, it is easy to solve the best-N problem with the proposed algorithm. But a more challenging task in the inference is classification. For a new object described by a set of primary events, some of the primary events may be significantly associated with a class, while some other may not. To classify such an object, one has to consider this problem.

4.4 Classifying a New Object

As stated earlier, when an observation is made, a set of primary events are available. The user may require the system to determine the most possible value of an unobserved attribute, or, in other words, to classify the new observation against a certain attribute. If the observation $\underline{\mathbf{X}}$ and a probable value of the predicting attribute Y, y_i, together form a significant association $(\underline{\mathbf{X}}, y_i)$, the weight of evidence of $\underline{\mathbf{X}}$ on y_i can be easily computed. The only problem is that, some of the primary events in $\underline{\mathbf{X}}$ are important for determining the value of Y, some others may be irrelevant. In the discovery process, it is not always possible to find significant associations whose order is as high as $(\underline{\mathbf{X}}, y_i)$. More likely, some compound events which are sub-compound events of $(\underline{\mathbf{X}}, y_i)$ are significant, while some other sub-compound events are not significant. For inference, it is quite possible that more than one statistically significant compound event is associated with one of Y's possible values. Some of them may provide evidence supporting Y to take on y_i whereas others may provide evidence against it. There is a need for an uncertainty measure to quantitatively combine and compare the positive and negative evidence provided by the observations so as to classify the new object.

Similar to Osteyee's definition [74] but extended to cover multiple variables, a more general weight of evidence measure is defined. It measures the evidence provided by observation $\underline{\mathbf{X}} = (x_{1+}, \cdots, x_{(j-1)+}, x_{(j+1)+}, \cdots, x_{L+})$ in favor of $Y(X_j)$ taking on the value $y_i(x_{ji})$ as opposed to the other values:

$$\begin{aligned} & W(Y = y_i / Y \neq y_i \mid x_{1+}, \cdots, x_{(j-1)+}, x_{(j+1)+}, \cdots, x_{L+}) \\ = \; & \log \frac{O(X_j = x_{ji} / X_j \neq x_{ji} \mid x_{1+}, \cdots, x_{(j-1)+}, x_{(j+1)+}, \cdots, x_{L+})}{O(Y = y_i / Y \neq y_i)} \end{aligned} \tag{4.11}$$

$$= \log \frac{O(Y = y_i / y \neq y_i \mid \underline{\mathbf{X}})}{O(Y = y_i / Y \neq y_i)} \tag{4.12}$$

where $O(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}})$ is the odds in favor of Y taking on the value y_i as opposed to the other values given that $\underline{\mathbf{X}}$ is characterized by $x_{1+}, \cdots, x_{(j-1)+}, x_{(j+1)+}, \cdots, x_{L+}$ and is defined as:

$$O(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}) = \frac{Pr(Y = y_i \mid \underline{\mathbf{X}})}{Pr(Y \neq y_i \mid \underline{\mathbf{X}})}. \tag{4.13}$$

and $O(Y = y_i / y \neq y_i)$ is the odds without the observation:

$$O(Y = y_i / Y \neq yi) = \frac{Pr(Y = y_i)}{Pr(Y \neq y_i)}. \tag{4.14}$$

The weight of evidence W can therefore be rewritten as:

$$W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}) = \log \frac{Pr(Y = y_i \mid \underline{\mathbf{X}})}{Pr(Y = y_i)} - \log \frac{Pr(Y \neq y_i \mid \underline{\mathbf{X}})}{Pr(Y \neq y_i)}, \tag{4.15}$$

or equivalently:

$$\begin{aligned} W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}) &= \log \frac{Pr(\underline{\mathbf{X}} \mid Y = y_i)}{Pr(\underline{\mathbf{X}})} - \log \frac{Pr(\underline{\mathbf{X}} \mid Y \neq y_i)}{Pr(\underline{\mathbf{X}})} \\ &= \log \frac{Pr(\underline{\mathbf{X}} \mid Y = y_i)}{Pr(\underline{\mathbf{X}} \mid Y \neq y_i)}. \end{aligned} \tag{4.16}$$

Eqn. 4.16 is quite similar to the previous definition of weight of evidence illustrated by Eqn. 4.9. The difference is that, in Eqn. 4.9, $(\underline{\mathbf{x}}, y_i)$ is a significant association, while in Eqn. 4.16, such constraint is released.

In the first order inference, only the relations between the predicting event $Y = y_i$ and each primary event in $\underline{\mathbf{X}}$ are examined. Under the assumption of

conditional independence [17], Eqn. 4.16 becomes:

$$
\begin{aligned}
& W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}) \\
= \; & \log \frac{Pr(X_1 = x_{1+} \mid Y = y_i)}{Pr(X_1 = x_{1+} \mid Y \neq y_i)} + \cdots + \log \frac{Pr(X_{j-1} = x_{(j-1)+} \mid Y = y_i)}{Pr(X_{j-1} = x_{(j-1)+} \mid Y \neq y_i)} \\
& + \log \frac{Pr(X_{j+1} = x_{(j+1)+} \mid Y = y_i)}{Pr(X_{j+1} = x_{(j+1)+} \mid Y \neq y_i)} + \cdots + \log \frac{Pr(X_L = x_{L+} \mid Y = y_i)}{Pr(X_L = x_{L+} \mid Y \neq y_i)} \\
= \; & W(Y = y_i / Y \neq y_i \mid X_1 = x_{1+}) + \cdots + W(Y = y_i / Y \neq y_i \mid X_{j-1} = x_{(j-1)+}) \\
& + W(Y = y_i / Y \neq y_i \mid X_{j+1} = x_{(j+1)+}) + \cdots + W(Y = y_i / Y \neq y_i \mid X_L = x_{L+}) \\
= \; & \sum_{l=1, l \neq j}^{L} W(Y = y_i / Y \neq y + i \mid X_l = x_{l+}). \qquad (4.17)
\end{aligned}
$$

But for high order inference, this decomposition is no longer valid. For example, two overlapped compound events $[A = a, B = b]$ and $[B = b, C = c]$ cannot be assumed independent conditioned by a fourth variable, say $D = d$. However, if two subsets of $\underline{\mathbf{X}}$, say $\underline{\mathbf{X}}_1$ and $\underline{\mathbf{X}}_2$, have no intersection, the assumption that $\underline{\mathbf{X}}_1$ and $\underline{\mathbf{X}}_2$ are conditionally independent on $Y = y_i$ can still be made. Under such an assumption, the weight of evidence in Eqn. 4.15 is extended to high orders and can be expressed as:

$$
\begin{aligned}
& W(Y = y_i / y \neq y_i \mid \underline{\mathbf{X}}) \\
= \; & \log \frac{Pr(\underline{\mathbf{X}}_1 \mid Y = y_i)}{Pr(\underline{\mathbf{X}}_1 \mid Y \neq y_i)} + \cdots + \frac{Pr(\underline{\mathbf{X}}_n \mid Y = y_i)}{Pr(\underline{\mathbf{X}}_n \mid Y \neq y_i)} \\
= \; & W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}_1) + \cdots + W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}_n) \\
= \; & \sum_{k=1}^{n} W(Y = y_i / Y \neq y_i \mid \underline{\mathbf{X}}_k) \qquad (4.18)
\end{aligned}
$$

where $\underline{\mathbf{X}}_k$ is a sub-compound event of $\underline{\mathbf{X}}$ and satisfies:

$$
\underline{\mathbf{X}}_p \bigcap \underline{\mathbf{X}}_q = \phi, \quad p \neq q, 1 \leq p, q \leq n \text{ and}
$$

$$\bigcup_{p=1}^{n} \underline{\mathbf{X}}_p = \underline{\mathbf{X}}$$

Based on [17], events which are not statistically significant can be considered irrelevant for the inference process. The underlining model used in the pattern discovery process assumes that the attributes are mutually independent. This implies that, if a compound event is not statistically significant, the primary events in it are randomly combined. The mutual information (Eqn. 4.5) is approximately zero. When calculating the weight of evidence, these events can be eliminated. Thus, only the significant event associations discovered from the data set are used in the inference process. Then the calculation of weight of evidence is to find a proper set of disjoint significant event associations from $\underline{\mathbf{X}}$ and to sum each individual weight of evidence provided by the sub-compound event. That is to maximize

$$W(Y = y_i/Y \neq y_i \mid \underline{\mathbf{X}}) = \sum_{q=1}^{m} W(Y = y_i/Y \neq y_i \mid \underline{\mathbf{X}}_q) \tag{4.19}$$

with sub-compound events $\underline{\mathbf{X}}_1, \cdots, \underline{\mathbf{X}}_m$, such that $(\underline{\mathbf{X}}_q, Y = y_i)$ $(1 \leq q \leq m)$ is a significant event association, and the intersections between two different sub-compound events are empty. Using significant association patterns in the inference process makes it possible not to go back to the original data set.

In practice, it may be time-consuming to exhaustively try all combinations of the proper set of disjoint significant sub-compound events to maximize Eqn. 4.19[1]. Heuristics may apply. Here, it is assumed that statistically significant compound events of higher orders describe the properties of the domain more accurately and more specifically than those of lower order events. In calculating the weight of

[1]It is not impractical to try all combinations since the number of significant associations related to y_i and subset by $\underline{\mathbf{X}}$ may not be too large because a domain either has limited number of attributes or the user manually binds the highest order in the discovery process.

evidence given an observation, the highest order significant compound event in the observation should be considered first. To avoid re-consideration of a single value, the primary events should be eliminated from the observations after they make their contribution to the entire weight of evidence. The following algorithm shows an approach to calculating the weight of evidence in support of the predicting attribute Y taking on value y_i given an observation $\underline{\mathbf{X}}$ and a set of significant event associations $\mathcal{A}$ discovered from the original data set.

Notation:

$\mathcal{A}$	All significant event associations discovered from D
$\mathcal{A}'$	Set of significant associations with the same order
a	The association in $\mathcal{A}'$ with the highest adjusted residual
$\underline{\mathbf{X}}$	An observation
y_i	A possible value of the predicting attribute Y
$\underline{\mathbf{X}}'$	Set of uncovered primary events
W	Weight of evidence
W_q	Weight of evidence of $\underline{\mathbf{X}}_q$ w.r.t. y_i

```
Begin Procedure weight_of_evidence( A, X, y_i )
    X' := X                                          // initialize X'
    W := 0                                           // initialize WOE
    while X' ≠ φ do begin                            // there are uncovered primary events
        A' := extract_assoc(A, X', y_i)              // find the highest order associations
        if A' = φ                                    // no association left
            X' := φ
        else do begin
            X_q := a \ {y_i}                         // association with highest absolute
```

```
                                                        // adjusted residual
            W_q := single_WOE( X_q, y_i )     // WOE of a significant assoc.
            W := W + W_q
            X' = X' \ X_q                     // exclude covered primary events
        end else
    end while
    output W                                            // the result
End Procedure
```

In the above algorithm, $extract\_assoc(\mathcal{A}, \underline{\mathbf{X}}', y_i)$ is called to extract from $\mathcal{A}$ the highest order associations which contain y_i and a subset of $\underline{\mathbf{X}}'$. To calculate the weight of evidence of a single significant association, function $single\_WOE(\underline{\mathbf{X}}_q, y_i)$ utilizes Eqn. 4.9.

If the predicting attribute Y has its domain $\{y_i \mid 1 \leq i \leq m_Y, m_Y < \infty\}$, when an observation is available, the most plausible value of Y is the one with the highest weight of evidence provided by the observation. If Y is binary, two weights of evidence will be calculated and compared. Let W_1 be the weight of evidence provided by observation $\underline{\mathbf{X}}$ in support of y_1 and W_2 in support of y_2. If $W_1 > W_2$ and $W_1 > 0$, y_1 will be a more plausible value of Y than y_2. We say that $\underline{\mathbf{X}}$ provides more evidence to support Y to take on the value y_1 against y_2.

The classification process based on weight of evidence can be summarized as follows. A set of primary events $\underline{\mathbf{X}}$ are observed. The value of an unobserved attribute Y is going to be determined with the significant event associations discovered from the training data set. Given a set of significant associations related to attribute Y, the weight of evidence for each possible value of Y provided by the observation is calculated applying the algorithm described in the previous sub-section. These weights are compared to find the most plausible value of Y. The value y_i can be

considered as the most plausible value if the following conditions stand:

$$\begin{cases} W(Y = y_i/Y \neq y_i \mid \underline{\mathbf{X}}) & > & W(Y = y_j/Y \neq y_j \mid \underline{\mathbf{X}}) \\ W(Y = y_i/Y \neq y_i \mid \underline{\mathbf{X}}) & > & 0 \end{cases} \tag{4.20}$$

where $1 \leq j \leq m_Y$ and $j \neq i$. As defined earlier. m_Y denotes the the number of possible values Y can take on.

Several special cases are worthy of further discussion. The first is that it is possible for two (or more) different values of Y to have similar greatest weight of evidence. In this case. one can only conclude that the evidence provided by the observation suggests that Y may have either one of the two (or more) values, or there is not enough evidence to distinguish them. If no more information is available. neither value will be suggested to avoid an inaccurate conclusion. Another possible approach is to assign Y to the value with larger marginal probability in the training data set. In the second case. the absolute value of the weights of evidence are very close to zero. This phenomenon suggests that the attribute Y is either random or more observations should be made to determine the value of Y. If Y is believed to be random in this case. it can be assigned with the most frequent value in the data set. Otherwise, no conclusion should be made. A third case is when the maximum weight of evidence is negative. It implies that none of the values of Y is suitable for the new observation. Two causes may result in such a situation. One is that the previous domain of Y does not cover all the possible values. An unknown value is missing both in the data set and the domain description. In such a case, data must be re-collected and discovery process re-conducted. The other possibility is that the observation $\underline{\mathbf{X}}$ is not enough. In either situations, no conclusion should be made to the value of attribute Y.

The algorithm of classification is illustrated by the following pseudocode:

Notation:

$\mathcal{A}$	All significant event associations discovered from D
$\underline{\mathbf{X}}$	An observation
Y	Predicting attribute
y_i	A possible value of the predicting attribute Y
W	$\{W_i \mid 1 \leq i \leq \alpha_Y\}$
W_i	Weight of evidence for y_i
α_Y	Number of possible values of attribute Y

```
Begin Procedure classify(X, Y, A)
    forall i : 1 ≤ i ≤ α_Y                          // calculate all WOE
        W_i = weight_of_evidence( A, X, y_i )
    if W_i < 0, 1 ≤ i ≤ α_Y                         // all WOE negative
        output NULL
    if two or more WOE are similar                  // similar WOE
        output NULL      // or the value with the highest marginal prob.
    if W_i ≈ 0, 1 ≤ i ≤ α_Y                         // all WOE close to zero
        output NULL      // or the value with the highest marginal prob.
    i = select_WOE(W)                               // find the index to the highest WOE
    output y_i
End Procedure
```

In the above algorithm, function $select\_WOE(W)$ returns the index to the greatest weight of evidence in W.

Let us go back to the zoo database for an example. If an animal has four legs, a tail, a backbone and teeth; it is neither aquatic nor airborne, and it is predatory and lays eggs. We would like to know to which type this animal belongs. This problem

can be approached as that we have the observations $\{Eggs = yes, Aquatic = no, Predator = yes, Toothed = yes, Legs = four, Tail = yes\}$ and that the attribute $Type$ is missing. Suppose we also have the attributed hypergraph which represents the associations of the training data set, the following weights of evidence are calculated.

$$
\begin{aligned}
& W(Type = mammal/Type \neq mammal | Eggs = yes, Aquatic = no, \\
& \quad Predator = yes, Toothed = yes, Legs = four, Tail = yes) \\
= \; & W(Type = mammal/Type \neq mammal | Aquatic = no, Toothed = yes, \\
& \quad Legs = four, Tail = yes) + W(Type = mammal/Type \neq mammal | Eggs = yes) \\
= \; & (+3.82) + (-5.30) \\
= \; & -1.48 \\
& W(Type = bird/Type \neq bird | Eggs = yes, Aquatic = no, \\
& \quad Predator = yes, Toothed = yes, Legs = four, Tail = yes) \\
= \; & W(Type = bird/Type \neq bird | Eggs = yes, Aquatic = no, Tail = yes) \\
& \quad + W(Type = bird/Type \neq bird | Legs = four) \\
& \quad + W(Type = bird/Type \neq bird | Toothed = yes) \\
= \; & (+3.81) + (-\infty) + (-\infty) \\
= \; & -\infty \\
& W(Type = reptile/Type \neq reptile | Eggs = yes, Aquatic = no, \\
& \quad Predator = yes, Toothed = yes, Legs = four, Tail = yes) \\
= \; & W(Type = reptile/Type \neq reptile | Eggs = yes, Toothed = yes, Legs = four, \\
& \quad Tail = yes) + W(Type = reptile/Type \neq reptile | Aquatic = no) \\
& \quad + W(Type = reptile/Type \neq reptile | Predator = yes)
\end{aligned}
$$

$$= (+4.82) + (-1.44) + (+0.56)$$

$$= 3.94$$

$$W(Type = fish/Type \neq fish|Eggs = yes, Aquatic = no,$$

$$Predator = yes, Toothed = yes, Legs = four, Tail = yes)$$

$$= W(Type = fish/Type \neq fish|Eggs = yes, Tail = yes)$$

$$+W(Type = fish/Type \neq fish|Legs = four)$$

$$+W(Type = fish/Type \neq fish|Aqutic = no)$$

$$= (+1.76) + (-\infty) + (-\infty)$$

$$= -\infty$$

$$W(Type = amphibian/Type \neq amphibian|Eggs = yes, Aquatic = no,$$

$$Predator = yes, Toothed = yes, Legs = four, Tail = yes)$$

$$= W(Type = amphibian/Type \neq amphibian|Aquatic = no)$$

$$= -\infty$$

$$W(Type = insect/Type \neq insect|Eggs = yes, Aquatic = no,$$

$$Predator = yes, Toothed = yes, Legs = four, Tail = yes)$$

$$= W(Type = insect/Type \neq insect|Eggs = yes, Aquatic = no)$$

$$+W(Type = insect/Type \neq insect|Legs = four)$$

$$= (+2.15) + (-\infty)$$

$$= -\infty$$

$$W(Type = invertebrate/Type \neq invertebrate|Eggs = yes, Aquatic = no,$$

$$Predator = yes, Toothed = yes, Legs = four, Tail = yes)$$

$$= W(Type = invertebrate/Type \neq invertebrate|Legs = four, tail = yes)$$

$$+W(Type = invertebrate/Type \neq invertebrate|Eggs = yes)$$

$$
\begin{aligned}
& +W(Type = invertebrate/Type \neq invertebrate|Toothed = yes) \\
= \; & (-\infty) + (+2.03) + (-2.72) \\
= \; & -\infty
\end{aligned}
$$

According to the weights of evidence, it can then be concluded that this animal is a reptile because the weight of evidence which supports the attribute *Type* to take the value *reptile* is the greatest of all the seven weights. One point that should draw our attention is the $-\infty$. It indicates that the observed event prevents the predicting attribute from taking on a certain value for sure. In the training data, compound events such as $[Legs = four,\ Type = insect]$ never happen. The odds in favor of *Type* taking on the value *insect* is zero. Hence, if a creature has four legs, the weight of evidence supporting it to be an insect is negative infinity.

Chapter 5

Experiments and System Performance

To evaluate the performance of the proposed methods, a prototype system is implemented. On such a system, experiments are conducted using both synthetic and real-world data sets. The experiments are grouped into two categories. In the first category, the pattern discovery algorithm introduced in Chapter 3 is tested, while in the second, the classification algorithm presented in Chapter 4 is tested with common data sets used by other machine learning algorithms. Then, comparisons are made and explained. A brief introduction to the system implementation will precede the presentation of the experiments. This chapter closes with a brief summary and some discussions on the experimental results.

5.1 System Implementation

Fig. 5.1 shows the skeleton of the prototype system. Basically, this system consists of three components. The pattern discovery engine detects significant event associations from a given database. The patterns are then represented by an attributed hypergraph in a data structure which bridges discovery and inference. The inference engine is invoked by a query from the user. If the user would like to find the associations related to a certain (group of) attribute value(s), the hypergraph is searched and an answer is given. If a new instance is to be classified against an attribute, the weights of evidence regarding the values of this attribute are calculated and compared.

The system does not target a specific format of the database. For relational databases, a universal relation (UR) model (similar to a table) can always be used to represent the data. Other database structures have to be transformed into a table-like format first. In the system implementation, the data are stored in a data file (ASCII or binary) in a two dimensional table with a data dictionary for the symbols and attribute names.

A double linked list with a hash table is used as the data structure to represent the attributed hypergraph in the computer. The double linked list stores the entries to the hyperedges of the same order. The hash table acts as the index to a certain hyperedge of that order associated with the linked list node. Such a data structure is described in Fig. 5.1. The pseudocode of each node in the linked list structure can be found in Fig 5.3. For a given order, the vertices (primary events) contained in the hyperedges are hashed. This data structure is efficient for locating a hyperedge, or hyperedges, given a set of vertices. This operation is frequently called by the inference process when associations regarding one or more attribute values are

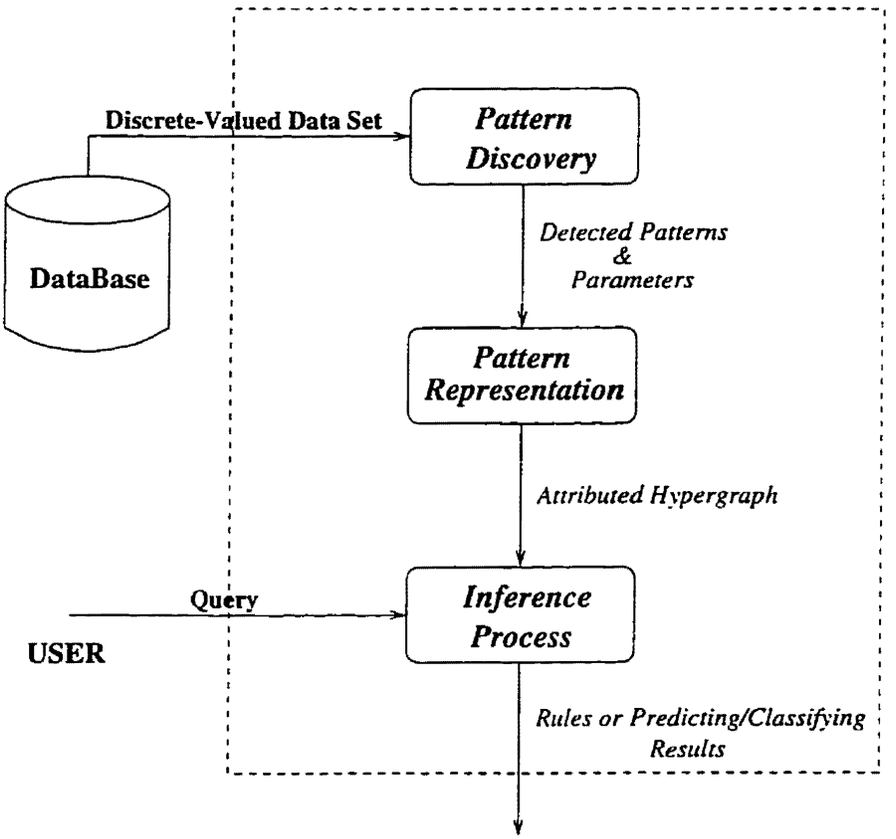

Figure 5.1: Overview of the Prototype System

searched.

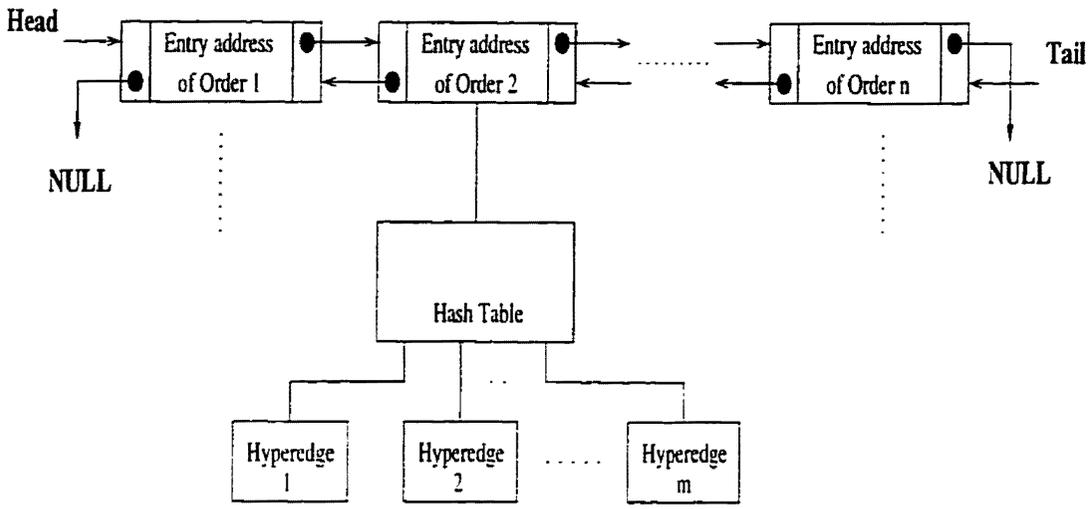

Figure 5.2: Data Structure of Attributed Hypergraph

```
typedef struct _HyperEdge
{
  int order;
  HTable *hash_table;
  HyperEdge *next;
  HyperEdge *previous;
} HyperEdge;
```

Figure 5.3: Data Structure of a Node in the Linked List

The prototype system is implemented in the programming language C using the data structures described above.

The pattern discovery unit in Fig. 5.1 is realized by the implementation of the discovery and candidate selection algorithms given in Chapter 3. It is basically a process of generating an attributed hypergraph from the training data offered as a data file. First the primary events of the domain described by the data set are

identified. The pattern candidates of each order are generated on the fly using the candidate selection algorithm. The significance of the candidates is tested by calculating their adjusted residual under the independence model. If their significance is verified, a hyperedge is generated at that order and stored in the hash table. When the algorithm stops, an attributed hypergraph is completed and can be used for pattern analysis and inference tasks in the next stage.

With the attributed hypergraph available, the analysis and inference process can now be carried out. This process is initiated by a query from an external user and it is driven by the algorithms implemented for the best-N problem and the missing-value problem (see Chapter 4 for the details of the algorithms). Hence, the queries from the user can have two formats: one containing a number together with a set of constraints for the desired rules, and the other containing a set of observed primary events together with the index to the attribute whose value is missing. To work on a problem, the user will be asked to choose one of the two tasks, and to provide the necessary information respectively.

The experiments are carried out on a Pentium 90 PC with 16 MB RAM running the Linux operating system.

5.2 Experiments on Pattern Discovery

In order to evaluate the proposed pattern discovery method, experiments are carried out to answer the following questions: 1) Can different order associations inherent in a data set be detected by the proposed method? 2) How well does the proposed method handle noise? and 3) Is the proposed method computationally acceptable? To answer these questions, the method is applied to four data sets. The first is an XOR data set consisting of 3 binary attributes. The second is a group of

multi-valued data sets with a different number of attributes and a different number of instances. The third is the zoo database obtained from the UCI repository of machine learning databases [71]. The fourth is a large real-world database of injury records of an electrical company over a period of time.

Each experiment on different data sets is selected to test the system on its various aspects. Table 5.1 shows the data sets and their testing goals.

Table 5.1: Data Sets Selected for Testing Various Pattern Discovery Goals

	XOR	1st Synth.	2nd Synth.	Zoo	Injury
Noise Screening	√		√		
Supervised Discovery	√			√	√
Unsupervised Discovery	√			√	√
Comput. Complexity		√	√		√
Significant Levels			√		√
Candidate Selection		√			√
Pattern Meaningfulness	√			√	
Pattern Consistency					√
Pattern Distribution					√

5.2.1 XOR Problem

The XOR problem was chosen because it can be solved only by *polythetic* learning strategies. The goal is to see if the proposed method can detect high-order patterns, and to find out how unsupervised and supervised discovering strategies perform when applied to the XOR problem.

To obtain an objective average performance evaluation of the method, ten XOR data sets have been randomly generated, each of which has 102 instances. From

the theoretical construct, it is possible to discover patterns of all orders in the deterministic case. To investigate the performance of the method in the presence of uncertainty, 10% noise was added to each data set by randomly changing the logic relationships of the instances. The pattern discovery algorithm was applied to these data sets sequentially. The experimental results of the individual sets as well as the overall average performance will be examined.

First the pattern discovery method is applied to each of these data sets without specifying an attribute as the "class" (the *unsupervised* setting.) There are altogether six primary events, i.e., [**A**=*F*], [**A**=*T*], [**B**=*F*], [**B**=*T*], [**C**=*F*] and [**C**=*T*]. The algorithm starts by searching for second order patterns in the data set. Because no *a priori* knowledge is available, all the possible combinations (second order pattern candidates) have to be tested. To demonstrate both the qualitative and the quantitative associations, tables instead of attribute hypergraphs are used here to show the outcomes of the pattern detecting algorithm. The results of second-order pattern detection for the 10 data sets are similar. Only some quantitative differences occur. One set of the results is reported in Table 5.2. Since there are no statistically significant second order patterns, no hyperedge is generated. At the end of the search of second order patterns, the algorithm proceeds for the search of third order patterns, since none of the third order pattern candidates should be eliminated from further consideration. The reasons are: 1) one cannot conclude from the occurrence and the expectation that any higher-order candidates will be noise, and 2) second-order is not the highest order one can go. Thus, the algorithm continues searching for higher-order (third-order) patterns. Again, there is no qualitative difference among the results of all the data sets. Table 5.3 shows the results of the third order patterns for the same data set. It is easy to see that there are four positively significant patterns and four negative ones. The four pos-

itive hyperedges can be found in Fig. 3.2. The expectations of all the events are greater than the threshold (here 5) for valid statistical test. If there is no noise, the occurrence of the negatively significant events will be zero and the algorithm will stop no matter whether or not there are possible higher order patterns. In the presence of noise, a threshold larger than zero should apply for effective candidate screening. Even if there were possible higher-order patterns, those containing the sub-compound events whose occurrences are less than this threshold will not be further considered. Since third-order is the highest in this problem, the algorithm stops. The output reflects exactly the XOR relationship.

Table 5.2: Second-Order Pattern Detection from XOR Data Set : Unsupervised

Candidate	o_x	e_x	d_x	Pattern	Eliminate
A=*F*, B=*F*	24	27.0	-1.20	No	No
A=*T*, B=*F*	27	24.0	1.18	No	No
A=*F*, B=*T*	30	27.0	1.20	No	No
A=*T*, B=*T*	21	24.0	-1.18	No	No
A=*F*, C=*F*	27	25.9	0.44	No	No
A=*T*, C=*F*	22	23.1	-0.44	No	No
A=*F*, C=*T*	27	28.1	-0.44	No	No
A=*T*, C=*T*	26	24.9	0.44	No	No
B=*F*, C=*F*	26	24.5	0.59	No	No
B=*T*, C=*F*	23	24.5	-0.59	No	No
B=*F*, C=*T*	25	26.5	-0.59	No	No
B=*T*, C=*T*	28	26.5	0.59	No	No

In the second setting, the variable C is considered as the known class attribute (the *supervised* setting). Thus, there are two classes, *Class 1*: [C=*F*] and *Class 2*: [C=*T*]. Under this situation, only pattern candidates which contain one of the two classes are examined. Without the loss of generality, the results of searching for the first order patterns associated with the classes are shown by Table 5.4 using the

Table 5.3: Third-Order Pattern Detection from XOR Data Set : Unsupervised

Candidate	o_x	e_x	d_x	±ve
A=*F*,B=*F*,C=*F*	23	13.0	5.58	+
A=*T*,B=*F*,C=*F*	3	11.5	-4.84	-
A=*F*,B=*T*,C=*F*	4	13.0	-5.04	-
A=*T*,B=*T*,C=*F*	19	11.5	4.28	+
A=*F*,B=*F*,C=*T*	1	14.0	-7.18	-
A=*T*,B=*F*,C=*T*	24	12.5	6.46	+
A=*F*,B=*T*,C=*T*	26	14.0	6.64	+
A=*T*,B=*T*,C=*T*	2	12.5	-5.91	-

same data set as in the unsupervised cases. Because of the nature of XOR data, none of a single primary event such as [A=*F*] can give classification information to either *Class 1* or *Class 2*. Thus, Table 5.4 can be considered as a part of Table 5.2, where the tuples not containing the class attribute are deleted. Again, at this level of search for first order association of the pattern with the class labels (or second order pattern-class associations), neither statistically significant patterns are found, nor are pattern candidates eliminated from future consideration. The final output of the algorithm is shown in Table 5.5 without significant negative associations. Fig. 5.4 gives the *AHG* representation of *Class 1* where quantitative values are not shown. The hypergraphs generated by the 10 data sets are the same except that the values of the attributes such as *occurrences*, *expectation* and *adjusted residual* are subject to minor variations. If these results are input into a rule generator where the logic relations between events are checked such as in AQ algorithms [63], one can easily obtain the rules such as *"If Attribute A is equal to Attribute B then the object belongs to Class 1"*.

In summary, the proposed method can successfully detect high-order associa-

Table 5.4: Second-Order Pattern Detection from XOR Data Set : Supervised

Class	Object	o_x	e_x	d_x	Pattern	Eliminate
C=F	A=F	27	25.9	0.44	No	No
	A=T	22	23.1	-0.44	No	No
	B=F	26	24.5	0.59	No	No
	B=T	23	24.5	-0.59	No	No
C=T	A=F	27	28.1	-0.44	No	No
	A=T	26	24.9	0.44	No	No
	B=F	25	26.5	-0.59	No	No
	B=T	28	26.5	0.59	No	No

Table 5.5: Patterns Detected from XOR Data Set : Supervised

Class	Object	o_x	e_x	d_x
C=F	A=F.B=F	23	13.0	5.58
	A=T.B=T	19	11.5	4.28
C=T	A=F.B=T	26	14.0	6.64
	A=T.B=F	24	12.5	6.46

tions such as XOR from the data set even in the presence of noise. In the XOR problem, since the analysis would stop at the third order, the elimination of insignificant high order events is not necessary. The unsupervised pattern discovery setting here tests every possible candidate while the supervised version tries only those related to a class. The total number of patterns detected by unsupervised version will be more than, or at least equal to, that detected by the supervised version. In these two experiments, the proposed algorithm finds the correct associations: the unsupervised version finds the false associations (noise), such as rows 2, 3, 5 and 8 of Table 5.3, as negative associations, whereas the supervised version

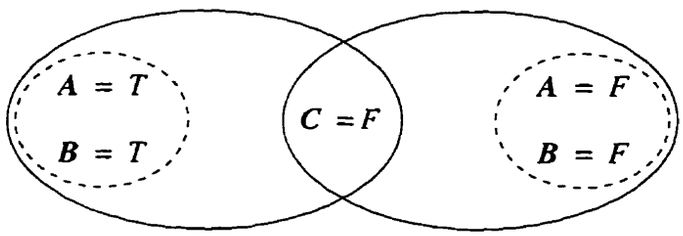

Figure 5.4: *Class 1* of XOR

does not include noise associations into its classification scheme. The running time of the supervised version is shorter than that of the unsupervised version due to the decrease in the number of pattern candidates to be tested. With the supervised setting, predictions are made on the class attribute, i.e. the membership (class) of the object. However in the unsupervised case, the prediction is more flexible as any attribute value can be predicted with other observation(s) given. This is the "flexible prediction" which Fisher refers to in [31].

5.2.2 Synthetic Data Sets

The XOR data set in the first test is described by binary attributes and has only third-order patterns. In the next experiment, it is desirable to know: 1) how the algorithms will behave if the data set contains different order patterns; 2) how the impossible high order patterns are to be eliminated from future consideration; and 3) what the learning time will be for a more complicated situation. To investigate the above questions, two groups of artificial data sets have been generated. The first group contains sets of random data. In the second group, data sets containing known associations are used.

Seven sets of random data are first generated. Each data set is described by 8 attributes with the same domain of 5 possible values. The number of instances

from the data sets are 1K, 5K, 10K, 50K, 100K, 500K and 1000K, respectively. The actual sizes of these data sets range from 25K to 25M as shown in Table 5.6. In the table, data sets are named in the form of Ax-Dy-Iz where x, y and z represent the number of attributes, the number of values and the number of instances respectively. Since no previously known associations have been added to the data set, the seven data sets are almost in the worst case which means that each possible combination has the same probability and none of them may be significant but have to be tested, although few associations are expected to be determined. In such a case, the candidate selection techniques discussed in Section 3.5 do not provide much improvement in favor of the executing time. In this way, however, one can have a sense of the upper bound of the complexity. Fig. 5.5 shows the result of computing time against the number of instances in the data sets. It is easy to see that the computing time is quite linear except for the left end of the curve. The data sets falling into this section are those with less data points. If a data set has less instances, the requirement for valid statistical test discussed in Section 3.5 is less likely to be fulfilled as the order goes up. This criteria thus limits the highest order the data sets can go. As shown in the figure, the proposed algorithm scales linearly against the number of instances in a data set.

In addition to the number of instances, another important factor on the performance of the system is the number of attributes. To investigate this effect, the number of instances has been kept as 10K and the number of attributes in a data set is now set between 2 and 14. The execution time is shown in Fig. 5.6, where the solid curve represents the actual execution time. The dotted curve represents the execution time if exhaustive search is used.

As Fig. 5.6 shows, the solid curve of executing time has the shape similar to an exponential function. The benefit of applying the candidate selection algorithm is

Table 5.6: Data sets and Execution Time

Name	Attribute	Domain	Instance	Size	Time (sec)
A8-D5-I1K	8	5	1K	25K	11.6
A8-D5-I5K	8	5	5K	125K	249.5
A8-D5-I10K	8	5	10K	250K	411.6
A8-D5-I50K	8	5	50K	1.25M	906.1
A8-D5-I100k	8	5	100K	2.5M	1503.8
A8-D5-I300K	8	5	300K	7.5M	4044.9
A8-D5-I500K	8	5	500K	12.5M	6286.5
A8-D5-I700K	8	5	700K	17.5M	8829.1
A8-D5-I1000K	8	5	1000k	25M	12327.7

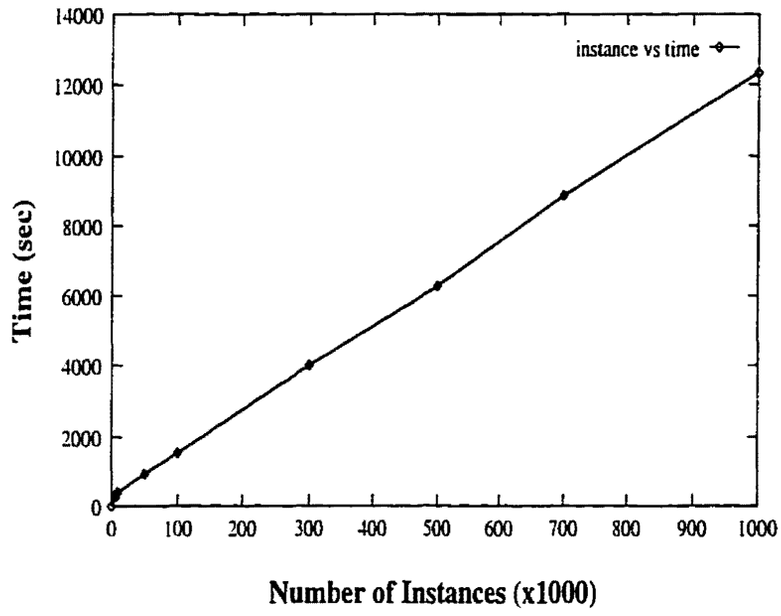

Figure 5.5: Computing Time vs Number of Instances

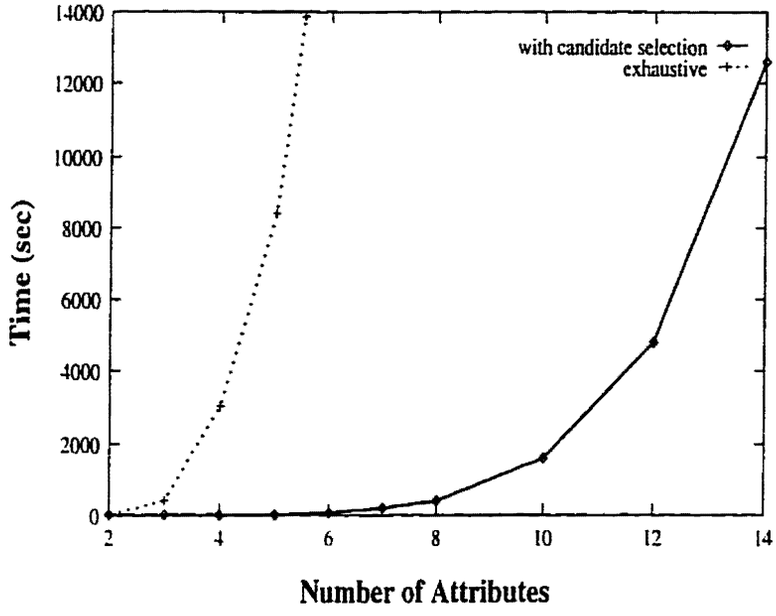

Figure 5.6: Computing Time vs Number of Attributes

that even in the worst case the highest order is bounded. The search will stop before reaching the highest order which is the number of attributes. Table 5.7 shows the number of tested candidates with the data set A6-D5-I10K. Since no candidates are generated at order 5, the search stops here. To investigate the difference in running-time between the proposed method and the exhaustive search, another method using multi-way contingency tables was implemented. This approach generates all the high-dimensional contingency tables and exhaustively tests each cell in every table. Using this method, the execution time will be truly exponential. As shown by the dotted curve in Fig. 5.6, it has been tried only up to A6-D5-I10K with exhaustive searching. The process is terminated after 4 hours, because the trend is already known. As mentioned earlier, the data sets used are random, which means every event has approximately the same probability. In the real world, however, it is expected that some significant associations inherent in the data set will reduce the searching space significantly.

Table 5.7: Candidates Generated with A6-D5-I10K

Order	2	3	4	5	6
Candidates	375	2500	9300	0	-

In addition to investigating the computing characteristics, the following situations are also of interest: 1) how the algorithms will behave if the data set contains different order associations: and 2) how the impossible high order patterns are to be eliminated from future consideration. To answer these questions, another artificial data set has been generated. It consists of 8 attributes, each of which could take on one of the four values, say v_i^1, v_i^2, v_i^3 and v_i^4, $1 \leq i \leq 8$. Fig. 5.7 shows how the data set is generated.

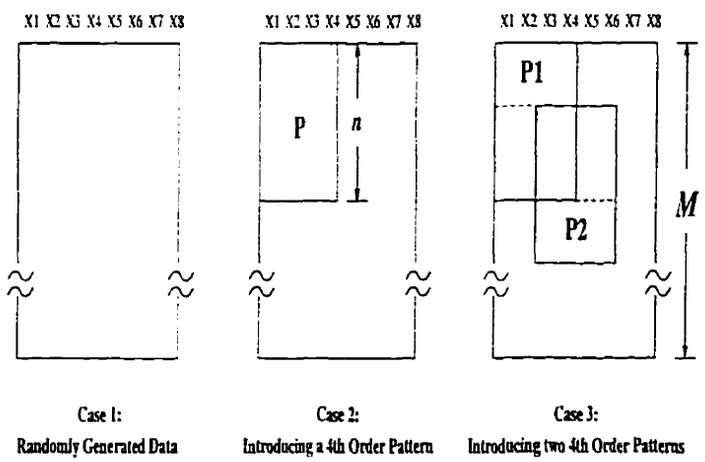

Figure 5.7: Multi-valued Data Sets for Testing

First, 800($M = 800$) samples are randomly generated and organized in a tabulated form as shown by Case 1 of Fig. 5.7. At the second step (Fig. 5.7, Case 2), a fourth-order association $[X_1 = v_1^3,\ X_2 = v_2^1,\ X_3 = v_3^4,\ X_4 = v_4^4]$ (note that $s = \{1, 2, 3, 4\}$) is added to the originally randomly generated data set of Case 1.

The number of the instances (or realizations) fitting this pattern is n. The number n is then changed to investigate the behavior and performance of the proposed method. At the beginning, n is as small as 5 to 10. In these cases, the algorithm does not detect any patterns. Although some occurrences and expectations of certain compound events do vary from the original data, they still cannot pass the significance test. When n is further increased (around 40), the fourth-order association emerges, but none of the lower order associations are detected. The reason is, although n is big enough for the fourth-order pattern to pass the significance test, it is not large enough for the lower order patterns to pass the test. This demonstrates that the existence of higher-order patterns does not indicate the existence of lower-order patterns. If n keeps on growing, third-order associations, such as $[X_1 = v_1^3,\ X_2 = v_2^1,\ X_3 = v_3^4]$ and $[X_1 = v_1^3,\ X_3 = v_3^4,\ X_4 = v_4^4]$ then emerge (when n is around 65). Further on, second-order associations, such as $[X_1 = v_1^3,\ X_2 = v_2^1]$ and $[X_2 = v_2^1,\ X_4 = v_4^4]$, appear when n is larger than 80. It is observed that, in order to be statistically significant, lower order candidates need larger n than higher order candidates. That the difference of n between two adjacent orders $(n_{i-1} - n_i)$ is not a constant is also noted.

In the third step (Fig. 5.7, Case 3), two fourth-order associations ($\mathbf{P}_1$ and $\mathbf{P}_2$) are introduced. $\mathbf{P}_1$ is of the same association as that in Case 2 of Fig. 5.7. $\mathbf{P}_2$ is $[X_3 = v_3^1,\ X_4 = v_4^3,\ X_5 = v_5^2,\ X_6 = v_6^4]$ (with $s_2 = \{3, 4, 5, 6\}$). For simplicity, the number of instances for each association is kept at the level that only fourth-order patterns can be detected. Let this number be n_4. $\mathbf{P}_2$ is then shifted in the data set and merged with $\mathbf{P}_1$ to show how the two associations interact and what associations would be detected. After $\mathbf{P}_2$ merges with $\mathbf{P}_1$ and forms a new association $\mathbf{P}_3$, the values of X_3 and X_4 in $\mathbf{P}_3$ will adopt those of $\mathbf{P}_1$. See Fig. 5.8 for details of this experiment.

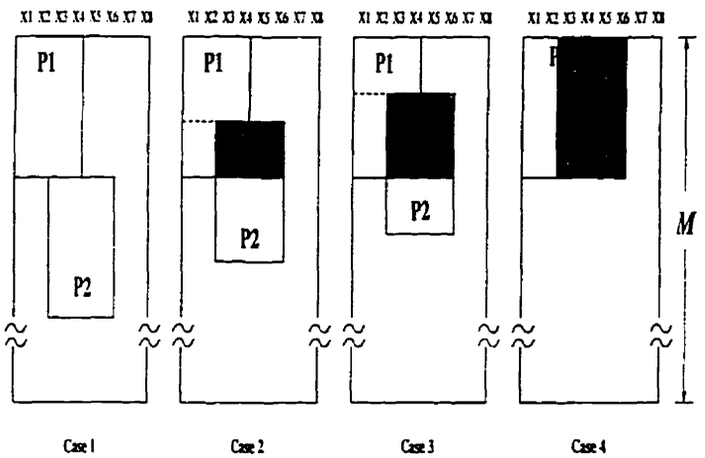

Figure 5.8: Moving Patterns in the Data Set

When $\mathbf{P}_1$ and $\mathbf{P}_2$ do not overlap (Fig. 5.8, Case 1), the test result is quite similar to the single association case (Fig. 5.7, Case 2). In this case, both $\mathbf{P}_1$ and $\mathbf{P}_2$ are discovered. Then, as $\mathbf{P}_2$ gradually decreases, $\mathbf{P}_3$ ($[X_3 = v_3^4,\ X_4 = v_4^4,\ X_5 = v_5^2,\ X_6 = v_6^4]$, $s_3 = \{3, 4, 5, 6\}$) emerges and gradually becomes predominant. However, the total number of instances in $\mathbf{P}_2$ (denoted by n_{P_2}) and $\mathbf{P}_3$ (denoted by n_{P_3}) remains at n_4 (Fig. 5.8, Case 2 to Case 4). Because of the decrease of n_{P_2}, $\mathbf{P}_2$ would not be significant any more when $n_{P_2} < n_4$. When n_{P_3} is small (< 10), only $\mathbf{P}_1$ will be detected. Both $\mathbf{P}_2$ and $\mathbf{P}_3$ will be considered as insignificant. This case is the same as Case 2 in Fig. 5.7. As n_{P_3} increases to about 25 (Case 2), besides $\mathbf{P}_1$, a new sixth order association $[X_1 = v_1^3,\ X_2 = v_2^1,\ X_3 = v_3^4,\ X_4 = v_4^4,\ X_5 = v_5^2,\ X_6 = v_6^4]$ ($s_4 = \{1, 2, 3, 4, 5, 6\}$) is first detected. When n_{P_3} increases to 31, $\mathbf{P}_1$, the sixth order association as well as the six fifth order associations are all detected (Case 3). In Case 4, $\mathbf{P}_2$ vanishes completely. Now, all the 22 patterns (i.e., the sixth order pattern, six fifth order patterns, and 15 fourth order patterns including $\mathbf{P}_1$ and $\mathbf{P}_3$) are detected.

The learning time (pattern detection time) varies with each case. Approxi-

mately, the running time is 12 seconds. If multi-way contingency tables are used to find the significant compound events of Case 4, the results are the same but the program runs for more than 25 minutes, which is 120 times slower.

For associations of different orders to be statistically significant, the requirements for the occurrences of the candidates are usually different. Since complete contingency tables are not generated, only possible compound events are tested. Hence, the computational complexity is much less. Once a compound event is considered to have no contribution to the description of the data set, its elimination from further consideration is justified. It follows that all the higher order compound events having this event as one of their sub-compound events can also be eliminated simultaneously.

5.2.3 Zoo Database

To demonstrate the efficiency and meaningfulness of this new method in dealing with real world problems, the zoo database is tested. This database is obtained from the UCI repository of machine learning databases [71] and originally shown in Forsyth's PC/BEAGLE User's Guide. The zoo data is a simple database containing 101 instances of animals, each described by 17 attributes in addition to animal names, which are unique to each animal. All the attributes except two are boolean. The two exceptions are the number of legs, which is integer ranging from 2 to 8, and the type of animals, which is one of the seven types: mammal, bird, reptile, fish, amphibian, insect and invertebrate. All the attributes are listed in Table 5.8. The purpose of this test is to determine whether or not the patterns detected by the proposed system are meaningful and usable later in reasoning.

When the pattern discovery algorithm is applied to the data set, a total of

Table 5.8: Attributes of Zoo Database

0	Animal Name	Unique for each animal
1	Hair	Boolean
2	Feathers	Boolean
3	Eggs	Boolean
4	Milk	Boolean
5	Airborne	Boolean
6	Aquatic	Boolean
7	Predator	Boolean
8	Toothed	Boolean
9	Backbone	Boolean
10	Breathes	Boolean
11	Venomous	Boolean
12	Fins	Boolean
13	Legs	Numeric
14	Tail	Boolean
15	Domestic	Boolean
16	Catsize	Boolean
17	Type	Categorical

1,104 hyperedges representing 1,104 significant associations are found, of which the highest order is five and the lowest is two. Some of the compound events are eliminated from consideration quite early. For example, the compound event $[Feathers = yes,\ Milk = yes]$ which never occurs in the database is considered as a negatively significant compound event. Hence, any higher-order events which contain $[Feathers = yes,\ Milk = yes]$ cannot be more significant than this event and are eliminated from consideration. Such eliminations influence not only the negative but also the positive sub-compound events. Fig. 5.9 is a small subgraph of the attributed hypergraph representing the associated patterns generated by the algorithm from this data set. It provides explicit information on the inherent

patterns in the data set.

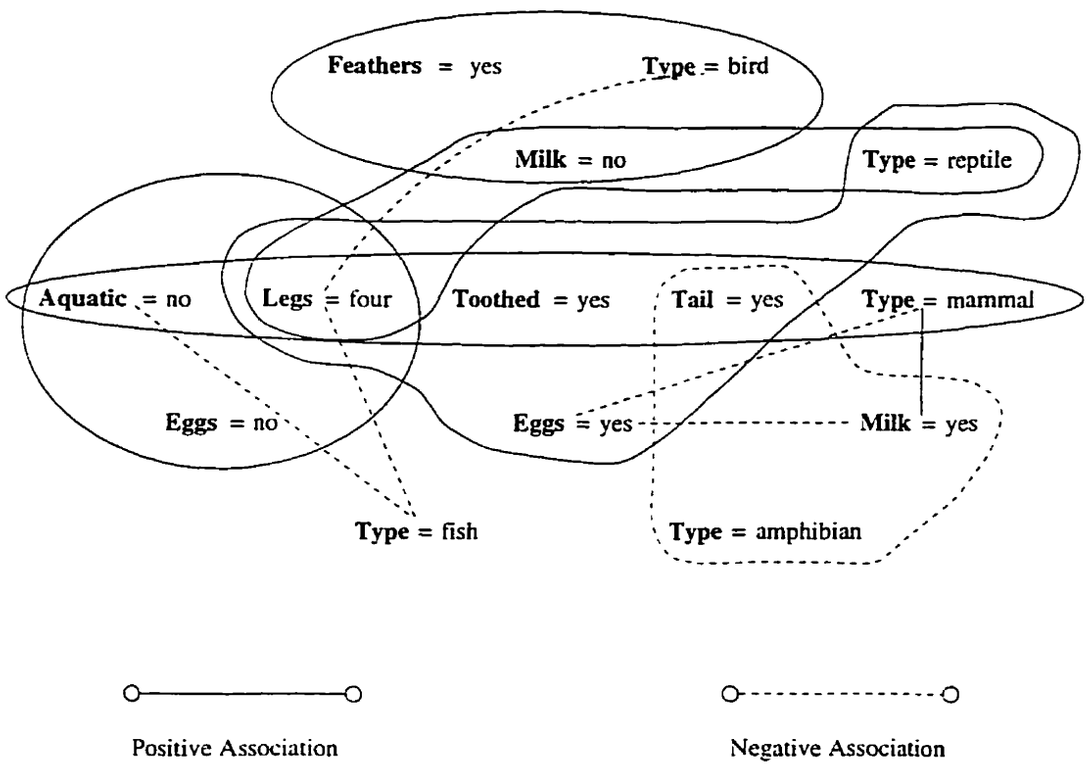

Figure 5.9: Part of the AHG for Zoo Database

To check their meaningfulness, 15% of the 1,104 associations are selected randomly. First, a random sequence of 166 (15% of 1,104) elements ranging from 1 to 1,104 is obtained by a pseudo random number generator of the computer. Then the patterns corresponding to the elements in the sequence are picked up. Each of these patterns is examined by applying domain knowledge of animal genealogy. It is found that all the examined patterns are meaningful from the perspective of the domain knowledge, though some of them contain redundant information. For example, the events $[Feathers = yes, Type = bird]$, $[Hair = no, Feathers = yes, Eggs = yes]$ and $[Hair = no, Eggs = yes, Backbone = yes, Type = bird]$ are all significant. It is common knowledge that birds always have feathers and they lay eggs. They

normally do not have hair but they have a backbone. People may infer that creatures which do not have hair yet have feathers and lay eggs are birds. There is definitely redundant information describing the relations. However, this redundant information is useful for inference, as one does not know which attribute is going to be predicted and which events have been observed. In other words, to achieve flexible prediction, redundant information is needed.

The program runs for 1.42 seconds.

To test the supervised learning performance, the variable *Type* is considered as the class attribute and the database is fed to the pattern discovering process for classification. The system reports 284 significant compound events of order 2 to 5. With the same sampling method mentioned before, all these significant associations are meaningful. Actually the patterns found through the supervised learning strategy are a subset of those found by the unsupervised learning program. The only difference is that the search space of supervised learning is much smaller than that of unsupervised learning. In this case, the program runs for less than 1 second.

5.2.4 Injury Database

In addition to the artificial data sets, the proposed method is applied to a real world database. This database stores the injury records of an electric company from 1965 to 1995. There are about 75,000 instances and the database itself is about 10M bytes in the format of FoxPro version 3.1. Associated with each injury case are various details describing, for example, the nature of the injury, the type of work, the age and sex of the employee and the severity of the injury. These details form the *fields* of each injury case. Table 5.9 shows all the fields under investigation

in the database. This database contains potentially valuable relationships among the various fields. For example, the relationship between the type of work and the severity level is embedded within the voluminous database. The knowledge and awareness of these relationships may suggest preventive or precautionary measures, identify risk groups or hazardous activities, and may highlight patterns of injury deserving further investigation.

Table 5.9: Fields in the Injury Database

Field Name	Inj. Season	Branch	Age	Svc. Yrs	Sex	Trade
Size of Domain	4	13	5	6	2	46
Field Name	OR Flag	Inj. Type	Severity	Voltage	Body Part	Desc.
Size of Domain	2	6	8	4	25	19

Since the current implementation can handle only discrete variables, the fields with continuous data have to be discretized before being fed into the system. Discretization is a difficult task which has a great impact on the performance of the system, but it is beyond the scope of this thesis. In the experiments, intervals are manually assigned to continuous variables to make them discrete. For example, the injury dates are categorized into four seasons and employee age into twenties, thirties, forties, fifties and sixties. In this way, a finite domain for each field is well defined.

To make the experiment easier to present, the discovery process is first confined to detect the association related to a special field *Severity*. The execution time, the number of associations with respect to orders and the number of tested hypotheses for the test are given in Table 5.10. As shown in the table, at the lower orders, the number of associations and the number of candidates increase with the order.

It is easy to understand that the possible combinations of attribute values are increasing. But even at lower order, not all the combinations are tested. The candidate selection algorithm helps to filter out a great portion of uninformative combinations. These combinations will not be considered in higher order cases. Fig. 5.11 shows the relative number of possible combinations, the number of tested candidates and the number of significant associations at both order 2 and order 3. When the detection algorithm passes order 6, both the number of candidates and the number of associations begin to drop. The performance of the candidate selection algorithm which takes the advantage of the sample size and the negative associations to eliminate combinations is more remarkable when the order goes higher. The ratio of significant associations and tested candidates is about 10% – 30%. Fig. 5.10 shows several significant associations found in the database.

Table 5.10: Order, Candidates, Associations and Execution Time

Order	Candidates	Associations	Execution Time (sec)
2	110	13	1.36
3	1367	283	27.49
4	6892	1780	542.68
5	17345	4989	3523.97
6	24269	7307	8742.78
7	19418	5908	8623.29
8	8544	2577	4292.55
9	1800	527	1155.81
10	119	31	153.03
11	0	0	10.54
>12	-	-	-

Fig. 5.12 shows two associations with similar frequencies in the database. The first association has 122 instances in support and the second has 117 in support.

<Events> BRANCH <IS> Des. & Const <AND> TRADE <IS> Construction <AND> SEVERITY <IS> 0 to 49 <ARE> <POSITIVELY> associated <WITH ADJUSTED RESIDUAL> 4.082	<Events> BRANCH <IS> Engr. Service. <AND> INJURY SEASON <IS> Winter <AND> SEVERITY <IS> 0 to 49 <ARE> <NEGATIVELY> associated <WITH ADJUSTED RESIDUAL> -6.739
<Events> INJURY TYPE <IS> MED <AND> AGE <IS> THIRTIES <AND> SEX <IS> Male <AND> BRANCH <IS> Des. & Const <AND> BODY PART <IS> Upper Back <AND> OR FLAG <IS> N SEVERITY <IS> 0 to 49 <ARE> <POSITIVELY> associated <WITH ADJUSTED RESIDUAL> 6.210	<Events> SVC. YRS <IS> Less than 2 <AND> SEX <IS> Female <AND> AGE <IS> Twenties <AND> BRANCH <IS> Production <AND> INJURY TYPE <IS> MED <AND> VOLTAGE <IS> Medium <AND> SEVERITY <IS> 0 to 49 <ARE> <NEGATIVELY> associated <WITH ADJUSTED RESIDUAL> -4.276

Figure 5.10: Association Examples Found in the Injury Database

The probabilities of the two events are quite close. But as indicated in the figure, the adjusted residuals which reflect their significance level are far from equal. The first one is only 3.985 and the second 11.235. Note that the number of cases supporting an association and the association's significant level may not be consistent over different associations. Two associations with similar frequency may have disparate significant levels. This example further demonstrates that absolute probability is not suitable for significance testing as discussed in Section 3.4.

It is also informative to examine the distribution of associations versus the absolute values of the adjusted residuals. Fig. 5.13 graphs the number of detected significant associations against the adjusted residual level. Associations with very high confidence measures occur so frequently that their existence is deemed obvious.

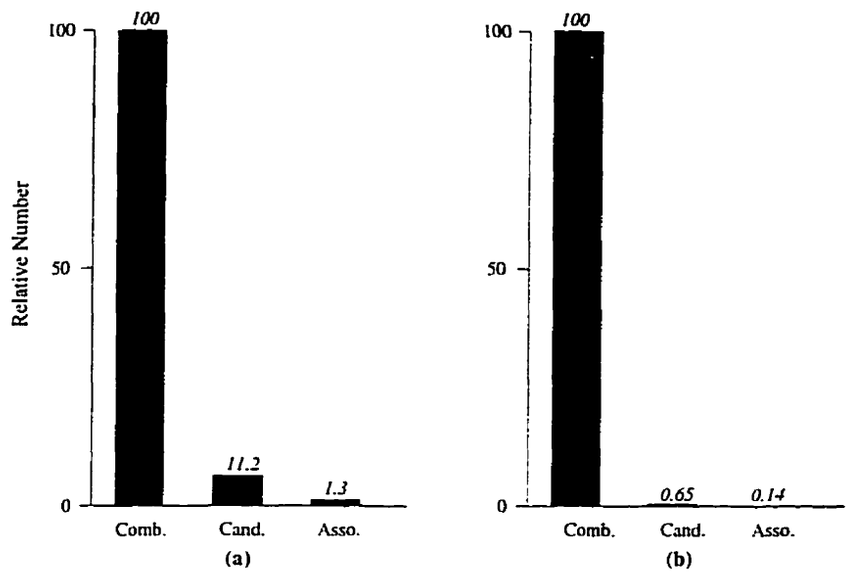

Figure 5.11: Relative Numbers of Possible Combinations, Selected Candidates and Significant Associations: (a) Order 2 (b) Order 3

Standard statistical testing or even sometimes simple inspection can reveal the presence of some of these obvious associations. However, at lower significance levels, the discovered associations are less obvious and may not be detected by standard analysis. In the examined database, the majority of significant associations occur around an adjusted residual range of 4 to 6. These constitute associations which are subtle and not easily determined. The proposed method here aptly unveils both the obvious and the subtle associations within the database.

Additional tests are made to investigate the consistency of the associations throughout each year. In the first test, the second order associations related to the field *Severity* in the year 1991 are compared with those in the year 1993. A noteworthy observation is that 8 out of 9 significant associations discovered from the 1993 data set also occur as significant in the 1991 data set. This suggests the detection of important associations that exist from year to year. The consistency of the

<Events> INJURY SEASON <IS> Winter <AND> BODY PART <IS> Upper Back <AND> SEVERITY <IS> 0 to 49 <ARE> <POSITIVELY> associated <WITH ADJUSTED RESIDUAL> 3.984 122 instances in support	<Events> BRANCH <IS> Regions <AND> TRADE <IS> Foresters <AND> SEVERITY <IS> 0 to 49 <ARE> <POSITIVELY> associated <WITH ADJUSTED RESIDUAL> 11.235 117 instances in support.

Figure 5.12: Patterns with Similar Frequencies But Different Levels of Confidence

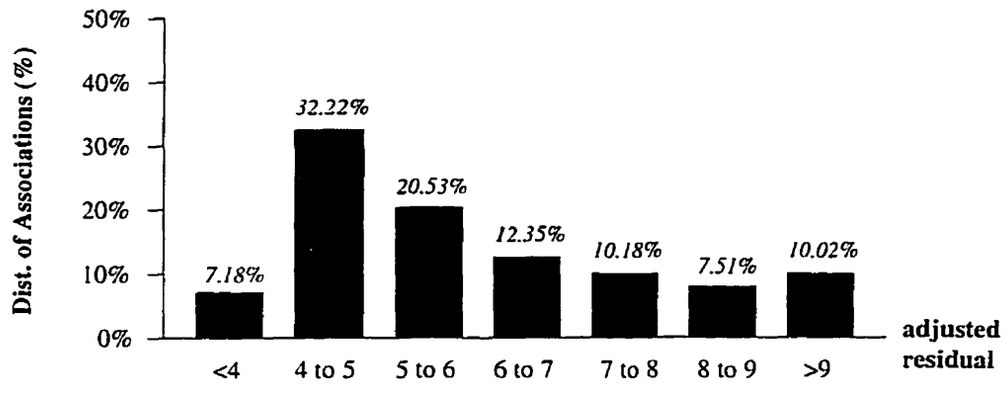

Figure 5.13: Distribution of Associations vs Adjusted Residual

associations is also verified by the 20 associations with highest adjusted residual for each year. Among them, 12 are common to both the 1991 and the 1993 databases. Again, this suggests that the time-invariant associations that are important over different years have been discovered. Lastly, no contradictory associations have been found between the two data sets.

In addition to discovering associations related to the field *Severity*, an unsupervised setting to find all the associations has also been tested with this database. The results resemble the previous ones with only quantitative differences. They will not be presented here in detail. It should be noted that, on one hand, the asso-

ciations discovered this way outnumber those in a supervised scenario considerably. On the other hand, the candidate selection technique plays an important role in truncating the search space for a feasible execution time.

Experiments with the real-world database show that the proposed method is able to handle a relatively large database with only limited computational power (an IBM PC) within acceptable execution time. The candidate selection algorithms effectively reduce the number of hypotheses to be tested, yet are able to discover the significant associations of different orders. As observed earlier, significant associations do not have similar significant levels. A great portion of them are "subtle" rather than "obvious". which means simple inspections or standard tests may not be able to find them. The actual number of occurrences of an association is not directly related to its significance. When applied to the injury database, this new method succeeds in discovering consistent significant associations over two different years of data. Such encouraging result indicates clearly that the inherent structure in the data set can be unearthed, at least to a certain extent.

5.3 Experiments on Classification

In this section, the experimental results of classification on four data sets are presented. The four data sets used are the zoo data set, the breast cancer data set, the mushroom data set. and the data sets for the monk's problems. All four can be obtained from the UCI repository of machine learning database [71]. The accuracies obtained with the new method are compared with those of other well-known classification algorithms. This section shows that, although the proposed system is a pattern discovery system and not a destined classifier, it can accomplish classification tasks using weight of evidence. Besides, in terms of performance, the new

system is comparable to, if not better than, other classification algorithms. This section also covers a comparison of classification accuracy and some discussions on misclassifications.

Table 5.11 shows the testing goals versus the four data sets used in the classification.

Table 5.11: Data Sets Selected for Testing Various Pattern Analysis Goals

	Zoo	Breast-Cancer	Mushroom	Monk's
Noise Screening				√
Flexible Prediction	√			
Classification Accuracy	√	√	√	√
Deterministic Discov.				√
Large Samples			√	
Understandability		√		

5.3.1 Zoo Database

The same zoo data set in Section 5.2.3 is again used here for classification tests. The description of the attributes and their domains are shown in Table 5.8.

In the test, a total of 20 randomly selected instances have been used as testing instances, representing about 20% of the total number of available instances, and the remaining 80% as the training data. This process is repeated 5 times to obtain an average performance. Creatures are to be classified against two different attributes. The first is *Type*, given a creature described by 16 other attributes. The other is *Legs*. To make it easy to compare, it is assumed that only the value of the class attribute is missing.

The highest order of found event associations is 5. Due to the small data size, most of the associations are of order 3 or less. The majority of higher order associations are contributed by mammals since this type of animal makes up 40% (41 out of 101) of the total instances.

When the attribute *Type* is used to classify the animals, the class distribution is as follows: 41 mammals, 20 birds, 5 reptiles, 13 fish, 4 amphibians, 8 insects, 10 invertebrates. An average classification accuracy of 96% (low 90%, high 100%) is achieved. Most of the 4% errors are rejections, in which case, there is no positive weight of evidence supporting any of the seven types. Cross examining the original data set finds that this situation happens to the reptiles and amphibians. Since they have fewer instances in the data set, to find positive associations is unlikely. The highest weights of evidence are close to zero. If it were allowed to assign a class to a creature whose highest weight of evidence is almost zero, a slightly higher accuracy could be achieved for this database, but for other databases in general, such an approach risks assigning random class membership to the classifying object. Further studies are recommended here to investigate the trade-off. Besides rejections, the errors are largely due to the misclassifications of ambiguous creatures. For example, termite is the only insect without wings in the database. Since the insect class has many attributes in common with the invertebrate class, and none of the invertebrates can fly, termite is sometimes assigned to the wrong class.

Since the zoo data set has not been used to test many other algorithms besides Forsyth's PC/BEAGLE User's Guide, the classification accuracy is compared with only the earlier work (APACS) of this research. APACS [17] is a first-order monothetic learning algorithm which detects the relationships between a class and one attribute value. Table 5.12 shows the comparison. The proposed method is able to detect high order associations, which help to explain some complicated cases

which are difficult for monothetic algorithms such as APACS. For example, the animal platypus is the only mammal (out of 41) in the database that lays eggs. In the second order associations, there is a strong evidence against its assignment to the mammal class. Therefore, APACS either assigns it to a wrong class or rejects it since there is no sufficient positive weight of evidence. Yet using the current approach, platypus is assigned to the correct mammal class when the high order event associations are considered, the negative evidence provided by $eggs = yes$ is balanced by other higher order associations which distinguish it from other types such as insects (no backbone) and birds (all have feathers).

Table 5.12: Classification Accuracy on Attribute *Type* of Zoo Data

System	Classification Accuracy
APACS	85.0%
Proposed Method	96.0%

When the attribute *Legs* is used to classify the creatures, the class distribution becomes: 38 four-leggers, 27 two-leggers, 23 legless, 10 six-leggers, 2 eight-leggers, 1 five-leggers. An average classification accuracy of 97% (low 95%, high 100%) is achieved. Misclassification happens to the five-leg starfish, eight-leg scorpion and octopus. They are all invertebrates. Other types of creatures have tight correlation with the number of legs, except invertebrates. From a small sample for five-leg and eight-leg creatures, no positive weight of evidence can be obtained. In such a case, either rejection or the "most frequently happened" class is assigned. Since there is no previous classification result on this attribute, no comparison can be made. The classification on *Legs* demonstrates that the proposed approach can make classification on any attribute given an attributed hypergraph built from the

data set.

5.3.2 Wisconsin Breast-Cancer Database

The Wisconsin breast-cancer database was originally provided by Dr. William H. Wolberg [96] and used by a number of researchers in pattern recognition and machine leaning. The current database available from the UCI database repository contains 699 cases, each of which is described by ten attributes in addition to the unique code number (see Table 5.13). The class of each instance is either *benign* or *malignant* whose distribution is as follows: 458 (or 65.5%) benign, 241 (or 34.5%) malignant. Other attributes take discrete values labeled from 1 to 10.

Table 5.13: Attributes of Breast-Cancer Database

0	Code Number	Unique
1	Clump Thickness	1 - 10
2	Uniformity of Cell Size	1 - 10
3	Uniformity of Cell Shape	1 - 10
4	Marginal Adhesion	1 - 10
5	single Epithelial Cell Size	1 - 10
6	Bare Nuclei	1 - 10
7	Bland Chromatin	1 - 10
8	Normal Nucleoli	1 - 10
9	Mitoses	1 - 10
10	Class	Boolean

In this test, the focus is on the event associations related to the attribute *Class* and how to classify the cases using these found patterns. Of the total cases, 80% (599 cases) are used in training and the remaining 20% (140 cases) are classified. In the pattern discovering process, 42, 76 and 258 event associations are gener-

ated at second, third and fourth order respectively. The number of associations begins to decrease at the fifth order and no seventh order pattern has been found. Fig. 5.14 shows part of the AHG representation of the event associations detected from the database. The class labels are singled out to make it easier to read. Their associations with other primary or compound events are shown by the connected solid lines. Significant compound events associated with a class are enclosed by dotted curves. The results show that: 1) there is no overlapping associations between classes, i.e. if event $[\mathbf{x}_j^s, Class_1]$ is significant, $[\mathbf{x}_j^s, Class_2]$ is not significant; and 2) if $\mathbf{x}_j^s$ is associated to one class, none of the primary events in $\mathbf{x}_j^s$ appear in associations containing the other class. In this sense, the breast-cancer database is less complicated than the zoo database.

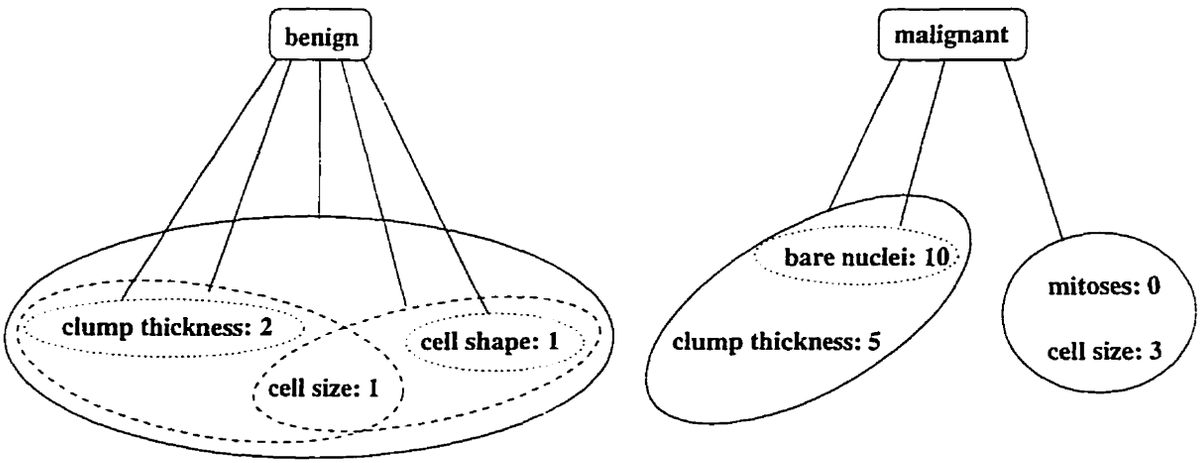

Figure 5.14: Part of the Significant Associations from Breast-Cancer Database

It is also noticed that 32 out of the 48 second order associations are related to *malignant*, but only 3 third order associations are related to this class. No higher order associations contain this class. It implies that most of the patterns related to *malignant* are of low order. To distinguish it from *benign*, one does not need to detect patterns of orders higher than 3. Hence, to classify a new case, low order

(no higher than 3) pattern discovery should be sufficient. It partially explains the phenomenon that monothetic learning strategies can achieve high classification accuracy in this problem.

The new method achieves 95.7% classification accuracy. This accuracy is comparable with that of other classification oriented systems using the Wisconsin breast-cancer data (see Table 5.14). In the table, APACS [17] which considers only pairwise associations yields 93.9% accuracy. Wolberg and Mangasarian [96] report they achieved a single-trial classification accuracy range of between 93.5% and 95.9% depending on the number of cases used in training (50% to 67% respectively). Their method is based on the hyperplane data separation on a subset of the data set (369 cases when their test was conducted). Zhang [107] reports a classification accuracy range of between 92.2% and 93.7% using the same subset as that in [96].

Table 5.14: Classification Accuracy of the Wisconsin Breast-Cancer Domain

System	Classification Accuracy
APACS (with rejection)	93.9%
Wolberg's Method	93.5 - 95.9%
Zhang's Method	92.2 - 93.7%
Proposed Method	95.7%

5.3.3 Mushroom Database

The mushroom database is a relatively large database containing 8124 instances. The instances are characterized by 23 attributes including the class label (*edible* and *poisonous*) whose distribution is as follows: 4208 (or 51.8%) edible, 3926 (or 48.2%) poisonous. All the rest 22 attributes are categorical (see Table 5.15).

Table 5.15: Attributes of Mushroom Database

1	cap-shape	bell, conical, convex, flat, knobbed, sunken
2	cap-surface	fibrous, grooves, scaly, smooth
3	cap-color	brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
4	bruises	bruises, no
5	odor	almond, anise, creosote, fishy, foul, musty, none, pungent, spicy
6	gill-attachment	attached, descending, free, notched
7	gill-spacing	close, crowded, distant
8	gill-size	broad, narrow
9	gill-color	black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
10	stalk-shape	enlarged, tapering
11	stalk-root	bulbous, club, cup, equal, rhizomorphs, rooted, missing
12	stalk-surface-above-ring	ibrous, scaly, silky, smooth
13	stalk-surface-below-ring	ibrous, scaly, silky, smooth
14	stalk-color-above-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
15	stalk-color-below-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
16	veil-type	partial, universal
17	veil-color	brown, orange, white, yellow
18	ring-number	none, one, two
19	ring-type	cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone
20	spore-print-color	black, brown, buff, chocolate, green, orange, purple, white, yellow
21	population	abundant, clustered, numerous, scattered, several, solitary
22	habitat	grasses, leaves, meadows, paths, urban, waste, woods
23	class	edible, poisonous

The experiment is carried out in two ways. First, a subset of 500 instances rather than the entire data set is randomly sampled from the original 8124 instances. Of the 500 selected instances, 400 (80%) are used in the discovery phase while the remaining 100 (20%) are used for testing. This process is repeated for 10 times to gain an average performance. In the classification phase, only the class label is classified for easy comparison with the other approaches. The limitation of the training sample size prevents the association discovery process from going to higher order. In the ten trials, the lowest classification accuracy obtained is 97% and the highest is 100%. The average accuracy is 98.9%.

In the second setting, the entire data set is used. The original 8124 instances are divided into a subset of 5416 (66.7%) for pattern discovery and a subset of 2708 (33.3%) for classification testing. The time consumed by the discovery process is significantly longer, but it is almost linear with respect to the number of instances. The accuracy for this one-trial classification is 99.1%, slightly better than the average performance of the ten-trial approach mentioned earlier.

These performances are consistent with other researchers' experiments using the same data set. Table 5.16 is a comparison result (classification results of other algorithms are obtained together with the data set from UCI). In the table, STAGGER is proposed by Schlimmer [86] and HILLARY is proposed by Ida, et al [58].

Table 5.16: Classification Results for Mushroom Data

System	Training Sample	Classification Accuracy	Note
STAGGER	1,000	95%	asymptoted
HILLARY	1,000	95%	ten-trial average
Our approach	500	98.9%	ten-trial average
Our approach	5416	99.1%	one-trial

It should also be noted here that the data set has not been randomly arranged. If the first 1,816 instances are used for training and classification tests, one can always obtain 100% accuracy with only one simple rule:

$$odor = pungent \rightarrow class = poisonous.$$

In the first case of the testing, the weight of evidence of the significant association $[order = pungent, class = poisonous]$ in support of $class = poisonous$ will be infinity. In the actual experiments, such a setting has been intentionally avoided.

5.3.4 The Monk's Problems

The Monk's problems [93] are considered synthetic, although the third problem contains some noise. These sets of data are chosen because there is a thorough comprehensive comparison of many major systems using these data sets. Three data sets record three artificial problem domains related to robots which can be described by six attributes (see Table 5.17). The task is a binary classification problem: deciding whether or not a robot belongs to a class. For each problem, there are totally 432 instances of pre-classified robots. Only a subset of the 432 instances have been used in training.

The three problems are generated as follows:

Problem 1:

head shape = body shape $\vee$ jacket color = *red* $\Longrightarrow$ class 1

otherwise $\Longrightarrow$ class 2

Problem 2:

exactly two of the six attributes take on their first value $\Longrightarrow$ class 1

Table 5.17: Robot Attributes in the Monk's Problem

x1	head shape	round (1) / square (2) / octagon (3)
x2	body shape	round (1) / square (2) / octagon (3)
x3	is smiling	yes (1) / no (2)
x4	holding	sword (1) / balloon (2) / flag (3)
x5	jacket color	red (1) / yellow (2) / green (3) / blue (4)
x5	has tie	yes (1) / no (2)

otherwise $\Longrightarrow$ class 2

Problem 3:

(jacket color = *green* $\wedge$ holding = *sword*) $\vee$

(jacket color $\neq$ *blue* $\wedge$ body shape $\neq$ *octagon*) $\Longrightarrow$ class 1

otherwise $\Longrightarrow$ class 2

For the first problem, 124 instances are used in training, and the remaining used for testing. This problem is expected to be the easiest among the three, since it is basically a first order classification problem. Table 5.18 shows the significant associations detected from the first MONK's problem. Here, the highest order is set to be 3. A comparison of the first MONK's problem and Table 5.18 shows that the logic relationship regarding $Class = 1$ has been correctly captured. If the highest order is not bound, such patterns as $[Attr_1 = 1$ and $Attr_2 = 1$ and $Attr_5 = 1]$ and $[Attr_1 = 1$ and $Attr_2 = 1$ and $Attr_3 = 2$ and $Attr_4 = 3]$ may also be detected.

The second problem is believed to be the hardest among the three, since the combined values of all six attributes must be considered. In this case, 169 of 432 samples have been randomly selected for training. Due to the size of the sample and the consideration of overfitting, to find all the six-order associations is unlikely.

Table 5.18: Associations Detected from the First MONK's Problem

Class	Pattern
$Class = 1$	$x_5 = 1$
	$x_1 = 1$ and $x_2 = 1$
	$x_1 = 2$ and $x_2 = 2$
	$x_1 = 3$ and $x_2 = 3$

Partial information has to be used to provide evidence to classify a new instance not in the training set. For an algorithm which can correctly classify new instances in this deterministic problem, it must have deductive capability since the training set is far from complete with respect to the domain.

The third Monk's problem is similar to the first one, but 5% random noise in terms of misclassification has been added to the training set. The testing set remains clean. In the training set, there are 122 instances.

In terms of classification accuracy, it is obvious that the proposed method is among the best of the algorithms compared in [93]. Table 5.19 shows the comparison of this new approach with some other well-known algorithms. Connectionist algorithms are excluded from the comparison because neural networks are considered not suitable for data mining applications [1]. The new system emphasizes more on association discovery. Classification is treated as an application of discovered patterns at the request of the user.

Among the compared methods in Table 5.19, AQ based algorithms perform the best. AQ based algorithms generalize concepts from a seed in a deterministic domain. While they can solve the logic problems such as the Monk's problems, they may have a problem when applied to a much larger, noisy, real-world database.

Table 5.19: Classification Results for the Monk's Problems

System	MONK-1	MONK-2	MONK-3
New approach	100%	84.5%	100%
AQ15-GA	100%	86.8%	100%
AQ17-HCI	100%	93.1%	100%
AQ17-DCI	100%	100%	94.2
ID3	98.6%	67.9%	94.4
CN2	100%	69.0%	89.1%
ECOBWEB	82.7%	71.3%	68.0%

containing only probabilistic patterns. The new approach performs well compared with other probabilistic/information theory based algorithms. The misclassifications occurred in the second Monk's problem are mostly due to rejection caused by a weak weight of evidence. It suggests that a symbolic generalization mechanism is needed to deduct concept descriptions from the discovered event associations. Such a mechanism will improve the classification accuracy of the new system when dealing with deterministic/logic applications such as that in the Monk's problems.

5.4 Summary

This chapter evaluates the performance of the proposed method. A prototype system is implemented and tested with both synthetic and real-world databases. The results show that the new system is capable of detecting complicated patterns (associations) of different orders from a data set. It is also shown that this method is applicable to large databases. Using weight of evidence, the inference engine of the proposed system achieves flexible prediction. Also, its classification accuracy is comparable to that of other classification-oriented approaches.

Experiments in Section 5.2 are designed to evaluate the performance of the pattern discovery algorithm. Different data sets are used to test the different characteristics of the proposed system, including its ability to detect different order patterns, the meaningfulness and consistency of the detected patterns, and finally the computational expense.

Experiments on the data sets with previously known patterns such as the XOR data and the second sets of synthetic data show that the new method can correctly uncover the inherent patterns. With the XOR problem, the detected patterns clearly depict the logic relationship among the three variables. In the experiments on the second sets of synthetic data, patterns are detected in the presence of noise. The requirement for a compound event to be significant at different orders is also investigated. The proposed algorithm can discovery "all" the significant event associations inherent in a data set, as both positive and negative patterns of different orders can be successfully detected. The meaningfulness of the detected event associations is examined with the domain knowledge in the test with the zoo database. Although redundancy exists in the discovered patterns, all the patterns are found to be meaningful. The experiments with the first group of synthetic data sets help to evaluate the computational expense of the new method, especially when it is upscaled to deal with large data sets. It is found that the computing time of the new algorithm is almost linear to the sample size of the data set in the worst case. Obviously then, this allows the algorithm to be applied to large real-world databases. Actually in the test, the largest data set used contains one million instances with a file size over 25MB. In real-world databases, significant patterns are expected to be sparsely distributed in the feature space, thus different from the randomly generated data sets where each possible compound event has almost the same possibility. Sparse distribution of patterns allows an algorithm to apply heuristics to

truncate the search space. In the new system, candidate selection schemes will kick in to reduce the actual search space as evidenced by the experiment with the injury database in Section 5.2.4. The selected candidates occupy only a small fraction of all possible combinations of the domain. Intelligently truncating the searching space enables a system to analyze complicated problem domains without losing important information. Further, the same experiment shows a consistency of the patterns discovered over different subsets of the original data. Consistent patterns help to make better decisions on the execution of the domain, and they indicate that some fundamental regularities of the domain have been uncovered.

Regarding its analysis and inference aspects, the performance of the system is tested with four classification oriented data sets. The emphasis is on not only the classification accuracies, but also flexible prediction and the understanding of the domain. The experiments demonstrate that, even though the system is not designed specifically for classification tasks, its classification accuracies are comparable with those of other inductive systems. One would also note that the patterns discovered from the training set can precisely depict the relationships among the attributes thus providing a better understanding of the problem domains. As for flexible prediction, it is achieved through the use of the information measure, weight of evidence. Flexible prediction allows the user to focus on a certain slice of the entire pattern set.

With the zoo database, it is demonstrated that the proposed method is able to predict the value of any attribute given a set of observations. A creature instance can be classified according to its type or the number of its legs depending on what interests the user. Such flexible prediction is important for a data mining system in which the user's interest changes from time to time. As for the breast cancer data set, the patterns (in the form of an AHG) found in it show the importance

of understanding the problem at hand. The highest order of the patterns related to the class *malignant* is only 3. It suggests that, to determine whether or not a case is malignant, two observations are enough. Hence, we need only two lab tests to determine if there is a cancer. Obviously, this will improve system performance and simultaneously help cut healthcare budget intelligently. Indeed, understanding the problem sometimes is more important than mere the classification accuracy.

It should be noted that, in the discovery phase, it is not assumed that the attribute to which classification is going to be made is known. If such information is available, the weights of evidence can be calculated during the discovery process. If the absolute value of a weight of evidence is large enough (greater than a threshold or equal to infinity), the search of associations in that direction can then be terminated. Thus, the computing time will be further reduced. But on the other hand, the associations not related with the attribute to be classified can not be discovered. In such a case, the entire system is turned to a classification oriented system.

Chapter 6

Conclusion and Future Research

The research presented in this thesis is on a uniform framework for pattern discovery and analysis. It proposes a new system of pattern discovery, pattern representation and pattern analysis/inference for databases.

This system identifies statistically significant event associations inherent in a given data set, using residual analysis to test whether or not a pattern candidate is significantly deviated from a default log-linear model. Once a pattern is discovered, it is stored as a hyperedge in an attributed hypergraph. This attributed hypergraph will be regarded as a knowledge collection for future inference tasks. Significant event associations can be directly retrieved from the attributed hypergraph at the request of the user. Two other common inference problems, the best-N or rule induction problem, and the missing-value or classification problem have been studied. Associated algorithms are developed to solve these problems. An information measure, known as the weight of evidence, has been proposed to evaluate the evidence provided by a significant event association in support of, or against, an attribute taking on a certain value. Extended to higher order, the weight of evidence measure

has also been applied to classify a new observed object at the user's request.

The work described in this dissertation was motivated by the recognition of: (1) increasingly large amounts of raw data in databases which require the facilitation of an automatic pattern discovery and analysis system for a better understanding of the problem domain reflected by the data; (2) the pressing need to develop intelligent systems which are able to learn from data collections and make predictions; (3) the potential applications of knowledge discovery from databases and data mining in both scientific and business worlds; (4) the inability of most available discovery/learning algorithms to cope with large data collections which contain incomplete, inconsistent and/or inaccurate data; and (5) the application limitation of most existing systems which solve only a particular problem and therefore, not general enough to render an integrated discovery/analysis framework for real-world applications.

6.1 Summary of Contributions

A better way to summarize the contributions of the research is to evaluate the research outcome against the objectives as stated at the outset of this thesis. The following statements are an account corresponding to the points outlined in Section 1.2.2.

1. The current research brings forth a general and versatile framework from pattern discovery to inference process.

   Within this framework, knowledge (patterns) is extracted from raw data and represented by an attributed hypergraph. Patterns are regarded as significant event associations inherent in a data set. With this in mind, major tasks

in knowledge discovery and data mining can be formulated into one system, thus removing the need of looking pattern discovery, rule induction, classification and clustering as different topics. Supervised learning and unsupervised learning are also unified as the processes of discovering different kinds of patterns from data sets. Two fruitful components arising from this approach are: 1) a discovery engine for detecting significant patterns of different orders, and 2) a generalized weight of evidence based inference mechanism for different kinds of inference tasks. The former provides an automated pattern extraction solution for large databases while the latter allows the system to respond to the user's interest and give answers to his/her queries.

2. The statistical discovery algorithm is able to detect different order patterns in the presence of uncertainty.

   Because of its inherent statistical characteristics, the discovery method can handle noisy data properly. It does not require that the original data be correct or complete. It guarantees (with a fixed confidence level) that the detected patterns are significant. Unlike traditional statistical pattern recognition approaches which are variable-oriented, this new method works at the event level. It reveals not only if several attributes are related but also how they are related. Hence, this event-based approach gives more precise descriptions of the detected patterns in a way similar to how people describe concepts. This characteristic of transparency is one of the most desirable properties of AI.

3. The attributed hypergraph is proposed for the representation of complicated high order event associations.

An attributed hypergraph representation is a graphical representation which is able to distinguish between set connectivity and connectivity among elements. Hence, it is suitable to represent patterns of different orders. Traditional graphical representations such as trees and networks use pairwise relationship to represent data dependence. For complicated patterns in real-world database applications, these representations are inadequate. An attributed hypergraph is perhaps the simplest and the most direct and effective representation for different orders of inter-dependencies. Using attributed hypergraphs, both the qualitative relations and the quantitative relations are encoded. With a graph language, a good number of mature algorithms can be directly applied for pattern retrieval, combination and inference purposes. Pattern visualization is much easier with such a representation.

4. The weight of evidence measure makes it possible to integrate various inference tasks into one system. Thus, flexible prediction is achievable.

   One of the advantages of using weight of evidence is that the total weights of evidence provided by two patterns are "addable" if the two patterns are conditionally independent. Hence, on one hand, the evidence provided by a set of observations, positive and negative, can be combined for comparison when a new object is being classified against a specified attribute. On the other hand, the weight of evidence also depicts the strength of the association among various elements. When the association is transformed into an association rule, this measure then furnishes the strength of the rule. Unlike this measure, those used in most rule induction methods for comparing and selecting rules cannot be applied to classify new objects, thus crippling the system for achieving generality and applicability.

5. Empirical tests on synthetic and real-world data sets indicated that the system is superior to (at least comparable with) many existing algorithms in terms of efficiency of discovery, classification accuracy and scalability to large databases.

6. A pattern discovery and analysis software system has been prototyped. The potential applications of such a system is numerous.

   The prototyped system can detect significant patterns given a categorical or discrete-valued data collection and perform rule induction and classification tasks at the request of the user. This system works well especially in situations where no *a priori* knowledge is available, where the data sample is large, where the decision maker needs more information than that of a single attribute, and where different inference processes would be used to induce different attributes. Some possible applications of such a system are listed below:

   - Building knowledge bases for expert systems;
   - Analyzing large databases such as stock market records, connection records of telecommunication companies and basket data of department stores for prediction and forecasting;
   - Discovering patterns for quality control, diagnosis, and decision support systems; and
   - Encoding and retrieving information from various data sets or databases.

In addition to these contributions, there are some other points worth mentioning:

- By discovering high order patterns, polythetic learning is achieved. This function enables the system to analyze highly complicated data. At the same

time, the system does not apply pre- or post-pruning stages which are used by many other systems to achieve polythetic analysis even though they are very time consuming and domain dependent.

- The candidate selection technique discussed in Section 3.6.2 helps to avoid exhaustive search for detecting high order significant patterns. This technique speeds up the pattern discovery process by eliminating uninformative pattern candidates in the early stages, thus allowing it to be applied to large databases.

- Since we cannot assume the completeness of a data set (database), negative information has to be considered in addition to positive patterns. The negative information is used to eliminate uninformative candidates in the discovery process, as well as in the inference process. This is different from some other pattern discovery systems which detect only positive patterns, a case that may lead to the loss of information and incorrect classification.

- The event based approach in this research makes it unnecessary in the inference stage to go back to the original data set for parameter estimation, or to record large correlation matrices, as the variable-based approaches. Going to the event level also contributes to reducing the computational complexity. This makes it possible to test only the possibly significant events other than exhaustively searching all the possible combinations.

6.2 Suggested Future Research

Several interesting problems related to this research are still open for future investigation. The following is a list of some possible directions presented as the

conclusion of this thesis.

Generalization of event associations into concept descriptions

In deterministic and game-playing domains such as the Monk's problems [93] discussed in Section 5.3, data can be represented by a small number of simple rules. The ability of generalizing discovered event associations into such kind of rules will not only simplify the representation but also increase the classification accuracy. One example is that the data set is generated by a rule $(A = B) \longrightarrow Class\ 1$. The discovery process will give the event associations such as $(A_1 = B_1) \longrightarrow Class\ 1$, $(A_2 = B_2) \longrightarrow Class\ 1$, etc. It is desirable that the original rule can be explicitly stated, especially at high order. The second Monk's problem is a good example. A possible solution is to feed the discovered patterns into an AQ-based algorithm so as to generate rule descriptions. For probabilistic domains, a careful investigation is even more important. Though the discovery process filters out most of the noise, the concept generation procedure should study how to use the statistical measures rigorously when they are incorporated with each discovered event association.

Discretization of continuous attributes and mixed-mode data analysis

Discussion in the current research is limited to discrete attributes or pre-discretized continuous attributes. The choice of discretization algorithms for continuous attributes is very important for the performance of the discovery process. Although the earlier work [99] [21] yielded some insight into this problem, further investigation is definitely necessary. In mixed-mode data analysis, one chooses either to discretize the continuous attribute and then apply discrete analytical algorithms, or to analyze mixed-mode data directly without discretization. The trade-off of the two choices is worth studying. To

my best knowledge, few good methods are available to analyze mixed-mode data directly. Undoubtedly this opens a new research direction.

Other default statistical models for adjusted residual analysis

In Section 3.5, an independence model is assumed to be the default model for the residual analysis with a fixed sample size. Although it is stated that other log-linear models can also be chosen, no details, especially the parameter estimates, have been discussed. In Appendix A, more details of hierarchical models are given. With other models and a changing data sample size, one has to derive new formulas to calculate the maximum-likelihood estimates of the parameters so as to conduct residual analysis. To be able to derive the maximum-likelihood estimates, an understanding of the problem domain and skills of statistical analysis are required.

More efficient implementation

It is possible to implement the algorithms in a way so as to take advantage of parallel and/or distributed computation. When the database is distributedly located and the number of attributes is large, parallel and distributed implementation will speed up the process significantly. The current implementation does not assume any format of the data. If the data are stored in a (relational) database, the DBMS (DataBase Management System) can be involved to provide more efficient resource management.

There are of course many other worthwhile research possibilities that are not included in the list. I believe that because of the challenging topics and the tremendous potential applications, pattern discovery and analysis in databases will continue to receive more and more attention in both the scientific and the industrial worlds.

Appendix A

Hierarchical Models

The discussion of hierarchical models will first be illustrated by three-way contingency tables involving three random variables. The conclusions can be easily extended to high dimensional cases. For more details, interested readers can refer to the original paper by Goodman [42] and the books by Haberman [46] [47] and Andersen [5].

As defined in Section 3.5.3, the hierarchical models with respect to three random variables A, B, and C. have the form:

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} + \lambda_{ik}^{AC} + \lambda_{jk}^{BC} + \lambda_{ijk}^{ABC} \tag{A.1}$$

where e_{ijk} is the expectation of the actual count o_{ijk} of the cell (i, j, k), $1 \leq i \leq L$, $1 \leq j \leq W$ and $1 \leq k \leq H$, in the three-way table. All the λ_+^+'s are unique parameters that satisfy:

$$\sum \lambda_i^A = \sum \lambda_j^B = \sum \lambda_k^C = \sum_i \lambda_{ij}^{AB} = \sum_j \lambda_{ij}^{AB} = \sum_i \lambda_{ik}^{AC} = \sum_k \lambda_{ik}^{AC}$$

$$= \sum_j \lambda_{jk}^{BC} = \sum_k \lambda_{jk}^{BC} = \sum_i \lambda_{ijk}^{ABC} = \sum_j \lambda_{ijk}^{ABC} = \sum_k \lambda_{ijk}^{ABC} = 0. \qquad \text{(A.2)}$$

In hierarchical models except the saturated model, some λ-parameters are set to 0. The hierarchical restriction is followed that if any λ-parameter with superscript S is set to 0, then any λ-parameter of the same or higher order is set to 0. Here a λ-parameter has the same or higher order if its superscript contains each letter of S.

It is also assumed that the sample size M is fixed, so that the table has a multinomial distribution and the cell probability is denoted as p_{ijk}. In addition to the above definitions, the following marginal notations are repeatedly used:

$$o_i^A = \sum_j o_{ij}^{AB} = \sum_k o_{ik}^{AC} = \sum_j \sum_k o_{ijk}$$

and

$$o_{ij}^{AB} = \sum_k o_{ijk},$$

$$M = \sum_i o_i^A = \sum_i \sum_j o_{ij}^{AB} = \sum_i \sum_j \sum_k o_{ijk}.$$

The following enumeration lists all the nine possible hierarchical models in a three-way table. With each case, the physical meaning of the model and the correspondent maximum-likelihood estimate of the parameters are given if explicit formula exists. All of the nine possible models are log-linear.

Case 1. The saturated model

In the saturated model, no restrictions are imposed on Eqn. A.1, so

$$\hat{e}_{ijk} = o_{ijk} \qquad \text{(A.3)}$$

The saturated model is not a very interesting model. We cannot really get anything from such a model.

Case 2. No three-factor interaction

With this model, we have

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} + \lambda_{ik}^{AC} + \lambda_{jk}^{BC}$$

The no three-factor interaction model is always used to compare relative risks such as $\log(e_{ij1}/e_{ij2})$. In such a model, the marginal estimates have explicit expressions:

$$\begin{aligned} \hat{e}_{ij}^{AB} &= o_{ij}^{AB} \\ \hat{e}_{ik}^{AC} &= o_{ik}^{AC} \\ \hat{e}_{jk}^{BC} &= o_{jk}^{BC} \end{aligned}$$

Solution of $\hat{e}_{ijk}^{ABC}$ requires iterative methods such as the Newton-Raphson or the iterative proportional fitting algorithm. There is no correspondent independence explanation of this model.

Case 3. Conditional independence

This case contains three models. Correspondingly, one of the three two-factor interactions is set to zero. Just for illustration purposes, we just consider one situation in which λ_{jk}^{BC} is set to zero, that is:

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB} + \lambda_{ik}^{AC}$$

by definition of the hierarchical models, the three-factor interaction λ_{ijk}^{ABC} is zero.

This model is equivalent to the hypothesis that given A, B and C are condi-

tionally independent [8]. Hence, the probability

$$p_{jk|i}^{BC|A} = p_{j|i}^{B|A} \cdot p_{k|i}^{C|A}$$

holds. In this equation.

$$p_{jk|i}^{BC|A} = \frac{p_{ijk}}{p_i^A}$$

is the conditional probability of B_j and C_k given A_i,

$$p_{j|i}^{B|A} = \frac{p_{ij}^{AB}}{p_i^A}$$

is the marginal conditional probability of B_j given A_i, and

$$p_{k|i}^{C|A} = \frac{p_{ik}^{AB}}{p_i^A}$$

is the marginal conditional probability of C_k given A_i.

The maximum-likelihood estimate $\hat{e}_{ijk}$ of e_{ijk} satisfies the equation

$$\hat{e}_{ijk} = \frac{o_{ij}^{AB} o_{ik}^{AC}}{o_i^A} = M \cdot \hat{p}_i^A \hat{p}_{j|i}^{B|A} \hat{p}_{k|i}^{C|A}. \tag{A.4}$$

where

$$\begin{aligned} \hat{p}_i^A &= \frac{o_i^A}{M} \\ \hat{p}_{j|i}^{B|A} &= \frac{o_{ij}^{AB}}{o_i^A} \\ \hat{p}_{k|i}^{C|A} &= \frac{o_{ik}^{AC}}{o_i^A}. \end{aligned} \tag{A.5}$$

For adjusted residual analysis, the maximum-likelihood estimate $\hat{c}_{ijk}$ of the vari-

ance c_{ijk} is given by

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \left(1 - \frac{o_{ij}^{AB}}{o_i^A}\right)\left(1 - \frac{o_{ik}^{AC}}{o_i^A}\right). \tag{A.6}$$

The conditional independence model, as well as the mutually independence model and the mode of two variables independent of the third, are of the most interest in the hierarchical models.

Case 4. Two variables independence of the third

As in the previous case, there are three similar models, in each of which two two-factor interactions are set to zero. We use the following model to show the general properties of this group, i.e.

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_k^C + \lambda_{ij}^{AB}.$$

This model holds when the variable C is independent of the variable pair (A, B). Note that C is independent of (A, B) if

$$p_{ijk} = p_{ij}^{AB} p_k^C$$

For this model, the maximum-likelihood estimate of e_{ijk} satisfies the equation

$$\hat{e}_{ijk} = \frac{o_{ij}^{AB} o_k^C}{M} = M \cdot \hat{p}_{ij}^{AB} \hat{p}_k^C \tag{A.7}$$

where

$$\begin{aligned} \hat{p}_{ij}^{AB} &= \frac{o_{ij}^{AB}}{M} \\ \hat{p}_k^C &= \frac{o_k^C}{M}, \end{aligned} \tag{A.8}$$

and the maximum-likelihood estimate of the variance of the residual is

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \left(1 - \frac{o_{ij}^{AB}}{M}\right)\left(1 - \frac{o_k^C}{M}\right). \tag{A.9}$$

Case 5. All variables mutually independent

This is the model we discussed in Section 3.5.3 as the default model of the system. We will not repeat the discussion here.

The left cases are less common in practice and residual analysis.

Case 6. All categories of one variable equiprobable given the other two

Similarly there are three models in this case which suggests that the probabilities of the events of one variable are equal given the other two variables. One of them is represented as

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B + \lambda_{ij}^{AB}$$

The model holds if and only if given A and B, each categories of C is equally probable, so that

$$p_{k|ij}^{C|AB} = \frac{1}{H}.$$

The maximum-likelihood estimate of e_{ijk} is given by

$$\hat{e}_{ijk} = \frac{1}{H} o_{ij}^{AB} = \frac{1}{H} M \hat{p}_{ij}^{AB} \tag{A.10}$$

The maximum-likelihood estimate of c_{ijk}

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \frac{1}{H}\left(1 - \frac{1}{H}\right). \tag{A.11}$$

Case 7. All categories of one variable equiprobable given the other two, and the

other two variables independent

There are three models in this case. We use only one to show the results. It is

$$\log e_{ijk} = \lambda + \lambda_i^A + \lambda_j^B$$

It holds when A and B are independent and, given A and B, each categories of C is equally probable. Thus

$$p_{ij}^{AB} = p_i^A p_j^B$$

and

$$p_{k|ij}^{C|AB} = \frac{1}{H}$$

The maximum-likelihood estimate of e_{ijk}

$$\hat{e}_{ijk} = \frac{1}{H}\frac{o_i^A o_j^B}{M} = \frac{1}{H} M \hat{p}_i^A \hat{p}_j^B \tag{A.12}$$

where

$$\begin{aligned} \hat{p}_i^A &= \tfrac{o_i^A}{M} \\ \hat{p}_j^B &= \tfrac{o_j^B}{M} \end{aligned} \tag{A.13}$$

The maximum-likelihood estimate of c_{ijk}

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \left[1 - \frac{1}{H} + \frac{1}{H}\left(1 - \frac{o_i^A}{M}\right)\left(1 - \frac{o_j^B}{M}\right)\right]. \tag{A.14}$$

Case 8. Given one variable, all combinations of categories of the other two variables are equally probable

In this case, only one main effect factor remains. As an illustration, we choose:

$$\log e_{ijk} = \lambda + \lambda_i^A.$$

Given A, the model holds when all the combinations of the events of B and C have equal probabilities, then

$$p_{jk|i}^{BC|A} = \frac{1}{WH}.$$

The maximum-likelihood estimate of e_{ijk}

$$\hat{e}_{ijk} = \left(\frac{1}{WH}\right) o_i^A = \left(\frac{1}{WH}\right) M\hat{p}_i^A \tag{A.15}$$

The maximum-likelihood estimate of c_{ijk}

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \left(1 - \frac{1}{WH}\right). \tag{A.16}$$

Case 9. All combinations of the three variables are equally probable

Basically this model represents a uniform distribution. All variable factors are set to be zero. So that

$$\log e_{ijk} = \lambda.$$

It suggests that all the combinations of A, B and C are equally likely. Thus,

$$p_{ijk} = \frac{1}{LWH}$$

The maximum-likelihood estimate of e_{ijk}

$$\hat{e}_{ijk} = \frac{M}{LWH} \tag{A.17}$$

The maximum-likelihood estimate of c_{ijk}

$$\hat{c}_{ijk} = \hat{e}_{ijk} \cdot \left(1 - \frac{1}{LHW}\right). \tag{A.18}$$

To extend the results of three-way contingency table to high-way table is not difficult. Three principal changes occur as the number of variables increases [47]. The number of possible hierarchical models increases very rapidly; a decreasing fraction of hierarchical models have explicit expressions for maximum-likelihood estimates; and interpretation of hierarchical models becomes increasingly complex.

The following theorem gives the sufficient and effective condition of a model having explicit maximum-likelihood estimates. It is first stated by Goodman [41] and proved by Haberman [46] and Andersen [4].

Theorem A.1 (Goodman 1968) *There is an explicit solution to the likelihood equations and the estimated expected numbers are direct functions of the sufficient marginals, if and only if the model is decomposable.*

In the case of mutual independence, no matter how high the order is, the model is always decomposable, hence, the explicit formula for maximum-likelihood estimates always exist. Actually, Haberman [46] gives a general format of the maximum-likelihood estimates of the expected count and the variance of the residual.

Bibliography

[1] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *VLDB-92*, pages 560–573, 1992.

[2] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonon, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, 1996.

[4] A. H. Andersen. Multidimensional contingency tables. *Scandinavian Journal of Statistics*, 1:115–127, 1974.

[5] E. B. Andersen. *The Statistical Analysis of Categorical Data*. Springer-Verlag, third edition, 1994.

[6] S. Avner. Extraction of comprehensive symbolic rules from a multilayer perception. *Engineering Applications*, 9(2):137–143, 1996.

[7] C. Berge. *Hypergraph: Combinatorics of Finite Sets*. North Holland, 1989.

[8] M. W. Birch. Maximum likelihood in three-way contingency tables. *Journal of Royal Statistical Society*, B-25:220–233, 1963.

[9] E. Bloedorn and R. S. Michalski. Data-driven constrictive induction in AQ17-DCI: A method and experiments. Technical report, Machine Learning and Inference Lab., Center for Artificial Intelligence, George Mason University, 1991.

[10] I. Brakto and I. Kononenko. Learning diagnostic rules from incomplete and noisy data. In B. Phelps, editor, *Interactions in Artificial Intelligence and Statistical Methods*, pages 142–153. Technical Aldershot, 1987.

[11] I. Bratko. Machine learning in artificial intelligence. *Artificial Intelligence in Engineering*, 8(3):159–164, 1993.

[12] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Belmont, 1984.

[13] Y. Cai, N. Cercone, and J. Han. Attribute-oriented inductional databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 12, pages 213–228. AAAI Press / MIT Press, 1991.

[14] J. G. Carbonell and P. Langley. Learning, machine. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, volume 1, pages 464–488. Wiley, 1987.

[15] B. Cestnik, I. Kononenko, and I. Bratko. ASSISTANT 86: A knowledge elicitation tool for sophisticated users. In I. Bratko and N. Larvac, editors, *Progress in Machine Learning: Proc. of EWSL 87*, pages 31–45, 1987.

[16] K. C. C. Chan. *Induction Learning in the Presence of Uncertainty*. PhD thesis, Department of Systems Design, University of Waterloo, 1989.

[17] K. C. C. Chan and A. K. C. Wong. APACS: A systems for automated pattern analysis and classification. *Computational Intelligence*, 6(3):119–131, 1990.

[18] K. C. C. Chan and A. K. C. Wong. A statistical technique for extracting classificatory knowledge from databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 6, pages 107–123. AAAI Press / MIT Press, 1991.

[19] P. Cheeseman, J. Kelly, M. Self, J Stutz, W. Taylor, and D. Freeman. AutoClass: A Bayesian classification system. In *Proc. of the 5th Int'l Conf. on Machine Learning*, 1988.

[20] J. Y. Ching. Class-dependent discretization of continuous attributes for inductive learning. Master's thesis, Department of Systems Design, University of Waterloo, 1993.

[21] J. Y. Ching, A. K. C. Wong, and K. C. C. Chan. Class-dependent discretization for inductive learning from continuous and mixed-mode data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):631–641, 1995.

[22] D. K. Y. Chiu. *Pattern Analysis Using Event-Covering*. PhD thesis, Department of Systems Design, University of Waterloo, 1986.

[23] D. K. Y. Chiu and A. K. C. Wong. Synthesizing knowledge: A cluster analysis approach using event-covering. *IEEE Trans. on Systems, Man and Cybernetics*, 16(2):251–259, 1986.

[24] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, 14(3):462–467, 1968.

[25] P. Clark and T. Niblett. Induction in noisy domains. In I Bratko and N. Larvac, editors, *Progress in Machine Learning: Proc. of the 2nd European Working Session on Learning*, pages 11–30. Sigma Press, 1987.

[26] P. Clark and T. Niblett. Learning if-then rules in noisy domains. In B. Phelps, editor, *Interaction in AI and Statistical Methods*, pages 154–168. Technical Aldershot, Hants. England, 1987.

[27] G. F. Cooper and E. Herskovits. A Baysian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[28] D. R. Cox and E. J. Snell. A general definition of residuals. *Journal of Royal Statistical Society*. B-30:248–265, 1968.

[29] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley, New York, 1966.

[30] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI Press / MIT Press, 1996.

[31] D. H. Fisher. Conceptual clustering, learning from examples, and inference. In *Proc. of the 4th Int'l Workshop on Machine Learning*, pages 38–49, 1987.

[32] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[33] D. H. Fisher and P. K. Chan. Statistical guidance in symbolic learning. In *Annals of Mathematics and Artificial Intelligence*, volume 2, pages 135–147. J. C. Baltzer, 1990.

[34] D. H. Fisher and K. B. McKusick. An empirical comparison of ID3 and back-propagation. In *Proc. of the 11th Int'l Joint Conf. on Artificial Intelligence*, volume 1, pages 788–793, 1989.

[35] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.

[36] R. M. Fung and S. L. Crawford. Constructor: A system for the induction of probabilistic models. In *Proc. of the 8th National Conf. on Artificial Intelligence, AAAI'90*, volume 2, pages 762–769, 1990.

[37] R. M. Fung, S. L. Crowford, L. Appelbaum, and R. Tong. An architecture for probabilistic concept-based information retrieval. In *Proc. of the 13th Int'l Conf. on Research and Development in Information Retrieval*, 1990.

[38] S. N. Gelfand, C. S. Ravishankar, and E. J. Delp. An interactive growing and pruning algorithm for classification tree design. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2):163–174, 1991.

[39] J. S. Gero and R. Stanton. *Artificial Intelligence Development and Applications*. North Holland, 1987.

[40] E. Godehardt. *Graphs as Structural Models: The Application of Graphs and Multigraphs in Cluster Analysis*. Friedr, Vieweg & Sohn Verlagsgesellschaft, 1990.

[41] L. A. Goodman. The analysis of cross-classified data: Independence, quasi-independence and interactions in contingency tables with and without missing entries. *Journal of the American Statistical Association*, 63:1091–1131, 1968.

[42] L. A. Goodman. The multivariate analysis of qualitative data: Interactions among multiple classifications. *Journal of the American Statistical Association*, 65:226–256, 1970.

[43] R. M. Goodman and P. Smyth. Decision tree design from a communication theory standpoint. *IEEE Trans. on Information Theory*, 34(5):979–994, 1988.

[44] R. M. Goodman and P. Smyth. Information-theoretic rule induction. In *Proc. of the 8th European Conf. on Artificial Intelligence*, pages 357–362, 1988.

[45] S. J. Haberman. The analysis of residuals in cross-classified tables. *Biometrics*, 29:205–220, 1973.

[46] S. J. Haberman. *The Analysis of Frequency Data*, volume 4 of *Statistical Research Monographs*. University of Chicago Press, 1974.

[47] S. J. Haberman. *Analysis of Qualitative Data*, volume 1. Academic Press, 1978.

[48] H. J. Hamilton and D. R. Fudger. Estimating DBLearn's potential for knowledge discovery in databases. *Computational Intelligence*, 11(2):280–296, 1995.

[49] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in database: An attribute-oriented approach. In *Proc. of the 18th Int'l Conf. on Very Large Data Base, VLDB'92*, 1992.

[50] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40, 1993.

[51] J. Han and Y. Fu. Attributed-oriented induction in data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 16, pages 399–421. AAAI Press / MIT Press, 1996.

[52] F. Hayes-Roth and J. McDermott. An interference matching technique for inducting abstractions. *Communications of the ACM*, 21(5):401–410, 1978.

[53] D. Heckerman. Bayesian networks for knowledge discovery. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 11, pages 273–305. AAAI Press / MIT Press, 1996.

[54] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–244, 1995.

[55] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, volume 1, chapter 3, pages 77–109. MIT Press, Cambridge, MA, 1986.

[56] M. Holsheimer and A. Siebes. Data mining: The research for knowledge in databases. Technical Report CS-R9406, CWI, 1995.

[57] X. Hu and N. Cercone. Learning in relational databases: A rough set approach. *Computational Intelligence*, 11(2):323–338, 1995.

[58] W. Iba, W. James, and P. Langley. Trading off simplicity and coverage in incremental concept learning. In *Proc. of the 5th Int'l Conf. on Machine Learning*, pages 73–79, Ann Arbor, Michigan, 1988.

[59] T. Kohonen. *Self-Organizing Maps.* Springer. Berlin, Germany, 1995.

[60] R. Kowalski. *Logic for Problem Solving.* North Holland, 1979.

[61] P. Langley. Machine learning and concept formation. *Machine Learning*, 2(4):99–102, 1987.

[62] P. Langley and J. G. Carbonell. Approaches to machine learning. *Journal of the American Society for Information Science*, 35(5):306–316, 1984.

[63] R. Michalski and R. Chilauski. Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study involving soybean pathology. *Int'l J. Man-Machine Studies*, 12:63–87, 1980.

[64] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111–161, 1983.

[65] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The AQ15 inductive learning system: An overview and experiments. Technical Report UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

[66] R. S. Michalski and P. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(4):396–409, 1983.

[67] T. M. Michell. Version spaces: A candidate elimination approach to rule learning. In *Proc. of the 5th Int'l Joint Conf. on Artificial Intelligence*, pages 305–316, 1977.

[68] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[69] A. Mrozek. Rough sets and dependency analysis. *Journal of Man-Machine Studies*, 30(4):448–457, 1989.

[70] S. Muggleton. *Inductive Logic Programming*. Academic Press, 1992.

[71] P. M. Murph and D. W. Aha. *UCI Repository of Machine Learning Databases*. Dept. of Information and Computer Science, Univ. of California: Irvine, 1991.

[72] H. Narazaki, M. Yamamoto, and T. Watanabe. Reorganizing knowledge in neural networks: An explanation mechanism for neural networks in data classification problems. *IEEE Trans. on Systems, Man and Cybernetics*, 26(1):107–117, 1996.

[73] T. Niblett. Constructing decision trees in noisy domains. In I. Bratko and N. Larvac, editors. *Progress in Machine Learning: Proc. of EWSL 87*, pages 67–78, 1987.

[74] D. B. Osteyee and I. J. Good. *Information, Weight of Evidence, the Singularity between Probability Measures and Signal Detection*. Springer-Velag, Berlin, Germany, 1974.

[75] G. Pagallo and D. Haussler. Two algorithms that learn DNF by discovering relevant features. In *Proc. of Int'l Workshop on Machine Learning*, pages 119–123. Morgan Kaufmann, 1989.

[76] Z. Pawlak. *Rough Sets – Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991.

[77] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[78] J. R. Quinlan. Discovering rules by induction from large collections of examples. In D. Michie. editor, *Expert Systems in the Micro-Electronic Age*, pages 168–201. Edinberg University Press, 1979.

[79] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106. 1986.

[80] J. R. Quinlan. Simplifying decision trees. *International Journal of of Man-Machine Studies*. 27(2):221–234. 1987.

[81] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[82] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. Inductive knowledge acquisition: A case study. In J. R. Quinlan, editor, *Applications of Expert Systems*. pages 157–173. Addison-Wesley, Australia, 1987.

[83] D. E. Rumelhart. G. E. Hinton. and R. J. Williams. Learning representations by backpropagation errors. *Nature*, 323(6088):533–536, 1986.

[84] A. R. Safavian and D. Landgrede. A survey of decision tree classifier methodology. *IEEE Trans. on Systems. Man, and Cybernetics*, 21(3):660–674, 1991.

[85] W. S. Sarle. Neural networks and statistical models. In *Proc. of the 9th Annual SAS Users Group Int'l Conf.*, pages 1538–1550, Cary, NC, 1994. SAS Institute.

[86] J. C. Schlimmer. *Concept Acquisition Through Representational Adjustment*. PhD thesis, Dept. of Information and Computer Science, University of California, Irvine, 1987.

[87] A. Silberschatz, M Stonebraker, and J. D. Ullman. Database systems: Achievements and opportunities. *Communications of ACM*, 34(10):110–120, 1991.

[88] H.A. Simon. Why should machines learn? In J. G.; Michalski, R. S.; Carbonell and T. M. Mitschell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume 1, chapter 2, pages 25–38. Tioga Publishing Co., 1983.

[89] P. Smyth and R. M. Goodman. Information theoretic approach to rule induction from database. *IEEE Trans. on Knowledge and Data Engineering*, 4(4):301–316, 1992.

[90] P. Smyth, R. M. Goodman, and C. Higgins. A hybrid rule-based/Bayesian classifier. In *Proc. of the 9th European Conf. on Artificial Intelligence*, pages 610–615, Stockholm, Sweden, 1990.

[91] D. Spiegelhalter and S. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990.

[92] C. J. Thornton. *Techniques in Computational Learning*. Chapman & Hall, London, UK, 1992.

[93] S. B. Thrun. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, 1991.

[94] Q. R. Wang and C. Y. Suen. Large tree classifier with heuristic search and global training. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):91–102, 1987.

[95] Y. Wang and A. K. C. Wong. Representing discovered patterns using attributed hypergraph. In *Proc. of the Second International Conference on Knowledge Discovery and Datamining, KDD'96*, Portland, OR., August 1996.

[96] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. of the National Academy of Sci.*, 87(23):9193–9196. 1990.

[97] A. K. C. Wong and K. C. C. Chan. Learning from examples in the presence of uncertainty. In *Proc. of the Int'l Computer Science Conf. '88: Artificial Intelligence: Theory and Applications*, 1988.

[98] A. K. C. Wong and D. K. Y. Chiu. An event-covering method for effective probabilistic inference. *Pattern Recognition*, 20(2):245–255, 1987.

[99] A. K. C. Wong and D. K. Y. Chiu. Synthesizing statistical knowledge form incomplete mixed-mode data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9:796–805, 1987.

[100] A. K. C. Wong and T. S. Liu. A decision-directed clustering algorithm for discrete date. *IEEE Trans. on Computer*, 26(1):75–82, 1977.

[101] A. K. C. Wong and D. C. C. Wang. DECA: A discrete-valued data clustering algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 1(4):342–349, 1979.

[102] A. K. C. Wong and Y. Wang. Discovery of high order patterns. In *Proc. of the 1995 IEEE Int'l Conf. on SMC*, volume 2, pages 1142–1148, Vancouver, BC, Canada, 1995.

[103] A. K. C. Wong and Y. Wang. High order pattern discovery from discrete-valued data. *IEEE Trans. on Knowledge and Data Engineering*, to appear.

[104] W. A. Woods. Knowledge representation: What's important about it? In N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 44–79. Springer-Verlag, New York, 1987.

[105] N. Wrigley. *Categorical Data Analysis for Geographers and Environmental Scientists*. Longman, 1985.

[106] Y. Xiang, S. K. M. Wong, and N. Cercone. Quantifying uncertainty of knowledge discovered from databases. In W. Ziarko, editor, *Rough Sets, Fuzzy Sets and Knowledge Discovery*, pages 63–73. Springer-Verlag, 1993.

[107] J. Zhang. Selecting typical instances in instance-based learning. In *Proc. of the 9th International Machine Learning Conference*, pages 470–479, Aberdeen, Scotland, 1992.

[108] W. Ziarko. The discovery, analysis, and representation of data dependencies in databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 195–209. AAAI/The MIT Press, 1991.

[109] W. Ziarko. Rough sets and knowledge discovery: An overview. In W. Ziarko, editor, *Rough Sets, Fuzzy Sets and Knowledge Discovery*, pages 11–15. Springer-Verlag, 1993.

[110] J. M. Żytkow and J. Baker. Interactive mining of regularities in databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, chapter 2, pages 31–53. AAAI Press / MIT Press, 1991.