Chapter 1

Introduction

The current information age is characterized by an extraordinary expansion of data that are being generated and stored about all kinds of human endeavors. An increasing proportion of these data is recorded in the form of computer databases, in order that the computer technology may easily access it. The availability of very large volumes of such data has created a problem of how to extract from them useful, task-oriented knowledge. An enormous proliferation of databases in almost every area of human endeavor has created a great demand for new, powerful tools for turning data into useful, task-oriented knowledge. In efforts to satisfy this need, researchers have been exploring ideas and methods developed in machine learning, pattern recognition, statistical data analysis, data visualization, neural nets, etc. These efforts have led to the emergence of a new research area, frequently called data mining and knowledge discovery [93, 95].

1.1. Data Mining and Knowledge Discovery in Databases (KDD)

Data mining and Knowledge Discovery in Databases (KDD) [37] have been attracting a significant amount of research, industry, and media attention of late. Historically, the notion of finding useful patterns in data has been given a variety of names, including data mining, knowledge extraction, information discovery, information harvesting, data archaeology, and data pattern processing. The term \textit{data mining} has mostly been used by statisticians, data analysts, and the management information systems (MIS) communities. It has also gained popularity in the database field. The phrase \textit{Knowledge Discovery in Databases} was coined at the first KDD workshop in 1989 to emphasize that knowledge is the end product of a data-driven discovery. KDD has evolved, and continues to evolve from the intersection of research in such fields as databases, machine learning, pattern recognition, statistics, artificial intelligence and reasoning with uncertainty, knowledge acquisition for expert systems, data visualization, machine discovery, scientific discovery, information retrieval and high-performance computing.
KDD software systems incorporate theories, algorithms, and methods from all of these fields.

KDD refers to the overall process of turning low-level data into high-level knowledge and data mining refers to a particular step in this process that consists of applying data analysis and discovery algorithms [97]. The additional steps in the KDD process [37], such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data.

A generally accepted definition of KDD is given by Fayyad et al. [37] as: the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Here, data are a set of facts $F$ (for example, cases in a database), and pattern is an expression $E$ in some language $L$ describing the facts in a subset $F_E$ of $F$ i.e. data or a model applicable to the subset. $E$ is called a pattern if it is simpler than the enumeration of all facts in $F_E$. The term process implies that KDD comprises many steps, which involve data preparation, search for patterns, knowledge evaluation, and refinement, all repeated in multiple iteration. The term nontrivial, means that some search or inference is involved; that is, it is not a straightforward computation of predefined quantities like computing the average value of a set of numbers. The discovered patterns should be valid on new data with some degree of certainty. We also want patterns to be novel (at least to the system and preferably to the user) and potentially useful, that is, lead to some benefit to the user or task. Finally, the patterns should be understandable, if not immediately then after some post-processing. A measure of certainty, measuring the validity of discovered patterns, is a function $C$ mapping expressions in $L$ to a partially or totally ordered measure space $M_C$. An expression $E$ in $L$ about a subset $F_E \subset F$ can be assigned a certainty measure $c = C(E, F)$. Novelty of patterns can be measured by a function $N(E, F)$ with respect to changes in data or knowledge. Patterns should potentially lead to some useful actions, as measured by some utility function $u = U(E, F)$ mapping expressions in $L$ to a partially or totally ordered measure space $M_U$. The goal of KDD is to make patterns understandable to humans. This is measured by a function $s = S(E, F)$ mapping expressions $E$ in $L$ to a partially or totally ordered measure space $M_S$. Interestingness of a pattern combines validity, novelty, usefulness, and understandability, and can be expressed as $i = I(E, F, C, N, U, S)$ which maps expressions in $L$ to a measure space $M_i$. 

2
A pattern \( E \in L \) is called knowledge if for some user-specified threshold \( i \in M_i \), \( I(E, F, C, N, U, S) > i \). One can select some thresholds \( c \in M_C \), \( s \in M_S \), and \( u \in M_U \), and term a pattern \( E \) knowledge

\[
\text{iff } C(E, F) > c, \text{ and } S(E, F) > s, \text{ and } U(E, F) > u.
\]

The role of interestingness is to threshold the huge number of discovered patterns and report only those which may be of some use. There are two approaches to designing a measure of interestingness of a pattern, viz., objective and subjective. The former uses the structure of the pattern and is generally used for computing rule interestingness. However often it fails to capture all the complexities of the pattern discovery process. The subjective approach, on the other hand, depends additionally on the user who examines the pattern. Two major reasons why a pattern is interesting from the subjective (user-oriented) point of view are [116]: (i) unexpectedness: when it is 'surprising' to the user, and (ii) actionability: when the user can act on it to her/his advantage. For example, user's concepts are incorporated in the knowledge discovery process in order to discover subjectively interesting rules [24, 88].

The KDD process is interactive and iterative, involving numerous steps with many decisions made by the user (see Fig.1.1). A practical view of the KDD process,
emphasizing the interactive nature of the process and broadly outline some of its basic steps is given in [19, 37, 55, 104].

The field of automated learning and discovery [134] often called data mining, machine learning, or advanced data analysis- is under going a revolution. The progressing computerization of professional and private life, paired with a sharp increase in memory, processing and networking capabilities of today’s computers, make it now more than ever possible to gather and analyze vast amounts of data. For the first time ever, the people all around the world are connected to each other electronically through the Internet, making available large amount of online data at an astonishingly increasing pace. Recently data mining is becoming much more important as the number of databases and database size keeps growing.

Data mining is a very unique and challenging process [144]. That is, not straightforward analysis nor does it necessarily equate with machine learning [20]. The control flow of the data mining process [40] is shown in Fig.1.2. A data mining session is usually an interactive process of data mining query submission, task analysis, data collection from the database, interesting pattern search, and findings presentation.

![Diagram of data mining process](image)

Fig.1.2: Control flow of the data mining process [40].

It is often the case that the knowledge discovered by a data mining algorithm needs to undergo some kind of post-processing. Discovered-knowledge post-processing usually aims at improving the comprehensibility and/or the interestingness of the knowledge to
be shown to the user. This step may involve, for instance, the selection of the most interesting rules, among the discovered rule set, (see Fig.1.3). There are two main motivations for such post-processing. First, when the discovered rule set is large, we often want to simplify it - i.e., to remove some rules and/or rule conditions - in order to improve knowledge comprehensibility for the user. Second, we often want to extract a subset of interesting rules, among all discovered ones. The reason is that although many data mining algorithms were designed to discover accurate, comprehensible rules, most of these algorithms were not designed to discover interesting rules, which is a rather more difficult and ambitious goal [39].

Fig.1.3: An overview of the knowledge discovery process [39].

1.1.1. Data-Mining Tasks

One of the greatest strengths of data mining is reflected in its range of methodologies that can be applied to a host of problem sets [144]. The data-mining methods can be viewed as consisting of three primary algorithmic components [37]: (i) The model: the function of the model (e.g., classification, clustering) and its representational form (e.g., linear discriminates, neural networks). A model contains parameters that are to be determined from the data. (ii) The preference criterion: A basis for preference of one model or set of parameters over another, depending on the given data. The criterion is usually some form of goodness-of-fit function of the model to the data, perhaps tempered by a smoothing term to avoid overfitting, or generating a model with too many degrees of freedom to be constrained by the given data. (iii) The search algorithm: The specification of an algorithm for finding particular models and parameters, given the data, model(s), and a preference criterion. The two high-level primary goals of data mining in practice tend to be prediction and description.
Prediction involves using some variables or fields in the database to predict unknown or future values of other variables of interest, and description focuses on finding human-interpretable patterns describing the data. An overview of the data mining tasks is given in [55, 104, 144].

**Classification** is learning a function that maps (classifies) a data item into one of several predefined classes [143]. In this task the data being mined is divided into two mutually exclusive and exhaustive data sets, the training set and the test set. The data mining algorithm has to discover rules by accessing the training set only. In order to do this, the algorithm has access to the values of both the predicting attributes and the goal attribute of each example (record) in the training set [35]. Examples of classification methods used as part of knowledge discovery applications include the classifying of trends in financial markets [4] and the automated identification of objects of interest in large image databases [82].

A classification structure is a set of hierarchically related classes, where hierarchically higher classes subsume their lower counter-parts. Building such a structure is a difficult task for the knowledge engineer [96], and automated discovery of such hierarchies is one of the important research areas in KDD.

**Regression** is learning a function that maps a data item to a real-valued prediction variable [37].

**Production Rules Mining** is the discovery of the connections between a class and the features of the data in the training set. A production rule is in the form of "\( P_1 \land P_2 \land \ldots \land P_n \rightarrow D \)" which means decision / class D tend to appear with the features (premises) \( P_1, P_2, \ldots, P_n \) in the target data. Discovering production rules at multiple levels gives the relationship between the different classes in the target data.

**Clustering** is often an important initial step of several in the data mining process [8, 68]. It is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data [67]. Clustering maps a data item into one of several clusters [97], where clusters are natural groupings of data items based on similarity metrics or probability density models. Several efforts [34, 52, 99, 131, 132, 150, 151] have been made for hierarchical clustering of large databases. Typical pattern clustering activity involves the following steps [35]:

6
• **Pattern representation** refers to the number of classes, the number of available patterns, and the number, type, and scale of the features available to the clustering algorithm. There are no theoretical guidelines that suggest the appropriate patterns and features to use in a specific situation. A good pattern representation can often yield a simple and easily understood clustering; a poor pattern representation may yield a complex clustering whose true structure is difficult or impossible to discern.

• **Pattern proximity** is usually measured by a distance function defined on pairs of patterns. A variety of distance measures are in use in the various communities. The most popular metric for continuous features is the *Euclidean distance* which is a special case of the Minkowski metric. The Euclidean distance has an intuitive appeal as it is commonly used to evaluate the proximity of objects in two or three-dimensional space. It works well when a data set has "compact" or "isolated" clusters.

• **The grouping step** can be performed in a number of ways. The output clustering can be hard or fuzzy [22] (where each pattern has a variable degree of membership in each of the output clusters). Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity.

• **Data abstraction** is the process of extracting a simple and compact representation of a data set. Here, simplicity is either from the perspective of automatic analysis (so that a machine can perform further processing efficiently) or it is human-oriented (so that the representation obtained is easy to comprehend and intuitively appealing).

• **Cluster validity analysis**, a clustering structure is valid if it cannot reasonably have occurred by chance or as an artifact of a clustering algorithm. When statistical approaches to clustering are used, validation is accomplished by carefully applying statistical methods and testing hypotheses.

1.1.2. Challenges for KDD

The following are the challenges for KDD [154]:
• **Larger Databases:** Databases with hundreds of fields and tables, millions of records, and multi-gigabyte size are quite commonplace, and terabyte ($10^{12}$ bytes) databases are beginning to appear.

• **High Dimensionality:** A very large number of fields (attributes, variables) makes the dimensionality of the problem high. A high dimensional dataset creates problems in terms of increasing the size of the search space, it increases the chances that a data mining algorithm will find spurious patterns that are not valid in general. Approaches to this problem include methods [47] to reduce the effective dimensionality of the problem and the use of prior knowledge to identify irrelevant variables.

• **Changing Data and Knowledge:** Rapidly changing (non-stationary) data may make previously discovered patterns invalid. In addition, the variables measured in a given application database may be modified, deleted, or augmented with new measurements over time. Possible solutions include incremental methods [36, 48, 145] for updating the patterns and treating change as an opportunity for discovery by using it to cue the search for patterns of change only.

• **Missing and Noisy Data:** Important attributes may be missing if the database was not designed with discovery in mind. Possible solutions include more sophisticated statistical strategies to identify hidden variables and dependencies.

• **Complex Relationships between Fields:** Hierarchically structured attributes or values, relations between attributes, and more sophisticated means for representing knowledge about the contents of a database will require algorithms that can effectively utilize such information.

• **Understandability of Patterns:** In many applications it is important to make the discoveries more understandable by humans. Possible solutions include graphical representations [72, 81], rule structuring with directed a-cyclic graphs [45], and techniques for visualization of data and knowledge [50, 110].

1.2. **Representation of Discovered Knowledge and KDD**

In the following sections we discuss various KDD approaches with reference to representation of discovered knowledge:
1.2.1. Production Rules

The most predominant representation of the discovered knowledge is the standard Production Rules (PR) in the form

\[
\text{If } \text{<Precondition>} \text{ Then } \text{<Decision>} \quad \ldots (1.1)
\]

i.e.,

\[ P \rightarrow D. \]

where the decision in right part of the rule, sometimes called consequent, tend to appear with the preconditions in left part of the rule, sometimes called antecedent, in the target data. That is, the decision part is asserted when the precondition part is satisfied. Production rules have the following five major properties [112]:

(a) Adaptive determination of the best sequence of rules to execute.
(b) The incorporation of human skills in conditional "if...then..." rules.
(c) An increase in skill proportional to the enlargement of the knowledge base.
(d) The ability to solve complex problems by selecting rules and combining the result.
(e) Explanation of their result by reversing the line of reasoning. Rules are good for representing knowledge which occurs as a large number of discrete facts.

Rule classification schemes include ID3, C4.5, C5, rough sets, classification and regression trees and association rules. Here, induction classification rules [47, 87, 91, 101] are inferred from examples through the recursive partitioning of the output space into regions mapped by the input space of the attributes. Most inductive learning approaches [71] focus on inducing specific rules by generalizing from existing examples. The rules induced are then used to classify new events.

The association rule mining technique, pioneered by Agarwal et al. [1, 2] was initially developed for finding correlations in shopping basket. For a database maintained by a store, an association rule involving product X and product Y may be of the form X→Y (support 5%, confidence 80%). This means that 5% of all database transactions contain the data item X and Y, and 80% of transactions that have the item Y. Mining interesting or optimal association rules [5, 23, 79, 141] is useful in domains where a constraint-based rule miner produces too many rules or requires too much time.

An interesting Inductive learning algorithm, known as ILA [135] was originally designed for noise-free domains. The induction bias used in ILA is to extract a rule
from a set of promising rules for a class if and only if the rule covers the greatest proportion of the number of positive and none of the negative examples. ILA2 [136] is an extension of ILA by using a new evaluation metric that handles uncertainty in a given dataset.

Fuzzy production rule is an important way of imprecise knowledge representation and approximate reasoning [140]. The goal of fuzzy systems [32, 42, 60, 139] has always been to model human expert knowledge and to produce systems that are easy to understand.

Fuzzy representation bridges the gap between symbolic and non-symbolic data and plays a very important role in different fields - database summary discovery [83], problems which can be expressed under the forms of constraints and criteria [18], generalized association rule mining [63, 142], object-oriented hierarchy of fuzzy classes [137], classification rules [62, 113], and medical data [22].

1.2.2. Rules with Exceptions

A production rule system can capture much of the simple human problem solving capability. However, not all the human problem solving methods are easily representable in the Production Rules systems [11].

Exceptions, which focus on a very small portion of a data set and represent systematic deviation from common sense rules, have been ignored or discarded as noise in machine learning but the goal of KDD is broader, and it is always interesting to discover exceptions, as they challenged the existing knowledge [126]. Discovery of pairs of an exception rule and a common sense rule under threshold scheduling has been successful in efficient discovery of interesting rules [129]. Exception can take an important role in making critical decisions [100] and reduces the complexity of the discovered knowledge [89]. Therefore, a simple rule with exceptions is better than a complicated one without exceptions, especially when the exceptions occur rarely.

An increasing number of studies is devoted to exception detection. Examples of such studies are competitive exception learning [6, 7], mining exception rules based on a measure which estimates interestingness of discovered rules [65], learning rules that have high frequency exceptions [80], KB revision that will result in an extended and domain-adapted ontology [84], and handling approach uses enhancement knowledge in
a class definition [149]. Further, exception rule discovery considered in detail by Suzuki [125 - 130] and the results mainly concern interestingness measure, reliability evaluation, practical application, parameter reduction, and knowledge representation. Also, several algorithms [17, 33, 49, 66, 86, 89, 148] are proposed for exception-rule discovery.

Ripple Down Rule (RDR) is an effective knowledge acquisition methodology proposed by Compton and Jansen [26] which constrains the interactions between the expert and a shell to acquire only correct knowledge [25, 28]. RDR are rules with hierarchical exceptions [44, 77, 78, 103, 115], used in knowledge acquisition because they provide a well intelligible and modifiable representation such as simplification and transformation of RDRs into flat lists of rules and ripple down rules sets.

Logically, RDRs [30, 43] have exactly the same representational and deductive capabilities as standard production rules. Where they differ is in the strong context provided by the ‘if-true’ and ‘if-false’ links. Although RDR works well, it is only suitable for the problem of providing a single classification for a set of data. Multiple Classification Ripple-Down Rules (MCRDR) is an extension of RDR which allows multiple independent classifications. MCRDR may provide a basis for building a general problem solver for a range of problems beyond classification [74, 124]. The Reuse of Knowledge in RDR Knowledge Based Systems is based on the MCRDR knowledge acquisition and representation technique [109].

RDR has been extended to heuristic search [9], fuzzy logic [21, 92], a more general RDR system for construction tasks [27], distinguish between knowledge acquisition methods [29], Nested RDR (NRDR) to allow experts to use predicate logic in expressing their rules [36], document retrieval [75], configuration [106], conceptual lattice [107, 110], provide the user with sufficient information and browsing tools to develop their own line of reasoning [108], and to learn classification taxonomies from a classification knowledge based system [122].

1.2.3. Hierarchical Structures

The most common way of representing knowledge in classification schemes has been in hierarchies. Hierarchies are useful for incorporating knowledge about relationships between items, the detail with which items can be described; hierarchies are useful
within well-known, mature fields, containing well-defined, clear class boundaries; hierarchies are useful where the relationships between entities within the hierarchy are well developed. Hierarchies are a simpler version of ontologies, albeit very useful [51, 133, 152]. Hierarchies allow the user to view the discovered rules at different levels of details, and to focus his/her attention on those interesting aspects [90]. Hierarchies give comprehensible knowledge structure that carries more information than simple collection of rules.

Hierarchical structures [56] are commonly found in datasets of various applications. Previous machine learning methods mainly focused on utilizing the hierarchy of class labels. The classification task is divided into a sequence of sub-class classification tasks where each sub-class is associated with a certain node in the class label hierarchy. Other than class hierarchy, some datasets may contain features with categorical domain values organized in a hierarchical structure. Such a hierarchy reflects existing knowledge about feature values and reveals their inter-relationships in different levels.

As one of the most important background knowledge, concept hierarchy plays a fundamentally important role in data mining. It organize data and concepts in hierarchical forms or in certain partial order, which helps expressing knowledge and data relationships in databases in concise, high level terms, and thus, plays an important role in knowledge discovery processes [54]. Learning by discovering concept hierarchies can be used to solve complex problems in machine learning [153].

Automatic generation of hierarchies can be a post-processing step, but integrating hierarchy generation process into the data mining algorithm can dramatically reduce the execution time [119]. Studies have shown that taxonomies are not only effective in revealing the valuable information buried in a mass of unstructured data, they tremendously reduces the search time to find them as well [120], and therefore hierarchies, being a good mechanism for organizing large quantities of information, are ubiquitous.

Hierarchies are often used to encode knowledge, and have been used in a variety of ways for text classification [76], mining association rules [53, 64, 118], evolutionary learning [3, 52, 111], attribute-oriented tree ascension [61], decision support [16], fuzzy modeling [147], searching in large databases [73, 114], and various other tasks where similarity plays role [15, 46, 105, 151].
Several efforts have been made in the recent past towards automated discovery of hierarchical structures in large databases [41, 85, 138].

1.2.4. Hierarchical Structure with Exceptions

The present work focuses on the development of discovery algorithms using the following representation of discovered knowledge:

1.2.4.1. Censored Production Rules (CPRs)

The standard production rules however, are unable to handle exceptions and do not exhibit variable precision.

The major shortcomings of an ordinary logic based reasoning system based on standard production rule If P Then D, is that you cannot tell much about the way you want it to perform its task. For example, you cannot give the following instructions [94]:

- Give me a reasonable answer immediately even if it is somewhat general and if there is enough time give me a more specific answer.
- Give me a reasonable answer immediately. If there is enough time tell me you are more confident in the answer or change your mind and give me a better answer.
- Give me only a highly certain answer even if it is somewhat general, and if there is enough time then give me a more specific answer.
- Give me a reasonable answer immediately even if it is somewhat less certain, and if you have enough time then give me a more specific answer.

A system having represented real-world knowledge should also be capable of handling these types of requirements for natural and efficient reasoning. In the real world, both humans and computers often have to reason using insufficient, incomplete, or tentative premises. Moreover, both are subject to constraints of time and memory. Variable precision logic (VPL) [94] is concerned with problems of reasoning with incomplete information, subject to resource constrains and the problem of reasoning efficiently with exceptions. VPL offers mechanisms for handling trade-off between the precision of inferences and computational efficiency of driving them. Specificity and certainty are two aspects of precision. Certainty refers to the degree of belief in a statement, whereas specificity refers to the degree of detail of a description. A system that gives more specific answers given more time (or resources in general) is called a “variable
specificity system°. A system that gives more certain answers given more time is called a "variable certainty system". There can be various combinations of the two systems, reflecting that specificity and certainty are inversely related; we can gain specificity at the expense of certainty, or vice-versa.

As an extension of standard production rule, Michalski and Winston [94] have suggested the Censored Production Rule (CPR), analogous to Winston [146], as an underlying representational and computational mechanism to capture the uncertain and imprecise knowledge about the real world i.e. to enable logic based systems to exhibit variable precision in which certainty varies while specificity stays constant. A CPR has the following form

\[
\text{If } P \text{ Then } D \text{ Unless } C \quad \ldots(1.2)
\]

where C (Censor) is the exception condition. Such rules are employed in situations in which the conditional statement 'If P Then D' holds frequently and the assertion C holds rarely. By using a rule of this type we are free to ignore the censor (exception) conditions, when the resources needed to establish its presence are tight or there is simply no information available as to whether it holds or does not hold. As time permits the Censor condition C is evaluated establishing the conclusion D with higher certainty if C does not hold or simply changing the polarity of D to \(\sim D\) if C holds.

Let us now give a more quantitative definition of a censored production rule:

\[
P \Rightarrow D \mid C \quad \ldots(1.3)
\]

where P is a premise, D is a decision, and C is a censor. Although the unless operator, \(\mid\), is logically equivalent to the commutative exclusive-or operator, the unless operator has an expositive aspects which is not commutative. In order to capture the asymmetry precisely, let us associate two parameters, \(\gamma_1\) and \(\gamma_2\), with rule (1.3).

\[
P \Rightarrow D \mid C : \gamma_1, \gamma_2 \quad \ldots(1.4)
\]

Both \(\gamma_1\) and \(\gamma_2\) are point probabilities, one indicating the strength of the relationship between P and D, and the other, between P and C. Now consider the following sets: \(\Omega\) is a finite sample of events; \(\Omega_P\) is the set of events for which P holds; \(\Omega_{PD}\) is the subset of events for which both P and D hold; \(\Omega_{PC}\) is a subset of events for which both P and C hold. Given these sets, the parameters \(\gamma_1\) and \(\gamma_2\) are defined as follows:
\[
\gamma_1 = \frac{\Pr[P,D]}{\Pr[P]} = \Pr[D|P] \approx \frac{|\Omega_{PD}|}{|\Omega_D|} \quad \ldots (1.5)
\]
\[
\gamma_2 = \frac{\Pr[P,C]}{\Pr[P]} = \Pr[C|P] \approx \frac{|\Omega_{PC}|}{|\Omega_C|} \quad \ldots (1.5)
\]

where \(|\Omega_i|\) denotes the cardinality of \(\Omega_i\). Also we assume that \(\Omega_D \cap \Omega_C = \emptyset\) and \(\Omega_D \cup \Omega_C = \Omega_P\); thus \(\Pr[P|D] + \Pr[C|P] = 1\).

Assuming that there are significantly more events for which both \(P\) and \(D\) hold, then \(\Omega_{PD}\) is considerably larger than the set \(\Omega_{PC}\):

\[
|\Omega_{PD}| \gg |\Omega_{PC}| \quad \ldots (1.6)
\]

Thus, taking (1.5) into consideration, we have

\[
\gamma_1 \gg \gamma_2 \quad \ldots (1.7)
\]

From the logical point of view, according to our interpretation of the unless operator, the sets \(\Omega_{PD}\) and \(\Omega_{PC}\) must be disjoint. Consequently, the sum \(\gamma_1 + \gamma_2\) must always equal 1. Thus knowing \(\gamma_1\), it is easy to compute \(\gamma_2\), therefore rule (1.4) can be simplified:

\[
P \Rightarrow D \mid C: \gamma \quad \ldots (1.8)
\]

where \(\gamma\) stands for \(\gamma_1\). Thus rule (1.8) permits us to generate a number of inferences of varying certainty, depending on what is given and what is unknown. The relationship between certainty and knowledge has an important operational consequence. To illustrate, consider two cases:

(a) \(P\) is known to hold, and there is insufficient time or space to determine \(C\). A system can infer the decision \(D\), with certainty \(\gamma\).

(b) \(P\) is known to hold, and there are sufficient resources to determine \(C\). A system can determine \(C\) and subsequently conclude \(D\) or \(-D\), depending on \(C\), with certainty 1.

A censored production rule having more than one censor conditions, say, \(C_1, C_2, \ldots, C_n\) is called Multiple Censored Production Rule (MCPR) denoted as:

\[
\text{If } P \text{ Then } D \text{ Unless } (C_1 \lor C_2 \lor \ldots \lor C_n) \quad \ldots (1.9)
\]
1.2.4.2. Hierarchical Censored Production Rules (HCPRs)

To address the various problems and shortcomings with CPRs system, Bharadwaj and Jain [11] have introduced a concept of Hierarchical Censored Production Rules (HCPRs) system as an underlying representational and computational scheme for variable precision logic. They defined a HCPR as a CPR augmented with specificity and generality information. The general form of a HCPR is

\[
\text{Decision IF } \text{<condition> UNLESS <censor> GENERALITY <general info> SPECIFICITY <specific info> \ldots (1.10)}
\]

i.e. \( D \ (:- \ P) (\leq C) (G\% G) (\$ S) \)

where the symbols ‘:-‘, ‘\( \leq \)‘, ‘G\%‘, and ‘\$‘ are used for ‘If’, ‘Unless’, ‘Generality’ and ‘Specificity’ operators respectively and symbols ‘P’, ‘C’, ‘G’, and ‘S’ denote the corresponding information relegated with them. Symbol ‘D’ represents the decision part of the rule.

It is possible for related HCPRs with different levels of specificity to be treated in a tree structure. It is advantageous to do so because then we have related chunks of knowledge. Systemically arranging available information is very helpful in its manipulation and use. This hierarchical linking of HCPRs gives rise to the concept of an HCPRs tree. It is again emphasized that this tree structure is possible due to the generality and specificity information in the HCPR. A further advantage, as will be seen, is that redundancy is minimized in the listing of properties in the IF part of HCPR.

HCPR can be made to exhibit variable precision in the reasoning such that both the certainty of belief in a conclusion and its specificity may be controlled by the reasoning process. HCPRs based system provides a general framework for intelligent systems that supports variable precision logic, excellent mechanism for handling exceptions, various machine learning paradigms (Symbolic [11, 69], Genetic [10, 58] and Neural [12]) and various inference mechanisms (CF [11], DST [14], Fuzzy logic [98], and Parallel model [13]). HCPRs based systems would have numerous applications where decision
must be taken under resource (time/ space) constraints and with uncertain information-
examples range from medical expert systems for operating rooms to domestic robots.

In addition to the above representation we have used Production Rules with Fuzzy
Hierarchy [18, 83, 137, 142] and Hierarchical Ripple-Down Rules (proposed extension
of RDR by incorporating hierarchical structure) as underlying representation of
discovered knowledge.

1.3. Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 presents an algorithm for
automated discovery of Multiple Censored Production Rules (MCPRs) based on
inductive learning approach. In chapter 3, we have introduced a novel concept of
Frequency matrix (Freq) for summarizing large datasets. Chapter 4 presents a novel
method for discovering production rules with fuzzy hierarchy that takes into account
the general view of inheritance. In chapter 5, an integrated approach to the discovery of
exceptions and fuzzy hierarchy generation is taken to develop a scheme for the
discovery of production rules with exceptions and fuzzy hierarchy. As an extension to
Ripple-Down Rules (RDRs) technique, chapter 6 presents a knowledge acquisition
technique for discovering Hierarchical Ripple-Down Rules (HRDR). Finally, chapter 7
concludes with accomplishments and future research directions.