BUILDING BAYESIAN NETWORK BASED EXPERT SYSTEMS FROM RULES

by

SARAVANAN THIRUMURUGANATHAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2010

ACKNOWLEDGEMENTS

I am very grateful to my advisor Dr. Manfred Huber. His encyclopedic knowledge and boundless energy have been a tremendous inspiration to me. It was a privilege to work with him for the past two years. I would also like to thank Dr. Gergely Zaruba and Dr. David Levine for being in my thesis committee. Special thanks to Dr.Zaruba for his incisive comments and tips on presentation and research.

I would also like to thank my family and friends who have made my life a delight. I would like to extend my gratitude towards everyone in Computer Science department for the wonderful experience I had here.

July 19, 2010

ABSTRACT

BUILDING BAYESIAN NETWORK BASED EXPERT SYSTEMS FROM RULES

Saravanan Thirumuruganathan, MS

The University of Texas at Arlington, 2010

Supervising Professor: Manfred Huber

Combining expert knowledge and user explanation with automated reasoning in domains with uncertain information poses significant challenges in terms of representation and reasoning mechanisms. In particular, reasoning structures understandable and usable by humans are often different from the ones for automated reasoning and data mining systems.

Rules are a convenient and human understandable way to express domain knowledge and build expert systems. Adding certainty factors to these rules presents one way to deal with uncertainty in rule based expert systems. However such systems have limitations in accurately modeling the domain. A Bayesian Network, on the other hand, is a probabilistic graphical model that allows accurate modeling of a domain and automated reasoning. But inference in Bayesian Networks is harder for humans to comprehend.

In this thesis, we propose a method to combine these two frameworks to build Bayesian Networks from rules and derive user understandable explanations in terms of

these rules. Expert specified rules are augmented with strength parameters for antecedents and are used to derive probabilistic bounds for the Bayesian Network's conditional probability table. The partial structure constructed from the rules is fully learned from the data. The thesis also discusses methods for using the rules to provide user understandable explanations, identify incorrect rules, suggest new rules and perform incremental learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

There are multiple frameworks to perform reasoning in domains with uncertain information. An ideal framework must have convenient mechanisms to perform theoretically consistent and automated reasoning. If human users are involved, it must be able to generate user understandable explanations for its predictions. If the expert system also involves a domain expert, it should have mechanisms to elicit and store the domain knowledge of the expert.

It is well known that experts are most comfortable in specifying their domain knowledge using sets of rules because rules model experts' decision making process naturally. Research has shown that explanation of the diagnosis in terms of the expert specified rules is much more intuitive to the user than a set of raw values [Shortliffe and Buchanan, 1984]. By adding certainty factors to rules, we can enable them to also model uncertainty in the domain. Such a system will have convenient mechanisms to elicit expert knowledge and explain the reasoning process in a user understandable way using the expert specified rules. [Shortliffe and Buchanan, 1984].

Rule based systems have limited ability to completely model the domain. A typical rule base is not exhaustive and hence only few of the dependencies that exist in the domain are modeled. Since rule based expert systems rely entirely on the rule base for prediction, the modeling of the domain remains inexact. Moreover certainty factors expect all rules to be specified in the exact direction in which they will be reasoned. It is

not possible to use a causal rule to perform diagnosis as rule inversion is not defined in the framework. This means that a diagnostic problem solving system must only have diagnostic rules. Mixing of predictive and diagnostic rules in the same rule set results in inconsistencies. Certainty factors also make additional assumptions about the rules that undermine their formal consistency. In addition, certainty factor based expert systems are static – they consist of a set of rules and certainty factors which do not change during the operation of the system. This means that the availability of additional data will not result in fine tuning of the system. The system expects the set of rules to be exhaustive and, as a result, if a set of observations are not handled by the existing rules then the system cannot make any meaningful diagnosis based on the partial data.

Bayesian Networks, on the other hand, are an example of a probabilistic graphical model that has mechanisms to accurately model the domain's dependencies and perform fully automated reasoning. They have a mathematically consistent way to specify uncertainty in the system. They allow a combination of expert knowledge and data where an expert can specify dependencies among nodes or partial network structure and the complete structure and parameters are learned from data. If the expert is not available to provide us with the structure, it can also be learned from the data. Given a network, it is very easy to perform automated reasoning and for the network to make predictions based any a given set of evidence. In addition it is possible to make predictions about any node in the network based on the evidence. There are multiple algorithms that adapt the network based on additional data so that it can fit the data very well.

In practice, however, it is well known that experts have problems specifying their domain knowledge using probability and likelihood. This means that without data it is

very hard for the expert to encode his entire domain knowledge in a Bayesian Network. In addition, it is hard to concisely explain the inference process in a user understandable way.

In this thesis, we propose a mechanism that combines the advantages of both frameworks. The expert specifies his domain knowledge using rules and certainty factors. These rules are used to bootstrap a partial Bayesian Network that is then fully learned from the available data. Using the Bayesian Network, we can perform fully automated reasoning in a consistent way and also generate the rationale for the prediction using rules.

To further increase the descriptive power of the rules, we augment the rules specified by the expert with strength parameters and use it to estimate the residual evidence of the related rules that can be inferred from the original rule. This allows us to make maximum use of the expert specified rules to bootstrap the network. Based on the constructed network, we also introduce mechanisms for performing inference in the Bayesian Network and use the expert specified rules to generate user understandable explanations for the network's predictions. In addition to this, the proposed approach also makes it possible to perform incremental learning as new data arrives. Furthermore, using the information obtained from the data, the system provides mechanisms to identify bad rules and suggest new rules that can provide better explanations for predictions.

The rest of this thesis is organized in the following fashion. Chapter 2 describes some of the related research in certainty factors, Bayesian Networks and Dempster Shafer theory. Chapter 3 describes the approach that this thesis takes to combine the advantages of the individual frameworks to construct a Bayesian Network that can generate

explanations in a user understandable fashion. Chapter 4 discusses the implementation details and the experiments performed in two different domains. Chapter 5 provides conclusions and future work.

CHAPTER 2

RELATED WORK AND TECHNICAL BACKGROUND

There have been multiple attempts in the past to build expert systems that can perform automated uncertainty reasoning and generate user understandable explanations. This chapter discusses two of the prominent frameworks to build expert systems for uncertain domains – rule based systems with certainty factors and Bayesian Networks. The certainty factor framework has convenient ways to elicit experts' domain knowledge but its ability to perform automated reasoning is limited. Bayesian Networks, on the other hand, provide mechanisms to perform automated reasoning. However, intuitively explaining the predictions made to the users is hard. After discussing both the frameworks and their relative advantages and disadvantages, this chapter also discusses Dempster Shafer theory and the concept of evidence.

2.1 Rule-Based Systems With Certainty Factors

2.1.1 Adding Uncertainty to Rules

Rule based expert systems represent the knowledge of a domain using a set of rules. Rules can be considered to represent the relationships between entities/facts in the domain. The rules can be causal (from causes to effect) or diagnostic (from effects to causes). The certainty-factor (CF) model is a convenient framework to manage uncertainty in a rule-based system. It was originally designed for the MYCIN expert system [Shortliffe and Buchanan 1975]. It soon became the standard approach for

uncertainty management in rule-based systems due to the convenient way it provides to elicit rules and confidence on the rules from domain experts.

A diagnostic rule-based system, such as MYCIN, contains rules of the form "if e then h". Here, e denotes evidence and h denotes the hypothesis. An expert expresses uncertainty in diagnostic rules by attaching a single certainty factor to each rule. In the original interpretation, Certainty Factors represent the expert's change in belief in the hypothesis h given the evidence e. A positive value indicates that the expert's belief in h given e increases and a negative value indicates that the expert's belief decreases. It must be noted that a certainty factor does not represent a person's absolute degree of belief in h given e. In other words, certainty factors are not the same as probability values.

2.1.2 Combining Rules

The certainty factor framework has two operators to combine different rules under certain conditions. If two rules have the same consequent, then a new rule can be created by combining the two rules. This is called a parallel combination function. Assuming CF1 and CF2 are the certainty factors for the two rules, the certainty factor of the new rule is given by,

$$CF = \begin{cases} CF_1 + CF_2 - CF_1 CF_2 & If\ CF_1, CF_2 \geq 0 \\ CF_1 + CF_2 + CF_1 CF_2 & If\ CF_1, CF_2 < 0 \\ \dfrac{CF1 + CF2}{1 - min(|CF1|, |CF2|)} & otherwise \end{cases}$$

If there are two rules where the evidence of the first rule is the hypothesis of the second, i.e. rules of the form "If A then B" and "If B then C", they can be combined to a new rule of the form "If A then C". This is called the serial combination of rules. Assuming CF1 and CF2 are the certainty factors for the two rules, the certainty factor of the new rule is given by,

$$CF = \begin{cases} CF_1 CF_2 & if\ CF_1 > 0 \\ 0 & if\ CF_1 \leq 0 \end{cases}$$

More details about Certainty Factors can be found in [Heckerman and Shortliffe, 1992] and [Heckerman, 1992] .

2.1.3 Probabilistic Interpretation Of Certainty Factors

For a diagnostic rule "if e then h", the certainty factor represents the expert's change in belief in the hypothesis h given the evidence e and not the absolute probability. [Heckerman, 1986] proved that the original probabilistic interpretation given in [Shortliffe and Buchanan, 1984] is inconsistent with the rules of probability. He went on to derive a new probabilistic interpretation of certainty factors based on their axioms that is also consistent with the framework of probability. The new formulation gives a probabilistic interpretation of certainty factors based on the likelihood ratio,

$$\lambda(\mathrm{h}, \mathrm{e}) = \frac{\mathrm{p}(\mathrm{e}|\mathrm{h}, \xi)}{\mathrm{p}(\mathrm{e}|\neg\mathrm{h}, \xi)}$$

[Heckerman, 1986] showed that any monotonic transformation of the likelihood ratio produces a valid probabilistic interpretation of certainty factors. For this thesis, we will use the following interpretation:

$$\mathrm{CF}\ (\mathrm{h} \rightarrow \mathrm{e}|\xi) = \frac{\mathrm{p}(\mathrm{h}|\mathrm{e}, \xi) - \mathrm{p}(\mathrm{h}|\xi)}{\mathrm{p}(\mathrm{h}|\xi)\ (1 - \mathrm{p}(\mathrm{h}|\mathrm{e}, \xi))\ +\ \mathrm{p}(\mathrm{h}|\mathrm{e}, \xi)(1 - \mathrm{p}(\mathrm{h}|\xi))}$$

Where CF ($h \rightarrow e|\xi$) is the certainty factor for the rule "if e then h" given by an expert with background knowledge ξ. $p(h|\xi)$ is the expert's probability (degree of belief) for h given ξ . $p(h|e, \xi)$ is the expert's probability for h given evidence e and ξ.

2.1.4 Problems Using Certainty Factors for Complex Domains

Certainty factors, while very useful, have limited expressive power to accurately model real world dependencies. Heckerman [Heckerman and Shortliffe, 1992] showed that the parallel and serial combination functions impose assumptions of conditional independence on the propositions involved in the combinations. In particular, when we use the parallel-combination function to combine CFs for the rules "if e1 then h" and "if e2 then h," we assume implicitly that e1 and e2 are conditionally independent given h and ¬h. Similarly, when we use the serial-combination function to combine CFs for the rules "if a then b" and "if b then c," we assume implicitly that a and c are conditionally independent, given b and ¬b. Heckerman also showed that the combination functions for disjunction and conjunction impose specific forms of conditional dependence on the propositions involved in the combinations. Similarly, he showed that certain series of operations in the framework violate the modularity property.

Besides their implicit assumptions and formal inconsistencies, rule based expert systems have other limitations. In particular, in a certainty factor based expert system diagnostic and causal rules cannot exist in the same rule base as their interaction will result in inconsistent results. Reversing of rules is not defined in the framework. This means that a rule based system with causal rules cannot be used for diagnostic problem solving. In addition, certainty factors are subjective values specified by the expert and there is no way in the framework to use data to tweak the values. Moreover, it is not possible to perform inference based on partial observation.

[Heckerman and Shortliffe, 1992] contains additional discussion of the most important problems with the framework.

2.2 Bayesian Networks

Certainty factor based rule based systems can be considered heuristic models as they use an expert's change in belief instead of mathematically consistent probability values. Bayesian Networks are an example of probabilistic graphical models which use graph and probability theory to manage uncertainty in reasoning.

A Bayesian Network is a directed acyclic graph (DAG) whose nodes represent the variables in the problem domain and edges represent direct probabilistic dependencies among nodes. Lack of edges between nodes corresponds to conditional independence. Each node in a Bayesian Network is associated with a set of probability distributions given by the conditional probability table.

Conditional independence relationships among nodes in Bayesian Networks are determined by the notion of d-separation [Koller and Friedman, 2009]. Additionally, a node is independent of its ancestors given its parents. This means that, we can represent the joint distribution in a more compact way using Bayesian Networks.

Bayesian Networks are a rigorous framework that allows accurate modeling of the domain and provides mechanisms for fully automated reasoning. It is possible to use a Bayesian Network to represent causal relationships between nodes in the domain and then use it to perform diagnostic problem-solving. More details can be found in [Koller and Friedman, 2009].

2.2.1 Inference

Inference is the most common operation performed on a Bayesian Network. Probabilistic inference can be defined as computing the conditional probability distribution over values of unobserved nodes given the values of observed nodes. By

taking advantage of conditional probability assumptions, we can perform inference very effectively in most of the cases. Inference can be done by both exact and approximate methods.

A common use of Inference is to perform diagnostic problem solving given the observation of the effects. The most likely cause that results in the observed effects is obtained by inference.

2.2.2 Learning

It is not always the case that we are provided with both the structure of the Bayesian Network and its probability distribution. In most cases we may not have either of them. Given the data, we can learn both of these.

2.2.2.1 Parameter Learning

The simplest scenario is that of parameter learning where we have a known structure (possibly from an expert) and the aim is to learn the conditional probability distribution that maximizes the likelihood of training data. The data can be fully or partially observed. For fully observed data, we can use straight forward maximum likelihood estimation to find the parameters. In the case of partially observed data, we can use an EM algorithm to find the MLE of the parameters. Alternatively, we can use Bayesian parameter estimation using (possibly uniform) dirichlet priors [Jordan, 1999].

2.2.2.2 Structure Learning

In structure learning, only the data is available and the structure and parameters of the network have to be recovered from the data. This is a harder problem as [Robinson 1977] showed that the number of possible Bayesian Networks for n nodes is super exponential in the order of $n^{2^{O(n)}}$.

There are two major approaches to learn the structure. The first is constraint-based Structure Learning [Jordan, 1999]. In this case, the algorithm tries to find the independence relations between variables and then use them to find a network which best captures them. There are multiple tests for finding independence among variables [Jordan, 1999]. These algorithms are very intuitive and easy to implement. But they are very sensitive to the results of the individual independence tests. A few incorrect independence results are enough to produce an incorrect network structure.

The second approach is score based learning [Jordan, 1999]. Here we consider all possible Bayesian Networks with n nodes and use some scoring function to determine how each Bayesian Network fits the original data. Due to the large number of candidates, we need to use heuristic-based search methods and an efficient way to calculate the structure scores. We use decomposable scores which can be written as the sum or product of functions that depend only on a node and its parents. If a score is decomposable, then it is easy to reuse the partial scores. BIC or Bayesian scores are the most commonly used scores [Koller and Friedman, 2009].

Score-based learning defines a set of operators on the network like arc addition, deletion or reversal. Each of these operations is applied to the network and the score of the resulting network is calculated. The network with highest score is returned. If the data is not fully observed, then we can use EM based algorithms like Structural EM for structure learning [Friedman, 1998]. More details on learning Bayesian Networks are available in [Heckerman, 1999] and [Koller and Friedman, 2009].

2.3 Dempster-Shafer Theory

Dempster-Shafer theory (DST) is a mathematical theory based on belief functions and evidential reasoning. It is primarily based on two ideas: obtaining degrees of belief for one question from subjective probabilities for a related question and using Dempster's rule for combining such degrees of belief [Shafer and Pearl,1990].

2.3.1 Terminology

DST uses a number in the range [0,1] to indicate belief in a hypothesis given evidence. This number is considered as the degree to which the evidence supports the hypothesis. The hypothesis can be a set of related hypotheses and the impact of the evidence on the subset of hypotheses is represented by a basic probability assignment function denoted as **m**. m(A) is a measure of the portion of total belief committed to hypothesis *A*. m(A) can be considered as a generalization of a probability density function as it assigns a mass to each of the subset of the entire set of hypotheses, Θ.

$$\sum_{A \in \Theta} m(A) = 1$$

A belief function denoted as *bel* assigns the degree of belief to every subset *A* of hypotheses, the sum of beliefs committed to every subset of *A* by *m*. [Shafer and Pearl,1990].

$$bel(A) = \sum_{B|B \subseteq A} m(B)$$

If A^C represents the complement of a set A, plausibility of a subset A denoted as pl(A) is defined as

$$pl(A) = 1 - bel(A^c)$$

Together, belief and plausibility define a belief interval in which the true probability resides. Belief can be considered as the lower bound and plausibility as the upper bound. As we get additional evidence, the bounds tighten and ultimately converge to the true probability.

Given two independent basic probability assignment functions m_1 and m_2, we can use Dempster's rule of combination to combine them to get a new function [Shafer and Pearl,1990]. If there are two evidences that confirm the same hypothesis, then Dempster's rule increases the belief in *both* of the evidences. Similarly, if two evidences contradict each other, then the belief in both of them is reduced.

2.3.2 Rules as Evidences

[Gordon and Shortliffe,1990] showed a possible interpretation of certainty factors in the evidence framework. They consider a rule to be an evidence which supports only one hypothesis to a certain degree. If a rule confirms a hypothesis with degree *s*, then it can be interpreted that the rule assigns *s* to the single hypothesis corresponding the consequent of the rule and *1-s* to the remaining hypotheses. In this interpretation, the certainty factors associated with the rules can be viewed as basic probability assignment functions. [Gordon and Shortliffe,1990] also show that with this interpretation, the combination rules in certainty factors can be considered as a generalization of Dempster's rule of combination.

2.4 Summary

Certainty factors were one of the early mechanisms to handle uncertain reasoning using rules. One of its advantages lies in the observation that rules are a convenient and human understandable way to build expert systems which produce meaningful

explanations of the reasoning results. But they do not model the domain completely and are not very useful for automated reasoning due to potential non modular interactions. The certainty factor framework does not have the ability to utilize additional data to tweak the expert system. Moreover, the framework does not have any mechanisms to invert diagnostic rules for predictive reasoning (or vice versa) and this relies heavily on the expert to define exact and sufficient rule sets in the format needed to solve the desired inference tasks.

Bayesian Networks, on the other hand are, a theoretically consistent framework to accurately model a domain and perform fully automated reasoning. They allow a combination of expert knowledge and data where partial structure can be obtained from expert and parameters are learned from data. They have well known algorithms to do incremental learning and fast inference, and can handle partial observations and still perform inference. However it is hard to explain the inference performed by a Bayesian Network in a human understandable way. Also studies have shown that formulating knowledge in the form of probabilities is non intuitive even for experts [Henrion, 1986].

In this thesis, we propose a mechanism that combines the advantages of the certainty factors and Bayesian Network frameworks. The proposed mechanism can accept rules, bootstrap a network using these rules, learn the complete network from data, perform automated reasoning and explain the inference using human understandable rules.

CHAPTER 3

METHODOLOGY

Rules with certainty factors are a convenient way to express knowledge in a domain with uncertain information and generate user understandable explanations for predictions using the expert specified rules. Bayesian Network is framework that can efficiently perform automated reasoning. Using these two frameworks to create an expert system that can combine expert knowledge, user explanation along with automated reasoning poses a unique set of challenges. This chapter discusses ways by which the expert specified rules are used to bootstrap a Bayesian Network which in turn is used to learn the complete network from data. The resulting network's structure is influenced by the expert specified rules. The completed network can be used for inference and this chapter discusses how to generate explanations for the predictions using the rules. Additional challenges like identifying incorrect rules, proposing new rules for explanation and incremental learning are also discussed. Moreover, this chapter introduces a parameter to certainty factors that allow an expert to specify additional types of uncertainty in the rules that significantly increases their descriptive range and their ability to bootstrap the Bayesian Network parameters.

3.1 From Certainty Factors to Bayesian Networks

[Henrion, 1986] and [Shortliffe and Buchanan, 1984] showed that experts are more comfortable with specifying their domain knowledge using rules and certainty factors instead of absolute probabilities or likelihood. But [Heckerman and Shortliffe,

1992] argued that uncertain reasoning using certainty factors in rule based systems can potentially result in non modular interactions which undermine their theoretical consistency. On the other hand, Bayesian Networks are one mechanism to do uncertain reasoning in a consistent and theoretically sound way. Moreover, they facilitate the combination of domain knowledge and data easily.

Due to their expressive nature, many rule based expert systems like HEPAR and PathFinder have been converted to utilize Bayesian Networks [Korver and Lucas , 1993], [Heckerman, 1989]. But in all these cases, original rules were primarily utilized as a way to identify domain variables. They were not used either in the construction of the Bayesian Network structures or the computation of its conditional probability tables. Others like [Zhang and Luo, 1997] have tried to estimate probabilities from Certainty Factors but require the expert to give an extensive set of related rules which may not be practical.

In this chapter, we propose a method to combine the certainty factor and Bayesian Network frameworks by augmenting the rules with an additional strength parameter which is used to estimate the evidence contributed by the related rules. A partial Bayesian Network whose structure is guided by the rules is constructed and assigned with a consistent set of parameters based on the rules' evidence. Given a set of data, the complete network is learned from data and used to perform automated reasoning. Using the rules and the network, we also discuss ways to explain the inference in a way understandable by users by using the expert specified rules. Furthermore, we also introduce mechanisms to identify wrong rules, identify new rules and perform incremental learning.

3.2 Rules and Certainty Factors

A typical rule based expert system using the certainty factor framework consists of rules in the following format:

RM1: If A AND B THEN C (CF1)

Where RM1 is a rule in the rule set and CF1 is the certainty factor associated with the rule. A temporal rule can potentially involve antecedents in different time frames. A simple example can be,

RM2: If current response for Q1 = previous response for Q1 THEN D (CF2)

3.3 Augmentation of Rules

For the purpose of this thesis, an additional *strength* parameter is added to each of the antecedents in the rule. With this parameter, the rule will look like:

RM3: If A (S1) AND B (S2) THEN C (CF3)

Compared with RM1, the new rule RM3 has two additional parameters – S1 and S2. S1 and S2 provide information about the strength of the antecedents A and B. Intuitively, strength of an antecedent is the degree to which it has to be present for the rule to be applicable. These values represent the absolute strength of individual antecedents and do not depend upon the values of the other antecedents in the same rule. This means that the strengths need not add up to 1 and that they do not represent the probability of the corresponding antecedents.

Given a single rule with strength parameters for its antecedents, it is now possible to estimate the residual evidence of the expert specified rule given that antecedents are not true and thus to provide evidence bounds on negated antecedents. As discussed in section 2.3.2, a rule with certainty factor can be considered as representing a belief

function that adds a piece of evidence for its conclusions given the antecedents. With the addition of the strength parameter, the rules become more expressive. Now, they become a belief function that can potentially provide evidence given each of the subsets of the hypothesis space formed by the rule's premises.

3.3.1 Derivation of Residual Evidence

Consider the following rule:

RM5: If A (0.7) AND B (0.8) THEN C (0.9)

The values 0.7 and 0.8 are the strengths of antecedents A and B. In other words, they reflect the degree to which the expert expected the antecedents A and B to be in the rule. A rule with strength parameters can be considered as a belief function that can provide evidence for the related rules whose antecedents are the variations of the original rule's antecedents. Using the interpretation that rule with strength parameter is a belief function and strength of an antecedent is the degree to which the expert believes the antecedent has to be present in the rule, it is possible to estimate the residual evidence that RM5 provides if some of the antecedents of RM5 do not hold.

In RM5, the expert has provided the strength of A as 0.7. If the antecedent does not hold, then the evidence provided by the RM5 is diluted by 0.3 (1-0.7). In other words, the expert believes that in the absence of the antecedent A, the rule provides only 30% of the evidence it provided when all its antecedents hold. An alternative way is to consider the rule as a line of reasoning. A rule provides reasoning for the consequent to hold when its antecedents are true. Now, if part of the antecedents is not true, then the line of reasoning might still hold but it is proportionally weakened. This means that the evidence

provided by the pseudo rule which incorporates the partial antecedents must be proportionally reduced based on the degree of belief for the original antecedents.

This leads to the creation of a pseudo-rule from RM5 of the form,

RM6: If ¬A AND B THEN C (0.27)

The certainty factor for RM6 is derived by subtracting A's strength from 1 and multiplying it with the RM5's certainty factor. Considering certainty factor as evidence, we proportionally reduce the evidence provided by RM5 by the degree of belief we had on antecedent A for RM5.

The same method applies when A is a discrete non binary variable. Assuming, A can have 3 different values and we have the following rule:

RM7: If A=1 (0.8) AND B (0.6) THEN C (0.8)

The bound for rules involving other values of A are estimated as follows: For RM7, any other value of A can have strength of at most 0.2 (1-0.8). This means that the evidence provided by any variation of RM7 not having a value of A=1 is reduced by 20%. This results in two rules each with a certainty of 0.16.

RM8: If A=2 AND B THEN C (0.16)

RM9: If A=3 AND B THEN C (0.16)

It is important to note here that the certainty factor in these rules represent solely the residual evidence from rule RM5 and not the evidence that the expert would assign to the new rule if he/she were to provide it manually. As such it is more clearly interpreted as a minimum of the evidence that an explicitly specified rule should have.

3.4 Construction of Bayesian Network from Rules

It is well known that experts are comfortable specifying predictive rules and that causal Bayesian Networks result in the most compact network [Henrion, 1986] [Horvitz et al, 1988]. To construct a Bayesian Network from rules, it must be possible to map the rules to the Bayesian Network's structure and parameters. This section discusses ways to bootstrap a Bayesian Network from the rule base.

3.4.1 From Rules to Bayesian Network Structure

A rule indicates a statistical dependency between the antecedent and the consequent and decides whether the relation between them is causal or diagnostic. Intuitively, if the expert provides a set of rules, a partial Bayesian Network with the antecedents as the parent nodes and the consequent as the child node can be constructed. If all the rules are causal, the result is a partial causal Bayesian Network. It is possible that the rule base contains a mix of causal and diagnostic rules. In a typical learning algorithm, parent and child are determined by a statistical independence tests. Here, we assume that the expert's domain knowledge allows him to identify some of the parents and their children nodes in the Bayesian Network.

Consider a domain with 5 boolean variables - A,B,C,D and E. If the rule base contains the rule,

RM4 : IF A (0.7) AND B (0.8) THEN C (0.9)

Then we can infer that the rule implies the partial Bayesian Network structure as shown in Figure 3-1.

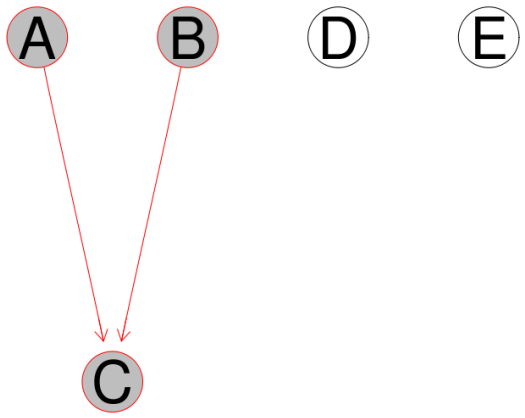

Figure 3-1 Partial network structure learned from rules

3.4.2 From Certainty Factors to Probabilities

Once we have a partial Bayesian Network a set of related rules with evidence, the next step is to fill conditional probability entries for the node. The probabilistic interpretation in [Heckerman, 1986] provides a way to convert a certainty factor for a rule to a conditional probability value.

$$\mathrm{CF}\ (\mathrm{h}\ \rightarrow\ \mathrm{e}|\xi) = \frac{\mathrm{p}(\mathrm{h}|\mathrm{e},\xi) - \mathrm{p}(\mathrm{h}|\xi)}{\mathrm{p}(\mathrm{h}|\xi)\ (1 - \mathrm{p}(\mathrm{h}|\mathrm{e},\xi))\ +\ \ \mathrm{p}(\mathrm{h}|\mathrm{e},\xi)(1 - \mathrm{p}(\mathrm{h}|\xi))}$$

Where CF (h → e|ξ) is the certainty factor for the rule "if e then h" given by an expert with background knowledge ξ . p(h|ξ) is the expert's probability (degree of belief) for h given ξ . p(h|e, ξ) is the expert's probability for h given evidence e and ξ.

After basic algebraic manipulations, we can derive the formula for the conditional probability p(h|e, ξ) given the certainty factor for the rule and the prior p(h|ξ).

$$\mathrm{p}(\mathrm{h}|\mathrm{e},\xi) = \frac{p(h|\xi)[1 + CF(h \rightarrow e|\xi)]}{1 + CF(h \rightarrow e|\xi)[2\mathrm{p}(\mathrm{h}|\xi) - 1]}$$

Consider a rule:

RM10: If A(0.7) AND B(0.8) THEN C (0.9)

For this rule, we can infer that nodes corresponding to A and B are parents of node C. Also from Heckerman's formula, we will be able to derive the value of P(C|AB). Using the strength factors, it is possible to derive the residual evidence provided by RM10 to related rules.

RM11: If ⌐A AND B THEN C (0.27)

RM12: If A AND ⌐B THEN C (0.18)

RM13: If ⌐A AND ⌐B THEN C (0.054)

From RM11-RM13, we can calculate the corresponding estimated residual probabilities for P(C|⌐AB), P(C|A⌐B) and P(C|⌐A⌐B) using Heckerman's formula. However as stated previously, these residual probabilities only reflect the evidence left over from the original rule and might not therefore underestimate the actual conditional probability. To address this, the approach presented in the thesis takes a more evidence-based stance to derive the conditional probability estimate from the rules as detailed in the following sections 3.4.2.1 and 3.4.2.2. Assuming the prior probability of C is 0.6, Table 3-1 summarizes our knowledge about the conditional probability table for C so far.

Table 3-1 From Rules to Conditional Probability Table- I

	Certainty factors(specified or residual)	Probability estimated from Heckerman's formula
P(C|AB)	0.9	0.96
P(C|⌐AB)	0.27	0.72
P(C|A⌐B)	0.18	0.68
P(C|⌐A⌐B)	0.054	0.62

3.4.2.1 From Certainty Factors to Evidence

Even though it is possible to calculate conditional probabilities from each rule using Heckerman's formula, it is unlikely that the calculated probabilities are accurate. One reason is that the probabilities are highly sensitive to the certainty factors provided by experts which are subjective in nature. Also, to calculate an exact conditional probability for say, P(C|AB), we need an exhaustive set of rules involving C . Even if some of the rules involving C are not explicitly specified by the expert then the probability calculated will not be accurate. Additionally, it is possible that the rule base contains rules which overlap and that some rules that are subsets of others. In this case, to identify the exact conditional probability, the independence relations between rules have to be known. This relation can be potentially analyzed from the data, but if data is not available then there is no reliable mechanism to find the relation between two overlapping rules and deduct the influence. Hence the probability calculated from each rule will not be accurate. But given a set of rules, it is possible to find the bounds for the conditional probability. If one of the rules provides conditional probability (or belief) x, then we know that the correct conditional probability is at least x. It is possible that some other unspecified rule exists which increases the conditional probability, but given that the rule is correct, the ultimate conditional probability should lie between x and 1.

A similar argument can be made for related rules (like RM11-RM13) whose residual evidence is derived from the expert specified rule (RM10). Consider pseudo rule RM11.

RM11: If ¬A AND B THEN C (0.27)

If the expert has not specified RM11 explicitly, then 0.27 is the residual evidence provided by RM10 to RM11. On the other hand if the expert has specified RM11 explicitly, there are two consistent possibilities: i) The certainty factor of the rule can be larger or ii) smaller than that of RM10(0.27). If it is greater, then RM11 provides a more precise bound for the evidence and is used. If it is smaller, then the residual evidence from RM10 is higher than the explicitly specified rule. We can infer that the strength factors in RM10 are not consistent with the explicitly specified rule RM11. The residual evidence from RM10 provides a more precise bound and will be used. In either case, the residual evidence obtained from RM10 is the lower bound for the ultimate evidence.

Instead of directly using the probabilistic interpretation of certainty factor to bootstrap the Bayesian Network parameters, this work uses an evidence based interpretation. The evidence (either explicitly specified or residual) forms the belief for the hypothesis and is used as a lower bound for the derivation of the actual conditional probability values.

From the certainty factors of the rule and the strength of individual antecedents, we can directly calculate the lower probabilistic bound for each of the entries in the conditional probability table. We cannot use a similar approach for the calculation of upper bounds as individual rules do not specify an estimate of the counter evidence. The system presented, therefore assigns the default value of 1 for the upper bound for each entry in the conditional probability table unless we have additional knowledge to refine the bound.

Intuitively, it can be argued that the upper bound is unlikely to be 1. If any entry in the conditional probability table has a value of 1, then we have potentially a rule with

stronger predictive power than the existing rules in the rule base. This also means that the expert would have most likely specified this rule. But since it cannot be assumed that the expert will always specify the complete set of highly predictive rules, we have to look for other concrete ways to refine the upper bound.

From an evidence standpoint, the plausibility of a belief interval of a hypothesis is given by subtracting the belief we have in its complement from 1. If two hypotheses are complementary, we can use one of them to determine the plausibility of the other. For example, consider the rule:

RM14: If A (0.8) AND B (0.6) THEN C (0.9)

It is straightforward to derive the lower bound for P(C|AB) and other entries in the conditional probability table for C. But unless our rule base contains a lower bound for P(¬C | A B) we will not be able to refine the upper bound of P(C|AB). To achieve this, our rule base must contain the following (or related) rule,

RM15: If A (S1) AND B (S2) THEN ¬C (CF19)

If such a rule exists, then we can refine the upper bound for P(C|AB) as by the rules of probability P(C|AB) and P(¬C|AB) must sum to 1. Thus rules RM14 and RM15 provide an upper bound for all the entries in the conditional probability for C.

Based on the previous discussions, we can augment our knowledge about the conditional probability table for C with their probabilistic bounds as shown in Table 3-2.

Table 3-2 From Rules to Conditional Probability Table- II

	Certainty factors(specified or residual)	Probability estimated from Heckerman's formula	Probabilistic bounds
P(C|AB)	0.9	0.96	[0.96,1]
P(C|¬AB)	0.27	0.72	[0.72,1]
P(C|A¬B)	0.18	0.68	[0.68,1]
P(C|¬A¬B)	0.054	0.62	[0.62,1]

3.4.2.2 Evaluating the Consistency of Rules

When the expert specifies multiple rules, it is possible that some of the certainty factors associated with rules are not consistent with each other. Since we are performing a probabilistic interpretation of certainty factors, it is easy to detect inconsistencies as they will violate the axioms of probability. This section examines some of the common scenarios:

RM16: If A (S1) AND B (S2) THEN C (CF16)

RM17: If A (S3) AND B (S4) THEN ¬C (CF17)

Consider RM16 and RM17. RM16 determines P(C|AB) and RM17 allows us to determine P(¬C | AB). By the axioms of probability, the individual bounds for P(C|AB) and P(¬C | AB) must contain at least a pair of values that sum to 1. This will allow us to determine inconsistent values for CF16 and CF17.

3.4.2.3 Constructing a Consistent Conditional Probability Table

Once we have a set of probabilistic bounds for entries in the conditional probability table for the partial Bayesian Network, the next step is to determine a set of probability assignments that are consistent with the individual bounds and also the axioms of probability. All the nodes of the Bayesian Network which are not involved in any rules are assigned a probability distribution with uniform dirichlet priors.

A consistent value for a conditional probability entry lies between the lower and upper probabilistic bound (inclusive) derived from the rules. Additionally, the sum of the entry and its negation must sum to 1. For e.g. P(C|AB) and P(¬C|AB) must sum up to 1. If there are two rules, one of which is the superset of other we will have an additional

constraint to enforce the marginalization rule. This means that if we have two rules like IF A AND B AND C THEN D and IF B AND C THEN D then we have an additional constraint to ensure that

$$P(D|BC) = P(D|ABC)P(A) + P(D|\neg ABC)P(\neg A)$$

Where P(A) and P(¬A) are constants calculated from data. If the rules that overlap like IF A AND B THEN D and IF B AND C THEN D, we have two additional constraints:

$$P(D|AB) = P(D|ABC)P(C) + P(D|AB\neg C)P(\neg C)$$

$$P(D|BC) = P(D|ABC)P(A) + P(D|\neg ABC)P(\neg A)$$

As above the value of P(A), P(¬A), P(C) and P(¬C) are calculated from data. If any of the prior probabilities cannot be learned from the data, then they are assumed to be uniformly distributed. Additionally, to ensure that the consistent values fall in the middle of the probabilistic bound as possible, some tolerance variables are introduced whose absolute values lie between half of the width of the probabilistic bound. For e.g. if the probabilistic bound for P(Z|XY) is [0.5,1] then the value of tolerance variable ΔP(Z|XY) lies between ±0.25.

Thus, from the rule base we can derive a set of constraints for the conditional probability entries. The next step is to assign consistent values that satisfy these constraints. This can b performed using either Linear Programming or by sampling the probability space. The current system uses Linear Programming for solving the constraints. As an example, the constraints for rules RM10-RM13 are the following:

a) $0.96 \leq P(C|AB) \leq 1$, $0.72 \leq P(C|\neg AB) \leq 1$, $0.68 \leq P(C|A\neg B) \leq 1$ and $0.62 \leq P(C|\neg A\neg B) \leq 1$

b) $P(C|AB) + P(\neg C|AB) = 1$, $P(C|\neg AB) + P(\neg C|\neg AB) = 1$, $P(C|A\neg B) + P(\neg C|A\neg B) = 1$ and $P(C|\neg A\neg B) + P(\neg C|\neg A\neg B) = 1$

c) $-0.02 \leq \Delta P(C|AB) \leq 0.02$, $-0.14 \leq \Delta P(C|\neg AB) \leq 0.14$, $-0.16 \leq \Delta P(C|A\neg B) \leq 0.16$ and $-0.19 \leq \Delta P(C|\neg A\neg B) \leq 0.19$

d) $1 - P(C|AB) + \Delta P(C|AB) = P(C|AB) - 0.96$, $1 - P(C|\neg AB) + \Delta P(C|\neg AB) = P(C|\neg AB) - 0.72$, $1 - P(C|A\neg B) + \Delta P(C|A\neg B) = P(C|A\neg B) - 0.68$ and $1 - P(C|\neg A\neg B) + \Delta P(C|\neg A\neg B) = P(C|\neg A\neg B) - 0.62$

The objective function is to maximize $P(C|AB) + P(\neg C|AB) + P(C|\neg AB) + P(\neg C|\neg AB) + P(C|A\neg B) + P(\neg C|A\neg B) + P(C|\neg A\neg B) + P(\neg C|\neg A\neg B)$

Performing linear programming for the rules RM10-RM13 provides one set of probability values for the conditional probability table that satisfy the constraints.

Table 3-3 From Rules to Conditional Probability Table- III

	Certainty factors(specified or residual)	Probability estimated from Heckerman's formula	Probabilistic bounds	Consistent probability value from linear programming
$P(C\|AB)$	0.9	0.96	[0.96,1]	0.99
$P(C\|\neg AB)$	0.27	0.72	[0.72,1]	0.87
$P(C\|A\neg B)$	0.18	0.68	[0.68,1]	0.85
$P(C\|\neg A\neg B)$	0.054	0.62	[0.62,1]	0.82

3.5 Learning the Bayesian Structure

From the previous steps, we obtain a partial Bayesian Network structure and a consistent conditional probability table for each node involved in the rules. If a node is not used in any rule, then it is initialized with uniform dirichlet priors. For each rule, the consequent becomes a child and the antecedents become the parent. To further complete and adjust the network structure, available data can be used in the context of a structure

learning algorithm. The structure learning algorithm is bootstrapped with the partial Bayesian Network obtained from the rules. The current system implements modified versions of two popular structure learning algorithms – Greedy Structure Learning and Structural Expectation Maximization (SEM).

3.5.1 Greedy Structure Learning

Greedy Structure Learning [Chickering, 2002] is a popular search based algorithm. It accepts an initial Bayesian Network that in this case has edges corresponding to rules and conditional probability table entries from constraints. This is the partial structure that is implied by the rules. The basic algorithm defines three operators to iteratively modify the Bayesian Network in order to obtain an incremental improvement of the structure – edge addition, edge deletion and edge reversal. In the approach presented here, we constrain the algorithm such that none of the edges specified by the original rules are modifiable. This means that any edge created using rules cannot be deleted or inverted. For each step, the algorithm finds the potential neighbors obtained by performing a single operation on the network, scores each of them and selects the structure that leads to the best improvement in the score. This process is continued until any single operator does not improve the score. In the modified greedy search used here, only the original structure cannot be modified. However, we allow the local probability distributions of all the nodes to float. This means that the conditional probability entries in the learned structure need not be the same as the initial entries determined from the rules and can potentially violate the probabilistic bounds that were calculated. Such violations can later be used to flag potential issues with rules that are inconsistent with the data (see section 3.6)

3.5.2 Structural EM Algorithm

Structural EM [Friedman, 1997], [Friedman, 1998] is an iterative structure learning used when the data is not fully observed. This is useful in projects where the data can only be partially observed. It is based on the Expectation-Maximization principle [Dempster et al, 1977]. The algorithm here starts from an initial structure that is constructed based on the rules. At each step, it generates a complete data set based on the current model. It then computes the expectation of the score for all the Bayesian Networks in the neighborhood of the current model and chooses the one which maximizes the score. It uses greedy search to learn the optimal structure at each step. Using RM10-RM13 as an example, the complete network learned from the data is shown in Figure 3-2. The conditional probability table for node C that is learned from the data is shown in Table 3-4.

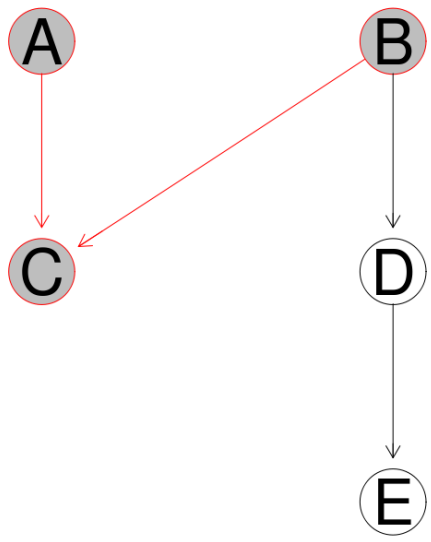

Figure 3-2 Complete structure learned from data

Table 3-4 From Rules to Conditional Probability Table- IV

	Certainty factors(specified or residual)	Probability estimated from Heckerman's formula	Probabilistic bounds	Consistent probability value from linear programming	Probability learned from Data
P(C|AB)	0.9	0.96	[0.96,1]	0.99	0.96
P(C|⌐AB)	0.27	0.72	[0.72,1]	0.87	0.82
P(C|A⌐B)	0.18	0.68	[0.68,1]	0.85	0.78
P(C|⌐A⌐B)	0.054	0.62	[0.62,1]	0.82	0.88

3.5.3 Incremental Learning

Some of the domains in which the approach presented in this thesis is going to be applied have a steady stream of incremental data. In this case, the system uses rules, certainty factors and the available data to learn the initial structure. As we get more data, the existing Bayesian Network must be adjusted so that it still fits the data. This incremental learning happens periodically. The new data can result in a modification of parameters and potentially even the structure. Currently, the system performs parameter estimation of the conditional probability table for the existing structure using the data. In other words, it does not start the structure learning from scratch each time we get new data. If there is a large divergence between the model before and after the learning from the incremental data, we can make a decision to perform structure learning again. This gives the system the ability to perform batch and online learning [Bauer et al,1997].

3.6 Detecting Potential Incorrect Rules

As indicated in section 3.5.1, the incorporation of data into the network structure and parameters can lead to conditional probability entries that are inconsistent with the bounds derived from the rules. This, in turn, can indicate that the certainty and strength factors or, in extreme cases, even a rule might be incorrect. To use this, the learned

Bayesian Network structure and parameters are compared with the original Bayesian Network that is used as a seed. In particular, the learned probability values are compared to the bounds that were derived from the rules. If the learned values fall outside the bounds by more than a tolerance value, the system identifies a potential problem with the corresponding rule and asks the expert to verify the rules. This is an inexpensive way to determine any issues in rules as it uses the original probability bounds to determine deviations.

An alternate, but expensive, method which would identify rule conflicts in more detail and could attempt to automatically resolve them is to perform structure learning for different subsets of rules. For each subset, this corresponds to ignoring some subset of the rules and to learn a new structure with the reduced set of rules. If any of the structures learned have a higher score than the structure learned with all the rules, the potentially incorrect rule can be identified whose veracity can then be verified with the expert.

3.7 Explanation of Predictions

Once a Bayesian Network has been constructed, it can be used for predictions. For example, the system can be used for classification where given a set of records inference is used to find the most likely value for a target random variable. Similarly, the method can be used interactively where the user enters observations and is interested in the distribution of specific unobserved variables. For example, in a credit card domain (see section 4.2), the user can provide the details of a transaction and might be interested in the probability that it is fraudulent.

3.7.1 Generating Explanations

Once the system has a distribution of the variable, it has to explain the prediction to the user. Doing this purely from the Bayesian Network is very difficult as explanations in terms of the network's probability distribution are exceedingly difficult for users (and even experts) to understand. On the other hand, knowledge encoded in the rules should be understandable to the expert. As a consequence, the system presented here derives explanation for Bayesian Network predictions using the original rules in the system.

The user provides a set of observations and variables whose distribution he is interested in. The system performs the inference and finds the rules in the rule base whose consequent matches the most likely value of the predicted variable. The system finds the Bayes factor for each of the matching rules. A higher value of Bayes Factor means that the rule is strongly supported by the data and has a higher likelihood of occurrence that its complement. Bayes factor BF [Yuan and Lu, 2008] for the a rule of the form "If A then B" is defined as,

$$BF(A \rightarrow B) = \frac{\mathrm{P(B|A)}}{1 - \mathrm{P(B|A)}}$$

Bayes Factor is calculated for each matching rule and the rules are sorted in descending order based on their Bayes Factor. The rule with the highest Bayes factor can be considered as the strongest rule. The system will use the rules (from strongest to weakest) in explaining the rationale of the prediction. In the case of overlapping rules, it is possible that multiple rules apply to a single record. The system takes the strongest rule and gives the user the percentage of the data that the strongest rule explains. After that, all the records where strongest rule applies are removed. The other rules are considered to augment the primary rule and the process is continued.

For example, consider the system with the following rules:

RM18 : IF A AND B THEN D

RM19 : IF C AND E THEN D

To obtain a prediction, the user will provide a set of observations and also indicate the set of variables whose distribution he is interested in (e.g. D). After inference, if the system finds that D is likely, it displays the belief to the user. It also tries to use the set of applicable rules to explain its rationale. In this scenario, we notice that both RM18 and RM19 are applicable.

Let us consider that we have 100 records which have the value of D that the system predicted. 60 of the records satisfy RM18 and 50 of them satisfy RM19. There is also an overlap of 30 records for which both RM18 and RM19 hold. The system performs the inference and gets the probability that D is true. Using the Bayes factor, if the system determines that RM18 is the strongest rule, it removes those 60 records. Of the remaining 40 records, 20 of them can be explained by RM19 and there are no rules in the rule base that explain then other 20.

So the explanation of the system will provide the following : The probability that the system expects D to hold, the fraction of evidence that agree with strongest rule, the fraction of evidence that other applicable rules add to the strongest rule and the fraction of evidence that cannot be explained from the rules (or inferred from the data).

3.8 Suggesting New Rules

In a typical scenario, most of the evidence can be explained by the rules available in the dataset. For example, in the example described above, only 20% of the evidence was not explained by the rules. If for some prediction the system was not able to use rules

to explain a significant amount of the evidence, this indicates that this is a potential scenario to mine for new rules. In this case, the system uses lack of sufficient rules for explanation to mine new rules.

Continuing the example above, if the user tries to find the likelihood of say E, the system will not be able to use any of the rules in the rule set to explain the reason for its prediction. The system will then try to perform association rule mining [Agarwal et al, 1993] on the set of records for which E holds to find potential rules which can explain the prediction for E.

The new rules that are mined are compared with the existing rules to make sure that they are not subsets or supersets of the existing rules. The system also calculates the certainty factor of the rule using Heckerman's formula and assigns default strength as 1. The set of rules which have a certainty factor above some threshold are suggested to the user who may potentially accept or reject them. Accepted rules are added to the rule set and are used for future explanations. They do not result in any immediate alteration of structure or parameters.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Implementation Details

This thesis was primarily implemented using Python. The Open Bayes [Open Bayes, 2004] library provided the framework for representing Bayesian Networks. The code consists of modules to represent Bayesian Networks , perform inference, perform parameter and structure learning, parse rules, perform rule based structural learning and explanation of predictions. The system has two inference engines - Junction tree for exact inference and MCMC for approximate inference. It also consists of two structure learning algorithms – greedy structural learning for complete data and SEM for incomplete data. The GNU Linear Programming Kit (glpk) was used to find the consistent probability assignments.

4.2 Data Sets

Two datasets were primarily used for performing experiments. Alarm is one of the common benchmark networks used to test structure learning algorithms. It is introduced in [Beinlich et al, 1989] and was designed to monitor patients in an intensive care unit. The primary purpose is to assist anesthetists in interpreting changes in vital signs in patients. The network shown in Figure 4-1 consists of 37 discrete variables whose cardinality ranges from two to four. The variables represent observations from an intensive care unit like the patients history and device readings.

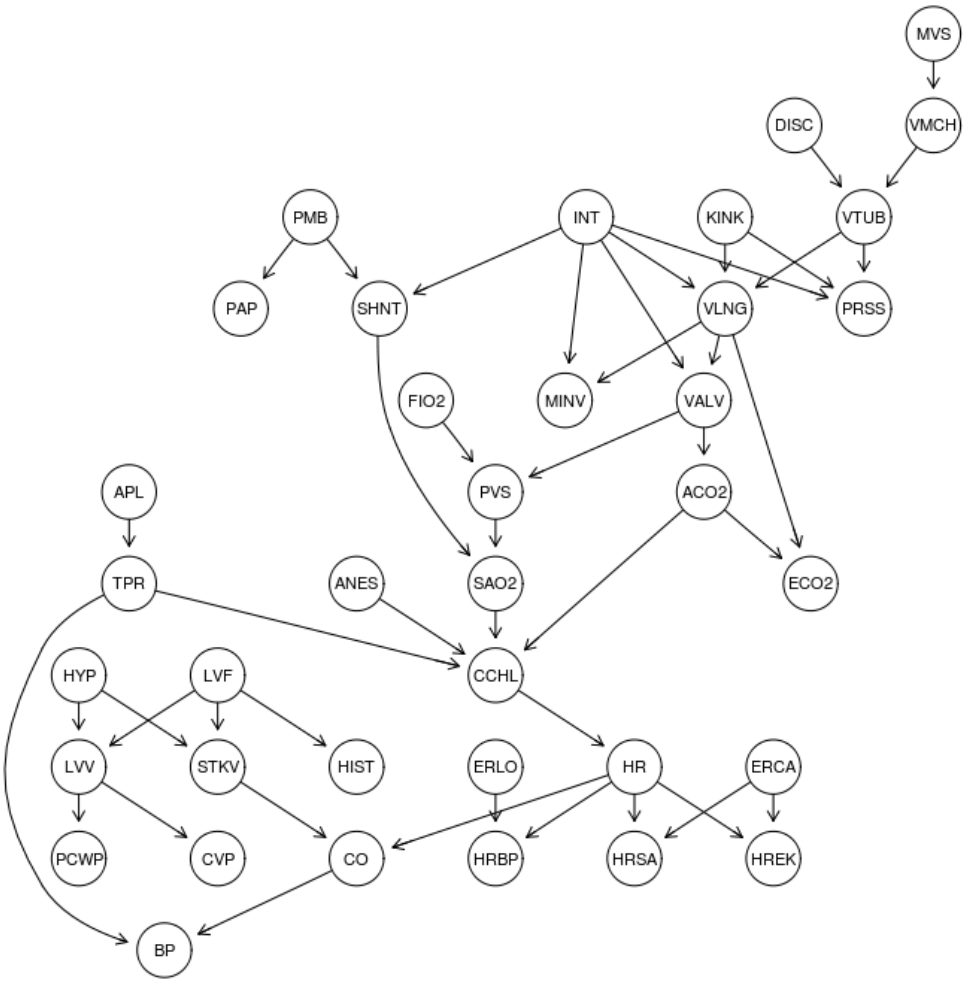

Figure 4-1 Original Alarm Network

Some of the variables of the network that were used in the rules are explained here. For further discussion refer to [Beinlich et al, 1989].The heart rate of the patient is modeled by the variable HR. HREK represents the heart rate as measured by the EKG monitor. Heart rate also influences heart rate / blood pressure represented by the variable HRBP. Failure of the heart to circulate blood is modeled by left ventricular failure (LVF) . Hypovolemia (HYP) occurs if the heart does not have enough blood to circulate. The stroke volume (STKV) is the amount of blood pumped out of the heart. Stroke volume (STKV) and left ventricular end-diastolic volume (LVV) provide a way to monitor the

heart's performance. Hypovolaemia (HYP) and left ventricular failure (LVF) adversely affect the heart's performance. Heart rate (HR) and stroke volume (STKV) influence the amount of blood circulated per minute, also called cardiac output (CO). The resistance of the blood vessels to blood circulation is modeled by total peripheral resistance (TPR). The cardiac output (CO) and the total pulmonary resistance (TPR) determine the blood pressure (BP). Anaphylaxis (APL) corresponds to the medical condition where TPR is reduced. The breathing pressure of the patient is modeled by PRSS.

The second data set used for experiments is from UCSD's Data Mining contest from 2009 [UCSD, 2009]. The data consisted of e-commerce transactions from FICO and the aim is to predict whether a transaction is fraudulent. The dataset consists of 19 attributes including transaction details like time, customer address location, amount, originating domain and other proprietary fields. The attributes corresponding to amount and total amount were discretized.

4.3 Representation of Rules

All the rules in the thesis were represented in accordance with Java Rules Engine API Specification (JSR 94). This is a generic specification that allows rules to be specified and accessed regardless of the language. Additional attributes were added to represent certainty factors and the strength for the individual antecedents. Currently the system considers only rules with AND operators. A sample rule and its representation are given below:

RE1 : IF A (0.6) AND B (0.7) THEN C (0.8)

Here, 0.8 is the certainty factor for the rule R3 and 0.6 and 0.7 are the respective strength of variables A and B. This rule is represented as:

<rule name="RE1">
<parameter identifier="record"/>
<condition strength='0.6'>record.A == 1<condition>
<condition strength='0.7'>record.B == 1<condition>
<consequence cf='0.8'>record.C == 1</consequence>
</rule>

This framework also allows for arbitrarily complex rules using multiple parameters and conditions. As an example, consider a rule with temporal condition that indicates that the user's response must match his/her previous response. The rule and its equivalent representation is:

RE2: IF current response for Q1 = previous response for Q1 (0.7) THEN D (0.9)

<rule name="RE2">
<parameter identifier="curRecord"/>
<parameter identifier="prevRecord"/>
<condition strength='0.7'>curRecord.Q1 == prevRecord.Q1<condition>
<consequence cf='0.9'>curRecord.D == 1 </consequence>
</rule>

4.4 Setup

Typically the rules and their associated parameters will be specified by an expert. For the purpose of this thesis, the rules for the alarm and online transaction domain were mined from the data using association rule mining [Agarwal et al, 1993]. For each of the rules, the values of support, confidence and certainty factor are calculated. The rules with a highest values for these factors were added to the rule base. For each of the rules added to the dataset, the certainty factor was calculated using Heckerman's formula [Heckerman, 1986]. The values of prior and conditional probability used in Heckerman's formula were obtained from the data. If the data is not available, a uniform prior is assumed.

For the alarm network, a dataset of 20,000 records is used for training. A separate dataset of 10,000 records sampled from the original alarm network [Beinlich et al, 1989]

was used for testing. In the case of online transaction network, there are around 95,000 records in the original dataset. Repeated random sub sampling is used to split the data into training and testing datasets. This procedure is repeated for 26 times and the results were averaged.

Two measures are used to compare the desirability of the candidate networks. The first measure used is the BIC score of the candidate network using the testing dataset[Koller and Friedman, 2009]. The second is the classification error which is defined as the ratio of incorrectly classified records to the total number of records.

4.5 Experiments

Different sets of experiments were conducted to highlight the various modules of the proposed system. In the first set of experiments, the system was provided with a rule base consisting of correct rules (with certainty factors and strength). This demonstrates the bootstrapping of initial network and how the rules influence the complete structure that is learned from the data. This experiment also analyzes the advantages provided by the bootstrapping approach proposed by this thesis. Compared with a network that learns its entire structure from data, the network that is bootstrapped from rules needs a smaller dataset before it can make meaningful predictions. This experiment also analyzes the impact of the size of the rule base by comparing the networks learned from rule bases with different sizes.

In the second set of experiments, incorrect rules were added to the rule base and the deviation between initial and learned Bayesian Network is observed. This experiment also shows how the system can inform the user about potentially incorrect rules. In the

third set of experiments, the system is asked to make predictions based on observations and explain its predictions using the expert specified rules.

4.5.1 Bootstrapping and Learning a Bayesian Network from Rules

In this experiment, we look at how the system learns the network structure from a rule base which contains rules with appropriate strength and certainty factors.

4.5.1.1 Alarm Network Experiments

As a simple example, let us analyze the scenario where the rule base consists of only the following rule:

RE3: If LVF = FALSE (0.99) AND HYP = FALSE (0.8) THEN STKV = NORMAL (0.9)

Using the methods described in the thesis, we can derive the evidence for related rules that can be obtained from RE3. The related rules and the residual evidence are:

RE4 : If LVF = FALSE AND HYP = TRUE THEN STKV = NORMAL (0.18)

RE5 : If LVF = TRUE AND HYP = FALSE THEN STKV = NORMAL (0.009)

RE6 : If LVF = TRUE AND HYP = TRUE THEN STKV = NORMAL (0.018)

By applying Heckerman's formula we can derive the conditional probability and use it to estimate the probabilistic bounds. A set of probability assignments consistent with the bounds are obtained using linear programming. Finally, the structure learning algorithm learns the complete structure from the bootstrapped structure and parameters provided.

Table 4-1 shows the bounds and bootstrapped values for the conditional probabilities resulting from the rules as well as the final learned probability value after refinement with data.

Table 4-1 Alarm - Conditional Probability table for a correct rule

LVF	HYP	Certainty Factor	Probabilistic Bound for STKV = NORMAL	Consistent value of Probability	Probability after learning
False	False	0.9	[0.85,1]	0.93	0.89
False	True	0.18	[0.54,1]	0.77	0.51
True	False	0.009	[0.08,1]	0.53	0.04
True	True	0.018	[0.11,1]	0.56	0.01

From this experiment, we can see that the final conditional probability for the original rule that is learned from the data stays within the bounds calculated from the rule. Furthermore, we can see that the final learned structure has both LVF and HYP as parents for STKV.

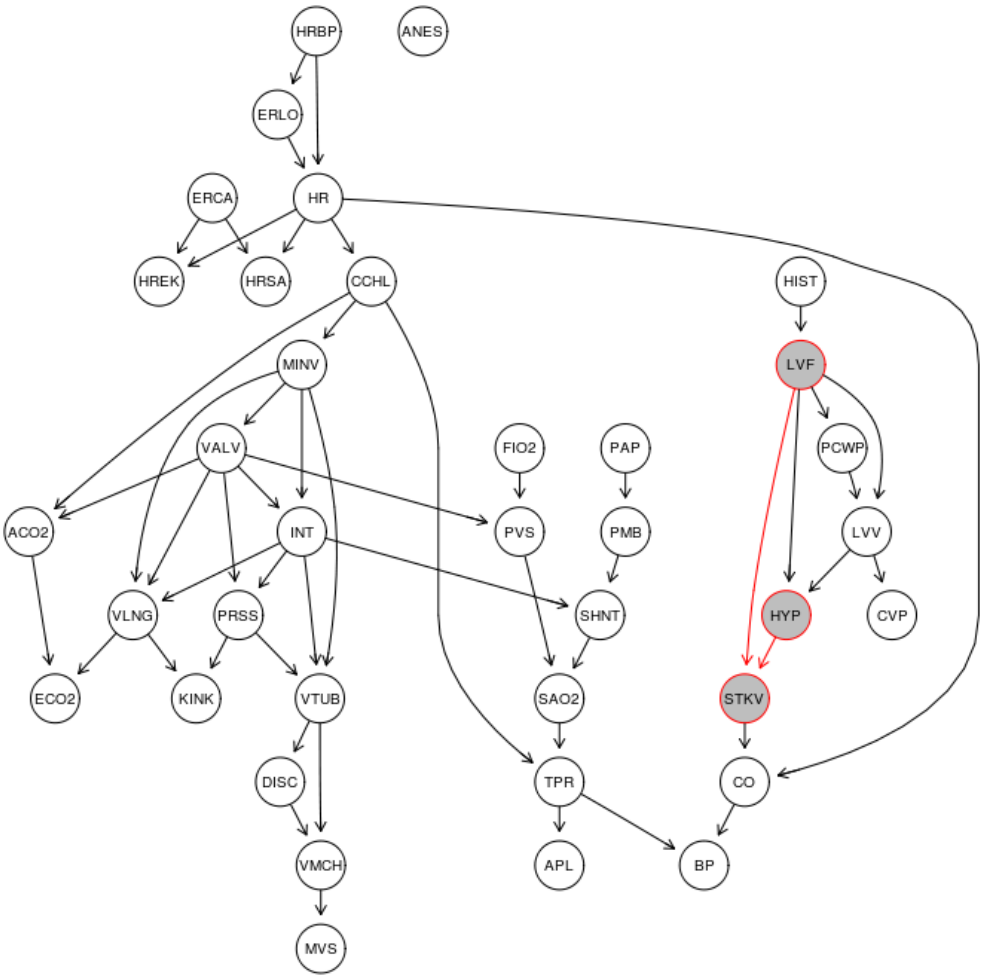

Figure 4-2 Alarm network learned with single rule

In a typical scenario, the rule base will contain multiple rules. To illustrate the performance of the system under these conditions, a rule base for the Alarm network with the following rules was constructed.

RE7: IF LVF = TRUE (0.6) AND HYP=TRUE (0.6) THEN LVV = LOW (0.9)

RE8: If LVF = FALSE (0.99) AND HYP = FALSE (0.8) THEN STKV = NORMAL(0.9)

RE9 : IF HR = HIGH(0.9) AND STKV = HIGH (0.6) THEN CO = HIGH (0.9)

RE10 : IF HR = LOW (0.6) AND STKV = LOW (0.9) THEN CO = LOW (0.99)

RE11 : IF CO = HIGH(0.8) AND TPR = HIGH(0.6) THEN BP = HIGH (0.8)

RE12 : IF CO=NORMAL(0.7) AND TPR=NORMAL(0.6) THEN BP = NORMAL (0.8)

RE13 : IF APL = TRUE (0.6) THEN TPR = LOW (0.9)

RE14 : IF HR = HIGH(0.99) THEN HREK = HIGH (0.6)

RE15 : IF HR = HIGH(0.99) THEN HRBP = HIGH (0.6)

RE16 : IF VLNG = LOW(0.7) AND ACO2 = LOW (0.8) THEN ECO2 = LOW (0.9)

RE17 : If KINK=FALSE(0.9) AND VTUB=LOW (0.8) THEN PRSS=HIGH (0.6)

RE18 : If VTUB=NORMAL(0.6) AND INT=NORMAL(0.9) THEN PRSS=HIGH (0.8)

These rules were obtained by analyzing the dependencies among Alarm network's diagnostic and intermediate nodes [Beinlich et al, 1989]. The prior and conditional probabilities were obtained from the data and the certainty factor for each rule was calculated from the Heckerman's formula [Heckerman, 1986]. The strength factor for an antecedent in a rule is estimated as the ratio of the number of records where both the antecedent and the consequent hold to the total number of records where the consequent holds. The values of the certainty factors and strength calculated were rounded to the nearest tenth of a fraction.

If this rule base is used to bootstrap the alarm network, then the system learns the network structure shown in Figure 4-3. The nodes involved in the rules are highlighted. As in the initial example, the bootstrapped network was able to accurately represent the data generated from the original alarm network.

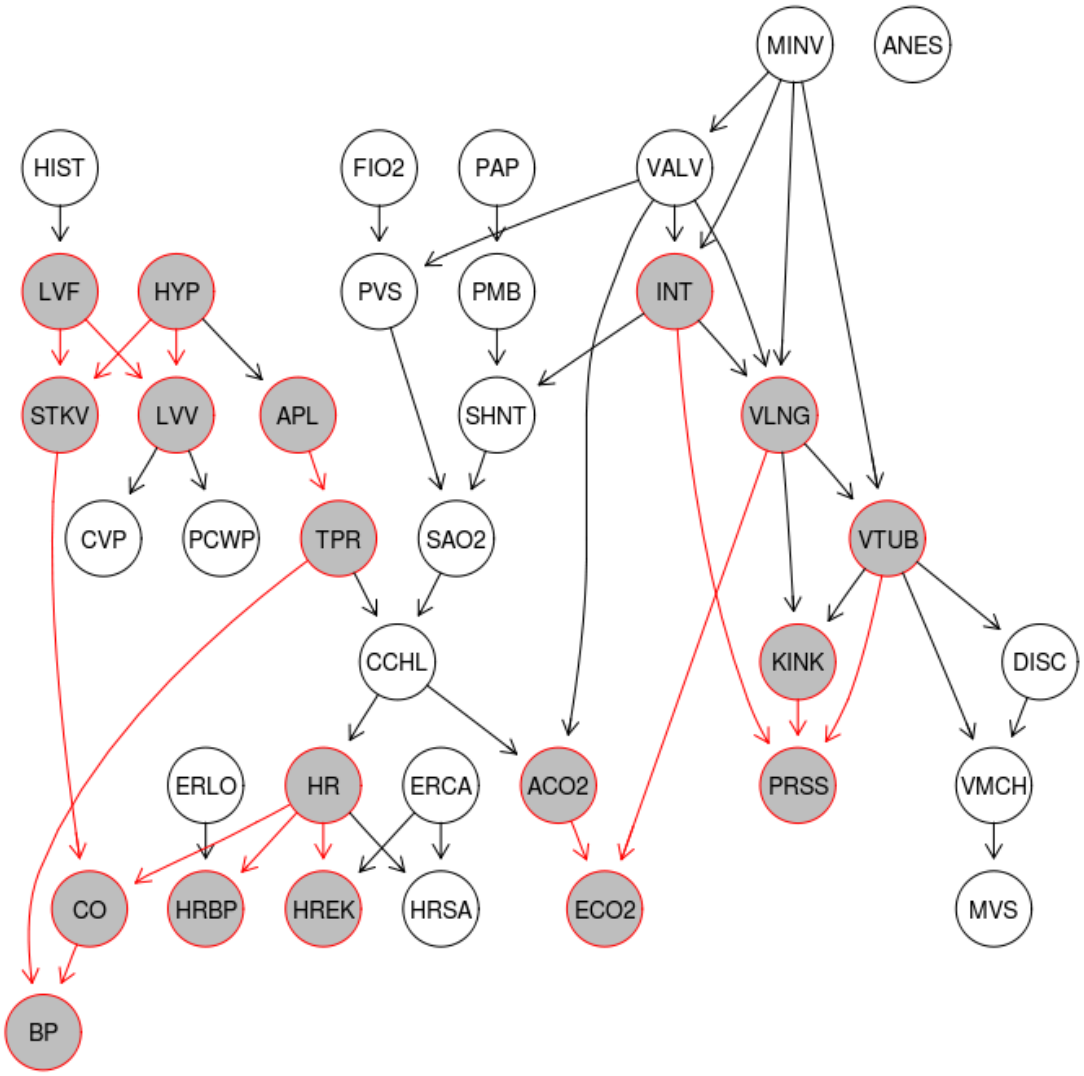

Figure 4-3 Alarm network learned from rule base

To evaluate the benefits of bootstrapping the network, we test if bootstrapping the Bayesian Network reduces the number of records that the network needs before it can perform predictions with reasonable accuracy.

The training dataset for the alarm network has 20000 records. To evaluate the effect of more data, 1000 records are initially selected at random from the training dataset. This partial dataset is used to learn two networks – one whose structure is bootstrapped from the rules and another with an empty initial structure. Both these

networks are compared according to their score, size and their classification error on the testing dataset. This process is repeated for 26 times and the average value of BIC score and classification error is noted. The whole experiment is repeated for partial datasets whose size varies from 1000 to 20000 in multiples of 1000 and the resulting BIC score and classification error rates are presented in Table 4-2 and Table 4-3 respectively.

Table 4-2 Impact of dataset size on network learned I

Dataset Size	Network bootstrapped with rules		Network seeded with empty structure	
	#Edges	BIC Score	#Edges	BIC Score
1000	44	-112605.4	46	-113218.3
2000	45	-111472	47	-112666.2
3000	47	-111324.3	48	-112601.8
4000	46	-110896.3	47	-112117.9
5000	47	-110855.9	47	-112117.9
10000	48	-110486.3	52	-111246.5
15000	48	-110486.3	52	-111246.6
20000	50	-110500	53	-111281.5

From Table 4-2, we can see with as few as 4000 records, the bootstrapped network converges to a structure that is close to the structure that is learned using the entire training dataset with 20,000 records. When no rules were used to bootstrap the network, around 10,000 records were needed before the network approached the structure learned with the entire dataset.

Each of the networks learned with partial datasets were also used for predicting the value of nodes involved in the rules. As an example, the learned network is used to predict the value of STKV in the test dataset. From Table 4-3, it can be noticed that when the network is bootstrapped with rules, the learned network approaches the classification error of the network which uses the entire training dataset with as few as 4000 records. Additionally, when we do a paired t-test for both the networks, the

difference between classification errors are statistically significant for up to 4000 records. After 5000 records, the two networks converge to almost similar classification error rates. It must be noted that when the number of records used for training is low, MCMC based inference engines perform the best.

Table 4-3 Impact of dataset size on network learned II

Dataset Size	Mean Classification error (CE) for STKV using Network bootstrapped with rules	Classification error for STKV using Network bootstrapped with empty structure	Statistics		
			Mean of difference in CE	Std Dev of difference in CE	P value for t(25)
1000	0.176	0.269	0.093	0.021	2.20E-016
2000	0.173	0.263	0.091	0.023	2.20E-016
3000	0.173	0.220	0.047	0.017	1.91E-013
4000	0.173	0.186	0.012	0.031	0.0514
5000	0.175	0.176	0.001	0.010	0.6011
10000	0.173	0.174	0.001	0.009	0.8448
15000	0.174	0.175	0.001	0.004	0.4739
20000	0.174	0.175	0.001	0.003	0.05629

To test the impact of the rule base size on the network quality, a new rule base with only 3 out of the 12 rules from the original rule base was constructed. The rules included are:

RE7: IF LVF = TRUE (0.6) AND HYP=TRUE (0.6) THEN LVV = LOW (0.9)

RE8: If LVF = FALSE (0.99) AND HYP = FALSE (0.8) THEN STKV = NORMAL(0.9)

RE16 : IF VLNG = LOW(0.7) AND ACO2 = LOW (0.8) THEN ECO2 = LOW (0.9)

The experiment was very similar to the previous experiment. 1000 records are initially selected at random from training dataset and were used to create two networks – one whose structure is bootstrapped with the partial rule base and another with an empty structure. The score, size and classification error are calculated on the testing dataset. This process is repeated for 26 times and the average values are calculated. The whole experiment is repeated for datasets with size ranging from 1000 to 20,000 in multiples of 1000. The structure that is learned from the partial rule base is given in Figure 4-4 .

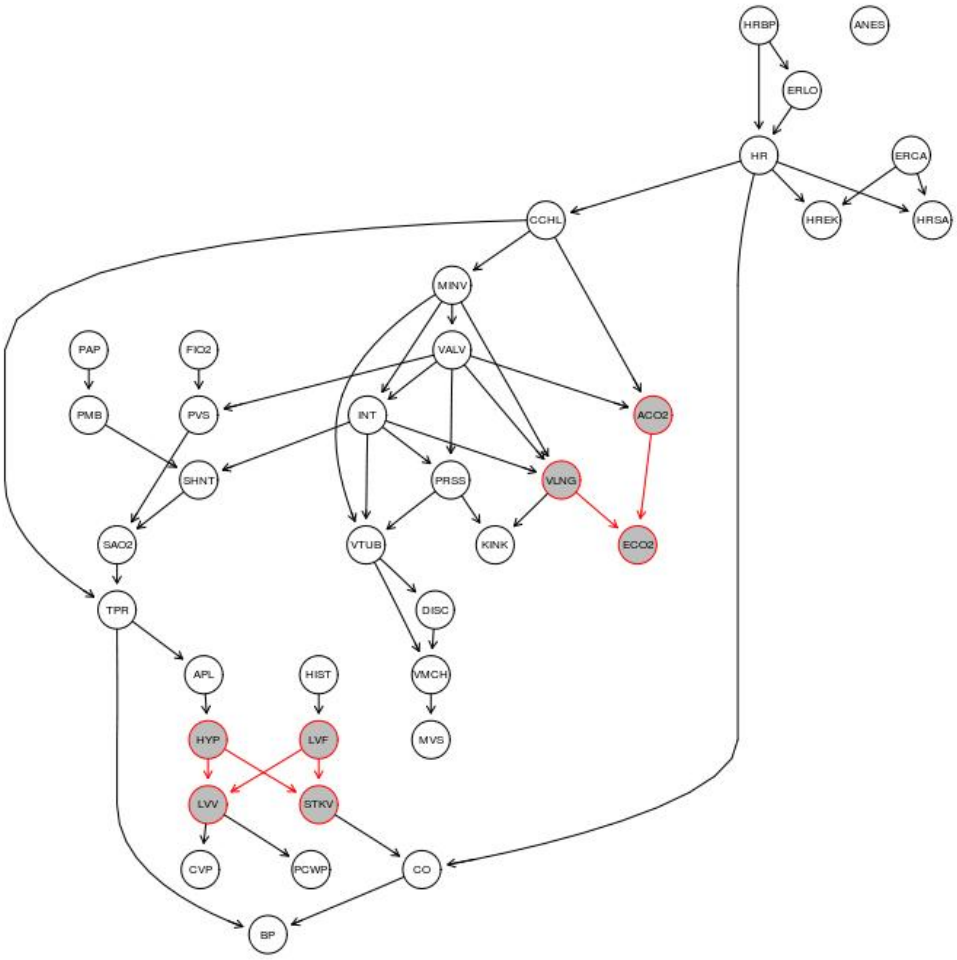

Figure 4-4 Alarm Network learned from smaller rule base

Table 4-4 and Table 4-5 show the results of the experiment. When the rule base has a reduced number of rules, the network still needs less data to converge to a final

structure. The structural complexity and score are worse than the network learned with full rule base but better than the network learned with incorrect rules.

Table 4-4 Impact of Rule Base Size on Network Learned I

Dataset Size	Network bootstrapped with full set of rules		Network seeded with partial set of rules	
	#Edges	BIC Score	#Edges	BIC Score
1000	44	-112605.4	44	-112705.5
2000	45	-111472	45	-111586.3
3000	47	-111324.3	46	-111521.9
4000	46	-110896.3	46	-111063.4
5000	47	-110855.9	46	-111063.4
10000	48	-110486.3	50	-111166.5
15000	48	-110486.3	50	-111166.6
20000	50	-110500	52	-111203

From Table 4-5 we can see that the classification error of the network learned with partial rule base is very close to the classification error of the network learned with a larger rule base. We can also see that when we do a paired t-test for networks learned with different rule bases, the difference between classification errors are statistically significant for up to 4000 records. After 5000 records, the two networks converge to almost similar classification error rates. This shows the quality of the Bayesian Network that is learned is influenced by the size (and quality) of the rule base. A larger rule base result in a superior network but the system does provide good results even when bootstrapped with a smaller rule base.

Table 4-5 Impact of Rule Base Size on Network Learned II

Dataset Size	Mean Classification error (CE) for STKV using Network bootstrapped with rules	Classification error for STKV using Network bootstrapped with empty structure	Statistics		
			Mean of difference in CE	Std Dev of difference in CE	P value for t(25)
1000	0.175	0.269	0.094	0.024	2.20E-016
2000	0.176	0.263	0.087	0.024	6.95E-016
3000	0.175	0.220	0.045	0.023	4.84E-010
4000	0.173	0.186	0.013	0.031	0.04898
5000	0.175	0.176	0.001	0.015	0.8208
10000	0.174	0.174	0.001	0.007	0.849
15000	0.172	0.175	0.002	0.004	0.006793
20000	0.174	0.175	0.001	0.003	0.1559

This experiment shows that the presented approach to incorporating rules to bootstrap a Bayesian Network significantly decreases the need for data to make accurate prediction and is capable of efficiently translating the rules into Bayesian Network structure that accurately reflect the expert knowledge and the data.

4.5.1.2 Online Transaction Experiments

Consider the following rule base for the online transaction dataset:

RE19 : IF FLAG1 = 1(0.8) THEN FLAG4 = 1 (0.9)

RE20: IF FIELD3 = 2247 (0.6) THEN RESULT =1 (0.9)

RE21 : IF FIELD3 = 2389 (0.6) THEN RESULT =1 (0.9)

RE22 : IF FIELD1=4 (0.5) THEN RESULT=1 (0.8)

RE23 : IF FLAG5=6 (0.5) THEN RESULT=1 (0.8)

The rules were obtained by using association rule mining [Agarwal et al, 1993] on the entire transaction dataset. For each of the candidate rules with high confidence, the certainty factors were calculated. Rules with high certainty factors are added to the rule base. The certainty factor and strength parameters are rounded to the nearest tenth. The network that is learned after bootstrapping with the rule base is show in Figure 4-5. The nodes involved in the rules are highlighted.

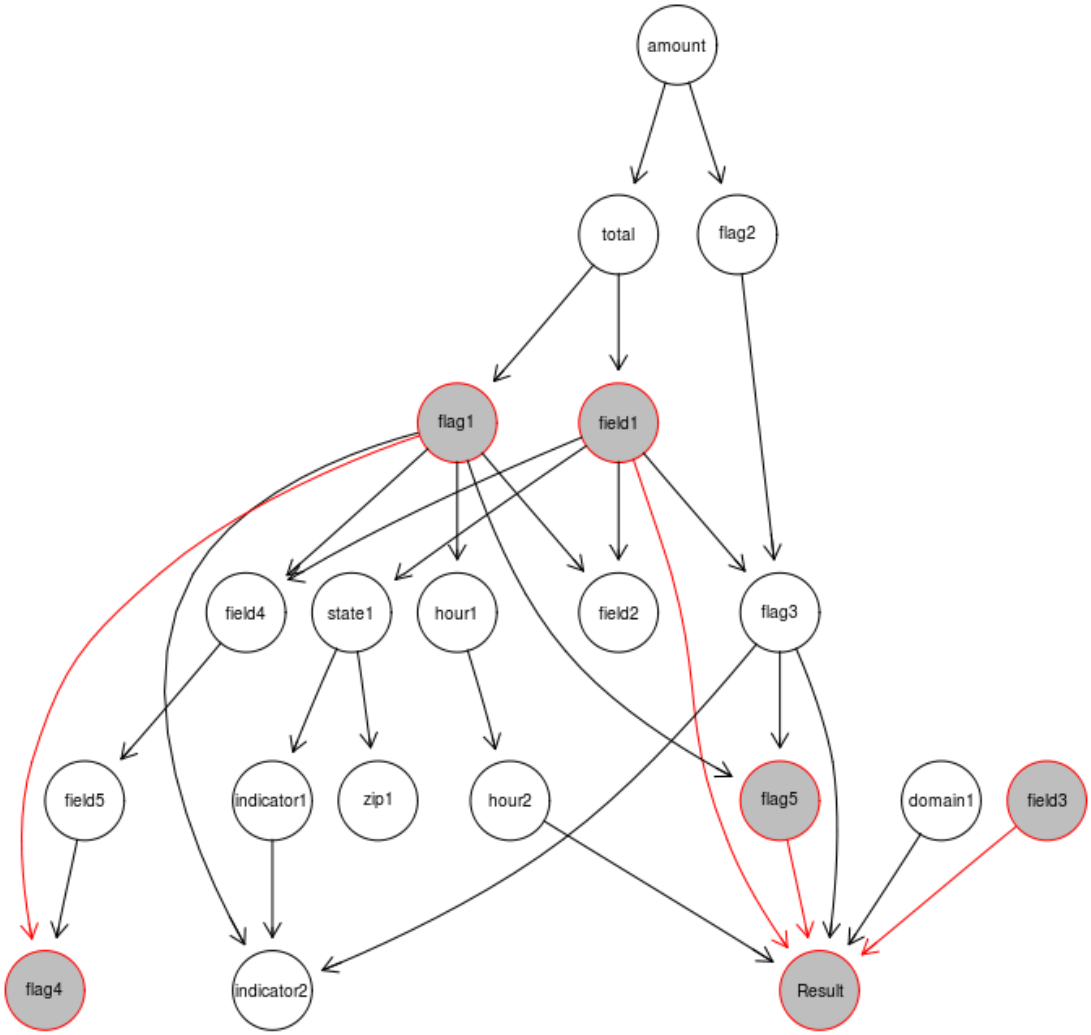

Figure 4-5 Online transaction network learned from rule base

Again, the network was able to accurately reflect the relation in the transaction data set.

4.5.2 Detection of Incorrect Rules

In this experiment, we add an incorrect rule to the rule base and observe how the system recognizes the incorrect rule. Consider that the expert specified the following incorrect rule for the alarm dataset:

RE24 : If PAP = NORMAL (0.9) AND ANES = TRUE (0.9) then LVF = TRUE (0.9)

Using the methods described in the thesis, we can derive the evidence for related rules from RE24. The related rules and the residual evidence are:

RE25 : If PAP = LOW AND ANES = TRUE then LVF = TRUE (0.09)

RE26: If PAP = LOW AND ANES = FALSE then LVF = TRUE (0.009)

RE27 : If PAP = HIGH AND ANES = TRUE then LVF = TRUE (0.09)

RE28 : If PAP = HIGH AND ANES = FALSE then LVF = TRUE (0.009)

RE29 : If PAP = NORMAL AND ANES = FALSE then LVF = TRUE (0.09)

Table 4-6 provides the conditional probabilities, their probabilistic bounds, the bootstrapped values provided by linear programming and the final values learned by the structure learning algorithm.

Table 4-6 Alarm - Conditional Probability table for an incorrect rule

PAP	ANES	Certainty Factor	Probabilistic bound for rule LVF = TRUE	Consistent value of Probability	Probability after learning
NORMAL	TRUE	0.9	[0.6,1]	0.84	0.04
LOW	TRUE	0.09	[0.059,1]	0.48	0.05

Table 4-6 - *Continued*

LOW	FALSE	0.01	[0.0508,1]	0.48	0.05
HIGH	TRUE	0.09	[0.059,1]	0.48	0.03
HIGH	FALSE	0.01	[0.0508,1]	0.48	0.06
NORMAL	FALSE	0.09	[0.059,1]	0.48	0.05

From this data, it can be noted that the final conditional probability that is learned from the data for RE24 falls significantly from the probability bounds derived from the rule. This is an indication that the rule that is specified by the expert might be incorrect.

This can be used to elicit a rule refinement from the expert which not only leads to better explanation but also improves the final network structure. The latter is due to the fact that incorrect rules results in a Bayesian Network whose structural complexity is high. To show this, let us compare the Bayesian Network learned from the incorrect rule RE24 (shown in Figure 4-6) with the one learned from rules RE7-RE18 (Figure 4-3).

Table 4-7 Comparison of Alarm Bayesian Networks

	Bayesian Network generated with Correct Rules [RE7-RE18]	Bayesian Network generated with Incorrect Rule [RE24]
BIC Score	-110500.0	-111386.4
#Edges	50	55
Classification Error for LVF	0.0156	0.0472

From Table 4-7 which shows the network's complexity and accuracy, we can see that the Bayesian Network generated from incorrect rule RE24 has 5 more edges than the one learned from correct set of rules. This network also has a lower BIC score [Koller and Friedman, 2009] and has a classification error that is around 3 times higher than the Bayesian Network learned from correct set of rules.

The system provides the following message: “Please verify rule RE24. The final probability learned from the data does not fall within the bounds estimated from the values specified in the rules”.

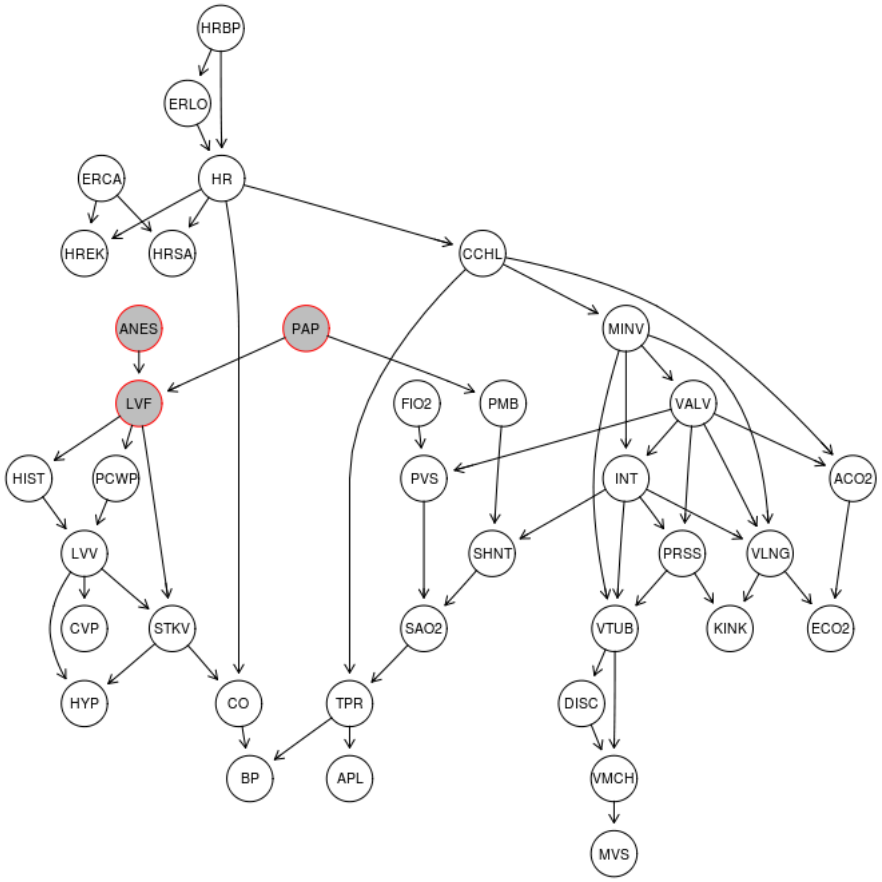

Figure 4-6 Alarm network learned with incorrect rules

In addition to specifying a rule with incorrect certainty factor, the expert can also specify a rule with incorrect strength factors. When the certainty factor is different from the correct value, it can be detected when the final conditional probability for the rule that is learned from data deviates from the estimated probabilistic bounds. Similarly, it is possible to detect incorrect strength values when some of the probability values in the conditional probability table deviates from the estimated probabilistic bounds. Consider a variation of rule RE8 with incorrect strength factor for LVF.

RE31: If LVF = FALSE (0.8) AND HYP = FALSE (0.8) THEN STKV = NORMAL (0.9)

The related rules and the residual evidence that can be derived from RE31 are:

RE32 : If LVF = FALSE AND HYP = TRUE THEN STKV = NORMAL (0.18)

RE33 : If LVF = TRUE AND HYP = FALSE THEN STKV = NORMAL (0.18)

RE34: If LVF = TRUE AND HYP = TRUE THEN STKV = NORMAL (0.036)

The network is bootstrapped from the rules and the complete structure is learned from the data. The conditional probability table for STKV that is learned from data is given in the table below. The conditional probability entry where both LVF and HYP are false lies within the estimated probabilistic bound. This implies the rule has an acceptable value for certainty factor. The conditional probability entries for the last two rows where LVF = TRUE deviates from the estimated probabilistic bound beyond a tolerance value. When this happens we can conclude that the strength value for LVF is incorrect. When the value of strength is updated to 0.99 (from 0.8) as in RE8, all the final probabilities occur between the estimated probabilistic bound which shows that the rule has correct values for certainty factor and strength.

Table 4-8 Alarm - Conditional Probability table for an incorrect rule II

LVF	HYP	Certainty Factor	Probabilistic Bound for STKV = NORMAL	Consistent value of Probability	Probability after learning
FALSE	FALSE	0.9	[0.85,1]	0.93	0.89
FALSE	TRUE	0.18	[0.54,1]	0.77	0.51
TRUE	FALSE	0.18	[0.54,1]	0.77	0.04
TRUE	TRUE	0.04	[0.13,1]	0.57	0.01

Similarly rule verification capabilities can be shown with the online transaction data. Consider that the expert specified an incorrect rule for the online transaction rule base:

RE30 : If FLAG4=1(0.9) THEN RESULT=1 (0.9)

If this rule is used to bootstrap the network we get the network in Figure 4-7 .

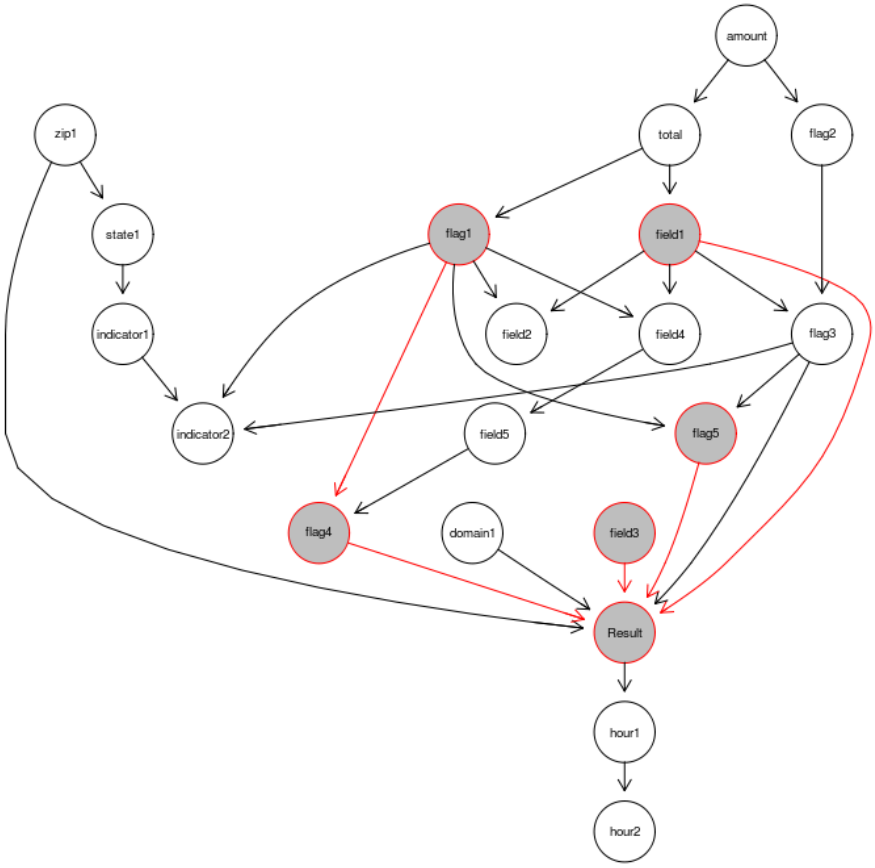

Figure 4-7 Online transaction network learned from incorrect rule

Comparing this network with the one that was bootstrapped with rules RE25-RE29, we can again see that this network has a lower score, slightly higher classification error and edges.

Table 4-9 Comparison of Online transaction networks learned

	Bayesian Network generated with Correct Rules [RE25-RE29]	Bayesian Network generated with Incorrect Rule [RE30]
BIC Score	-320671.4	-327134.7
#Edges	29	30
Classification Error	0.0124	0.0217

These experiments show the ability of the presented approach to identify incorrect or partially correct rules, thus providing an important tool for an expert to validate prior knowledge expressed in the form of rules.

4.5.3 Providing Rule Based Explanations

This experiment shows the explanation capabilities of the system. For this experiment, we use the alarm network. Let us assume that the user made the observation that variable KINK has the value FALSE. The user wants to predict the value of pressure reading. The system performs inference and finds the following:

a) P(PRESS=ZERO) = 0.016, P(PRESS=LOW) = 0.15, P(PRESS=NORMAL) = 0.13 and P(PRESS=HIGH) = 0.71.

b) There are two rules which satisfy the suggestion:

   a. *RE17 : If KINK=FALSE AND VTUBE=LOW THEN PRESS=HIGH*

   b. *RE18 : If VTUBE=NORMAL AND INT=ESOPHAGAL THEN PRESS=HIGH*

The system estimates the conditional probability of each applicable rule based on the observations. For RE17, P(PRESS=HIGH| KINK=FALSE AND VTUBE=LOW) = 0.48 and for RE18, P(PRESS=HIGH|VTUBE=NORMAL,INT=ESOPHAGAL) = 0.7.

From this we can calculate that the Bayes Factor for RE17 is 0.92 and RE18 is 2.3. This means that RE18 is the rule with most explanatory power.

The system generates the following explanation: “We believe with 71% percent confidence that the value of PRESS is HIGH. Our reasoning is based on the following evidence: 95% of the past records under similar circumstances agreed with rule RE18 (If KINK=FALSE AND VTUBE=LOW THEN PRESS=HIGH). Also 3% of the past records add additional evidence that can be explained by RE17 (If VTUBE=NORMAL AND INT=ESOPHAGAL THEN PRESS=HIGH)”.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Certainty factor based expert systems and Bayesian Networks are two popular frameworks to perform uncertain reasoning. Both have their own advantages and disadvantages. In this thesis, we have proposed an approach that combines the ability to efficiently elicit expert knowledge and generate user understandable explanation using rules with the automated reasoning capabilities of the Bayesian Network. The domain knowledge is expressed using rules with certainty factors that are augmented with strength parameters for antecedents. Using the certainty factors and strength parameters, the residual evidence for rules that are variations of the expert specified rules can be estimated and is used to derive their probabilistic bounds for conditional probabilities. A consistent set of probability assignments that satisfy the constraints is then obtained using linear programming. These conditional probability values together with partial structure determined from the rules are used to bootstrap a Bayesian Network. From the data, a complete Bayesian Network whose structure is influenced by the rules is learned. This network can then be used to perform automated inference and can explain its predictions using the expert specified rules. Other issues like incremental learning, identifying incorrect rules, proposing new rules are also addressed, resulting in a consistent framework that combines the advantages of rule based systems with the data driven capabilities of Bayesian Networks.

5.2 Future Work

Currently, the proposed method can handle only static rules and networks. Rules with temporal components have to be added using static nodes. The use of dynamic Bayesian Networks should be explored as a way to more accurately model temporal rules. This will allow the system to handle more expressive rules.

Also, currently the system can only notify the expert of potentially incorrect rules. If the expert insists on keeping the rule, a mechanism must be developed which can find the next best possible structure. This might involve modification of other rules or modification of the network structure. Similarly, it is possible that during incremental learning, the best way to fit the data might require a structural change. Handling incremental structural learning under the constraint of existing rules has to be explored.

REFERENCES

[Agarwal et al, 1993] R. Agrawal, T. Imielinski, A. Swami. Mining Association Rules Between Sets of Items in Large Databases", SIGMOD Conference 1993: 207-216

[Bauer et al,1997] Bauer, E., Koller, D., & Singer, Y. Update rules for parameter estimation in Bayesian Networks. Proceedings of the 13th Annual Conference on Uncertaintyin Artificial Intelligence UAI-97

[Beinlich et al, 1989] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system. In Proc. 2'nd European Conf. on AI and Medicine. , 1989.

[Chickering, 2002] D.M. Chickering. Optimal structure identification with greedy search. Journal of Machine Learning Research, 3:507–554, November 2002b.

[Dempster et al, 1977] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, B 39:1–38, 1977.

[Divya et al,2008] D. Ramachandran, A. Maiti, G. Mwangi, M. Huber, D. Levine, F. Kamangar, R. Schoech, and G.V. Záruba, "Technology to Help Patients Adhere to Treatment Plans - A Brief Introduction to the Teleherence Project," Proceedings of the Biotechnology and Bioinformatics Symposium (BIOT-2008), Arlington, Texas, October 17-18, 2008.

[Friedman, 1997] Learning Belief Networks in the Presence of Missing Values and Hidden Variables. In ICML, 1997.

[Friedman et al, 1997] Friedman, N., & Goldszmidt, M. Sequential update of Bayesian Network structure. Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence UAI-97.

[Friedman, 1998] N. Friedman. The Bayesian structural EM algorithm. In UAI, 1998.

[Gordon and Shortliffe,1990] Gordon J, Shortliffe E. The Dempster Shafer Thoery of Evidence. Shafer, Glenn, and Judea Pearl, eds. (1990). Readings in Uncertain Reasoning. Morgan Kaufmann.

[Heckerman, 1986] Heckerman, D. (1986). Probabilistic interpretations for MYCIN's certainty factors. In Kanal, L. and Lemmer, J., editors, Uncertainty in Artificial Intelligence, pages 167–196. North-Holland, New York.

[Heckerman, 1989] D. Heckerman, E. Horvitz, and B. Nathwani. Update on the Pathfinder project. In Proceedings of the Thirteenth Symposium on Computer Applications in Medical Care, Washington, D, pages 203-207. IEEE Computer Society Press, Silver Spring, MD, November 1989.

[Heckerman, 1992] D. Heckerman. The certainty-factor model. In S. Shapiro, editor, Encyclopedia of Artificial Intelligence, Second Edition, pages 131-138. Wiley, New York, 1992.

[Heckerman and Shortliffe, 1992] D. Heckerman and E. Shortliffe. From certainty factors to belief networks. Artificial Intelligence in Medicine, 4:35-52, 1992.

[Heckerman, 1999] D. Heckerman. A Tutorial on Learning with Bayesian Networks. In Learning in Graphical Models, M. Jordan, ed.. MIT Press, Cambridge, MA, 1999.

[Henrion, 1986] Henrion, M. (1986). Should we use probability in uncertain inference systems? In Proceedings of the Cognitive Science Society Meeting, Amherst, PA. Carnegie–Mellon.

[Horvitz et al, 1988] Horvitz, E., Breese, J., and Henrion, M. (1988). Decision theory in expert systems and artificial intelligence. International Journal of Approximate Reasoning, 2:247–302.

[Jordan, 1999] Learning in Graphical Models , M. Jordan, ed.. MIT Press, Cambridge, MA, 1999.

[JSR 94, 2006] http://www.jcp.org/en/jsr/detail?id=94

[Koller and Friedman, 2009] D. Koller and N. Friedman. Probabilistic Graphical Models: Principles and Techniques , MIT Press , 2009.

[Korver and Lucas, 1993] M. Korver , P. J. F. Lucas. Converting a rule-based expert system into a belief network, Medical Informatics 23:219-241, 1993

[Open Bayes, 2004] Open Bayes for Python . http://www.openbayes.org/

[Robinson, 1977] R. W. Robinson. Counting unlabeled acyclic digraphs. In C. H. C. Little, editor, Combinatorial Mathematics V, volume 622 of Lecture Notes in Mathematics, pages 28–43, Berlin, 1977. Springer.

[Shafer, 1976] Shafer, G. (1976). A Mathematical Theory of Evidence. Princeton: Princeton University Press.

[Shafer and Pearl,1990] Shafer, Glenn, and Judea Pearl, eds. (1990). Readings in Uncertain Reasoning. Morgan Kaufmann.

[Shortliffe and Buchanan, 1975] E.H. Shortliffe and B.G. Buchanan. A model of inexact reasoning in medicine. Mathematical Biosciences, 23:351–379, 1975.

[Shortliffe and Buchanan, 1984] B.G. Buchanan and E.H. Shortliffe, editors. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison–Wesley, Reading,MA, 1984.

[UCSD, 2009] UCSD Data Mining Competition Data Sets. http://mill.ucsd.edu/index.php?page=Datasets

[Yuan and Lu, 2008] Changhe Yuan , Tsai-Ching Lu, A general framework for generating multivariate explanations in Bayesian Networks, Proceedings of the 23rd national conference on Artificial intelligence, p.1119-1124, July 13-17, 2008.

[Zhang and Luo, 1997] Chengqi Zhang and Xudong Luo . Transformation between the EMYCIN Model and the Bayesian Network , Proceedings of the Workshops on Commonsense Reasoning, Intelligent Agents, and Distributed Artificial Intelligence: Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications pages 205 - 220 1997.

BIOGRAPHICAL STATEMENT

Saravanan Thirumuruganathan received his bachelors in Computer Science engineering in 2005 from Sri Krishna College of Engineering and Technology under Anna University, Chennai, India. He worked for two years in Infosys Technologies and one year in Amazon. He started his Masters in Computer Science in 2008 in University of Texas At Arlington. He is continuing with his PhD in Fall 2010. His areas of interests are data mining and artificial intelligence. His hobbies include programming, blogging, reading books, writing poems and contributing to open source projects.