THEORETICAL AND PRACTICAL ASPECTS OF DECISION SUPPORT SYSTEMS BASED ON THE PRINCIPLES OF QUERY-BASED DIAGNOSTICS

by

Parot Ratnapinda

B.E., King Mongkut's Institute of Technology Ladkrabang, 2003

M.S., University of Pittsburgh, 2008

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Parot Ratnapinda

It was defended on

April 18, 2014

and approved by

Marek J. Druzdzel, PhD, Associate Professor, School of Information Sciences

Roger R. Flynn, PhD, Associate Professor, School of Information Sciences

Stephen C. Hirtle, PhD, Professor, School of Information Sciences

Michael C. Lewis, PhD, Professor, School of Information Sciences

Gregory F. Cooper, PhD, Professor, Department of Biomedical Informatics

Dissertation Director: Marek J. Druzdzel, PhD, Associate Professor, School of Information Sciences

Copyright © by Parot Ratnapinda
2014

THEORETICAL AND PRACTICAL ASPECTS OF DECISION SUPPORT SYSTEMS BASED ON THE PRINCIPLES OF QUERY-BASED DIAGNOSTICS

Parot Ratnapinda, PhD

University of Pittsburgh, 2014

Diagnosis has been traditionally one of the most successful applications of Bayesian networks. The main bottleneck in applying Bayesian networks to diagnostic problems seems to be model building, which is typically a complex and time consuming task.

Query-based diagnostics offers passive, incremental construction of diagnostic models that rest on the interaction between a diagnostician and a computer-based diagnostic system. Every case, passively observed by the system, adds information and, in the long run, leads to construction of a usable model. This approach minimizes knowledge engineering in model building.

This dissertation focuses on theoretical and practical aspects of building systems based on the idea of query-based diagnostics. Its main contributions are an investigation of the optimal approach to learning parameters of Bayesian networks from continuous data streams, dealing with structural complexity in building Bayesian networks through removal of the weakest arcs, and a practical evaluation of the idea of query-based diagnostics. One of the main problems of query-based diagnostic systems is dealing with complexity. As data comes in, the models constructed may become too large and too densely connected. I address this problem in two ways. First, I present an empirical comparison of Bayesian network parameter learning algorithms. This study provides the optimal solutions for the system when dealing with continuous data streams. Second, I conduct a series of experiments testing control of the growth of a model by means of removing its weakest arcs. The results

show that removing up to 20 percent of the weakest arcs in a network has minimal effect on its classification accuracy, and reduces the amount of memory taken by the clique tree and by this the amount of computation needed to perform inference. An empirical evaluation of query-based diagnostic systems shows that the diagnostic accuracy reaches reasonable levels after merely tens of cases and continues to increase with the number of cases, comparing favorably to state of the art approaches based on learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This dissertation is the summary of almost six years of work in the Decisions Systems Laboratory (DSL) at the University of Pittsburgh. I would like to use this section to thank several people who have supported me over the years.

First of all, I would like to thank my advisor Marek Druzdzel, for all his advice, helpful feedback, and great lessons throughout the years. I took Marek's Decision Analysis and Decision Support Systems class in 2007 and this was critical in shaping my dissertation topic – the MARILYN system was the term project in this course. From Marek, I learned many useful skills to become a good professor, such as being a good researcher, a decent writer, and a better person. I am grateful to the others members in my Ph.D. committee, as well. Dr. Roger Flynn was my first academic advisor when I started my M.S. program. Roger asked useful questions and gave feedback that helped me in preparing for my preliminary exam. I enjoyed taking courses from Dr. Stephen Hirtle which improved my reading and writing skills during my earlier year in the program. I worked as an assistant to Dr. Michael Lewis for one semester and enjoyed interacting with him. Dr. Greg Cooper's work is directly related to my research. I took his course, Probabilistic Methods in Fall 2010, which gave me a lot of insight into the field of Bayesian network. Greg gave useful feedback and comments when I presented my work in the Intelligent Systems Program. I am grateful to the staff in the School of Information Sciences. Special thanks go to Brandi Belleau, Wesley Lipschultz and Mary Stewart. I would like to thank my supervisor Karen Bassett at the School of Education for giving me the opportunity to work as a lab consultant, and for giving me permission to use the help desk data in my experiments.

I would also like to thank members of the Decisions Systems Laboratory, who are my colleagues and also friends. Special thanks go to Martijn de Jongh and Mark Voortman

for helping me with coding and commenting on my experiments. I would like to thank Christopher Faraglia and Jidapa Kraisangka for their works on improving the MARILYN interface significantly.

I would like to thank my American families and friends for all the great times I have had with them outside of my studies. I give special thanks to Zach Leight and Jeff Guerrero for proofreading my papers and my dissertation. My host family Ken and Julie Sackman for their support emotionally and financially. I would like to thank my immediate family, especially my mom who is watching me from heaven. She supported me emotionally in the beginning of my studies, but did not live to see me graduate. Last, but not least, my most sincere thanks go to my wife, Wiranya, for her love, care, and support throughout my studies.

"We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard." — John F. Kennedy

1.0 INTRODUCTION

Bayesian networks (BNs) are convenient tools for building models in various domains. One of the most successful applications of Bayesian networks has been diagnostics. However, constructing BN models is a complex and time consuming task. Building a BN for a domain involves three tasks: (1) identifying important variables, (2) identifying relationships among these variables, and (3) obtaining probabilities [Druzdzel and van der Gaag, 2000]. For a sufficiently complex diagnostic model, it is not uncommon for these tasks to take hundreds of hours [Onisko, 2003].

We can also learn models from data. Given a sufficiently large set of past cases, we can learn both the structure and the parameters of a Bayesian network [Pearl and Verma, 1991, Cooper and Herskovits, 1992, Spirtes et al., 1993]. Although model construction from data can significantly reduce knowledge engineering effort, large and complete data sets are hard to find. There are many complex devices or systems, such as airplanes for example, that do not break often, or at least are not supposed to break often.

There have been several lines of research outside of learning from data that focus on model building. The first approach focuses on providing more expressive building tools. The noisy-OR model [Pearl, 1988, Henrion, 1989] and its generalizations [Díez, 1993, Srinivas, 1993] simplify the representation and elicitation of independent interactions among multiple causes. Kraaijeveld and Druzdzel [2005], for example, describes a system that aids rapid interactive construction of large diagnostic models, simplified by means of a three level structure and noisy-OR gates. Heckerman [1990] developed the concept of *similarity networks* in order to facilitate structure building and probability elicitation. The second approach, usually referred to as knowledge-based model construction (KBMC), emphasizes aiding of model building by an automated generation of decision models from a domain knowledge-

base guided by the problem description and observed information [Breese et al., 1994]. The third approach is to apply system engineering and knowledge engineering techniques for aiding the process of building Bayesian networks. Mahoney and Laskey [1996], Laskey and Mahoney [1997] address the issues of modularization, object-orientation, knowledge-base, and evaluation in a spiral model of the development cycle. Koller and Pfeffer [1997], Pfeffer et al. [1999] developed Object-Oriented Bayesian Networks (OOBN) that use objects as organizational units to reduce the complexity of modeling and increase the speed of inference. Lu et al. [2000] proposes mechanism-based model construction, in which models are constructed from a collection of mechanisms based on scientific laws or pieces of existing models. Ibargengoytia et al. [2006] proposes to learn a Bayesian network model for a normal mode of operation, for which data is typically available, and then detect anomalies as deviations from this model.

1.1 PROBLEM STATEMENT AND MOTIVATION

Even though Bayesian network (BN) models have proven useful in diagnostic domains [Heckerman and Breese, 1999], they are quite hard to field in practice. Interestingly, it is not computational complexity that is critical here. The main hurdle in applying Bayesian networks to complex diagnostic problems seems to be model building.

Query-based diagnostics [Agosta et al., 2008] offers passive, incremental construction of diagnostic models that rest on the interaction between a diagnostician and a computer-based diagnostic system. Every case, passively observed by the system, can add information to the model, and in the long run, construct a usable model. This approach minimizes knowledge engineering in model building.

While this idea is appealing, it has undergone only limited testing in practice. There are two existing prototypes implementing this approach. An industrial prototype of the system has been implemented and fielded at Intel and tested in the domain of diagnostics and corrective maintenance of factory equipment [Agosta et al., 2010]. A widely accessible prototype, called MARILYN [Pols, 2007] has been developed at the University of Pittsburgh.

Neither of the two prototypes, nor the very idea of a system that eliminates completely the knowledge engineering phase and learns successively from diagnostic cases have undergone a formal evaluation. This dissertation focuses on the theoretical and practical aspects of the MARILYN system, including its formal evaluation.

There are several technical problems that need to be addressed for practical query-based diagnostic systems to be built. One such problem is processing of continuous streams of data. Each query-based diagnostic system must work in an environment where there are new data records/cases added systematically to the body of data. A practical question is whether they should be processed incrementally or periodically in batch mode. Another problem is the continuous increase in complexity of the models' structure. At some point, if unmanaged, the models may become intractable. One way of controlling the growth of a model is to systematically simplify its structure by removing its weakest arcs. There are two research questions: (1) how many arcs can we remove with minimal impact on the model's accuracy?, and (2) what are the benefits of removing weakest arcs in terms of the reduction of memory requirements and computation time?

1.2 CONTRIBUTIONS

The main contributions of this dissertation are: (1) an investigation of the optimal way of learning parameters of Bayesian networks from continuous data streams, (2) dealing with structural complexity in building Bayesian networks, and (3) a practical evaluation of the idea of query-based diagnostics.

Query-based diagnostics approach requires fewer cases for learning than the other simple Bayesian networks in performing classification.

I describe a series of experiments that subject a prototype implementing passive, incremental model construction to a rigorous practical test. While query-based diagnostics are more suitable to diagnostics application, we can also apply the query-based diagnostics approach to classification tasks which contain only discrete variables. I evaluate the prototype (MARILYN) systematically, based on five data sets: three data sets from the UCI

Machine Learning Repository and two help desk data sets at the University of Pittsburgh. I compare the system's diagnostic accuracy with other Bayesian learning algorithms: Naive Bayes, a Bayesian search algorithm Greedy Thick Thinning, and Tree Augmented Naive Bayes algorithm [Ratnapinda and Druzdzel, 2009, 2011].

There is no significant difference in terms of parameter accuracy and classification accuracy among the three Bayesian network parameter learning approaches: the batch learning, the incremental batch learning, and the online learning when learning Bayesian network parameters from continuous data streams.

To find an optimal approach to perform Bayesian network parameter learning algorithms for continuous data streams, I conduct a series of experiments to compare two notable parameter learning algorithms: the EM algorithm and the online EM algorithm. I use several real data sets to create gold standard Bayesian network models. I use these models to generate continuous streams of data. I learn the parameters from these streams of data using three approaches: *batch learning*, *incremental batch learning*, and *online learning*. I measure the time taken by the learning procedure, compare the accuracy of the learned parameters to the original (gold standard) parameters that have generated the data, and test the diagnostic accuracy of the learned models [Ratnapinda and Druzdzel, 2013].

The complexity of a Bayesian network can be reduced by removing its weakest arcs, without compromising its accuracy.

Practical models based on Bayesian networks often reach the size of hundreds or even thousands of variables. In some applications, such as query-based diagnostics, models are built dynamically and grow in size with time. The amount of computation necessary to perform belief updating may become prohibitive. One way of controlling the growth of a model is to systematically simplify its structure by removing its weakest arcs. I perform an empirical evaluation of structural simplification of Bayesian networks by removing weak arcs. I use six networks built from real data sets. I systematically remove arcs from the weakest to the strongest, relying on four measures of arc strength called strength of influence. I measure the classification accuracy of the resulting simplified models, the amount of memory taken by the clique tree and the amount of computation needed to perform inference [Ratnapinda and Druzdzel, 2014].

1.3 ORGANIZATION OF THE DISSERTATION

The remainder of this dissertation is structured as follows. Chapter 2 introduces terms and concepts that are necessary for subsequent chapters. Chapter 3 describes the idea of query-based diagnostics and its implementation. Chapter 4 presents an approach for learning parameters of Bayesian networks from continuous data streams. Chapter 5 addresses the issues of dealing with structural complexity in building Bayesian networks. Chapter 6 presents an empirical evaluation of query-based diagnostics. Chapter 7 evaluates and discusses several aspects of query-based diagnostics in a practical domain. Chapter 8 summarizes the dissertation and outlines directions for further research.

2.0 BACKGROUND

This chapter gives a review of the concepts involved in query-based diagnostic systems, notably Bayesian networks, noisy-OR gates, the EM algorithm and the strength of influence.

2.1 BAYESIAN NETWORKS

Bayesian networks [Pearl, 1988] are acyclic directed graphs representing joint probability distributions over sets of variables. Every node is the graph represents a random variable. Lack of an arc between two nodes represents conditional independence between the variables that these nodes represent. Nodes are quantified by means of conditional probability tables (CPTs), representing the probability distribution of the variables that they represent, conditional on their parent variables in the graph. Nodes without parents are specified by prior probability distributions. The joint probability distribution over a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ [1] can be obtained by taking the product of all prior and conditional probability distributions:

$$\Pr(\mathbf{X}) = \Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i | Pa(X_i)) \ . \tag{2.1}$$

Figure 1 shows a simple Bayesian network modeling two computer hardware problems. The variables in this model are: *Computer is old*, *Damaged CPU*, *Damaged VGA card*, *Hard disk LED does not work* and *Monitor LED never goes to steady green*. Each of the variables in the model is binary, i.e., has two outcomes: *True* and *False*.

[1] I will use the following notation. The capital letters are variables (e.g., X), and lower case letters are outcomes (e.g.,x).

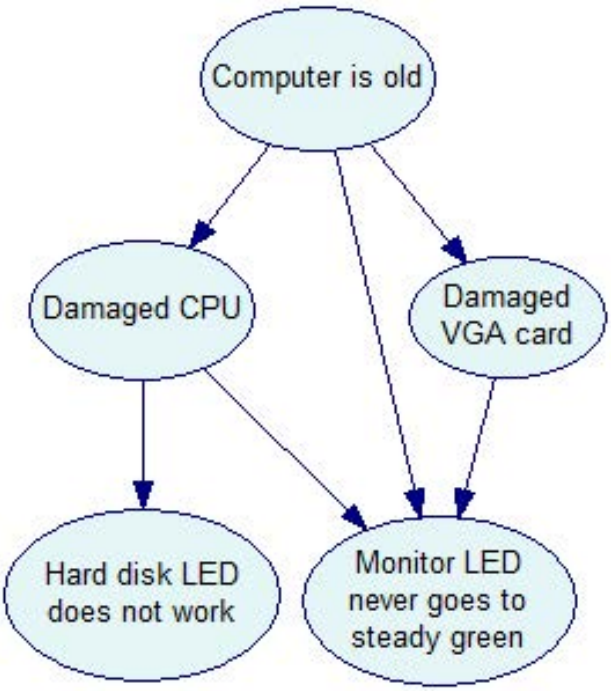

Figure 1: An example Bayesian network modeling computer hardware problems

A directed arc between *Damaged CPU* and *Hard disk LED does not work* indicates that *Damaged CPU* will affect the probability that *Hard disk LED does not work*. Similarly, an arc from *Computer is old* to *Damaged VGA card* indicates that computer age influences the likelihood of a damaged VGA card.

The most important type of reasoning in Bayesian networks is known as belief updating. Belief updating amounts to computing the probability distribution over variables of interest given the evidence. For example, in the model of Figure 1, the variable of interest could be *Damaged CPU* and the BN could compute the posterior probability distribution over this node given the observed values of *Computer is old*, *Hard disk LED does not work*, and *Monitor LED never goes to steady green*. Once the network has updated the probability values, these can be used to make a diagnostic decision.

2.2 THE NOISY-OR GATE

Bayesian networks suffer from a practical problem: Because CPTs represent the probability distribution of a node conditional on all combinations of parent variables, their size grows exponentially with the number of parents. Table 1 shows the CPT for the node *Monitor LED never goes to steady green*. The node has three parents and CPT consists of $2^3 = 8$ different probability distributions.

Table 1: Conditional probability table of the node *Monitor LED never goes to steady green*

Damaged CPU	True				False			
Damaged VGA card	True		False		True		False	
Computer is old	True	False	True	False	True	False	True	False
True	0.7696	0.712	0.424	0.28	0.712	0.64	0.28	0.1
False	0.2304	0.288	0.576	0.72	0.288	0.36	0.72	0.9

One solution to the exponential growth of CPTs is application of Independence of Causal Influences (ICI) models [Díez and Druzdzel, 2006]. The ICI models assume that parent variables can cause the effect independently of each other. This assumption allows to reduce the number of parameters needed to specify an interaction from exponential to linear in the number of parents.

The noisy-OR gate [Pearl, 1988, Henrion, 1989] is a probabilistic extension of the deterministic OR gate. Each variable in a noisy-OR gate is binary and has two states: *present* and *absent*. Presence of the parent variables X_i effects the presence of the child variable Y. If all the parent variables are *absent*, then the child variable is also *absent*.

This can be modeled in propositional logic by the following Boolean function:

$$Y = X_1 \vee X_2 \vee \ldots \vee X_n \ . \tag{2.2}$$

The noisy property of the noisy-OR gate states that if a parent cause variable X_i is *present* and all other parent variables are *absent*, X_i has a probability p_i of causing the effect y. The probability of variable Y present given that only X_i is present is:

$$p_i = Pr(y|\overline{x}_1, \ldots, x_i, \ldots, \overline{x}_n) \ .$$

The probability of y given a subset $\mathbf{X}$ of causes present is:

$$p(y|\mathbf{X}) = 1 - \prod_{i=1}^{n}(1 - p_i) \ . \tag{2.3}$$

In general, it is infeasible to explicitly include all possible causes of an effect. I use an extension of the noisy-OR gate called the *leaky noisy-OR* gate [Henrion, 1989, Díez, 1993] for query-based diagnostics. The parameter p_i of a leaky noisy-OR gate is defined as the probability that Y will be true if X_i is present and every other parent of Y, including unmodeled causes of Y (the leak), are absent.

2.3 BAYESIAN NETWORK PARAMETER LEARNING

The most flexible algorithm for learning Bayesian network parameters is the EM (Expectation Maximization) algorithm [Dempster et al., 1977, Lauritzen, 1995]. While there are several variants of the EM algorithm, two are most notable: the basic EM algorithm [Dempster et al., 1977] and the *online EM* algorithm [Sato and Ishii, 2000, Liang and Klein, 2009, Cappe, 2010].

The Expectation-Maximization (EM) algorithm [Dempster et al., 1977] is a widely used method of computing maximum likelihood estimates given incomplete data. One application of the EM algorithm is in learning parameters of Bayesian networks. The EM algorithm consists of two steps: (1) the expectation step (E-step) that uses the current parameters to compute the expected values of the missing data, and (2) the maximization step (M-step), during which the maximum likelihood of the parameters are estimates based on the expected values from the E-step. The EM process repeats until it converges to the maximum likelihood or it reaches a pre-defined improvement threshold.

In the *basic EM algorithm*, during each iteration, we perform the E-step to calculate the expected sufficient statistics across all observation. Then, we do the M-step once at the end to re-estimate the parameters using sufficient statistics from the E-step. Following Cappe [2010], we describe the basic EM algorithm as follows: Given n observations, $Y_1, \ldots, Y_n$, and an initial parameter guess θ_0, do, for $k \geq 1$.

$$\begin{aligned} \textbf{E-step:} \quad S_{n,k} &= \frac{1}{n}\sum_{t=1}^{n} E_{\theta_{k-1}}\left[s(X_t, Y_t)\middle| Y_t\right] \\ \textbf{M-step:} \quad \theta_k &= \bar{\theta}\left(S_{n,k}\right) \ . \end{aligned}$$

We define X_t as a random variable corresponding to a variable Y_t.

The *online EM algorithm* performs the E-step and follows it by the M-step after each observation. In the E-step, the online EM algorithm uses stochastic approximation instead of sufficient statistics [Cappe, 2010].

$$S_n = S_{n-1} + \gamma_n \left(E_{\bar{\theta}(S_{n-1})}\left[s(X_n, Y_n)\middle| Y_n\right] - S_{n-1}\right) \ .$$

Updating the model after each observation may lead to a poor approximation, which the online EM algorithm avoids by interpolating between S_{n-1} and the expectation values. The value that weights between a previous value and an expected value is a positive step size called γ_n. We use the generalized version of the online EM algorithm, proposed by Cappe [2010], in the following way. Given S_0, θ_0 and a sequence of step sizes $(\gamma_n)_{n\geq 1}$, do, for $n \geq 1$.

$$\begin{aligned} \textbf{E-step:} \quad S_n &= (1-\gamma_n)S_{n-1} + \gamma_n E_{\theta_{n-1}}\left[s(X_n, Y_n)\middle| Y_n\right] \\ \textbf{M-step:} \quad \theta_n &= \bar{\theta}(S_n) \ . \end{aligned}$$

We have to select a step size γ_n for the online EM. We use the assumption often used in the stochastic approximation literature that $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$. If we use $\gamma_n = 1/n^\alpha$, then the range of values for $0.5 < \alpha \leq 1$ is valid.

2.4 STRENGTH OF INFLUENCE

Strength of influence [Koiter, 2006] measures the strength of an arc between two nodes in a model. It is based on the differences between the posterior probability distributions of a child node for each possible state of the parent. While for some applications (e.g., visualization or sensitivity analysis) Koiter [2006] suggested taking the maximum difference, in this dissertation, I base the link strengths on the average over all differences because it covers a broader range of situations. The strength of influence is defined as follows:

$$\frac{1}{n}\sum_{i=0}^{n} D(P(A), P(B|A)) \ ,$$

where node B is the child of node A, A has n states and D is a function measuring the distance between two distributions.

Koiter [2006] uses four distance measures D: Euclidean, Hellinger, J-divergence and CDF. If P and Q are two probability distributions over n states, the Euclidean distance between them is defined as:

$$E(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \ .$$

Euclidean distance focuses on absolute differences between probabilities and does not correctly account for their relative differences. For example, the same absolute difference of 0.1 between two small probabilities (e.g., 0.01 and 0.11) seems much larger than between two moderate probabilities (e.g., 0.6 and 0.7). Two popular measures have been proposed to account for relative differences: Hellinger distance [Kokolakis and Nanopoulos, 2001] and Kullback-Leibler (KL) distance [Kullback and Leibler, 1951].

Hellinger distance between P and Q is defined as follows:

$$H(P,Q) = \sqrt{\sum_{i=1}^{n}(\sqrt{p_i} - \sqrt{q_i})^2} \ .$$

J-divergence [Johnson and Sinanovic, 2000] is calculated by using the average of the two possible values of the KL distance. KL distance is defined as:

$$KL(P,Q) = \sum_{i=1}^{n}(p_i log_2(\frac{p_i}{q_i})) \ .$$

J-divergence is then defined as:

$$J(P,Q) = \frac{KL(P,Q) + KL(P,Q)}{2} \ .$$

Cumulative Distribution Functions (CDF) distance [Kraaijeveld, 2005] is defined as:

$$C(P,Q) = \frac{1}{n-1}\sum_{i=1}^{n}|P(P \leq p_i) - P(Q \leq q_i)| \ .$$

All four measures of link strength are symmetric. The range of these measures of link strength is normalized to the interval $[0,1]$. I demonstrate the differences among the four measures of link strength by means of the following example after Ebert-Uphoff [2007]. Figure 2 shows a simple Bayesian network with two binary nodes A and B and b as their only free numerical parameter. Figure 3 shows how the four measures of strength of the arc $A \rightarrow B$ change as a function of b. Please note that when there is only one parent, Euclidean distance and CDF distance are equal to each other. The values of the strength of influence decreases monotonically as the conditional distribution $P(B|A)$ becomes symmetric. J-divergence is the least linear of the four measures.

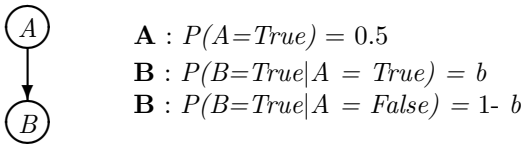

Figure 2: A simple Bayesian network illustrating the four measures of link strength. The interaction between nodes A and B is quantified by means of one free parameter b.

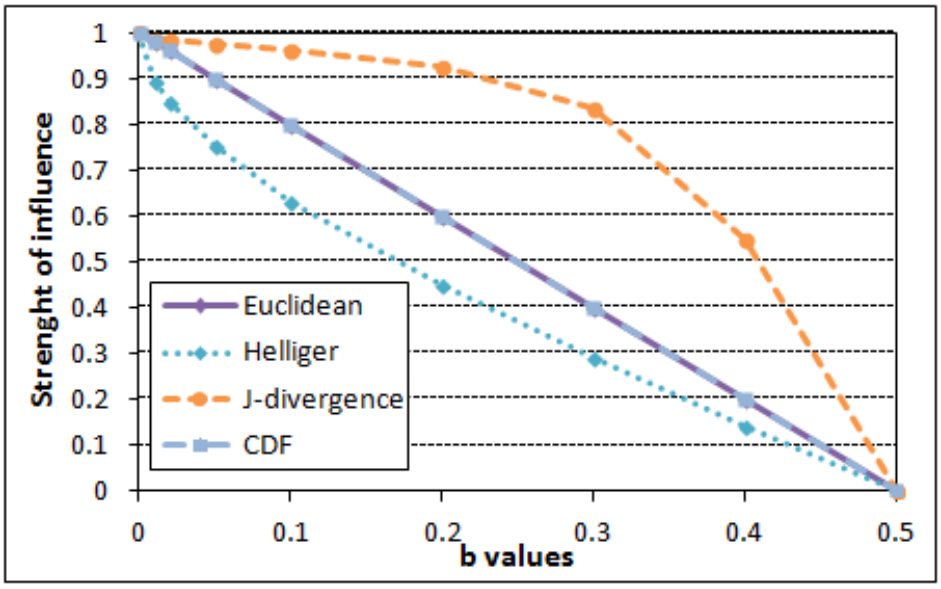

Figure 3: Strength of influence as a function of the parameter b for the network of Figure 2

3.0 QUERY-BASED DIAGNOSTICS

This chapter describes the idea of query-based diagnostics (QBD) and a web-based implementation of query-based diagnostics called MARILYN [Pols, 2007]. MARILYN is written in C# and ASP.NET, using the Microsoft SQL database to store data. It utilizes the Bayesian reasoning engine SMILE©[1] running under the Microsoft Windows Vista Server. MARILYN is available on the Decision Systems Laboratory's web site at the following location: `http://barcelona.exp.sis.pitt.edu/`.

3.1 FUNDAMENTALS OF QUERY-BASED DIAGNOSTICS

Agosta et al. [2008] proposed an approach that eliminates knowledge engineering altogether. In what they call query-based diagnostics, they propose embedding a diagnostic aid in existing systems for diagnostic record keeping. A diagnostician working on a case, recording symptoms and other findings along with the final diagnosis, without being aware of it, participates in constructing a simplified Bayesian network model that supports future cases. From the theoretical perspective, the idea is a combination of structure elicitation and incremental learning. The diagnostician provides the system with a basic distinction between symptoms, background information, and the final diagnosis. Past cases solved by diagnosticians can provide considerable information about the domain. Every new case acquired by the system adds useful information and, in the long run, leads to building a usable model. As cases accrue, the system refines the structure and the parameters of such model and improves its accuracy.

[1] `http://genie.sis.pitt.edu/`

3.2 MODEL STRUCTURE

The Bayesian networks constructed by MARILYN use a simplified structure called the BN3M model [Kraaijeveld and Druzdzel, 2005], which distinguishes three fundamental types of variables:

- *Fault* variables, which represent the problems that the diagnostician wants to identify (e.g., a disease or a device malfunction).

- *Observation* variables, which include observed symptoms and results of diagnostic tests.

- *Context* variables, which are the background, history, or other information known by the technician performing the diagnosis that may influence the probability of a fault and, therefore, are relevant to the diagnosis.

The structure of BN3M networks consists of three levels, with the *context information* variables on the top, the *fault* variables in the middle, and the *observation* variables at the bottom. Influences are possible only between neighboring layers.

Figure 4 shows an example of this structure. The first context variable, *User is a registered student*, influences the variable *User has no print quota*. The second context variable *Computer lab is busy* influences the faults *Printer is backing up* and *Printer is out of paper*. *No print job out* is influenced by any three of the fault variables. *Trays 5 and 6 are empty* is influenced only by the fault *Printer is out of paper*.

MARILYN works with a simplified structure of the network, sometimes called BN2O (Bayesian Network with 2-layer of noisy-OR gates). The structure of BN2O networks consists of two layers, with the *fault* variables in the top layer, and the *observation* variables at the bottom layer. Influences are possible only between neighboring layers. All nodes in the BN2O networks are discrete and, therefore, we can apply the BN2O structure to classification problems containing only discrete variables.

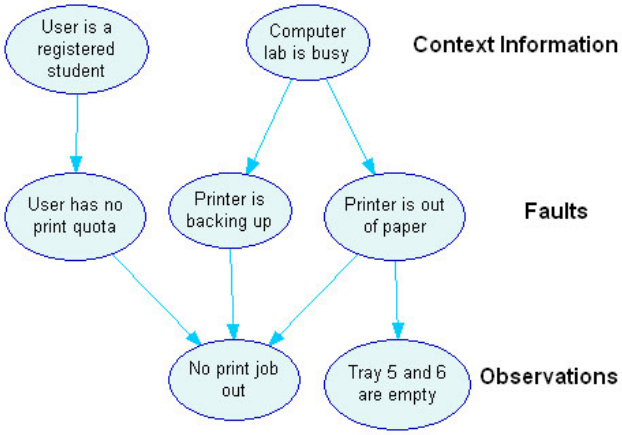

Figure 4: An example of a BN3M model

3.3 IMPLEMENTATION

Figure 5 shows MARILYN's architecture. MARILYN appears to a user as a computer program for logging case data. A user interacts with it through a web browser, entering elements of the case at hand. The case data are entered in free text format, and the system performs simple text matching to suggest values entered in prior cases.

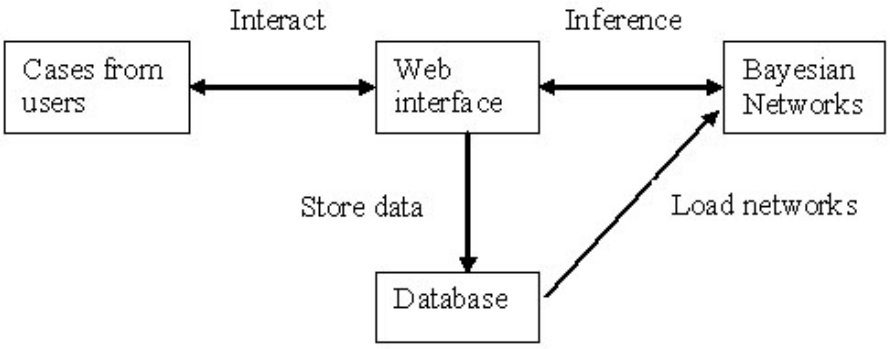

Figure 5: MARILYN's architecture

Figure 6 shows MARILYN's web user interface. There are three panels of a diagnostic session: symptom, background information, and diagnosis. The two tabs on the left hand side

are for adding symptoms, and background information. The right hand side panel presents a user with a list of most likely diagnoses implied by the data entered so far. Behind the scene, MARILYN constructs a Bayesian network from the prior cases stored in the database and, ultimately, adds the current case to the database.

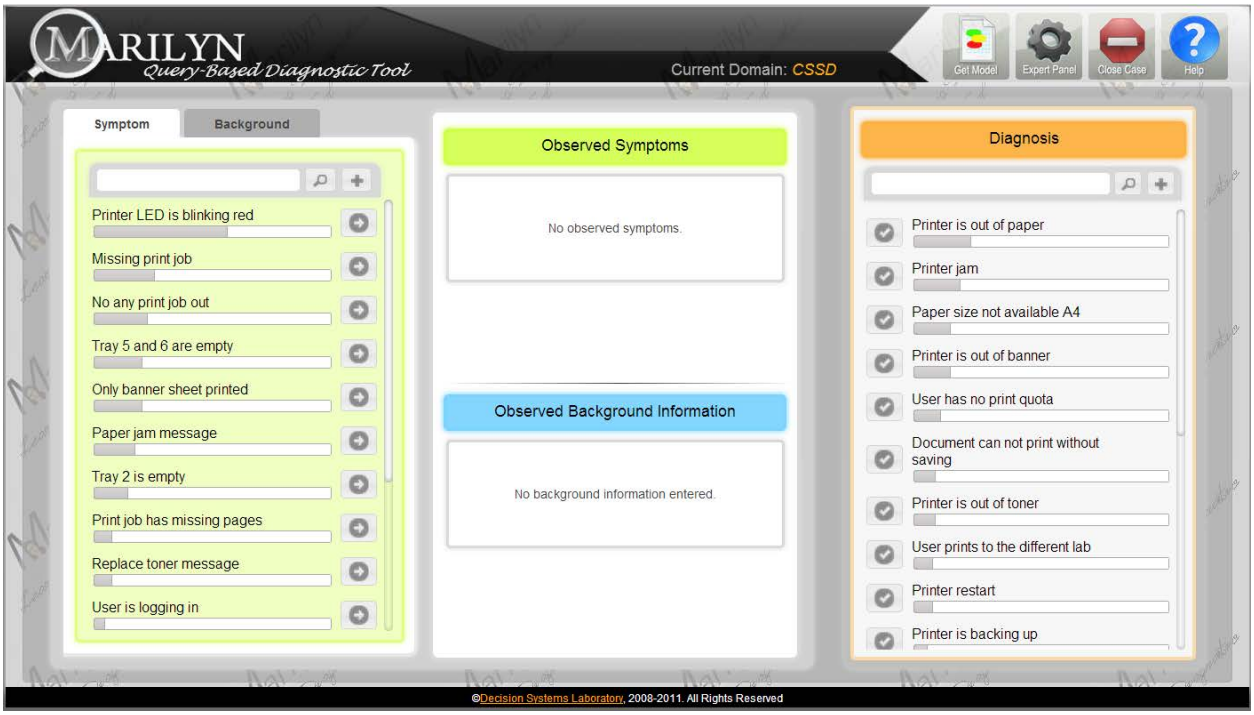

Figure 6: Implementation of MARILYN's website

3.3.1 Prior knowledge

Algorithm 1 Query-based diagnostics

1: **for** each session **do**
2: [Build a Bayesian network]
3: [Learn Bayesian network parameters using the EM algorithm]
4: [Add the case to the database of cases]
5: **end for**

Algorithm 2 Build a Bayesian network

Input: *nodes* table NT; *arcs* table AT
for each node X in *nodes* table NT **do**
  Add X to the model M
  Set the node X name
  Set the node X outcomes to *True* and *False*
end for
for each node X in M **do**
  Find the number of parent nodes P per child nodes C of X in *arcs* table AT
  Add arc for C for each parents P to the model M
  switch P **do**
    case *0* :
      Set the node C type to CPT type
      Set the node C prior probability distributions to 0.1, 0.9
    case *1* :
      Set the node C type to CPT type
      Set the node C conditional probability distributions to 0.8, 0.2, 0.1, 0.9
    otherwise :
      Set the node C type to noisy-MAX type
      Set the node C conditional probability distributions for all P to 0.8, 0.2
      Set the node C leak parameters to 0.1, 0.9
end for
Save the initial model M to GeNIe file called *dataem.xdsl*

Algorithm 1 explains fours steps of query-based diagnostics for each diagnosis session. When Marilyn starts, it constructs a Bayesian network from the existing database (in the very beginning, this database is empty). The database consists of six tables: *arcs*, *diagnosis*, *domains*, *nodes*, *lablog*, and *emlog*. The first four tables store the information about causal interactions among variables, the number of diagnostic sessions that have been stored by the system, the diagnostic domains, and variables, respectively. The last two tables store data

Algorithm 3 Learn Bayesian network parameters using the EM algorithm

1: [Loading Data Phase]
2: Copy *lablog* table LT to *emlog* table EMT
3: Drop column *id* and *date* in EMT
4: Export EMT to a text file *output.txt* OP
5: [EM Phase]
6: Load *dataem.xdsl* from Algorithm 2
7: Load OP
8: Set Equivalent Sample Size to 10
9: Invoke EM parameter learning
10: Save the final model M to a GeNIe file called *lem.xdsl*

for each session and store the diagnostic logs used in refining the model parameters.

Marilyn constructs the BN3M structure by going through all diagnostic cases entered so far and connecting all context variables and and all observation variables to the fault node observed in the case (i.e., the final diagnosis, as indicated by the diagnostician). This provides a graph skeleton that is subsequently quantified in the following way. All prior probability distributions are set to 0.1/0.9. All conditional probability distributions are set to 0.8/0.2 (see algorithm 2). The EM algorithm, which Marilyn subsequently invokes, treats these initial values as prior probability distributions. I perform the parameters smoothing by setting the Equivalent Sample Size to 10. Then, the EM refines the prior probability distributions by means of the records accrued during the past diagnostic cases. While the above priors are arbitrary, I found that they are capable of inducing reasonable behavior on the part of the model, even if the number of existing records is small. There is an increasing body of evidence that the precise values of parameters are not crucial in practice [Pradhan et al., 1996, Oniśko and Druzdzel, 2013].

The final model, i.e., model obtained after the parameter refinement stage, is used by Marilyn to generate a list of most likely diagnoses for the current diagnostic case (see algorithm 3).

Algorithm 4 Add the case to the database of cases

1: **Input:** *nodes* table NT; *arcs* table AT; *lablog* table LT; a list of observation LO; a list of context LC; a diagnosis D
2: [Saving diagnosis D]
3: **if** D not exists in NT **then**
4: Add D to NT
5: **end if**
6: Get DID id from NT and add DID to LT
7: [Saving context C]
8: **for** each C in LC **do**
9: **if** C not exists in NT **then**
10: Add C to NT
11: **end if**
12: Get CID id from NT and add CID to LT
13: Add an arc from C to D into AT
14: **end for**
15: [Saving observation O]
16: **for** each O in LO **do**
17: **if** O not exists in NT **then**
18: Add O to NT
19: **end if**
20: Get OID id from NT and add OID to LT
21: Add an arc from O to D into AT
22: **end for**

3.3.2 User interface

This section explains the beginning state in Algorithm 4.

The MARILYN interface follows a diagnostician work-flow by default. There are three phases of a diagnostic session: observation, context information, and diagnosis. These correspond to different panels (1) *Symptoms*, (2) *Background*, and (3) *Diagnosis*.

A user can enter information in the text box located on top of each panel. Under the text box, a list of suggested information related to the current panel ranked from the most to the least probable. A user will click on the *Symptom* and the *Background* label to switch between these steps.

Suppose a lab user has told a lab consultant that she has not received her print job. This user is a registered student, who tried to print a PDF document from the University *CourseWeb* web site.

In the first step, the lab consultant, who will work with a new Symptom panel, in which he can input some related observations to the lab user problems which is *No print job out*. A lab consultant can selecting the *No print job out* on the lists to enter this observation shown in Figure 7. A lab consultant can use a search function to find the related information shown in Figure 8. The search function will match any texts (not case sensitive) with three or more letters. Once she has finished entering observations, the lab consultant can click the *Background* panel next to the *Symptom* panel to proceed onto the context information screen.

In the second step, the lab consultant may input history or information of the problems. She may check the lab user printing balance to see if the lab user has enough balance to print. With printing problems, the lab consultant will browse the print queue server called *Pharos* to check the history of the lab user print jobs. After gathering all related information, she will input three variables: *User balance is not zero*, *No print job in Pharos system* and *Print PDF from CourseWeb* in any order from the lists in the *Background* panel shown in Figure 9. Once entering context information is done, the lab consultant can proceed to the final panels or go back to work on the *Symptom* panel.

In the third step, the user enters the final diagnosis for the case at hand. She can

Figure 7: A user selects an observation in the *Symptom* panel.

select the diagnoses which were suggested by the MARILYN model or enter a new diagnostic. MARILYN diagnostic panel suggests a list of possible diagnoses ranked by their posterior probabilities, as implied by the underlying model shown in Figure 10. In this example, the lab consultant worked with the network that is based on twenty five computing lab help desk cases, and the system was able to give a correct suggestion to the lab consultant. The first suggestion was *Document can not print without saving*. This refers to a specific requirement of the computing lab that the lab users can not print some documents directly from the web site without saving them first to local space on the disk. Next, the lab consultant selects the causes by clicking on check box in the *Diagnosis panel* shown in Figure 11. Figure 12 shows a final screen after a lab consultant entered all the information.

The consultant confirms the session by clicking the *Close Case* icon on the top right hand corner in order to save the session in the database. If the causes do not appear on the list, the lab consultant can type in a text field and then click the add (+) button , the new causes will be added to the *causes* list. The user can export a built model in .xdsl format

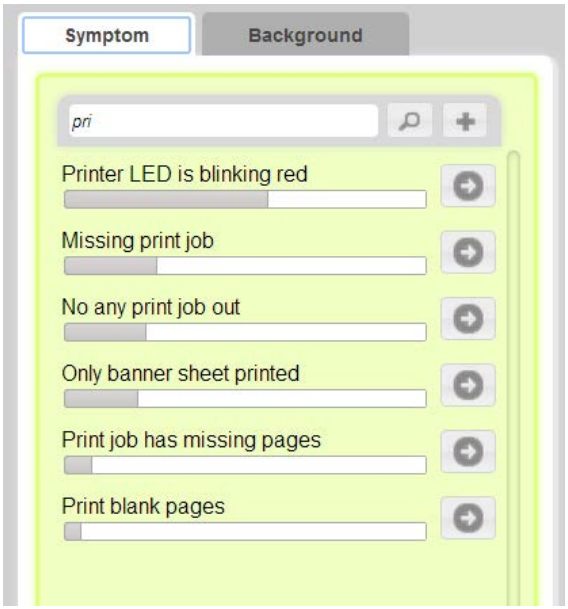

Figure 8: A user finds related information using MARILYN search function.

file by clicking the *Get Model* icon on the top right hand corner. The .xdsl file needs to be opened in a program called GENIE. MARILYN allows for working with other domains by using the *Expert Panel* icon.

The MARILYN interface can be adapted to work with a typical classification data set by applying the simplified BN2O structure. In this case, there are two phases of a diagnostic session: observation and diagnosis. These correspond to different panels (1) *Symptoms*, and (3) *Diagnosis*. One limitation of MARILYN is that it only works with discrete variables, which is a common requirement in Bayesian network.

Suppose a user is working with the *SPECT Heart* data set (details in Chapter 6) on a record with three symptoms present (id4, id9, and id17).

In the first step, the user can input three observations: *id4*, *id9*, and *id17* (Figure 13). Once the user has finished entering observations, he can proceed to the final panels.

In the last step, the user enters the final diagnosis for the case at hand. The user finds the final diagnosis from the original data set, which is *Present*, in which matches the MARILYN suggestion. Figure 14 shows the final screen after the user has entered all the information.

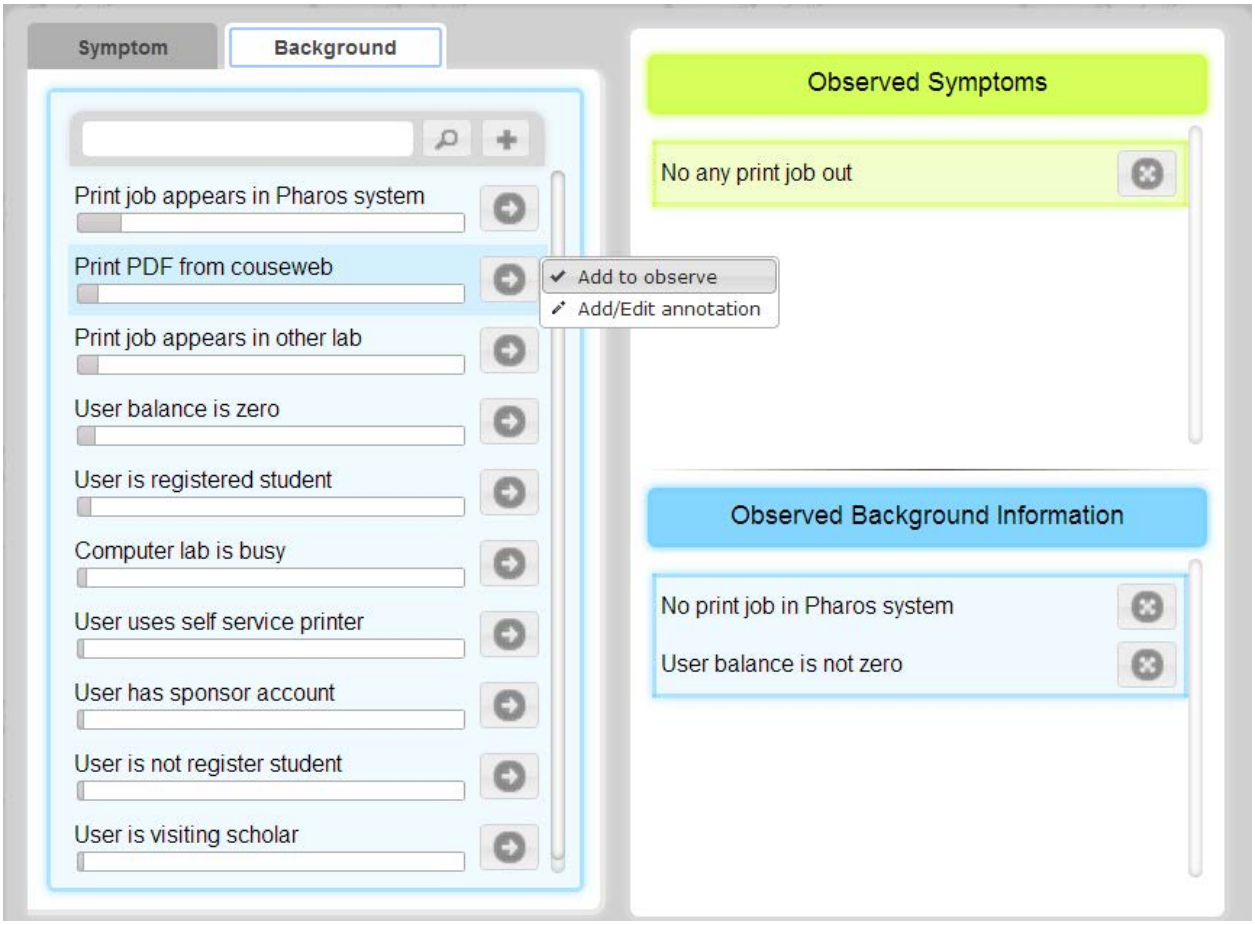

Figure 9: A user selects background information in the *Background* panel.

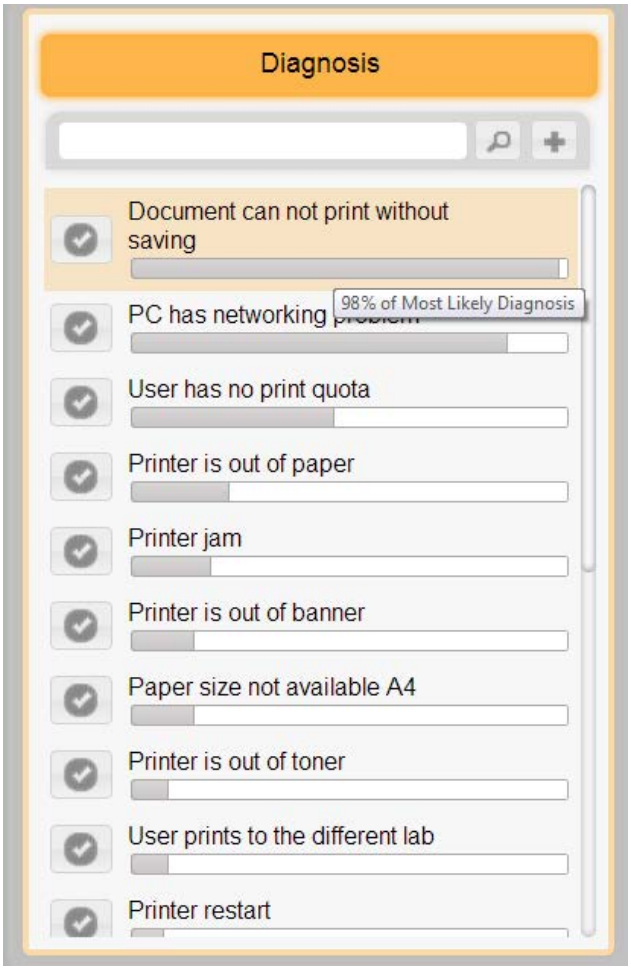

Figure 10: The *Diagnosis* panel.

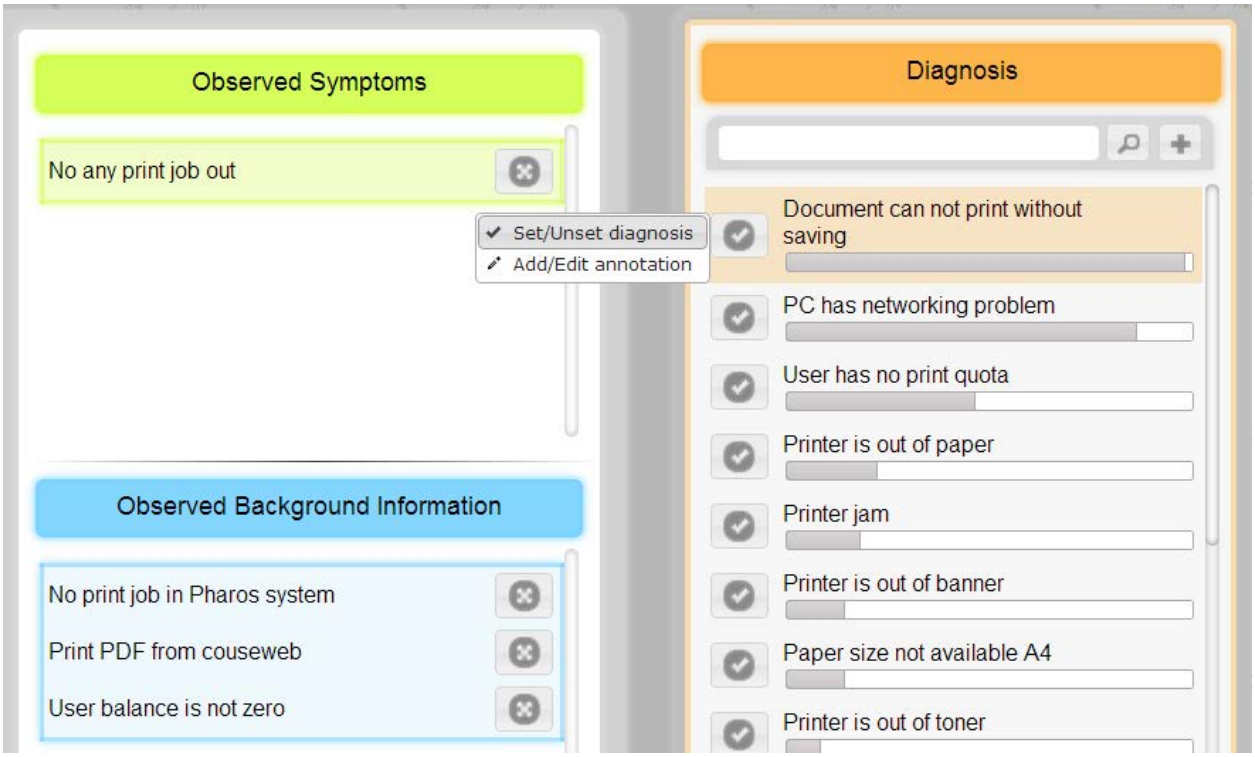

Figure 11: A user selects the diagnosis in the *Diagnosis* panel.

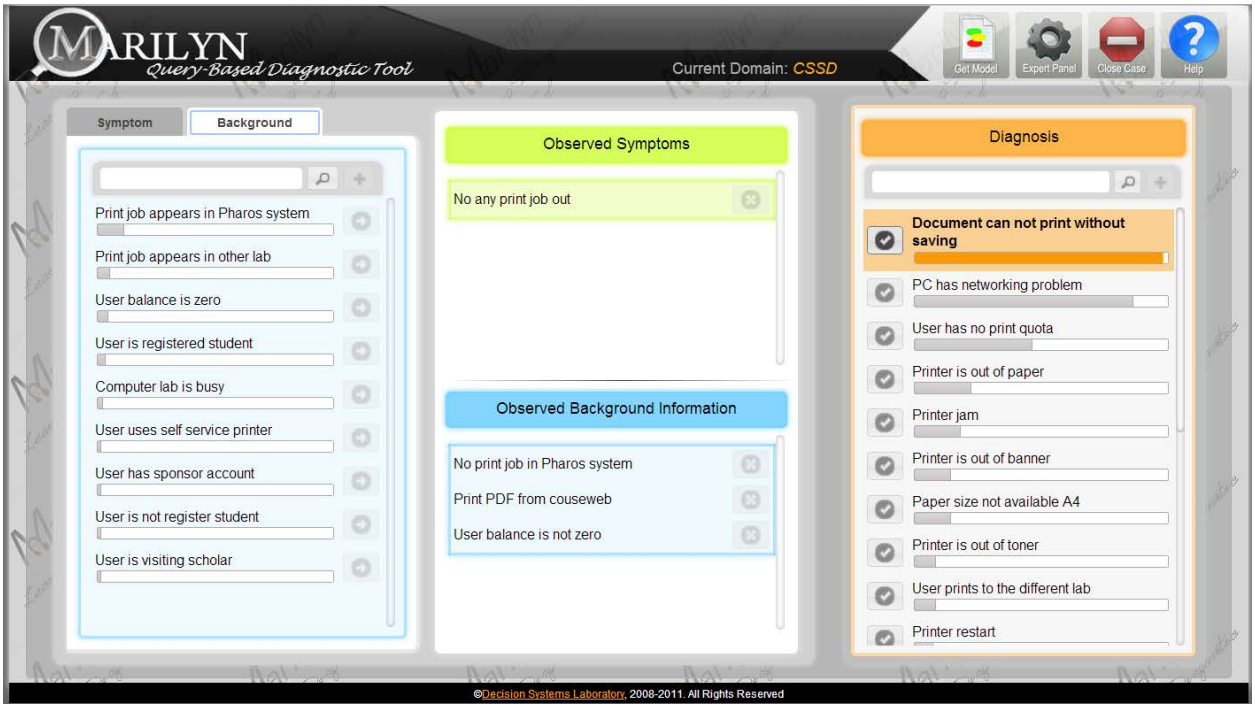

Figure 12: An example of a completed diagnosis session for computer lab domain.

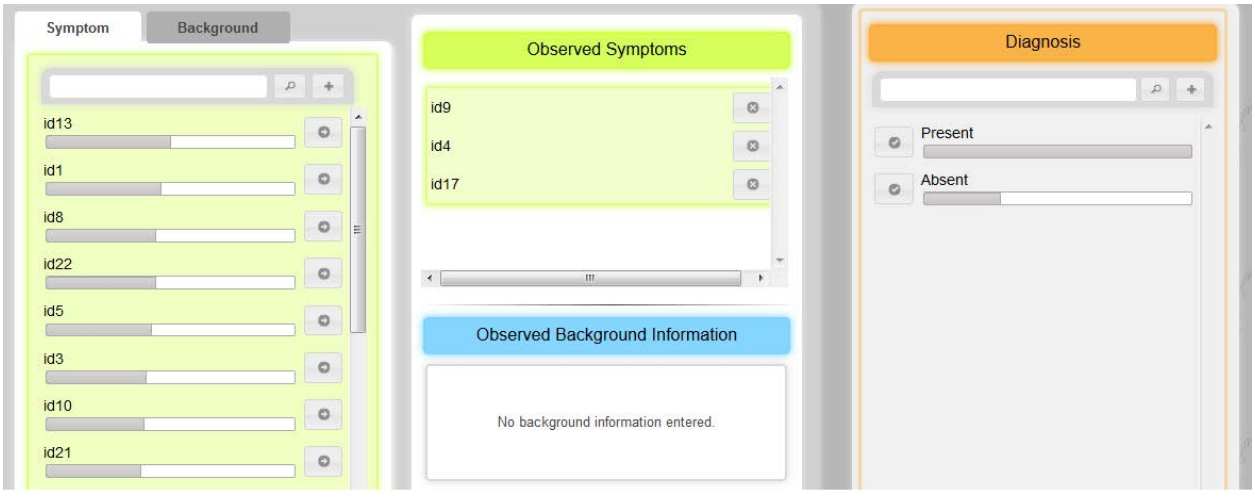

Figure 13: A user selects three observations in the *Symptom* panel for SPECT Heart domain.

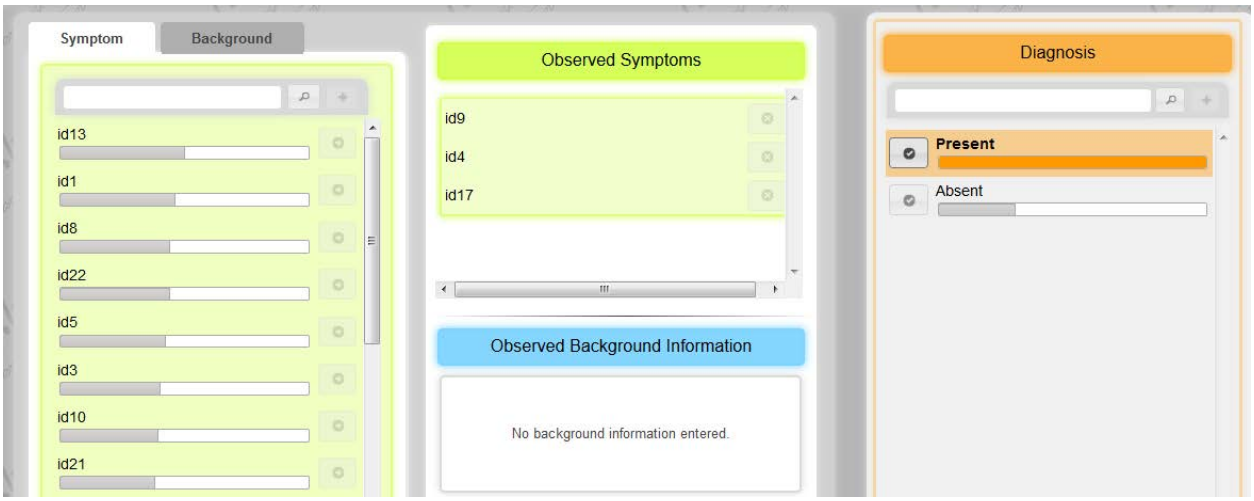

Figure 14: An example of a completed diagnosis session for SPECT Heart domain.

4.0 PARAMETER LEARNING IN QUERY-BASED DIAGNOSTICS

An increasing number of domains involve continuous collection of massive amounts of data. World Wide Web-based systems, for example, often generate records for every user transaction. Real-time monitoring systems obtain sensor readings in fraction of a second increments. A corporate call center may deal with hundreds or even thousands of new cases daily. There exist systems that specialize in continuous data streams and that operate in real-time, e.g., [Tucker et al., 2003, Olesen et al., 1992, Ratnapinda and Druzdzel, 2011]. They all need to learn from the incoming massive amounts of data and systematically update whatever they know about the system that they are monitoring.

When building practical query-based diagnostic systems for these types of systems, the questions arise on how we should process these continuous streams of data. This chapter presents an empirical comparison of Bayesian network parameter learning algorithms for continuous data streams.

4.1 LEARNING CONTINUOUS DATA STREAMS IN BAYESIAN NETWORKS

There are two fundamental approaches to processing continuous data streams, which we will call *batch learning* and *incremental learning*. In the batch learning approach, we repeatedly add new records to the accumulated data and learn anew from the entire data set. When the number of data records becomes very large, this approach may be computationally prohibitive. In addition, it requires storing and efficiently retrieving the entire data set, which may not be feasible. In the incremental learning approach, we assume that the model

learned in the previous step summarizes all the data collected up to that step and use the newly acquired data to refine the model. Incremental learning approach can be divided into two types: *incremental batch learning* and *online learning*. The incremental batch learning or mini-batch learning updates the model by processing the incoming data in chunks, i.e., groups of records. The online learning updates the model by processing records one at the time as they arrive.

The most flexible algorithm for learning Bayesian network parameters is the EM (Expectation Maximization) algorithm [Dempster et al., 1977, Lauritzen, 1995]. While there are several variants of the EM algorithm, two are most notable: the basic EM algorithm [Dempster et al., 1977] and the *online EM* algorithm [Sato and Ishii, 2000, Liang and Klein, 2009, Cappe, 2010].

The most common mode of operation of the basic EM algorithm is *batch learning*, i.e., learning from an entire data set. The basic EM algorithm can be also applied to incremental batch learning, in which case the existing set of parameters, learned previously from a database of cases, is assigned a level of reliability, expressed by a number called the *equivalent sample size*. Equivalent sample size expresses the number of data records that have been used to learn the existing parameters. While updating the existing parameters, the EM algorithm weights the new cases against the existing parameters according to the relative sizes of the data sets. The computational complexity of the incremental batch learning depends primarily on the size of the set of additional records, i.e., the mini-batch. The *online EM* algorithm is a modification of the basic EM algorithm that allows for processing new data into the existing model one record at a time. Its complexity at each time step, both in terms of computation time and memory use, is thus minimal. I should state clearly here that it is based on different principles than incremental EM, so the two algorithms are not equivalent when the increment is equal to one record (Figures 15 and 16).

The question that I pose in this chapter is which of the three approaches is best in practice when learning Bayesian network parameters from continuous data streams. I focus on the impact of choice of each of the learning schemes on (1) computational complexity of learning (speed), (2) accuracy of the learned parameters, and (3) the model's ultimate accuracy. I pose the third question in the context of classification tasks, which is a common application

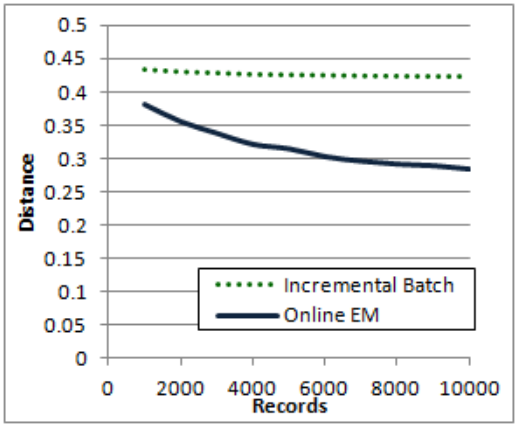

Figure 15: A plot of the average Hellinger distance as a function of the number of records comparing the incremental EM and the Online EM algorithm when processing one record at the time (Letter data set).

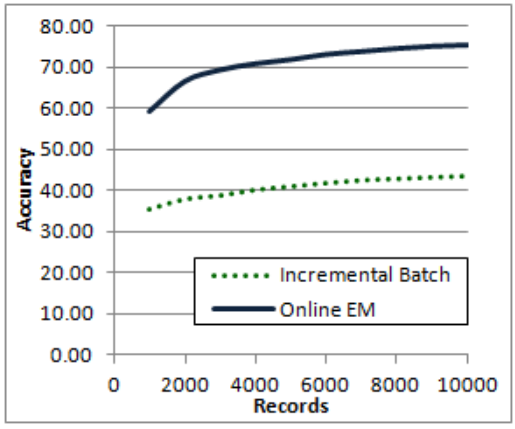

Figure 16: A plot of the classification accuracy as a function of the number of records comparing the incremental EM and the Online EM algorithm when processing one record at the time (Letter data set).

of Bayesian networks. While there exists literature that is related to this question, no comprehensive comparison has been made so far in the context of Bayesian networks. Some papers focus on the comparison of batch learning to incremental learning, e.g., [Carbonara and Borrowman, 1998, Wilson and Martinez, 2003]. They agree on the obvious truth that the online learning is computationally more efficient than batch learning and show experimentally that it also achieves accuracy that is similar to that of the batch learning. Cappe [2010], who compares batch EM to online EM, suggests that the decision to select between the two algorithms depends on the size of the data set. His experiments indicate that when the size of the data is smaller than 1,000 records, EM is preferred over online EM. Holmes et al. [2004] study how mini-batch size affects the performance of incremental learning in terms of classification accuracy and speed. They demonstrate that larger chunk sizes lead to higher classification accuracy.

4.2 AN EMPIRICAL COMPARISON OF BAYESIAN NETWORK PARAMETER LEARNING ALGORITHMS FOR CONTINUOUS DATA STREAMS

I selected seven data sets from the UCI Machine Learning Repository [Frank and Asuncion, 2010] in order to create gold standard Bayesian network models. I subsequently used these models to generate large data sets (each containing 1,000,000 records) to simulate continuous data streams in my experiments. I re-learned Bayesian network parameters from these data streams using (1) batch learning, (2) incremental batch learning, and (3) online learning.

I implemented the EM and the online EM algorithms in C++. I performed my tests on a Windows 7 computer with 8 GB of memory, and an Intel Core i5-3317U processor running at 1.70 GHz.

4.2.1 The data

I selected seven data sets from the UCI Machine Learning Repository: Adult, Australian Credit, Bank Marketing, Chess (King-Rook vs. King-Pawn), Letter, Mushroom, and Nursery, using the following selection criteria:

- The data include a known class variable so that I could test the accuracy of the learned models on a real problem.
- The data contain a reasonably large number of records. I used the EM algorithm for learning parameters in the gold standard models. The EM algorithm learns parameters more accurately from large data sets and this increased the quality of my initial models. In addition, because I check the accuracy of the models on the original data, the larger the data set, the more reliable my results.
- The majority of the attribute types should be discrete in order to reduce the need for discretization, which would be a confounding factor in my experiments.
- The data do not contain too many missing values (not more than 1/3 of the data set). Missing values require special treatment in structure learning algorithms, which would be an additional confounding factor in my experiments.

- The selected data sets have a wide range in the number of attributes (8–36), so that I obtain models of different size for testing.

I decided to use real rather than synthetic data because I wanted my experiments to be as close as possible to real world applications. I listed all selected data sets and summarized their properties in Table 13.

Adult, Australian Credit, Bank Marketing, Chess (King-Rook vs. King-Pawn), Letter, Mushroom, and Nursery,

The Adult data set predicts a person incomes whether a person makes over 50K a year. There are 14 attributes of demographic data. The Australian Credit data set classifies as positive (+) or negative (-) on 14 anonymized variables. The Bank Marketing data set classification goal is to predict if the client will subscribe a term deposit. There are 16 attributes of demographic data. The Chess (King-Rook vs. King-Pawn) data set classifies as win or no win. The 36 attributes describe a board-descriptions for the chess endgame. The Letter data set objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. There are 16 integer attributes which values range from 0 through 15. The Mushroom data set classifies mushrooms as edible or poisonous based on 22 attributes of the shape and texture. The Nursery data set objective is to rank applications for nursery schools into five classes:not recommend, recommend, very recommend, priority, and special priority. There are 8 attributes of demographic data.

4.2.2 Selecting learning rate for online EM algorithm

I use the generalized version of the online EM algorithm, proposed by Cappe [2010], in the following way. Given S_0, θ_0, and a sequence of step sizes $(\gamma_n)_{n\geq 1}$, do, for $n \geq 1$.

$$\textbf{E-step:} \quad S_n = (1-\gamma_n)S_{n-1} + \gamma_n E_{\theta_{n-1}}\left[s(X_n, Y_n)\middle| Y_n\right] \tag{4.1}$$

$$\textbf{M-step:} \quad \theta_n = \bar{\theta}(S_n)\ . \tag{4.2}$$

Table 2: Data sets used in my experiments. #I denotes the number of records, #A denotes the number of attributes, #CV denotes the number of class variables, #FP denotes the number of free parameters, and MV indicates presence of missing values.

Data set	**#I**	**#A**	**#CV**	**#FP**	**MV**
Adult	48842	14	2	3762	Yes
Australian Credit	690	14	2	388	No
Bank Marketing	45211	16	2	1180	No
Chess	3196	36	2	972	No
Letter	20000	16	26	25279	No
Mushroom	8124	22	2	4786	Yes
Nursery	12960	8	5	645	No

I have to select a step size γ_n for the online EM. Cappe [2010] suggests that the most robust value of α is 0.6. To find the optimal values for α, I measure the accuracy of each learning rate α by calculating the Hellinger distance between the parameters in the learned models and the original parameters in the gold standard models. I show a typical plot of Hellinger distance as a function of the number of records for the Adult data set (Figure 17). Hellinger distance for online EM increases with the value of the α and, therefore, I select the lowest α values of 0.501 for my experiments.

4.2.3 Experiments

To learn the gold standard Bayesian networks, I applied the standard Bayesian learning algorithm proposed by Cooper and Herskovits [1992]. This algorithm does not handle missing values and continuous variables. I first discretized continuous attributes using equal frequency discretization with 5 intervals, removed all records with missing values, and used the Bayesian learning algorithm to learn the model structure. Subsequently, I used the entire data sets (i.e., including the records with missing values) to learn the models' numerical parameters. The models constructed in this way were my gold standard models.

I used the gold standard models to generate data sets of 1,000,000 records each (no missing values). I used these records to simulate data streams in my experiments. I used

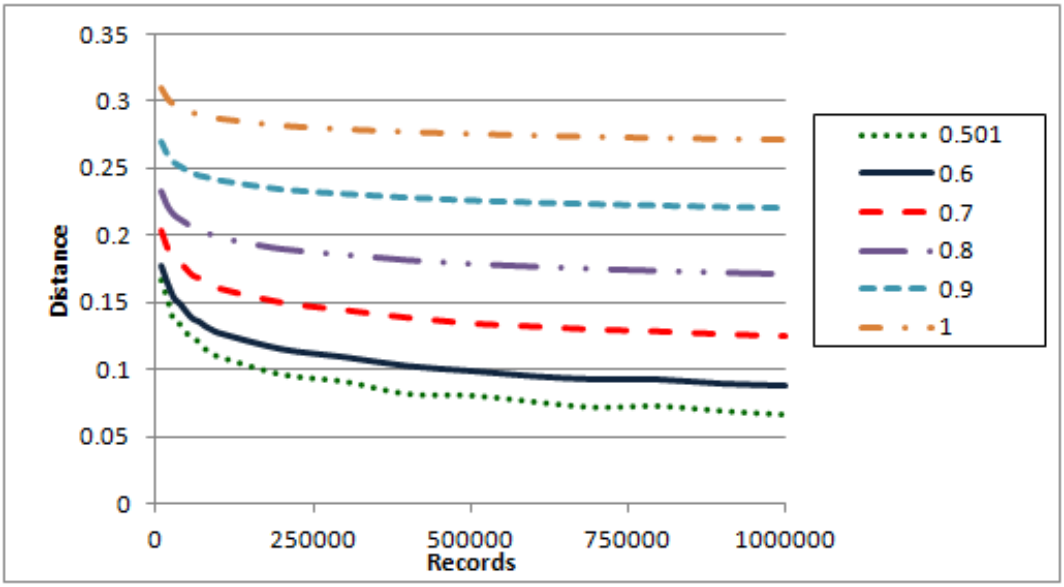

Figure 17: Average Hellinger distance as a function of the number of records in the data stream for the Adult data set using different learning rate α

the structures of the gold standard models as skeletal models for learning parameters.

In my experiments, I compared the following three algorithms for parameter learning from continuous data streams:

1. The basic EM algorithm applied at each step to the entire data set. I referred to it as the *batch learning approach*. I started running the batch learning procedure at 10,000 records. Then I invoked the batch learning algorithm after every 10,000 records. I used uniform priors and equivalent sample size of 1 for all runs.

2. *The batch incremental learning approach* means that the learning happens after each k new instances and these new records serve to refine the existing model. In my experiments, I set $k = 10,000$. In the first step (the first 10,000 records), I used uniform priors and equivalent sample size of 1. In each subsequent step, I used the existing model as the base model and ran the EM algorithm with the equivalent sample size parameter equal to the number of data records that had been used to learn the existing model. For example, when processing the records between 30,000 and 40,000, I set the equivalent sample size

to 30,000 (the existing model had been learned from the previous 30,000 records). The basic EM thus combines the new parameters with the existing parameters, according to the relative size of the data sets.

3. *The online learning approach* updates the network parameters each time a new record becomes available. I ran the online EM algorithm until it reached 1,000,000 records. According to Cappe [2010], to get better performance in parameter learning, it is better not to perform the maximization step for the first 20 records. Following his idea, I started the maximization step only after the first 20 records had been processed. I used the learning rate $\alpha = 0.501$ for all runs.

I measured the CPU time consumed by each of the algorithms. I measured the accuracy of parameters in the learned models by comparing them to the parameters in the gold standard models. I tested the classification accuracy of the learned Bayesian network models on the original data sets from the UCI Machine Learning Repository.

4.2.4 Results

4.2.4.1 Speed Figure 18 shows that the difference between incremental learning and batch learning algorithms grows larger as the number of records increases. The larger the number of attributes in a Bayesian networks, the larger the savings in computation time. I report the run time of each algorithm processing the last 10,000 records (i.e., from 990,000 to 1,000,000) in Table 3. The batch incremental learning approach and the online learning approach use constant amount of time for each run and spend less computation time than the batch learning approach. Times on the order of a second are practically negligible in a system employed in practice — data usually come at a lower speed.

4.2.4.2 Parameter accuracy I show the final average Hellinger distance for all data sets in Table 4. Because the shape of the distance curves as a function of the number of records seems quite regular, we also added a second number that indicates the slope of the curve at the last 100,000 records. When equal to -0.01, for example, it leads to an absolute reduction of Hellinger distance of 0.01 per 100,000 records.

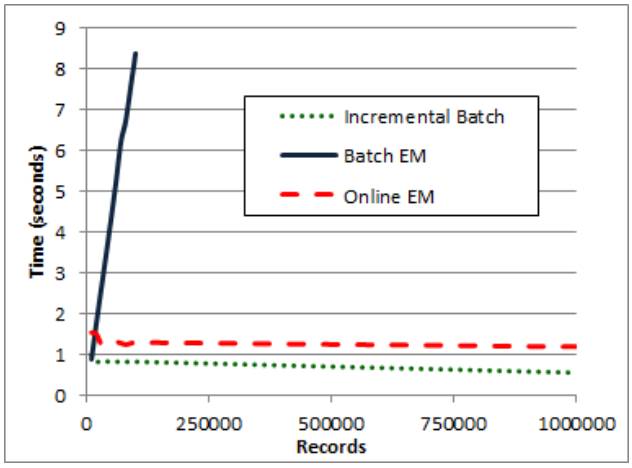

Figure 18: Adult data set computation time. Time taken by the batch algorithm is linear in the number of records, I omit its run time after 100,000 records in order to show the details for the incremental batch EM and the online EM algorithms

Table 3: Computation time required to process the last 10,000 records in a data set of 1,000,000 records shown in seconds. Please note that from records 990,000th to 1,000,000th, incremental batch learning runs only one time; the the online learning algorithm runs 10,000 times.

Data set	**Incremental Batch**	**Batch**	**Online**
Adult	0.55	84.22	1.18
Australian Credit	0.55	78.66	0.66
Bank Marketing	0.73	112.41	0.94
Chess	1.93	276.55	1.89
Letter	1.03	157.72	4.99
Mushroom	1.61	165.47	1.86
Nursery	0.36	51.72	0.48

In six of the seven cases, the batch learning approach resulted in the smallest Hellinger distance, i.e., the highest accuracy of retrieving the original parameters from data. The online learning performed best only on the Mushroom data set. This result differs somewhat from the results obtained by Liang and Klein [2009], who observed that online EM is often

Table 4: Final Hellinger distance

Data set	Incremental Batch	Batch	Online
Adult	0.07642	**0.01786**	0.06608
	-0.00006	-0.00073	-0.00268
Australian Credit	0.02750	**0.00527**	0.01660
	-0.00012	-0.00018	-0.00310
Bank Marketing	0.05668	**0.00710**	0.03072
	-0.00012	-0.00030	-0.00048
Chess	0.02984	**0.00818**	0.02116
	-0.00003	-0.00053	-0.00065
Letter	0.13040	**0.04762**	0.09783
	-0.00009	-0.00043	-0.00257
Mushroom	0.09482	0.04188	**0.02109**
	-0.00005	-0.00108	-0.00028
Nursery	0.06070	**0.01341**	0.04219
	-0.00002	+0.00001	-0.00313

more accurate than batch EM on unsupervised tasks. The negative slopes of the curves (the second number in the table) indicate that each of the algorithms improves its accuracy over time. However, this improvement is typically smaller for the incremental batch algorithm.

I show plots of of Hellinger distance as a function of the number of records for the all data sets Figure 19, 20, 21, 22, 23, 24, and 25). Hellinger distance for both batch EM and online EM decreases with the number of records. Incremental batch learning typically seems to reach a plateau beyond which it hardly improves the accuracy of parameters.

4.2.4.3 Classification accuracy In testing the classification accuracy of the learned models on the original UCI Machine Learning Repository data sets, I used the simplest possible criterion, which is that the model guesses the most likely class to be the correct class for each record. Table 5 shows the final accuracy for all data sets. While the difference in accuracy seems minimal, the batch learning approach resulted in the best classification accuracy on all data except for the Nursery data set. We show plots of models' classification accuracy as a function of the number of records in Figures 26, 27, 28, 29, 30, 31, and 32.

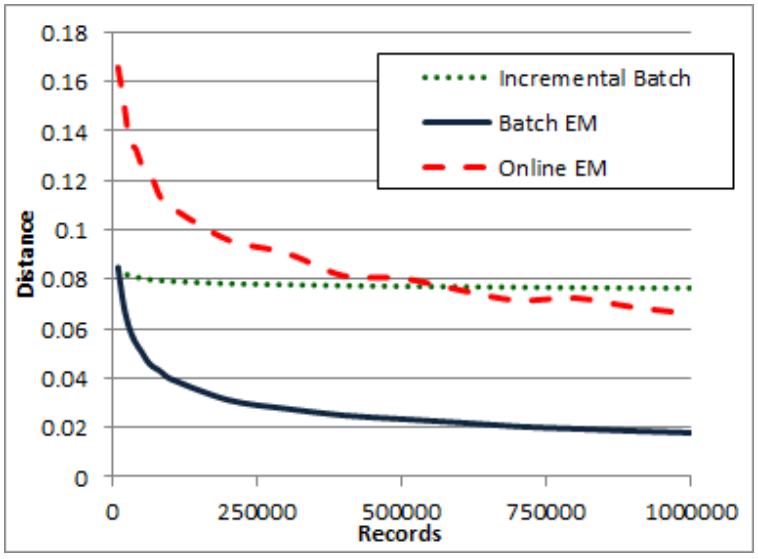

Figure 19: Average Hellinger distance as a function of the number of records in the data stream for the Adult data set

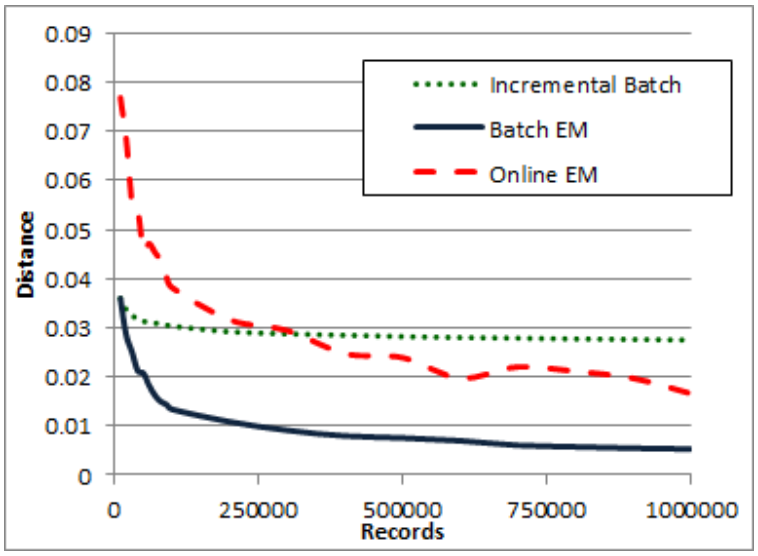

Figure 20: Average Hellinger distance as a function of the number of records in the data stream for the Australian Credit data set

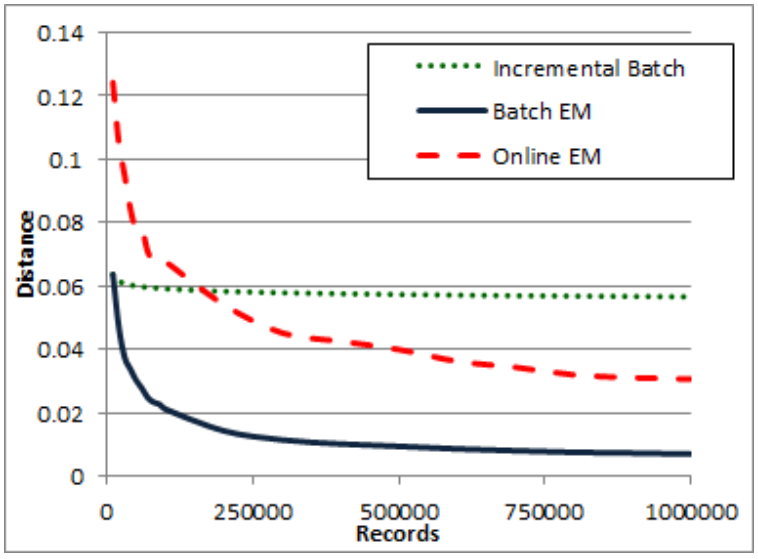

Figure 21: Average Hellinger distance as a function of the number of records in the data stream for the Bank Marketing data set

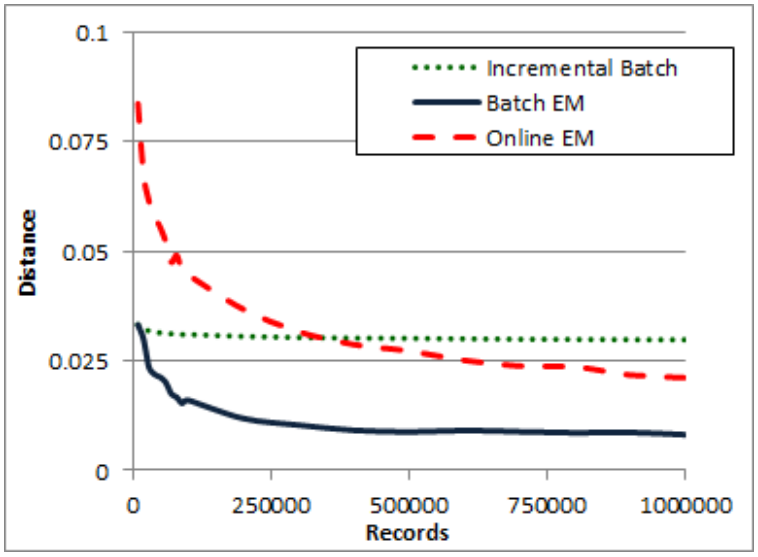

Figure 22: Average Hellinger distance as a function of the number of records in the data stream for the Chess (King-Rook vs. King-Pawn) data set

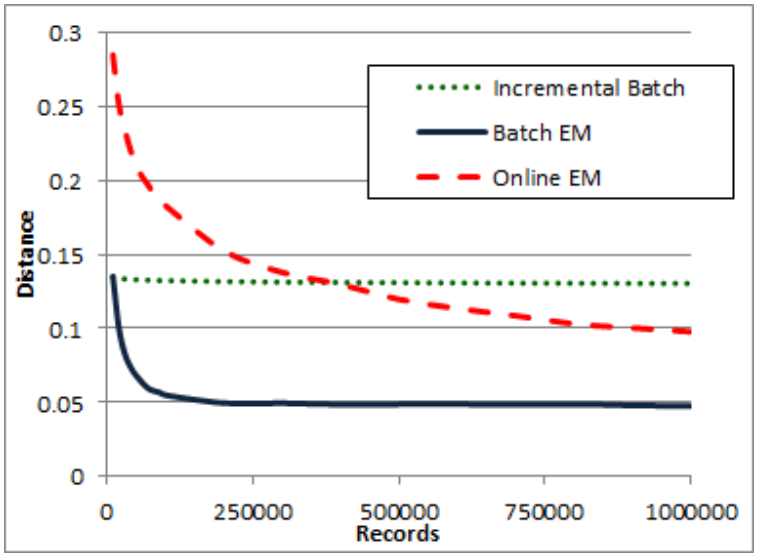

Figure 23: Average Hellinger distance as a function of the number of records in the data stream for the Letter data set

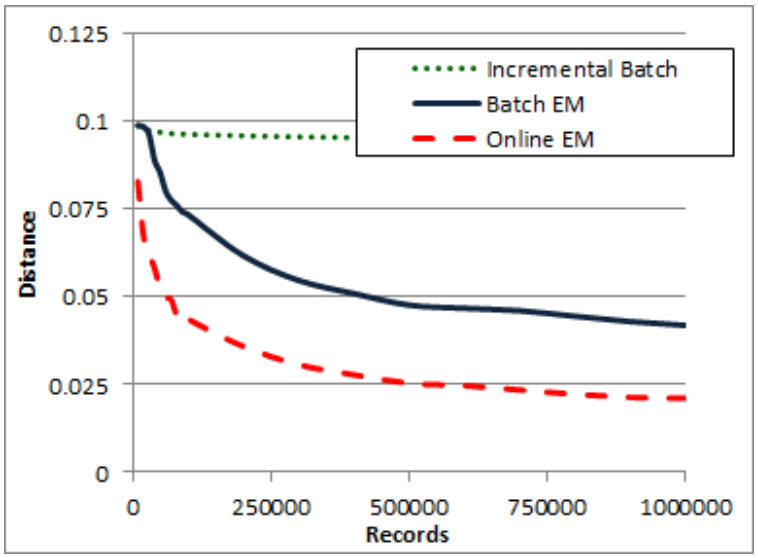

Figure 24: Average Hellinger distance as a function of the number of records in the data stream for the Mushroom data set

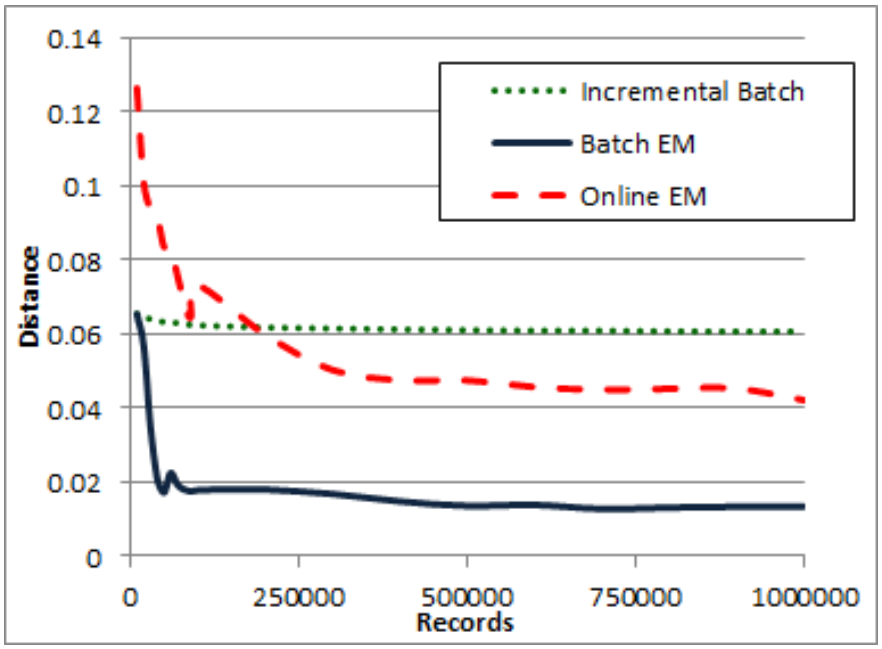

Figure 25: Average Hellinger distance as a function of the number of records in the data stream for the Nursery data set

Table 5: Final classification accuracy

Data set	Inc. Batch	Batch	Online	Gold
Adult	84.21%	**84.22%**	84.20%	84.23%
Australian Credit	**85.51%**	**85.51%**	**85.51%**	85.51%
Bank Marketing	89.48%	**89.54%**	89.52%	89.55%
Chess	**94.21%**	**94.21%**	**94.21%**	94.21%
Letter	83.97%	**87.57%**	86.01%	87.61%
Mushroom	99.85%	**99.90%**	**99.90%**	99.90%
Nursery	**94.74%**	94.64%	94.63%	94.65%

4.2.4.4 Missing Values In order to check whether missing data have impact on the results, I used the gold standard models again to generate data sets of 1,000,000 records with 10 percent and 30 percent values missing at random. I repeated my experiments on all data sets. I obtained qualitatively similar results with a slight overall increase of the Hellinger distance and decrease of classification accuracy (Tables 6 and 9).

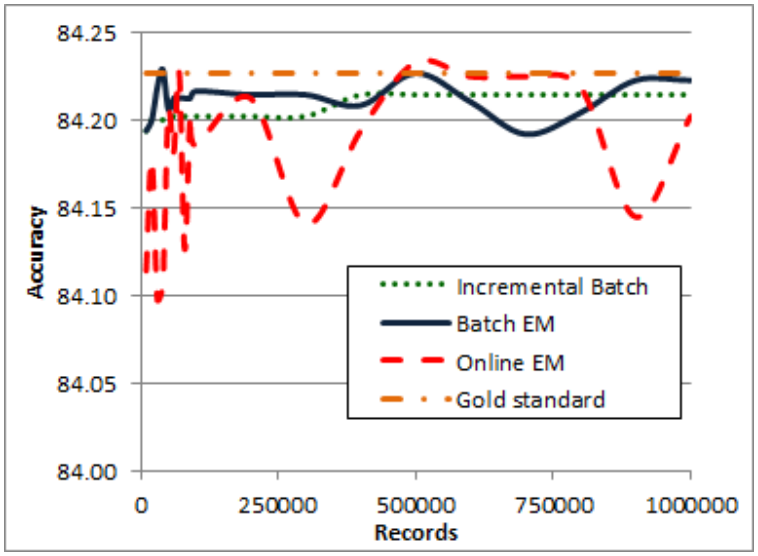

Figure 26: Classification accuracy as a function of the number of records for the Adult data set

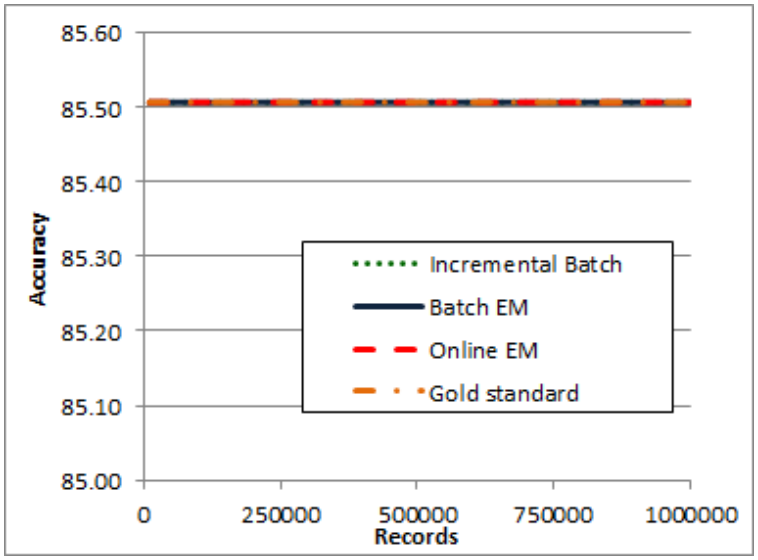

Figure 27: Classification accuracy as a function of the number of records for the Australian Credit data set

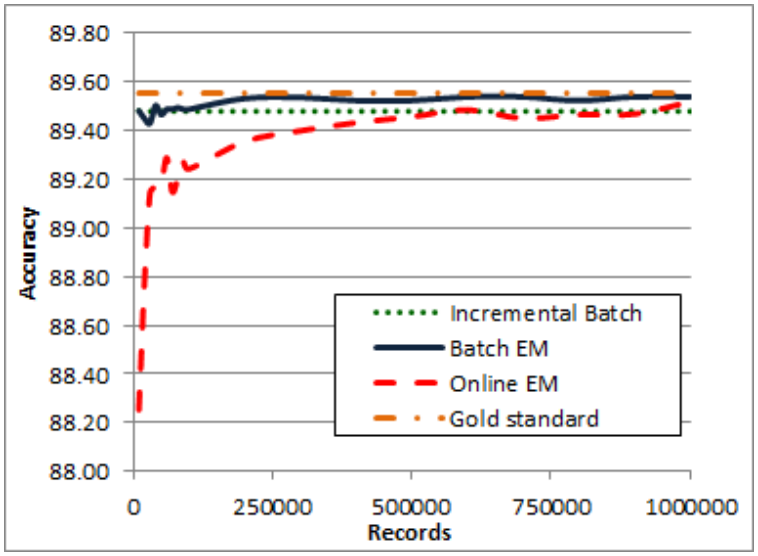

Figure 28: Classification accuracy as a function of the number of records for the Bank Marketing data set

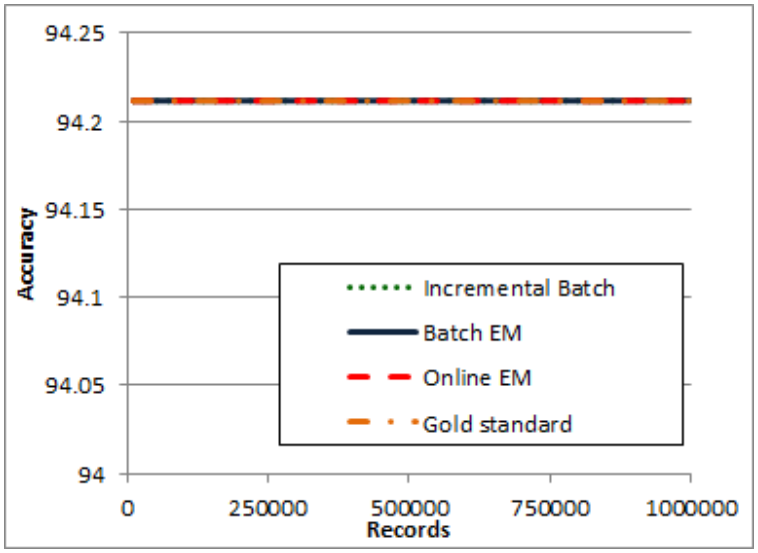

Figure 29: Classification accuracy as a function of the number of records for the Chess (King-Rook vs. King-Pawn) data set

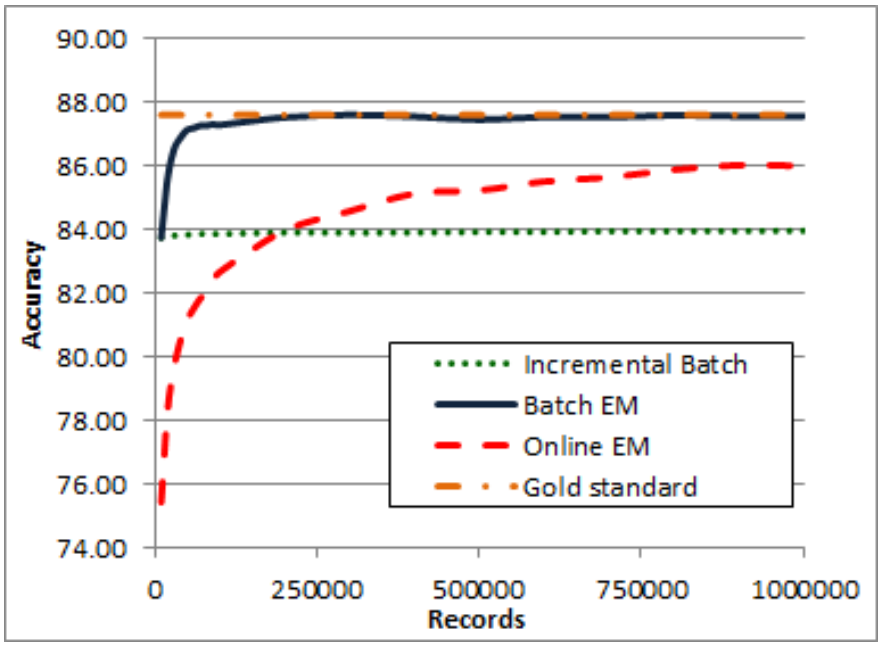

Figure 30: Classification accuracy as a function of the number of records for the Letter data set

I performed the Wilcoxon test [Demšar, 2006] on the data in Tables 6 to compare the three learning approaches on the resulting Hellinger distance. Tables 7 and 8 show the results of this test.

There are significant differences among the approaches in the two-tailed test. One-tailed test results show that the batch learning significantly more accurate than the other two approaches. The online learning is significantly better than the incremental batch approach.

I performed the Wilcoxon test among the three learning approaches for the classification accuracy in Tables 9. Table 10 and Table 11 show the results of this test.

Two-tailed test shows significant differences between the incremental batch learning and the batch learning and between the incremental batch learning and the online learning. There is no significant difference between the batch learning and the online learning. The one-tail test results show that the batch learning and the online learning are significantly more accurate than the incremental batch learning.

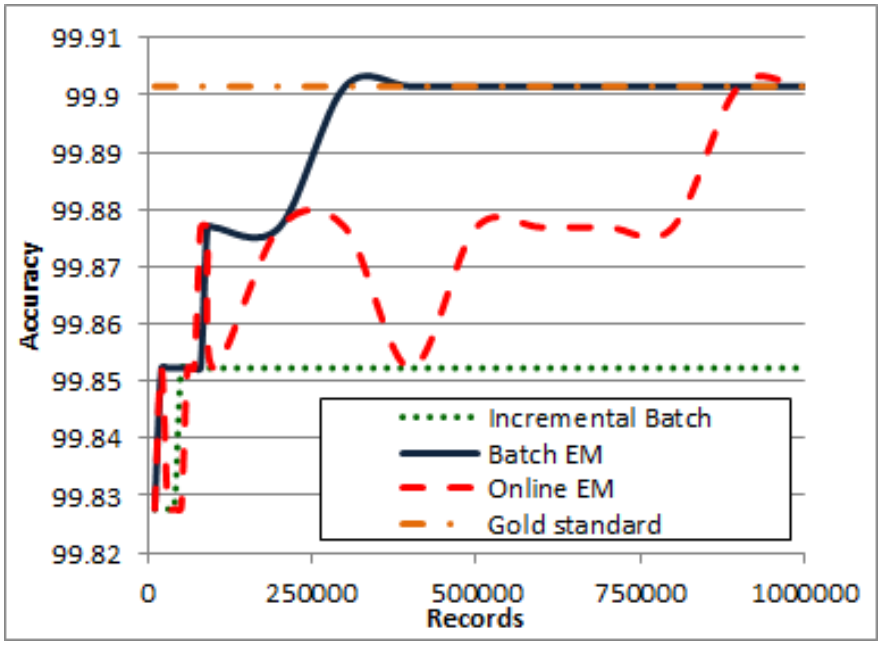

Figure 31: Classification accuracy as a function of the number of records for the Mushroom data set

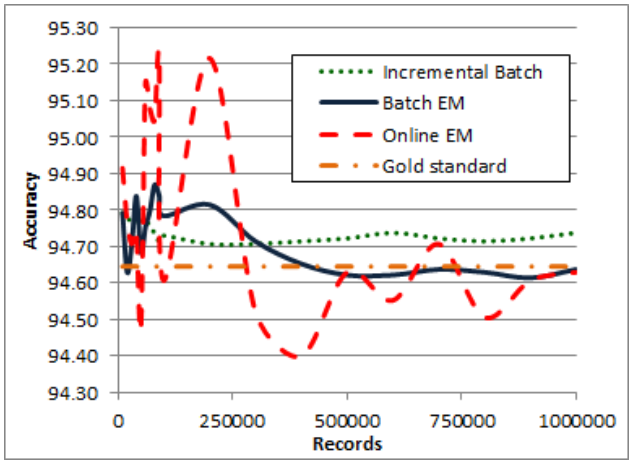

Figure 32: Classification accuracy as a function of the number of records for the Nursery data set

Table 6: Final Hellinger distance with values missing at random. Baseline denotes no missing values, 10% denotes 10 percents values missing at random, and 30% denotes 30 percents values missing at random.

Data set	**Incr. Batch**			**Batch**			**Online**		
	Baseline	10%	30%	Baseline	10%	30%	Baseline	10%	30%
Adult	0.0764	0.0844	0.1033	0.0179	0.0213	0.0267	0.0661	0.0722	0.0839
A. Credit	0.0275	0.0360	0.0367	0.0053	0.0052	0.0090	0.0166	0.0182	0.0231
Bank M.	0.0567	0.0678	0.0850	0.0071	0.0080	0.0194	0.0307	0.0332	0.0422
Chess	0.0298	0.0361	0.0455	0.0082	0.0084	0.0180	0.0212	0.0225	0.0342
Letter	0.1304	0.1461	0.1776	0.0476	0.0485	0.0622	0.0978	0.1078	0.1364
Mushroom	0.0948	0.1037	0.1043	0.0419	0.0432	0.0615	0.0211	0.0243	0.0338
Nursery	0.0607	0.0578	0.0621	0.0134	0.0111	0.0182	0.0422	0.0438	0.0474

Table 7: Wilcoxon's two-tailed test among the incremental batch learning, the batch learning, and the online learning in terms of the Hellinger distance.

	Batch	**Online**
Incr. Batch	**0.000128067**	**2.861023e-06**
Batch	N/A	**2.221482e-03**

Table 8: Wilcoxon's one-tailed test among the incremental batch learning, the batch learning, and the online learning in terms of the Hellinger distance.

	Incr. Batch	**Batch**	**Online**
Incr. Batch	-	1	1
Batch	**6.403349e-05**	-	**0.003332223**
Online	**1.430511e-06**	0.9990115	-

Table 9: Final classification accuracy with values missing at random. Baseline denotes no missing values, 10% denotes 10 percents values missing at random, and 30% denotes 30 percents values missing at random.

Data set	**Incr. Batch**			**Batch**			**Online**		
	Baseline	10%	30%	Baseline	10%	30%	Baseline	10%	30%
Adult	84.21%	84.19%	84.00%	84.22%	84.21%	84.22%	84.20%	84.22%	84.20%
A. Credit	85.51%	85.51%	85.51%	85.51%	85.51%	85.51%	85.51%	85.51%	85.51%
Bank M.	89.48%	89.30%	89.27%	89.54%	89.55%	89.52%	89.52%	89.43%	89.46%
Chess	94.21%	94.15%	94.21%	94.21%	94.21%	94.21%	94.21%	94.21%	94.21%
Letter	83.97%	83.46%	81.10%	87.57%	87.53%	87.34%	86.01%	85.79%	84.37%
Mushroom	99.85%	99.85%	99.80%	99.90%	99.90%	99.88%	99.90%	99.90%	99.85%
Nursery	94.74%	94.51%	94.56%	94.64%	94.64%	94.66%	94.65%	94.71%	94.78%

Table 10: Wilcoxon's two-tailed test among the incremental batch learning, the batch learning and the online learning in terms of the classification accuracy.

	Batch	**Online**
Incr. Batch	**0.006811761**	**0.006811761**
Batch	N/A	0.086282340

Table 11: Wilcoxon's one-tailed test among the incremental batch learning, the batch learning and the online learning in terms of the classification accuracy.

	Incr. Batch	**Batch**	**Online**
Incr. Batch	-	1	1
Batch	**0.00340588**	-	0.1294235
Online	**0.00340588**	0.9628983	-

4.3 DISCUSSION

This chapter addresses the problem of learning Bayesian network parameters from continuous data streams. I have described an experiment that focuses on a comparison of three approaches: (1) batch learning, (2) incremental batch learning, and (3) online learning, in terms of the computational efficiency, the resulting accuracy of parameters, and the resulting classification accuracy.

There is no significant difference in terms of parameter accuracy and classification accuracy among the three Bayesian network parameter learning approaches: the batch learning, the incremental batch learning, and the online learning when learning Bayesian network parameters from continuous data streams.

I rejected the null hypothesis because there are significant differences among these approaches in terms of parameter accuracy and classification accuracy.

The results indicate that while batch learning, i.e., the basic EM algorithm, makes the largest computational and storage demands, it also offers the highest resulting parameter accuracy. Online EM requires less computation time and no storage while achieving similar results to incremental EM algorithm in terms of accuracy. Incremental EM shows better accuracy for small data sets.

I advise to use batch learning applied to the entire data set whenever computation time and memory space permit. When the computation becomes too long or the complete data set uses too much storage, switching over to the online algorithm is a safer choice. It seems that both the, batch and the online EM algorithms, make significant improvements in accuracy in the beginning. A possible hybrid strategy is to start with the batch EM algorithm and transform it to the online EM algorithm when its run time becomes prohibitive. We show two typical plots for different strategies (1) Hellinger distance as a function of the number of records in Figure 33 and (2) models' classification accuracy as a function of the number of records in Figure 34. An alternative strategy for all systems in which real-time response is critical, is to use the online EM algorithm during daily operations and the batch EM during maintenance hours. This should ensure that the starting points of the online algorithm for real-time operations is always the best possible.

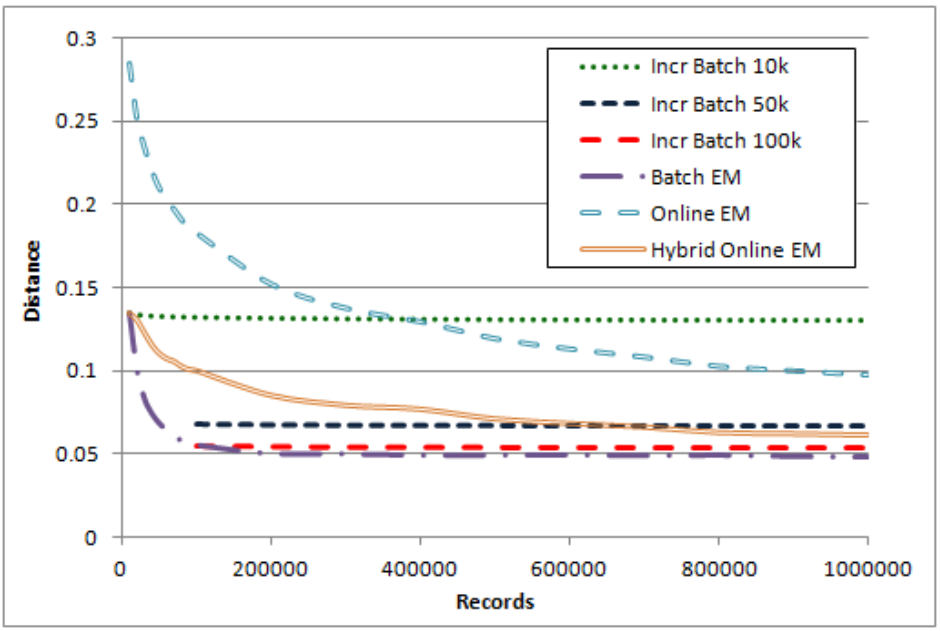

Figure 33: A typical plot of the average Hellinger distance as a function of the number of records in the data stream with different strategies (Letter data set).

I observed that classification accuracy did not change much with refinement of parameters. My experiments have confirmed an earlier finding of Oniśko and Druzdzel [2013] that Bayesian networks are quite insensitive to precision of their numerical parameters.

There is a fundamental question that one needs to ask when processing continuous streams of data: Does the system from which the data originate remain constant over time? In this work, I assumed tentatively that it does and, hence, I learned from all accumulated records. Real systems, however, can evolve over time (e.g., Łupińska-Dubicka and Druzdzel [2012]). In all such cases, it is quite appropriate to assign more recent parameters a higher weight. It may be natural in such cases to discard older records altogether and, hence, avoid the problem of excessive data storage or prohibitive computation. My approach and results hold for all such cases. One might treat the data sets in our experiment as belonging to the sliding windows from which the parameters are learned.

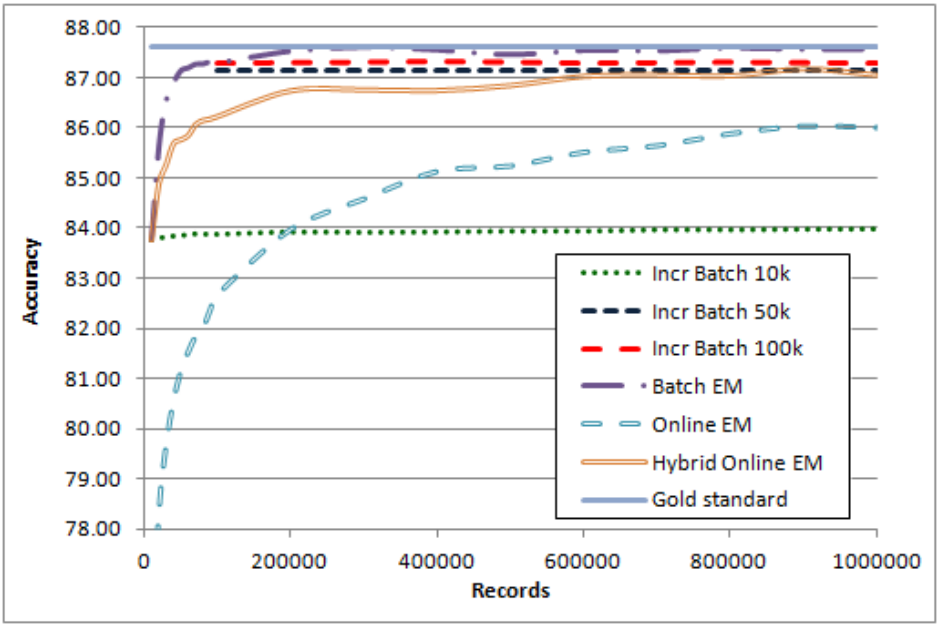

Figure 34: A typical plot of the classification accuracy as a function of the number of records with different strategies (Letter data set).

5.0 DEALING WITH STRUCTURAL COMPLEXITY

Practical models based on Bayesian networks (BNs) reach often the size of hundreds or even thousands of variables. When these are densely connected, both the amount of memory to store a compiled clique tree and the amount of computation necessary to perform belief updating may become prohibitive [Cooper, 1990]. In MARILYN, it is a necessity to control their growth. Otherwise, at a certain point, an uncontrolled model is bound to become intractable.

One way of controlling the growth of a model is to systematically simplify its structure by removing its weakest arcs. There have been two approaches to arc removal in Bayesian networks. The first approach focuses on minimizing the KL-divergence between the joint probability distributions represented by the original and the approximated networks (e.g., [Kjaerulff, 1994, van Engelen, 1997, Choi et al., 2005, Choi and Darwiche, 2006b,a, Renooij, 2010]). The second approach introduces a measure of arc strength and then approximates the model by removing its weakest arcs. Boerlage [1992] defines link strength for arcs connecting binary nodes as the maximum influence that the parent node can exert on the child node. Nicholson and Jitnah [1998] use mutual information as a measure of link strength. They show how inference can be simplified by averaging the conditional probabilities of all parents. Ebert-Uphoff [2007, 2009] proposes a link strength measure called True Average Link Strength (TALS). Similarly to the work of Nicholson and Jitnah [1998], TALS uses mutual information to calculate link strength and uses the average values of the probability of a parent by conditioning on all other parents to calculate mutual information. The scale of TALS is not linear and also not very intuitive. Therefore, as Ebert-Uphoff [2007, 2009] herself states, it does not seem suitable to use as a threshold for arcs removal. Koiter [2006] proposed another measure of strength of influence, resting on the analysis of differences

among the posterior marginal probability distributions of a child node for different states of the parent node. He calculates the difference between distributions using four distance measures: Euclidean, Hellinger, J-divergence and CDF. This strength of influence has been applied in practice for the purpose of model visualization (e.g., [Hsu et al., 2012, Theijssen et al., 2013]), and has been a standard element of the GeNIe software for the last seven years.

The question that I pose in this chapter is how much the accuracy of classification models will suffer as I simplify them by removing their weakest arcs. I pose two questions: (1) how many arcs can we remove with minimal impact on the model's accuracy?, and (2) what are the benefits of removing weakest arcs in terms of the reduction of memory requirements and computation time? I describe an experiment, in which I use several real data sets from the UCI Machine Learning Repository [Frank and Asuncion, 2010] to create Bayesian network models. I subsequently use these models as gold standards to test how removing their weakest arcs impacts their accuracy, memory demands, and inference time. I perform this for each of the four measures proposed by Koiter [2006].

My goal was evaluating the practical impact of model simplification by removing weak arcs on performance measures such as classification accuracy, memory requirements, and computational demands. I selected six data sets from the UCI Machine Learning Repository in order to create gold standard Bayesian network models for the experiments. I decided to use real rather than synthetic data sets because I wanted the experiments to be as close as possible to real world applications.

5.1 THE DATA

I selected six data sets: Chess (King-Rook vs. King-Pawn), Letter, Molecular Biology (Splice-junction Gene Sequences), Mushroom, Nomao and Optical Recognition of Handwritten Digits (ORHD) using the following selection criteria:

- The data include a known class variable so that I could test the accuracy of the models on a real problem.

- The data set contains a reasonably large number of records (more than 1,000). The main reason for this is that I used the EM algorithm [Dempster et al., 1977], which learns parameters more accurately from large data sets. In addition, because I check the accuracy of the models on the original data, the larger the data set, the more reliable the results.

- The selected data sets should not be too small in terms of the number of attributes (16–72), so that I obtain models with reasonably large number of arcs and a challenging total clique tree size.

- The majority of the attribute types should be discrete in order to reduce the need for discretization, which would be a possible confounding factor in the experiments.

- The data set should not contain too many missing values (not more than 1/3 of the data set). Missing values require special treatment in structure learning algorithms, which would be an additional confounding factor in the experiments.

- The probability over the class variable distribution is not too strong biased toward one class. Nomao has the probability of the most likely class equal to 73% while the other data sets varies from 11% to 52%.

Table 13 lists the key properties of the data sets selected for the experiments.

I described the Chess (King-Rook vs. King-Pawn), the Letter and the Mushroom data set description in Chapter 4. The Molecular Biology (Splice-junction Gene Sequences) data set objective is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out) which are neither, EI or IE. There are 60 attributes of the sequence of a character (a, g, t, c). The Nomao data set is a collection of data from multiple sources on the web. The goal is to predict if data refer to the same place (+1) or (-1). There 72 attributes of geographic data. The Optical Recognition of Handwritten Digits (ORHD) data set classified bitmaps of handwritten digits into single digit numbers (0..9). There are 64 integer attributes which values range from 0 through 16.

Table 12: Data sets used in the experiments. #I denotes the number of records, #A denotes the number of attributes, #C denotes the number of classes, #R denotes the number of arcs, #CB denotes total clique tree size of the original BN model, #CT denotes total clique tree size of the original TAN model, and MV indicates presence of missing values.

Data set	**#I**	**#A**	**#C**	**#R**	**#CB**	**#CT**	**MV**
Chess	3196	36	2	100	311KB	284B	No
Letter	20000	16	26	39	664KB	9KB	No
Molecular Biology	3190	60	3	101	10MB	4KB	No
Mushroom	8124	22	2	47	21KB	894B	Yes
Nomao	28575	72	2	279	168MB	2KB	No
ORHD	5620	64	10	82	317KB	150KB	No

5.2 EXPERIMENTS

To learn the gold standard Bayesian networks, I applied the standard Bayesian search-based learning algorithm (BS) proposed by Cooper and Herskovits [1992] and Tree Augmented Naive Bayes algorithm (TAN) proposed by Friedman et al. [1997]. Both algorithms do not handle missing values and continuous variables. I first discretized continuous attributes using equal frequency discretization with 5 intervals, removed all records with missing values, and used each algorithm to learn the model structure. Subsequently, I used the entire data sets (i.e., including the records with missing values) to learn the models' numerical parameters. I present some characteristics of the resulting models in Table 13. No BS models resemble Naive Bayes structure. The BS models have much larger total clique size than TAN models.

I used the models constructed in this way as the gold standard models, which I subsequently simplified by removing weak arcs using four distance measures: Euclidean, Hellinger, J-divergence and CDF. I calculated the strength of influence for each arc in the gold standard network. Figure 35 shows the gold standard model for letter data set after calculating the strength of influence. Subsequently, I removed all arcs that had the strength of influence below a threshold setting (0.1, 0.2, 0.3, ..., 0.9, and 1.0). For example, when I set the threshold at 0.4, I removed all arcs that had the strength of influence less than 0.4 shown in Figure 36.

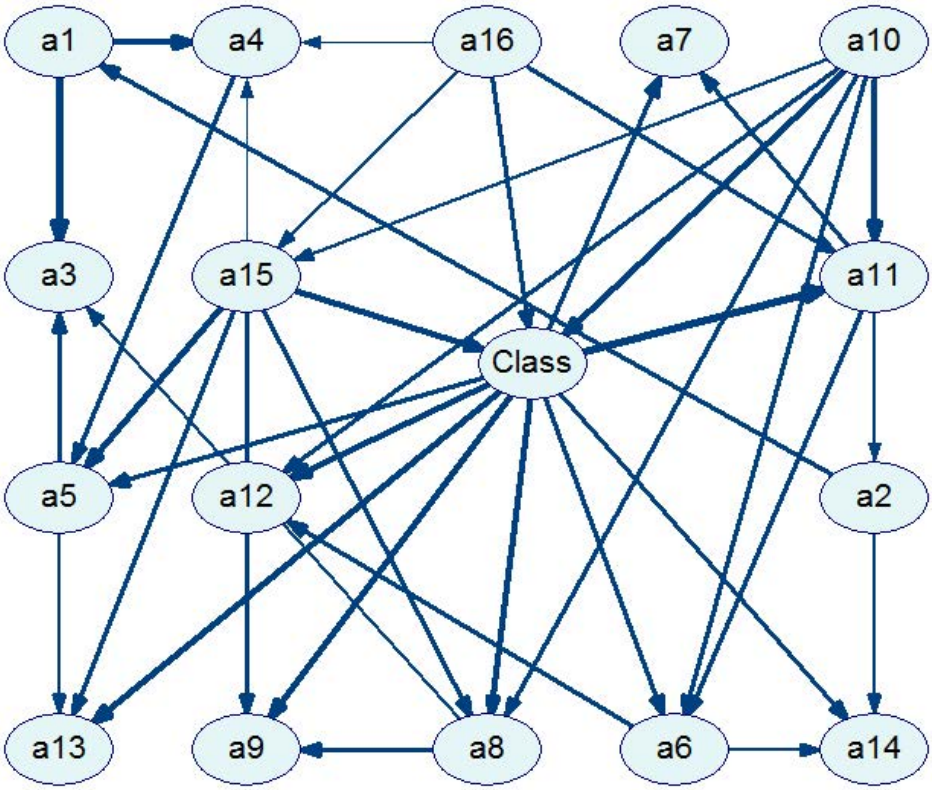

Figure 35: The gold standard model for letter data set. The thickness of the arcs represents the strength of influence between two nodes in Euclidean distance.

I tested the classification accuracy of the simplified Bayesian network models by means of 10-fold cross validation on the original data sets from Table 13. I measured the total clique tree size and the CPU time consumed by performing inference on each of the simplified networks. I performed the tests on a Windows Vista computer with 4 GB of memory, and an Intel Core 2 Quad Q6600 processor running at 2.4 GHz. I implemented all the code in C++.

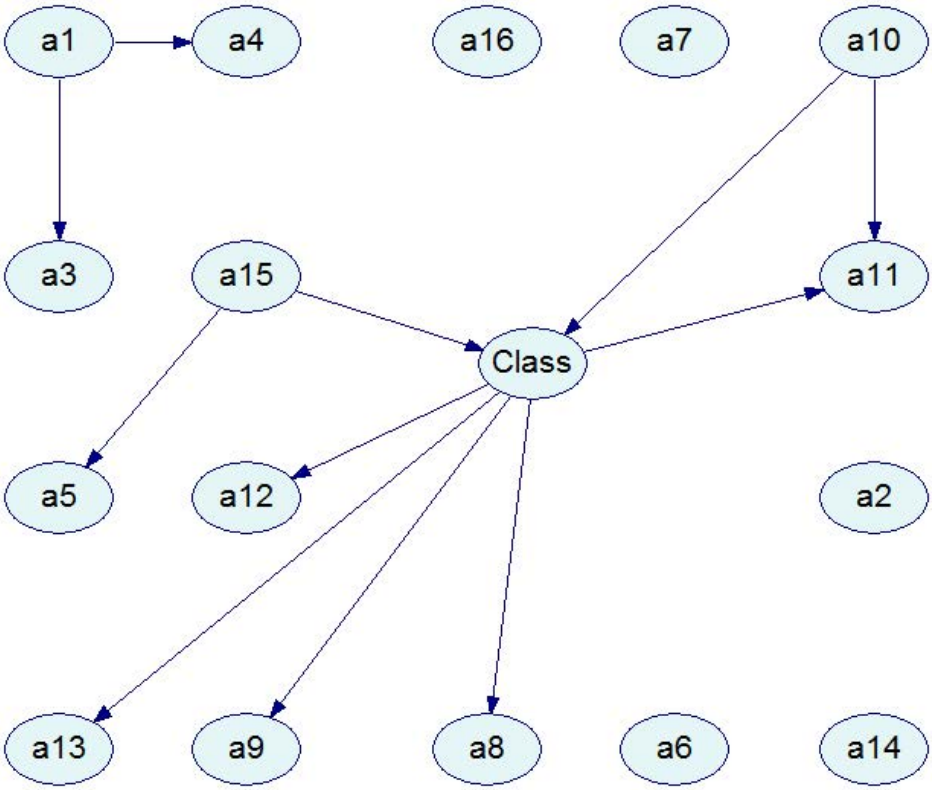

Figure 36: A simplified model for letter data set after removing all arcs that have the strength of influence below 0.4.

5.3 RESULTS

Because the qualitative behavior of the accuracy of the networks is more important in the experiment than the precise numerical results, I present the results of the experiment graphically.

5.3.1 Histograms of the link strengths

Figures 37, 38, 39, 40, 41 and 42 show histogram of the link strength for the six networks studied in the experiments, for each of the four measures of the link strength.

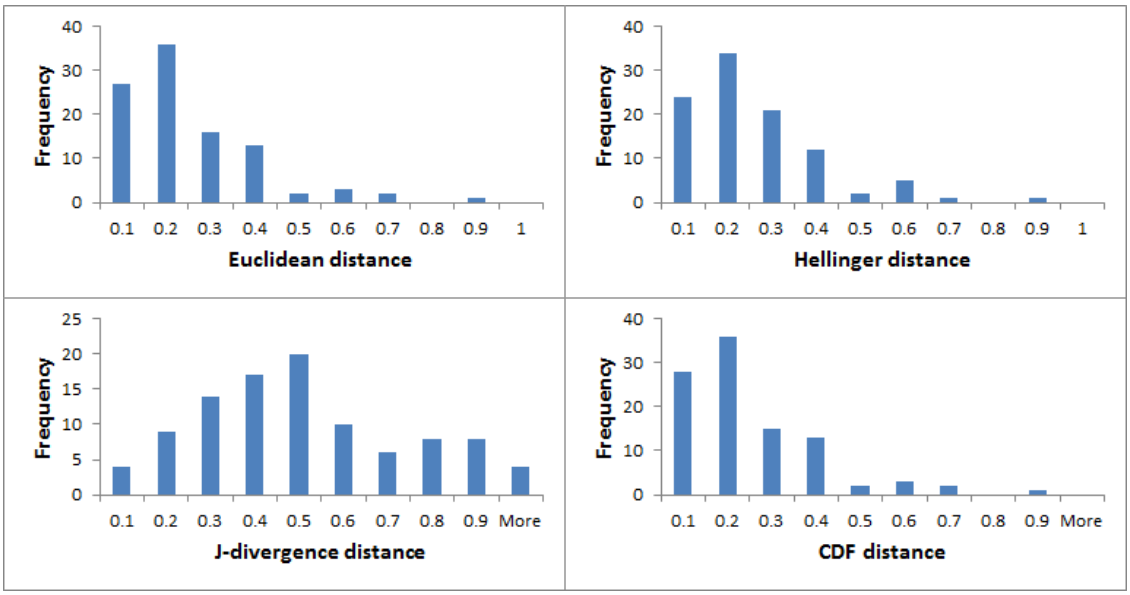

Figure 37: Histogram of the link strength for the Chess network

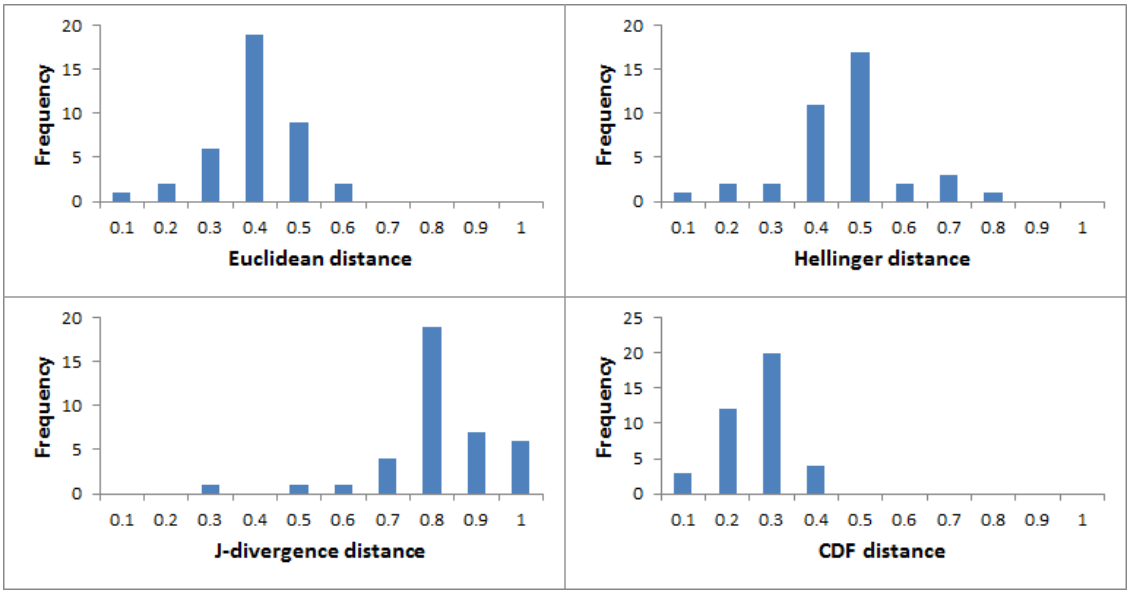

Figure 38: Histogram of the link strength for the Letter network

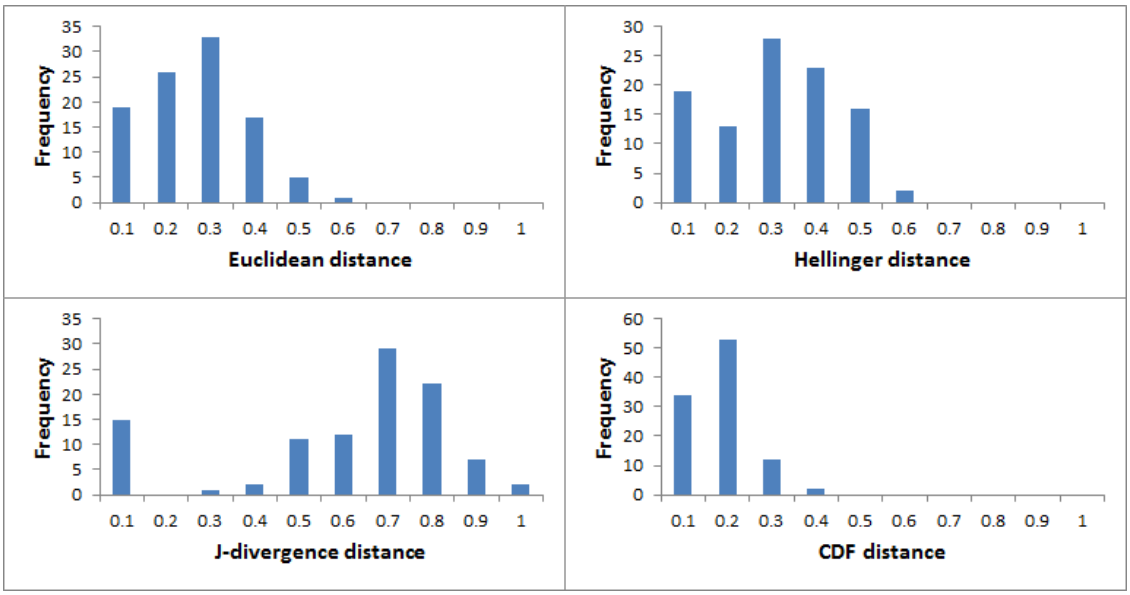

Figure 39: Histogram of the link strength for the Molecular Biology network

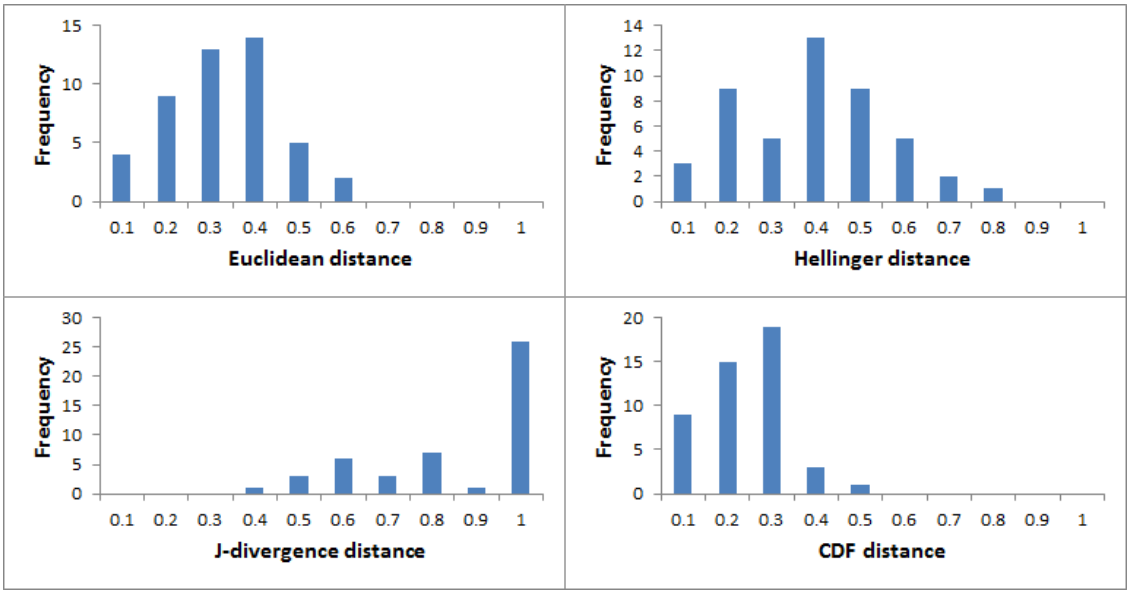

Figure 40: Histogram of the link strength for the Mushroom network

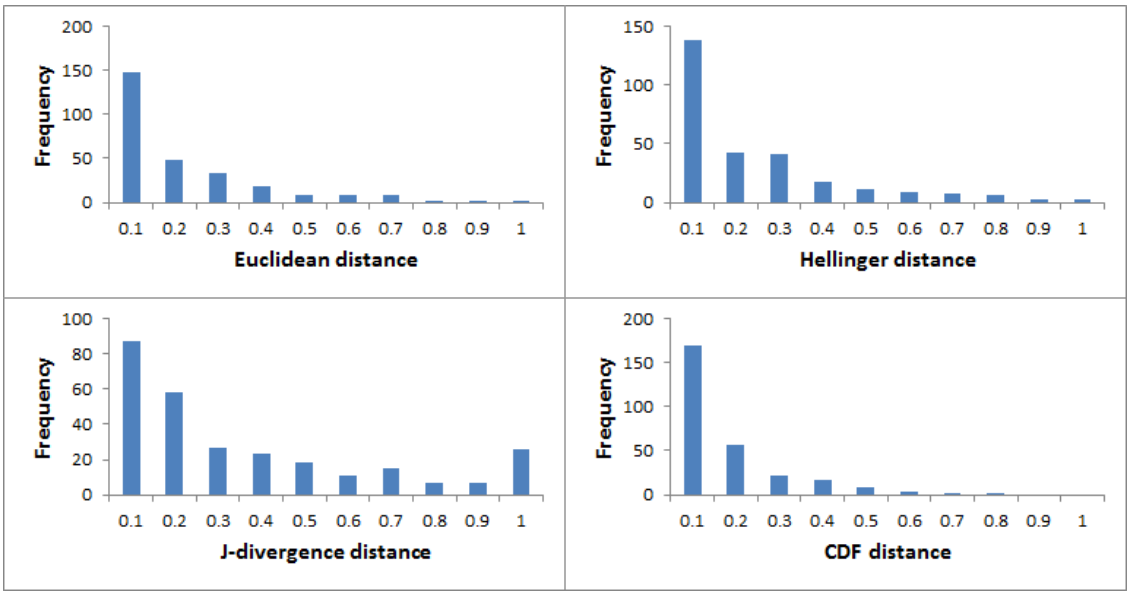

Figure 41: Histogram of the link strength for the Nomao network

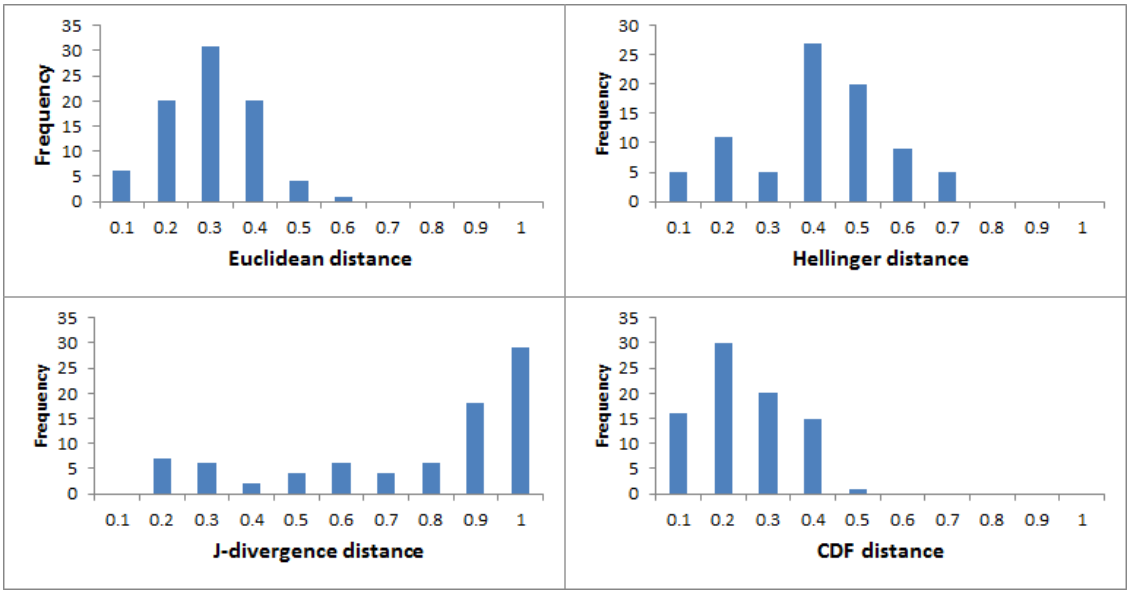

Figure 42: Histogram of the link strength for the ORHD network

5.3.2 Classification accuracy

In testing the classification accuracy of the simplified models on the original UCI Machine Learning Repository data sets, I used the simplest possible criterion, which is that the model guesses the most likely class to be the correct class for each record. I show eight plots of models' classification accuracy as a function of the four strength of influence measures in Figure 43 and 44. Figure 45 and 46 show eight plots of model's classification accuracy as a function of the percentage of the arcs removed for each of the networks.

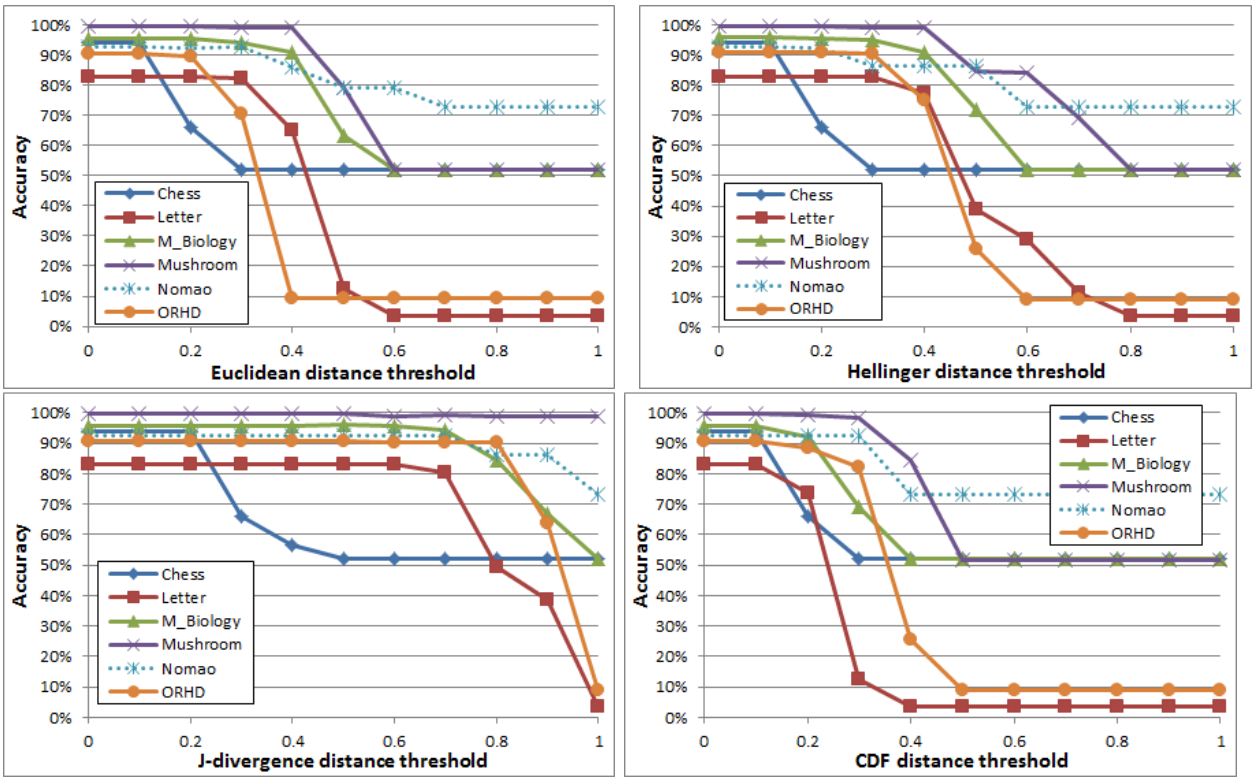

Figure 43: Classification accuracy as a function of the distance threshold for each of the four measures of the link strength for the BS models

The results show that for all link strength measures, except J-divergence, the classification accuracy does not decrease much from the gold standard when the threshold is below 0.2 (this corresponds to removal of around 20 percent of all original arcs). Then the accuracy drops sharply and reaches a plateau after roughly 0.6 (when roughly 60–80% of the arcs have been removed; see Figure 43 and 44).

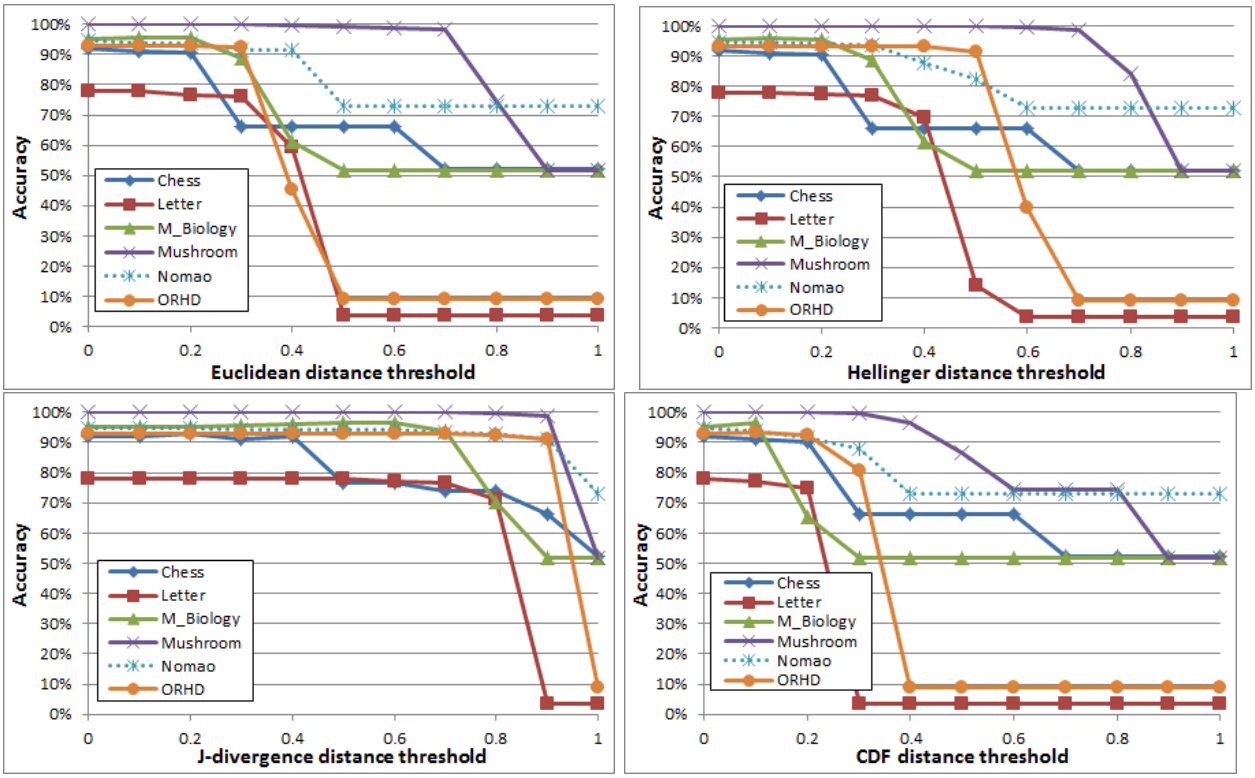

Figure 44: Classification accuracy as a function of the distance threshold for each of the four measures of the link strength for the TAN models

I explored further the reason for the sudden drop in the curves and found that this is related to removal of arcs between the class node and the nodes belonging to its Markov blanket. In the experiment, five of the six data sets contained no missing data. When there are no missing data, any node that is not in the Markov blanket of the class node will not affect the accuracy. In TAN models, all feature nodes belong to the Markov blanket of the class node. I show in Figure 47 that the accuracy reaches the plateau point when all the arcs in the Markov blanket are removed.

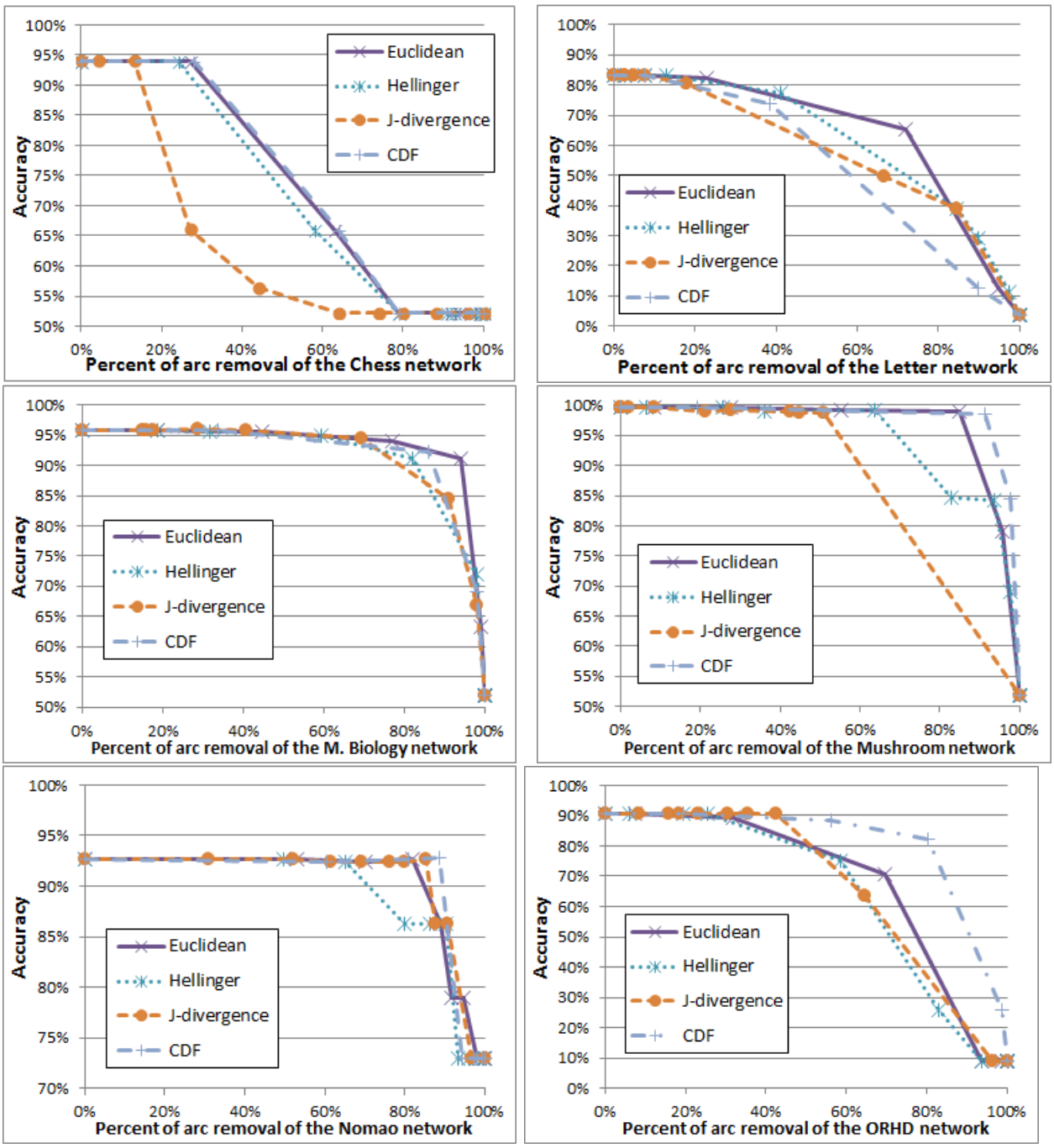

Figure 45: Classification accuracy as a function of the percentage of arcs removed for each of the six data sets

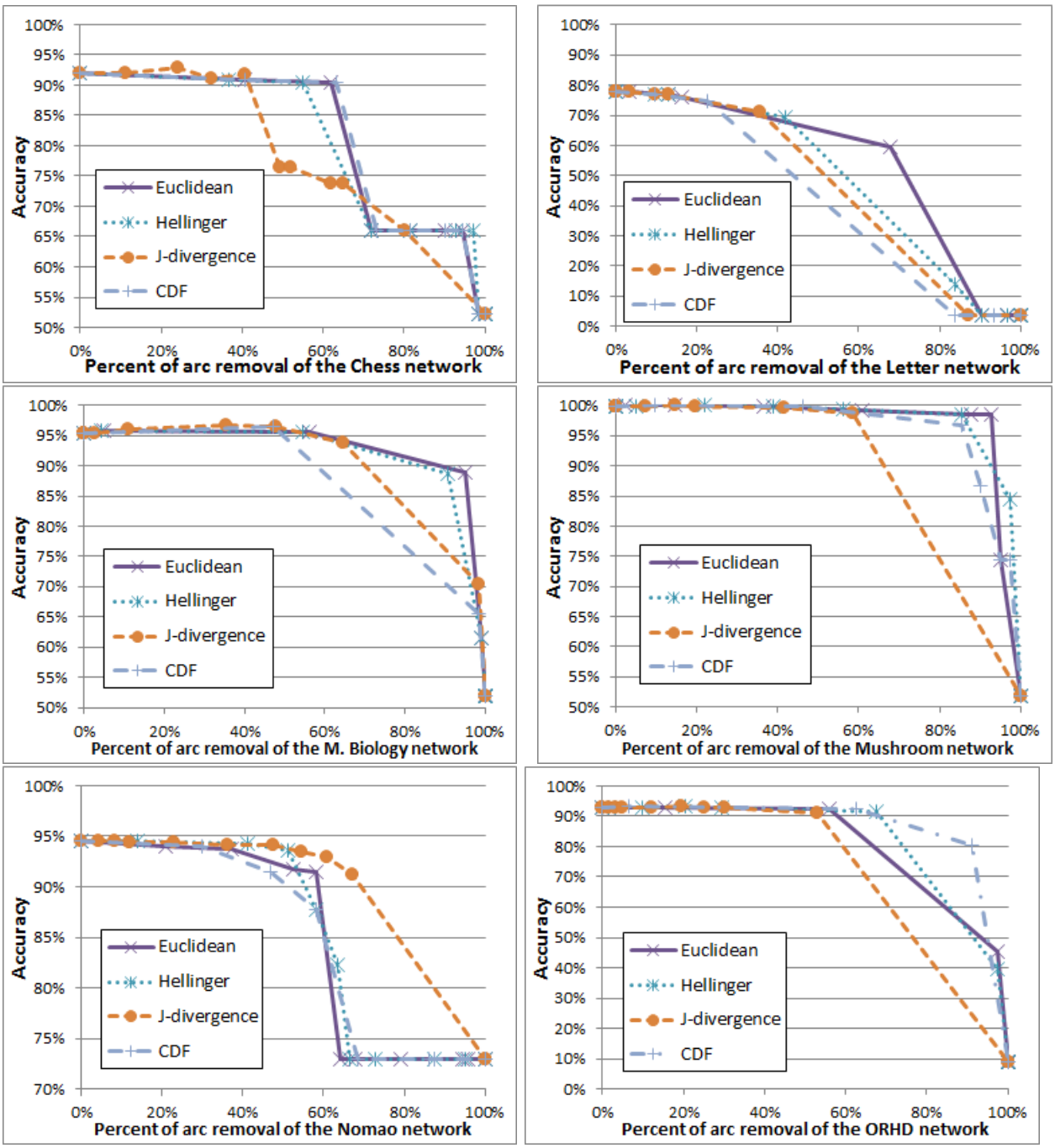

Figure 46: Classification accuracy as a function of the percentage of arcs removed for each of the six data sets for the TAN models

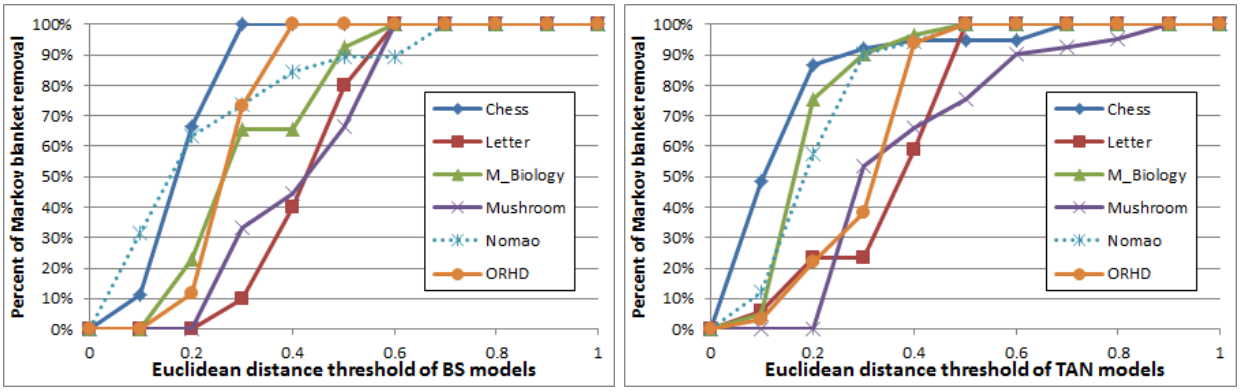

Figure 47: Percentage of arcs within the class node's Markov blanket removed as a function of the Euclidean distance threshold

5.3.3 Memory usage and computation time

I measured the memory usage and the time taken to perform inference on simplified networks relative to the memory usage and inference time on the original (gold standard) networks. Removal of weak arcs can lead to significant savings in memory. Figure 48 shows the total clique tree size as a function of the percentage of arcs removed. We can see that even with as few as 20 percent of the weakest arcs removed the savings in memory approach an order of magnitude, which can mean a difference between an intractable and a tractable network.

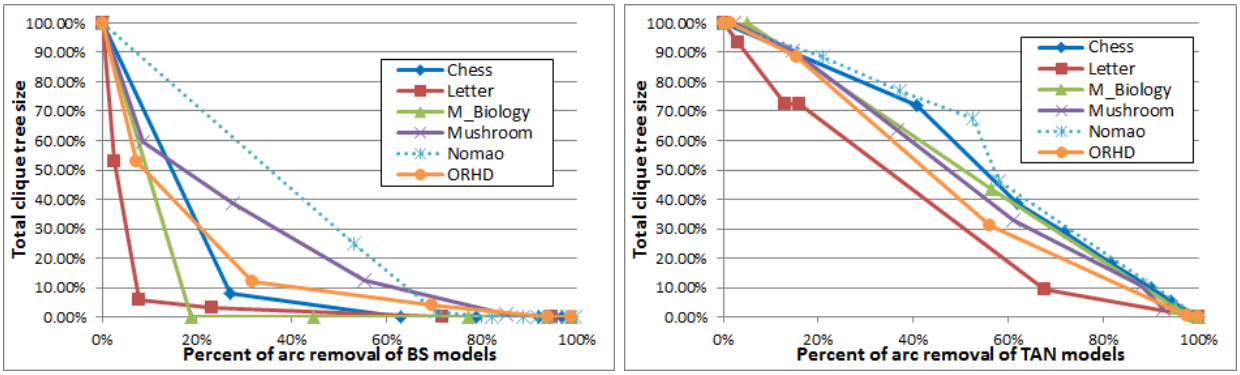

Figure 48: Total clique tree size as a function of the percentage of arc removed

Figure 49 shows the computation time as a function of the percentage of arcs removed.

Here also, we can see that even with as few as 20 percent of the weakest arcs removed the computational savings can be significant. The results here are somewhat confounded, as the computation time includes the time taken to create the clique tree, an integral part of the inference procedure as implemented SMILE©.

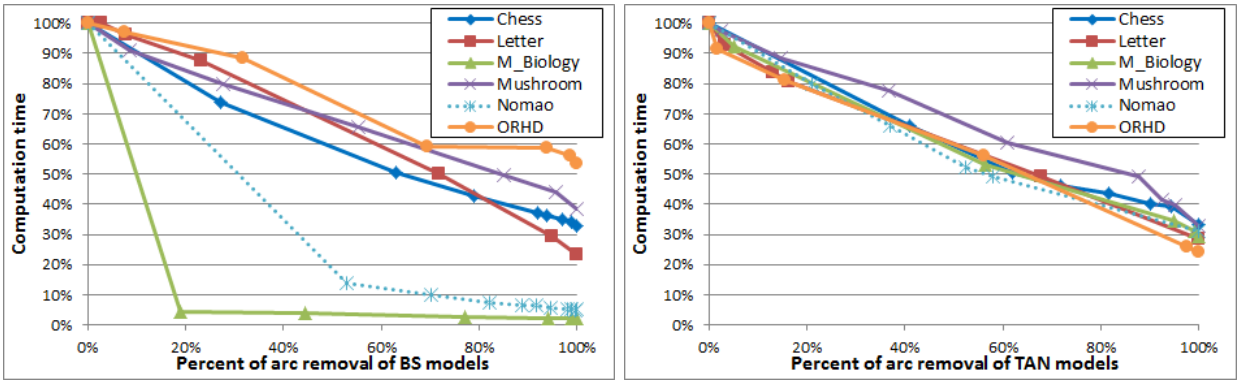

Figure 49: Computation time as a function of the percentage of arc removed

Clique tree size and computation time go hand in hand. Figure 50 shows this relationship between the two for each of the networks studied.

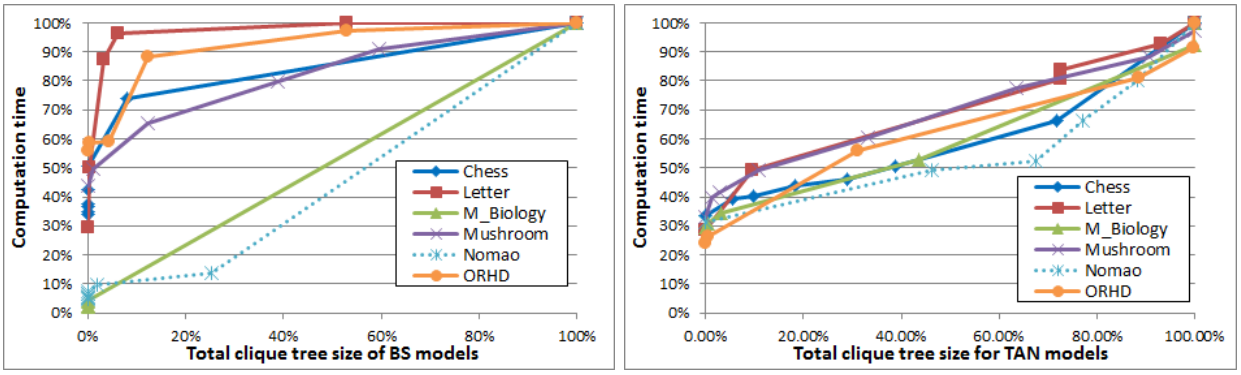

Figure 50: Computation time as a function of the total clique size

5.4 DISCUSSION

I reported the results of an empirical evaluation of simplifying the structure of Bayesian networks for the purpose of inference by removing weak arcs. I calculated four measures of strength of each arc and then used these measures to sort arcs from the weakest to the strongest. I conducted a series of experiments on six networks selected from the UCI Machine Learning Repository, in which I systematically removed arcs from the weakest to the strongest with the regular Bayesian network model and the TAN model. I measured the resulting classification accuracy of models, clique tree size, and computation time.

The complexity of a Bayesian network can be reduced by removing its weakest arcs, without compromising its accuracy.

The results support the hypothesis because the results for both models show that removing up to roughly 20 percent of the weakest arcs in a network has minimal effect of its classification accuracy. At the same time, both the amount of memory taken by the clique tree and the amount of computation needed to perform inference decreases significantly.

We can apply these results to the query-based diagnostics model. The TAN model and MARILYN's model share the same characteristics where all feature nodes belong to the Markov blanket of the class node.

6.0 EVALUATION OF QUERY-BASED DIAGNOSTICS

In this chapter, I evaluate a prototype of MARILYN systematically, based on several real data sets.

6.1 EXPERIMENT

6.1.1 The Data

I tested the accuracy of MARILYN on four different data sets listed in Table 13. I collected the computing lab data set over the course of two semesters at a help desk of a University of Pittsburgh campus computing lab. Typical campus computing lab help desk problems involve printing problems and printer troubleshooting. Among the four hundred cases in the data set, there are a total of 16 different observations, 12 different context variables, and 21 different problems. The remaining three data sets originate from the UCI Machine Learning repository and were selected based on the following four criteria:

- The data include a known class variable.
- The attribute types of all variables are discrete. I wanted to avoid the need for discretization, which could become a factor confounding my experiment.
- The number of cases in the data file should be over 100, which I believe to be large enough for the purpose of the experiment.
- The data should have been used in the literature in the past, so that I have information about baseline accuracy of learning algorithms.

The three medical UCI Machine Learning repository data sets that fulfilled the above requirements were *SPECT Heart*, *Breast Cancer* and *Lymphography*. Their properties are listed in Table 13.

Table 13: Data sets used in my experiments.

Data set	**Records**	**Attributes**	**Class variables**	**Missing values**
Computer lab	400	49	21	No
SPECT Heart	267	23	2	No
Breast Cancer	286	10	2	Yes
Lymphography	148	19	4	No

6.1.2 Methodology

I test the accuracy of MARILYN as a function of the number of cases that it has seen on each of the data sets listed in Table 13. This is of interest because the idea of query-based diagnostics is meant to work especially when there are no data that can be used to learn a model. Availability of a complete data set would make MARILYN useless, as the model could be learned from data by means of any of the Bayesian network learning methods available in the literature.

I imitated MARILYN's diagnostician's work-flow, which consists of entering three types of information: context information, observations, and the final diagnosis. While, in case of the computing lab help desk data, I had full knowledge of the three types of information, I did not know which of the features in the medical data sets were context variables and which were observations. Effectively, I treated all features in these data sets as observations. This is a conservative assumption, as it is an additional handicap for MARILYN in the experiments. The effect of my treatment of the medical data was that MARILYN constructed two layer BN2O networks in these cases, similarly to the QMR-DT model [Middleton et al., 1991].

I ran MARILYN 30 times for each data set, randomizing each time the order of records in the data file. The order of the records offered to MARILYN may affect its accuracy and

presenting different orders allows us to observe a range of behaviors. I used the simplest possible criterion in making a diagnostic decision and assumed that the most likely diagnosis is MARILYN's final diagnosis. This is, again, a conservative assumption, as the system displays the top n most likely diagnoses and this gives the user a chance to improve on the system, especially in the early stages, when the model is very crude.

6.2 RESULTS

I calculated MARILYN's cumulative accuracy after each record, so as to know how system's accuracy develops as a function of the number of diagnostic cases that the system has seen. Figure 51 shows the average accuracy of MARILYN as a function of the number of cases for each of the four data sets with range of the curves (vertical bars) plotted for selected number of records. The plots show that while MARILYN was rather weak in the beginning (during the first thirty cases or so), it became quite accurate after roughly 70 to 100 cases (this varied per data set). Interestingly, in case of the SPECT data set, MARILYN reached the accuracy of over 60% after fewer than ten cases. In all data sets, 40 or so cases were sufficient to reach a reasonable accuracy. This accuracy not only improved over time but also improved reliably, as indicated by smaller variance in the results of different random orders of records. Interestingly, there is some similarity between the plots of MARILYN's accuracy, as in Figure 51, and the so called *power curve of practice* in the psychology literature [Newell and Rosenbloom, 1981].

Cumulative accuracy for the last record entered is the final accuracy result of MARILYN on the data set. MARILYN's final accuracy on the four data sets was 90.25%, 78.75%, 77.18%, and 69.95% for the *Computer Lab*, *SPECT Heart*, *Breast Cancer*, and *Lymphography* data respectively (see the extreme right cumulative accuracy in Figure 51). It has to be added that in achieving this result MARILYN has seen (i.e., was trained on) the average of 50% of the records. When processing the first case, MARILYN has seen zero prior cases, when processing the 10th case, it has used only 9 preceding cases, when processing the last, nth case, it has seen $n-1$ preceding cases. The average number of training records is thus $n/2$.

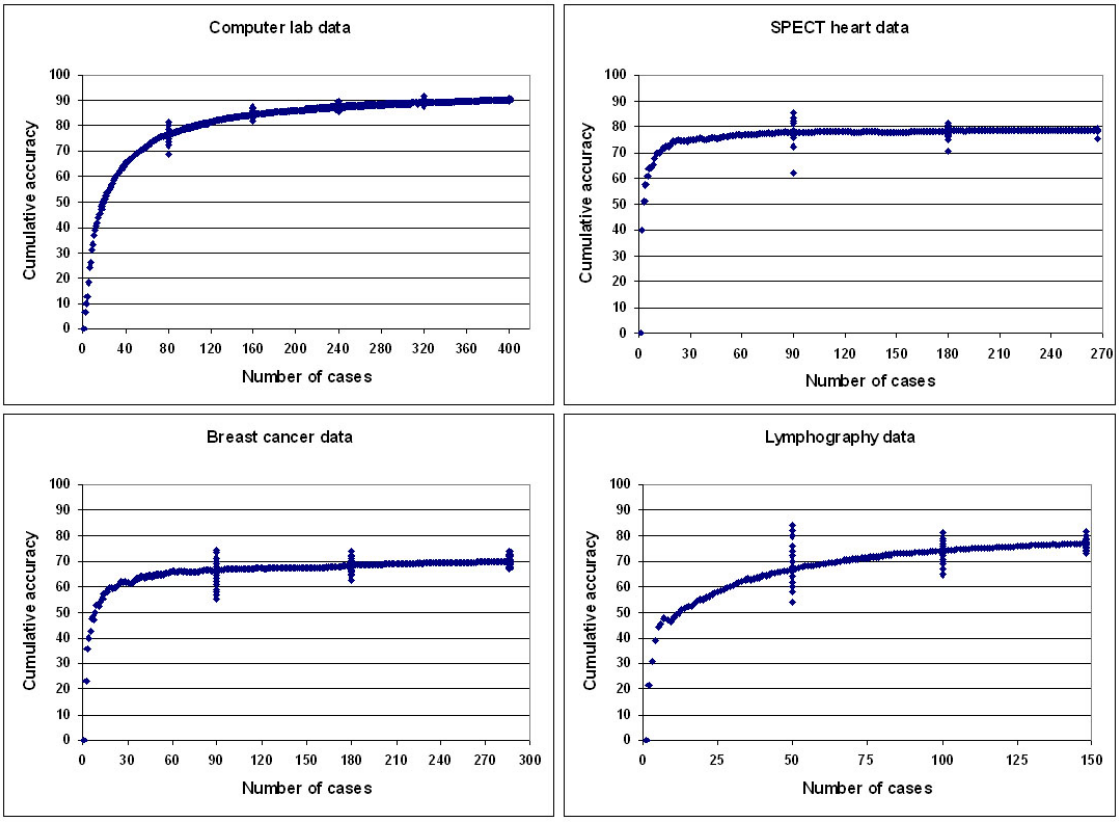

Figure 51: MARILYN's cumulative accuracy as a function of the number of cases seen

Table 14: Accuracy comparison results with Bayesian approaches using leave-one-out cross validation

Data set	MARILYN	**Naive Bayes**	**Greedy Thick Thinning**
CompLab	94.50%	94.25%	91.25%
SPECT	79.40%	79.40%	78.65%
BC	68.18%	42.57%	47.97%
Lymph	81.08%	66.08%	67.83%

In order to disambiguate the specific procedure that I used to obtain MARILYN's cumulative accuracy from the capability of the learning function by itself, I performed an experiment in which I allowed MARILYN to learn from all available records alongside with two Bayesian learning algorithms: (1) Naive Bayes [Langley et al., 1992], and (2) a Bayesian search algorithm Greedy Thick Thinning [Dash and Druzdzel, 2003]. I used the leave-one-out cross validation to measure the accuracy of the three classifiers, assuming that the diagnosis is correct when the most probable class matches the correct class. I show the results of this experiment in Table 14. MARILYN performed better than Naive Bayes and GTT on all data sets. I believe that some of MARILYN's power comes from its priors and structural information extracted from the data.

The three data sets that I chose for my experiments have been subject of experiments published in the literature. The best accuracy result for SPECT heart data with CLIP3 machine learning algorithm is 84% [Kurgan et al., 2001]. The best accuracy achieved on the Breast cancer data was by means of k-nearest neighbor (k-NN) algorithm and amounted to 79.5% [Kononenko et al., 1997]. The best accuracy on the Lymphography set was achieved by means of the Tree-Augmented Naive Bayes algorithm and was 85.47% [Madden, 2002]. I compared MARILYN's accuracy to each of these, repeating the experiment under the same conditions, i.e., with precisely the same cross-validation method as used in the experiments reported in the literature. Table 15 shows the accuracy for each of the data sets and each of the algorithms. For the SPECT Heart, I used verification test criteria are shown in Table 16. The results show that MARILYN was better in terms of accuracy and specificity but was lower in terms of sensitivity.

Table 15: Accuracy comparison results with state of the art approaches

Data set	MARILYN	**CLIP3**	**k-NN**	**TAN**
SPECT	93.58%	84%	N/A	N/A
BC	73.02%	N/A	79.50%	N/A
Lymph	81.92%	N/A	82.60%	85.47%

Table 16: Accuracy comparison results of SPECT Heart data set using verification test

	Sensitivity	**Specificity**	**Accuracy**
CLIP3	80%	84.30%	84%
MARILYN	46.67%	97.67%	93.58%

While MARILYN's accuracy is typically lower than that of the state of the art learning algorithms, it is certainly in the same ballpark. I would like to point out that the best results reported in the literature belong to different algorithms, i.e., there seems to be no algorithm that is uniformly best on all data sets. If the same algorithm were applied to all four data sets, there is a good chance that its accuracy on some of these could be worse than the accuracy of MARILYN.

6.3 DISCUSSION

I described a series of experiments that subject a prototype implementing passive, incremental model construction to a rigorous practical test. Data obtained from the UCI Machine Learning repository made the evaluation fairly realistic. The results of my experiments show that a system like MARILYN is capable of giving reasonable suggestions after a modest number of observed cases. Accuracy in the order of 70-90% typically occurred not later than after roughly 40 cases. Even though this experiment offers just a few data points and this type of systems need to be tested more in practice, I believe that the result is very promising and compares favorably with state of the art approaches based on learning.

7.0 PRACTICAL FIELDING OF QUERY-BASED DIAGNOSTICS

In this chapter, I describe an experiment that tests MARILYN in real diagnostic environments. I present an evaluation of MARILYN diagnostic model construction by means of six hundred help desk cases at the School of Education, University of Pittsburgh. An important issue with MARILYN is how the system handles the growth of the model structure. The challenge for the algorithm is how to handle the growth of the number of parents for a single node and the growth of the number of parameters. As the number of parameters and cases increases, the EM algorithm requires longer time for their parameters to converge and slows the network construction.

7.1 EMPIRICAL EVALUATION

7.1.1 Experiment Data

I has been working as a computer lab consultant between eight and ten hours per week from Fall 2009 to Spring 2014. This has offered an excellent opportunity to collect real help desk data. The School of Education technology department provides services to all School of Education faculty, staff, and students. Their services include computer labs, media services, the Help Desk, and the School of Educations intranet. Their operation hours are Monday through Friday 8:30 a.m.—5:30 p.m. The Help Desk locates at 5308 Posval Hall. A customer can walk-in, call or submit a School of Education help desk ticket on the School of Education support websites. There are four types of help desk tickets: problems, requests, questions, and others (see Table 17).

To collect the diagnostic cases for MARILYN, I used the problems type tickets. The help desk problems tickets include, but are not limited to, printing job problems, printer troubleshooting, software, hardware, and other diagnostic problems. I retrieved the help desk tickets from the Help Management System shown in Figure 52. In each case, the data were collected in three steps. First, I used the Ticket Search Options to list the tickets by year and selected *closed* on the status of tickets. Closed tickets denoted that the case was closed or completed. Second, I recorded all information available in the tickets as shown in Figure 53. Third, I clean up the data by distinguish all the data into three types of variables: observations, context information, and final diagnosis. However, I could not use all the available cases in Table 17. There were several reasons: tickets were closed due to the lack of communicate from customers, the consultants did not specific a final diagnosis on the tickets, or tickets were not related to diagnosis. I recorded all qualify cases in the Microsoft Office Excel work book shown in Figure 54. There were 600 qualifying tickets between January 2009 and December 2013. In this School of Education help desk domain, the collected data indicate that there is always at least one observation and only one problem; however the context information may be not available for all cases. Data range from one to four observations, context information range from zero to three and only one problem occurs. There are a total of 120 different observations, 24 different context information, and 60 different problems. I show the distribution of frequency for the six-hundred cases in Figure 55. This figure shows that the distribution of problems is skewed with the top five problems covering roughly 40% of the cases. As we have seen in the CSSD data set (Chapter 6), skewness of the problem set seems typical for practical diagnostic domains. Essentially, some problems are common and occur often with others occurring only sporadically.

7.1.2 Experimental Design

I used the same methodology in Chapter 6 to evaluate MARILYN's accuracy. In this help desk domain, I had full knowledge of the data set to identify three types of information to build a 3-layer BN3M model. I performed an additional practical test in a situation where the

Table 17: Number of the School of Education help desk tickets from 2009–2013

Ticket Type	2009	2010	2011	2012	2013
Others	4	8	13	5	53
Problems	782	579	954	705	590
Questions	5	2	2	12	11
Requests	393	116	77	182	101

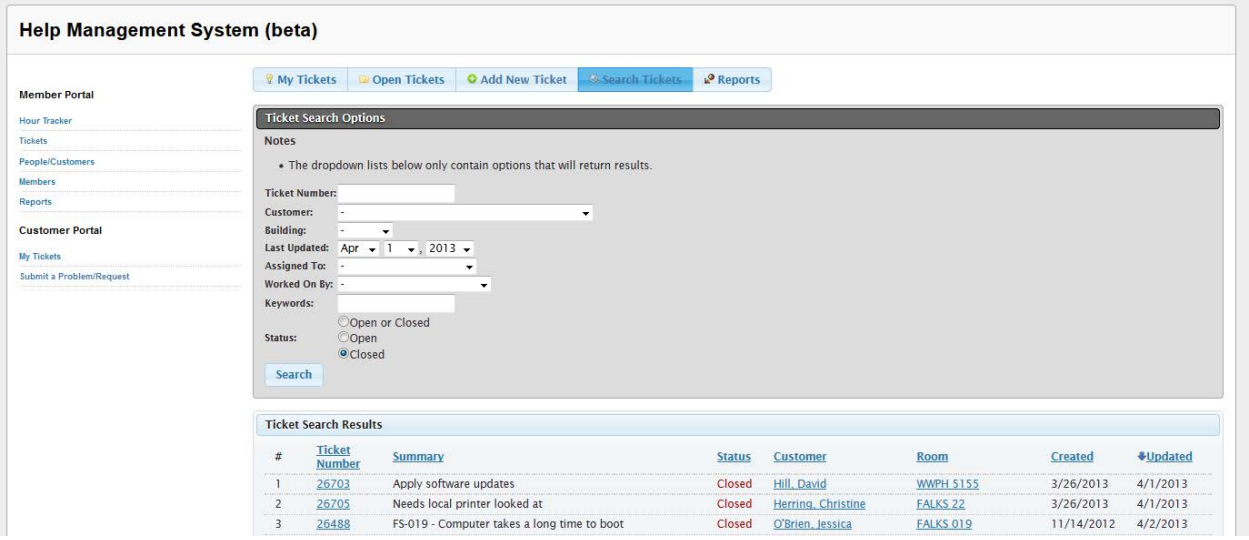

Figure 52: The Help Management System tickets search web page.

user of MARILYN do not have the full knowledge of the domain. In this situation, MARILYN treats all features in the data set as observations and constructs two-layer BN2O networks instead.

I ran MARILYN 20 times for each network type, randomizing each time the order of records in the data file. As the system displays the top n most likely diagnoses, it gives the user a chance to improve on the system. I sorted the results from the most to the lease

Ticket Summary

Cannot Log In to Intranet

Ticket #: 26775 **Customer:** Ralph Longo **Room:** WWPH 4320
Opened: 5/6/2013 (330 days ago) **Last Action:** 5/6/2013 (330 days ago) **Status:** Closed **Type:** Problem
Priority: Normal

Expired Password: Dr. Longo's password was expired. Ryan reset his password, and the problem was fixed. ...

Figure 53: An example of a completed help desk tickets.

case	Observation	Observation2	Observation3	Observation4	Context	Context2	Context3	Diagnosis
1	a user has trouble printing	printer is working						computer needs restart
2	calendar problem	a user does not sign into the Exchange server			MSOutlook			permission needed
3	email problem	not connected to server			Mac			SSL authentication
4	suspicious popup	Malwarebytes found infect objects						virus malware
5	unable to access Internet	multiple connections was found						too many connections connect to internet
6	unable to access Intranet							password

Figure 54: An example of help desk tickets data set.

likely. The default size of the diagnostic windows is nine. However, I only chose the first five suggestions of MARILYN to be used for comparing accuracy with the original data.

Figure 56 shows the number of different problems encountered as a function of the number of cases. The cases are processed in chronological order of the tickets. In a total of 60 problems in the domain, 30, 40, and 50 problems are covered in the first 80, 160, and 250 cases respectively. This number assumes that the MARILYN model should be able to give suggestions for most of the cases after seeing the first 250 cases.

7.1.3 Results

Figure 57 shows the 3-layer Bayesian networks created by MARILYN after six-hundred cases. There are a total of 204 nodes and a total of 350 arcs. The top layer has 24 *context informa-*

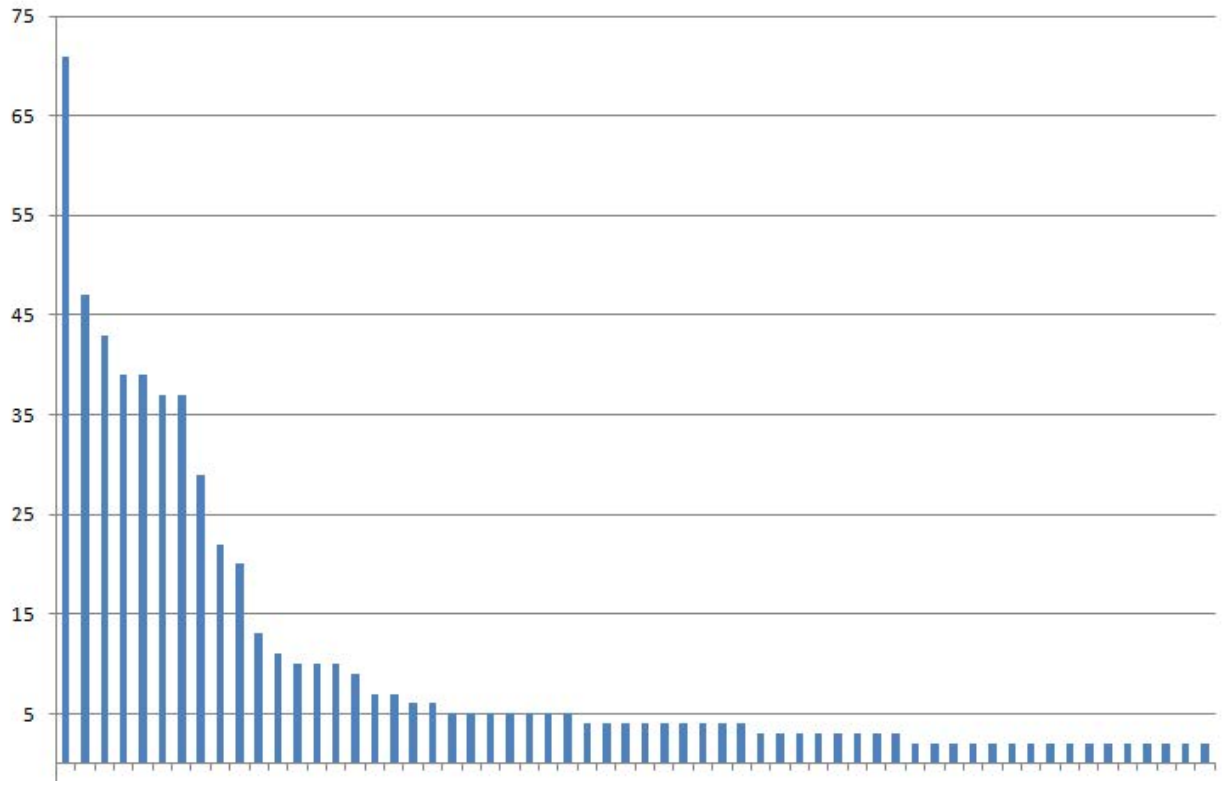

Figure 55: Problem distribution of six-hundred help desk data set.

tion nodes. The middle layer has 60 *fault* nodes and the bottom layer has 120 *observation* nodes. The maximum number of parents per child is 12. The average number of parents per node is 1.7.

I used the same cumulative accuracy in Chapter 6 to test MARILYN's accuracy. I applied the same measure of accuracy as Oniśko et al. [2001]. I am interested in MARILYN suggestions that the list of possible diagnoses contains the correct diagnosis for a small set of values. I chose a *window* of W=1, 2, 3, 4, and 5.

Figure 58 shows the average cumulative accuracy of MARILYN as a function of the number of cases that have been entered into the system. With six-hundred cases, MARILYN reached the average accuracy of 60.99%, 69.99%, 72.87%, 74.62%, and 76% for W1, W2, W3, W4, and W5 respectively.

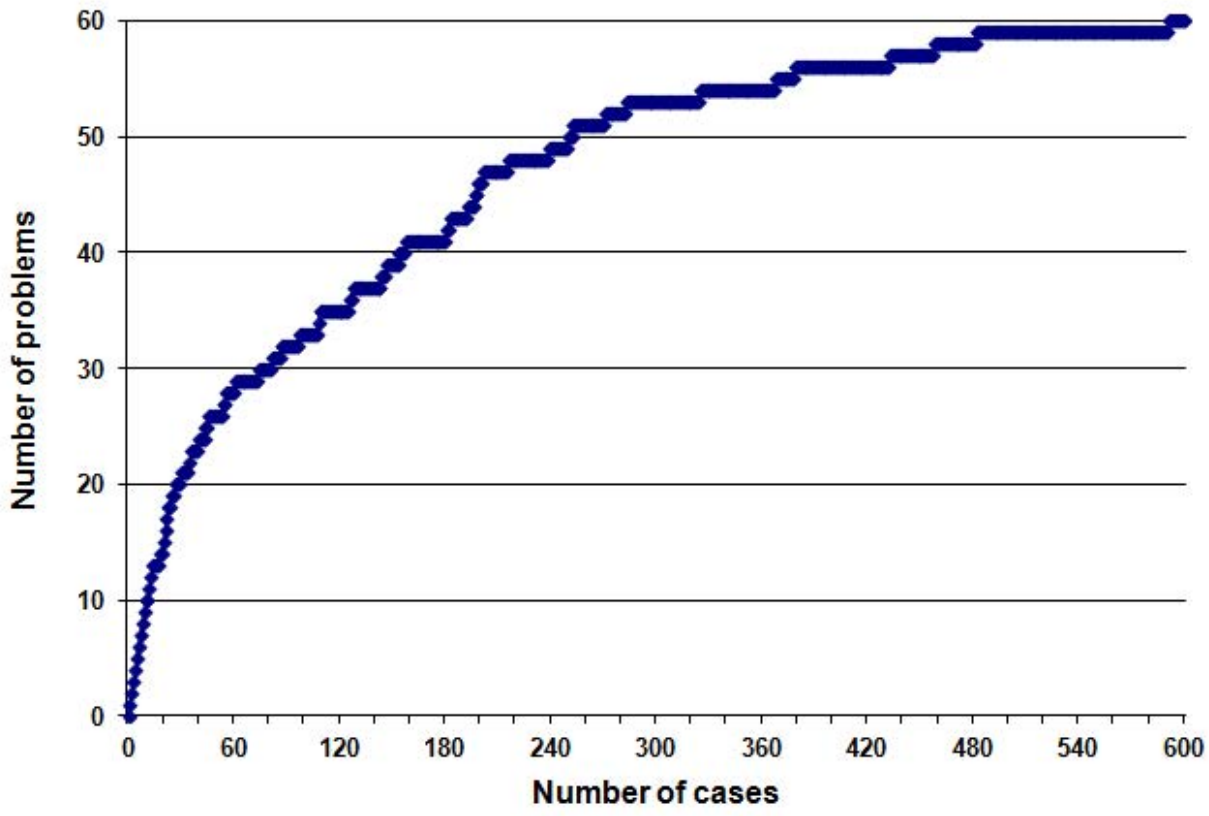

Figure 56: The accumulate of the help desk problems entering to the MARILYN.

For the 2-layer network, there are a total of 204 nodes and a total of 350 arcs. The top layer has 60 *fault* nodes and the bottom layer has 144 *observation* nodes. The maximum number of parents per child is 13. The average number of parents per node is 1.7.

Figure 59 shows average cumulative accuracy of MARILYN as a function of the number of cases that have been entered into the system. With six-hundred cases, MARILYN reached the average accuracy of 61.77%, 71.72%, 75.50%, 78.09%, and 79.72% for W1, W2, W3, W4, and W5 respectively.

Figures 60 and 61 show the average accuracy of MARILYN for the most likely diagnosis (W1) with range of the curves (vertical bars) plotted for selected number of records. The

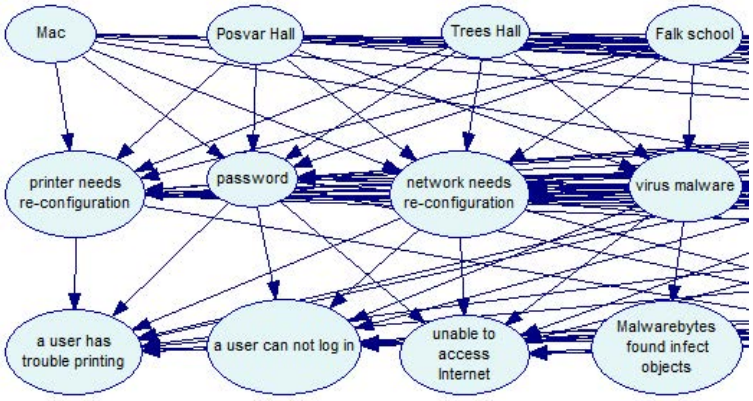

Figure 57: A section of the six hundred cases BN3M model built by Marilyn.

plots show a similar trend to the one observed in Chapter 6, i.e., MARILYN seems weak in the beginning, but it becomes quite accurate and consistent later.

In order to compare with others Bayesian learning algorithms, I performed MARILYN system's diagnostic accuracy with 10-fold cross validation. Here I use the simplest criterion that the most likely diagnosis is MARILYN's final diagnosis. Table 18 shows the results of this experiment. MARILYN 2-layer is more accurate than the others approaches.

Figure 62 shows histograms of classification accuracies ranked from the most to the least number of problems. It shows that MARILYN work well when there are only few number of cases per diagnosis. When the number of cases per diagnosis reaches around 30 to 40, all approaches are in the same order of magnitude.

7.1.4 Model management

There are two critical practical issues related to MARILYN model management: (1) dealing with continuous data streams, and (2) model growth.

As discussed in Chapter 4, batch learning makes the largest computational and storage demands but offers the highest resulting parameter accuracy. Online EM is a good alternative as it requires less computation time and no storage while achieving decent accuracy.

In the help desk application, real-time response is critical, the best practice is to use the

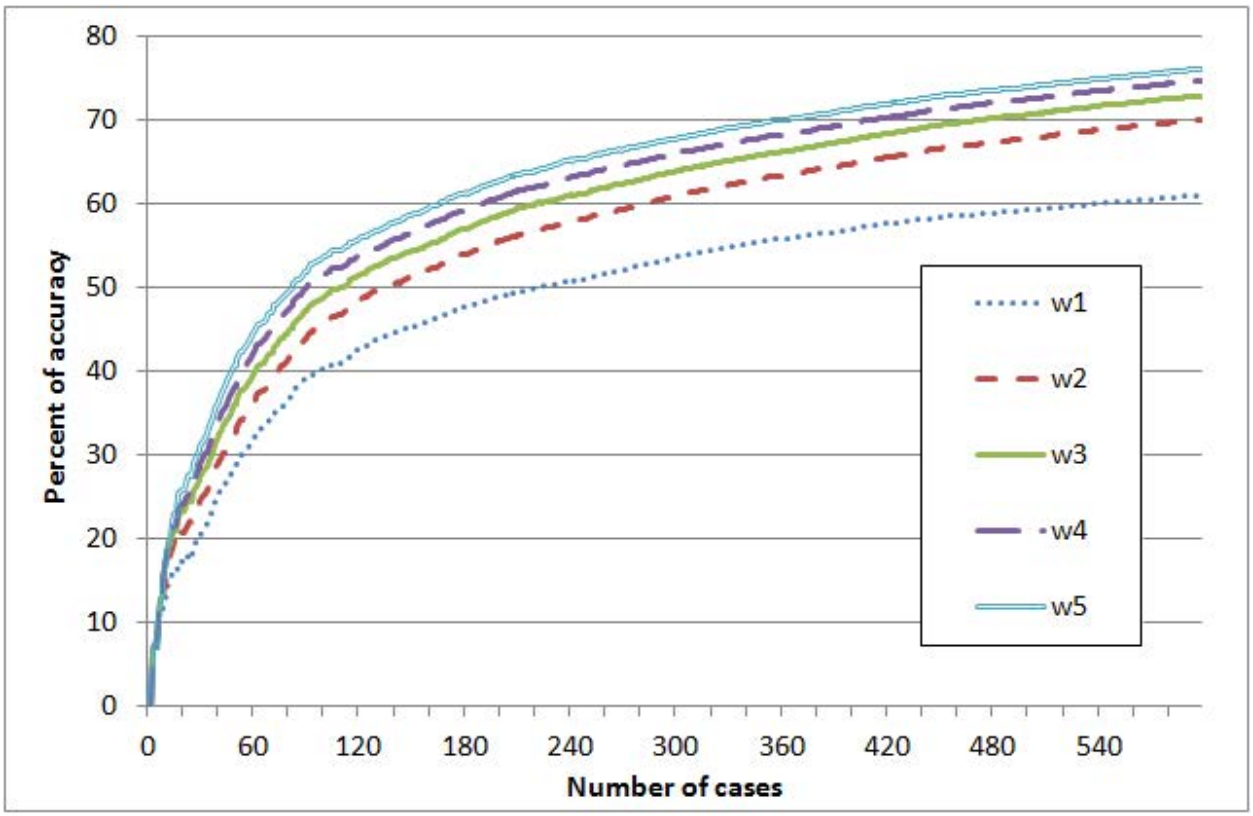

Figure 58: The graph shows average percent of accuracy as a function of the number of cases for 3-layer BN3M models

online EM algorithm during daily operations and the batch EM during maintenance hours. In the experiments described earlier in this chapter, I only used batch learning applied to the entire data set as the data arrived. Figure 63 shows that batch learning computation time grows linearly with the number of records. When there are fewer than 300 records, the computation time is less than 20 seconds. However, it becomes large later. Therefore, from 300th to 600th records, I invoked the batch learning every 10 records instead of every record. It not only saves computation time needed to run all the test but does not decrease the accuracy significantly. As I show in Figure 56, the MARILYN model encounters most of the cases after around the 250th case.

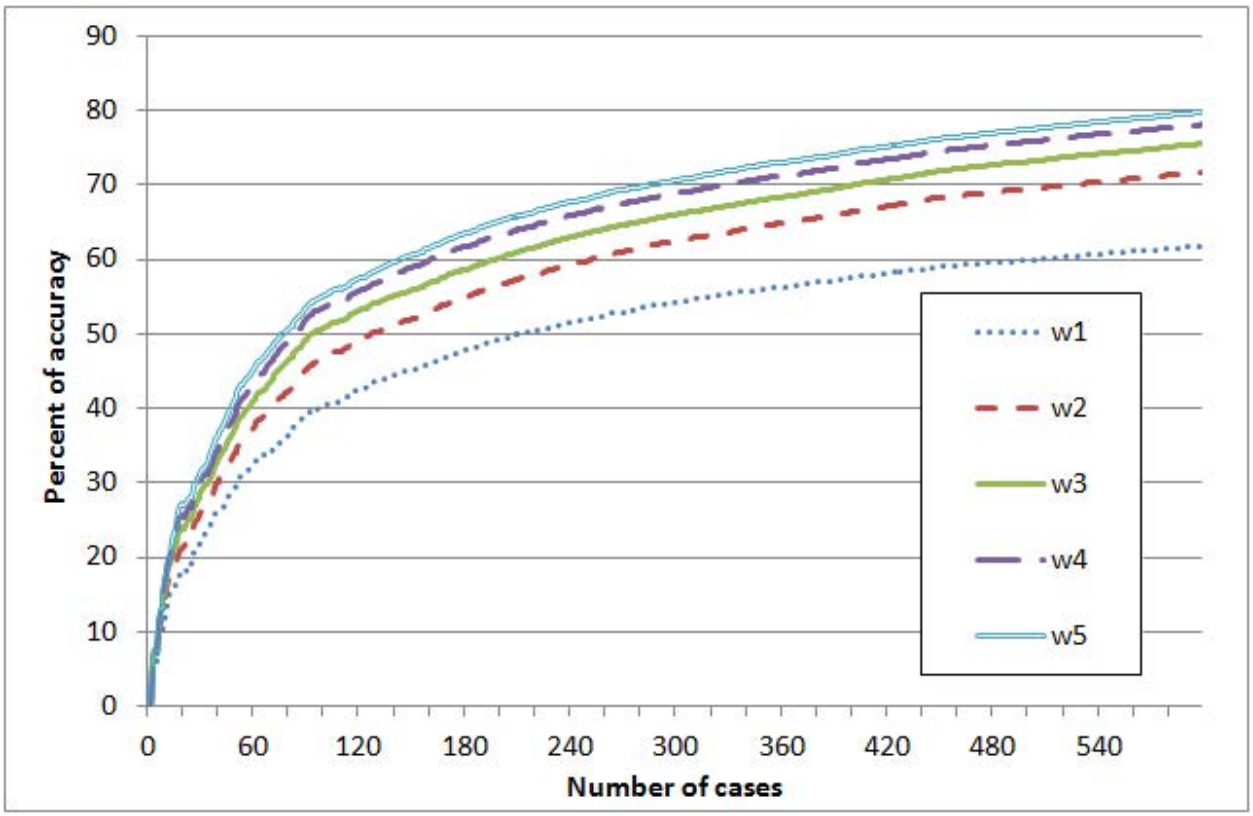

Figure 59: The graph shows average percent of accuracy as a function of the number of cases for 2-layer BN2O models

The second issue with MARILYN is how the system handles the growth of the model structure. The reasoning engine SMILE© is capable of handling large networks, and performs Bayesian inference in a fraction of a second. The challenge for the algorithm is how to handle the growth of the number of parents for a single node. In this experiment, even after six hundred cases have been entered in the system, this problem was not noticeable. The maximum number of parents per node for network was 13. This number are not large enough to degrade the speed of inference of MARILYN.

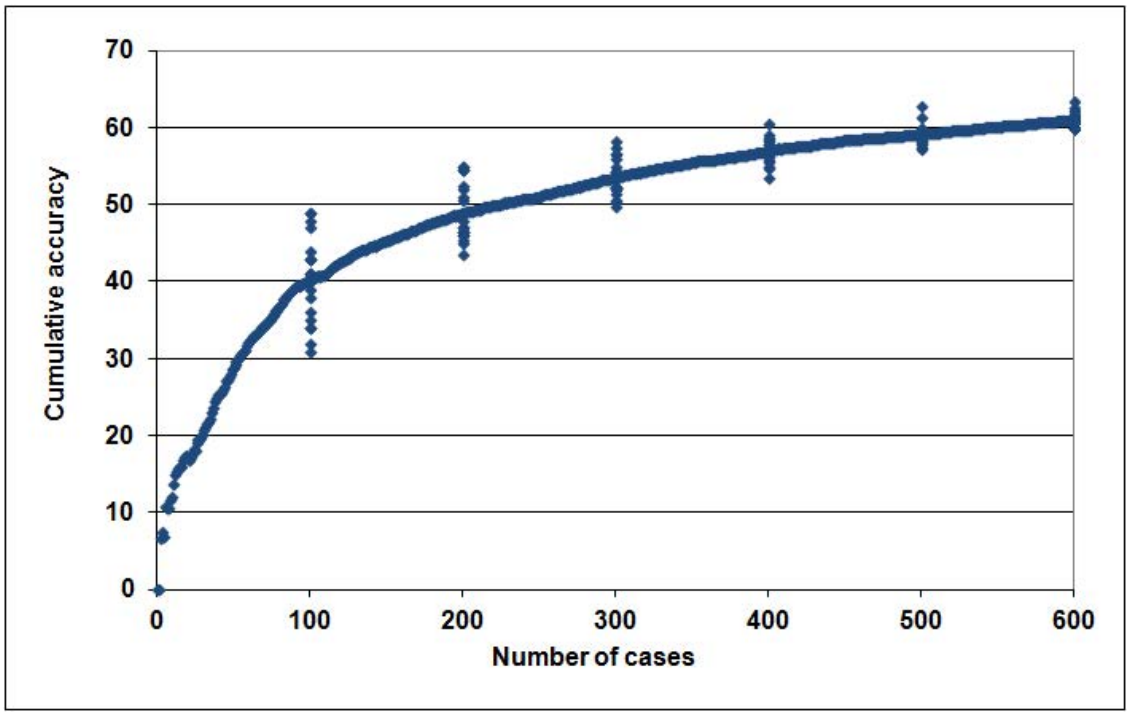

Figure 60: MARILYN's cumulative accuracy as a function of the number of cases seen for 3-layer BN3M models

7.2 3-LAYER VS 2-LAYER STRUCTURE

To find the difference in accuracy of MARILYN between a 3-layer BN3M model and two-layer BN2O model, I performed the Wilcoxon test [Demšar, 2006]. I could only use two data sets: CSSD and School of Education in which I have full knowledge of these domains. For CSSD data set, I need to ran additional two-layer BN2O MARILYN for 30 times with the same order of records as in Chapter 6. Then, I performed the Wilcoxon test for CSSD with 30 pairs of cumulative accuracy. For School of Education data set, I perform the Wilcoxon test with 20 pairs of cumulative accuracy. Table 19 shows the results of this experiment.

For the test results, the CSSD a 3-layer network is significantly more accurate than a 2-layer network. In case of the School of Education data set, the 2-layer network is significantly more accurate than the 3-layer network.

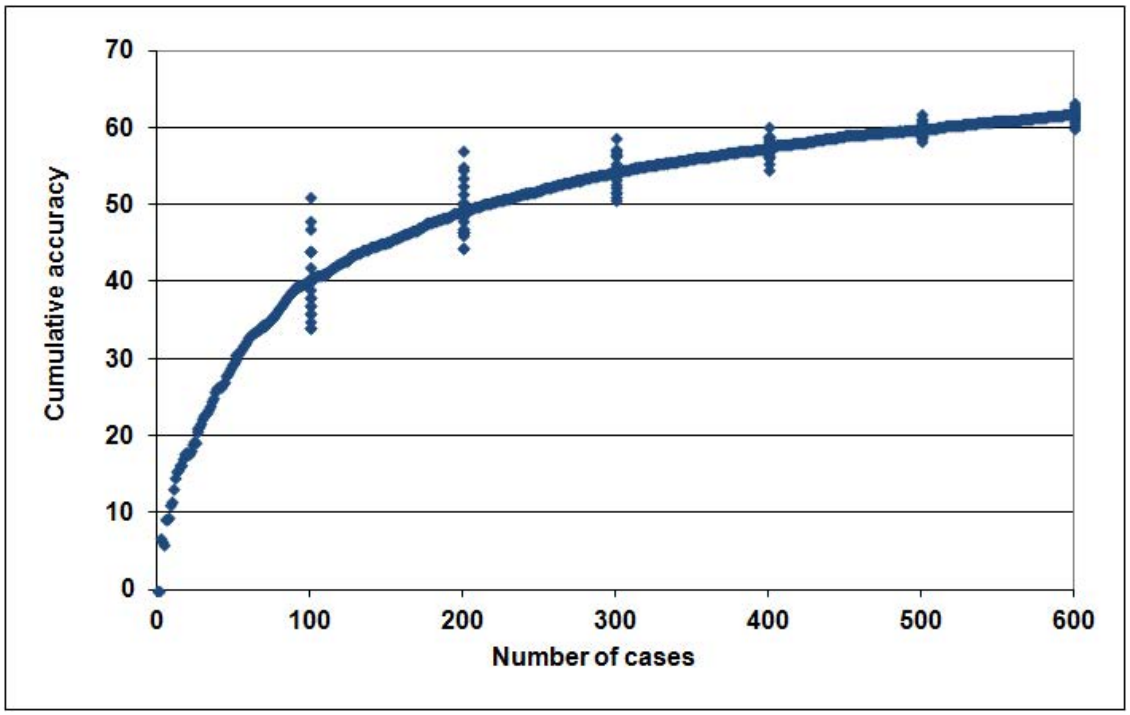

Figure 61: MARILYN's cumulative accuracy as a function of the number of cases seen for 2-layer BN2O models

Table 18: Accuracy comparison results with three Bayesian approaches using 10-fold cross validation

Learning algorithms	**Accuracy**
MARILYN 3-layer	70.33%
MARILYN 2-layer	**72.33**%
Naive Bayes	53.66%
Tree Augmented Naive Bayes	53.50%
Greedy Thick Thinning	40.33%

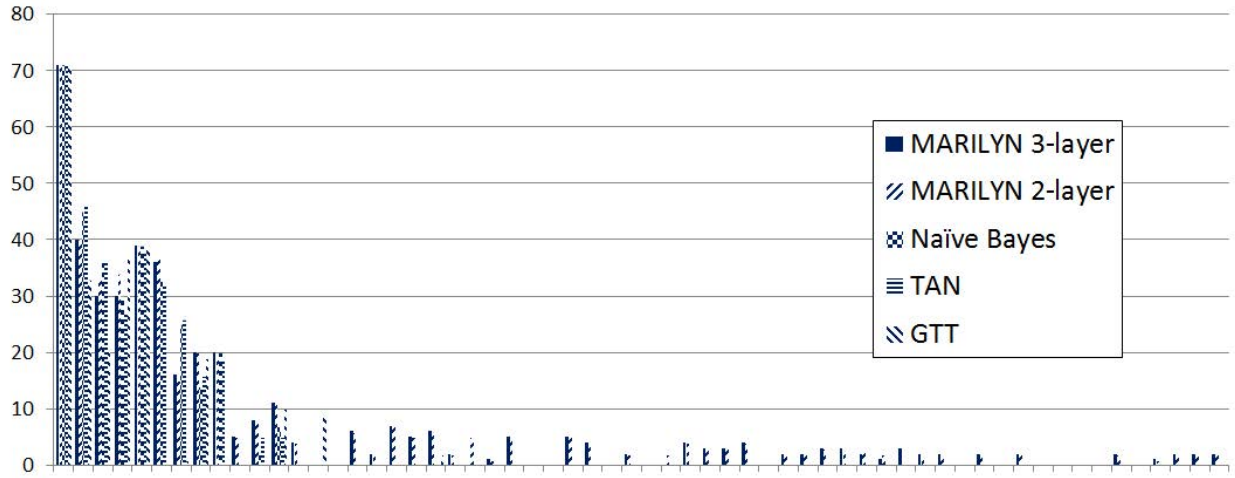

Figure 62: Distribution of number of correct classification cases among five approaches using 10-fold cross validation

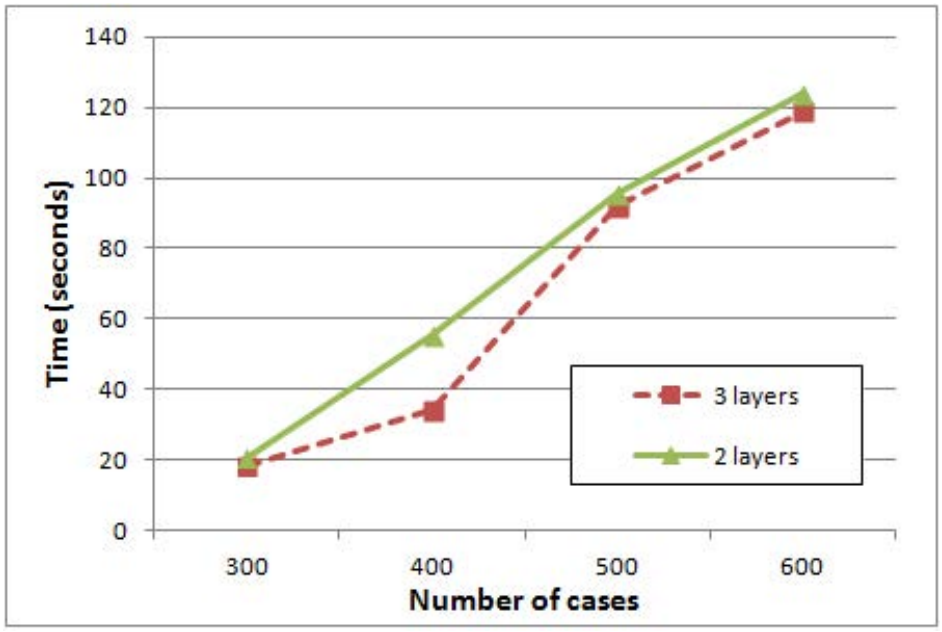

Figure 63: Help desk data set computation time by the batch algorithm

Table 19: Wilcoxon's statistical test between the 3-layer BN3M model and the two-layer BN2O model.

	CSSD	**School of Education**
Two-tail	**0.005735**	**0.0008841**
One-tail $3 > 2$	**0.002867**	0.9996
One-tail $2 > 3$	0.9974	**0.000442**

7.3 DISCUSSION

Query-based diagnostics approach requires fewer cases for learning than the other simple Bayesian networks in performing classification.

MARILYN is a passive diagnostic model construction tool that is able to give suggestions based on information entered by diagnosticians. I conducted an experiment to evaluate MARILYN's accuracy, testing the model by means of six-hundred cases of help desk data. This is a fairly realistic and similar to problems faced by most diagnositc situations. The results of the experiment showed that MARILYN is more accurate than the other Bayesian approaches based on learning when there are only few cases available as shown in Figure 62. This results support the hypothesis. I believe that some of MARILYN's power comes from its priors and structural information extracted from the data as discussed in Chapter 3.

I discuss practical issues related to MARILYN model management. In the help desk domain, it is advisable to use the online EM algorithm during daily operations and the batch EM during maintenance hours. The structural complexity of the networks is not large enough to justify the arc removal procedure.

8.0 DISCUSSION AND FUTURE WORK

Query-based diagnostics offers passive, incremental construction of diagnostic models based on the interaction between a diagnostician and a computer-based diagnostic system. Effectively, this approach minimizes knowledge engineering, the main bottleneck in practical application of Bayesian networks. While this idea is appealing, it has undergone only limited testing in practice. There are notable two issues that need to be addressed in practical query-based diagnostics systems: (1) processing of continuous streams of data and (2) the continuous increase in complexity of the models structure.

The main contribution of this dissertation are (1) an investigation of the optimal approach to learning parameters of Bayesian networks from continuous data streams, (2) dealing with structural complexity in building Bayesian networks through removal of the weakest arcs, and (3) a practical evaluation of the idea of query-based diagnostics.

I described a series of experiments that subject a prototype implementing passive, incremental model construction to a rigorous practical test. Data obtained from the UCI Machine Learning Repository and the two help desk domains made the evaluation fairly realistic. The results of our experiments show that a system like MARILYN is capable of giving reasonable suggestions after a modest number of observed cases. Accuracy on the order of 60-80% typically occurred not later than after roughly 60 cases. The result is very promising and compares favorably with state of the art approaches based on learning.

I conducted a series of experiments to find an optimal approach to perform Bayesian network parameter learning algorithms for continuous data streams. I compared two notable parameter learning algorithms: the EM algorithm and the online EM algorithm using several real data sets from the UCI Machine Learning Repository. The results show that the batch learning approach leads consistently to the best parameter accuracy and diagnostic

accuracy but may take orders of magnitude longer run times than incremental learning. The incremental batch learning approach uses the least computation time but its accuracy is typically inferior to both batch learning and online learning. The online learning leads to lower accuracy typically worse than batch learning but requires only a modest computational and storage effort.

I performed an empirical evaluation of structural simplification of Bayesian networks by removing weak arcs. The results show that removing up to roughly 20 percent of the weakest arcs in a network has minimal effect on its classification accuracy. At the same time, structural simplification of networks leads to significant reduction of both the amount of memory taken by the clique tree and the amount of computation needed to perform inference.

One direction of future work would be to apply query-based diagnostics in a real help desk environments. The user interface of MARILYN has been improved several time for the better ease of use. MARILYN allows its users to enter free text, which the program interprets as the name of a variable. Currently, the system does not deal with the problem of redundant nodes or wrong data. If a user enters redundant nodes or wrong data, it is possible to fix the model directly in the database in what I called "expert mode." To prevent redundancy and common typos in the future, the system needs to provide to the user, through a web interface, additional information from domain experts and ability to edit the past cases. To integrate the system seamlessly into any existing help desk environment would required additional functionality. A good example would be the Help Management System from the School of Educational portal, which provides a user the search functionality for the past/unfinished tickets and the ticket reports.

BIBLIOGRAPHY

John M. Agosta, Thomas R. Gardos, and Marek J. Druzdzel. Query-based diagnostics. In Manfred Jaeger and Thomas D. Nielsen, editors, *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM–08)*, pages 1–8, Aalborg, Denmark, 2008.

John M. Agosta, Omar Zia Khan, and Pascal Poupart. Evaluation results for a query-based diagnostics application. In Petri Myllymaki, Teemu Roos, and Tommi Jaakkola, editors, *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM–10)*, pages 1–9, Helsinki, Finland, 2010.

Brent Boerlage. Link strength in Bayesian networks. Master's thesis, University of British Columbia, 1992.

John S. Breese, Robert P. Goldman, and Michael P. Wellman. Introduction to the special section on knowledge-based construction of probabilistic and decision models. *IEEE Transactions on Systems, Man and Cybernetics*, 24(11):1577–1579, 1994.

Olivier Cappe. Online Expectation-Maximisation. *ArXiv e-prints:1011.1745*, pages 1–20, 2010. URL http://arxiv.org/abs/1011.1745.

Leonardo Carbonara and Alastair Borrowman. A comparison of batch and incremental supervised learning algorithms. In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, PKDD '98, pages 264–272, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65068-7.

Arthur Choi and Adnan Darwiche. A variational approach for approximating Bayesian networks by edge deletion. In *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 80–89, Arlington, Virginia, 2006a. AUAI Press.

Arthur Choi and Adnan Darwiche. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1107–1114. AAAI Press, 2006b. ISBN 978-1-57735-281-5. URL http://dl.acm.org/citation.cfm?id=1597348.1597365.

Arthur Choi, Hei Chan, and Adnan Darwiche. On Bayesian network approximation by edge deletion. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 128–135, Arlington, Virginia, 2005. AUAI Press.

Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

Denver Dash and Marek Druzdzel. Robust independence testing for constraint-based learning of causal structure. In *Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 167–174, San Francisco, CA, 2003. Morgan Kaufmann.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, December 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248548.

F. Javier Díez. Parameter adjustment in Bayes networks. The generalized noisy-OR gate. In *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI–93)*, pages 99–105, Washington, D.C., 1993.

F. Javier Díez and Marek J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD–06–01, Universidad Nacional de Educacin a Distancia, Madrid, Spain, 2006.

Marek J. Druzdzel and Linda C. van der Gaag. Building probabilistic networks: "Where do the numbers come from?" Guest editors' introduction. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):481–486, July–August 2000.

Imme Ebert-Uphoff. Measuring connection strengths and link strengths in discrete Bayesian networks. Technical Report GT-IIC-07-01, Georgia Institute of Technology, 2007.

Imme Ebert-Uphoff. Tutorial on how to measure link strengths in discrete Bayesian networks. Technical Report GT-ME-2009-001, Georgia Institute of Technology, 2009.

A. Frank and A. Asuncion. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml, 2010. URL http://archive.ics.uci.edu/ml.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, November 1997. ISSN 0885-6125.

David Heckerman. Probabilistic similarity networks. *Networks*, 20(5):607–636, August 1990.

David Heckerman and John S. Breese. Decision-theoretic troubleshooting. *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans)*, 26(6):838–842, 1999.

Max Henrion. Some practical issues in constructing belief networks. In L.N. Kanal, T.S. Levitt, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 161–173. Elsevier Science Publishing Company, Inc., New York, N. Y., 1989.

G. Holmes, R. Kirkby, and D. Bainbridge. Batch incremental learning for mining data streams. Working paper, Department of Computer Science, University of Waikato. http://researchcommons.waikato.ac.nz/handle/10289/1749, 2004.

William Hsu, Ricky K Taira, Suzie El-Saden, Hooshang Kangarloo, and Alex AT Bui. Context-based electronic health record: toward patient specific healthcare. *IEEE Transactions on Information Technology in Biomedicine*, 16(2):228–234, 2012.

Pablo H. Ibargengoytia, Sunil Vadera, and L. Enrique Sucar. A probabilistic model for information and sensor validation. *The Computer Journal*, 49:113–126, 2006.

Don H. Johnson and Sinan Sinanovic. Symmetrizing the Kullback-Leibler distance. Technical report, IEEE Transactions on Information Theory, 2000.

Uffe Kjaerulff. Reduction of computational complexity in Bayesian networks through removal of weak dependencies. In *Proceedings of the Tenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 374–382, San Francisco, CA, 1994. Morgan Kaufmann.

Joost R Koiter. Visualizing inference in Bayesian networks. Master's thesis, Delft University of Technology, June 2006.

G. Kokolakis and PH. Nanopoulos. Bayesian multivariate micro-aggregation under the Hellinger's distance criterion. *Research in Official Statistics*, 4(1):117–126, 2001.

Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)*, pages 302–313, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

I. Kononenko, I. Bratko, and M. Kukar. Application of machine learning to medical diagnosis. In I. Michalski, R.S. Bratko and M. Kubat, editors, *Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications*. John Wiley & Sons, 1997.

Pieter Kraaijeveld. GeNIeRate: an interactive generator of diagnostic Bayesian network models. Master's thesis, Delft University of Technology, 2005.

Pieter Kraaijeveld and Marek J. Druzdzel. *GeNIeRate*: An interactive generator of diagnostic Bayesian network models. In *Working Notes of the 16th International Workshop on Principles of Diagnosis (DX–05)*, pages 175–180, Monterey, CA, June 1–3 2005.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

Lukasz A. Kurgan, Krzysztof J. Cios, Ryszard Tadeusiewicz, Marek R. Ogiela, and Lucy S. Goodenday. Knowledge discovery approach to automated cardiac SPECT diagnosis. *Artificial Intelligence in Medicine*, 23:149, 2001.

Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI–92)*, pages 223–228. MIT Press, 1992.

Kathryn Blackmond Laskey and Suzanne M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)*, pages 334–341, San Francisco, CA, 1997. Morgan Kaufmann Publishers.

Steffen L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19(2):191–201, February 1995. ISSN 0167-9473.

Percy Liang and Dan Klein. Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Tsai-Ching Lu, Marek J. Druzdzel, and Tze-Yun Leong. Causal mechanism-based model construction. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI–2000)*, pages 353–362, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

Anna Łupińska-Dubicka and Marek J. Druzdzel. A comparison of popular fertility awareness methods to a DBN model of the woman's monthly cycle. In *The Sixth European Workshop on Probabilistic Graphical Models (PGM 2012)*, pages 219–226, 19–21 September 2012.

Michael G. Madden. Evaluation of the performance of the Markov blanket Bayesian classifier algorithm. *CoRR*, cs.LG/0211003, 2002.

Suzanne M. Mahoney and Kathryn B. Laskey. Network engineering for complex belief networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 389–396, San Francisco, CA, 1996. Morgan Kaufmann Publishers.

B. Middleton, M.A. Shwe, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST–1/QMR knowledge base: II. Evaluation of diagnostic performance. *Methods of Information in Medicine*, 30(4):256–267, 1991.

Allen Newell and Paul S. Rosenbloom. Mechanisms of skill acquisition and the law of practice. In John R. Anderson, editor, *Cognitive Skill and Their Acquisition*, pages 1–55. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

A. E. Nicholson and N. Jitnah. Using mutual information to determine relevance in Bayesian networks. In *Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*. Springer, 1998.

Kristian Olesen, Steffen Lauritzen, and Finn Jensen. aHUGIN: a system creating adaptive causal probabilistic networks. In *Proceedings of the Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)*, pages 223–229, San Mateo, CA, 1992. Morgan Kaufmann.

Agnieszka Onisko. *Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders*. PhD thesis, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw, March 2003.

Agnieszka Oniśko and Marek J. Druzdzel. Impact of precision of Bayesian network parameters on accuracy of medical diagnostic systems. *Artificial Intelligence in Medicine*, 57(3): 197–206, March 2013.

Agnieszka Oniśko, Marek J. Druzdzel, and Hanna Wasyluk. Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

Judea Pearl and Thomas S. Verma. A theory of inferred causation. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR–91)*, pages 441–452, Cambridge, MA, 1991. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Avi Pfeffer, Daphne Koller, Brian Milch, and Ken T. Takusagawa. SPOOK: A system for probabilistic object-oriented knowledge representation. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–99)*, pages 541–550, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

Erik Pols. Marilyn a guided maintenance system that represents direct probabilistic influences among diagnostic knowledge. Technical report, School of Information Sciences, University of Pittsburgh, Pittsburgh, PA, April 2007.

Malcolm Pradhan, Max Henrion, Gregory Provan, Brendan del Favero, and Kurt Huang. The sensitivity of belief networks to imprecise probabilities: An experimental investigation. *Artificial Intelligence*, 85(1–2):363–397, August 1996.

Parot Ratnapinda and Marek J. Druzdzel. Passive construction of diagnostic decision models: An empirical evaluation. In *International Multiconference on Computer Science and Information Technology (IMCSIT–09)*, pages 601–607, 2009.

Parot Ratnapinda and Marek J. Druzdzel. Does query-based diagnostics work? In *Working Notes of the Eight Bayesian Modeling Applications Workshop, Special Theme: Knowledge Engineering, Part of the Annual Conference on Uncertainty in Artificial Intelligence (UAI–2011)*, pages 117–124, 14 July 2011. URL http://abnms.org/uai2011-apps-workshop/.

Parot Ratnapinda and Marek J. Druzdzel. An empirical comparison of Bayesian network parameter learning algorithms for continuous data streams. In *Recent Advances in Artificial Intelligence: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS–2013)*, pages 627–632, 2013.

Parot Ratnapinda and Marek J. Druzdzel. An empirical evaluation of costs and benefits of simplifying Bayesian networks by removingweak arcs. In *Recent Advances in Artificial Intelligence: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS–2014)*, page To Appear, 2014.

Silja Renooij. Bayesian network sensitivity to arc-removal. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM–2010)*, pages 233–240, 2010.

Masa-Aki Sato and Shin Ishii. On-line EM algorithm for the normalized gaussian network. *Neural Computation*, 12(2):407–432, 2000. ISSN 0899-7667.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer Verlag, New York, 1993.

Sampath Srinivas. A generalization of the noisy-OR model. In *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI–93)*, pages 208–215, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

Daphne Theijssen, Louis ten Bosch, Lou Boves, Bert Cranen, and Hans van Halteren. Choosing alternatives: using Bayesian networks and memory-based learning to study the dative alternation. *Corpus Linguistics and Linguistic Theory*, 9(2):227–262, May 2013. ISSN 1613-7035.

Peter A. Tucker, David Maier, Tim Sheard, and Leonidas Fegaras. Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):555–568, March 2003. ISSN 1041-4347.

Robert A. van Engelen. Approximating Bayesian Belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machinde Intelligence*, 19(8):916–920, August 1997. ISSN 0162-8828.

D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003. ISSN 0893-6080.