2017

An Introduction to the Theory and Applications of Bayesian Networks

Anant Jaitha
Claremont McKenna College

Claremont McKenna College

An Introduction to the Theory and Applications of Bayesian Networks

Submitted to

Professor Arthur H. Lee

by

Anant Vickram Jaitha

for

Senior Thesis

Spring 2017

April 24, 2017

Abstract

Bayesian networks are a means to study data. A Bayesian network gives structure to data by creating a graphical system to model the data. It then develops probability distributions over these variables. It explores variables in the problem space and examines the probability distributions related to those variables. It conducts statistical inference over those probability distributions to draw meaning from them. They are good means to explore a large set of data efficiently to make inferences. There are a number of real world applications that already exist and are being actively researched. This paper discusses the theory and applications of Bayesian networks.

Acknowledgments

I would like to thank Professor Arthur Lee for his guidance throughout my thesis. I would also like to thank my family and friends for supporting me through this project and through the ups and downs during my four years at CMC.

I would like to thank the various professors at Claremont McKenna College, Pomona College, and Harvey Mudd College, for helping me discover my interests and passions. I would like to thank them for helping me achieve my potential by pushing me to the limit to always learn and for fostering a healthy learning environment conducive for this continuous learning.

Contents

List of Figures

List of Tables

1 Introduction

Imagine conducting a study where some of the data is incomplete. Or imagine a situation where you decide to add a new dimension to your data but cannot go back and fill in these blanks in the past data because these new variables were not recorded in earlier data collection. In such a case, how do you reason about the old, incomplete data? Do you scrap it saying it is useless? Or do you try to reason about it to draw inferences from it? Even if those inferences aren't guaranteed to be perfect, what if you can draw meaning with a certain degree of confidence? This seems promising. Here, Bayesian networks can be helpful.

Bayesian networks is a subfield within artificial intelligence that is rapidly gaining popularity. It is an active area of research both in academic and industrial settings because its power in leveraging data is being recognized. A number of practical applications of Bayesian networks are being discovered in an industrial capacity. This is leading to a number of companies and researchers implementing Bayesian networks to address various questions they face.

Bayesian networks make use of graph theory to model the structure of a problem. The nodes along with the topology of the network encode the variables in the problem along with the relationships that hold among those variables in the problem space. It uses probability theory, more specifically Bayesian statistics and inferential statistics, to discover and encode the degrees to which those relationships hold in the problem space.

Bayesian networks are particularly helpful in "encoding uncertain expert knowledge in expert systems" (Heckerman, 2008). Data modelers and Bayesian network designers can create models to represent expert systems and allow the network to learn from the available data. It makes use of as much data as is available to provide its best estimates on a problem.

In this paper we explore the theory of Bayesian networks and discuss real world applications of this theory. In Section 2 we explore the theoretical concepts neces-

sary for understanding Bayesian networks; more specifically, we study the theory of Bayes' theorem and Bayesian inference followed by concepts of graph theory. Section 3 discusses what exactly a Bayesian network is, how it is generally used, and what situations it is especially useful in. Followed by this, Section 4 discusses some real world applications specifically in classification and learning problems. We look at some of the research that has been done and some applications of Bayesian networks developed by researchers in both these types of problems.

2 Theory

Bayesian networks make use of probability theory and graph theory to search through a state space and make decisions in uncertainty. It uses probability theory to guide it's search to find the goal states faster. Therefore, the main theoretical concepts that we need to understand to implement a Bayesian network fall under probability theory (specifically Bayes' theorem) and graph theory.

2.1 Bayes' Theorem and Bayesian Inference

2.1.1 Bayes' Theorem

Bayes' theorem was developed my Rev. Thomas Bayes, an 18th century mathematician and theologian. It is a means of calculating conditional probability distributions given a set of interacting variables. Bayes' theorem is expressed as

$$P(H|E,c) = \frac{P(H|c) \times P(E|H,c)}{P(E|c)} \qquad (1)$$

where H is the hypothesis, E is the known evidence, and c is the background context. In general, $P(A|B)$ is read as "the probability of A given B", where A is the dependent variable and B is the independent variable. Bayes' theorem allows us to calculate the probability of our hypothesis H given our evidence

and background. That is, given the context c and evidence E, we can know the probability of hypothesis H by using the conditional probability of H given c, conditional probability of E given H and c, and the probability of E given c using Equation (2). Here, the term $P(H|E,c)$ is the probability of H after considering the probability of E on c and is called the *posterior probability*. The term $P(H|c)$ refers to the probability of the hypothesis given the background context ignoring the evidence we have, and is called the *prior probability*. The term $P(E|H,c)$ is the probability of the evidence assuming the hypothesis and background context are true. This term is called the *likelihood*. The term $P(E|c)$ is the probability of the evidence given the context. It is a scaling factor to get the probability of the hypothesis from the conditions of the evidence E and background context. (Niedermayer, 2008)

The formula is often simplified to the form

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \tag{2}$$

This is an acceptable substitution since the background context often doesn't change and thus we can assume it to be constant throughout our analysis. (Wang and Vassileva, 2005)

Bayes' theorem is then further simplified using the inequality (Koch, 2006)

$$P(x|y) = \frac{P(x,y)}{P(y)} \tag{3}$$

to be rewritten as

$$P(H|E) = \frac{P(H,E)}{P(E)} \tag{4}$$

A great benefit of Bayes' theorem is that you can calculate the probability of event A given event B when you can more easily calculate the probability of event B given event A. That is, we can derive $P(A|B)$ from $P(B|A)$ using Equation (2) above.

To see the benefit of such an application of the rule, let's consider the following situation. When diagnosing whether a person has a cavity in his/her tooth, we consider a number of symptoms including whether the person has a tooth ache or not. Given a population of patients with tooth cavities, it might be easy to calculate the probability $P(tooth\ ache|cavity)$. However, the more useful and interesting question is what is $P(cavity|tooth\ ache)$? This can be done using Equation (2) above.

Often, we may know the values to a number of variables. That is, we have have more than one variable that is known. In such a case, we can use the chain rule. The chain rule is an important application of Bayes' theorem. It is expressed in Equation (5) below. (Niedermayer, 2008)

$$P(X_1, ...X_n|c) = \prod_{i=1}^{n} P(X_i|X_1, ...X_{i-1}, c) \qquad (5)$$

In a set of a number of variables, it is possible that some of the variables are disjoint or independent of each other if we control for the effects of another variable. Variables A and B are said to be conditionally independent if $P(A, B) = P(A)P(B)$. In general, it is said that variables A and B are independent given variable C if

$$P(A, B|C) = P(A|C)P(B|C) \qquad (6)$$

Alternatively, A and B are independent if $P(A|B, C) = P(A|C)$. (Fenton).

The topic of conditional independence is extremely important when talking about variables that affect a hypothesis variable. Reasoning about conditional independence relationships tends to simplify many computations in the usage of a Bayesian network.

2.2 Graphical Models

A graph is made of a set of nodes, and edges connecting pairs of those nodes. Each edge shows a relationship between two nodes. The edges may be directed or undirected, where a directed edge goes from a parent node N_p to a child node N_c. An undirected edge can simply be thought of as a special case of directed edges going in both directions between the two nodes.

Each edge also encodes some information about the transformation from the parent node to the child node and that information is often useful in the process of traversing the graph. In the case where such information is missing, it is assumed that the weights of all edges are the same or of a unit weight. (Stephenson, 2000)

Take the example of a simple undirected and unweighted graph in Figure 1. Here we show connections among three nodes A, B, & C such that nodes A and C are indirectly connected through node B. They are independent of each other given node B.

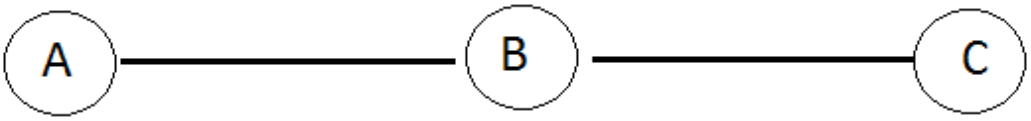


Figure 1: Undirected Unweighted Graph

Figure 1 above can be generalized to show the directions and weights as shown in Figure 2. The undirected graph can be replaced with edges going in both directions having a unit weight.

Properties of Graphs

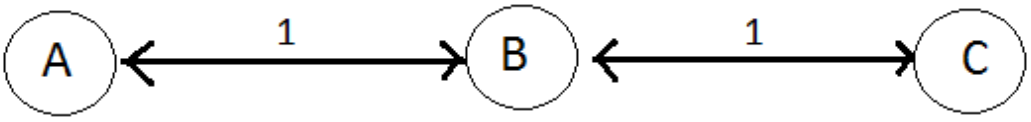


Figure 2: Directed Weighted Graph

In a graph, nodes u and v are said to be connected if there is a path from one to the other. In the examples above, the nodes A and C are connected because there was a path from one to the other even if not directly. In Figure 3 nodes A and B are disconnected because there is no possible path from one to the other. In the context of Bayesian Networks, paths help understand relationships between nodes.

Figure 3: Disconnected Nodes

Graphs start to get interesting when we look at directed graphs with varying weights. Figure 4 gives an example of a graph with various directed edges each with a weight associated to it. For example, going from node A to node B costs 4 units.

A graph is said to have a cycle if there is a way to get from one node back to itself through at least one edge. Figure 4 has a cycle since the nodes C, E, & F form a cycle.

Nodes in a graph each have in-degrees and out-degrees which means the number of nodes coming in to the node and going out of the node respectively. For example, node B in Figure 4 has an in-degree of 1 and an out-degree of 2. In the context of Bayesian networks, the in-degree of a node indicates how many variables affect the node and the out-degree indicates how many variables this node affects.

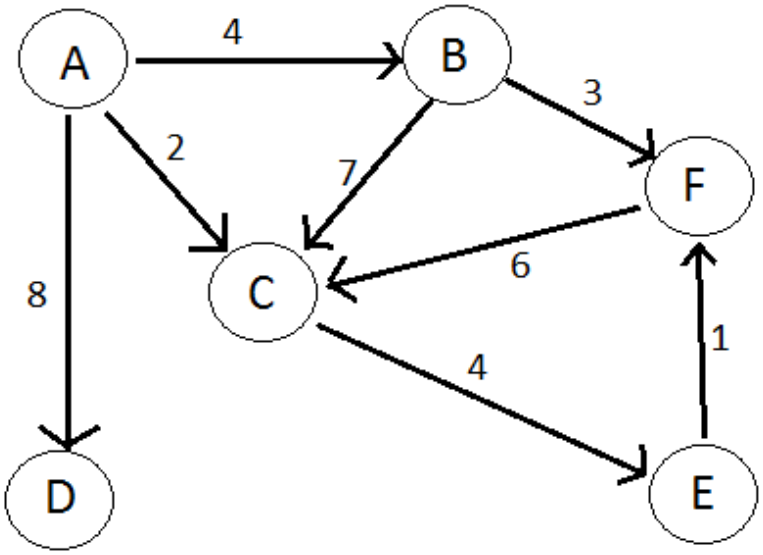


Figure 4: Directed Graph with Varying Edge Weights

3 Bayesian Networks

3.1 Introduction

Conditional probability is an extremely useful concept. A number of real world applications can easily be examined using conditional probability. In fact, people mentally analyze situations based on it all the time. For example, a medical doctor would listen to a person's symptoms and look at the most probable diseases based on the symptoms. A policy maker would analyze the most likely results of a policy he/she is considering.

A graphical model is extremely useful in connecting many such variables (for example, body temperature, redness of throat, medical history, and family medical history, etc.) with possible outcomes (for example, allergies, tuberculosis, diabetes, etc).

However, when looking at graphs with large branching factors, it might be impossible to explore all possible outcomes. A monitoring system for patients has 37 different components. In a simplified system where we assume that each system either gives an alert or not, we can have 2^{37} possible outcomes. Traversing all of these would be exponentially complex given standard traversal techniques

that don't make use of search heuristics. However, we can approximate solutions using search heuristics for traversing through our graph to prioritize more likely outcomes earlier (Example adapted from Niedermayer (2008)). Further, we can determine the probabilities of various events using the observed events and their probable effects. This is where a Bayesian network can be extremely useful.

3.2 Definition

A Bayesian network is a directed acyclic graph where each node has quantitative probabilistic information associated with it. Each node is associated to a random variable and arcs (edges) between itself and other nodes. The variables may be discrete or continuous (though continuous variables are often discretized). An edge going from node X to node Y means X is the parent of Y and it signifies a conditional relationship between them. Each node A_i has an associated conditional probability distribution represented by Equation (7).

$$P(A_i|parents(A_i)) \tag{7}$$

(Russell et al., 2009)

The set of nodes and edges - the topology of the network - shows the conditional (in)dependence relationships that hold in the domain. Each edge going from node X to node Y shows that variable X affects variable Y. That is, the value that variable X takes on affects the probability distribution of variable Y. Once a graph of such dependencies is established, we specify joint probability distributions for sets of dependent nodes to use the Bayesian network. Bayes' theorem is used here to calculate the probabilities of various different events.

If a node in the graph has an in-degree of 0, or it has no parents (no arcs or edges coming in to it), it only has a probability distribution for itself. If a node X has n parents, it has a conditional probability distribution for each parent Y_i.

That is, for each parent Y_i, we have a probability distribution table represented by $P(X|Y_i)$. (Uusitalo, 2007)

A Bayesian network is the combination of the topology (graph) and the conditional probabilities of the variables (nodes). These are together used to explore the effects of various variables on each other.

Determining the probabilities can be as simple as assigning them through joint probability distribution tables in some situations. However, for comprehensive Bayesian networks, these probabilities are adapted (through learning) as more data is collected. The learning provides improved knowledge by combining prior knowledge with data. (Heckerman, 2008) More on this will be discussed in Section 4.2.

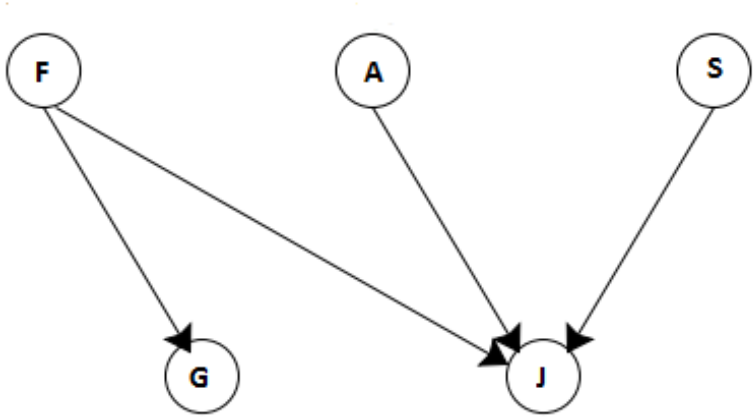
3.3 Inference in a Bayesian Network

Inference is the task of computing the probabilities of unknown events in a Bayesian network given the data on known events. Inference is fundamental in determining the most probable values of the variables and then drawing conclusions from the values. (Stephenson, 2000)

When certain events are known, we know more about the other possible events than if nothing was known. Thus, we can use this information to revise our knowledge on how probable other events are given the knowledge we have. The process of inference seeks to achieve exactly this goal of refining knowledge based on known information.

Consider the example Bayesian network in Figure 5. Our nodes represent variables for whether the transaction was fraudulent (F), whether there was a gas (G) purchase in the last 24 hours, whether there was a jewelery (J) purchase in the last 24 hours, the person's sex (S), and the person's Age (A). Here, we see a number of conditional dependencies in our graph. Consequently, we also see conditional independencies. For example. F and A are independent given J, G

Figure 5: Example Bayesian Network adapted from Heckerman (2008)



and J are independent given F, etc.

The interesting question in this situation is what's the probability of fraud given the other information? That is, it's easier to observe the other 4 variables then it is to observe whether there is going to be a fraudulent transaction or not. Additionally, we can use past data on fraudulent transactions where these variables were recorded to give estimates as to whether a transaction is fraudulent or not. We use the general formula adapted from Equation (3) in the form

$$P(f|j,g,s,a) = \frac{P(j,g,s,a,f)}{P(j,g,s,a)} \tag{8}$$

to get a more realistic look at the probability of event f when we know the outcomes g, s, a, and j. (Stephenson, 2000)

Given this knowledge, we can simplify Equation (8) using the knowledge of these independencies and plugging them in to Equation (6) and then using the chain rule in Equation (5) to get the equation

$$P(f|j,g,s,a) = \frac{P(j|s,a,f) \times P(g|f) \times P(f)}{\sum_{f_i} P(j|s,a,f_i) \times P(g|f_i) \times P(f_i)} \quad (9)$$

(Stephenson, 2000)

These individual values are much easier to compute than Equation (8). Thus, we can more easily compute a more accurate probability of fraud given the observable knowledge.

3.4 Usefulness of Bayesian Networks

So now that we know what Bayesian networks are, let's look at why they are useful. The rest of this section concentrates on the advantages of Bayesian networks over other state space search solutions. While there is the overhead of computing additional information, the benefit from the speed up is worth the additional computation.

3.4.1 Suitable for Small and Incomplete Datasets

In Bayesian networks, there's no such thing as "too little data". While more data is better, Bayesian networks work with as much data as is available to give pretty accurate results. Moreover, with each iteration it learns more and refines its model to give better results the next time. Bayesian networks are essentially mathematical models that are represented using graph concepts to make it easier to analyze, implement, and understand. (Uusitalo, 2007)

The conditional probabilities are estimated using various techniques and are used to give fairly accurate probabilities to various events. All it requires is for the model to be known. Since the model can be built to be flexible with the volume of data and weightage of old vs new data, it can be made as flexible as needed.

For example, we could have a very flexible model that weighs recent data more than past data. In that case, we might be trying to account for the changing background context for the experiments. On the other hand, we could weigh past data more than new data. This aims to establish an estimate in the beginning and then refine it with more data. However, in either case, they both can work with small and incomplete datasets to produce accurate results. (Uusitalo, 2007)

Uusitalo (2007) talks about how Bayesian networks are extremely useful in modelling environmental changes. Environmental data is often sparse and incomplete. For example, the incomplete data might miss special events or certain spans of time. Bayesian networks can then be used to work with this incomplete data to still produce meaningful results with mathematical reasoning and precision.

3.4.2 Structural Learning Possible

An extremely interesting use of Bayesian networks comes when discussing how to learn the structure of the model apart from just the probabilities they encode. Simple versions of Bayesian networks have experts in the field help establish the structure and that structure stays constant. While these may refine the conditional probability distributions, they do not create new dependencies or independencies from the data.

However, Bayesian networks can be made flexible enough to allow them to alter the graph structure as they learn from the data. This problem is a very tough one and usually algorithms aim to approximate such structures rather than compute the optimal version as computing the optimal version is very tough to implement on large networks. (Uusitalo, 2007)

Uusitalo (2007) claims there are two main approaches to this kind of structured learning - the Bayesian approach and the constraint-satisfaction based approach. the Bayesian approach requires the user/expert to first input a model with his/her knowledge along with the user's confidence in the model. The algorithm then uses the data to find the best fitting model. On the other hand,

the constraint-satisfaction approach does not need any expert knowledge or user input. It searches for conditional dependencies and independencies between pairs of variables and builds the structure using this knowledge that it establishes.

3.4.3 Combining Different Sources of Knowledge

A great advantage of Bayesian networks is that it allows us to combine prior knowledge with new data. That is, we can update our prior knowledge with new information. A benefit of this is that it allows us to combine data from different sources together. The prior knowledge obtained from one source can be combined with data from the new source to create new inferences that might have been lost or left out in the earlier source. This makes the learning process more scientific by accounting for the biases that different data sources may contain but which are not accounted for. This also aims to free the data from assumptions that are made by different data sources. (Uusitalo, 2007)

Since the models in Bayesian networks are weighing the data from different sources equally, they combine data while preserving the different degrees of accuracy that exist in various data sources. In addition, they make computation easier by combining expert qualitative knowledge with quantitative data to produce a mathematically accurate result. (Uusitalo, 2007)

In artificial intelligence, Bayesian networks are used in a variety of diverse ways in conjunction with other techniques. For example, it can be used with Markov chain decision problems, Monte-Carlo methods, and other techniques to gather information and test experiments without the necessity of vast comprehensive data. (Uusitalo, 2007)

3.4.4 Explicit treatment of uncertainty and support for decision analysis

It is extremely easy to encode uncertainty and freedom for action into a Bayesian network. For example, let's say a model for predicting industrial decisions is

established. However, we know that humans aren't always rational. The decision maker might have short/long term goals in mind while making decisions and these decisions might not align with the 'most rational' decision to make according to our model. Here, we can add a variable for a degree of randomness in decision making such that we can better understand how the person might actually interact. In the process, we can account for the outcomes from the actions that actually do take place as opposed to only accounting for those that the model considers most rational. Expected value outcomes can be analyzed to understand how the industry might actually turn out and get a more realistic picture as to the world the model is describing. (Uusitalo, 2007)

Another way to look at this is that when we don't know the certainty of outcomes, we can encode a degree of randomness that we cannot predict. For example, unforeseen circumstances and unexpected variables can be accounted for by accepting that there might be some deviation from our model that we have not accounted for. This is particularly useful for studying macro systems like environments, climate, economies, etc.

3.4.5 Fast Responses

Once a model is compiled, we can get very quick results by using the already established conditional probability distribution tables. The results requested can be provided by using the values in the tables along with various formulas like the chain rule and the conditional independence relationships. Thus, we do not need to get large clusters of computers or run highly specialized scripts to get results. This is extremely useful in understanding the results of a simulation and in communicating the results with others. Essentially, it is akin to running a guided depth first search as opposed to running a breadth first search where it benefits from the space advantages of depth first search. (Uusitalo, 2007)

4 Applications

We have seen how Bayesian networks are useful in general situations. They help answer questions with limited data quickly and efficiently. Now, we will look at how Bayesian networks are specifically helpful in problems relating to classification and learning, and in the process we'll explore some past academic work done in these fields.

4.1 Classification

Classification refers to the job of assigning *class* labels to various instances of a problem. The various class labels are distinct categories that the instances can be grouped under. The label is assigned using various attributes which help distinguish among different classes. In general, a Bayesian classifier works by learning from training data. It learns the conditional probability of each attribute A_i for each class C. During classification/testing, the classifier then applies Bayes' theorem to compute the probabilities of each class C_j given the attributes $A_1, A_2, \ldots, A_n$. It then predicts the class C_{MAX} with the highest posterior probability. (Friedman et al., 1997)

The simplest form of a Bayesian classifier is a Naive Bayesian classifier. A Naive Bayesian classifier assumes that all attributes are independent of each other given the classification variable. Naive Bayesian classifiers are extremely efficient when this assumption is applied. In other words, A_i is independent of A_j for all $i \neq j$ given the class C in question. This reduces the branching factor of our search tree from being exponential to being linear. An example of a Naive Bayesian network is shown in Figure 6. (Friedman et al., 1997)

While this assumption is not strictly accurate, it is true for most Bayesian networks. Designers of the network choose variables that do not affect each other so that they cover as many different variables that affect the class labels without having repeated information. This is done to remove the redundancy in the

variables.

A more rigorous explanation goes as follows - since a Bayesian network is a directed acyclic graph, each vertex represents a variable, and each edge represents a dependence relationship, the network ensures the following statement - "each variable is independent of its nondescendents in the graph given its parents." This statement and conditional independence above both hold true when we represent the Bayesian network as shown in Figure 6. (Friedman et al., 1997)

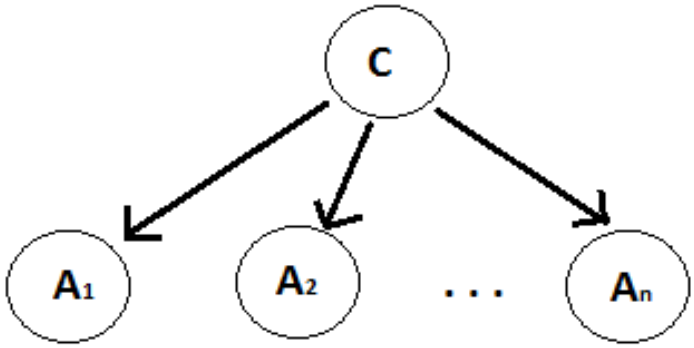


Figure 6: Naive Bayesian Graph adapted from Friedman et al. (1997)

Now that we have seen how in general Bayesian networks work on classification, let's see specific examples of how they work on some real world problems.

4.1.1 Medical Diagnosis

In medical diagnosis, a Bayesian network can be used to suggest possible diseases based on the observed symptoms. That is, a doctor can enter the observed symptoms into the program and the program then takes the inputs and computes the probabilities of a variety of diseases given the symptoms. Doctors simulate approximate conditional probabilities mentally and follow this process in their day to day work so here we can make this work automated through the use of Bayesian networks. Here, we have an example of a system where the initial probability distributions for the nodes do not need any kind of learning. It can be done using a medical professional's quantitative estimates or using available statistics on the

relationships among the symptoms and the diseases (Stephenson, 2000).

The challenge arises when these quantitative estimates and statistics are incomplete or not completely accurate. Bayesian networks can also help in this case to provide estimates while dealing with the incomplete and inaccurate data.

The challenge comes down to constructing conditional probability tables from partial statistical data for the nodes and edges in the network. Further, the problem is often that the data we need is not in the form most helpful for our model. For example, if we have two diseases D_1 and D_2, that both may cause symptom S, we might get statistical information on $P(S|D_1)$ and on $P(S|D_2)$ but these aren't as useful as $P(S|D_1, D_2)$, $P(D_1|S)$, and $P(D_2|S)$. Generalizing this to some number of n diseases makes the problem a lot tougher. If we consider each disease variable D_i as a binary variable, we would have 2^n entries in our conditional probability table but filling this out can be extremely tedious and computationally difficult. To overcome this, it might help to plug the problem into a Bayesian network and introduce additional constraints in the form of biases in favor or against some diseases. These biases are dependent on the degree to which the individual diseases affect or don't affect the symptom or set of symptoms in question. (Nikovski, 2000)

These biases are also called sensitivities and they encode the probability of $P(S|D)$ and $P(S'|D')$ in each disease node. Thus, we have reduced the problem from a factor of 2^n to $2n$. Now, we can use Bayesian network evaluation to determine exactly how probable a disease is given the sensitivities and relationships. (Nikovski, 2000)

An added benefit to using the sensitivity information is that each disease is independent of other diseases in the model. Rather than obtaining the most probable disease by exploring 2^n possibilities, we are able to look at the probability of each disease irrespective of others. In essence, we are looking at a system of **OR**s such that when we look at $P(S|D_i)$, we are essentially looking at the $P(S|D'_1, D'_2, \ldots, D_i, \ldots, D'_n)$ for each value of $i = 1 \ldots n$. (Nikovski, 2000)

Going back to the example above of diseases D_1 and D_2 both causing symptom S, we can look at the interesting question we thought of as, what is the probability of each disease given the observed symptoms. Given the probability distributions of $P(S|D_1)$ and $P(S|D_2)$, we can use Bayes' Rule to compute the values of $P(D_1|S)$ and $P(D_2|S)$. This is more useful to the doctor as now the doctor can identify the disease with a degree of confidence and treat for the disease.

A point to consider is that since this is a human made model and is limited by the knowledge of those creating the model, it is possible that there are diseases not accounted for which could cause those symptoms. Additionally, since this is a probabilistic approach, it is possible that there are certain false positives and false negatives. Further, some results can be too ambiguous to understand. For example, a result of a 50% probability of a given disease can be difficult to interpret. In such a case it might be necessary to get more information through additional tests.

All diseases aren't as binary as we just described. Further, each disease can cause multiple symptoms and some symptoms can change the likelihood of other symptoms. Even modeling such relationships is an easy task for a Bayesian network. Consider the following situation - two of the symptoms of the disease Cardiac Tamponade are breathlessness and rapid breathing. While both the symptoms have their own probabilities given the presence of the disease, the presence of rapid breathing becomes more likely if there is breathlessness. In such a case, the two symptoms are not conditionally independent of each other. One increases the probability of the other and so our network can account for this by adding an edge to the network from the causal symptom to the caused one. (Example adopted from Nikovski (2000))

In the case of unmodeled diseases, we overcome the problem by including a leak variable that accounts for all possible unaccounted diseases or unaccounted variables in general. (Nikovski, 2000). Further, all the diseases should be verified using more conclusive tests since the effects of wrong treatment can be detrimental.

However, at least using this model we can narrow down on some of the possibilities and know where to focus our attention.

4.1.2 Junk Email Classification

In our modern world we deal with email every day. Junk email (often called spam) annoys us daily as well. However, most email clients spend a lot of time and energy trying to find automated ways to minimize the amount of spam we receive. To do this, they check the emails we receive to predict what is spam and what is not. While there is a degree of error in their predictions, the classifiers can save humans a lot of time if they can achieve a fair degree of accuracy in their predictions.

In the process of detecting spam vs. not spam email, we want to look at certain properties of the individual email and also the person's interactions with the sender and with similar emails. Additionally, we want to see how the email matches up with other emails people receive. That is, we look at the content of the email, the formatting style, details about the sender, and some other properties as listed in the work of Sahami et al. and further discussed in this section.

Sahami et al. believed that in the context of email filtering, it is extremely important to represent emails using feature vectors so that Bayesian classifiers can be used directly. Each feature represents different information about emails and has a separate probability distribution describing the likelihood of an email being junk. In such a distribution, it is possible to allow each word ever seen in emails to be its own feature. While no one feature is a clear determinant of whether an email is junk or not, we can develop probabilities for each of those hypotheses and make decisions based on those.

For example, the presence of certain words can change the probability of an email being spam. The occurrence of certain words like **FREE** and **MONEY** seem to indicate an email might be spam. Phrases like **CLICK HERE** would indicate a phishing scam specially when the link in it is not verifiable to be safe. However, some features that make emails less likely to be spam are words like the

person's name. If the recipient's name is clearly written, there is a slightly lower probability of the email being spam.

If the person received the email directly as opposed to from a mailing list, it is more likely to be safe. Most spam is sent out through mailing lists. This trend is slowly changing given how marketing folk understand this trend in spam filters and try to send emails individually by automating the process.

Another feature worth noting is the domain of the sender. Almost never is spam sent from a `.edu` email. Similarly spam email is more likely to come from a domain with a number of garbled characters. Spam is less likely to come from an email which is in the person's address book and if the person has sent emails to it in the past. Junk email usually does not come from known email addresses and does not come from email addresses the user has interacted with in the past.

The presence of attached documents is an indicator that an email is not spam. Most spam does not come with attached documents because it makes it tougher for the marketers to blast the user with advertisement information. However, phishing scams might try to get you to download a virus through attachments or by directing you to a website that makes you download malware onto your computer so this is a downside that email clients try to overcome by scanning the attachments you receive.

The presence of a number of non-alphanumeric characters is also a sign towards the email being spam. For example, if there are a number of **$** signs in an email, it might be an indication of spam. Additionally, if there are a lot of **!** marks, it might be more likely to be spam.

Apart from the ones mentioned above, Sahami et al. devised a number of other domain specific features to help them classify emails as junk or not junk. Their classifier would examine all the encoded features in an incoming email and mark it as spam or not and then forward it to the appropriate folder in the person's inbox. Then they conducted some tests. These tests were conducted in phases adding a little more information each time to make the Bayesian classifier a little

more complex.

In their tests, they captured the features that had the highest impact on the problem and used them in the test model. This came from them entering a lot of training examples before testing on other examples. The first phase of testing involved only using the word features. Then they tried it again with words and phrases. At the end they added non-textual features to the model.

Tests revealed a great deal of accuracy in the way the features predicted whether an email is spam or not. The Bayesian classifier learnt from a number of training examples and it was able to apply its learning to the test data with a high degree of accuracy when using all the features they devised. With each added set of features the precision and recall rate improved. The precision refers to the percent of emails classified as junk (or legitimate) actually being junk (or legitimate). The recall rate refers to the percent of junk (or legitimate) messages in the test set that are actually classified as junk (or legitimate). In fact, when using the words, phrases, and non-textual features, all the emails classified as junk were in fact junk.

The Bayesian classifier is an ongoing classifier. It is continuously learning from data in the way that when a user classifies an email as junk or not, it updates its beliefs. That is, when a user notices a mistake in the classifier's label of an email and corrects the label, the classifier learns new information that it can work with. Further, new interactions of the user with the email client gives the classifier more data to base its decisions on.

Thus, the work of Sahami et al. showed that it is possible to automatically filter emails to eliminate a large proportion of junk email. This saves people hours of time every year that they would have otherwise wasted going through such junk. Further, when we consider the effects of junk email in the form of pornographic material, we could also prevent the adverse effects that such emails could bring with it.

4.1.3 Peer-to-Peer Networks

Peer-to-peer (P2P) networks are based on a trust model. P2P networks work because of the mutual trust of all those in the network. However, often malicious peers join in and disrupt this trust model. Since there is no central authority dictating terms in a P2P network, there is no accountability for the nodes in the network and therefore malicious nodes have no deterrent from behaving maliciously. In fact, the benefits from malicious activity could bring more of an incentive to behave maliciously. Thus, computers tend to communicate among each other to develop opinions of trust among others in the network. The work of Wang and Vassileva (2005) proposes an interesting Bayesian network-based trust model for P2P communication.

Each node in the network plays two roles, one as a file user (receiving files from others) and another as a file provider. As a file user, it has trust value distributions on how much it trusts every other node to supply it with a file and also on how much it trusts each other node on giving suggestions on other nodes as file providers. As a file provider, other nodes have trust values on how much they trust it to provide files and provide recommendations on other file providers. The trust models rely on two different aspects - how much a node can trust another node to give honest data and how much a node can trust another node to give useful recommendations on other nodes it is unfamiliar with.

Trusting a file provider to give legitimate and high quality files is an interesting idea since it is trying to quantify trust in a peer in a digital setting. Even in humans it is difficult to quantify trust but the concept is made possible by the idea that the numbers aren't as important as the relative difference between them. That is, a trust value of 0.78 means nothing till it is compared to another with a value of 0.43.

Different nodes in a network may have different preferences. A node's trust in file providers might be affected by a number of variables based on the priority

of the node receiving and using the file. For example, maybe a node in question finds the quality of the file more important than the speed at which it is shared. Thus, it will prefer a peer that consistently shares high quality files more than one that has faster sharing speed but less of a guarantee of quality of the files. Similarly, Wang and Vassileva (2005) describe three main features in such a P2P network - file quality, file type, and download speed. That is, Wang and Vassileva (2005) construct a Bayesian network with 4 variables for each node in the network - trust, file quality, file type, and download speed. It has edges going from trust to the other three to make it easy to model the relationship and use Bayes' theorem to calculate the probability of trust given values for the probability of the desired file quality, file type, and download speed. This model gives us a flexible model to solve the problem by use of Bayesian statistics.

Here we have an example of a Naive Bayesian network consisting of a single root node and several leaf nodes. The leaf nodes are the variables that we measure and we are able to assume safely that all the variables are independent of each other. We have seen this in earlier examples but in many of those cases it was a simplification we assumed knowing that it might not be completely true. However, in this case we can see how this assumption is valid since none of these three variables would have a significant impact on the others and can thus be modeled to be independent of one another.

The leaf nodes help an agent in the P2P network derive conditional probability distributions for other file providers in the network based on the kind of files they provide, at what speed they provide them, and at what quality they provide them. The data for the trust model is stored with the type of interactions it has had with the other nodes.

As an example, let's assume that an agent a only cares about the file type and how satisfying the interaction was. Let's also assume that there are 3 types of files - music, movies, and software - that are shared in the P2P network. For example, agent a might build a conditional probability table for file provider b as shown in

Table 1: The CPT of node b (adapted from Wang and Vassileva (2005))

	T=1	T=0
Music	p = (FT = "Music"|T = 1)	p = (FT = "Music"|T = 0)
Movies	p = (FT = "Movies"|T = 1)	p = (FT = "Movies"|T = 0)
Software	p = (FT = "Software"|T = 1)	p = (FT = "Software"|T = 0)

Table 1. Here, the value $T = 0$ refers to an unsatisfying transaction while $T = 1$ refers to a satisfying transaction with node b. Therefore, given the type of file node a is trying to receive, it can compute the probability of a satisfying transaction with node b given the file type using Bayes' Rule. It can then compute the similar values for other potential file providers and then make a decision on which file provider is the most trustworthy to go with. At the same time, it will look at other file providers that it has not interacted with in the past (and thus does not have trust value distributions on) to see how its peers recommend this new file provider and use that as a comparable to decide if it wants to interact with this new file provider or not. More on this is discussed later on in this section.

After the agent concludes an interaction, it needs to update its trust model. To do this, it rates the transaction as satisfying or not. It decides how to classify the download speed, how to classify the file quality, and then evaluates whether the interaction was satisfying or not. Based on the chosen learning rate, it updates its probability distributions such that a part of the probability is from the old knowledge and a part is from the newly acquired information. The learning rate determines whether this favors the new data more than the old data or vice-versa.

As discussed earlier, a file provider can give other agents recommendations on the trustworthiness of another file provider. That is, if a file user is not sure about the trustworthiness of a file provider (maybe due to lack of a previous interaction), it can ask other trusted file providers for recommendations on the new file provider.

For example, if an agent a has the ability to acquire a movie from file provider b but has never interacted with this provider before, it can ask its other trusted providers for recommendations. These trusted providers give recommendations

based on $P(T = 1|FT = \text{“}Movie\text{”}, FQ = \text{“}High\text{”}, DS = \text{“}Fast\text{”})$. Based on how much agent a trusts these individual providers to give suggestions, it will weight their suggestions and aggregate them. The value is finally normalized based on the sum of the weights so that we can get a fair estimate to be able to compare it to other file providers that we already have trust models on. (Wang and Vassileva, 2005)

Using Bayesian networks, we can now create a flexible representation of trust using three variables. While trust is a subjective emotion, they were able to find a way to identify certain easily identifiable variables and encode a decision making process based on comparisons using Bayesian networks.

4.2 Learning

Bayesian networks allow us to learn about causal relationships. Learning using Bayesian networks is particularly useful when exploring and acquiring new data. It allows us to test hypotheses using data. One major benefit of Bayesian networks in this context is that we can test to see if the effect of a new feature is desirable or not. For example, a company can test to see if a new advertisement campaign has had a significant impact on the sales by making use of Bayesian statistics to compare past and new data. Learning through the features of such models can help the company understand what aspects of an advertisement campaign have the most positive impact per Dollar spent. (Heckerman, 2008)

Bayesian networks are also extremely helpful in combining prior domain knowledge with data. That is, it can either use data to verify the validity of the prior knowledge or it can correct prior knowledge by learning from the data and refining its own beliefs about the model. (Heckerman, 2008)

It is also possible to learn the topology and the probability distributions of a Bayesian network. This allows us to learn the relationships and the degrees of these relationships.

When studying how learning takes place, we can divide all situations into four categories based on whether there is a known structure or not and whether there is full observability or not.

When the structure of the network is known and the data is fully observable, we have an easy way to determine the probability distributions. We can calculate

$$P(X = x_i|Y = y_i) = \frac{count\ of\ (x_i, y_i)}{count\ of\ y_i} \quad (10)$$

(Stephenson, 2000)

An alternative is to use Bayesian statistics to compute $P(X = x_i|Y = y_i)$ using Equation 3.

On the other hand, if the data is only partially observable (only a subset of the real data is available for the network to develop a probability distribution from) we cannot get exact probability values. However, we can estimate the values in ways similar to when we have full observability. (Stephenson, 2000)

These get more interesting when the network updates its probabilities when it sees more data during testing or after deployment. For example, if a network has seen five examples of $Y = 1$ and only one of those five had $X = 0$, it would set the $P(X = 0|Y = 1) = 1/5$. However, if it sees a new sample of $X = 0$ & $Y = 1$, it would update that value to now be $P(X = 0|Y = 1) = 2/6$. When converging to infinity, this value tends to reflect the true probability of the values.

Let's now look at a general situation of a network with unknown structure and full observability of data. Essentially this means that the network does not know the relationships among the variables but has all the possible data on it. Here, we can take two approaches. We could create a metric to compare potential structures to each other to decide which one is a better fit to the data. On the other hand, we can devise a search algorithm to find the best structure. While both

are valid solutions and are used in industry, searching for a structure might be more practical when the number of variables increase since the evaluation method might run into the issue of an exponentially growing set of structures to evaluate, specially when the evaluation method needs to evaluate all possible topologies. However, this could be overcome by an expert suggesting a set of probable structures and then evaluating only those ones. Even though the search problem is NP-hard, there are polynomial time algorithms to estimate the structures fairly close to optimally. (Stephenson, 2000)

The greatest challenge comes when the structure is unknown and when the data is only partially observable. While we can follow a process of searching for a structure and evaluating different structures, this might not be robust enough when we are unfamiliar with the different variables in the network. Therefore, we can combine this structure defining process with the following system. We add a variable representing a hidden variable. Then, we find the best structure for the given set of nodes and continue adding such nodes till the addition of a node makes the network get worse. (Stephenson, 2000)

Very often, learning and forecasting go hand in hand where we predict an outcome and learn from the real outcome when it takes place. In the next few subsections, we will see some real world applications of Bayesian networks in learning and how it applies learning through forecasting.

4.2.1 Environment Modeling

Marcot (2012) shows us a few different ways that Bayesian networks can be used to model animal population trends. He describes how he can predict the populations of polar bears. Here we have examples of known topology (or at least a topology being set) and partial observability of data.

Let us consider the case of studying polar bear populations. To begin, let's look at the network that he used. There were a total of 36 nodes and 44 edges among them. 17 of the 36 were inputs on the various environmental stressors (factors that

affect their habitat, prey and predator populations, etc.) affecting the polar bear population. They assumed that the 17 inputs were normally distributed variables and discovered from their data that 6 of them accounted for 92% of the sensitivity in the model.

They also ran these analyses by changing how the input variables were initialized. Initially, they were normal distributions but then they changed it such that various sets of them were set to best or worst case scenarios. By combining the analyses from these different scenarios, they were able to isolate the effects of the different variables on the polar bear population. They were able to determine that controllable features like sea traffic adversely affect polar bear populations but far less than larger scale stressors like climate change. While these are both man-made causes of adverse effects on polar bears, it is far tougher to control climate change than something like sea traffic.

An approach he could have taken is by not setting the topology (or setting a default one that is open to flexibility) so he can learn the impacts and the effects of each variable on each other as well as on polar bears. By allowing the topology to be flexible, he could have discovered relationships among variables he did not know previously existed. A slight modification to this method could be that he could try out a few different network structures and learn the probability distributions in each of them. Then, he can compare how well they all fit the data and conclude on the most appropriate relationships among variables.

An application of such an analysis is that such a network can be the part of a larger network where polar bear population is a variable along with a number of other variables. For example, consider modeling the future of an ecosystem where polar bear population is one of the many things that researchers are trying to study. This can serve as one component of the larger network that is the ecosystem. A number of smaller networks can be part of this larger one and each affecting each other and the network as a whole. For example, one of the variables in the polar bear population network was the availability of prey. What if that

variable was replaced with a Bayesian network predicting the population of prey? In fact, it will be able to account for the variation in the population of its prey given environmental factors that affect it.

In the process of forecasting the polar bear population, we can also update these beliefs as time passes as we gather more data to update our beliefs on what affects their population and how.

There are a number of models simulating environmental variables to predict outcomes based on current and expected trends. Uusitalo (2007) talks of a model built to look at ecosystem responses to different Baltic cod management practices. Based on their Bayesian network model, they were able to run multiple Monte Carlo simulations to look at the effects of various management practices. They modeled it after a fishery near the Great Barrier Reef and were able to learn a great amount about the effects of the various options they had on Baltic cod management.

4.2.2 Traffic Flow Forecasting

A number of commercial and government applications consider traffic prediction as an important issue to deal with. Here we consider the model created by Sun et al. (2006) as an example of how we can use Bayesian networks to forecast traffic and learn from it.

Here, each node is a road link and each edge shows the direction of traffic flow. Thus, we have a cause node and an effect node such that each road that leads to another is the cause node to the one it leads to. Further, each node also takes other inputs like the time of day, special events in the area, etc.

We follow the same general process as described above where we first train the model on some data such that it can learn the general process. Then, we let it run for some time on test data to allow it to learn the true probability distributions over time. When it is set into the real world, it constantly learns from its predictions and the real data.

For example, consider the case as follows - let's say we classify all traffic as either light, moderate, or heavy. Let's say at time t we forecast the traffic to be moderate. However, in reality it ends up being light. In such a situation, we will update the probability distribution table to reflect this correction given the input variables it received.

After a period of time, we would have collected a large volume of data and updated a lot of our estimates from our initial training set. While this will improve our forecasts, it will also help us understand the determinants of traffic along with the degree of each one's impact.

5 Conclusion

In this paper, we have seen the key concepts necessary in understanding Bayesian network theory. We have understood what a Bayesian network is and explored the situations where it is useful. This was followed by us examining how some important problems in research and industry, namely classification problems and learning problems, can be effectively tackled by Bayesian networks because of Bayesian statistics.

This paper has been a broad overview of the field; however, a lot more is available to be explored. We can extensively study the algorithms used in Bayesian network inference and traversal. Further, a lot of research is being done on fast algorithms to traverse these networks and conduct fast inference on the variables.

We have seen how Bayesian networks are useful to so many different kinds of people - environmental scientists, policy makers, medical professionals and scientists, network modellers, and general researchers among others. It's an extremely versatile tool in the toolbox of any industry and has great potential of being useful in a variety of scenarios; people just need to be willing to take the time to implement the network and give it the data to work off of. Further, a number of industries that are still unaware of the potential can gain immensely from Bayesian

networks.

However, we need to acknowledge that Bayesian networks are extremely complex and a lot of work goes behind implementing them. There are some downsides to Bayesian networks. Firstly, they force discretization of continuous variables. Secondly, it is difficult to model the expert knowledge in a field that is necessary to construct the model. Additionally, the computing power necessary to store and process the data in a continuous basis might be immense given the challenges it is trying to face. (Uusitalo, 2007)

However, it seems as though if someone has the resources to support the construction of a Bayesian network, they have the resources to find the expert knowledge and implementers to ensure we can get over all these challenges. This does create a kind of a barrier for entry but that might be an area for further study. Like a number of other technologies out in the market, it might only be time till researchers are able to provide this implementation power to the average person.

In summary, Bayesian networks are extremely useful in a number of contexts. We studied how it is useful in classification and learning problems. However, a number of other applications also exist and are being constantly discovered for this technique of reasoning about data. The potential of using data is being discovered and Bayesian networks are a good way to unleash this power and make what was otherwise useless data useful.

Appendix

List of Equations

1. $P(H|E,c) = \frac{P(H|c) \times P(E|H,c)}{P(E|c)}$
2. $P(H|E) = \frac{P(E|H) \times P(H)}{P(E)}$
3. $P(x|y) = \frac{P(x,y)}{P(y)}$

4. $P(H|E) = \frac{P(H,E)}{P(E)}$

5. $P(X_1, ...X_n|c) = \prod_{i=1}^{n} P(X_i|X_1, ...X_{i-1}, c)$

6. $P(A, B|C) = P(A|C)P(B|C)$

7. $P(A_i|parents(A_i))$

8. $P(f|j, g, s, a) = \frac{P(j,g,s,a,f)}{P(j,g,s,a)}$

9. $P(f|j, g, s, a) = \frac{P(j|s,a,f)\times P(g|f)\times P(f)}{\sum_{f_i} P(j|s,a,f_i)\times P(g|f_i)\times P(f_i)}$

10. $P(X = x_i|Y = y_i) = \frac{count\ of\ (x_i,y_i)}{count\ of\ y_i}$

References

Cheng, J. and Greiner, R. (1999). Comparing bayesian network classifiers. In *Proceeding UAI'99 Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*.

Fenton, N. Independence and conditional independence. https://www.eecs.qmul.ac.uk/~norman/BBNs/Independence_and_conditional_independence.htm. Accessed: 2017-03-06.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. In *Machine Learning Volume 29*.

Heckerman, D. (2008). *A Tutorial on Learning with Bayesian Networks*, pages 33–82. Springer Berlin Heidelberg, Berlin, Heidelberg.

Koch, K.-R. (2006). *Bayesian Inference with Geodetic Applications*, volume 31. Springer.

Marcot, B. G. (2012). Metrics for evaluating performance and uncertainty of bayesian network models. In *Ecological Modelling Volume 230*.

Niedermayer, D. (2008). An introduction to bayesian network theory and their contemporary applications. In *Innovations in Bayesian Networks Volume 156*.

Nikovski, D. (2000). Constructing bayesian networks for medical diagnosis from incomplete and partially correct statistics. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):509–516.

Russell, S. J., Norvig, P., and Davis, E. (2009). *Artificial intelligence: a modern approach*. Pearson, 3 edition.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *AAAI Workshop on Learning for Text Categorization*.

Stephenson, T. A. (2000). An introduction to bayesian network theory and usage. In *IDIAP Research Report*.

Sun, S., Zhang, C., and Yu, G. (2006). A bayesian network approach to traffic flow forecasting. In *Agents and Peer-to-Peer Computing*.

Uusitalo, L. (2007). Advantages and challenges of bayesian networks in environmental modelling. In *Ecological Modelling Volume 203*.

van Koten, C. and Gray, A. R. (2005). An application of bayesian network for predicting object-oriented software maintainability. In *Information and Software Technology Volume 48*.

Wang, Y. and Vassileva, J. (2005). Bayesian network trust model in peer-to-peer networds. In *Agents and Peer-to-Peer Computing*.