12-1994

An Analysis of Bayesian Networks as Classifiers

Gregory C. Ahlquist

AN ANALYSIS OF BAYESIAN NETWORKS AS CLASSIFIERS

THESIS

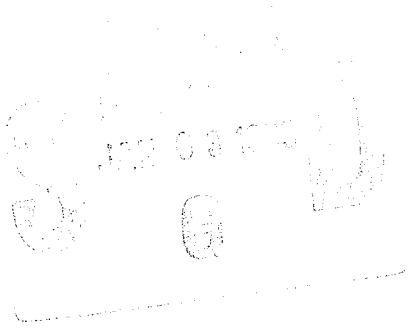
Gregory C. Ahlquist
Captain, USAF

AFIT/GCE/ENG/94D-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/94D-01

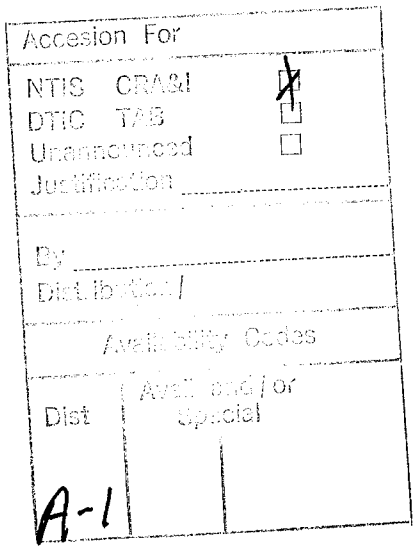


AN ANALYSIS OF BAYESIAN NETWORKS AS CLASSIFIERS

THESIS
Gregory C. Ahlquist
Captain, USAF

AFIT/GCE/ENG/94D-01

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.



AFIT/GCE/ENG/94D-01

AN ANALYSIS OF BAYESIAN NETWORKS AS CLASSIFIERS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Gregory C. Ahlquist, B.S.E.E.

Captain, USAF

December 1994

Acknowledgements

I owe my thanks to many people who substantially contributed to the accomplishment of this work. Thanks to my advisor, Dr Dennis Ruck for supplying timely advice and whip cracking when and where it was most needed. My thanks to my readers Dr Steve Rogers and Dr Eugene Santos, Steve for his enthusiasm, direction, and outside jumper, Doc for his numerous insights into Bayesian Networks and his wicked racquetball serve.

I extend heartfelt thanks to my sponsor, Marty Justice of Wright Laboratory's target recognition branch, for welcoming me as a member of the team. Marty handed me a challenge to pursue and introduced me to the good folks at Martin Marietta. An additional thank you goes to Wright Lab's Dr Tom Burns for his help in this area as well.

Special thanks to Steve Raney, Dr Noah Friedland, Brian Rothwell and several others at Martin Marietta for allowing me to join in their research. I especially appreciate their efforts to introduce me to their program, provide me with needed software, and answer my numerous questions. Without the help of Marty, Tom, Steve, Noah, and Brian, this thesis could not have been written.

Warm thanks to my classmates who, with their ability to make any situation humorous, made this effort infinitely more bearable. Of special note is my good friend, co-conspirator, and personal network supervisor Capt Eric Baenen. Certainly, overkill was our motto throughout our AFIT experience. Through Eric I've gained a valuable appreciation for Next machines and oriental cuisine, but it is our friendship I value most.

My deepest thanks and most loving appreciations go to the three people who put meaning in my life and made this whole AFIT experience bearable, my wife [redacted] and daughters [redacted] and [redacted]. Thank you for your patience and love and creating for me an island where, for a few blissful moments each day, I could forget AFIT existed. I love you all.

Finally, I offer thanks to my Heavenly Father for blessing me with the ability to accomplish this task.

Gregory C. Ahlquist

Table of Contents

List of Figures

List of Tables

Abstract

An analysis of Bayesian networks as classifiers is presented. This analysis results in an algorithm and several tools related to Bayesian network classifiers. The tools calculate and display the decision regions for two level Bayesian network classifiers. They collectively provide an approach to analyze the effects of changing network parameters on the network's decision regions. The algorithm defines a Bayesian network classifier to solve traditional classification problems. The algorithm is data driven, meaning that the resulting Bayesian network classifier is uniquely tuned to the classification problem at hand. Also, the algorithm contains procedures for defining the topology of a Bayesian network classifier and for precisely deriving the required conditional probabilities. A brief tutorial on Bayesian networks is also presented.

AN ANALYSIS OF BAYESIAN NETWORKS AS CLASSIFIERS

I. Introduction

Bayesian networks represent a promising new approach to the challenge of classifying objects in an image. Bayesian networks arise from the field of decision making theory which is a subset of artificial intelligence. Because classifiers decide among classes based on input, they are a special case of decision making systems. Thus, Bayesian networks may represent an excellent classifier architecture. This research examines the utility of Bayesian networks as classifiers. Three key areas are investigated. The first relates evidence input into a Bayesian network with the decision regions produced by the network. The second, develops methods to visualize a network's decision regions and to assess the impact of changing parameters. The third shows how to map Bayesian networks to classification problems. Before moving on to these topics, however, the basic terminology, background, and operation of Bayesian networks is presented. Further, the justification, scope, approach, and organization of this work needs to be set forth to establish context for discussion in subsequent chapters.

1.1 Definition of Terms

Bayesian networks bring with them their own specialized terminology and characteristics which must be introduced before discussing Bayesian networks in general. To begin, consider the terms decision making systems and uncertainty. Decision making systems comprise any system that must decide between competing courses of action based on current input. Advanced decision making systems incorporate uncertainty in their decision making where the uncertainty is a measure of how definite the evidence supporting each decision is or how often these decisions are correct given all possible scenarios (5). Bayesian networks are decision making systems that incorporate uncertainty. Their use of uncertainty is an important characteristic because it creates a means to store information in the network. The uncertainty

in a Bayesian network is associated with the network's architecture. Thus, to further the discussion, an example Bayesian network is presented. This example will serve as a basis for introducing all remaining terms and characteristics associated with Bayesian networks.

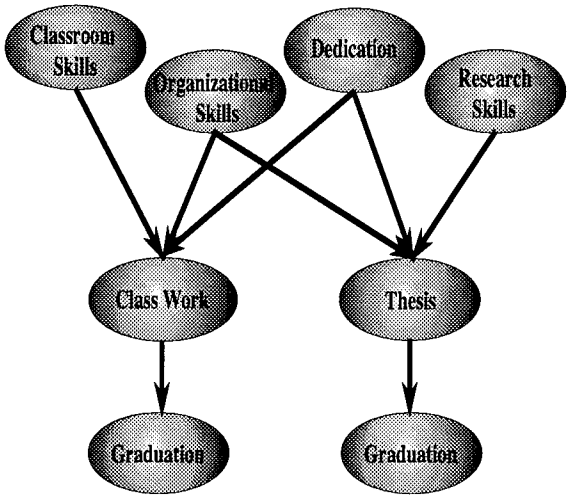


Figure 1.1 A Sample Bayesian Network

Figure 1.1 illustrates a sample Bayesian network. The network consists of a structural and conceptual definition. The structural definition consists of a set of nodes and arcs connected in a directed acyclic graph. The conceptual definition concerns the system the network attempts to represent. Conceptually, each node represents an entity which can assume one of several states. Parent nodes are typically defined to represent causes. These are connected via arcs to nodes that represent effects. Uncertainty is associated with each arc where it defines a relationship between the connected nodes. It represents a measure of how certain an effect is given a cause. Note that the Graduation node appears twice. This architecture results in an attempt to keep the propagation of information through the net simple. More thorough discussions of propagation mechanisms and issues of net simplicity appear in Chapter II.

In Figure 1.1, the top layer of nodes are causal while the middle layer are effect nodes for the top layer and causal nodes for the subsequent layer. The leaf nodes represent effects only. Using Figure 1.1 as an example, Organizational Skills, Dedication, and Classroom

Skills act as causes with their effect culminating in Class Work. Class Work, in turn, acts as a cause with the effect of Graduation.

Bayesian network terminology refers to the states within a node as hypotheses. The hypotheses are mutually exclusive, which means only one hypotheses can be true at a time, and collectively exhaustive, meaning all possible hypotheses are represented in the node. (8). Figure 1.2 shows the Thesis node of Figure 1.1 exploded to illustrate its hypotheses. In this example, the Thesis is the entity and the state of the Thesis is one of the node's hypotheses.

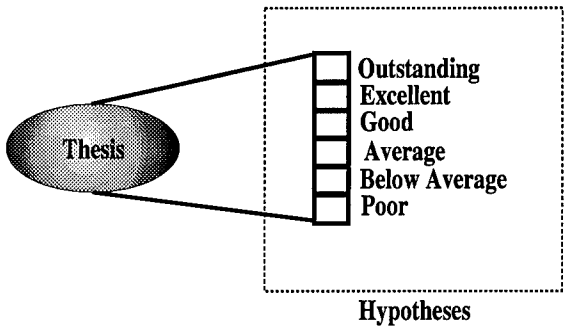


Figure 1.2 Hypotheses Associated with the Thesis Node

Each hypothesis has a probability of occurrence associated with it called a belief value. The belief value for a hypothesis is computed through Bayes law and it is this dependence on Bayes law that give Bayesian networks their name. Figure 1.3 adds belief values to Figure 1.2. Note the sum of the belief values must equal one because the hypotheses are collectively exhaustive.

Semantically, the belief value is a measure of the certainty that a given hypothesis is true and is dependent on the hypotheses of other nodes. As an example, a belief value of .5 associated with the Thesis hypothesis Good indicates that the probability the Thesis is a Good is fifty percent. This probability is conditional on the states of all the other nodes in the network. Bayes law expresses the condition and relates an hypothesis' belief value with the current belief values of hypotheses in neighboring nodes conditioned on the strength of relationships between nodes. Thus, the high mark in the Outstanding hypothesis of Figure 1.3 reflects high marks in the hypotheses of other nodes for which there is a strong causal

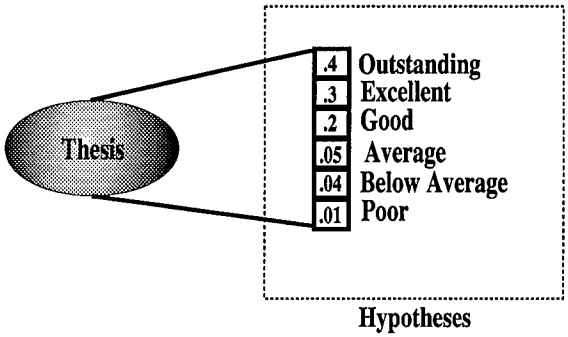


Figure 1.3 Belief Values Associated with Each Hypothesis

relation. As a further example to illustrate the point, assume the Graduation node of Figure 1.1 possesses the hypothesis Graduate With Honors. At AFIT, there is a strong causal relationship between graduating with honors and an outstanding thesis. Thus, the observation that a student graduated with honors, coupled with this strong association, would increase the belief that the student produced an outstanding thesis. The exact Bayes law relationship is discussed further in Chapter /refbackground.

Referring back to Figure 1.1, a matrix of conditional probabilities, representing uncertainty, is associated with each arc in the network. These conditional probabilities as a whole define relationships between the hypotheses of the nodes connected by the arc and represent information stored in the network. Each individual conditional probability associates two or more hypotheses together and gives a measure of the association's strength (1). For example, because the association between Graduating with Honors and an Outstanding Thesis is very strong, the value of .8 is a good approximation of the strength of the association. Thus, the conditional probability associating these hypotheses would be .8 and semantically represent the probability a student Graduates With Honors given an Outstanding Thesis. In contrast, the association between Graduating With Honors and a Poor Thesis would be quite low, perhaps .1, indicating the student producing a Poor Thesis rarely Graduates With Honors.

Thus, the basic terminology of Bayesian networks consist of nodes, arcs, hypotheses, belief values, and conditional probabilities. Each of these have not only a physical definition within the structure of the network, but also a semantic definition within the concept the

network represents. Discussions throughout the rest of these thesis will often refer to both definitions, but concentrate more heavily on the conceptual definitions.

1.2 Justification

This research is in direct support of efforts to develop better classifiers than those currently available. A classifier is a system that partitions a set of objects into a collection of related subsets or classes. It does this by comparing the data representing each object with a known sample or template of each class. Classifiers are key factors in the development of advanced military systems such as Automatic Target Recognition (ATR) and other systems that involve interpreting visual information. In fact, the reason we do not have such systems presently is due, in large measure, to the lack of capabilities of existing classifiers.

This work seeks to develop Bayesian networks as classifiers because Bayesian networks possess promising characteristics that may result in better classifiers. One such characteristic is the ability to incorporate uncertainty within the system. Currently, classifiers exist that perform flawlessly if they have access to all possible examples of the data set . However, supplying all possible data examples in a real world environment is an unachievable goal. Classifiers, therefore, must operate with only partial knowledge of the data which invariably degrades classifier performance as adequate information is simply not available (4). If ATR and similar systems are ever to become a reality, researchers must develop classifiers that can successfully operate at required levels with only partial access to the data. Bayesian network-based systems may result in such classifiers because of their ability to capture uncertainty. This ability is equivalent to storing a priori knowledge about the world in the classifier and may provide the information lacking due to an incomplete data set. Bayesian networks are further promising because they evolved from the discipline of decision theory. A classifier is a special case of decision making system because it must decide what class an object belongs to based on available data. Thus, Bayesian networks are promising as a classifier architecture.

The capabilities of Bayesian networks as classifiers is largely unknown. Because they arise from decision theory, most Bayesian network applications to date have been as decision

makers or controllers for complex systems (1). Few efforts have applied Bayesian networks specifically as classifiers. Thus, analyzing the potential of Bayesian networks as classifiers is a central motivation for this work.

1.3 Problem Statement and Scope

The focus of this research is directed toward developing Bayesian networks as classifiers. The main challenges include evaluating the decision regions produced by Bayesian networks and investigating how these decision regions change as network parameters vary. Also, few methods for mapping classification problems to Bayesian networks currently exist. This thesis presents and discusses one possible approach. The scope of this investigation will be limited to simple, two level Bayesian Networks where there is only a single path between any two nodes. This particular topology is inspired by current Bayesian network classification work performed at Martin Marietta in Denver, CO. Chapter II discusses The Martin Marietta effort in further detail and presents further assumptions designed to simplify the evaluation of the Bayesian networks.

1.4 General Approach

The approach for the first part of this thesis consists of deriving a relationship between the inputs to a network and the resulting network decision regions. Software is then developed to produce graphical representations of the decision regions. The development environment consists of Matlab running on a Sun Sparc 2 workstation located in the Computer Graphics Laboratory at the Air Force Institute of Technology. Programs written as Matlab functions create graphics to help visualize the decision regions of Bayesian networks deciding among two, three, and four hypotheses using evidence provided from two, three, and four sources respectively.

The second portion of this thesis presents an algorithm for mapping Bayesian networks to classification problems. The algorithm provides methods for exactly determining the topology of the Bayesian network and the values of the conditional probabilities that will optimize the

performance of the network. This approach is evaluated using simulated and real world data sets on a Sun Sparc 2.

1.5 Thesis Organization

Chapter II presents recent findings of noted researchers in the field of Bayesian networks and reviews the use of Bayesian networks in applications related to classification. Chapter II also presents an overview of Bayesian networks and their operation. Chapter III evaluates the decision regions of several Bayesian networks and presents the software for visualizing these regions. Chapter IV presents and evaluates an algorithm for mapping Bayesian networks to classification problems. Recommendations for future work and some brief conclusions close out this document.

II. Background Material

2.1 Introduction

Bayesian networks are new additions to methodologies for representing decision making systems incorporating uncertainty. As such, the background of Bayesian networks is relatively recent and, in many cases, still developing. As with many new concepts, the terminology of Bayesian networks varies widely throughout the research community. Bayesian networks are also known as Belief networks, influence diagrams, and probability nets. Researcher Judea Pearl is a pioneer is these areas and is the author on the definitive paper on Bayesian networks (11) (10). In fact, Pearl is credited with coining the term Bayesian networks (1). Fellow researcher Ross Shachter has also forwarded many important theories concerning influence diagrams. This chapter summarizes the work of both researchers and also looks at several applications of Bayesian networks in the literature. Theses applications deal mostly with Bayesian networks applied to control vision or classification systems. Very few efforts have specifically applied a Bayesian network as a classifier, with work performed by Martin Marietta a notable exception. Following the summary of applications is a general overview of Bayesian networks and their operation. This overview is important because the ideas presented are central to work presented in Chapters III and IV. Finally, the chapter concludes with a short discussion on drawbacks to Bayesian networks, most prominently, that the calculations involved are NP hard.

2.2 Bayesian Networks

2.2.1 Bayesian Networks - Judea Pearl: Judea Pearl's writings specify the form, function and properties of Bayesian networks (11). His is the definitive work in the field and forms the basis for the work performed in this thesis. Pearl's work is summarized here by way of example. This example introduces key concepts and terms associated with the operation of Bayesian networks and sets the stage for the more detailed overview presented later in the chapter.

For the example, assume a researcher constructs a Bayesian network to classify an object as either a tank, a truck, or a jeep . Figure 2.1 shows one possible topology for this classifier and serves as a reference for summarizing Pearl's work. The nodes in the network are exploded to show the hypotheses.

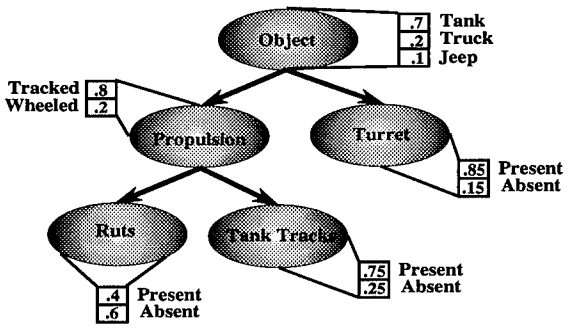


Figure 2.1 An Example Bayesian Network Classifier

Each node in the network receives information in the form of messages from its parent node and from its child nodes. The information from parent to child is referred to as anticipatory support. This information represents the likelihood of each hypothesis independent of any direct evidence in support of the hypothesis. It is a measure of what the network anticipates is the correct belief value for each hypothesis. The information from child to parent is referred to as diagnostic support. It represents direct information in support of each hypothesis in the node. This support flows upward from its source, the leaf nodes. Thus, information propagates in a Bayesian network from both the top down and from the bottom up.

Root and leaf nodes of the network are special cases. The leaf nodes of Figure 2.1 are called evidence nodes. These nodes receive evidence from the outside world. They are the main points through which new information is introduced into the network. The root node is referred to as a decision node. The anticipatory support for this node is called a priori information. This information represents initial beliefs about the likelihood, or the probability, of the decision node's hypotheses. A priori information is the researcher's expectations of the world before any evidence is acquired.

Before beginning the operation of a Bayesian network, the network must first be initialized. Example showing precisely how to initialize networks appear both in Section 2.3 and Appendix /reftutorial. However, the process is outlined here for continuity. A Bayesian network is initialized by establishing all the conditional probability values required by the network, establishing the a priori information, and placing values at the inputs of the evidence nodes that produce no effect within the network. The a priori information is allowed to propagate down through the network governed by equations defined by Pearl and presented later in the chapter. Propagating the a priori information through the network establishes initial belief values for all hypotheses. These initial values are a function of the a priori knowledge and the conditional probabilities and represent the networks expectation of the world devoid of any external evidence. The network is now considered instantiated and ready to receive evidence.

The network's operation initiates when evidence is presented to one of the evidence nodes. The network propagates the information contained in the evidence upward throughout the net, updating the belief values of all hypotheses as it goes. Intermediate nodes in the network serve as junction sites that fuse information together and forward information upward to parent nodes and downward to siblings. The presentation and propagation of evidence constitutes the central operation of the Bayesian network. Rules for propagating information, presented later in the chapter, ensure the network will eventually achieve equilibrium until new evidence is presented. After all evidence is presented and the network has achieved equilibrium, the hypothesis with the highest value represents the conclusion of the network for a given node.

Referring back to Figure 2.1 as an example, the evidence to this network consists of feature measurements corresponding to the evidence nodes. Thus, after the network is instantiated, measurements of the presence of Ruts, Tank Tracks, and a Turret are acquired and presented to the appropriate evidence nodes. The network propagates this evidence upward, updating belief values as it goes. After the network has achieved equilibrium, the highest belief value among the hypotheses of the root node represents the networks conclusion in regards to the class of the object.

Pearl's methods for propagating and fusing evidence is dependent on the topology of the network. The Bayesian network must assume the form of a polytree, meaning there can only be one path to any node. Figure 2.2 represents an illegal structure in a Bayesian Network.

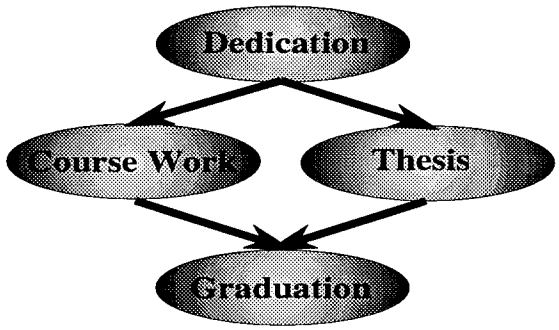


Figure 2.2 An Example Diamond Structure

This figure, known as a diamond, unfortunately represents many real world dependencies leading to Bayesian networks requiring a topology of this type. However, Pearl presents methods for designing Bayesian networks that avoid these topologies and for approximating these structures when they occur. These methods insure the overall network conforms to a polytree. One such approach is illustrated in Figure 1.1. Graduation is a result of both Course Work and Thesis. However, providing only one node representing Graduation and connecting Course Work and Thesis to it results in an illegal diamond structure. Providing a second but equal node alleviates this problem but does so at a cost. The second graduation node introduces the possibility of a contradiction in the network as the two nodes may favor different states. Thus, this simply techniques should only be used with caution.

2.2.2 Bayesian Networks - Ross Shachter: Shachter's work deals with influence diagrams (15). In influence diagrams, the underlying structures and methods are the same as Bayesian networks, but the interpretation of the results are different. Shachter focuses on a concept called a utility function which maps the beliefs of selected nodes to some value. The intent is to optimize the output of the utility function in some manner. This results in a decision making system that selects its next course of action based on how that action affects

the overlying utility function. The following paragraphs present a brief example of this idea. Figure 2.3 shows a simple Bayesian network expressed with the exploded format.

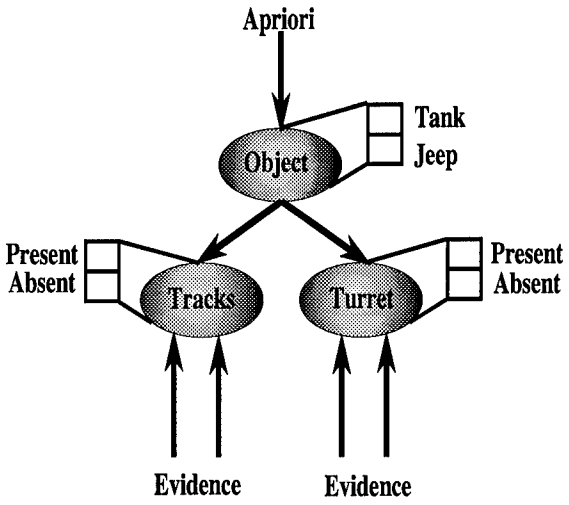


Figure 2.3 A Small Bayesian Network Classifier

This network is a classifier that will select between two classes using evidence concerning two features. The arrow at the top represents a priori information indicating a known or suspected probability distribution for the two classes. The arrows at the bottom represent numerical measures of whether the features are present or not. A function exists that relates the belief values of the decision node's hypotheses to the evidence measures in support of each feature. This function is dependent on the evidence values, the a priori values, and the conditional probabilities associating hypotheses . The function is derived using the equations for propagating information through the network. These equations will be covered in more detail later in this chapter.

The following equation expresses the function relating the belief value of the Tank hypothesis to the input measures. Because of the complexity of the equation, it is broken into parts.

$$\beta_1 = xycoef_1XY + xcoef_1X + ycoef_1Y + offset_1 \tag{2.1}$$

$$\beta_2 = xycoef_2XY + xcoef_2X + ycoef_2Y + offset_2 \tag{2.2}$$

$$\alpha = \beta_1 + \beta_2 \tag{2.3}$$

$$Belief_1 = \frac{\beta_1}{\alpha} \tag{2.4}$$

In Equations 2.1 and 2.2 the variable X represents the measure of evidence for the existence of Tracks. Y represents the measure of evidence for the existence of a Turret. All coefficients result from a combination of a priori values and conditional probabilities. The exact derivations of these coefficients are set forth in Chapter III. The value α serves as a normalization factor. $Belief_1$ represents the belief value for the Tank hypothesis and Equation 2.4 summarizes the relation between this value and the inputs on the evidence nodes. Figure 2.4 depicts this relationship graphically in three dimensions.

The dependency of the Tank hypothesis on changes in the evidence for each feature can be determined by taking the partial derivative of Equation 2.1 with respect to each feature. Equations 2.5 and 2.6 express these derivatives and Figures 2.5 and 2.6 graphically depict the results.

$$\frac{\partial}{\partial X} Belief_1 = \frac{\frac{\partial}{\partial X}\beta_1 \times \alpha - \frac{\partial}{\partial X}\alpha \times \beta_1}{\alpha^2} \tag{2.5}$$

$$\frac{\partial}{\partial Y} Belief_1 = \frac{\frac{\partial}{\partial Y}\beta_1 \times \alpha - \frac{\partial}{\partial Y}\alpha \times \beta_1}{\alpha^2} \tag{2.6}$$

Taking the difference of the absolute value of the results from Equation 2.6 and 2.5 shows where changing evidence values in one feature induces a greater change in the belief value for the Tank hypothesis than similar changes in the other feature. Equation 2.7 invokes

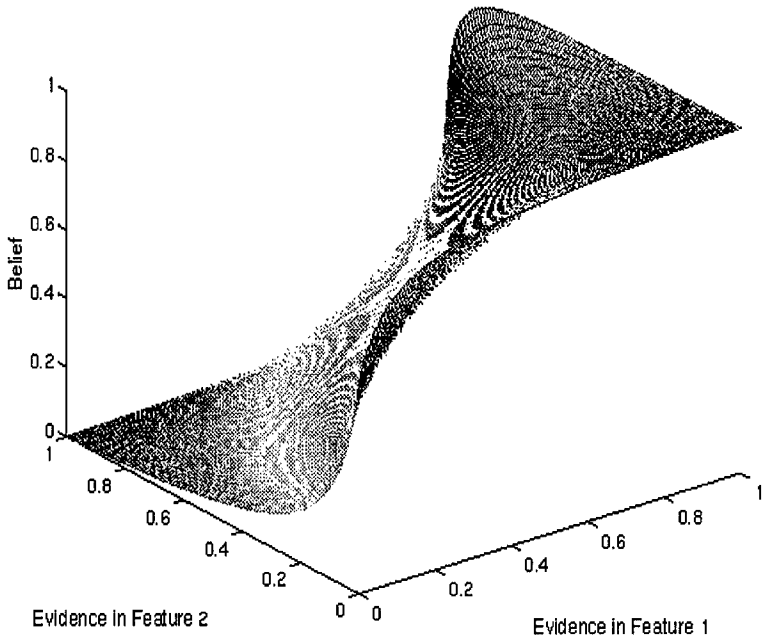


Figure 2.4 Belief in the Tank Hypothesis

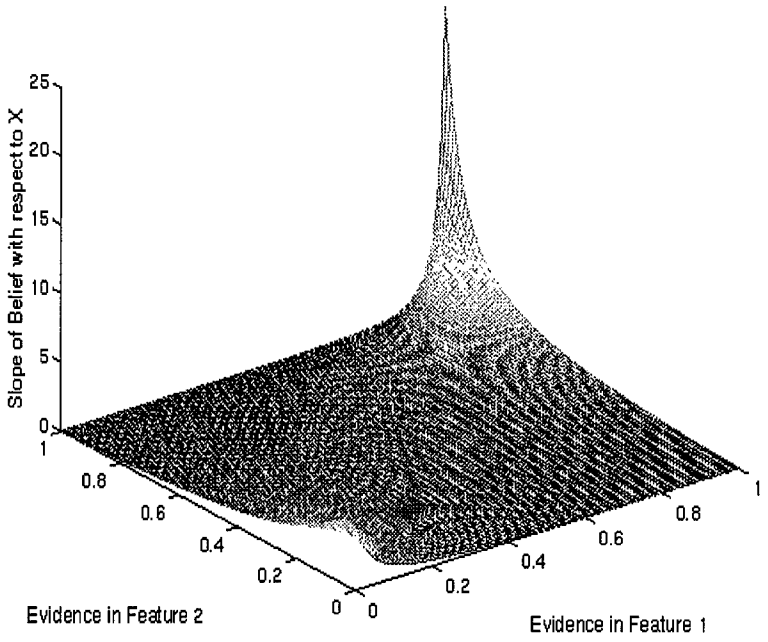


Figure 2.5 Derivative of the Belief with Respect to Tracks

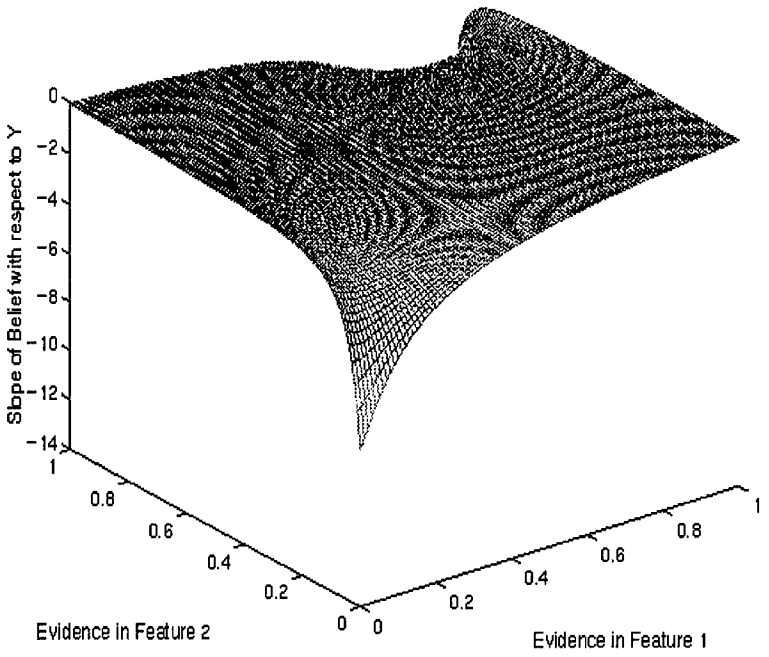


Figure 2.6 Derivative of the Belief with Respect to Turret

the Sign function on this difference. The Sign function returns a one if the value is zero or positive and a negative one otherwise. Figure 2.7 depicts the results.

$$diff = Sign\left(abs\left(\frac{\partial}{\partial X}Belief_1\right) - abs\left(\frac{\partial}{\partial Y}Belief_1\right)\right) \tag{2.7}$$

Figure 2.7 is a two dimensional graph showing that, at certain combinations of evidence values, accruing evidence for one feature produces a greater effect in the tank belief value than the other. Darker areas indicate where changes in evidence values for the Track feature produce a greater effect. Lighter areas indicate where changes in the evidence values for the Turret feature dominate. A researcher can now use this information to optimize an algorithm for applying functions to accrue evidence. Thus, Equation 2.7 is a utility function.

The other important concept Shachter forwards is the designer of the network can reduce the size of the network based solely on observations about the network's topology. Shachter proves that certain nodes in the net add no value and can be pruned. He also proves that

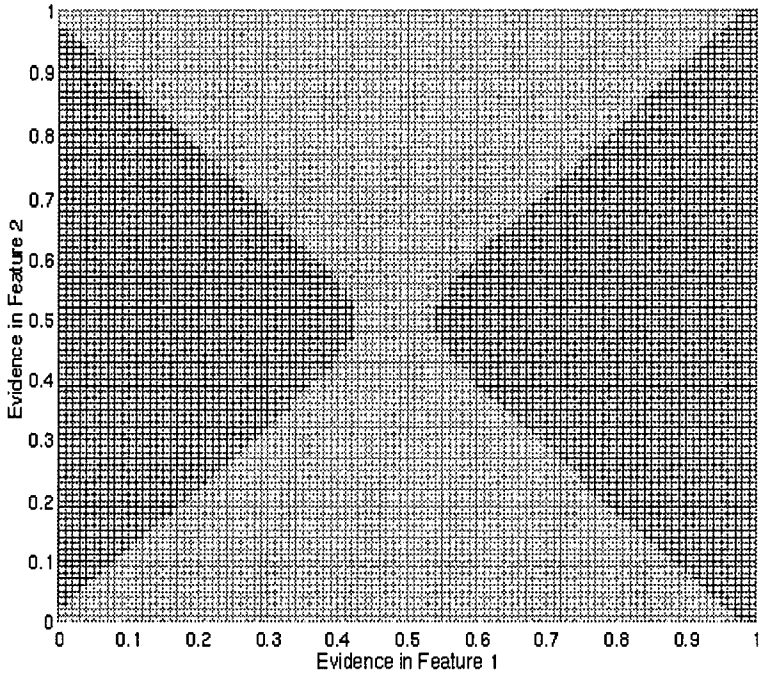


Figure 2.7 Regions of Greatest Change to the Belief Based on Changes in Evidence

any node in the network can be eliminated by simply representing the dependencies of the given node in alternative ways. Eliminating nodes based on network topology can reduce the size of the network while helping to insure it assumes a polytree form. This is an important issue because it helps develop efficient Bayesian networks that make the best use of their computational resources.

2.2.3 Bayesian Networks for Classification: Most applications to date apply Bayesian networks as a decision making system or controller for managing the operation of complex systems. This is due, in part to the association of Bayesian networks with decision theory. However, a few programs have specifically applied Bayesian networks as a classifier, The first of these is work led by Dr Noah Friedland and supported by Steve Raney, Brian Rothwell and others at Martin Marietta in Denver, CO. This work is highly important to this thesis because much of the work is directly based on or related to concepts developed in the

Martin Marietta work. The second area applies Bayesian networks in a ship classification system.

2.2.3.1 ARAGTAP. ARAGTAP stands for the Automatic Radar Air-to-Ground Target Acquistion Program. It is a USAF sponsored program to develop a system to automatically detect, segment, classify, and identify targets is Synthetic Aperture Radar (SAR) imagery. A research team at Martin Marietta in Denver, CO is responsible for the ARAGTAP effort (12).

ARAGTAP is a collection of algorithms for detecting targets in SAR imagery, segmenting the detected targets from the background, and identifying each target. Dr Noah Friedland is attempting to integrate a Bayesian network to the ARAGTAP system. The Bayesian network is designed to discriminate between closely related targets. The ARAGTAP system receives a SAR image containing a target and returns a set of closely scored hypotheses concerning the identity of the target. Martin Marietta researchers have observed that the hypothesis correctly specifying the identity and orientation of the target is usually a member of this set. To discriminate between, Dr Friedland and his fellow researchers apply a Bayesian network designed to discriminate among these closely scored hypotheses.

Dr Friedland and his co-researchers have developed software that takes this set of hypotheses and dynamically builds a Bayesian network around them. The hypotheses become the hypotheses of the root node. The Bayesian network itself consists of many levels and consists of n identical sub-graphs where n is the number of hypotheses. Evidence nodes at the bottom of each sub-graph receive evidence provided by feature extractors incorporated in ARAGTAP. This evidence is propagated up through each sub-graph. The root node fuses the evidence from the sub-graphs together and uses it to discriminated among the hypotheses.

The aspects of the ARAGTAP Bayesian network that are most important to this thesis include the topology of the network's top layers and a concept called the confusion matrix. The confusion matrix is a method for deriving the conditional probabilities that associate the hypotheses of the root node and its child nodes. A focal point of this thesis is an attempt to

analyze how changes to the confusion matrix, and hence the conditional probabilities, effect the belief values in the root node over changes in the evidence received at the child nodes. A full understanding of the significance of the confusion matrix requires a greater understanding of Bayesian networks so further discussion on this topic is postponed until Chapter III.

2.2.3.2 Bayesian Networks for Ship Classification: A research team lead by Musman has built a Bayesian network classifier to classify ships (9). The most important aspect of their research is their approach to organizing the network. They divide the classification problem into a hierarchy of subproblems. At the lowest level, they model the tasks of detecting base characteristics such as size, boundaries, and distinct features. Higher levels combine the outcomes of these low level operations to hypothesize about the ship's general shape and components of the superstructure. At an even higher level resides the nodes that hypothesize about the class of the ship. These high level classification nodes take the outputs of the lower levels as evidence to support their hypotheses. The Bayesian network itself is broken up into smaller nets each dedicated to detecting some aspect of the ship and then passing its findings to higher levels. This approach results in a means to collect intermediate information about the ship in question. Thus, even in cases where the actual class of the ship cannot be determined due to lack of evidence, the network generally can provide information on some characteristics of the ship. For example, the network may not be able to classify a ship as a battleship, but it may be able to determine the ship is a surface combatant. This is useful information in a military situation.

2.2.4 Bayesian Networks for Perceptual Organization: The applications covered in this section concern using Bayesian networks for perceptual organization. Although not classification per se, the end goal for these applications is essentially the same, identifying objects in a scene. Discussion of these applications to inform the interested reader about other areas where Bayesian networks are being applied in vision related applications.

Several researchers have applied Bayesian networks as a means to model the visual process. Rimey and Brown employ an influence diagram with a utility function to select

visual operations on the scene being analyzed (13). In this manner, they are able to direct their visual processing system to concentrate on areas that possess the highest potential of providing the most visual information for the computational resources expended. Sarkar and Boyer have developed a Bayesian network with a hierarchical organization (14) similar to Musman's except for the utility function. The base levels of the network accrue evidence concerning building blocks of the scene such as parallel lines and constant arcs. Upper levels of the net take the results of the lower levels as evidence to hypothesize about the existence of general shapes such as squares, circles and triangles. Even higher levels hypothesize the existence of certain objects in the scene. This network also possesses a predictive element where if a hypothesis begins to emerge as a front-runner, the system tunes itself to look for additional evidence in support of that hypothesis.

2.3 Bayesian Network Overview

2.3.1 Introduction. With the terms and concepts presented earlier in this chapter and in Chapter I serving as a primer, a further example is presented here to detail the general operation of a Bayesian network and present the equations that govern its operation. As part of the example, key observations are made concerning Bayesian networks that are applied throughout the remainder of this work. Additional examples will also be discussed to further highlight interesting aspects of Bayesian networks. The network that forms the context for the example appears in Figure 2.8 which shows the basic network in its uninstantiated state.

This network is the same as illustrated in Figure 2.1, however, the representation is different. The new representation presents several fields to display pertinent information. This includes the belief values of the hypotheses at each node, the parent messages disseminated down to child nodes, and the child messages sent up to parents. This representation is inspired by a similar display used in a Bayesian network tool built by Martin Marietta. The representation does not show the conditional probabilities because including them greatly complicates the representation. Tables in the next section list the conditional probabilities.

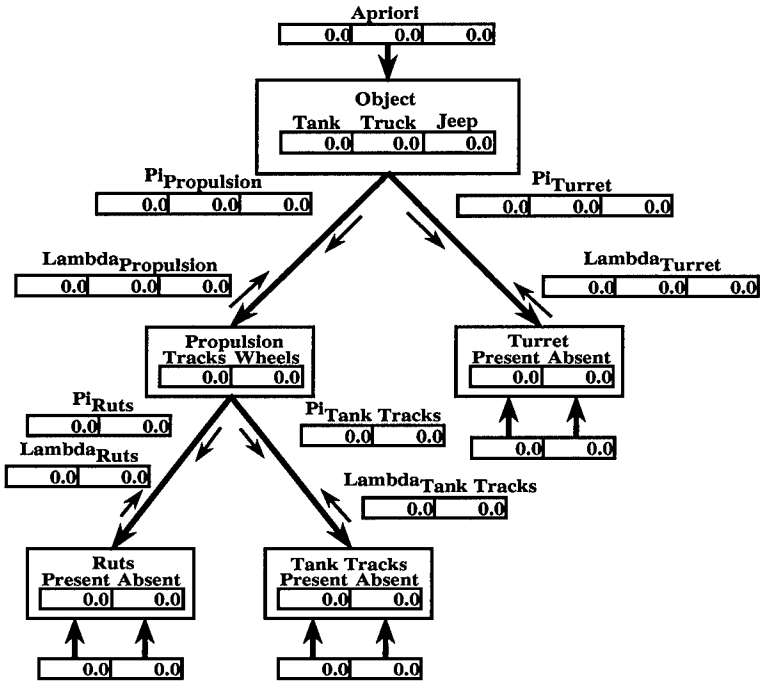


Figure 2.8 An Example Bayesian Network

The following sections briefly present the equations and definitions required to define the operation of the network as well as the values required to instantiate it. For the interested reader, a more thorough, step by step, tutorial on Bayesian networks appears in Appendix A.

2.3.2 Equations and Definitions. Several key equations govern the operation of a Bayesian network. They, and the general rules of thumb for propagating and interpreting information throughout the network are directly attributable to Judea Pearl (11). This presentation begins by calculating the belief values for the root node in Figure 2.8. The belief values are a function of anticipatory support and diagnostic support and the presentation continues by first explaining how the diagnostic support is derived and then explains the derivation of the anticipatory support.

The operation of the Bayesian network is centered on the belief vector where the elements of the belief vector are the belief values. Given the definitions of anticipatory and

diagnostic support earlier in this chapter , the belief values for the hypotheses in the root node are calculated by Equation 2.8.

$$BEL(Object_i) = \alpha\lambda(Object_i)\pi(Object_i) \tag{2.8}$$

$Object$ refers to the node Object in Figure 2.8 and $Object_i$ refers to the i^{th} hypothesis of that node. The variable $\lambda(Object_i)$ represents diagnostic support for $Object_i$ and $\pi(Object_i)$ represents anticipatory support for $Object_i$. The term α represents a normalization parameter. The sum of the belief values in any node is always normalized to one, indicating that the set of hypotheses is collectively exhaustive. Equation 2.8, therefore, fuses the diagnostic and anticipatory support for a hypothesis and produces a belief value for that hypothesis.

The full set of beliefs for a node is indicated as $BEL(Object)$ and is expressed for the network in Figure 2.8 in Equation 2.9.

$$BEL(Object) = [BEL(Tank),BEL(Truck),BEL(Jeep)] \tag{2.9}$$

This is a vector with one element for each hypothesis. Similarly, $\lambda(Object)$ and $\pi(Object)$ are also vectors. Equations 2.10 and 2.11 display these respectively.

$$\lambda(Object) = [\lambda(Tank),\lambda(Truck),\lambda(Jeep)] \tag{2.10}$$

$$\pi(Object) = [\pi(Tank),\pi(Truck),\pi(Jeep)] \tag{2.11}$$

The $\lambda(Object_i)$ in Equation 2.8 is the product of associated elements in child message vectors sent from the Object node's children. These child messages represent the diagnostic support sent to the root node from its children. Bayesian network terminology calls these messages λ message vectors. Figure 2.8 displays all the λ message vectors in the network and refers to them as $Lambda_X$. The subscript X indicates the node from which the message

vector originates. Equation 2.12 expresses the relation between the elements of a λ vector and the elements of the λ message vectors.

$$\lambda(Object_i) = \prod_{j=1}^{n} \lambda_j(Object_i) \tag{2.12}$$

In this equation, n is the number of child nodes and $\lambda_j(Object_i)$ is the i^{th} element of the λ message vector from the j^{th} child node. Each λ message vector has one element for each hypothesis in the parent node. Equation 2.12 represents the fusion of all diagnostic information received by the root node.

Each child node generates a λ message vector representing diagnostic information sent to its parent. Equation 2.13 expresses how to calculate the elements of the λ message vector from the Propulsion node to the Object node.

$$\lambda_{Propulsion}(Object_i) = \sum_{j} P\left[Propulsion_j | Object_i\right] \lambda\left(Propulsion_j\right) \tag{2.13}$$

This equation is the first to make use of the conditional probabilities associating the hypotheses of two nodes together. The conditional probability, $P\left[Propulsion_j | Object_i\right]$, expresses the strength of the association between hypotheses $Object_i$, and $Propulsion_j$. The network multiplies the strength of this association by the level of diagnostic support for $Propulsion_j$ received by the Propulsion node. For example, Equation 2.14 shows the calculation for determining the diagnostic support for the Tank hypothesis forwarded by the Propulsion node.

$$\lambda_{Propulsion}(Tank) = P[Tracks|Tank]\lambda(Tracks) + P[Wheels|Tank]\lambda(Wheels) \tag{2.14}$$

In words, Equation 2.14 states that the diagnostic support for the Tank hypothesis received from the Propulsion node is equal to the evidence for a tracked propulsion system multiplied by the probability a tank uses a tracked propulsion system, plus the evidence for a wheeled propulsion system multiplied by the probability a tank uses wheels to propel itself. As one might expect the $P[Tracks|Tank]$ is a value close to one, while $P[Wheels|Tank]$ is a much smaller value.

The network calculates the $\lambda_{Propulsion}(Truck)$ and $\lambda_{Propulsion}(Jeep)$ entries in similar fashion by inserting the appropriate conditional probabilities into Equation 2.14. Equation 2.15 expresses the complete $\lambda_{Propulsion}(Object)$ message vector.

$$\lambda_{Propulsion}(Object) = [\lambda_{Propulsion}(Tank), \lambda_{Propulsion}(Truck), \lambda_{Propulsion}(Jeep)] \quad (2.15)$$

The $\lambda\,(Propulsion_j)$ values present in Equations 2.13 and 2.14 are elements of the λ vector for the Propulsion node. The network calculates the Propulsion node λ vector exactly as it calculated the λ vector for the Object node. At this point, the algorithms for calculating λ vectors and λ messages vectors become recursive. The network simply applies the governing equations from the perspective of the current node.

Turning back to Equation 2.8, the π vector element $\pi(Object_i)$ is a measure of the anticipatory support for hypothesis $Object_i$. The elements of a nodes π vector are functions of the elements of the π message vector sent down from the parent node and the conditional probabilities associating the hypotheses in the parent node with those of the child. Figure 2.8 displays π message vectors for each node and labels them Pi_Y. The subscript Y indicates the message vector's destination.

Since the Object node is the root node and has no parent, the calculation of its π vector is a special case. Subsequent paragraphs will discus the π vector for root nodes. This section calculates the elements of the Propulsion node π vector to illustrate the general case. Equation 2.16 shows the calculation involved for deriving the Propulsion node π vector.

$$\pi\left(Propulsion_j\right) = \sum_i P\left[Propulsion_j | Object_i\right] \pi_{Propulsion}(Object_i) \qquad (2.16)$$

The conditional probabilities in the above equation hold the same interpretation as discussed before. The term $\pi_{Propulsion}(Object_i)$ refers to the element in the π message vector from the Object node to the Propulsion node for $Object_i$. As an example, Equation 2.17 calculates the π vector element for the Tracks hypothesis.

$$\begin{aligned} \pi(Tracks) &= P[Tracks|Tank]\pi_{Propulsion}(Tank) \\ &+ P[Tracks|Truck]\pi_{Propulsion}(Truck) \\ &+ P[Tracks|Jeep]\pi_{Propulsion}(Jeep) \end{aligned} \qquad (2.17)$$

In words, Equation 2.16 states that the anticipatory support for a Tracks hypothesis is equal to the probability that a tank has a tracked propulsion system multiplied by the current likelihood the object is a tank, plus the probability a truck has a tracked propulsion system multiplied by the current likelihood the object is a truck, plus the probability a jeep has a tracked propulsion system multiplied by the current likelihood the object is a jeep. The network calculates the π vector element for the hypothesis Wheels in a similar fashion.

One last equation to consider is the calculation of the elements of the π message vector from the parent node to the child. This message transmits anticipatory support for the hypotheses of the child node. It also conveys information about the state of the child node's siblings. Pearl (11) showed that the beliefs in the hypotheses of a child node effect the anticipatory support for hypotheses in sibling nodes. As a quick example, if the belief values in the Turret node suggest that a turret is present, this strengthens the anticipatory support for propulsion systems used by vehicles with turrets. Elements of the π message vectors capture

this relation. Equation 2.18 calculates the elements of the π message vector from the Object node to the Propulsion node.

$$\pi_{Propulsion}(Object_i) = \alpha\pi(Object_i)\lambda_{Turret}(Object_i) \tag{2.18}$$

The term α is again a normalization constant. As a more specific example, Equation 2.19 applies Equation 2.18 to calculate the anticipatory support due to the current belief in the Tank hypothesis and the diagnostic support for the Tank hypothesis provided by the Turret node.

$$\pi_{Propulsion}(Tank) = \alpha\pi(Tank)\lambda_{Turret}(Tank) \tag{2.19}$$

The network calculates anticipatory support stemming from the hypothesis Truck and Jeep similarly. Equation 2.20 illustrates the complete Object π message vector to the Propulsion node.

$$\pi_{Propulsion}(Object) = [\pi_{Propulsion}(Tank),\pi_{Propulsion}(Truck),\pi_{Propulsion}(Jeep)] \tag{2.20}$$

Equations 2.8 through 2.19 relate to the specific Bayesian network in Figure 2.8. Equations 2.21 through 2.25 summarize the general form of the key equations for propagating information through a Bayesian network. Figure 2.9 shows a general Bayesian network which provides the basis for Equations 2.21 through 2.25 Node B of Figure 2.9 serves as the reference node.

The belief vector elements for node B.

$$BEL(B_i) = \alpha\lambda(B_i)\pi(B_i) \tag{2.21}$$

The λ vector elements for node B.

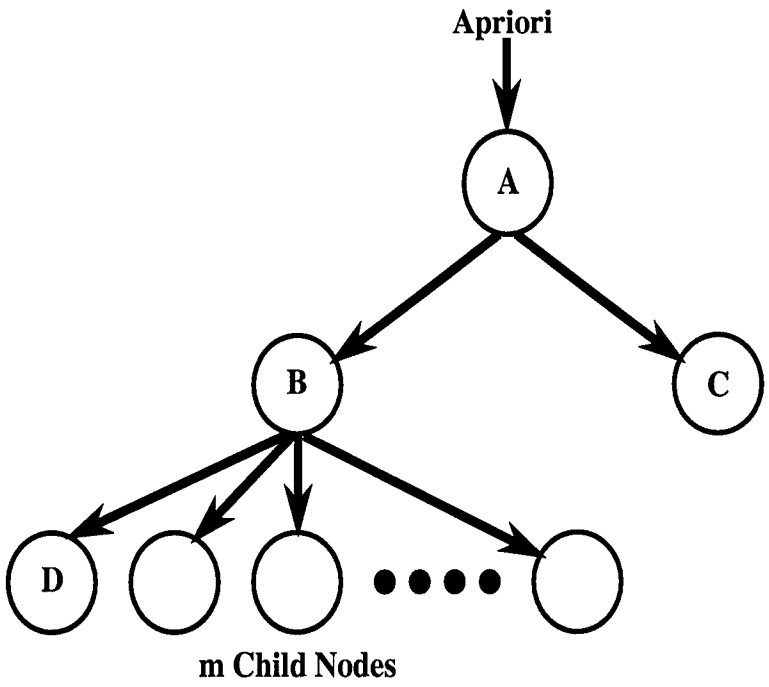


Figure 2.9 A Generic Bayesian Network

$$\lambda(B_i) = \prod_{j=1}^{m} \lambda_j(B_i) \tag{2.22}$$

The π vector elements for node B.

$$\pi\,(B_i) = \sum_j P\,[B_i|A_j]\,\pi_B(A_j) \tag{2.23}$$

The λ message vector elements from node B to node A.

$$\lambda_B(A_j) = \sum_i P\,[B_i|A_j]\,\lambda\,(B_i) \tag{2.24}$$

The π message vector elements from node B to node D.

$$\pi_D(B_i) = \alpha\pi(B_i) \prod_{m \neq D} \lambda_m(B_i) \tag{2.25}$$

The product term in Equation 2.25 is meant to produce the product of the i_{th} elements of all the λ message vectors except the λ message vector from the node for which the π message vector is intended.

The general equations for propagating information throughout a Bayesian network apply to every node within the network except for a few special cases. The special cases relevant to Figure 2.8 are the leaf nodes and the the root node. The leaf nodes have no child nodes so Equations 2.22 and 2.25 do not apply. For these nodes, the λ vector is equivalent to the evidence vector input to the node. On the other end of the network, the root node has no parent node. Equations 2.23 and 2.24 do not apply. For root nodes, the a priori vector serves as the π vector.

While Equations 2.21 through 2.25 control the flow of information through the network, the actual propagation of information through each node occurs in a series of steps. Assume a user has instantiated the Bayesian network by establishing all required conditional probabilities and a priori vector elements. The network enters a state of equilibrium and is prepared to accept evidence on one, some, or all of its input nodes. Introducing new evidence stimulates the effected leaf node to update.The general sequence of events for processing a node are as follows.

Step 1: Inspect all messages received from the parent and child nodes.

Step 2: Calculate the new λ vector via Equation 2.22

Step 3: Calculate the new π vector via Equation 2.23

Step 4: Calculate the new belief vector via Equation 2.21

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent.

Step 6: Calculate a new π message vector for each child node via Equation 2.25.Transmit these to the appropriate child nodes.

Updating nodes results in a flurry of new message vectors and the message vectors stimulate neighboring nodes to update. The new updates produce new message vectors and so on. A general rule of thumb for propagating messages throughout the Bayesian network is

that if a node is updating, it does not send a message back to the node that sent the stimulus to update. This ensures the network will eventually achieve equilibrium. For example, If a node is stimulated to update through receiving a π message from its parent, it does not send a λ message vector back to the parent. Conversely, if the node is stimulated to update by receiving a λ message from a child, it does not send a π message vector to the node from which the λ message originated. In this manner, the network eventually exhausts all updates and achieves a steady state.

Equations 2.21 through 2.25, coupled with the six steps for updating a node and the general rule for propagating information throughout the network, provide all the information necessary to specify the operation of of the network. The next section illustrates the application of these equations and rules to in an example Bayesian network.

2.3.3 A Bayesian Network Example. As a general overview of the operation of a Bayesian network, the network must first be instantiated which requires the conditional probabilities an a priori vector be established. Evidence is the acquired through some outside means and presented to the network. The network receives the evidence and propagates its effects throughout the network. Once the network has received all the evidence and has achieved equilibrium, the belief values at the node indicate the networks final conclusion given the evidence. One of the first issues that must be addressed, therefore, is the establishment of the conditional probabilities.

2.3.3.1 Establishing Conditional Probabilities. Establishing conditional probabilities is one of the central challenges in Bayesian networks for several reasons. First, the number of conditional probabilities in a network can be relatively large. Each arc connecting two nodes requires $m \times n$ conditional probabilities where m is the number of hypotheses in the parent node and n is the number of hypotheses in the child node. Second, there is usually no direct way to specify the conditional probabilities. In fact, most conditional probabilities in Bayesian networks are established through educated guesses (1). The conditional probabilities for this example are approximated values that semantically seemed plausible.

Bayesian network methodology often expresses the conditional probabilities in matrices. This thesis expresses the conditional probabilities in the form of tables to add labels to the rows and columns, thus conveying more information about which hypotheses the conditional probabilities associate. Table 2.1 expresses the conditional probabilities associating the hypotheses in the Object node of Figure 2.8 to the hypotheses in the Propulsion node.

Table 2.1 Conditional Probability Table Between Nodes Object and Propulsion

Hypothesis	Tank	Truck	Jeep
Tracks	.95	.05	.01
Wheels	.05	.95	.99

Each entry in the table represents the probability, $P[Propulsion_j|Object_i]$, that the method of $Propulsion_j$ is associated with $Object_i$. For instance, the value .95 in the Tank column and the Tracks row means $P[Tracks|Tank] = .95$. Semantically, this .95 value indicates a strong correlation between tanks and tracked propulsion systems. In other words, if the object is a tank, there is a .95 probability it will have tracks. The fact this value is not unity indicates that the network anticipates a small percentage of tanks do not employ tracked propulsion systems. In this manner, Bayesian networks account for the unusual or unanticipated case. As another example, the low values entered in the Truck and Jeep columns of the Track row indicate the network anticipates tracked propulsion systems are not expected on these vehicles. However, since the values are not zero, they do accommodate rare cases.

Note that the column entries of Table 2.1 sum to one. This means the network anticipates only tracked and wheeled propulsion systems. In other words, the Bayesian network pictured in Figure 2.8 cannot handle classifying tanks propelled by hovercraft. It would assign to this tank either tracks or wheels. Should the designer of the network find this restrictive, the designer must add additional hypotheses or add the hypothesis Other. The Other hypothesis accounts for all unknown objects. The network for this example, however, does not make use of Other hypotheses.

The conditional probability matrices for the other node to node connections in Figure 2.8 appear in Tables 2.2 through 2.4. There are four conditional probability tables total, one for each arc in the network.

Table 2.2 Conditional Probability Matrix Between Nodes Object and Turret

Hypothesis	Tank	Truck	Jeep
Present	.999	.02	.005
Absent	.001	.98	.995

Table 2.3 Conditional Probability Matrix Between Nodes Propulsion and Ruts

Hypothesis	Tracks	Wheels
Present	.2	.6
Absent	.8	.4

Table 2.4 Conditional Probability Matrix Between Nodes Propulsion and Tank Tracks

Hypothesis	Tracks	Wheels
Present	.94	.001
Absent	.06	.999

The other piece of information required to instantiate a Bayesian network is the a priori vector. The user provides this vector and it represents the expected distribution of the root node hypotheses. For this example, the a priori vector is established at [.45 .25 .30]. The vector's elements state that, for every 100 objects encountered, 45 are expected to be tanks, 25 trucks, and 30 jeeps. Establishing the a priori vector faces many of the same challenges as does establishing conditional probabilities. If no data is available to assess the elements of the vector, a common practice is to set them such that the elements of the vector are equal and sum to one.

Figure 2.10 shows the Bayesian network with the a priori vector established and inputs at the evidence nodes. The network has yet to propagate any information through the network at this stage and all evidence inputs are set to .5. Discussions in Chapter III will show this input is equivalent to presenting no diagnostic information to the network.

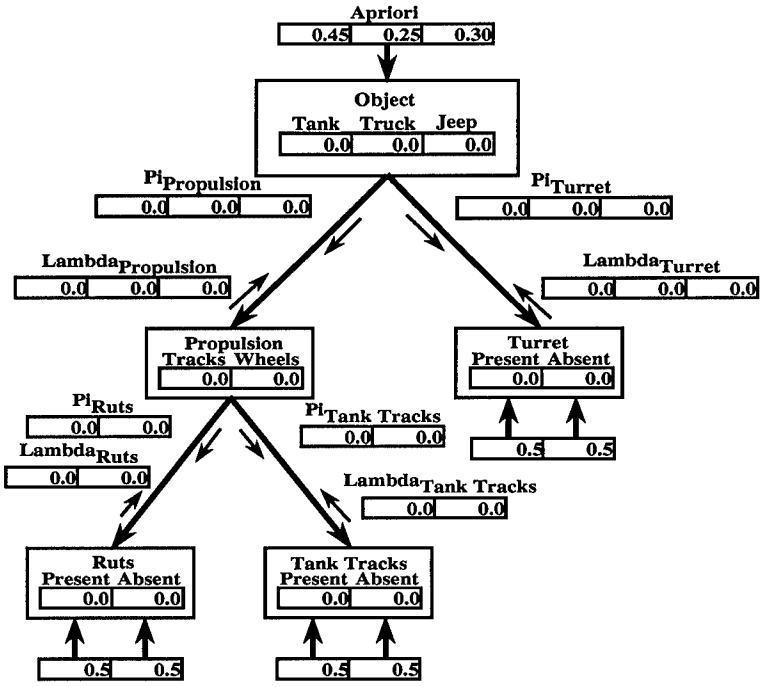


Figure 2.10 An Example Bayesian Network with A Priori Values Established

2.3.3.2 Initializing the Network. With the conditional probability tables and the a priori vector established, the network initializes by propagating the a priori information through the network. Propagating information through the network, especially while initializing, involves many tedious calculations. However, several general observations that help simplify the calculations.

1. The λ vector for an evidence node is equivalent to the input vector.

2. If the elements of a λ vector for a given node are all some value b then the elements of the node λ message vector will all equal b.

3. The π vector for a root node is equal to the a priori vector.

4. For any node, if the elements of the λ vector are equivalent and the sum of the π vector elements sum to one, then the belief vector will be equivalent to the π vector.

5. Similarly, for any node, if the elements of the λ vector are equal and the elements of the π vector sum to one, the π message vectors will be equivalent to the π vector.

6. If a node has only two λ message vectors, and one or both of those have elements with equal values, then the π message vectors equal the belief vector.

Appendix A detail the calculations that show these observations to be true. Applying these observations to simplify the calculations, the a priori information is allowed to propagate through the network. Figure 2.11 shows the network after it has achieved equilibrium.

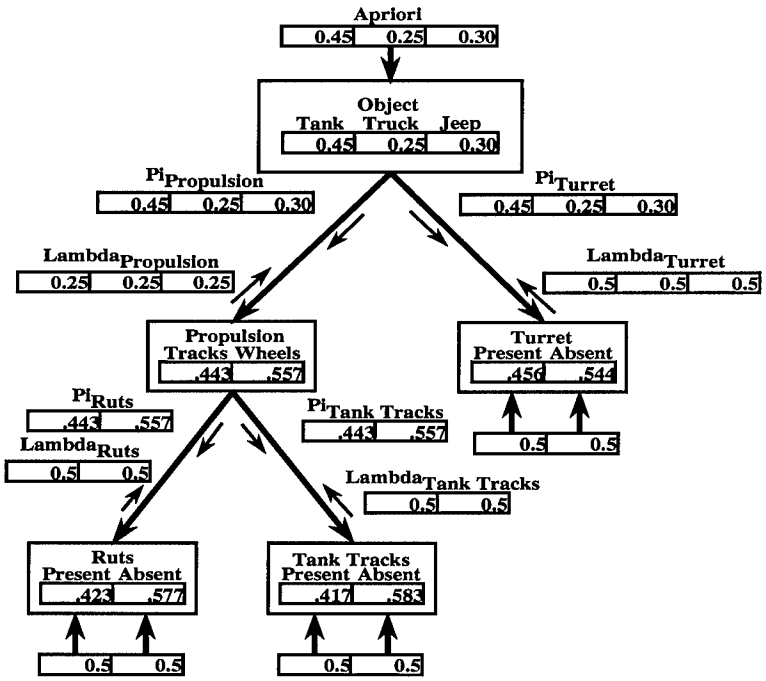


Figure 2.11 A Fully Initialized Bayesian Network

Several key observations help interpret the information contained in the network of Figure 2.11. First, all belief vectors are a function of anticipatory knowledge and the conditional probabilities only. Thus, they indicate what the network expects to encounter. Second, the belief vector of the root node is equal to the a priori vector. This is a semantically pleasing result because, without evidence, there is no impetus to sway the network from from its initial bias. This result also means that input values of .5 at the evidence nodes do not effect the root belief vector. Thus, evidence values of .5 serve as identity values for Bayesian networks. Third, The initial belief vectors at the Propulsion and Turret nodes show the network initially expects to encounter turretless, wheeled vehicles. This is primarily due to the higher a priori

distribution of wheeled vehicles versus tracked vehicles. Finally, it is interesting to note that the network anticipates encountering neither ruts nor tank tracks. This is again due to the higher a priori distribution of wheeled vehicles coupled with the relatively low probability of wheeled vehicles making ruts. To summarize, the network initially expects to encounter objects that leave no tracks, possess wheels, do not have turrets, and yet are tanks. This is the best the network can do without more direct evidence. However, the network is now fully initialized and ready to receive that evidence.

2.3.3.3 Propagation Evidence Through the Network. The network in Figure 2.11 receives evidence in a variety of ways. It may receive evidence incrementally to one evidence node, incrementally over all nodes, instantaneously at one evidence node, or instantaneously over all nodes. The method for presenting evidence does not effect the final belief vectors after the network has received all the evidence. However, observing the belief vectors change as the network receives each piece of information provides insight into the behavior of the network. For this example, therefore, the network receives evidence instantaneously one node at the time. This approach allows observation into how the belief vectors change as evidence propagates through the network.

Evidence presented to the network can originate essentially anywhere, from human operators, to the measurements of sensors, to the output of software routines. In this example, assume the network receives evidence from feature extractors run against a SAR image containing exactly one object. Before proceeding, however, an assumption must be made about the evidence presented to the network. That assumption is the evidence values presented to each evidence node must sum to one. The reason for this is that the network assumes all presented evidence is in support of one hypothesis or another. Evidence values that do not sum to one imply the existence of additional hypotheses that the network does not support.

To begin, assume the appropriate feature extractor presents an input vector of of [.43 .57] to the Ruts node. The new evidence stimulates the Ruts node to update and cycle through the six steps for performing an update. The update include calculating a new belief vector and all

required λ and π message vectors to neighboring nodes. Figure 2.12 shows the network after the update.

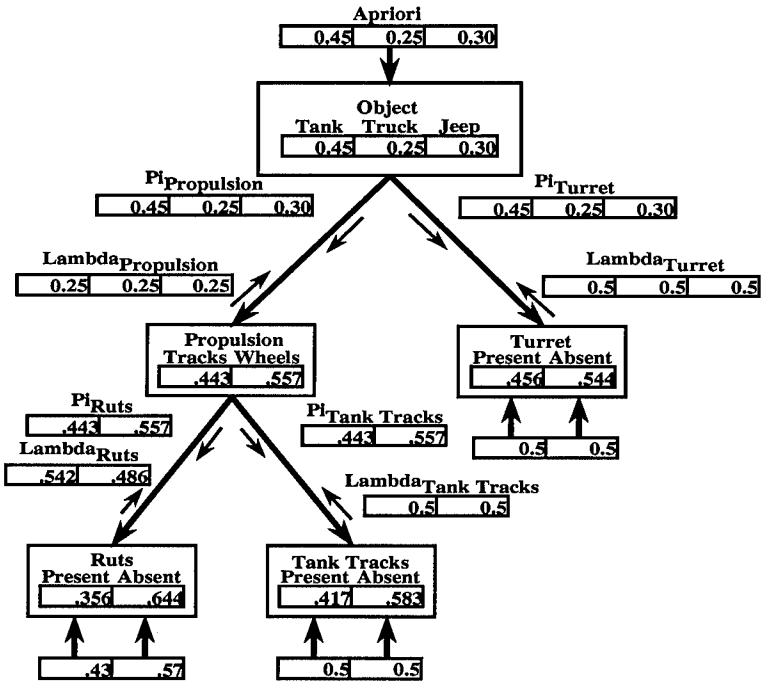


Figure 2.12 Update of the Ruts Node

A quick comparison with Figure 2.11 shows a decrease in the network's belief that ruts are present. This decrease is a result of minimal support for the presence of ruts supplied by the feature extractor. This weak evidence, coupled with the network anticipating the absence of ruts, decreases the networks belief that ruts are present. The λ_{Ruts} message vector conveys this information to the rest of the network. The elements of the vector indicate increased support for tracked vehicles.

The new λ_{Ruts} message vector to the Propulsion node stimulates it to update. The node runs through the the six steps to update. Figure 2.13 displays the resulting new belief vector as well as new λ and π message vectors.

A quick comparison with Figure 2.12 shows that the weak evidential support for the presence of ruts now manifests itself as weakened belief that the object is a wheeled vehicle. Consequently, the π vector message to the Tank Tracks node shows increased anticipatory

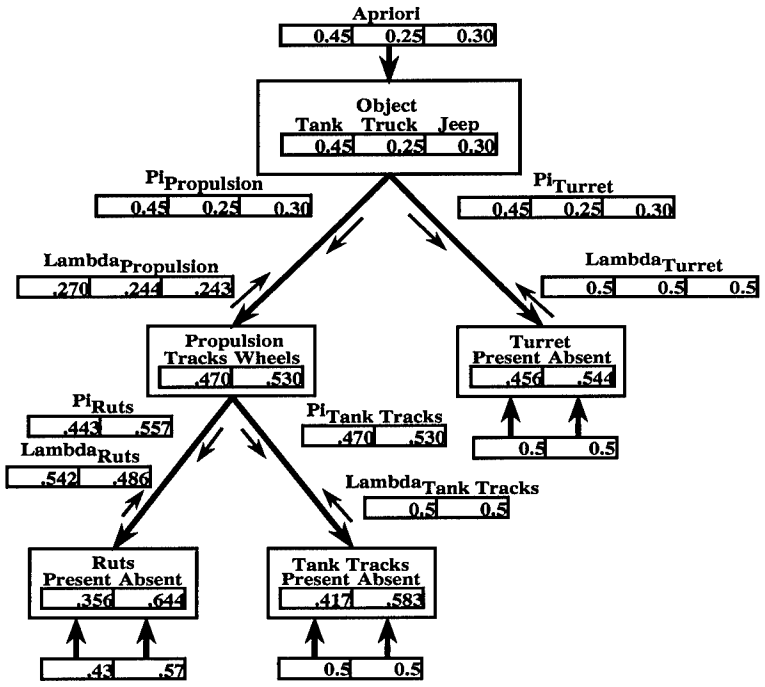


Figure 2.13 Update of the Propulsion Node

support for the presence of tank tracks and the λ message vector to the Object node reflects increased diagnostic support that the object is a tank. This second assertion may not be initially clear because the element values in the $\lambda_{Propulsion}$ message vector actually decrease. Pearl's algorithm for calculating λ message vectors does not call for normalizing them as it does for belief vectors or π message vectors. This makes it more difficult to assess at a glance the relative strengths of the diagnostic support for the hypotheses of the targeted node. However, elements with larger values indicate stronger support for the associated hypothesis.

Message vectors to the Tank Tracks and Object nodes stimulate both to update. The nodes may be updated in order or in parallel without effecting the final belief vectors of the network. This example updates the node in parallel. Both cycle through the six step for an update, producing new belief and message vectors as required. Figure 2.14 illustrates the updates to the Tank Tracks and Object nodes.

Comparing Figure 2.13 shows an increase in the belief the object has tank tracks. Thus, the weak support for the presence of ruts actually strengthens the network's belief in the

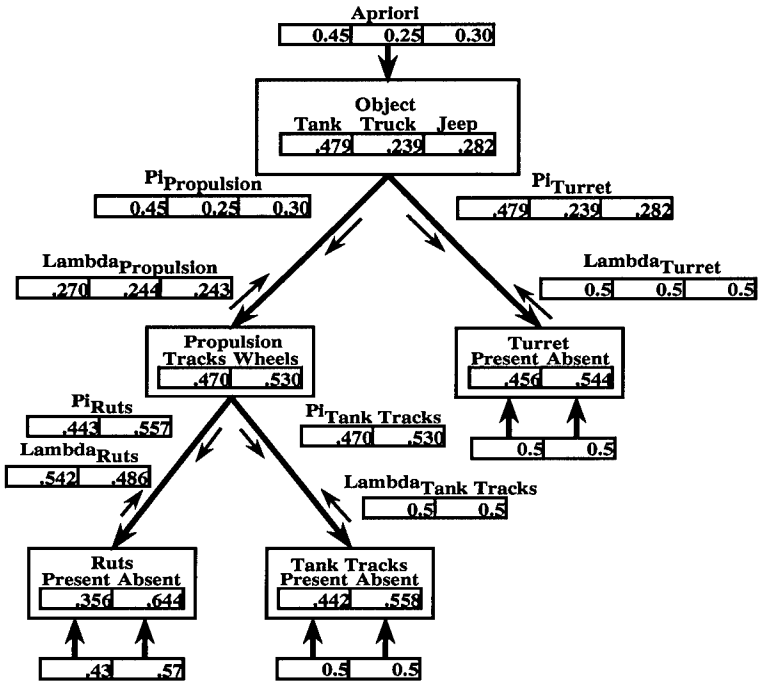


Figure 2.14 Update of the Object Node

presence of competing evidence. The comparison further shows that the weak evidence for the presence of Ruts also manifests itself as a slight increase in the Object node's belief that the object is a tank. This increased belief comes at the expense of the beliefs in wheeled vehicles. Thus the network, combining a priori information that anticipates tanks and weak diagnostic evidence for wheeled vehicles, makes a preliminary assertion that the target is a tank. With so little information collected, however, this assertion is, as indicated by a belief value less than .5, still very tentative and uncertain.

The update of the Tank Tracks node does not produce any message vectors and does not stimulate any nodes. However, the Object node produces a π message to the turret node, requiring an update. Figure 2.15 illustrates the update to the Turret node.

A quick comparison with Figure 2.14 shows the increased anticipation in tanks increases the belief in turret. The update to the Turret node generated no λ or π messages. As the network has updated every node, the propagation of information linked to the introduction of evidence

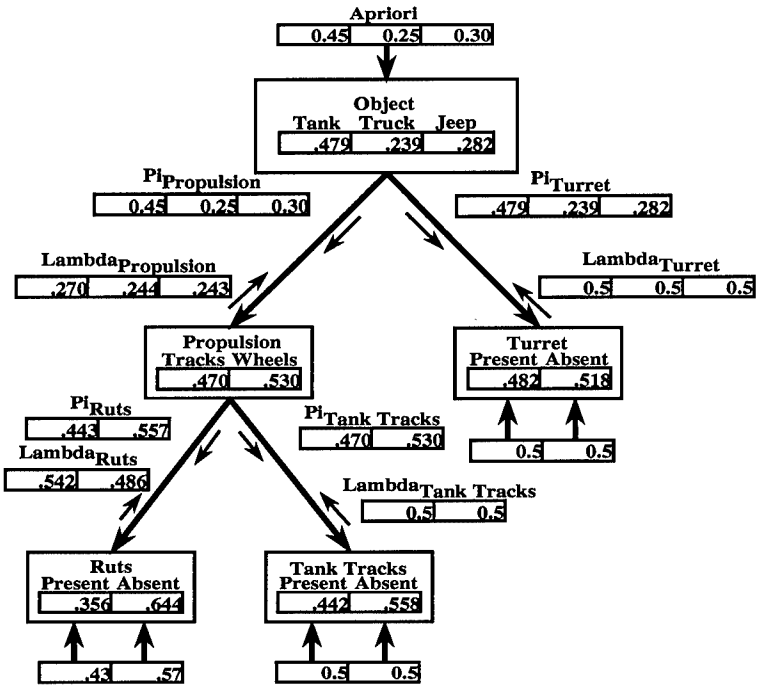


Figure 2.15 Update of the Turret Node

at the Ruts node ends here. The network in Figure 2.15 is now at equilibrium and will not change until new evidence is introduced.

To continue, the network receives a new input vector at the Tank Tracks node. The new input vector is [.91 .09], which greatly favors the existence of Tank Tracks. The new evidence stimulates the Tank Tracks node to update. The update calculates a new belief vector and generates a λ message vector for the Propulsion node. Figure 2.16 displays the results.

The update greatly increases the Tank Tracks belief vector. However, the increase is tempered by weak anticipatory support for the presence of tank tracks and results in a favoring belief value less then the corresponding input.

The strong evidential support for tank tracks also results in strong diagnostic support for tracks in the λ message vector to the propulsion node. The new message vector stimulates the Propulsion node to update with the update producing a new belief vector, a new $\lambda_{Propulsion}$

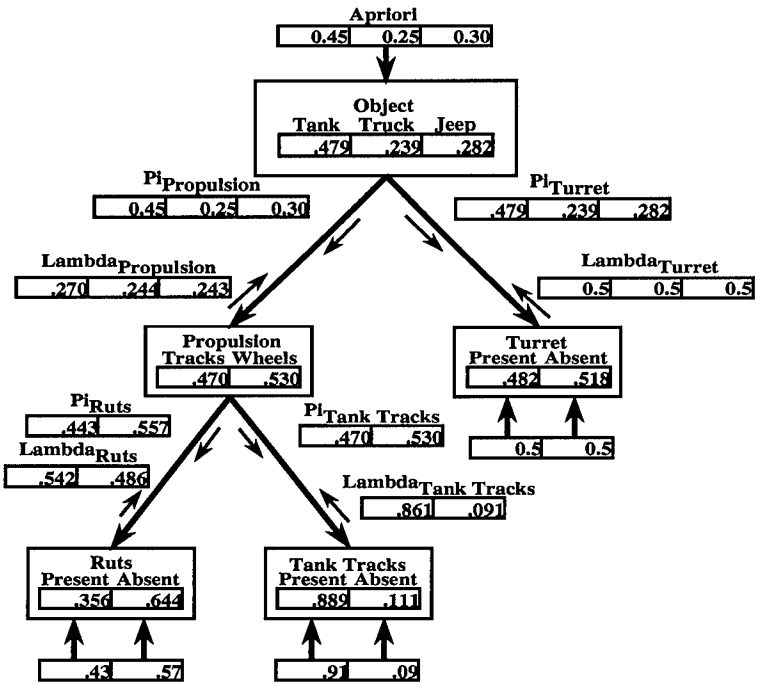


Figure 2.16 Update to the Tank Tracks Node

message vector to the Object node, and a π_{Ruts} message vector to the Ruts node. Figure 2.17 displays the results of the Propulsion node update.

A comparison with Figure 2.16 shows the strong diagnostic support forwarded by the Tank Tracks node manifests itself in a significant increase in the belief value for Tracks in the Propulsion node. This increased belief value is also effected by the weak diagnostic support for ruts and anticipatory support from the Object node that supports Tracks. The Propulsion node communicates its new belief vector through new λ and π message vectors. The $\lambda_{Propulsion}$ message vector sends strong diagnostic support for the hypothesis Tank to the Object node. The new π_{Ruts} message vector converts the strong diagnostic support for tank tracks increasingly weak anticipatory support for the presence of ruts.

The vector messages produced by the update to the Propulsion node stimulate the Ruts node and the Object node to update. The two nodes update in parallel and Figure 2.18 displays the results.

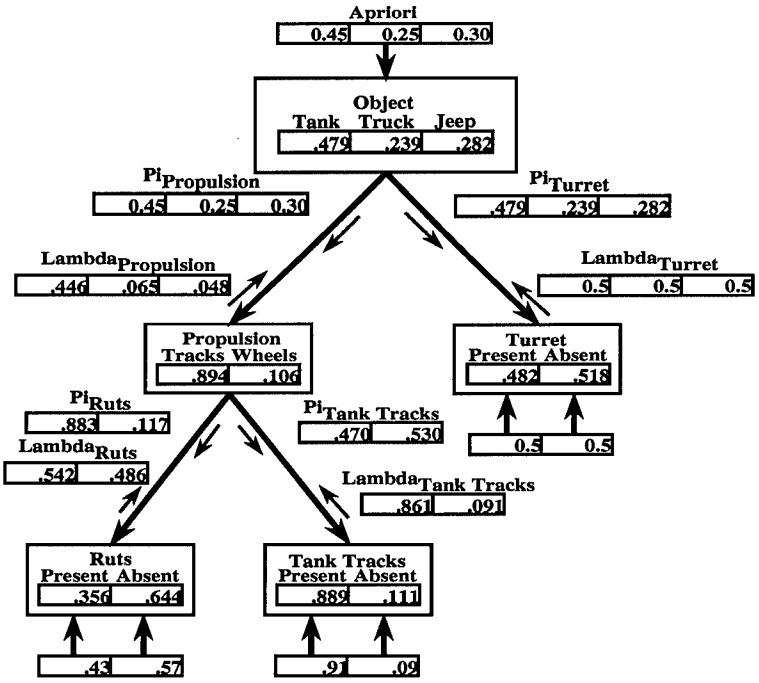


Figure 2.17 Update to the Propulsion Node

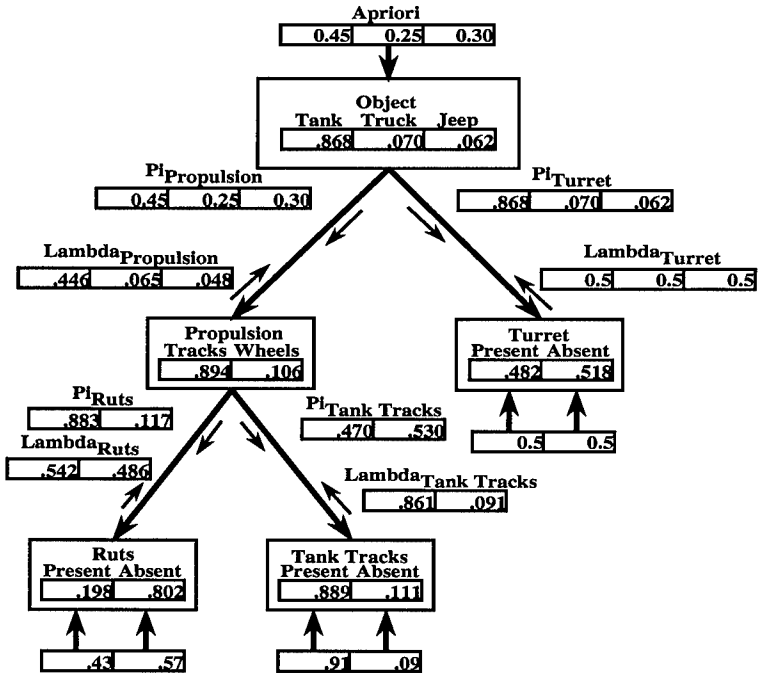


Figure 2.18 Update to the Nodes Ruts and Object

Comparing Figure 2.18 with Figure 2.17 shows an sharp decrease in the Ruts node belief value favoring the presence of ruts and a sharp increase in the Object node belief value favoring tanks. The weak anticipatory support for ruts sent down by the Propulsion node combines with the weak diagnostic evidence for ruts to decrease the belief in ruts to its lowest level. In the Object node, the strength of the strong diagnostic support in hypothesis Tank forwarded by the Propulsion node produces a dramatic increase in the belief the object is a tank. As always, the anticipatory support in terms of the a priori values further strengthens this belief.

This strong diagnostic support for tanks further combines with the a priori values to substantially strengthen the tank entry in the anticipatory support sent to Turret. The new π_{Turret} message vector stimulates the Turret node to update, producing a new Turret belief vector. The strong anticipatory support for tanks forwarded by the Object node pushes the belief in the Turret hypotheses up dramatically. Figure 2.19 displays these results.

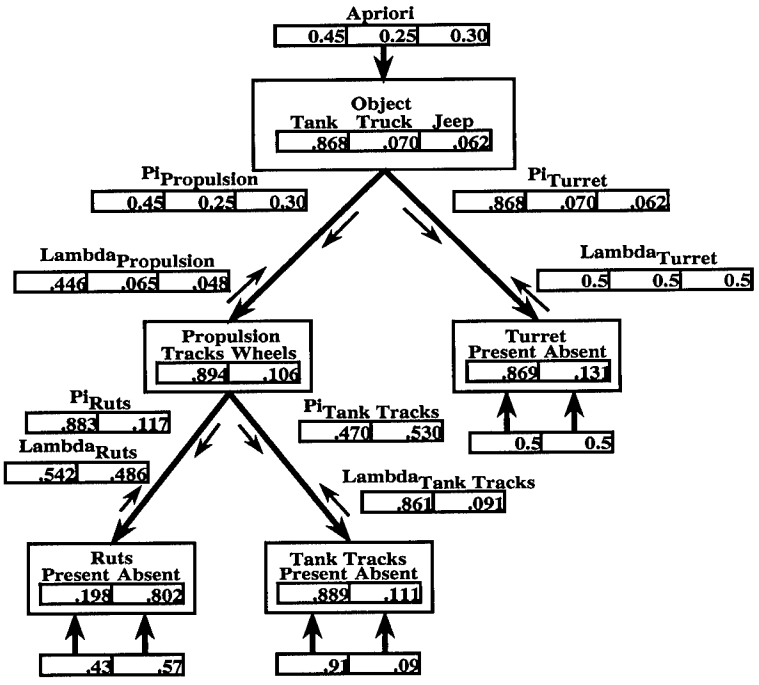


Figure 2.19 The Bayesian Network in a Steady State

As no new message vectors are produced by the update to the Turret node, the propagation of new evidence presented at the Tank Tracks node ends here. The network is again in a steady state. The evidence input to the Tank Tracks node produced large changes in all the belief vectors of the network. Most significantly, the evidence greatly increased the belief in hypothesis Tank and the network as a whole now strongly favors this conclusion.

The network anticipates one last piece of evidence which is given as an input vector of [.77 .23] received at the Turret node. As the steps involved in propagating the new information through the network are similar to the previous two cases, the information is allowed to propagate and all nodes are updated accordingly. Figure 2.20 shows the network in a steady state after the new information has propagated through.

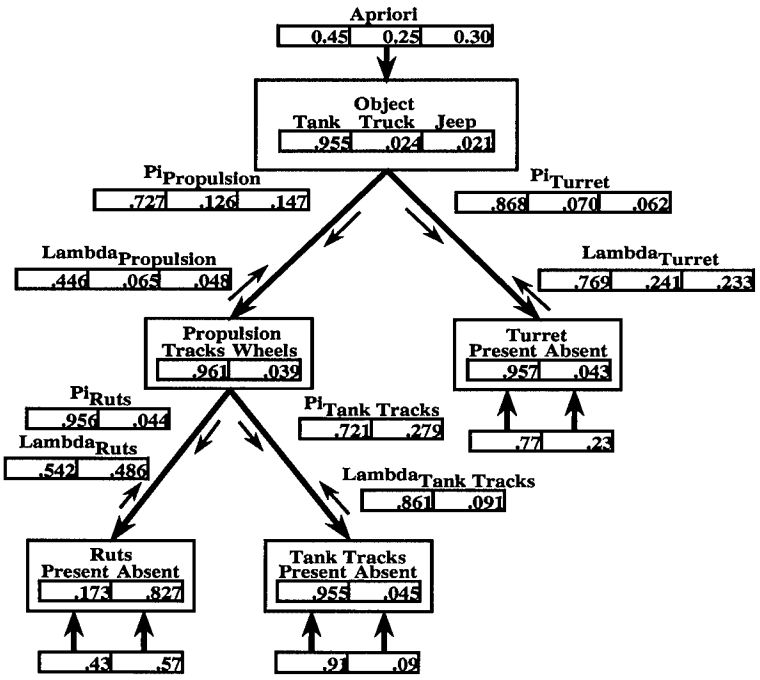


Figure 2.20 Bayesian Network in a Steady State After Receiving All the Evidence

A quick comparison with Figure 2.19 shows how the new information changes the belief vectors. The evidence supports the presence of a turret and, fusing with already strong anticipatory support from the Object node, produces an increase in the associated belief value in the Turret hypothesis. The Turret node forwards this result to the Object node in the form of

strong λ message vector support for the tank hypothesis. The Object node combines this new diagnostic support with the diagnostic support forwarded from the Propulsion node and the a priori vector to produce another increase in the belief value for hypothesis Tank. The Object node also converts the diagnostic support to anticipatory support and sends it via π message vectors to the Propulsion, Ruts, and Tank Tracks nodes. This results in further increases to the belief vector elements for tracks and the presence of tank tracks, while further decreasing the belief in the presence of ruts. Thus all the nodes in the network are effected by the new evidence.

With all evidence received, the network clearly comes to the conclusion that the object under consideration is a tank.

The evidence presented to the network of Figure 2.20 foreshadows the network's tank conclusion. The evidence vectors clearly supported the tank hypothesis. Figure 2.21 and Figure 2.22 show the same Bayesian network presented with different evidence vectors. The networks are shown in their steady state after all information has propagated through. For these examples, the evidence nodes receive the evidence simultaneously.

The evidence vectors for Figure 2.21 support the hypotheses for both wheeled vehicle hypotheses, Jeep and Truck. The evidence results in strong diagnostic support for the conclusion that the target is either a Jeep or Truck vehicle. This strong diagnostic support is complementary. Combining it with the information stored in the conditional probabilities serves to leverage the belief that the observations represented by the evidence vectors are true. This leverage manifests itself in belief vectors at the evidence nodes that are higher in value than the actual evidence vectors. In other words, the diagnostic support for the lack of a turret reinforces the belief that ruts are present and tank tracks are absent.

Tracking the flow of information through the network, the diagnostic support for ruts flows upward in the network where it combines with a lack of diagnostic support for tank tracks to produce a conclusion at the Propulsion node that the object is a wheeled vehicle. This conclusion is further strengthened by anticipatory support stemming from the lack of evidence in a turret. The Propulsion node converts the wheel conclusion to diagnostic support

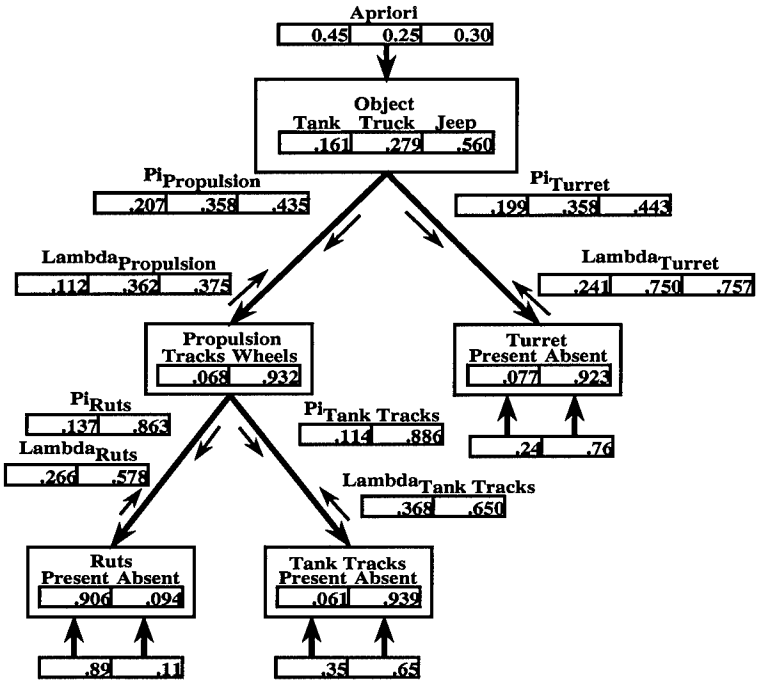


Figure 2.21 Bayesian Network After Receiving Evidence Favoring a Jeep Conclusion

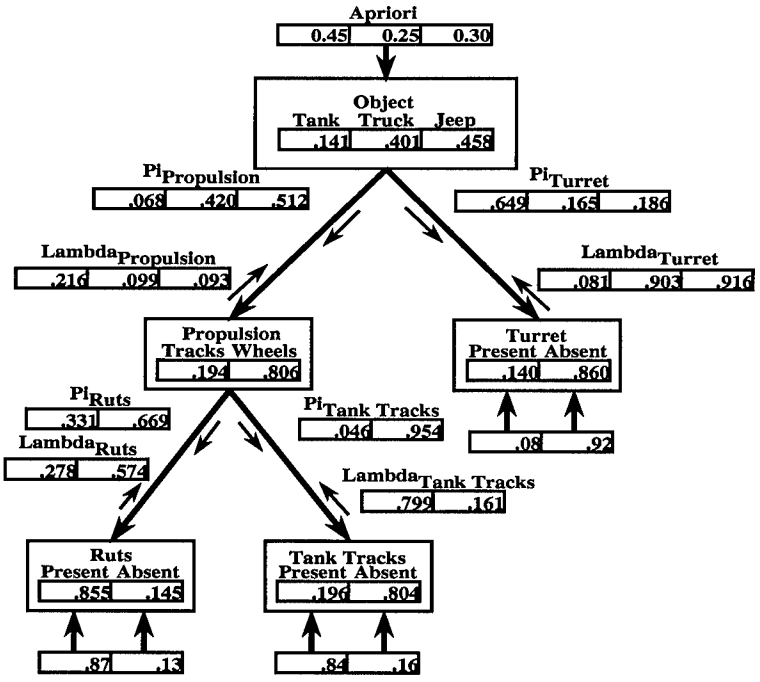


Figure 2.22 Bayesian Network After Receiving Contradictory Evidence

for wheeled vehicles, which the node sends to the Object node via a λ message vector. This diagnostic support combines with the lack of diagnostic support for turret and the a priori values at the Object node to produce a conclusion that the object is a jeep.

Actually, the evidence involved serves only to support a wheeled vehicle hypothesis over the tracked vehicle hypothesis. Only the a priori values and small differences in the conditional probabilities cause the network to favor the Jeep hypothesis over the Truck hypothesis. This observation demonstrates a small flaw in this example. The evidence provided to the network contains no information to distinguish between a Truck hypothesis and a Jeep hypothesis. The network really just distinguishes between tracked and wheeled vehicles. Because the a priori values and conditional probabilities are mostly static, the network would rarely if ever select the Truck hypothesis over the Jeep. To alleviate this flaw, a designer could add another evidence node receiving input for a feature that distinguishes between trucks and jeeps. Examples include a node receiving evidence on the size of the object or a node receiving evidence for the presence of a truck bed. The current lack of ability to distinguish between Truck and Jeep objects is why the belief in the Jeep hypothesis is relatively low. As the evidence presented supports both the Jeep and Truck hypotheses, the network makes its decision only with reservations. Essentially it makes an educated guess.

For the example shown in Figure 2.22, the presented evidence is ambiguous. The evidence vector for tank tracks conflicts with the values of the other two vectors. In this case, the network can produce no clear winner and even the most favored hypotheses have belief values that indicate the network is more inclined to believe the object is none of the above. The network does support the wheeled vehicle hypotheses over the tracked vehicle hypothesis. This is due to the strong dependence of the Tank hypothesis to the presence of a turret. Evidence to the contrary results in a low belief in the Tank hypothesis.

The high conditional probability linking the Tank hypothesis to the presence of a turret has a strong effect throughout the network. As an example, the strong lack of evidence in a turret results in strong anticipatory support for the absence of tank tracks. This support results in elements of the Tank Tracks belief vector approaching the opposite of the evidence

vectors. Essentially, the network suspects the feature extractor responsible for supplying the tank tracks evidence of lying.

The preceding examples begin to raise questions about how the belief vector in the root node varies over the range of possible evidence values. Chapter III presents software tools for calculating and displaying the decision regions of root node hypotheses over all possible input values.

2.3.4 Other Bayesian Network Topologies. The topology of the example Bayesian network displayed in Figure 2.10 to Figure 2.22 is just one of many possible architectures. Most topologies use the same update rules as presented in this chapter. However, architectures with multiple parent nodes require more complicated update rules for calculating π vectors and λ message vectors (11). Figure 2.23 shows a Bayesian network architecture where the node A has two parent nodes.

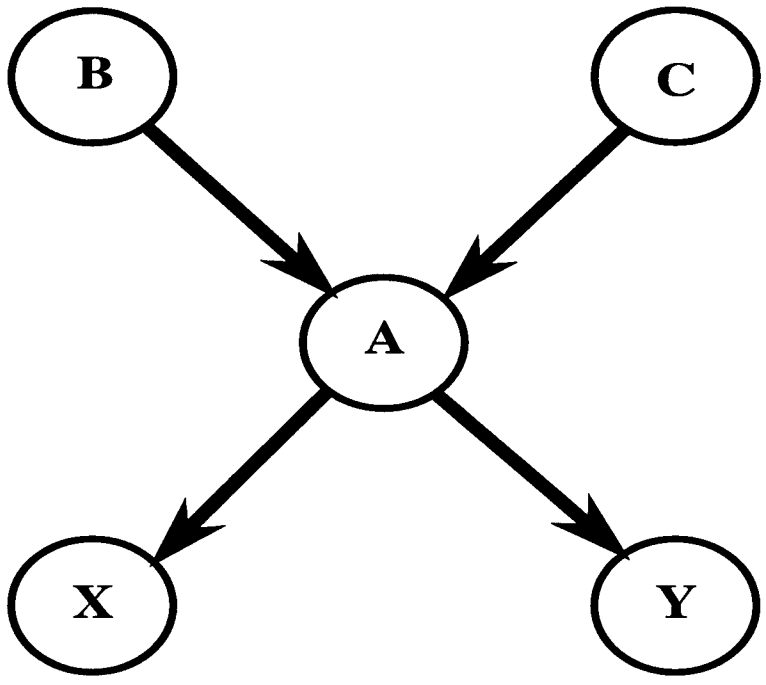


Figure 2.23 A Bayesian Network Featuring a Node with Two Parents

Equation 2.26 presents the expression for calculating the π values for node A.

$$\pi(A_i) = \sum_{jk} P\left[A_i | B_j C_k\right] \pi_A\left(B_j\right) \pi_A(C_k) \tag{2.26}$$

The terms $\pi_A\left(B_j\right)$ and $\pi_A\left(C_k\right)$ of Equation 2.26 refer to elements of the π message vectors transmitted to node A from its parents. The conditional probabilities now associate the hypotheses of node A to the various combinations of hypotheses of nodes B and C. Using Equation 2.26 to calculate π values for all hypotheses in A produces the π vector for A.

The equations for calculating A's λ, Belief and π message vectors are the same as Equations 2.22, 2.21, and 2.25 respectively. The lone difference is the belief vector and π message vector calculations use the π values calculated by Equation 2.26. The equation for calculating the elements of the A node λ message vector differs from Equation 2.24. Equation 2.27 shows the new expression for computing the λ message vector to the B node.

$$\lambda_A(B_i) = \alpha \sum_j \left[\pi_A\left(C_j\right) \sum_k \lambda(A_k) P\left[A_k | B_i C_j\right]\right] \tag{2.27}$$

Equation 2.27 is very similar to Equation 2.24. One difference is the values are now dependent on conditional probabilities that link the node A hypothesis to the various combinations of hypotheses in the B and C nodes. Another difference is the diagnostic support represented by the λ message vector to node B is tempered by the anticipatory support supplied by node C. Also note the entries of the λ message vector are now normalized to sum to one.

Equations 2.27 and 2.26 are easily extended to handle any number of parent nodes.

2.4 Bayesian Networks are NP-Hard

A significant disadvantage of Bayesian networks is that the propagation of information through the network is NP-hard (2). This disadvantage comes into play primarily in large Bayesian networks. As each node is added to the network, the amount of computation required to assess its impact on the nodes already in the network increases by a power of two. This

means that it behooves researchers to keep their Bayesian networks small or the computational resources required to craft a network becomes unwieldy. Researchers, primarily Pearl (11) felt that they the had found a way around this problem through approximating the effect of adding nodes to the network. However, Dagum and Luby recently proved that even these approximations are NP-hard (3). Exceptions to this disadvantage do exist however. A small class of network topologies isolate the effect of new nodes to their immediate neighborhoods. In this manner, the computation required to assess the effect of new nodes can be performed in polynomial time. The Bayesian networks discussed in this work are of this type.

III. Bayesian Network Analysis

3.1 Introduction

This chapter analyzes the decision regions of a Bayesian network designed to classify objects. The Bayesian network under consideration is an approximation of the top two levels of a multi-layer Bayesian network. The goal is to develop software to display the decision regions of the network and show how they vary based on changes to the network's parameters and inputs. The first sections of this chapter present the network and discuss how to establish the required conditional probabilities. Later sections derive the equations for calculating and displaying the decision regions and present several examples.

3.2 A Two Level Bayesian Network

Figure 3.1 shows the basic architecture of the network under consideration. The architecture consists of a two level Bayesian network where the root node has two to four child nodes.

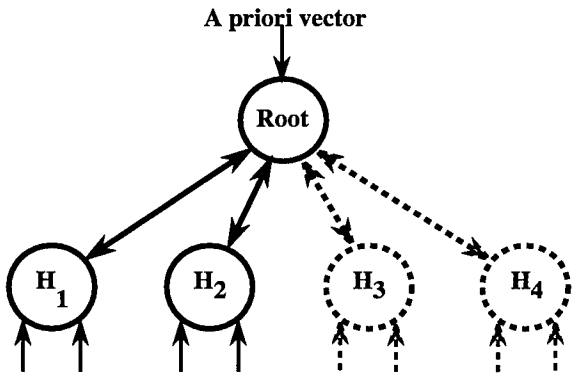


Figure 3.1 The Bayesian Network Analyzed in This Chapter. The Network May Have Between Two and Four Child Nodes

The architecture approximates the top two levels of Martin Marietta's Bayesian network for the ARAGTAP system or any other multi-layer network. The root node is the decision node and contains hypotheses equal to the number of child nodes. The intent of the network

in Figure 3.1 is to simulate the interaction between the root node and its children for any Bayesian network with two or more levels. For clarity and simplicity, the rest of this chapter refers to Figure 3.1 as a simulation network and the root node's children as H nodes. Further, a simulation network with n nodes and n hypotheses is referred to as a rank N Bayesian network.

3.3 The Confusion Matrix

The forthcoming analysis of the Bayesian network decision regions benefits greatly from a new representation for conditional probability values. This representation consists of a single matrix that holds all the conditional probability values required to relate the root node hypotheses to the H node hypotheses. The benefit of this new matrix is a reduction in size and complexity of the tables required to specify the conditional probabilities. As each new node added to a network requires a new conditional probability table, the number and size of these tables quickly becomes unwieldy. However, for the case where $m = 2$ for all H nodes, the network can use a single matrix that holds all the conditional probability values necessary. This can occur because the conditional probability table relating n root node hypotheses to two H node hypotheses has two rows and n columns where the second row entries equal one minus the first row entries. Thus, the only data that must be stored is the first row of the conditional probability tables. These rows are mapped to the columns of the new matrix. For reasons explained later, the new matrix is called a confusion matrix. Figure 3.2 illustrates the mapping of the new matrix into three conditional probability tables required for a rank 3 Bayesian network.

The name confusion matrix is a bit misleading and requires some explanation. The concept for representing conditional probabilities in this manner arises from the Martin Marietta ARAGTAP work. In their Bayesian network, each H node heads a sub-graph dedicated to collecting evidence supporting one and only one hypothesis. The H node serves as a collection point to fuse the support for a hypothesis and forward it to the root node. Because of this, the confusion matrix values take on the added semantic meaning of representing how similar two hypotheses are to each other. Thus, a large value in entry i,j of the confusion matrix

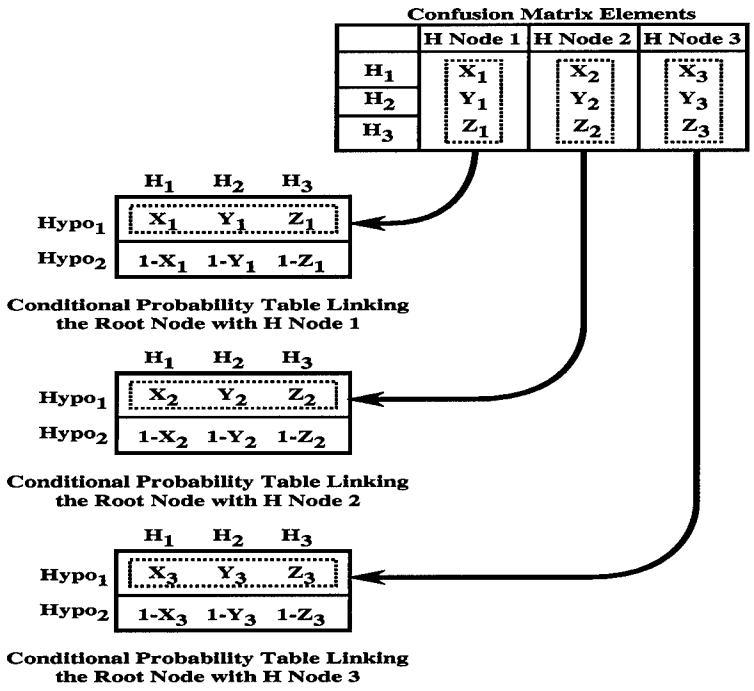


Figure 3.2 Mapping the Columns of the Confusion Matrix into the Rows of the Conditional Probability Tables

indicates that root node hypothesis i and hypothesis j produce very similar feature measurements. A small value in entry i,j indicate the two hypotheses are very dissimilar. Another way to interpret the matrix entry is as a measure of how easily a classifier can confuse feature measurements for hypothesis i with those of hypothesis j. Hence the name confusion matrix. When the confusion matrix representation was imported for this work, the name stuck, even though the related semantic interpretation did not.

In summary, the confusion matrix is an excellent representation because it concisely holds all the information necessary to populate the conditional probability tables for any simulation network. Each column of the confusion matrix maps to the first row of the conditional probability table associating the root node with an H node. Because the columns of the conditional probability tables must sum to one, the entries for the second row of the tables are just one minus the first row entry. The mapping has the limitation that all the H nodes must have two and only two hypotheses.

The confusion matrix supplies all the information necessary to instantiate the Bayesian network architecture of Figure 3.1. Therefore, the forthcoming decision region analysis will key on variations to network parameters in terms of the elements of the confusion matrix. In other words, the analysis will vary the entries in the confusion matrix and show how the decision regions change.

3.4 Assumptions and Modifications

To ensure the relevance of the analysis to all Bayesian networks, it is paramount the simulation network faithfully approximate the interaction between the top two levels of a Bayesian network. The root node receives all its inputs precisely as the root node in an actual multi-level network. The root node in Figure 3.1, therefore, should exhibit the behavior of a root node in any network. The H nodes, however, differ dramatically.

The H nodes in a Figure 3.1 are not fed by inputs from other nodes as in multi-level networks. The central concern, therefore, is to ensure the H nodes of Figure 3.1 accurately approximate the H nodes of a multi-level network. Two assumptions assure this is the case. First, the simulation must begin at the level of a multi-level network's H nodes after the nodes have fused the λ message vectors from the H node's children. Hence, the input vectors to the leaf nodes of Figure 3.1 represent the λ vectors of the H nodes in a multi-level network. In this manner, the network of Figure 3.1 can simulate the interactions between the root and H nodes over all values for the H node λ vectors without concerning itself with the underlying network that produced the vectors. In other words, the information achieved by Equation 2.22 essentially maps the multiple λ message vectors of a multi-level network into an input vector in the simulated network. The simulation begins at this point.

The second assumption is that any node can normalize the elements of the λ vector, or any vector in the Bayesian network, without altering the performance of the network and without any loss of generality. This assumption concerns the previous edict that the elements of the input vector must sum to one. This edict is critical to the simulation network as it greatly simplifies any analysis. However, the elements of a node's λ vector do not sum to

one, requiring the simulation to normalize the vector. This operation is not specified in Pearl's original definitions and makes the impact suspect. Thus, it must be shown that normalizing the λ vector does not effect the operation of the Bayesian network. The informal argument is that it is the differences in the element values that convey the required information and not the elements' exact values, allowing the network to normalize the vector. The following equations present a more formal argument. The equations assume a normalized λ vector at an H node.

$$\lambda_N = \frac{1}{\alpha_{\lambda_k}} \lambda_k \tag{3.1}$$

Equation 3.1 expresses the normalized λ vector where α_k is the summation of the elements of λ_k and λ_N is the normalized λ vector for node H_k. Equation 3.2 calculates the λ message vector, using the normalized λ vector.

$$\begin{aligned} \lambda_{H_k}(A_i) &= \sum_j P\left[B_j|A_i\right] \frac{1}{\alpha_{\lambda_k}} \lambda_k\left(B_j\right) \\ &= \frac{1}{\alpha_{\lambda_k}} \sum_j P\left[B_j|A_i\right] \lambda_k\left(B_j\right) \end{aligned} \tag{3.2}$$

This results in the λ message vector expressed in Equation 3.3

$$\begin{aligned} \lambda_{H_k} &= \left[\frac{1}{\alpha_{\lambda_k}} \sum_j P\left[B_j|A_1\right] \lambda_k\left(B_j\right), \ldots, \frac{1}{\alpha_{\lambda_k}} \sum_j P\left[B_j|A_n\right] \lambda_k\left(B_j\right)\right] \\ &= \frac{1}{\alpha_{\lambda_k}} \left[\sum_j P\left[B_j|A_1\right] \lambda_k\left(B_j\right), \ldots, \sum_j P\left[B_j|A_n\right] \lambda_k\left(B_j\right)\right] \\ &= \frac{1}{\alpha_{\lambda_k}} \hat{\lambda}_{H_k} \end{aligned} \tag{3.3}$$

The $\hat{\lambda_{H_k}}$ expression in Equation 3.3 represents the unnormalized λ message vector. Equation 3.4 fuses the λ message vectors at the parent node and produces λ_R, the parent node λ vector. The vector to vector multiplication in the following equations represents element by element multiplication and the $\frac{1}{\alpha_R}$ term normalizes the new vector.

$$\begin{aligned}
\lambda_R &= \frac{1}{\alpha_R}\prod_k \lambda_{H_k} \\
&= \frac{1}{\alpha_R}\prod_k \frac{1}{\alpha_{H_k}}\hat{\lambda_{H_k}} \\
&= \frac{1}{\alpha_R}\prod_k \frac{1}{\alpha_{H_k}}\prod_k \hat{\lambda_{H_k}} \\
&= \frac{1}{\beta}\prod_k \hat{\lambda_{H_k}} \qquad (3.4)
\end{aligned}$$

Thus, in Equation 3.4, all the normalization constants are subsumed into a new normalization constant $\frac{1}{\beta}$.

Equation 3.5 uses the parent node λ vector to compute the belief vector.

$$\begin{aligned}
Belief &= \frac{1}{\alpha}\pi_R\lambda_R \\
&= \frac{1}{\alpha}\pi_R\frac{1}{\beta}\prod_k \hat{\lambda_{H_k}} \\
&= \frac{1}{\alpha\beta}\pi_R\prod_k \hat{\lambda_{H_k}} \qquad (3.5)
\end{aligned}$$

Thus, the calculations for deriving the elements of the belief vector are exactly the same as the unnormalized case except for an additional normalization term β. This term merely accounts for the normalization that occurred in previous calculations resulting in no change to the values of the belief vector elements. Additionally, the derivation of Equation 3.4 shows that it does not matter if some of the λ vectors are normalized while others are not. Any vectors not normalized will merely have a α_{H_k} value of one. This equation also shows that

normalizing the λ vector messages does not have an effect on the belief vector elements either. Thus, the network may selectively normalize any vector in the network to its advantage without effecting the final results.

This is an important result because selectively normalizing and not normalizing λ vectors and λ vector messages greatly simplifies the analysis of the simulation network in subsequent sections. Also, Bayesian network displays can list these vectors in normalized form making it easier to grasp their significance at a glance. Indeed, Figure 3.3 shows the results of a Bayesian network example from Chapter A where the λ vector and λ vector messages have been normalized. A quick glance back to Figure 2.20 shows the final belief vectors are identical. However, it is now easier to grasp the semantic meaning behind the λ vector messages with the new normalized format.

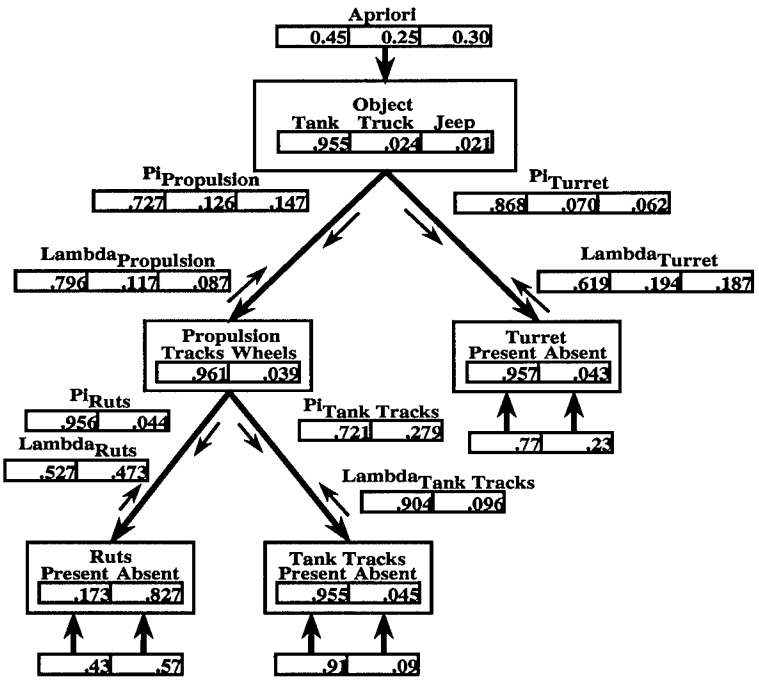


Figure 3.3 An Example Bayesian Network Using Normalized λ Message Vectors

By an analogous argument, the π vectors and π message vectors may or may not be normalized without effecting the final belief vectors of the network. This leads to a third

important result. The network need only normalize the belief vectors and only then as a last step. This results in a great computational savings when implementing a Bayesian network.

The results of the previous paragraphs allow the following observations about the network in Figure 3.1. First, The elements of the input vectors represent the normalized H node λ vectors in a multi-level network. The simulation network may make this assumption without any loss of information or loss of generality. Second, the simulation network need not normalize the λ vector messages to the root node. Third, as a result of observation one and two, the simulation network faithfully approximates the behavior of the top levels of a multi-level network. These observations are important for two reasons. They greatly simplify the forthcoming analysis of the simulation network and the results of the analysis are directly applicable to an implemented Bayesian network classifier, namely the Martin Marietta ARAGTAP network, employed to solve real world problems.

One last point to cover is the establishment of the a priori vector required by the network in Figure 3.1. A convention used by Martin Marietta is to assume the hypotheses of the root node are all equally probable. This results in an a priori vector with all elements equal to $1/n$ where n is the number of hypotheses. The simulation network in this research uses this a priori vector as well.

3.5 Analysis of Evidence Nodes

Using the observations of Section 3.4, this section derives equations that link the elements of the simulation network input vectors to the λ message vectors. Section 3.6 uses results from this section to derive equations relating the elements of the input vectors to the elements of the root node belief vector. To begin, Figure 3.4 illustrates a rank 2 Bayesian network receiving input values. This network form the context for subsequent discussion. The input values are the individual elements of a normalized input vector.

The network of Figure 3.4 is generic, meaning no semantic meaning is attached to the nodes, the hypotheses in the nodes, or the inputs. The characteristics of the network that are important to the forthcoming derivation is that all nodes have two and only two hypotheses.

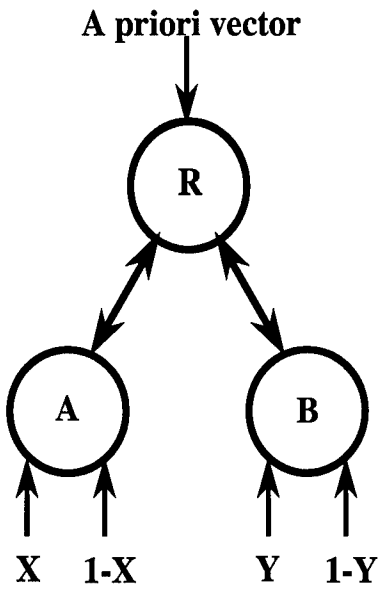


Figure 3.4 A Rank 2 Bayesian Network Receiving Input Values

Further, the input vectors are normalized to one and the variables X and Y are constrained such that $0 \leq X \leq 1$ and $0 \leq Y \leq 1$. For clarity, the root node is labelled R and its hypotheses are R_1 and R_2. The two input nodes are A and B with similarly associated hypotheses.

Note that the elements of an input vector in Figure 3.4 are now entirely defined in terms of one variable. Relating the elements of the λ message vectors to the input variables X and Y requires deriving the λ_A message vector in terms of X and the λ_B message vector in terms of Y. The following equations set forth the derivations of the λ message vectors in terms of their respective inputs and do so on an element by element basis.

For the λ_A message vector, applying Equation 2.24 results in Equation 3.6.

$$\begin{aligned} \lambda_A(R_1) &= \sum_j P[A_j|R_1]\,\lambda(A_j) \\ &= P[A_1|R_1]\lambda(A_1) + P[A_2|R_1]\lambda(A_2) \end{aligned} \tag{3.6}$$

Because the A node is an evidence node, the elements of the λ vector are equal to the elements of the input vector. Thus, X and 1-X replace the references to the λ vector elements in Equation 3.6. Also, the conditional probabilities used by Equation 3.6 make up a column in the conditional probability table and sum to one. Thus $1 - P[A_1|R_1]$ replaces $P[A_2|R_1]$. Equation 3.7 displays the results of these replacements.

$$\begin{aligned} \lambda_A(R_1) &= P[A_1|R_1]X + P[A_2|R_1](1-X) \\ &= P[A_1|R_1]X + (1 - P[A_1|R_1])(1-X) \\ &= (2 \times P[A_1|R_1] - 1)X + (1 - P[A_1|R_1]) \end{aligned} \tag{3.7}$$

A similar derivation produces Equation 3.8

$$\lambda_A(R_2) = (2 \times P[A_1|R_2] - 1)X + (1 - P[A_1|R_2]) \tag{3.8}$$

Equations 3.7 and 3.8 show that the diagnostic support sent from a two hypothesis child node is a linear relation on the elements of the node's λ vector, which is equivalent to the input vector for a leaf node. Further, there is one linear relation per hypothesis in the parent node, and the relations are a functions of one element of the λ vector and one conditional probability. Equation 3.9 shows the λ_A message vector in terms of Equations 3.7 and 3.8.

$$\begin{aligned} \lambda_A = \; & [(2 \times P[A_1|R_1] - 1)X + (1 - P[A_1|R_1]) \\ & (2 \times P[A_1|R_2] - 1)X + (1 - P[A_1|R_2])] \end{aligned} \tag{3.9}$$

Similar derivations produce an analogous Equations for the λ_B message vector.

$$\lambda_B(R_1) = (2 \times P[B_1|R_1] - 1)Y + (1 - P[B_1|R_1]) \quad (3.10)$$

$$\lambda_B(R_2) = (2 \times P[B_1|R_2] - 1)Y + (1 - P[B_1|R_2]) \quad (3.11)$$

$$\lambda_B = [(2 \times P[B_1|R_1] - 1)Y + (1 - P[B_1|R_1]) \\ (2 \times P[B_1|R_2] - 1)Y + (1 - P[B_1|R_2])] \quad (3.12)$$

Equations 3.7 through 3.12 are important because they relate the diagnostic support for a given root node hypothesis to the input vectors. Evaluating the extremes of this relation gives useful insight into the level of diagnostic support for an hypothesis based on the input and how the diagnostic support changes based on changes to the input and the conditional probabilities. Equation 3.7 serves as an example equation. Figures 3.5 shows how the diagnostic support for hypothesis R_1 varies over changes to the input value X for five different values of $P[A_1|R_1]$.

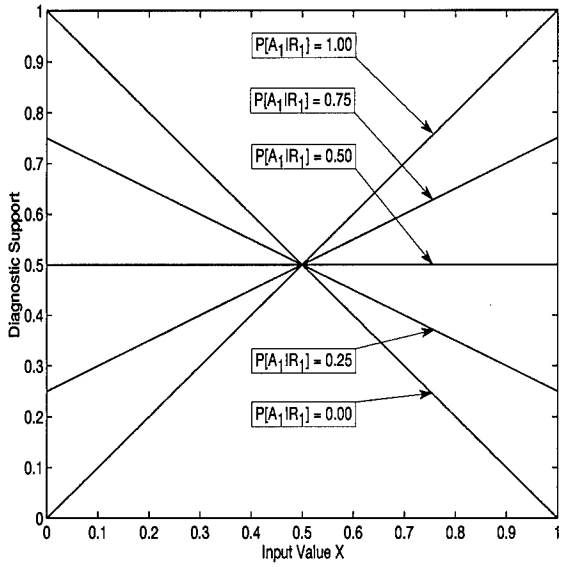


Figure 3.5 Diagnostic Support Versus Input for Several Conditional Probability Values

A quick analysis of Equation 3.7 shows it is a linear equation with slope $(2 * P[A_1|R_1] - 1)$ and intercept $(1 - P[A_1|R_1])$. Thus, any changes to the probability $P[A_1|R_1]$ changes the slope of the relation and the minimum and maximum values. Figures 3.5 bears this analysis

out. For $P[A_1|R_1] = 1$, node A essentially passes the input directly to parent node with no modification. The evidence directly becomes the diagnostic support. For $P[A_1|R_1] = .75$, node A tempers the evidence, decreasing the slope of the relation and limiting minimum and maximum values. Thus, the diagnostic support does not increase as significantly in relation to increases in evidence and the maximum diagnostic support is constrained.

A probability of $P[A_1|R_1] = .5$ essentially cuts node A off from the rest of the network. Changes to the input no longer effect the diagnostic support which stays constant at .5. This is a neutral value in Bayesian networks indicating diagnostic support equivalent to flipping a coin. A .5 conditional probability does not effect the values of the root node belief vector elements.

Probability values less than .5 invert the effect of the evidence. In other words, a probability of less than .5 causes the diagnostic support to decrease as the evidence increases. To what extent the diagnostic support decreases is a function of the magnitude of the probability. As the probability approaches 0, the diagnostic support decreases on a one to one ratio with an increase in evidence.

The fact that probabilities less than .5 decrease the diagnostic support as evidence increases allows designers to add input nodes to a Bayesian network that specifically looks for the absence of a feature. As evidence mounts for a feature, this node decreases the diagnostic support for, and ultimately the belief in, hypotheses that do not have the feature. The Bayesian network of Chapter A serves as an example. The conditional probability linking the presence of a turret to the Object node hypothesis Jeep is .001 or very close to zero. Thus, evidence for a turret effectively quells any diagnostic support for the hypothesis Jeep.

One last observation about the linear relationship between input and diagnostic support concerns the input value of .5. Chapter II made the claim that an input value of .5 served as the identity value for a Bayesian network, meaning this input has no effect on the belief values of the root node. Equation 3.13 supplies an X input of .5 to Equation 3.9 and shows why.

$$
\begin{aligned}
\lambda_A &= [(2 \times P[A_1|R_1] - 1).5 + (1 - P[A_1|R_1]) \\
&\quad (2 \times P[A_1|R_2] - 1).5 + (1 - P[A_1|R_2])] \\
\lambda_A &= [P[A_1|R_1] - .5 + 1 - P[A_1|R_1], \\
&\quad P[A_1|R_2] - .5 + 1 - P[A_1|R_2]] \\
\lambda_A &= [.5, .5] \qquad (3.13)
\end{aligned}
$$

Chapter II also showed that a λ message vector of equal inputs are lost during normalization and have no effect on the receiving node. Thus, inputs of .5 have no effect on Bayesian networks and can be used as the network's null input.

3.6 Analysis of Decision Regions

This section continues the derivation begun in Section 3.5 by relating the derived λ message vectors to the root node belief vector. The resulting relation is central to Matlab script files that display the decision regions of Bayesian networks.

To begin, recall the root node of Figure 3.4 receives the λ message vectors expressed in Equations 3.9 and 3.12 and combines them via Equation 2.22. Note that this derivation assumes the λ message vectors are not normalized nor is the resulting λ vector. These assumptions greatly simplify the derivation. Equations 3.14 and 3.15 calculate the elements of the λ vector.

$$
\begin{aligned}
\lambda(R_1) &= \lambda_A(R_1) \times \lambda_B(R_1) \\
\lambda(R_1) &= (2 \times P[A_1|R_1] - 1)X + (1 - P[A_1|R_1]) \\
&\quad \times (2 \times P[B_1|R_1] - 1)Y + (1 - P[B_1|R_1]) \qquad (3.14) \\
\lambda(R_2) &= \lambda_A(R_2) \times \lambda_B(R_2)
\end{aligned}
$$

$$\begin{aligned}\lambda(R_2) &= (2 \times P[A_1|R_2] - 1)Y + (1 - P[A_1|R_2]) \\ &\times (2 \times P[B_1|R_2] - 1)Y + (1 - P[B_1|R_2]) \end{aligned} \tag{3.15}$$

After calculating the λ vector, the root node multiplies it element by element with the π vector and normalizes to produce the elements of the belief vector. As the elements of the π vector are equal to each other, the normalization has the effect of dividing them out the final belief vector element values. Thus, this derivation does not consider the π vector further. The calculations required to derive the elements of the belief vector appear in Equations 3.16 and 3.17. Equation 3.18 relates the final belief vector in terms of the λ vector.

$$Belief(R_1) = \frac{\lambda(R_1)}{\lambda(R_1) + \lambda(R_2)} \tag{3.16}$$

$$Belief(R_2) = \frac{\lambda(R_2)}{\lambda(R_1) + \lambda(R_2)} \tag{3.17}$$

$$Belief = [\frac{\lambda(R_1)}{\lambda(R_1) + \lambda(R_2)}, \frac{\lambda(R_2)}{\lambda(R_1) + \lambda(R_2)}] \tag{3.18}$$

Substituting Equation 3.14 for $\lambda(R_1)$ and Equation 3.15 for $\lambda(R_2)$ in Equation 3.18 results in a relation between the belief vector element values and the evidence inputs. Equation 3.18, therefore, is the central equation for software that calculates the decision regions for a rank 2 Bayesian network classifier and displays them graphically.

3.6.1 Decision Regions for a Rank 2 Bayesian Network. This section presents software for displaying the decision regions of a Bayesian network designed to discriminate between two root node hypotheses based on evidence from two sources. The software consists of Matlab script files. These files are well documented and included in Appendix B. This section only briefly skims the algorithm for computing the decision regions and concentrates more on analyzing the resulting displays.

Equation 3.18 represents the central equation in the software and calculates the values of the belief vector elements as a function of the network inputs and conditional probabilities. The algorithm for computing the decision regions compares the belief vector elements and identifies the highest value. It then maps a specific color to a 2-D region based on which hypothesis currently possesses the highest belief. The 2-D region ranges over all possible input values.

As an example, Figure 3.6 shows the decision region graph produced by the Matlab script file make2D.m. Figure 3.1 is the representative Bayesian network for this example.

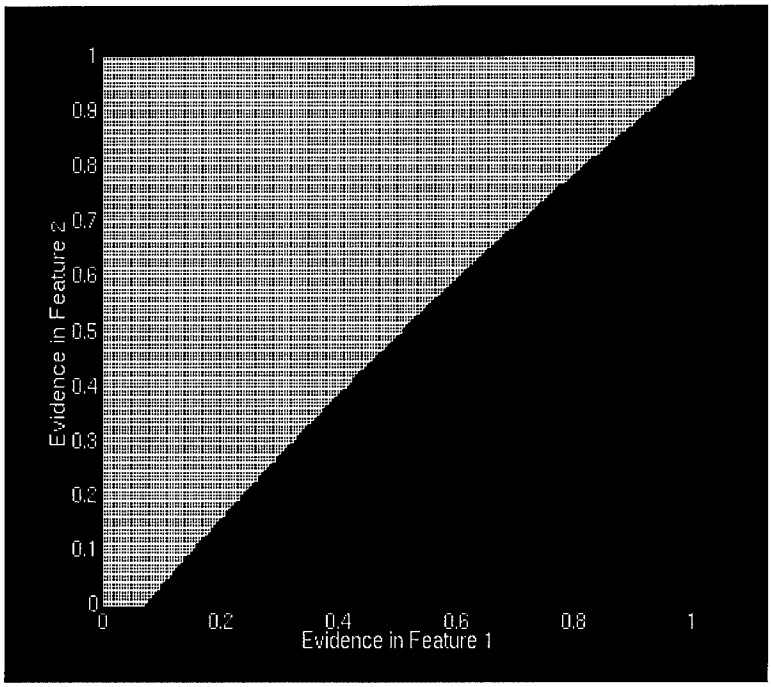


Figure 3.6 The Decision Regions Resulting from the Conditional Probability Values in Matrix C

The conditional probabilities that formed these decision regions are random. Matrix C expresses these probabilities in a confusion matrix format.

$$C = \begin{bmatrix} .9 & .35 \\ .4 & .83 \end{bmatrix}$$

The key at the top of the Figure 3.6 displays which decision region corresponds to which hypothesis. Moving from left to right, from lighter colors to darker, the colors map to the decision region for hypothesis one, decision region for hypothesis two, and a third color representing where the elements of the belief vector are equal. Regions of equal color indicate the input values for which the network favors the related hypothesis. As an example, the graph in Figure 3.6 indicates that the network of Figure 3.1 characterized by confusion matrix C will always produce a higher belief value for hypothesis R_1 for inputs ranging from .7 to 1 on both evidence nodes. The graph in Figure 3.6 is just one of many possible graphs dependent on the conditional probabilities. The Matlab script file that produced Figure 3.6, make2D.m, will produce the decision regions for any combination of confusion matrix and a priori vector.

The line that separates the decision regions in Figure 3.6 is important because it forms the boundaries between the decision regions. The equation for this line results from setting Equation 3.16 equal to Equation 3.17 and solving for y. Equation 3.19, expressed in terms of the elements of the confusion matrix, results.

$$y = \frac{x(d-c)+(h-g)}{x(a-b)+(e-f)} \tag{3.19}$$

where

$$a = (2\times C_{11}-1)\times(2\times C_{12}-1)$$

$$b = (2\times C_{21}-1)\times(2\times C_{22}-1)$$

$$c = (2\times C_{11}-1)\times(1-C_{12})$$

$$d = (2\times C_{21}-1)\times(1-C_{22})$$

$$e = (2\times C_{12}-1)\times(1-C_{11})$$

$$f = (2\times C_{22}-1)\times(1-C_{21})$$

$$g = (1-C_{11})\times(1-C_{12})$$

$$h = (1-C_{21})\times(1-C_{22})$$

Equation 3.19 leads to some interesting generalizations.

If $C_{11} = C_{22}$ and $C_{12} = C_{21}$, the boundary becomes y = x.

If $C_{11} = C_{21}$ and $C_{12} = C_{22}$ the boundary is undefined and the decision regions are equal everywhere. In this case, the probabilities linking an hypothesis to evidence for or against the hypothesis are equal and the network can make no decision.

If $C_{11} = C_{21}$, the boundary becomes y = .5. This is a case where the evidence for and against feature A is equally weighted. It adds nothing to the final belief which becomes totally dependent on y.

If $C_{12} = C_{22}$, the boundary becomes x = .5. This is a case where the evidence for and against feature B is equally weighted. It adds nothing to the final belief which becomes totally dependent on x.

These observations have the important limitation that the elements of the a priori vector be equal.

Applying a limitation used by Martin Marietta simplifies Equation 3.19 and allows further analysis. Martin Marietta sets the on-diagonal elements of the confusion matrix to one. This limitation simplifies Equation 3.19 substantially. Equation 3.20 expresses the new relation in terms of confusion matrix M.

$$M = \begin{bmatrix} 1 & M_{12} \\ M_{21} & 1 \end{bmatrix}$$

$$y = \frac{(M_{12} - 1)x}{2x(M_{12} - M_{21}) + 1 - M_{12}} \tag{3.20}$$

Equation 3.20 leads to a relation between the decision regions of different confusion matrices. If Equation 3.21 holds, then the Bayesian network decision regions characterized by confusion matrices I and G will be equal.

$$\frac{G_{21} - 1}{G_{12} - 1} = \frac{I_{21} - 1}{I_{12} - 1} \tag{3.21}$$

Equation 3.21 results from equating the decision region boundary line equations for matrices G and I. The following expressions show the derivation of Equation 3.21.

$$\frac{(G_{12}-1)x}{2x(G_{12}-G_{21})-(G_{21}-1)} = \frac{(I_{12}-1)x}{2x(I_{12}-I_{21})-(I_{21}-1)}$$
$$\frac{2x(G_{12}-G_{21})-(G_{21}-1)}{(G_{12}-1)} = \frac{2x(I_{12}-I_{21})-(I_{21}-1)}{(I_{12}-1)}$$
$$\frac{2x(G_{12}-G_{21})}{(G_{12}-1)} - \frac{(G_{21}-1)}{(G_{12}-1)} = \frac{2x(I_{12}-I_{21})}{(I_{12}-1)} - \frac{(I_{21}-1)}{(I_{12}-1)}$$

Thus, to make the decision regions equal, the x coefficients and the intercepts on both sides must be equal.

$$\frac{(G_{21}-1)}{(G_{12}-1)} = \frac{(I_{21}-1)}{(I_{12}-1)}$$
$$\frac{2(G_{12}-G_{21})}{(G_{12}-1)} = \frac{2(I_{12}-I_{21})}{(I_{12}-1)}$$

However, as shown below, the equalities expressed above are exactly the same.

$$\frac{2(G_{12}-G_{21})}{(G_{12}-1)} = \frac{2(I_{12}-I_{21})}{(I_{12}-1)}$$
$$\frac{G_{12}-1-G_{21}+1}{(G_{12}-1)} = \frac{I_{12}-1-I_{21}+1}{(I_{12}-1)}$$
$$\frac{(G_{12}-1)-(G_{21}-1)}{(G_{12}-1)} = \frac{(I_{12}-1)-(I_{21}-1)}{(I_{12}-1)}$$
$$1-\frac{(G_{21}-1)}{(G_{12}-1)} = 1-\frac{(I_{21}-1)}{(I_{12}-1)}$$
$$\frac{(G_{21}-1)}{(G_{12}-1)} = \frac{(I_{21}-1)}{(I_{12}-1)}$$

Equation 3.21 means that the confusion matrix characterizing a rank 2 Bayesian network can be converted to another matrix without changing the decision regions. As an example, matrix I is converted to matrix G where G_{12} is zero.

$$I = \begin{bmatrix} 1.000 & 0.300 \\ 0.400 & 1.000 \end{bmatrix} \Rightarrow \begin{bmatrix} 1.000 & 0 \\ 0.143 & 1 \end{bmatrix} = G$$

Equation 3.21 does come with a caveat. Solving for any of the variables in the equation may result in a negative number. Negative numbers imply the attempted conversion cannot be done within the semantic context of the Bayesian network.

Figure 3.7 and Figure 3.8 display the decision regions resulting from confusion matrices I and G respectively. The Matlab script file make2D.m produces the regions.

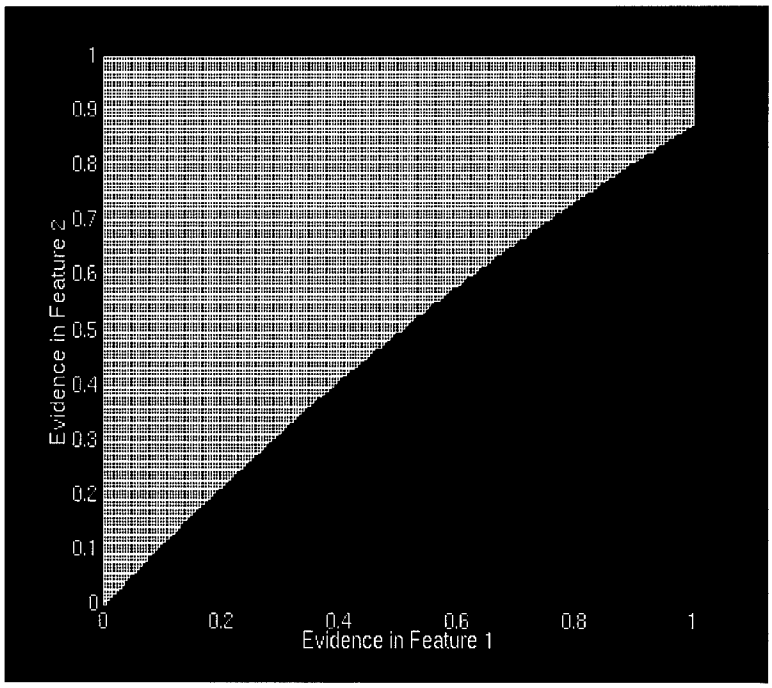


Figure 3.7 The Decision Regions Resulting from the Conditional Probability Matrix I

3.6.2 Belief Values for a Rank 2 Bayesian Network. Matlab script file netbelief2D.m performs similarly to make2D.m except netbelief2D.m produces a three dimensional graph

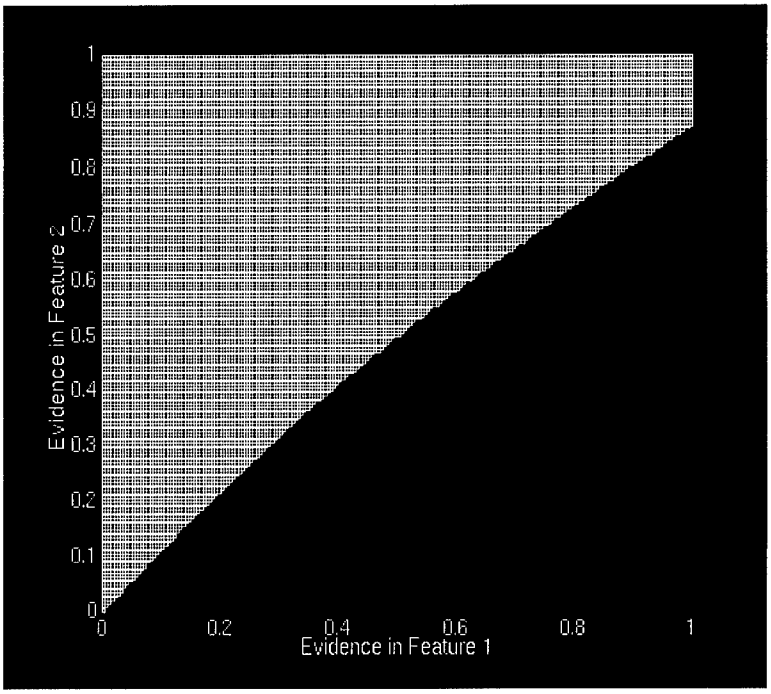


Figure 3.8 The Decision Regions Resulting from the Conditional Probability Matrix G

of the belief vector element values. The display produced by netbelief2D.m is actually two graphs superimposed over each other. The first graph show the belief values for hypothesis R_1 over all possible input values. The second graph similarly shows the belief values for hypothesis R_2. The graphs' intersection corresponds to the decision region boundary. Like make2D.m, netbelief2D.m is based on Equation 3.18.

Figure 3.9 displays the belief values for the rank 2 Bayesian network of Figure 3.1 characterized by confusion matrix C.

The graph produced by netbelief2D.m makes it difficult to ascertain which hypothesis currently dominates. However, overlaying both graphs provides visual cues to how dominate one hypothesis is to the other for given feature inputs. Matrix C dictates that evidence node A favors hypothesis R_1 and evidence node B favors hypothesis R_2. Thus, only when the evidence for feature 1 is relatively high with respect to feature 2 does hypothesis R_1 attain its greatest value. The converse is true for hypothesis R_2. As evidence values recede from the extremes of zero and one, the belief values begin to approach one another. In the vicinity of

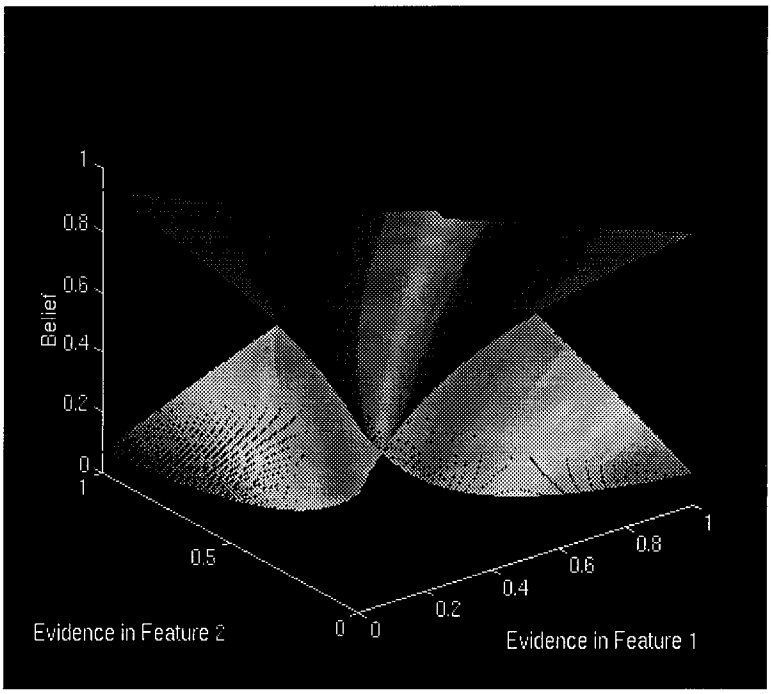


Figure 3.9 Belief Values Resulting from the Conditional Probability Matrix C

the decision region boundary, the network still favors one hypothesis over the other, but only by a narrow margin. The evidence received by the network in these cases allows the network to only slightly favor one hypothesis over the other.

The decision regions for matrices G and I are equal. The same cannot be said for their belief values over all evidence inputs. Figure 3.10 and Figure 3.11 display the belief values resulting from confusion matrices I and G respectively.

The graphs in Figures 3.10 and 3.11 differ substantially from that of Figure 3.9. The ones on the diagonals of matrices I and G account for much of the difference. Ones on the diagonal result in hypothesis R_1 completely dominating R_2 if there is a zero evidence value at the B node. Conversely, hypothesis R_2 dominates if there is a zero evidence value at the A node. The slopes of the graphs are also much steeper indicating the belief values diverge more quickly as evidence values tend toward the extremes.

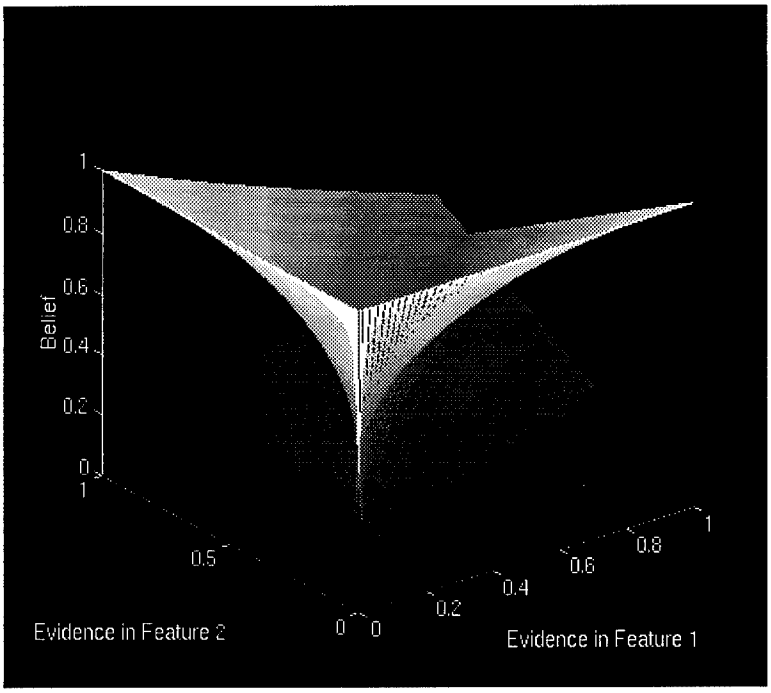


Figure 3.10 Belief Values Resulting from the Conditional Probability Matrix I

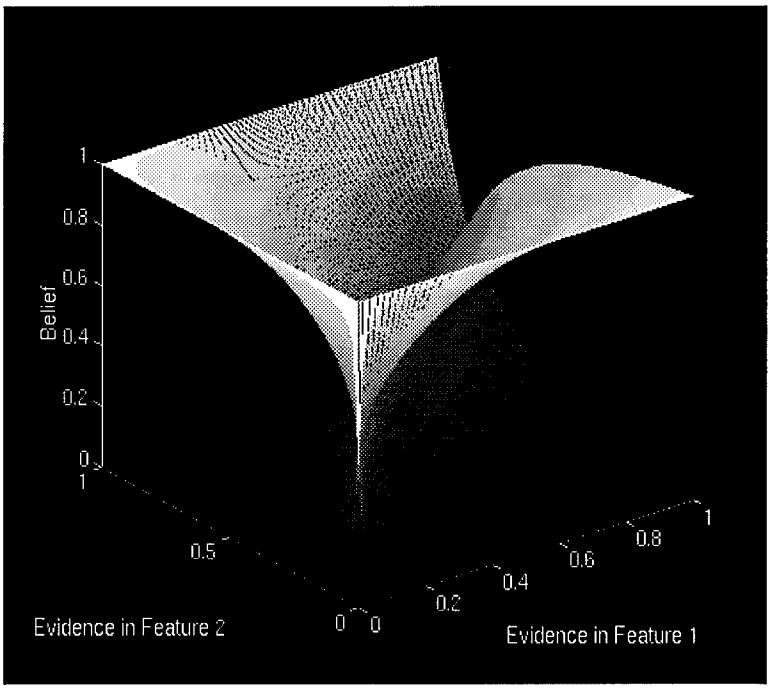


Figure 3.11 Belief Values Resulting from the Conditional Probability Matrix G

The biggest difference between the belief values resulting from matrix I and the belief values resulting from matrix G is the slope of the graphs are steeper for matrix G. Thus, the belief values for a network characterized by matrix G diverge much faster than the belief values for a network characterized by matrix I. From a semantic standpoint, this means the network of matrix G can more easily discern between the two hypotheses even though it shares the same decision regions as the network of matrix I. As a general observation, the greater the difference in values among the the column entries in a confusion matrix, the greater the slope of the resulting belief values. As an informal proof of why this is so, recall that each entry in a confusion matrix associates an hypothesis to the evidence. Values close to one indicate a strong association while values close to zero indicate a weak association. For any value, incoming evidence affords some support for all hypotheses. If the entries in a column are close in value, them approximately equal support is passed on to the hypotheses and the network has difficulty distinguishing them. If, on the other hand, the column values differ greatly, then a great difference in support for one hypothesis over the other results and the network can more easily distinguish between the two. Note the column entries in matrix G have a greater separation than those of Matrix I. As a further example, consider matrices J and K. These matrices result in the belief values displayed in Figure 3.12 and Figure 3.13.

$$J = \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{bmatrix} \quad K = \begin{bmatrix} 1.00 & 0.95 \\ 0.95 & 1.00 \end{bmatrix}$$

The decision regions of the network resulting from matrices J and K are equal. However, the belief value produced by these networks differ greatly. The network characterized by matrix J confidently discerns between the hypotheses at all evidence values except those in close proximity to the decision region boundary. The network characterized by matrix K can only confidently discern between the hypotheses at the extremes.

3.6.3 Movies for a Rank 2 Bayesian Network. The Matlab movie feature leads to a method for visualizing the effects of varying confusion matrix elements. This approach employs time as a fourth dimension to show how the decision regions and belief values vary

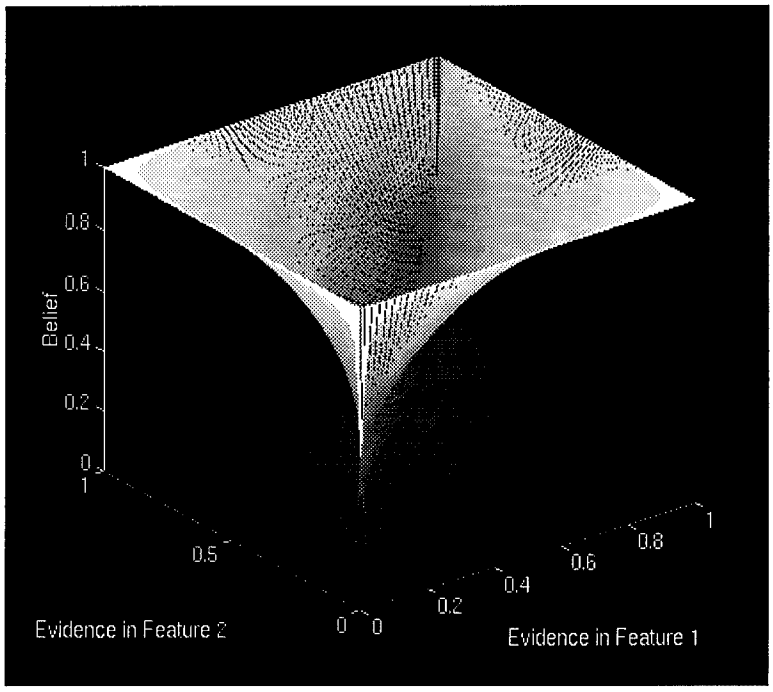


Figure 3.12 Belief Values Resulting from the Conditional Probability Matrix J

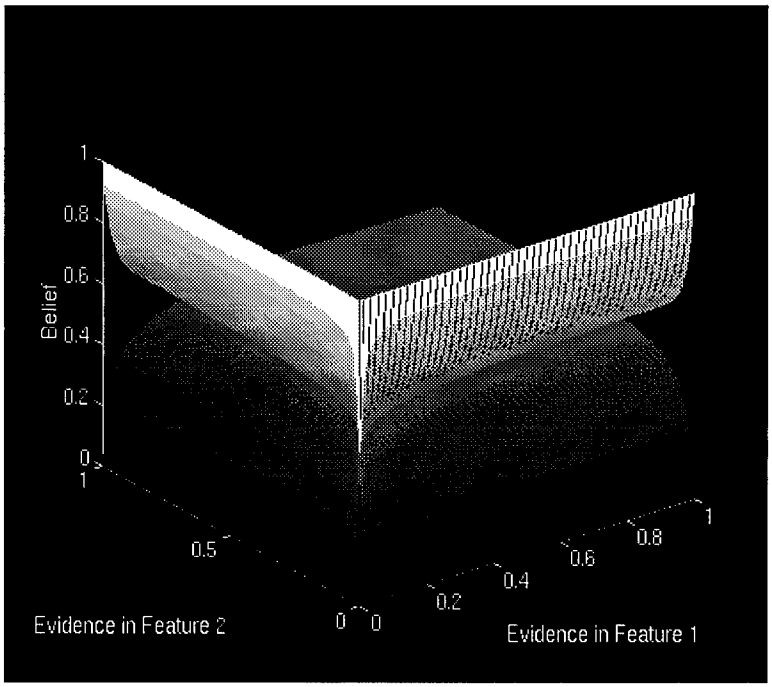


Figure 3.13 Belief Values Resulting from the Conditional Probability Matrix K

over changes in the confusion matrix. Three Matlab script files are available for creating movies. The files capture individual decision region and belief value graphs in a matrix. The Matlab movie command then uses this matrix to display the set of graphs in quick succession. The resulting movie shows how the graphs change as a function of a confusion matrix element.

The script movie2D.m produces a matrix of N decision region graphs as one confusion matrix element varies from zero to one in 1/(N-1) steps. Invoking the movie command on the resulting matrix shows graphically how the decision region changes over the different element values. Figure 3.14 shows nine frames from such a movie. The confusion matrix is C and the varying element is C_{12}. The step size is $1/(N-1) = .125$.

$$C = \begin{bmatrix} .9 & C_{12} \\ .4 & .83 \end{bmatrix}$$

As C_{12} varies, the decision region graphs pass through two reference points. The first occurs when $C_{12} \approx C_{21} \approx .4$. The fourth frame of Figure 3.14 shows the associated decision regions at this point. An earlier generalization showed that when $C_{11} = C_{22}$ and $C_{12} = C_{21}$ the decision region boundary line becomes y = x. Because $C_{11} \approx C_{22}$, the boundary line approximates y = x at $C_{12} = .375$. The other point is at $C_{12} = .75$ and $C_{12} = .875$ shown in the seventh and eighth frames. Another generalization showed that the decision boundary line becomes x = .5 when $C_{12} = C_{22}$. The element C_{12} comes closest to, but not exceeding C_{22} at the value .75. The associated frame shows a boundary line approximating x = .5. For $C_{12} = .875$, the element has passed through the point of equality with C_{22}. Frame eight shows that the boundary line passes from a positive to a negative slope between these points.

The matlab script file belmovie2D.m performs much the same function as movie2D.m except belmovie2D.m produces a matrix of belief value graphs. The program varies an element of the confusion matrix and creates a belief value graph at each value. The program then stores each graph as a movie frame in a matrix. Matlab's movie command uses this matrix to display each frame in quick succession. In this manner, belmovie2D.m displays how the belief values of a rank 2 Bayesian network change as one element of the confusion matrix varies from zero

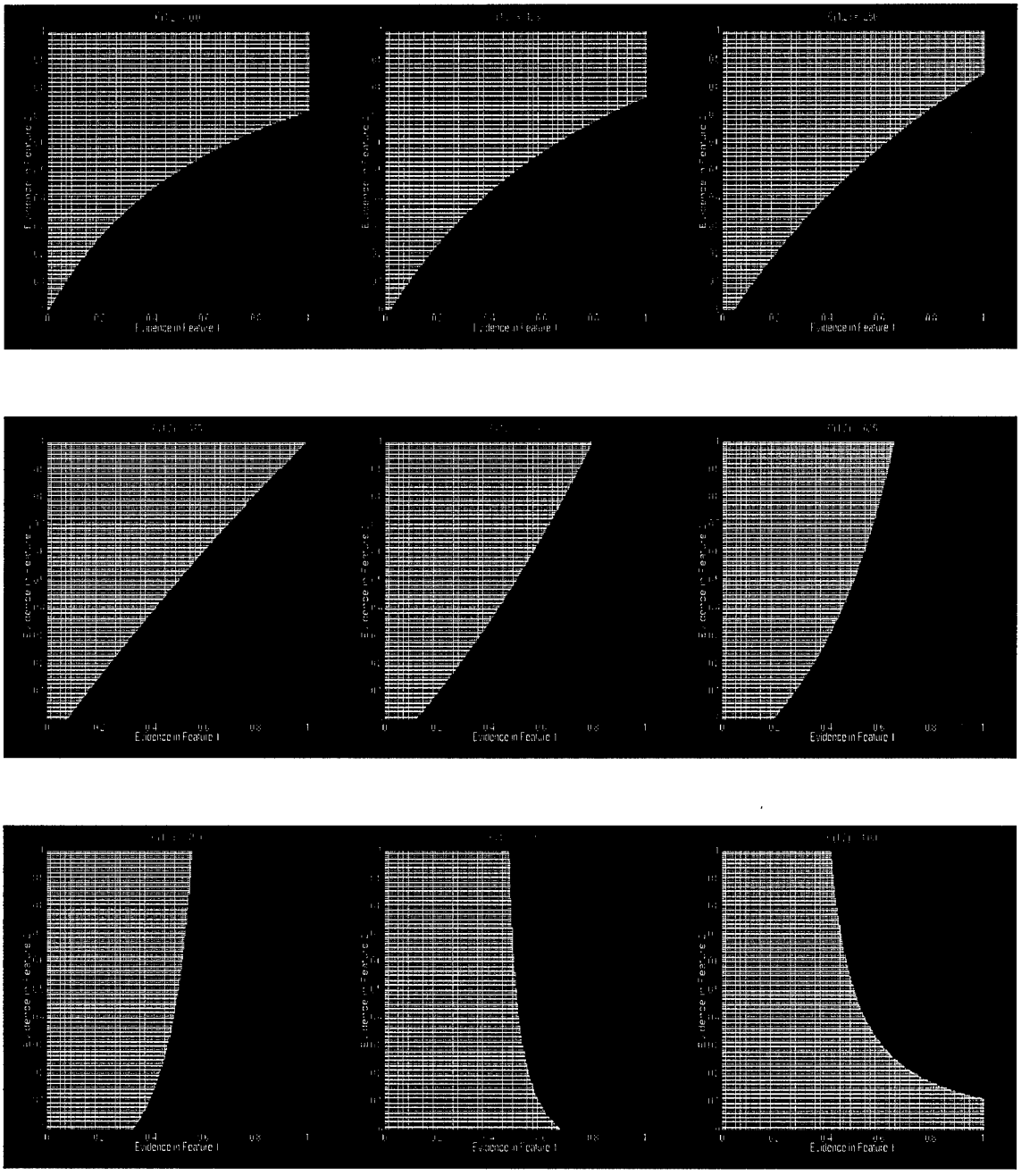


Figure 3.14 Set of Decision Regions Showing Effect of Changes to C_{12}

to one. Figure 3.15 shows nine frames of a belief graph movie. The associated confusion matrix is C and the varying element is again C_{12}. The step size is .125.

$$C = \begin{bmatrix} .9 & C_{12} \\ .4 & .83 \end{bmatrix}$$

As C_{12} approaches the value of C_{22}, the slope of the belief value graphs become more gradual indicating the network has a more difficult time discerning between the hypotheses. As C_{12} passes and moves beyond the value of C_{22}, the slope of the graphs begin to increase again. Note that as C_{12} increases, hypothesis R_1 becomes more and more dependent on the input to node B, corresponding to feature 2, until it becomes the dominate factor in R_1's belief value.

The third movie program is movieab2D.m. This script file is designed to work with Martin Marietta confusion matrices which have ones on the diagonal. The program steps the off-diagonal matrix element C_{ij} from the value one until $C_{ij} = C_{ji}$. It then varies C_{ji}. This produces N frames showing the resulting decision regions as the decision region boundary varies from the extremes or x = .5 or y = .5, to y = x, and then to the other extreme. Figure 3.16 shows nine frames created by movieab2D.m. The associated confusion matrix is I and the varying element is I_{12}. The step size is $(1 - I_{21})/((N-1)/2) = .15$.

$$I = \begin{bmatrix} 1.000 & I_{12} \\ 0.400 & 1.000 \end{bmatrix}$$

With $I_{12} = I_{22} = 1$, the decision region boundary becomes x = .5. As I_{12} decreases, the boundary deflects from x = .5 until it reaches y = x at $I_{12} = I_{21}$. At this point, movieab2d.m begins to vary I_{21}. As I_{21} increases, the boundary line deflects from y = x to eventually reach y = .5 at $I_{21} = I_{11} = 1$.

To summarize, there are five Matlab script files associated with a rank 2 Bayesian network. Make2D.m produces a graph of the decision regions. Netbelief2D.m produces to superimposed graphs of the belief values of the parent node hypotheses. Movie2D.m produces

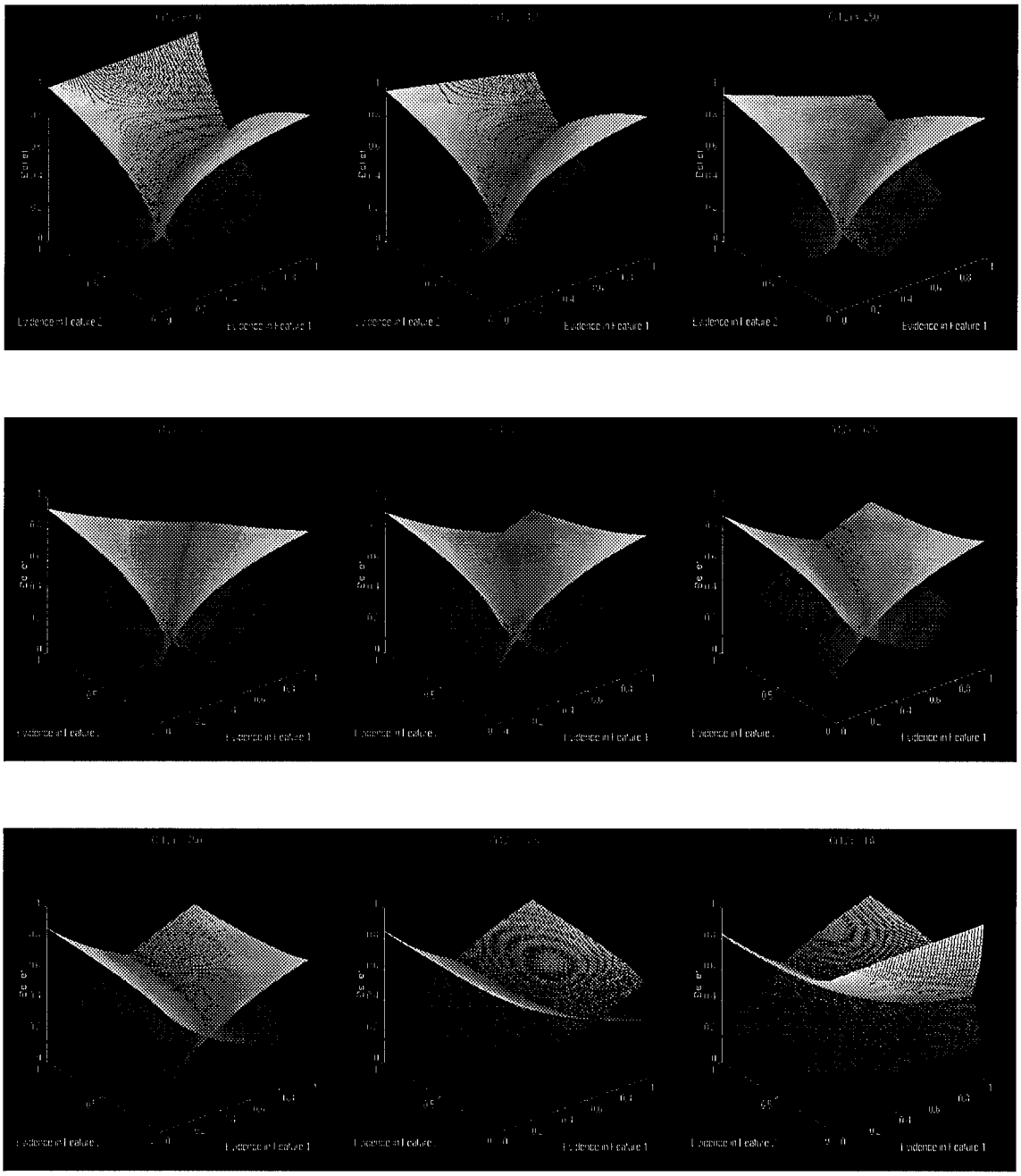


Figure 3.15 Set of Belief Value Graphs Showing the Effect of Changes to C_{12}

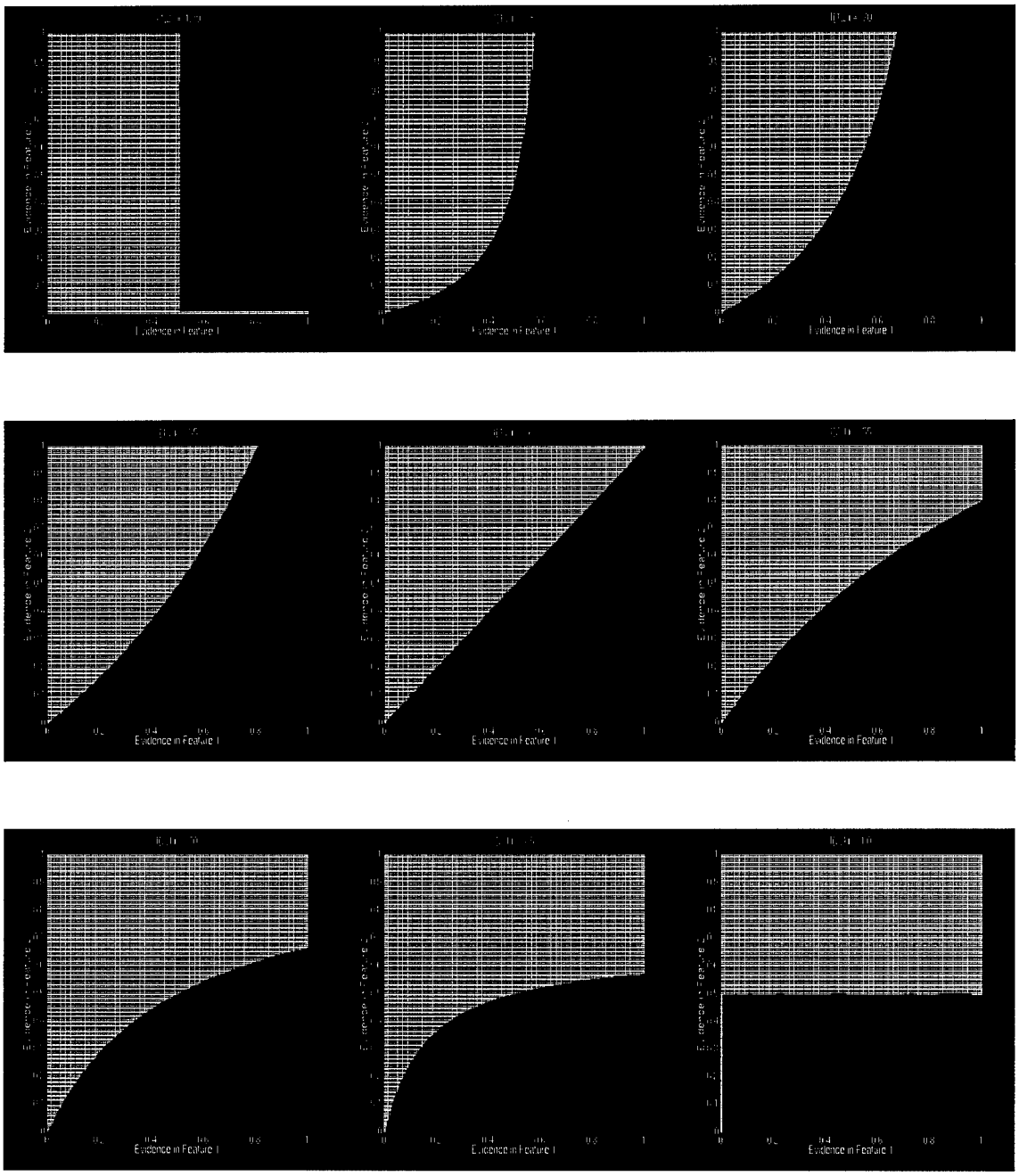


Figure 3.16 Set of Decision Regions Showing the Effect of Changes to I_{13} and I_{31} as I_{13} Varies From One to I_{31} Followed by I_{31} Ascending to One

a matrix of N frames where each frame shows the effect on the decision regions of varying a confusion matrix element. Belmovie2D.m produces a matrix of N frames showing how the belief values change versus changes to a confusion matrix element. Movieab2D.m expects a confusion matrix with ones on the diagonal. It produces a matrix of N frames showing how the decision regions change versus changes in the off diagonal confusion matrix values. All these script files take a confusion matrix and an a priori vector as input.

Appendix B lists these script files as well as two others, regions2D.m and spregions2D.m. These two files are subroutines called by make2D.m. Regions2D.m or spregions2D.m actually compute the decision regions and pass them back to make2D.m for display. Spregions2D.m is optimized for a confusion matrix with ones on the diagonal. It takes advantage of the ones to reduce the complexity of the calculations.

3.6.4 Decision Regions for a Rank 3 Bayesian Network. Analogous to the script files for a rank 2 Bayesian network are script files for a rank 3 Bayesian network. The form and function of these files are exactly the same as the rank 2 case. However, the rank 3 files extend to account for the the third hypothesis and input node. This extension manifests itself most prominently in requiring a third dimension to display the resulting decision regions. This section presents example output from some of the script files for the rank 3 case.

The required confusion matrix for a rank 3 Bayesian network is now 3×3. Figure 3.17 shows the graphical output of Make3D.m given confusion matrix D.

$$D = \begin{bmatrix} 0.945 & 0.107 & 0.414 \\ 0.815 & 0.440 & 0.577 \\ 0.289 & 0.734 & 0.062 \end{bmatrix}$$

Graphing the decision regions for three inputs results in a cube structure. Figure 3.17 shows two views of the cube so all six sides are visible. The decision region boundaries of the cube are planes instead of lines. Three planes are present, each separating two decision

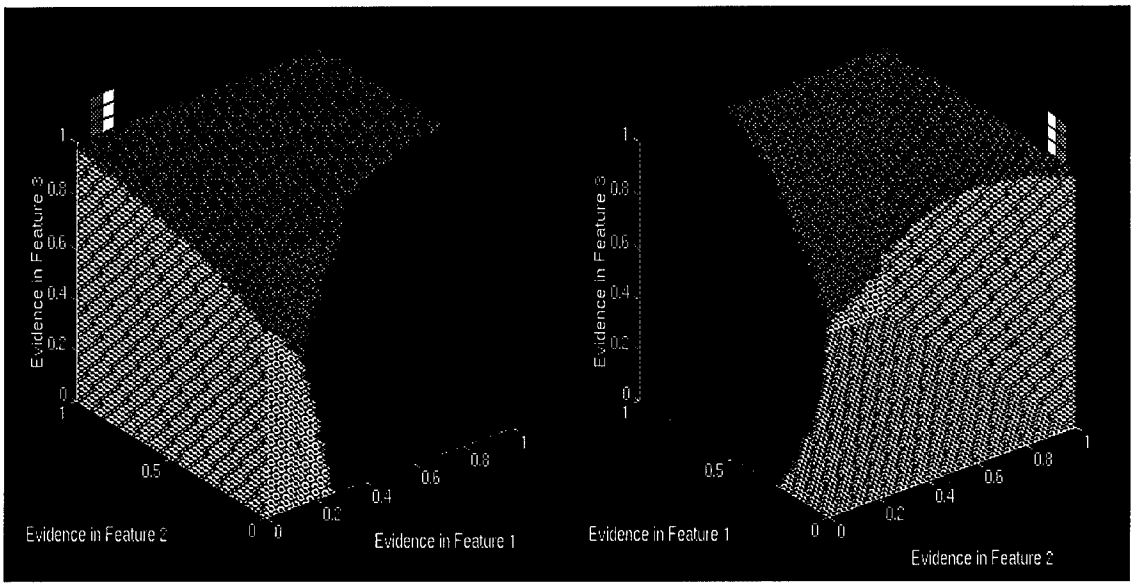


Figure 3.17 The Decision Regions Resulting from the Conditional Probability Values in Matrix D

regions. The equations specifying these planes are complicated functions of two variables. Equation 3.22 expresses the general equation in terms of a confusion matrix M.

$$z = \frac{(xyc2 - xyc1)xy + (xc2 - xc1)x + (yc2 - yc1) + o2 - o1}{(xyzc1 - xyzc2)xy + (xzc1 - xzc2)x + (yzc1 - yzc2)\,y + zc1 - zc2} \quad (3.22)$$

where

$$
\begin{aligned}
xyzc1 &= (2M_{r1} - 1) \times (2M_{r2} - 1) \times (2M_{r3} - 1) \\
xyzc2 &= (2M_{s1} - 1) \times (2M_{s2} - 1) \times (2M_{s3} - 1) \\
xyc1 &= (2M_{r1} - 1) \times (2M_{r2} - 1) \times (1 - M_{r3}) \\
xyc2 &= (2M_{s1} - 1) \times (2M_{s2} - 1) \times (1 - M_{s3}) \\
yzc1 &= (2M_{r2} - 1) \times (2M_{r3} - 1) \times (1 - M_{r1}) \\
yzc2 &= (2M_{s2} - 1) \times (2M_{s3} - 1) \times (1 - M_{s1}) \\
xzc1 &= (2M_{r1} - 1) \times (2M_{r3} - 1) \times (1 - M_{r2}) \\
xzc2 &= (2M_{s1} - 1) \times (2M_{s3} - 1) \times (1 - M_{s2}) \\
xc1 &= (2M_{r1} - 1) \times (1 - M_{r2}) \times (1 - M_{r3}) \\
xc2 &= (2M_{s1} - 1) \times (1 - M_{s2}) \times (1 - M_{s3}) \\
yc1 &= (2M_{r2} - 1) \times (1 - M_{r1}) \times (1 - M_{r3}) \\
yc2 &= (2M_{s2} - 1) \times (1 - M_{s1}) \times (1 - M_{s3}) \\
zc1 &= (2M_{r3} - 1) \times (1 - M_{r1}) \times (1 - M_{r2}) \\
zc2 &= (2M_{s3} - 1) \times (1 - M_{s1}) \times (1 - M_{s2}) \\
o1 &= (1 - M_{r1}) \times (1 - M_{r2}) \times (1 - M_{r3}) \\
o2 &= (1 - M_{s1}) \times (1 - M_{s2}) \times (1 - M_{s3})
\end{aligned}
$$

and

$$\forall r,s \in \{1,2,3\}, r > s$$

The complexity of Equation 3.22 does not lend itself easily to generalizations. In fact, only one general observation is offered. If the off-diagonal matrix elements are all equal, and the on-diagonal matrix values are equivalent, then the decision regions will be the same as those shown in Figure 3.18. Matrix M characterizes these regions while Matlab script file make3D.m produced them.

$$M = \begin{bmatrix} X & Y & Y \\ Y & X & Y \\ Y & Y & X \end{bmatrix}$$

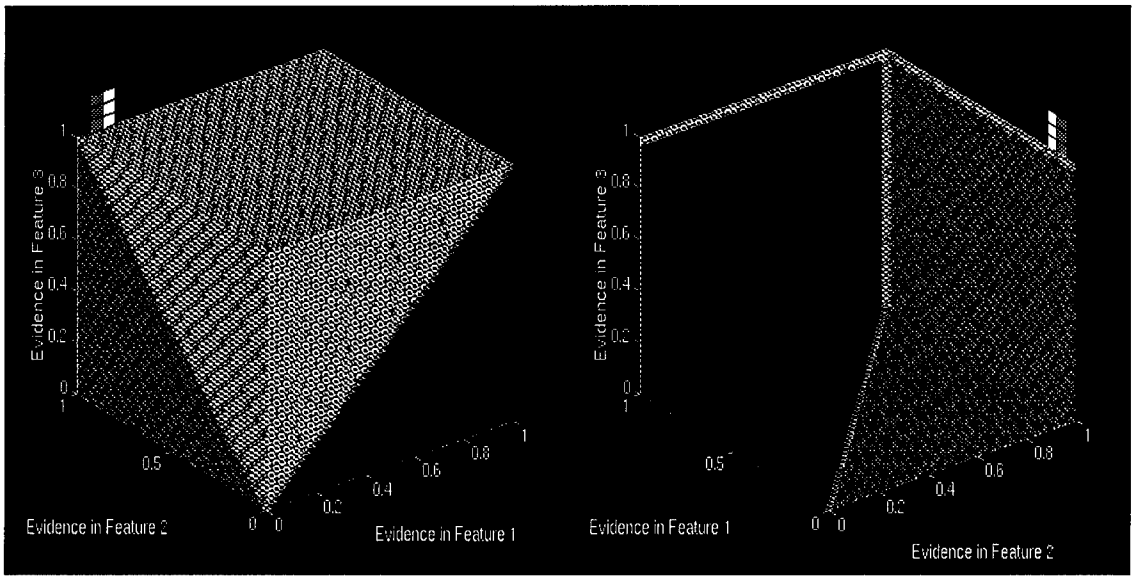


Figure 3.18 The Decision Regions Resulting from the Conditional Probability Values in Matrix M

The variables X and Y are any values between zero and one with the caveat that X is greater than Y. If X is less than Y, the entire decision region cube flips 180 degrees around the all three axes. In this case, the Bayesian network is classifying given a lack of evidence as opposed to its presence. If X and Y are equal, then the decision regions are equal throughout the cube, indicating the underlying Bayesian network cannot discriminate the hypotheses.

Only in these simple cases can generalizations about a rank 3 Bayesian network be made. Nonetheless, make3D.m is still a powerful tool as it will take any 3×3 confusion matrix and produce the resulting decision regions for rank 3 Bayesian network.

Make3D.m can also produce the decision regions for networks with three hypotheses in the root node and any combination of H nodes equal to or less than three. For example, make3D.m can produce the decision regions for a Bayesian network using two sources of

evidence to distinguish between three hypotheses. The earlier analysis of the evidence nodes in Section 3.5 shows why. The analysis showed that a conditional probability table filled with values of .5 essentially cuts off the node. Thus, by entering .5 values throughout the nth column of the confusion matrix, the nth evidence node can be eliminated from consideration by the network. Semantically, the network essentially ignores any evidence coming from the node.

Figure 3.19 shows the decision regions for the example Bayesian network of Chapter A. The inputs are taken from the nodes Propulsion and Turret. Matrix N expresses the confusion matrix for the top two levels of the network shown in Figure 3.3. The third column of N contains only values of .5. Both the front and backsides of the cube are shown.

$$N = \begin{bmatrix} 0.950 & 0.999 & 0.500 \\ 0.050 & 0.020 & 0.500 \\ 0.010 & 0.005 & 0.500 \end{bmatrix}$$

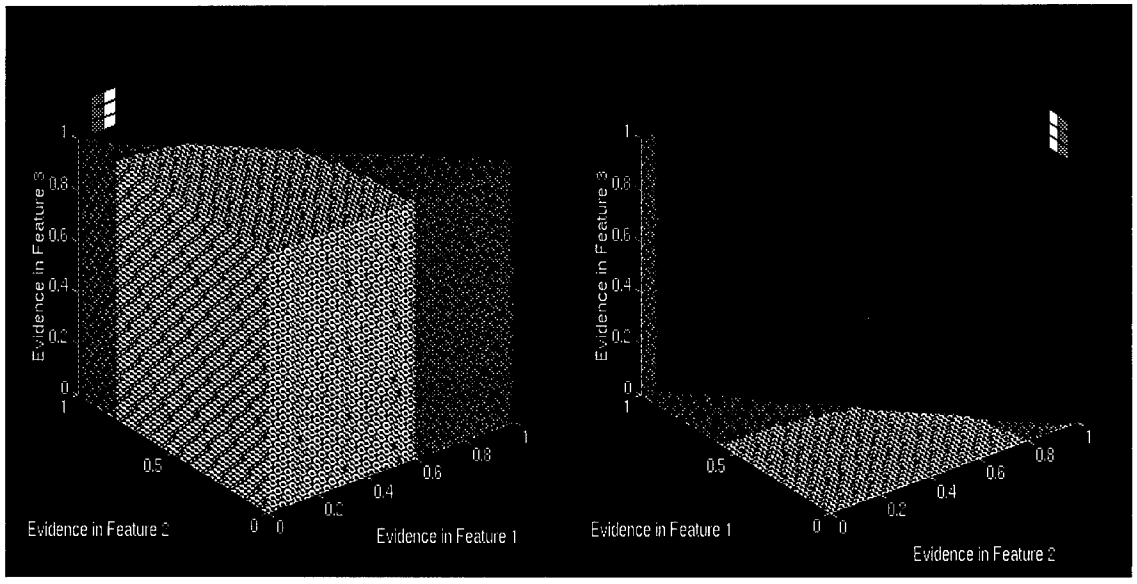


Figure 3.19 The Decision Regions Resulting from the Conditional Probability Values in Matrix N

Note the lines moving up the sides of the decision region cube. The fact the lines are straight show the decision regions are not varying with respect to evidence from the third evidence node. The network has cut it off.

In Figure 3.19 the lightest color represents the hypothesis Jeep , the darkest represents hypothesis Tank, and the middle color is hypothesis Truck. Chapter A made the claim that it was unlikely, given the current conditional probability tables, that the network would ever choose the hypothesis Truck. Figure 3.19 shows this assertion is not entirely true. The Truck hypothesis does dominate for certain λ vector element values at the Propulsion and Turret nodes. However, the Jeep hypothesis is far more prevalent and the Tank hypothesis clearly dominates. This example shows how these script files can be used to display and analyze the decision regions of a Bayesian network.

A drawback to the decision region cubes created by make3D.m is that only the surface of the cubes are visible. It is impossible to tell how the decision regions vary in the interior of the cubes. A separate Matlab script file, slice3D.m, alleviates this problem to some extent. Slice3D.m returns a graph of a slice of the decision region cube at some static value. The static value may be fixed along the x, y, or z axis returning a slice varying in the planes perpendicular to the x, y, or z plane respectively. As an example, if slice3D.m receives a character 'z' and a value of .5, the program returns a graph of the decision regions at the static z axis value of .5 as the x and y axes vary from zero to one. Figure 3.20 shows three slices of the decision region cube of Figure 3.17 at the static values of x = .5, y = .5, and z = .5 respectively. The figure also shows the three slices graphed on the same set of axes.

Slice3D.m provides the ability to look inside the decision region cube and see how the decision regions change internally. Now, given any point in the cube's space, a researcher can visualize the changes to the decision regions in the immediate neighborhood of that point.

As in the rank 2 case, a tool exists to display the belief values for a rank 3 Bayesian network. Unlike the rank 2 case there is not enough dimensions to graph the belief values for all input values possible at three evidence nodes. Thus, the netbelief3D.m script file accepts, along with the confusion matrix and a prior vector, a static value as the input to the third

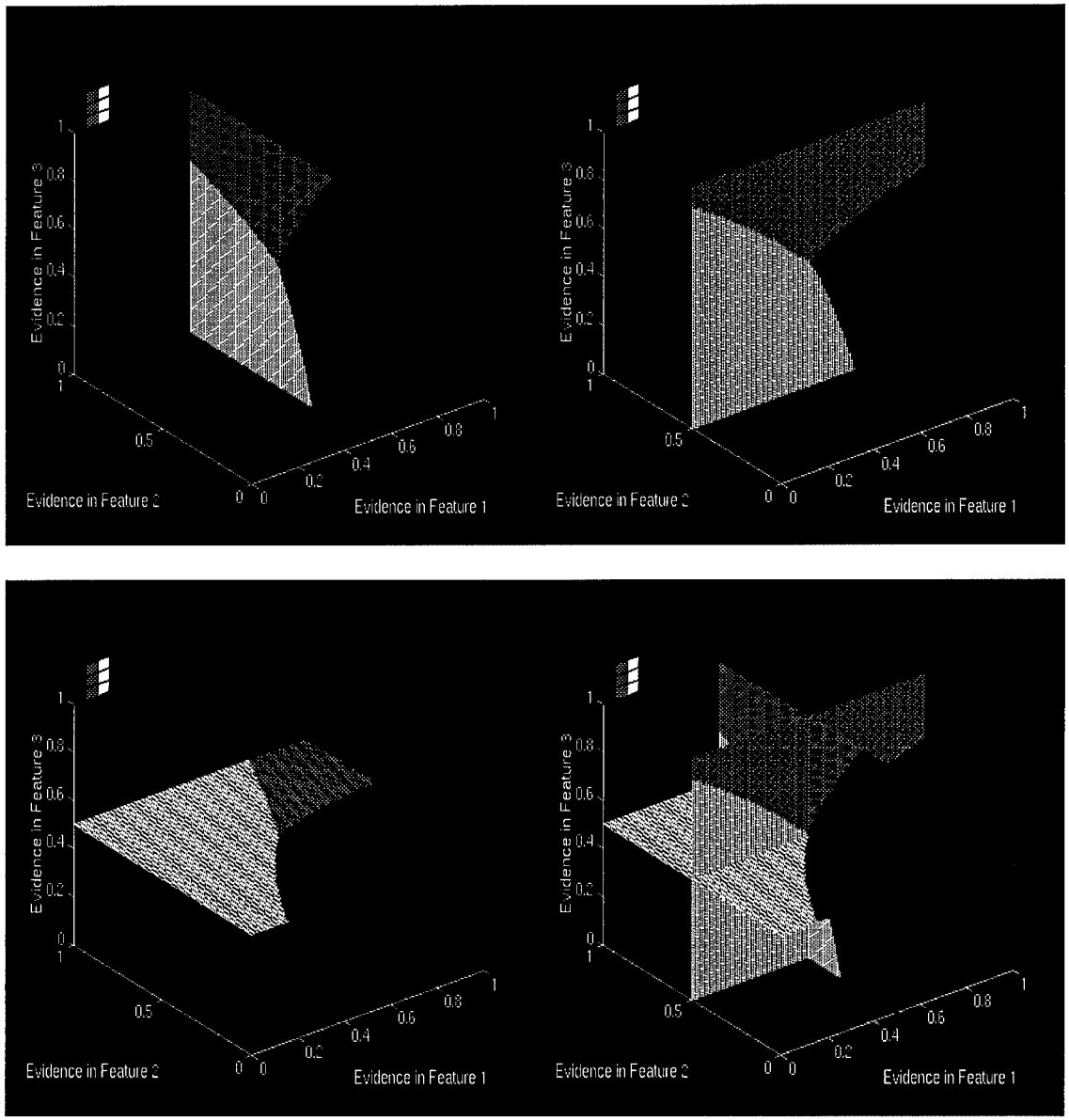


Figure 3.20 Three slices of the decision region cube characterized by matrix D at the static values of x = .5, y = .5, and z = .5 respectively.

evidence node. Figure 3.21 shows the graphs of the belief values for the rank 3 Bayesian network characterized by matrix D. The static value is .5 and is the input for the third node.

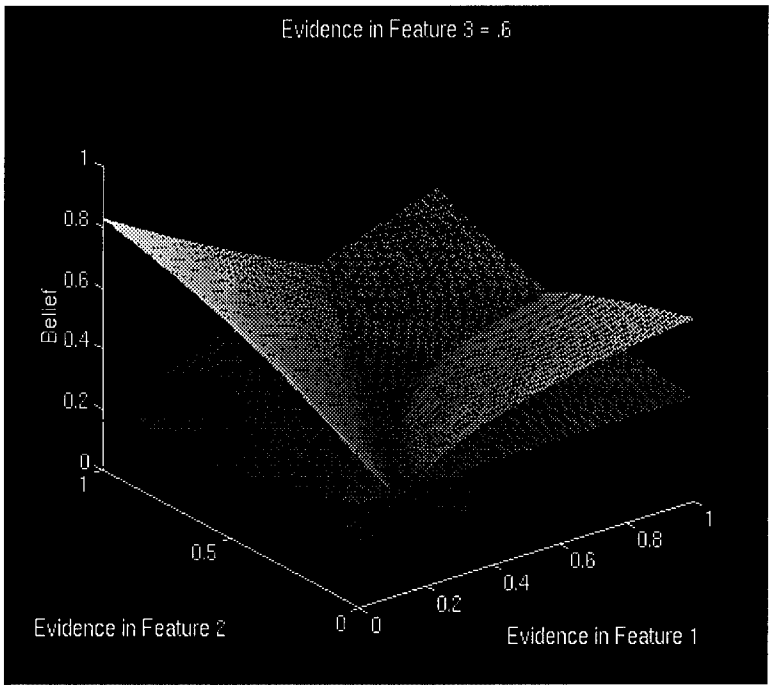


Figure 3.21 Belief Values for the Rank 3 Bayesian Network Characterized by Matrix D

The belief graphs are again superimposed. However, the resulting composite clearly shows how the belief values vary with respect to each other. As before, the columns of the confusion matrix again control the slope of the individual belief graphs.

Analogous to the programs for a rank 2 Bayesian network, the Matlab movie feature lends itself to a method for visualizing the effects of varying the confusion matrix elements in a rank 3 Bayesian network. The approach employs time as a fourth dimension to show how the decision regions vary over changes in the confusion matrix and how the belief values change over changes in the input to the third evidence node. Three Matlab script files are available for creating movies associated with a rank 3 Bayesian network, movie3D.m, belmovie3D.m and movieab3D.m. Much of the following discussion is analogous to the discussion in Section 3.6.3.

As a quick example Figure 3.22 shows nine frames created by the rank 3 movie tool movieab3D.m. Like its sibling file movieab2D.m, this script file is designed to work with confusion matrices with ones on the diagonal. The program steps the off-diagonal matrix element M_{ij} from the value one until $M_{ij} = M_{ji}$. It then varies M_{ji}. Unlike movieab2D.m , varying an element in this manner does not range the decision regions among extremes unless all the remaining off-diagonal elements are equal in value. The associated confusion matrix is T and the varying elements are T_{13} and T_{31}. The step size is $(1 - T_{31})/((N-1)/2) = .15$.

$$T = \begin{bmatrix} 1.00 & 0.75 & T_{ij} \\ 0.75 & 1.00 & .75 \\ 0.4 & 0.75 & 1.00 \end{bmatrix}$$

Because the off-diagonal matrix elements of matrix T are equal, the facing surfaces of the resulting cube take on the extreme configurations of a rank 2 decision regions as the varying elements approach the extremes. In fact, the surface along the x axis behaves precisely like Figure 3.16. Since the y axis value is zero along this face, this shows the T_{13} and T_{31} link together the first and third evidence nodes as if they were the evidence nodes of a rank 2 Bayesian network. All the surfaces of the cube appear similarly linked meaning that, in the specific case of a matrix with equal off-diagonal elements, the surface in the plane of evidence node i and evidence node j, varies with changes to element values T_{ij} and T_{ji} as if it were the decision region graph of a rank 2 Bayesian network with evidence node i and j and off-diagonal matrix elements T_{ij} and T_{ji}. However, this is only true in the limited and specific case where the off-diagonal matrix elements are equal and the on-diagonal matrix elements are one. It is not true in the general case.

To summarize, there are six Matlab script files associated with a rank 3 Bayesian network. Make3D.m produces a graph of the decision regions. The graph is in the form of a cube and the decision regions are separated by planes. Slice3D.m returns a graph of a slice of the decision region cube at some static value. The static value may be fixed along the x, y, or z axes returning a slice varying in the planes perpendicular to the x, y, or z plane

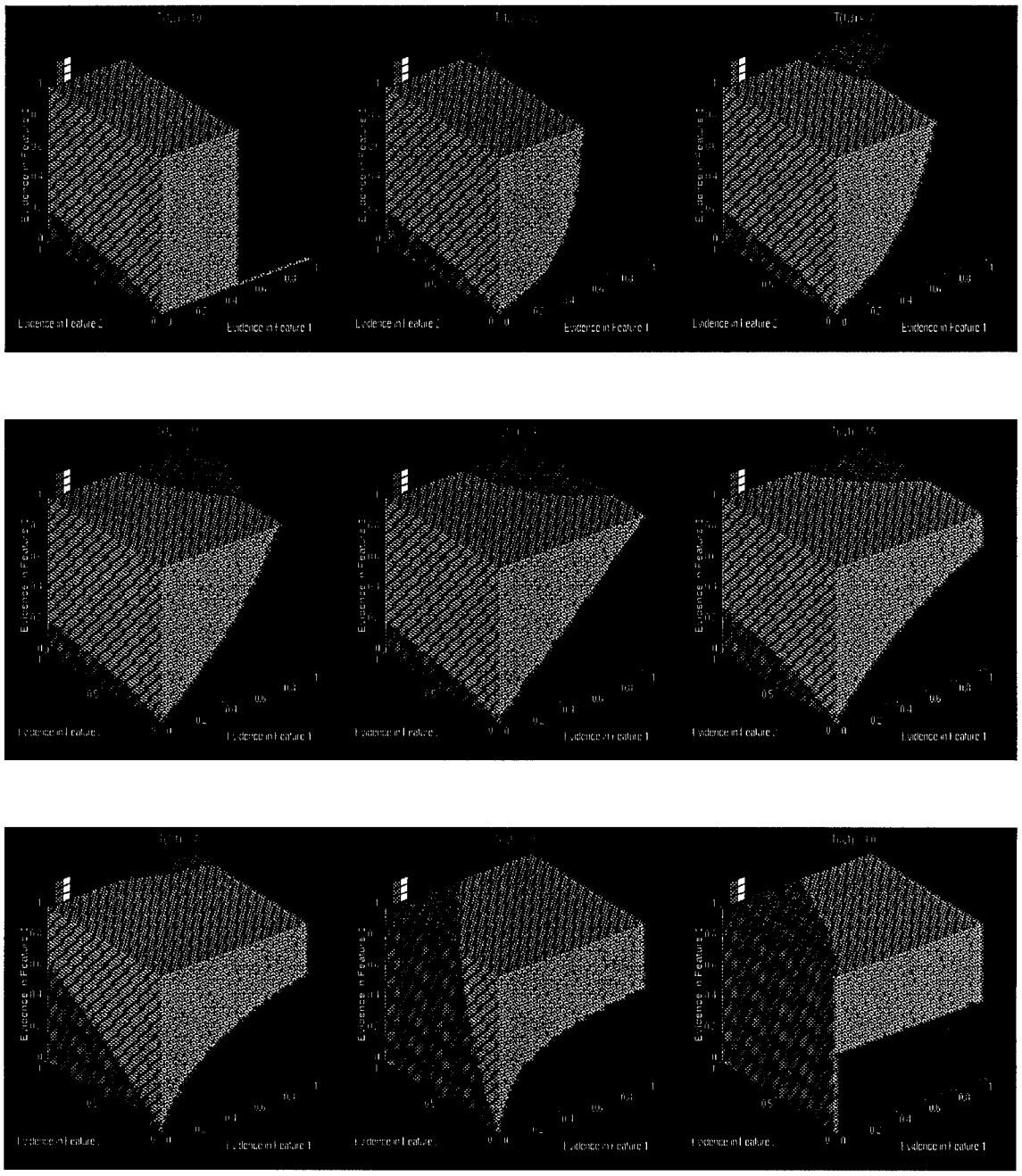


Figure 3.22 Set of Decision Regions Showing the Effect of Changes to T_{13} and T_{31} as T_{13} Varies From One to T_{31} Followed by T_{31} Ascending to One

respectively. Netbelief3D.m produces superimposed graphs of the belief values of the parent node hypotheses with one input held static. Movie3D.m produces a matrix of N frames where each frame shows the effect on the decision regions of varying a confusion matrix element. Belmovie3D.m produces a matrix of N frames showing how the belief values change over all input values. Movieab3D.m expects a confusion matrix with ones on the diagonal. It produces a matrix of N frames showing how the decision regions change verses changes in the off-diagonal confusion matrix values. All these script files take a confusion matrix and an a priori vector as input.

Appendix B lists these script files as well as two others, regions3D.m and spregions3D.m. These two files are subroutines called by make3D.m. Regions3D.m or spregions3D.m actually computes the decision regions and pass them back to make3D.m for display. Spregions3D.m is optimized to confusion matrix with ones on the diagonal. It takes advantage the ones to reduce the complexity of the calculations.

3.6.5 Decision Regions for a Rank 4 Bayesian Network. Tools that displays the decision regions for a rank 4 Bayesian network are also available. These tools, are directly analogous to the tools for the rank 2 and rank 3 cases, except they extend to handle the fourth hypothesis and the fourth input node. This extension manifests itself most prominently in the fact that there are not enough dimensions to account for all four inputs. Thus, the tools require a static input on one node. Another key difference is the rank 4 tools only handle matrices with ones on the diagonal. This restriction greatly simplifies the computational complexity of calculating the decision regions. Also, no tools exist to display the belief values for the rank 4 case and only one movie tool is available. The movie tool's main function is to display the decision regions as they change over changes to the fourth input. In this manner, the movie tool uses time as a fourth dimension. The last tool available for the rank 4 case is a slice4D.m tool analogous to the slice3D.m tool. This tool allows the user to look inside the cube at selected inputs.

As a quick example, Figure 3.23 shows the graphical output of Make4D.m given confusion matrix E. The static value is .5 and is associated with the fourth evidence node.

$$E = \begin{bmatrix} 1.000 & 0.844 & 0.022 & 0.436 \\ 0.267 & 1.000 & 0.161 & 0.582 \\ 0.970 & 0.456 & 1.000 & 0.752 \\ 0.247 & 0.817 & 0.708 & 1.000 \end{bmatrix}$$

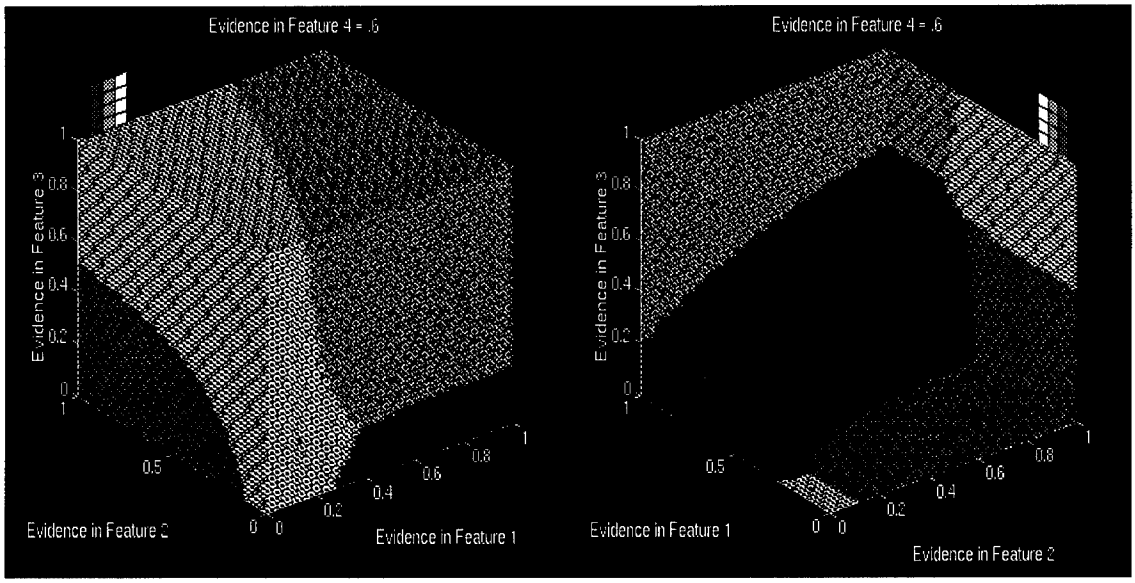


Figure 3.23 The Decision Regions Resulting from the Conditional Probability Values in Matrix E

Graphing the decision regions for four inputs again results in a cube structure but four different decision regions. Both the front and the back side of the cube are shown. The decision regions are again separated by planes. No attempt is made to derive the relations governing the decision region boundaries in this case. Effort was focused instead in developing a tool capable of displaying the decision regions in the rank 4 case.

To summarize, only three tools associated with a rank 4 Bayesian network are available. Make4D.m produces a graph of the decision regions where the input to one of the evidence nodes is held constant. The resulting cube shows four decision regions corresponding to four

hypotheses. Make4D.m can also produce the decision regions for a two level Bayesian network discriminating among four hypotheses with evidence supplied by two or three evidence node. This is accomplished by setting the column values of excess nodes to .5. These values in the nth column essentially cuts the nth evidence node off from consideration. Slice4D.m returns a graph of a slice of the decision region cube at combinations of two axes held static, and two allowed to vary. Movie4D.m produces a matrix of N frames where each frame shows the effect the decision regions as they vary over changes to all the inputs at the evidence nodes. No method is offered on how to visualize the decision regions over changes to elements of the confusion matrix. Note also that no tools exist graphing the belief values of the rank 4 Bayesian network.

Appendix B lists the script files make4D.m, slice4D.m, and movie4D.m as well as an additional file, regions3D.m. This file computes the decision regions of the rank 4 Bayesian network and passes them back to make4D.m for display. Regions4D.m is optimized to a confusion matrix with ones on the diagonal. It takes advantage the ones to reduce the complexity of the calculations.

3.7 Concluding Remarks

Overall, there are two significant results of this chapter. The first is the analysis of the evidence nodes and the equations associating the conditional probabilities to the amount of diagnostic support forwarded to the parent node. Conditional probability values greater than .5 result in diagnostic support values that increase linearly with increases in evidence for a feature. How much the diagnostic support increases is a function of the magnitude of the conditional probability linking the input to an hypothesis. Values close to one mean diagnostic support increases quickly as evidence increases. Values closer to .5 result in more gradual increases. Conditional probability values less than .5 reverse the slope of the linear relationship and decrease the amount of diagnostic support as the evidence in a feature increases. Conditional probability values close to zero mean diagnostic support decreases quickly in response to increased evidence. Values closer to .5 result in gradual decreases. Associating an hypothesis

with a feature with a conditional probability value less than .5 is semantically equivalent to adding a node that keys on the absence of a feature. Conditional probabilities equal to .5 cut the node off from consideration.

The second significant result is the collection of Matlab script files as a whole. With these tools, a researcher can visualize the decision regions for any two level Bayesian network, or the top two levels of a multiple level network, subject to certain restrictions. These restrictions are the root node have only two, three, or four hypotheses and receive evidence from a number of evidence nodes not to exceed the number of hypotheses. Further, the evidence nodes must have only two hypotheses and the inputs to the evidence nodes must also be binary and sum to one. These tools allow the researcher to visualize how the decision regions vary over changes to a modest number of network parameters. The use of the Matlab movie feature is paramount in this ability. For rank 2 and rank 3 Bayesian networks, the researcher can also view the belief values of the hypotheses.

IV. Bayesian Network Classifier Design

4.1 Introduction

This chapter addresses two key questions concerning Bayesian networks and their utility as classifiers. First, how can Bayesian networks solve traditional classification problems and, second, how are the required conditional probabilities established? To date, no set method is available to map Bayesian networks to classification problems. In fact, most Bayesian networks are pieced together using ad hoc approaches. The following sections detail an algorithm for mapping a two level Bayesian network to simple classification problems. The algorithm defines the topology for the Bayesian network and specifies how to establish the conditional probabilities. Several examples are presented and discussed. For clarity, this chapter uses the term Bayesian network classifier to refer to a Bayesian network designed to classify objects.

4.2 An Algorithm for Defining a Bayesian Network Classifier

Before presenting the algorithm, this section first defines what is meant by a traditional classification problem. These are classification problems found in standard texts on statistical pattern recognition or classification. The basic premise involves two classes sharing a common set of features. The features are measured using sensors with the ideal sensor measurements reporting markedly different values for each class. Each feature measurement for a given class clusters about some mean and exhibits a distribution about that mean. The distribution is usually assumed to be gaussian. The simplest classification problems discriminate between two classes using one or two features. The means and the standard deviations of the feature measurements result in distributions that do not appreciably overlap. More challenging problems involve more classes and features as well as distributions that overlap to some degree. Real world problems often encounter distributions that overlap to a large degree. The more overlap in the distributions the more challenging the classification becomes. This is because the feature measurements for one class can be equal to the feature measurement for

another class making them impossible to distinguish. Figure 4.1 illustrates a representative, simple, classification problem using one feature to discriminate between two classes.

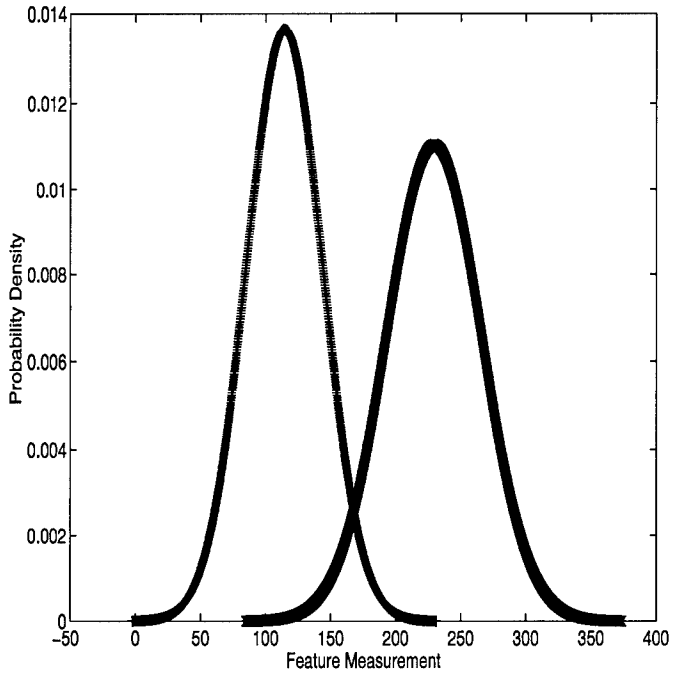


Figure 4.1 Feature Measurement Probability Distributions for a Two Class Problem

The x axis of Figure 4.1 represents the feature measurement. The y axis represents the probability density of a given measurement occurring. The feature measurements associated with a given class exhibit gaussian distributions centered on a mean. Note that the tails of the distributions overlap creating a region where a feature measurement may be from either class.

The classification problem presented by Figure 4.1 establishes context for the algorithm defining a Bayesian network classifier. The algorithm is best presented in a series of steps. Each will be elaborated on as presented. The algorithm is as follows.

Step one: Establish the basic topology of the network.

For any classification problem, only a two level Bayesian network is required. Figure 4.2 illustrates the topology.

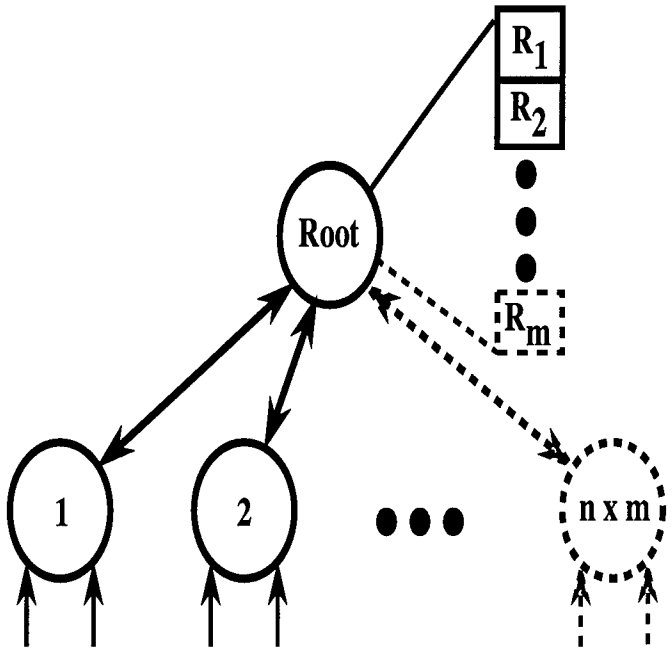


Figure 4.2 Two Level Bayesian Network Required to Solve Any Classification Problem

The topology is capped by a root node possessing a hypothesis for each class. The root node is connected to $m \times n$ child nodes where m is the number of classes and n is the number of features. The child nodes serve as evidence nodes where the purpose of each is to collect evidence for the existence of a feature measurement within the range required by a given class. Thus, the topology of a Bayesian network classifier has one root node connected to $n \times m$ child nodes.

Step two: Establish the underlying probability distributions of each feature for each class.

Establishing the probability distributions of each feature for each class involves isolating a given feature, establishing the mean and standard deviation of the feature's measurements for each class, and determining the associated probability distributions. To state this more formally, let C represent the set of all classes and F represent the set of all Features. Then, $\forall(f,c) \in \{F \times C\}$, isolate the data associated with feature f and class c and establish the mean, standard deviation, and probability distribution. For clarity, this work uses the ordered

pair (f_i, c_j) to indicate the probability distribution of the i^{th} feature measurement associated with the j^{th} class. The distribution may be known, assumed, or estimated by statistical techniques. Two distributions need not be of the same type. A situation where (f_i, c_j) has a gaussian distribution while (f_i, c_k) has a chi squared distribution does not present significant problems.

Step three: Establish decision boundaries that partition each feature space relative to each class.

For all features, $f_i \in F$, and within the subset $f_i \times C$ a set of k decision boundaries must be established. The purpose of the decision boundaries is to isolate a region of f_i measurements under the probability density curve for (f_i, c_j). Each distribution, (f_i, c_j), requires two decision boundaries which must always maintain their association with the distribution. The mean for class c_j should appear between the associated decision boundaries. The number of decision boundaries k in a subset, therefore, is equal to $2 \times m - 2$ where m is the number of classes. Subtracting out two boundaries is a result of two special cases discussed later. The total number of decision boundaries required for the entire classifier is $n \times k$ where n is the number of features. This large number of decision boundaries is necessary to establish all the conditional probability values required by the Bayesian network. To put it more formally, $\forall f_i \in F$, restrict the domain of the relation $F \times C$ to produce the subset $f_i \times C$. Within this subset establish decision boundaries for each $c_j \in C$ that isolates a range of feature measurements under the distribution curve (f_i, c_j). Two special cases exist. For the special case where $\forall c_k \in C \ni k \neq j$, the mean of (f_i, c_j) is greater than the mean of (f_i, c_k), the algorithm requires no decision boundary greater than the mean of (f_i, c_j). For the special case where $\forall c_k \in C \ni k \neq j$, the mean of (f_i, c_j) is less than the mean of (f_i, c_k), the algorithm requires no decision boundary less than the mean of (f_i, c_j).

The subject of precisely where to place the decision boundaries is an important one, but highly dependent on how the algorithm uses the boundaries. Therefore, the discussion on boundary placement is deferred until after the next step in the algorithm is discussed.

Step four: Using the decision boundaries as a reference point, establish the conditional probabilities required by the Bayesian network.

The decision boundaries are key to establishing the conditional probability values for the network. However, before explaining how, each evidence node in the network must be semantically linked with a distribution (f_i, c_j) and a column in the confusion matrix. Recall that each evidence node is assigned to collect evidence that a feature measurement f_i falls within the range required for class c_j. Thus each node is associated with a distribution (f_i, c_j). For each evidence node, there is a corresponding entry in a confusion matrix that holds all the conditional probability values required by the network. The columns, therefore, are labeled (f_i, c_j) associating them with distribution (f_i, c_j).

Based on the above semantic connection, the row entry in column (f_i, c_j) can be established as a function of the distribution (f_i, c_j) and the related decision boundaries. For column $(f_i, \mathrm{c_j})$, establish the subset $f_i \times C$. For each $c_k \in C \ni j \neq k$ apply the following algorithm to establish the entry for row k of the confusion matrix.

- If $c_k \geq c_j$

  Identify the decision boundary associated with (f_i, c_j) that is greater than the mean of (f_i, c_j). Integrate the probability density function for (f_i, c_k) from $-\infty$ to the decision boundary. Semantically, this value represents the probability that, given a feature measurement within c_j's region, the measurement is actually from c_k. Enter this value as the k^{th} entry of the confusion matrix column associated with (f_i, c_j).

- If $c_k < c_j$

  Identify the decision boundary for (f_i, c_j) that is less than the mean of (f_i, c_j). Integrate the probability density function for (f_i, c_k) from the decision boundary to $+\infty$. Semantically, this value represents the probability that, given a feature measurement within c_j's region, the measurement is actually from c_k . Enter this value as the k^{th} entry of the confusion matrix column associated with (f_i, c_j).

For the j^{th} entry of the (f_i, c_j) column integrate the probability density function associated with (f_i, c_j) from one decision boundary to the other.

Repeat this procedure for all $(f, c) \in (F \times C)$. Two special cases exist. For the special case where $\forall c_k \in C \ni k \neq j$, the mean of (f_i, c_j) is greater than the mean of (f_i, c_k), the algorithm requires no decision boundary greater than the mean of (f_i, c_j). In this case, the j^{th} entry in the (f_i, c_j) column of the confusion matrix is found by integrating the probability density function from the remaining decision boundary to $+\infty$. For the special case where $\forall c_k \in C \ni k \neq j$, the mean of (f_i, c_j) is less than the mean of (f_i, c_k), the algorithm requires no decision boundary less than the mean of (f_i, c_j). In this case, the j^{th} entry in the (f_i, c_j) of the confusion matrix is found by integrating the probability density function from $-\infty$ to the remaining decision boundary.

Knowing how the algorithm utilizes the decision region sets the stage for discussing where the decision boundaries should be placed. Clearly, the algorithm should place the decision boundaries to maximize the separation between the distributions involved. However, when the distributions overlap, the best placement may be difficult to determine. An observation made in Chapter III forms a basis for determining the optimal placement of the decision boundaries to ensure optimal performance by the Bayesian network. The observation states that the belief values of a Bayesian network range to the extremes more quickly when the elements of the confusion matrix differ most dramatically. In other words, Bayesian networks discern more easily between hypotheses when the differences among confusion matrix column elements are maximized. Another observation pointed out that Bayesian networks afford the greatest diagnostic support to hypotheses associated with evidence nodes by large values in the confusion matrix. Since, under the topology defined by this algorithm, each evidence node (f_i, c_j) is directly associated with the hypothesis c_j, the network should therefore maximize the j^{th} entry in the (f_i, c_j) column while minimizing the others. This will maximize the diagnostic support for the related class hypothesis while optimizing the discrimination capabilities of the network. The algorithm to achieve this is as follows.

To optimally place the decision boundaries for distribution (f_i, c_j), restrict the domain of $(F \times C)$ to establish the subset $(f_i \times C)$. For the decision boundary greater than the mean of (f_i, c_j) determine the element (f_i, c_k) in $(f_i \times C)$ such that the mean of (f_i, c_k) is greater

than the mean of (f_i, c_j) and the overlap of the associated probability density functions is maximal. Given the probability density functions for (f_i, c_j) and (f_i, c_k) determine a feature measurement such that the difference between the area under the (f_i, c_j) probability density function from $-\infty$ to the feature measurement and the area under the (f_i, c_k) probability density function from the feature measurement to $+\infty$ is maximal. Equation 4.1 states this more precisely. The value m represents the feature measurement that maximizes y and the point at which the decision boundary should be placed.

$$y = max\left(\int_{-\infty}^{m} g_{(f_i,c_j)}(x) - \int_{-\infty}^{m} g_{(f_i,c_k)}(x)\right) \tag{4.1}$$

An analogous argument produces the decision boundary less than the mean of (f_i, c_j). Determine the element (f_i, c_k) in $(f_i \times C)$ such that the mean of (f_i, c_k) is less than the mean of (f_i, c_j) and the overlap of the associated probability density functions is maximal. Given these probability density functions, determine the feature measurement such that the difference between the area under the (f_i, c_j) probability density function from the feature measurement to $+\infty$ and the area under the (f_i, c_k) probability density function from $+\infty$ to the feature measurement is maximal. Equation 4.2 states this more precisely. The value m represents the feature measurement that maximizes y and the point at which the decision boundary should be placed.

$$y = max\left(\int_{m}^{+\infty} g_{(f_i,c_j)}(x) - \int_{m}^{+\infty} g_{(f_i,c_k)}(x)\right) \tag{4.2}$$

The algorithm is subject to the special cases noted earlier. If either of these special cases exists, the algorithm simply does not establish a decision boundary.

This approach guarantees the j^{th} entry in the (f_i, c_j) column of the confusion matrix is the largest value in the column. Because, the algorithm uses the probability density function that results in the most overlap, the algorithm also guarantees the maximum separation among elements in the column. Thus, in theory, the confusion matrix values resulting from this method should optimize the performance of the Bayesian network classifier. In practice, tuning the

placement of the decision boundaries by small amounts may improve the performance of the classifier slightly.

To summarize the algorithm to this point, the designer of the Bayesian network classifier first determines the topology of the network based on the number of classes and features. The root node contains hypotheses equal to the number of classes and connects to a number of evidence nodes equal to the number of features multiplied by the number of classes. The designer next determines the mean and standard deviation for each (feature, class) pair. The probability density function must then be derived from statistical means or assumed for each feature, class pair. Using the distribution data, the designer establishes two decision boundaries for each (feature, class) pair. These decision boundaries should be set such that they maximize the difference between the area under the associated probability density function and the area under the neighboring probability density functions with the most overlap. The decision boundaries serve as references to determine the conditional probabilities required by the network. The decision boundaries serve as upper or lower limits of integration.

Step five: Instantiate the network

With the information gleaned from steps one through four the designer can instantiate the Bayesian network classifier. However, an a priori vector is also required. The a priori vector represents expectations about the distribution of the objects the network will classify. For the forthcoming examples in this chapter, the elements of the a priori vector will be equivalent and sum to one.

Step six: Normalize the data so all input values fall between zero and one.

Once the network is instantiated, it is ready to receive data to classify. Each evidence node receives a feature measurement. However, the network, more often than not, cannot receive the data in its raw form because it violates the Bayesian network criteria that the input values lie between zero and one. To ensure the input values at a node fall between zero and one, the data must be normalized with respect to the associated probability curve. The goal of the normalization is to convert feature measurements on or near the mean to values at or

close to one. As feature measurements move away from the mean, the normalized value should progress to zero. The algorithm to normalize the data is as follows. For each feature measurement, apply the probability density function associated with the node to determine the probability the measurement falls with the node's associated distribution. Next, divide this probability by the probability at the mean. More formally, for some $f_i \in F$ domain restrict the relation $F \times C$ to get the subset $f_i \times C$. For every $c_j \in C$, determine the value of the feature measurement under the probability density curve for (f_i, c_j). Normalize this value by dividing by the value of the probability density function for (f_i, c_j) evaluated at the mean. This produces a value for every (f_i, c_j) that is close to one if the measurement is close to the mean and that tends to zero as the measurement moves away from the mean. The input to a node (f_i, c_j) now becomes this normalized value. This algorithm assumes a distribution where the mean maps to the maximum value of the probability density function.

Given the normalized input values, the instantiated network classifies the inputs by propagating the information they convey throughout the network and establishing belief values at the root node. Once the network reaches equilibrium , it concludes that the hypothesis with the greatest belief value is the class described by the current inputs.

The algorithm presented in this section defines a Bayesian network classifier keyed to discriminate among a given set of classes using input from a given set of features. Note that the key values for defining the network, probability distributions, decision boundaries, and confusion matrix elements, are all based on the underlying data. This makes the algorithm completely data driven. A key feature of the algorithm is it precisely defines a method for determining the conditional probability value, a capability Chapter II showed to be lacking in most Bayesian network implementations. The next three sections five examples of the Bayesian network classifier algorithm in action.

4.3 An Example Bayesian Network Classifier Receiving One Feature

To illustrate the use of the Bayesian network classifier algorithm, this section applies the algorithm to the classification problem illustrated in Figure 4.4. The problem consists of

discriminating between two classes, class A and class B, based on one feature, F. The topology of the classifier consists of one root node with two hypotheses and two child nodes. Figure 4.3 displays the Bayesian network classifier for this problem.

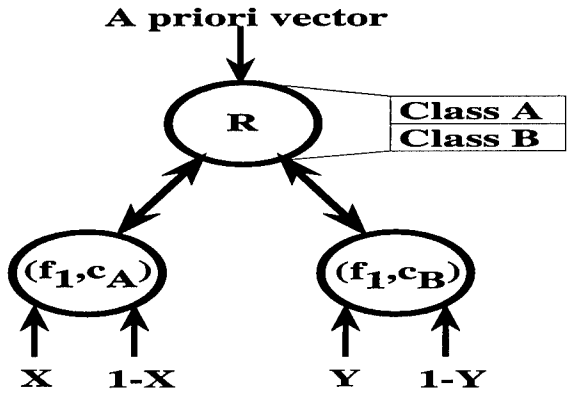


Figure 4.3 A Bayesian Network Classifier For a Two Class, One Feature Problem

Four data sets are required to adequately test the algorithm. The first two data sets are 100 points each, one set for each class. The algorithm will use this data to train, or establish the parameters for, the Bayesian network classifier. The other two data sets contain test data of 1000 points each. The data is generated by the Matlab command RANDN which generates strings of gaussian random numbers that serve as the basis for the data sets. Each string is then scaled by a predetermined standard deviation and offset by a predetermined mean. The standard deviation and mean values producing the training and test data sets for class A are 30.0 and 110.0 respectively. The standard deviation and mean values producing the training and test data sets for class B are 37.0 and 230.0. Because the data sets are produced by a random number generator, they accurately approximate real world sensor measurements with an underlying gaussian probability distribution.

The Bayesian network classifier algorithm uses the training data to define a Bayesian network classifier capable of discriminating the two classes. The first part of the algorithm produces the Bayesian network classifier topology shown in Figure 4.3. The second step establishes the underlying probability density functions of training data sets. Because a gaussian random number generator produces the data sets, this example simply assumes a

gaussian distribution. The calculated means and standard deviations of the data sets appear in Table 4.1.

Table 4.1 Calculated Standard Deviation and Mean Values For Data Sets One and Two

Parameter	(f_1, c_A)	(f_1, c_B)
Std Dev	29.2237	36.1166
Mean	114.3596	229.2758

The standard deviation and mean values result, assuming a gaussian distribution, in the probability density functions shown in Figure 4.4. The dashed lines indicate the distribution (f_1, c_A). The dotted lines denote the distribution (f_1, c_B).

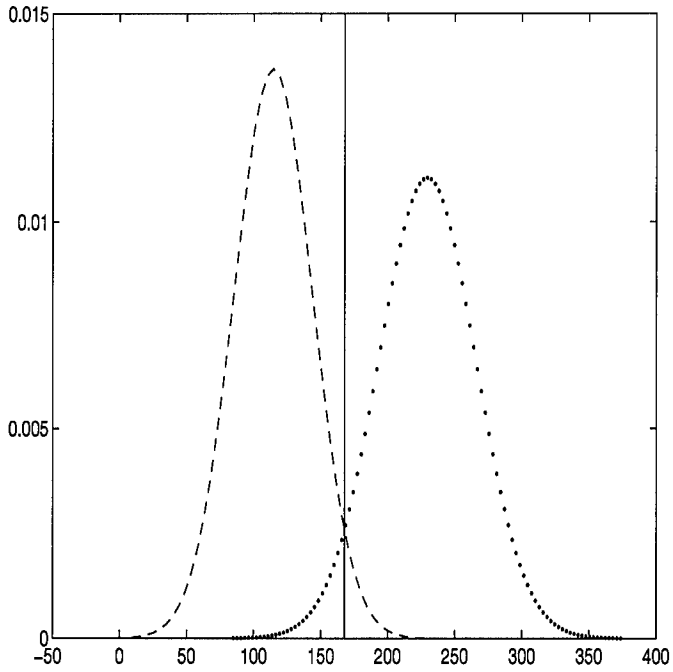


Figure 4.4 Probability Density Functions and Decision Boundaries for the Data Sets in a Two Class Problem

The algorithm uses the distribution data to establish the decision boundaries that optimize the separation between the probability density curves. Four decision boundaries are required, two for each distribution. However, since there are no density functions with means greater than the mean of (f_1, c_B), the algorithm need not specify a decision region greater than this

mean. By converse argument, the algorithm need not specify a decision region less than the mean of (f_1, c_A). Thus, only two decision boundaries need be specified by the algorithm. The optimal location of the boundaries are both found to be at 167.7. Figure 4.4 also displays the decision boundaries

This placement of both decision boundaries at the same location illustrates a general rule of thumb. For most cases, the decision boundaries between two distributions that maximally overlap one another fall on the same coordinate with that coordinate being the point where the two distributions cross. This suggests the algorithm could be simplified by specifying only one decision boundary and not two. However, this work does not show that only one decision boundary is required in all cases. Further, specifying the boundaries according to the algorithm only guarantees the columns of the confusion matrix are optimal to the operation of the Bayesian network. Tuning the boundaries about the optimal point may improve the performance of the classifier slightly. Thus, advantages do exist for specifying both decision boundaries. However, this observation leads to a shortcut method for establishing the decision boundaries. The shortcut places the decision boundaries separating distribution (f_i, c_j) from (f_i, c_k) at the intersection of the distribution curves.

The algorithm uses the decision boundaries as upper and lower limits of integration to pick off the conditional probability values for the confusion matrix. In this case, there is relatively small overlap between the two probability density functions. Thus, the resulting confusion matrix has values close to one on the diagonal and values close to zero in the off-diagonal entries. Table 4.2 shows the confusion matrix values derived by the algorithm.

Table 4.2 Confusion Matrix Values Derived by the Algorithm

Class	(f_1, c_A)	(f_1, c_B)
Class A	.9669	.0331
Class B	.0430	.9570

The separation between each column's entries is large. This fact, combined with the observations put forth in Chapter III point to a Bayesian network classifier that should classify correctly a large percentage of the time.

Joining the confusion matrix defined in Table 4.2 with an a priori vector defined with equal entries results provides all the information required to instantiate the network. Once, instantiated, the network is prepared to receive test data. However, the algorithm must first normalize the test data to the respective density functions. With this accomplished, the network is ready to classify the data.

The Matlab script file classify2D.m implements the Bayesian network classification network algorithm and gathers statistics for analyzing the resulting network. The script file receives two training data sets and two test data sets as well as an a priori vector and the decision boundaries. It uses the training data to normalize the test data and to establish the parameters of the Bayesian network classifier. It then simulates the resulting classifier and gathers statistics on how well the network performed against the test data. Table 4.3 shows the results reported by classify2D for the problem defined in Figure 4.4.

Table 4.3 Performance Measurements for a Two Class, One Feature, Bayesian Network Classifier

	Class A	Class B	Total
Percent Correct	96.3	96.2	96.3
Percent Incorrect	03.7	03.8	03.7

The results show the classifier performed very well. In fact, the reported error rate approximates the Bayesian error rate, which has been shown as an upper limit to the performance of a classifier. As a belated observation, the algorithm establishing the decision boundaries places them at the same point from which the Bayes error is calculated. This was noticed after the algorithm for placing the decision boundaries was developed and lends evidence that the algorithm does indeed place the decision boundaries at the optimal points.

The performance of a Bayesian network classifier may improve slightly by tuning the placement of the decision boundaries about their optimal points and changing the values of the a priori vector. However, these adjustments are highly dependent on the current test data set. Tuning that increases performance on this test set may decrease it for a different test set. A second caveat is that changing these parameters may also decrease performance.

The only way to improve performance of the Bayesian network classifier in general is to add more information in the form of additional features. The next section extends the example of this section to the two feature case.

4.4 An Example Bayesian Network Classifier Receiving Two Features

To handle the two feature case, the algorithm simply extends to accommodate the second feature. It performs the exact same functions on the second feature as it did the first. As an example, the classification problem from the last section is expanded to handle another feature. Now the problem involves discriminating between two classes given two feature inputs. The new Bayesian network classifier for this problem requires four nodes which is equal in number to the cardinality of $F \times C$. The root node still has the same two hypotheses. Figure 4.5 displays the Bayesian network classifier for this problem.

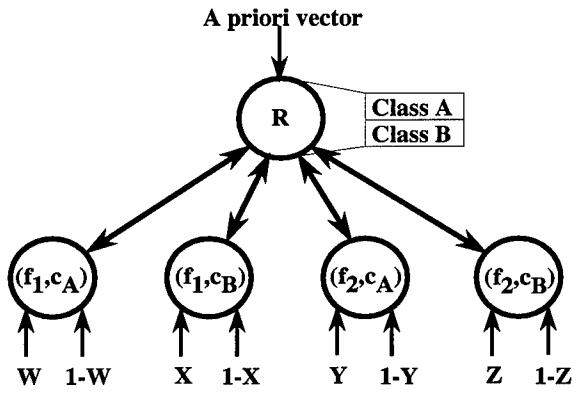


Figure 4.5 A Bayesian Network Classifier For a Two Class, Two Feature Problem

The data sets for the additional feature are again generated by the Matlab RANDN command, scaled by predetermined standard deviations, and offset by predetermined means. The mean and standard deviation for the second feature data associated with class A are 30 and 7 respectively. The mean and standard deviation for the data associated with second feature data for class B are 60 and 12.

The Matlab script file classify4D implements the algorithm to define a Bayesian network classifier for a two class, two feature classification problem. Classify4D performs the same

functions as classify2D except it accounts for the second feature. It begins by receiving four data sets on which to train, one each associated with each feature class pair. It calculates a mean and a standard deviation for each data set and assumes underlying probability distribution is gaussian. The calculated standard deviation and mean for feature one appear in Table 4.1. Table 4.4 presents the calculated values for the second feature.

Table 4.4 Calculated Standard Deviation and Mean Values for a Second Feature

Parameter	(f_2, c_A)	(f_2, c_B)
Std Dev	6.3367	12.1726
Mean	30.8944	60.7808

Figure 4.4 shows the probability density functions for the data sets associated with feature one. Figure 4.6 does the same for the data sets associated with feature two.

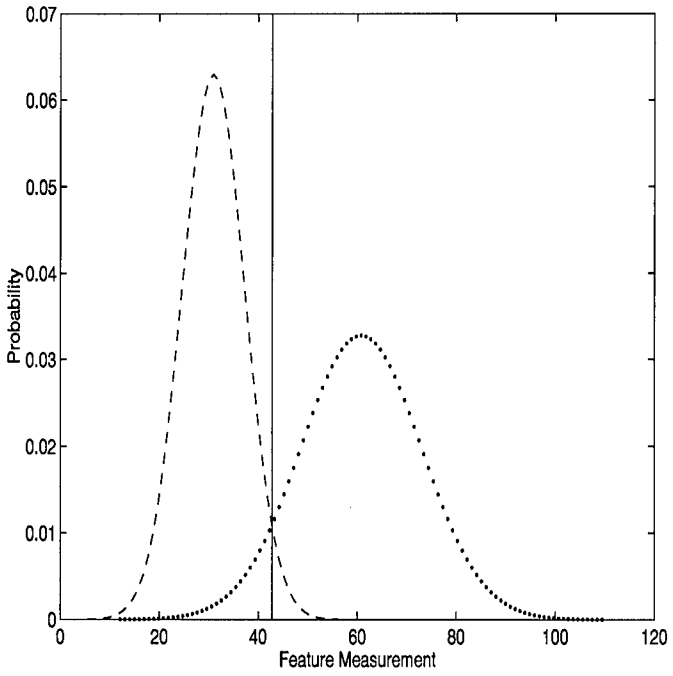


Figure 4.6 Probability Density Functions and Decision Boundaries for the Data Sets Associated with the Second Feature

As an alternative representation, the feature measurements for each class can be graphed onto a 2-D display. Figure 4.7 shows the training data graphed versus feature one and feature two. The asterisks belong to class A while the plus signs belong to class B.

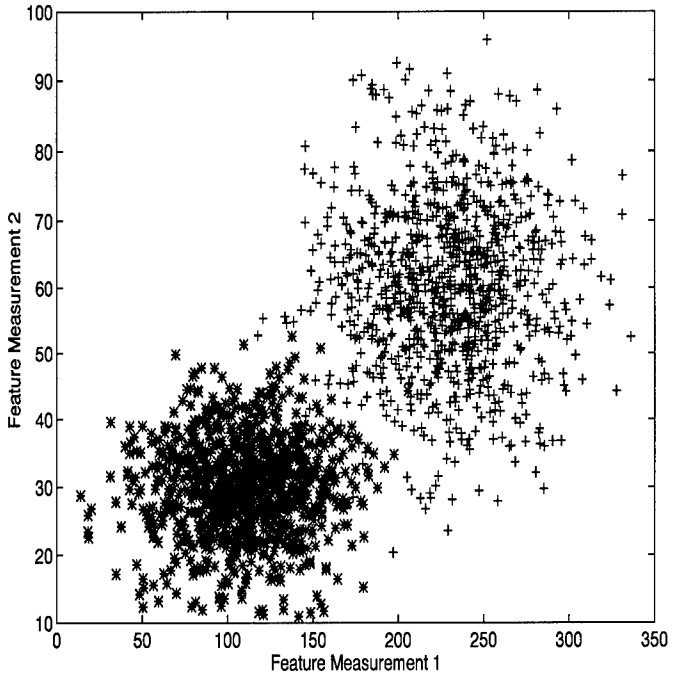


Figure 4.7 Scatter Plot of the Test Data

Figure 4.7 is useful in visually establishing the relationship between the two classes and gives a visual indication of how overlapped the data sets are. However, this representation conveys little information useful to the Bayesian network classifier algorithm. Instead, graphing each feature over all classes, such as Figures 4.4 and 4.6, is more useful.

The algorithm now determines the decision boundaries. The decision boundaries associated with feature one were derived earlier and displayed in Figure 4.4. Figure 4.6 also shows the derived decision boundaries for the second feature. The shortcut method is used to determine the decision boundaries for the second feature. The algorithm simply places the boundaries at the intercept point between the probability density curves. Using the decision boundaries as limits of integration, the algorithm calculates the conditional probabilities and populates the confusion matrix. Table 4.5 lists the elements of the matrix.

An a priori vector is defined with two equal elements that sum to one. With this vector and the confusion matrix available, classify4D simulates the defined Bayesian network classifier. It first, however, normalizes the four test data sets to their respective probability

Table 4.5 Confusion Matrix Values Derived by the Algorithm

Class	(f_1, c_A)	(f_1, c_B)	(f_2, c_A)	(f_2, c_B)
Class A	.9669	.0331	.9701	.0299
Class B	.0430	.9570	.0694	.9306

density curves. This accomplished, the algorithm presents the normalized data to the classifier. Table 4.6 shows the results. For comparison, the estimated Bayes error rate for this problem is 1.7 percent. This estimated value was computed using linear discriminate functions tested on 20,000 data points. The Matlab script files for performing the deriving the estimate appear in Appendix B.

Table 4.6 Performance Measurements for a Two Class, Two Feature, Bayesian Network Classifier

	Class A	Class B	Total
Percent Correct	99.7	91.7	95.7
Percent Incorrect	00.3	08.3	04.3

The overall results are a little disappointing because they decrease slightly from the the one feature example in the previous section. However, this example is successful in showing the utility of the Bayesian network classifier algorithm in mapping the Bayesian network to this two feature problem.

Extending the algorithm to handle more features and classes is straightforward and is just a matter of extending the algorithm to account for the editions. Adding another feature merely requires performing a third iteration of the algorithm for that feature. Adding a new class simply adds a new probability density function for which new decision boundaries must be established. In both cases, additional nodes and data sets are required, but the algorithm easily extends to add new nodes to the root and normalize the new data sets.

4.5 The Bayesian Network Classifier Algorithm Applied to Real World Data

A last example involves applying the Bayesian network algorithm to real data. The data consists of two feature measurements collected for a mammography classification study

conducted by Kocur (6) (7). The Kocur work attempted to classify tumors displayed in mammographs as either benign or malignant based on 21 measured features. The features used in this example are two of the 21 in the Kocur study. These two features were picked specifically because they displayed the greatest disparity between distribution plots of the data for benign an malignant tumors.

The size of each feature set consists of 94 data points, 63 measurements for benign tumors, 31 for malignant. For this example, the data for the two features are divided into two sets. One set is used to train the Bayesian network classifier and the other to is used to test. Thus, 31 of the 63 benign tumor measurements and 15 of the 31 malignant tumor measurements are used to establish the parameters of the network. The remaining samples are used for training. The partitioned data appears in Appendix C.

The goal is to define a Bayesian network classifier capable of discriminating between benign and malignant tumors based on the two selected features. The algorithm begins by defining the topology of the network. Because there are two classes, benign and malignant, and two features, the resulting topology has four evidence nodes connected to a root node with two hypotheses. This is the same topology as displayed in Figure 4.5. Based on the data set aside for training, the algorithm calculates mean and variances for both the benign and malignant classes. Table 4.7 displays the resulting values.

Table 4.7 Calculated Standard Deviation and Mean Values For Mammography Features

Parameter	$(f_1, Benign)$	$(f_1, Malignant)$	$(f_2, Benign)$	$(f_2, Malignant)$
Std Dev	149.945	182.474	168.336	113.323
Mean	76.174	91.721	121.992	66.086

A gaussian distribution is assumed for this data. Figure 4.8 shows the resulting probability density curves for feature one and feature two. The first set of curves correspond to feature one, the second set for feature two. The feature measurement distributions rendered with dashed lines belong to the malignant class.

The algorithm applies the shortcut method to establish the decision boundaries. Figure 4.9 shows the placement of the decision boundaries for both features.

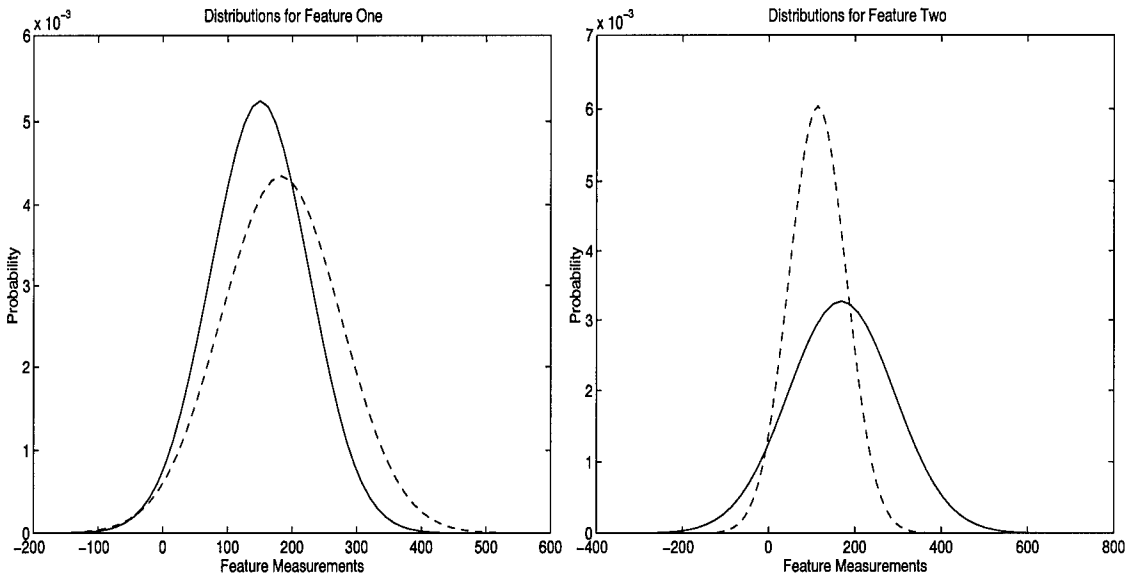


Figure 4.8 Probability Density Functions Resulting From the Calculated Means and Standard Deviations. Dashed Lines Represent the Malignant Class

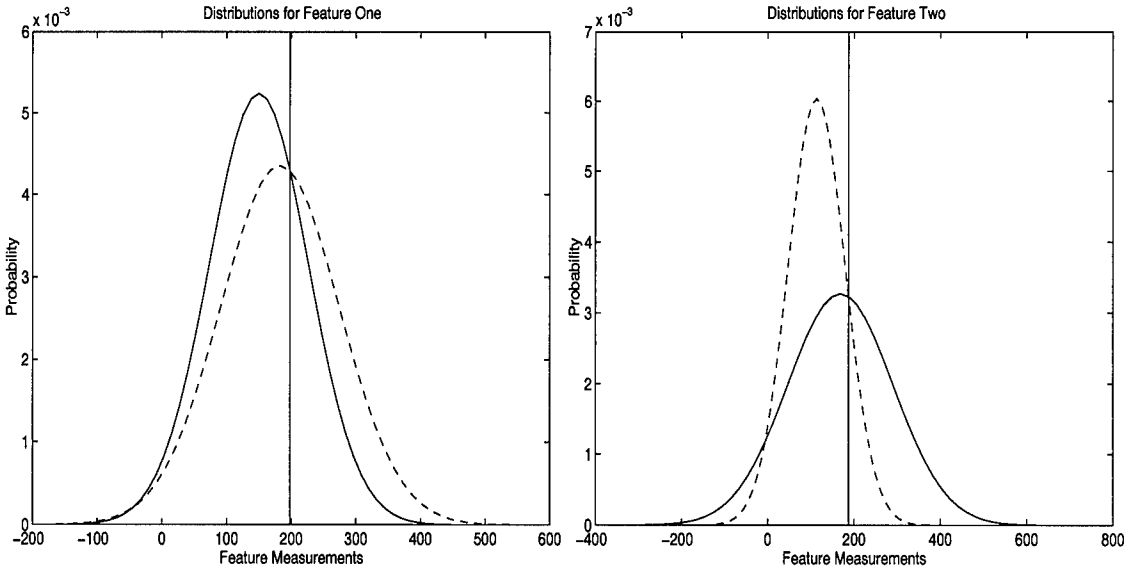


Figure 4.9 Decision Boundaries Separating the Classes Within Each Feature Measurement.

The placement of the decision boundaries at these locations guarantees the columns of the confusion matrix will enjoy the maximum separation between their elements. Table 4.8 lists the elements of the confusion matrix derived by the algorithm.

Table 4.8 Confusion Matrix Values Relating the Benign and Malignant Classes to the Feature Measurements

Class	$(f_1, Benign)$	$(f_1, Malignant)$	$(f_2, Benign)$	$(f_2, Malignant)$
Benign	.7391	.2609	.4344	.5656
Malignant	.5710	.4290	.1297	.8703

The entries in the confusion matrix give the first indication that this classification problem is more difficult than those presented previously. The column entries are much closer to each other and none approach the extremes of one or zero. The scatter plot of Figure 4.10 provides further visual evidence that this is indeed the case.

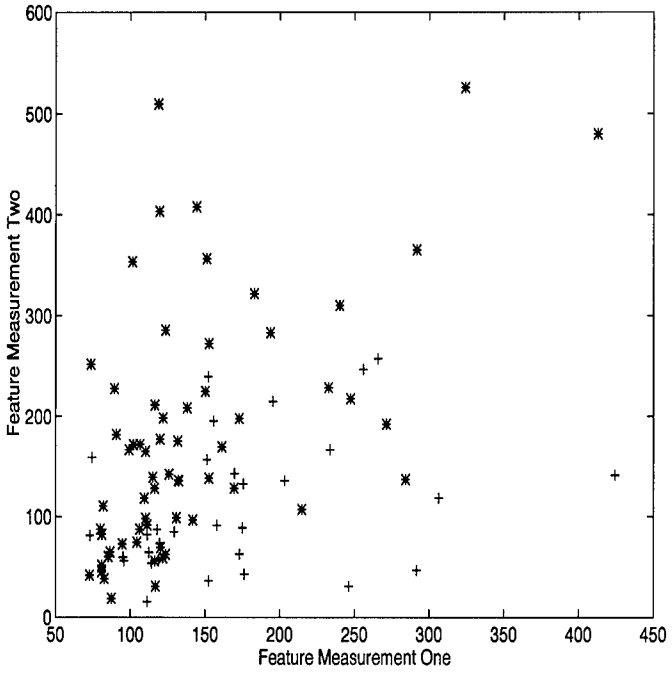


Figure 4.10 Scatter Plot of the Entire Data Set

The Figure shows all 94 data point plotted based on their respective feature measurements. Asterisks indicate the point belongs to the benign class, plus signs indicate malignant.

The high level of overlap and interspersion of the data points between both classes shows the two classes are indeed difficult to separate, leading to the values in the confusion matrix.

The a priori vector is defined to have equal entries that sum to one. The vector, along with the confusion matrix, allows the algorithm to instantiate the Bayesian network classifier. The test data sets are normalized against their associated distribution functions. The classifier receives the normalized inputs and produces the results shown in Table 4.9.

Table 4.9 Performance of the Bayesian Network Classifier Discriminating Between Benign and Malignant Tumors

	Benign	Malignant	Total
Percent Correct	65.6	68.8	66.7
Percent Incorrect	34.3	31.2	33.3

As predicted by the conditional probability values in the confusion matrix, discriminating between benign and malignant tumors using these two feature sets proved very difficult. In fact, the defined Bayesian network classifier did little better than if it had merely classified everything as benign. However, the classifier did produce results comparable to other classifying techniques. Table 4.10 compares the results achieved here with results achieved by three other classifiers using the same feature sets.

Table 4.10 Performance Comparisons of Four Classifiers Using the Same Feature Sets

Classifier	Percent Benign Correct	Percent Malignant Correct	Total
Bayesian Network	65.6	68.7	66.7
Gaussian	88.9	12.9	63.8
Neural Network	74.6	25.8	58.5
K Nearest Neighbor	87.3	25.8	67.0

Of note is the marked increase in correctly classified malignant tumors exhibited by the Bayesian network classifier. If a cost function is assigned such that the cost for incorrectly classifying a malignant tumor is greater than the cost of incorrectly classifying a benign tumor, the Bayesian network classifier actually performs better. At any rate, these results demonstrate the Bayesian network classifier algorithm is capable of defining a classifier that performs comparable to other classifiers.

4.6 Concluding Remarks

An algorithm now exists to define a Bayesian network classifier for a given classification problem. The algorithm specifies the topology of the classifier and provides an method for deterministically establishing the conditional probabilities required for a Bayesian network. Further, the establishment of these parameters is data driven. The algorithm is scalable to any number of features and classes. It is also applicable to multi-level Bayesian networks through the realization that the results of one level are merely the inputs to the next. Several examples involving a small number of features and classes were presented in this chapter to illustrate the operation of the algorithm and to show that the resulting Bayesian network classifiers produced results approximating the Bayes error rate and comparable to other classifiers. It should be pointed out that this chapter did not present exhaustive analysis on the performance of the Bayesian network classifier. Thus, no claims concerning the performance of the classifier are being made here. However, the examples do show that the Bayesian network classifier algorithm can define classifiers capable of producing "ball park" results.

V. Recommendations and Conclusions

5.1 Recommendations for Future Research

As a result of this work, several key issues are prime for future research. These include removing limitations from the decision region display models of Chapter III and assessing the performance of classifiers defined by the Bayesian network classifier algorithm of Chapter IV. Specific recommendations are outlined in the following paragraphs.

Future work must assess the performance of classifiers defined by the Bayesian network classifier algorithm. The research should include exhaustive testing on both simulated and real world data sets to determine the capabilities of the classifiers. Comparisons should also be made to other classifiers to assess relative performance. This area is the most critical issue requiring additional work.

Additional work concerning the Bayesian network classifier algorithm should extend the procedure to define multi-level Bayesian network classifiers. Conceptually, extending the algorithm is just a matter of converting the output of one level into the input for the next and applying the algorithm at this point. However, characterizing the output data in terms of input probability distributions may prove difficult. These difficulties need to be assessed.

Turning attention to the decision region models, future work should assess the effect of the a priori vector on the decision regions. Due to time constraints, this work paid little attention to this important vector in the case where the vector elements are not equivalent. The current software for producing the decision regions can handle any a priori vector and represent a good starting point.

Future work should extend the decision region models to systems with arbitrary numbers of input nodes and hypotheses. The coding techniques used in this work are sufficient to accomplish this extension, but recoding the governing equations in an object oriented language may result in easier coding and more flexibility.

Finally, follow on work should extend the decision region models to account for multi-level Bayesian networks. An observation key to realizing this extension is that the λ vector of any node in a Bayesian network serves as the input node to the next level. Conceptually then, future work could link together the existing models such that the output of one feeds the input of another. The current implementation of the models as Matlab script files may not be ideal for this extension. It may be better to recode the model in an object oriented language where the governing equation for each node can be encapsulated.

The preceding suggestions for follow on work to this thesis list only a few key areas. Much is still to be learned concerning Bayesian network and their utility as classifiers.

5.2 Conclusions

In an effort to develop advanced systems such as Automatic Target Recognition, extensive research is being conducted to develop better classifiers. Bayesian networks represent a promising new classifier architecture that may result in the better classifiers being sought. This work focused on developing Bayesian networks as classifiers and produced three key results.

First, analyses of evidence nodes show that a linear relationship exists between the inputs to the nodes and the diagnostic support output from the node. The slope of the relationship is controlled by the associated conditional probability value. A probability value greater than .5 results in a positive slope, indicating the diagnostic support for a hypothesis increases given increases in the evidence. A probability value less than .5 results in a negative slope, indicating the diagnostic support for a hypothesis decreases given increases in the evidence. A probability value equal to .5 results in a slope of zero, essentially cutting the evidence off from consideration by the network. A key conclusion from these observations is that lack of evidence can be just as important for classifying objects in a Bayesian networks as the presence of it.

The second key result is the set of Matlab script files for displaying and analyzing the decision regions formed by a Bayesian network. These script files receive a confusion matrix

and an a priori vector and produce the resulting decision regions. With these software tools, researchers can now visualize the decision regions formed by a Bayesian network and analyze the effect on these regions as parameters change.

The third, and perhaps most important, result is an algorithm that defines a Bayesian network classifier for a given classification problem. The algorithm incorporates methodologies to precisely define the topology of the network and to precisely establish the conditional probabilities required by the network. The algorithm is completely data driven, meaning each defined classifier is uniquely specified for the classification problem at hand. The Bayesian network classifiers defined by the algorithm exhibit performance approximating the Bayes error rate and comparable to other classification methods for both simulated and real world classification problems. However, further testing must be accomplished before any general claims concerning the performance of Bayesian network classifiers can be made.

The tools and algorithms developed in this work are humbly submitted to help researchers understand Bayesian networks and their utility as classifiers. It is hoped that this research will lead to robust and viable Bayesian network classifiers applied to a multitude of classification challenges.

Appendix A. Bayesian Network Tutorial

A.1 Introduction

The examples of Section 2.3 are revisited here except they are presented in painstaking detail. Each step for initializing the networks is covered as well as each step for propagating information through the network once initialized. Every calculation is also presented and discussed. This discussion begins with initializing the example network of Figure A.1.

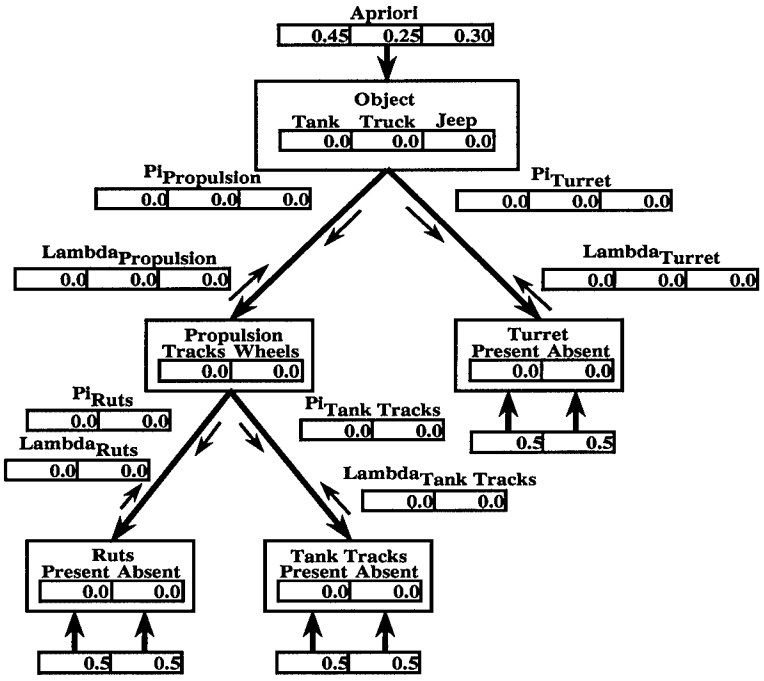


Figure A.1 An Example Bayesian Network with A Priori Values Established

A.1.1 Initializing the Network. With the conditional probability tables and the a priori vector established, the network initializes by propagating the a priori information through the network. Precisely how the network accomplishes this is slightly ambiguous. The only real information presented to the network at this time is the a priori vector. This requires the root node to update. However, Equation 2.21 shows the belief vector of the root node cannot be updated without a λ vector. These in turn are dependent on λ message vectors which

themselves are dependent on the λ vectors of the child nodes. This begins a dependency chain between λ vectors and λ message vectors that terminates at the evidence nodes. The required first step, therefore, is to propagate the inputs on the evidence nodes through the network to establish λ message vectors and λ vectors at every level. This is not enough information to establish belief vectors at every node. It facilitates only the calculation of the belief vector for the root node. The network must still propagate π message vectors back down the network and establish π vectors at each node. With π and λ vectors in hand, the nodes can quickly calculate belief vectors.

Concentrating on the Ruts node and its inputs, the network begins by applying Equation 2.22. However, as the inputs to an evidence node is equivalent to a single λ message vector, Equation 2.22 degenerates into a product of a single term. Thus, the λ vector for a leaf node is equivalent to the input vector. Note that even though the Ruts node now has an established λ vector, it cannot calculate a belief vector because no π vector is established. The network instead derives a λ message vector by applying Equation 2.24.

$$\begin{aligned} \lambda_{Ruts}(Tracks) &= P\left[Present|Tracks\right]\lambda\left(Present\right) \\ &+ P\left[Absent|Tracks\right]\lambda\left(Absent\right) \qquad \text{(A.1)} \\ \lambda_{Ruts}(Wheels) &= P\left[Present|Wheels\right]\lambda\left(Present\right) \\ &+ P\left[Absent|Wheels\right]\lambda\left(Absent\right) \qquad \text{(A.2)} \end{aligned}$$

Table 2.3 provides they required values. Note that the conditional probabilities for these calculations consume all the entries in a single column of the Table.

$$\lambda_{Ruts}(Tracks) = .2 \times .5 + .8 \times .5 = .5$$
$$\lambda_{Ruts}(Wheels) = .6 \times .5 + .4 \times .5 = .5$$

The calculations result in a λ_{Ruts} message vector equal to the Ruts λ vector which, in turn, is equal to the input vector. This observation leads to a generalization of Equation 2.24. If the elements of a λ vector for a given node are all some value b then Equation A.3 results, which is valid over all j.

$$\lambda_B(A_j) = \sum_i P\left[B_i|A_j\right] b = b \sum_i P\left[B_i|A_j\right] = b \tag{A.3}$$

The summation disappears in Equation A.3 because it sums column entries of a conditional probability table which sum to one. Using Equation A.3, the network in Figure A.1 quickly establishes λ vectors and λ message vectors for all nodes. Figure A.2 shows the Bayesian network of Figure A.1 populated with λ message vectors.

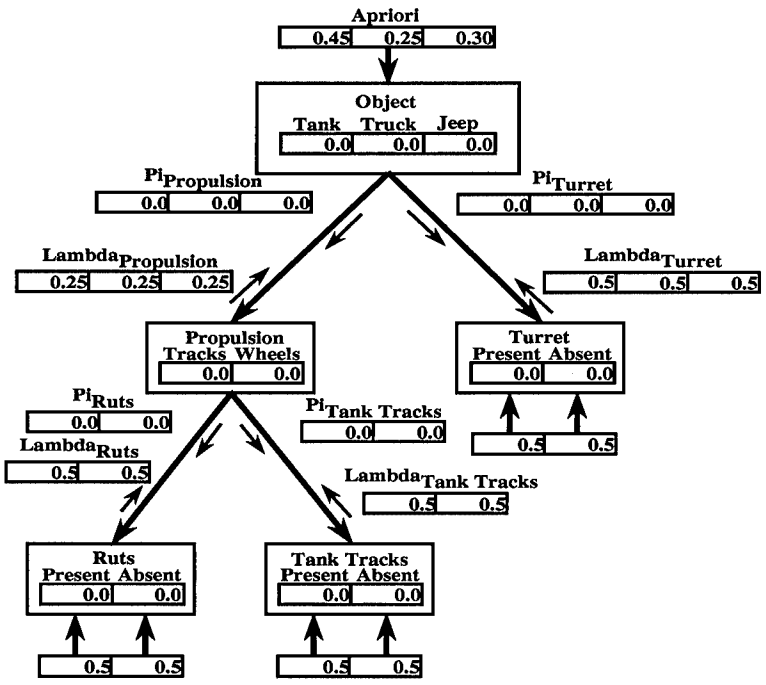


Figure A.2 Update to λ Message Vectors

The $\lambda_{Propulsion}$ message vector requires some explanation. Message vectors λ_{Ruts} and $\lambda_{TankTracks}$ combine via Equation 2.22 to form the λ vector for the Propulsion node. As this combination is merely an element by element multiplication of two equal vectors, the

elements of the resulting vector are also equivalent and equal to the square of the elements of the message vectors. Because the elements of the Propulsion node's λ vector are equivalent, Equation A.3 applies where $b = .5^2$. The λ vector for the root node is a similar combination of the $\lambda_{Propulsion}$ and λ_{Turret} message vectors. The λ vector for the Object node is thus [.125 .125 .125].

With the root node λ vector established, the network can now calculate the Object node belief vector via Equation 2.8. As the Object node is a root node, the π vector is the a priori vector. The following calculations employ an intermediate step because the Equation 2.21 requires a the intermediate values to calculate the normalization constant α.

$$\begin{aligned} \beta(Tank) &= \pi(Tank)\lambda(Tank) \\ \beta(Truck) &= \pi(Truck)\lambda(Truck) \\ \beta(Jeep) &= \pi(Jeep)\lambda(Jeep) \end{aligned}$$

$$\begin{aligned} \beta(Tank) &= .45 \times .125 \\ \beta(Truck) &= .25 \times .125 \\ \beta(Jeep) &= .30 \times .125 \end{aligned}$$

The normalization value $1/\alpha$ is the sum of the β values.

$$\alpha = .45 \times .125 + .25 \times .125 + .30 \times .125 = .125(.45 + .25 + .30) = .125$$

Finally, the elements of the belief vector are the β values multiplied by α.

$$BEL(Tank) = \frac{.45 \times .125}{.125} = .45$$
$$BEL(Truck) = \frac{.25 \times .125}{.125} = .25$$
$$BEL(Jeep) = \frac{.30 \times .125}{.125} = .30$$

Figure A.3 shows the update to the belief vector of the Object node. Because the elements of the Object λ vector are equivalent and the elements of the π vector sum to one, the belief vector is equivalent to the π vector. This leads to another generalization. For any node, if the elements of the λ vector are equivalent and the sum of the π vector elements sum to one, then the belief vector will be equivalent to the π vector. Subsequent calculations in this example will rely heavily on this generalization.

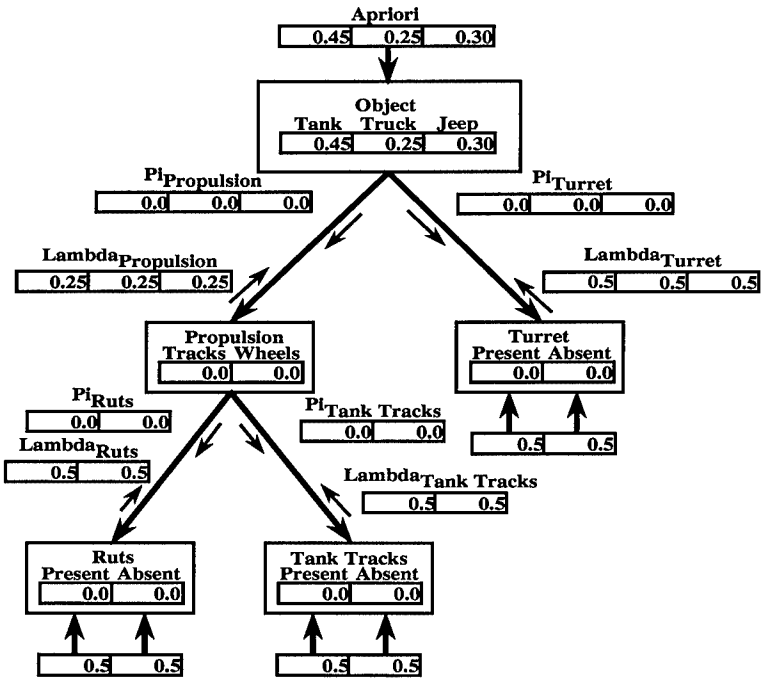


Figure A.3 Update of Object Node Belief Vector

The network, therefore, initially adopts the a priori vector as its belief vector for the Object node. Conceptually, this is a satisfying result as the network has yet to receive any information to sway the belief away from what the network anticipates. The network now

propagates this initial belief throughout the network establishing initial belief vectors in all the nodes. Equation 2.25 calculates π message vectors to the child nodes of the Object node. For this node, the product term in the equation is a term of only one value because there are only two child nodes providing λ message vectors and Equation 2.25 includes only one. The following calculations produce the elements of the vectors. The π message vector for the Propulsion node is calculated first.

$$\begin{aligned}
\pi_{Propulsion}(Tank) &= \alpha\pi(Tank)\lambda_{Turret}(Tank) \\
\pi_{Propulsion}(Truck) &= \alpha\pi(Truck)\lambda_{Turret}(Truck) \\
\pi_{Propulsion}(Jeep) &= \alpha\pi(Jeep)\lambda_{Turret}(Jeep)
\end{aligned}$$

$$\begin{aligned}
\pi_{Turret}(Tank) &= \alpha\pi(Tank)\lambda_{Propulsion}(Tank) \\
\pi_{Turret}(Truck) &= \alpha\pi(Truck)\lambda_{Propulsion}(Truck) \\
\pi_{Turret}(Jeep) &= \alpha\pi(Jeep)\lambda_{Propulsion}(Jeep)
\end{aligned}$$

These calculations are similar to the previous calculation of the belief vector for the Object node. The elements of the λ vector involved are equal, the elements of the π vector sum to one, and the resulting vector is normalized by a normalization factor α. Thus, by analogy, the vectors $\pi_{Propulsion}$ and π_{Turret} are equal to the Object node's π vector. Figure A.4 illustrates the update to the π message vectors.

New π message vectors arriving the the Turret and Propulsion nodes stimulate them to calculate new π vectors via Equation 2.23. The following equations show these calculations beginning with the Propulsion node.

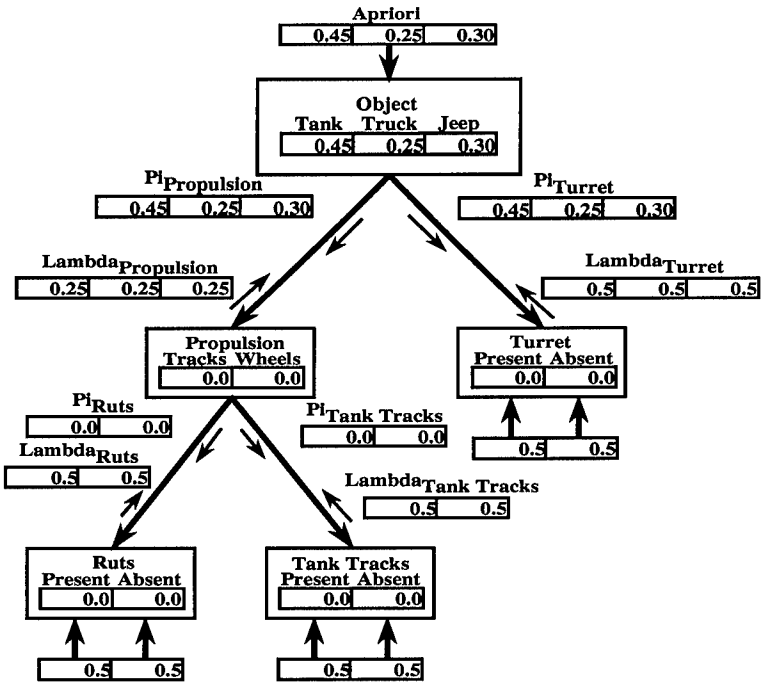


Figure A.4 Update of π Message Vectors

$$\begin{aligned}\pi(Tracks) &= P[Tracks|Tank]\,\pi_{Propulsion}(Tank) \\ &+ P[Tracks|Truck]\,\pi_{Propulsion}(Truck) \\ &+ P[Tracks|Jeep]\,\pi_{Propulsion}(Jeep)\end{aligned}$$

$$\begin{aligned}\pi(Wheels) &= P[Wheels|Tank]\,\pi_{Propulsion}(Tank) \\ &+ P[Wheels|Truck]\,\pi_{Propulsion}(Truck) \\ &+ P[Wheels|Jeep]\,\pi_{Propulsion}(Jeep)\end{aligned}$$

$$\begin{aligned}\pi\,(Present) &= P\,[Present|Tank]\,\pi_{Turret}(Tank)\\ &+ P\,[Present|Truck]\,\pi_{Turret}(Truck)\\ &+ P\,[Present|Jeep]\,\pi_{Turret}(Jeep)\end{aligned}$$

$$\begin{aligned}\pi\,(Absent) &= P\,[Absent|Tank]\,\pi_{Turret}(Tank)\\ &+ P\,[Absent|Truck]\,\pi_{Turret}(Truck)\\ &+ P\,[Absent|Jeep]\,\pi_{Turret}(Jeep)\end{aligned}$$

Figure A.4 and Tables 2.1 and 2.2 provide the required values for these calculations.

$$\pi(Tracks) = .95 \times .45 + .05 \times .25 + .01 \times .30 = .443$$
$$\pi(Wheels) = .05 \times .45 + .95 \times .25 + .99 \times .30 = .557$$

$$\pi(Present) = .999 \times .45 + .02 \times .25 + .005 \times .30 = .456$$
$$\pi(Absent) = .001 \times .45 + .98 \times .25 + .995 \times .30 = .544$$

The π vector for the Propulsion node is thus [.443 .557] and for the Turret node [.456 .544]. Note that the elements of these vectors sum to one. A node's π vector combines the likelihood that an object will be encountered with the likelihood that a certain feature is present given the object. The output is a measure of the anticipation that a feature is present before examining any direct evidence. Thus the anticipatory information supplied to the Propulsion node indicates that, without external evidence, the network predicts an object is more likely to have wheels than tracks. Likewise, the Turret node indicates an object is less likely to have a turret. This is consistent with the information supplied to the network

at present. The a priori vector makes clear that the network anticipates discovering more vehicles with wheels and no turrets than vehicles with tracks and turrets.

With π vectors in hand, the network can now calculate belief values for the Propulsion and Turret nodes. This step combines anticipatory information provided by the π vectors with diagnostic information from the λ vectors and has the effect of tempering what the network expects with what it currently observes. Conversely, this step tempers what the network observes with what it expects. The largest values in belief vectors occur when the π and λ values agree. In other words, the network achieves its best results when it sees what it believes and believes what it sees.

For the case of the Propulsion and Turret nodes, the elements of the λ vectors are all equal and the elements of the π vectors sum to one. Thus the general case spoke of earlier exists and the belief vectors for these nodes equate to the π vectors. This is a consistent result because the network has yet to receive any diagnostic information to sway it from its anticipated beliefs. Figure A.5 illustrates the update to the belief values of the Propulsion and Turret nodes.

The network has only belief vectors for nodes Ruts and Tank Tracks left to establish to complete the initialization. Calculating the π message vectors from the Propulsion node to the Ruts and Tank Tracks nodes is much the same as calculating π vector messages from the Object node. The elements of both opposing λ vector messages from the child nodes are equal. The elements of the Propulsion node π vector sum to one. Thus, as shown previously, the π vector is merely forwarded as the π message. Figure A.6 records the change.

Calculating π vectors for nodes Ruts and Tank Tracks requires Equation 2.23, Figure A.6, and Tables 2.3 and 2.4. In the following calculations the π vector for the Ruts node is calculated first.

$$\pi\left(Present\right) \;=\; P\left[Present|Tracks\right]\pi_{Ruts}(Tracks)$$

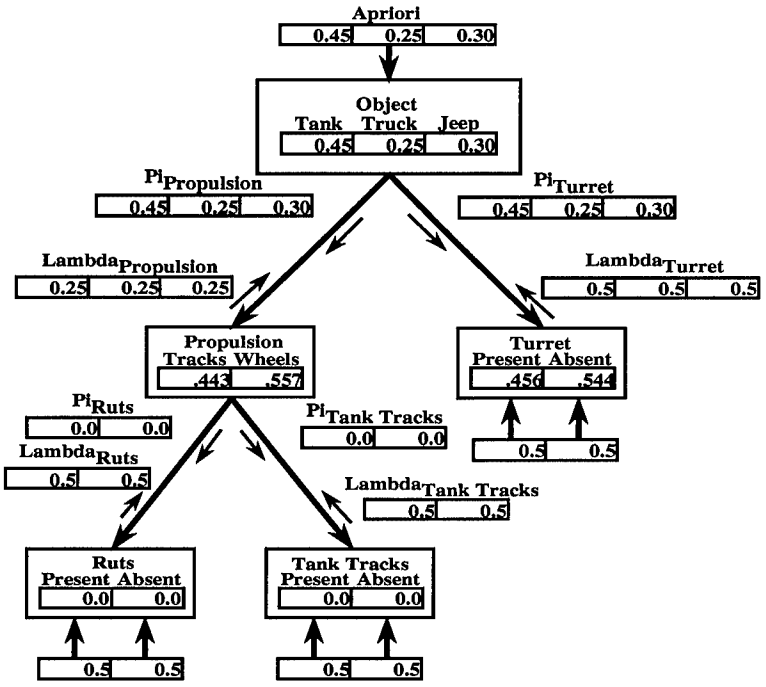


Figure A.5 Update of Propulsion and Turret Node Belief Vectors

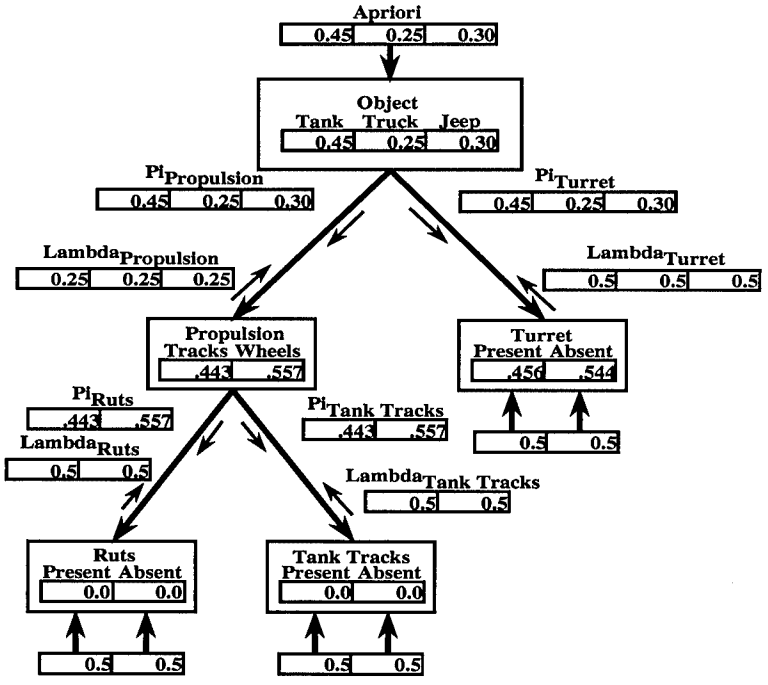


Figure A.6 Update of π Message Vectors from the Propulsion Node

$$+ \quad P\left[Present|Wheels\right]\pi_{Ruts}(Wheels)$$

$$\begin{aligned}\pi\left(Absent\right) &= P\left[Absent|Tracks\right]\pi_{Ruts}(Tracks)\\ &+ P\left[Absent|Wheels\right]\pi_{Ruts}(Wheels)\end{aligned}$$

$$\begin{aligned}\pi\left(Present\right) &= P\left[Present|Tracks\right]\pi_{TankTracks}(Tracks)\\ &+ P\left[Present|Wheels\right]\pi_{TankTracks}(Wheels)\end{aligned}$$

$$\begin{aligned}\pi\left(Absent\right) &= P\left[Absent|Tracks\right]\pi_{TankTracks}(Tracks)\\ &+ P\left[Absent|Wheels\right]\pi_{TankTracks}(Wheels)\end{aligned}$$

$$\pi(Present) = .20 \times .443 + .60 \times .557 = .423$$
$$\pi(Absent) = .80 \times .443 + .40 \times .557 = .577$$

$$\pi(Present) = .94 \times .443 + .001 \times .557 = .417$$
$$\pi(Absent) = .06 \times .443 + .999 \times .557 = .583$$

The π vector for the Ruts node is thus [.423 .577] and for the Tank Tracks node [.417 .583]. Its interesting to note that, initially, the network is anticipating encountering neither ruts or tank tracks. This is due to the higher a priori distribution of wheeled vehicles versus tracked vehicles and the relatively low probability of a wheeled vehicle making ruts.

Calculating the belief vectors for nodes Ruts and Tank Tracks is an easy step from this point. The elements of the nodes' λ vectors are equal and the elements of the π vectors sum to one. Thus, via the previously established generalization, the belief vectors are equal to the π vectors. This is again consistent because the network has yet to receive outside evidence . The network is now fully initialized and ready to receive evidence. Figure A.7 shows the fully initialized network.

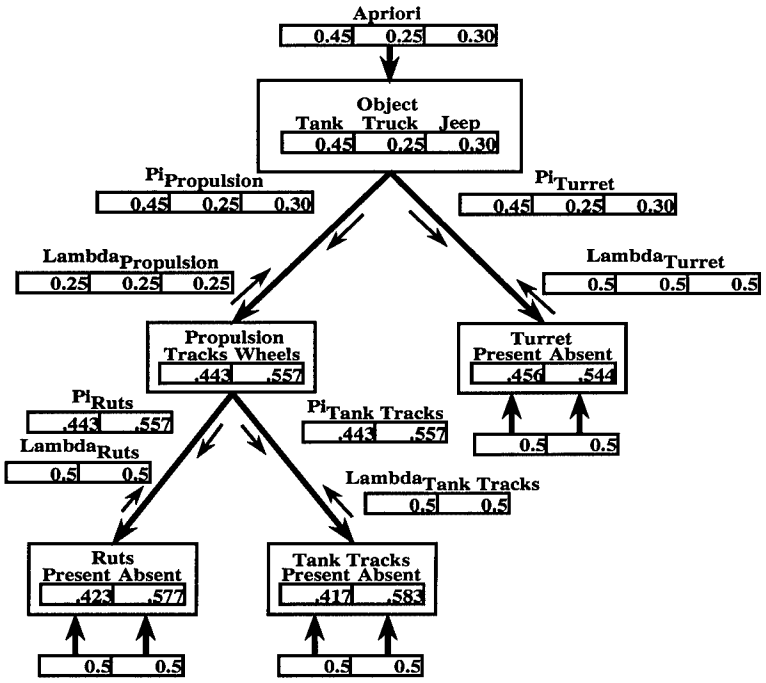


Figure A.7 A Fully Initialized Bayesian Network

A.1.2 Propagating Evidence Through the Network. The network in Figure A.7 receives evidence in a variety of ways. It may receive evidence incrementally to one evidence node, incrementally over all nodes, instantaneously at one evidence node, or instantaneously over all nodes. The method for presenting evidence does not effect the final belief vectors after the network has received all the evidence. However, observing the belief vectors change as the network receives each piece of information provides insight into the behavior of the network. For this example, therefore, the network receives evidence instantaneously one node

at the time. This approach allows observation into how the belief vectors change as evidence propagates through the network.

Evidence presented to the network can originate essentially anywhere, from human operators, to the measurements of sensors, to the output of software routines. In this example, assume the network receives evidence from feature extractors run against a SAR image containing exactly one object. Before proceeding, however, an assumption must be made about the evidence presented to the network. That assumption is the evidence values presented to each evidence node must sum to one. The reason for this is that the network assumes all presented evidence is in support of one hypothesis or another. Evidence values that do not sum to one imply the existence of additional hypotheses that the network does not support.

To begin, assume the appropriate feature extractor presents an input vector of of [.43 .57] to the Ruts node. Figure A.8 shows the Bayesian network after the appropriate feature extractor presents evidence to the Ruts node.

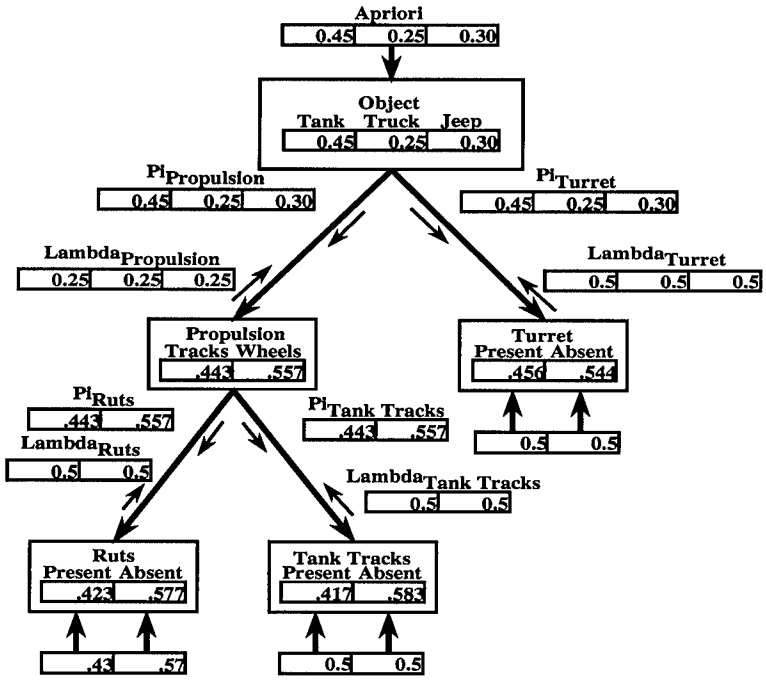


Figure A.8 Evidence Applied to the Ruts Node

New evidence stimulates the Ruts node to update and cycle through the six steps for performing it.

Step 1: Inspect all messages received from the parent and child nodes.

The Pi_{Ruts} field of Figure A.8 provides the π message vector from the parent node. As Ruts is an evidence node, the lone λ message vector is the input itself.

Step 2: Calculate the new λ vector via Equation 2.22

As there is only one λ message vector, Equation 2.22 degenerates into a product of one term. The λ message vector becomes the λ vector. This observation holds for all evidence nodes and nodes with only one λ message vector. The λ vector for the Ruts node is [.43 .57].

Step 3: Calculate the new π vector via Equation 2.23

The required conditional probabilities reside in Table 2.3. This calculation appeared previously but it is repeated for convenience.

$$\pi(Present) = .20 \times .443 + .60 \times .557 = .423$$
$$\pi(Absent) = .80 \times .443 + .40 \times .557 = .577$$

Thus the π vector is [.423 .577].

Step 4: Calculate the new belief vector via Equation 2.21.

Normalizing the result requires the node to calculate the belief vector in steps.

$$\beta(Present) = .423 \times .43 = .182$$
$$\beta(Absent) = .577 \times .57 = .329$$
$$\frac{1}{\alpha} = \beta(Present) + \beta(Absent) = .182 + .329 = .511$$
$$BEL(Present) = .182/.511 = .356$$
$$BEL(Absent) = .329/.511 = .644$$

The new belief vector is [.356 .644]

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent.

$$\lambda_{Ruts}(Tracks) = .20 \times .43 + .80 \times .57 = .542$$

$$\lambda_{Ruts}(Wheels) = .60 \times .43 + .40 \times .57 = .486$$

The new λ_{Ruts} message vector is [.542 .486]

Step 6: Calculate a new π message vector for each child node via Equation 2.25. Transmit these to the appropriate child nodes.

Since the Ruts node has no child nodes, this step is not applicable.

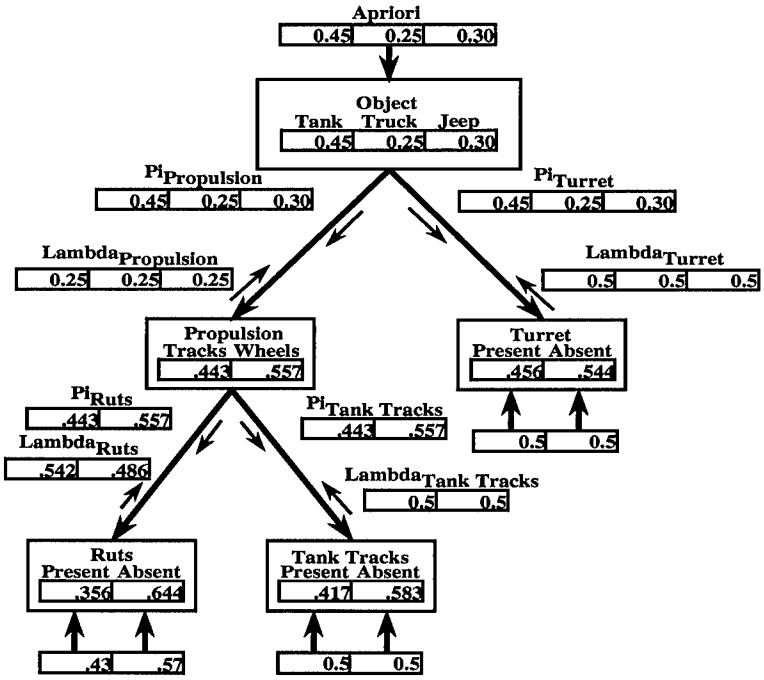


Figure A.9 Update of the Ruts Node

A quick comparison with Figure 2.11 shows a decrease in the network's belief that ruts are present. This decrease is a result of minimal support for the presence of ruts supplied by the feature extractor. This weak evidence, coupled with the network anticipating the absence of ruts, decreases the networks belief that ruts are present. The λ_{Ruts} message vector conveys

this information to the rest of the network. The elements of the vector indicate increased support for tracked vehicles.

The new λ_{Ruts} message vector to the Propulsion node stimulates it to update. The node runs through the six steps to update.

Step 1: Inspect all messages received from the parent and child nodes.

The $Pi_{Propulsion}$ field of Figure A.9 provides the π message vector from the parent node. There are two λ message vectors, λ_{Ruts} and $\lambda_{TankTracks}$.

Step 2: Calculate the new λ vector via Equation 2.22

$$\lambda(Tracks) = .542 \times .5 = .271$$
$$\lambda(Wheels) = .486 \times .5 = .243$$

The λ vector for the Propulsion node is [.271 .243].

Step 3: Calculate the new π vector via Equation 2.23

The required conditional probabilities reside in Table 2.1. This calculation appeared previously but it is repeated for convenience.

$$\pi(Tracks) = .95 \times .45 + .05 \times .25 + .01 \times .30 = .443$$
$$\pi(Wheels) = .05 \times .45 + .95 \times .25 + .99 \times .30 = .557$$

Thus the π vector is [.443 .557].

Step 4: Calculate the new belief vector via Equation 2.21.

Normalizing the result requires the node to calculate this vector in steps.

$$\beta(Tracks) = .443 \times .271 = .1200$$

$$\beta(Wheels) = .557 \times .243 = .1354$$

$$\frac{1}{\alpha} = \beta(Present) + \beta(Absent) = .120 + .1354 = .2554$$

$$BEL(Tracks) = .1200/.2554 = .470$$

$$BEL(Wheels) = .1354/.2554 = .530$$

The new belief vector is [.470 .530]

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent.

$$\lambda_{Propulsion}(Tank) = .95 \times .271 + .05 \times .243 = .270$$

$$\lambda_{Propulsion}(Truck) = .05 \times .271 + .95 \times .243 = .244$$

$$\lambda_{Propulsion}(Jeep) = .01 \times .271 + .99 \times .243 = .243$$

The new $\lambda_{Propulsion}$ message vector is [.270 .244 .243]

Step 6: Calculate a new π message vector for each child node via Equation 2.25. Transmit these to the appropriate child nodes.

Under the rules for propagating information through the Bayesian network, an updating node sends no message to the node that caused it to update. Thus calculating a π message vector for the Ruts node is not required. However, a π message vector is required for the Tank Tracks node. The normalization factor requires the network to perform the calculation is steps.

$$\beta_{TankTracks}(Tracks) = \pi(Tracks)\lambda_{Ruts}(Tracks)$$

$$\beta_{TankTracks}(Wheels) = \pi(Wheels)\lambda_{Ruts}(Wheels)$$

$$\beta_{TankTracks}(Tracks) = .443 \times .542 = .240$$

$$\beta_{TankTracks}(Wheels) = .557 \times .486 = .271$$

$$\frac{1}{\alpha} = \beta(Tracks) + \beta(Wheels) = .240 + .271 = .511$$

$$\pi_{TankTracks}(Tracks) = .240/.511 = .470$$

$$\pi_{TankTracks}(Tracks) = .271/.511 = .530$$

The new $\pi_{TankTracks}$ message vector is [.470 .530]. This is the same as the belief vector and points out another generalization. If a node has only two λ message vectors, and one or both of those have elements with equal values, the the belief vector and the π message vector will be the same. The following equations show why. Let λ_A be a message vector from node A and λ_B be a message vector from node B. Message vector π_B is from the parent node to node B. The vector notation in the following equations are unorthodox and meant to express element by element multiplication.

$$\lambda = \lambda_A \lambda_B$$

$$\lambda = [x\, y\, z][b\, b\, b]$$

$$\lambda = [(x \cdot b)\ (y \cdot b)\ (z \cdot b)]$$

$$\beta_{BEL} = \pi\lambda$$

$$\beta_{BEL} = [i\, j\, k][(x \cdot b)\ (y \cdot b)\ (z \cdot b)]$$

$$\beta_{BEL} = [(i \cdot x \cdot b)\ (j \cdot y \cdot b)\ (k \cdot z \cdot b)]$$

$$\alpha = (i \cdot x \cdot b)\ + (j \cdot y \cdot b) + (k \cdot z \cdot b)$$

$$\alpha = b((i \cdot x)\ + (j \cdot y) + (k \cdot z))$$

$$BEL = \frac{1}{\alpha}\beta_{BEL}$$

Normalizing by α divides a b out of every element in the β_{BEL} vector. The normalization factor becomes $\alpha_\pi = \alpha/b$.

$$BEL = \frac{1}{\alpha_\pi}[(i \cdot x)\ (j \cdot y)\ (k \cdot z)]$$

$$BEL = \frac{1}{\alpha_\pi}\pi\lambda_A = \pi_B$$

Thus, a π vector messages equals the belief vector if the node receives only two λ message vectors and one of those vectors has equal elements.

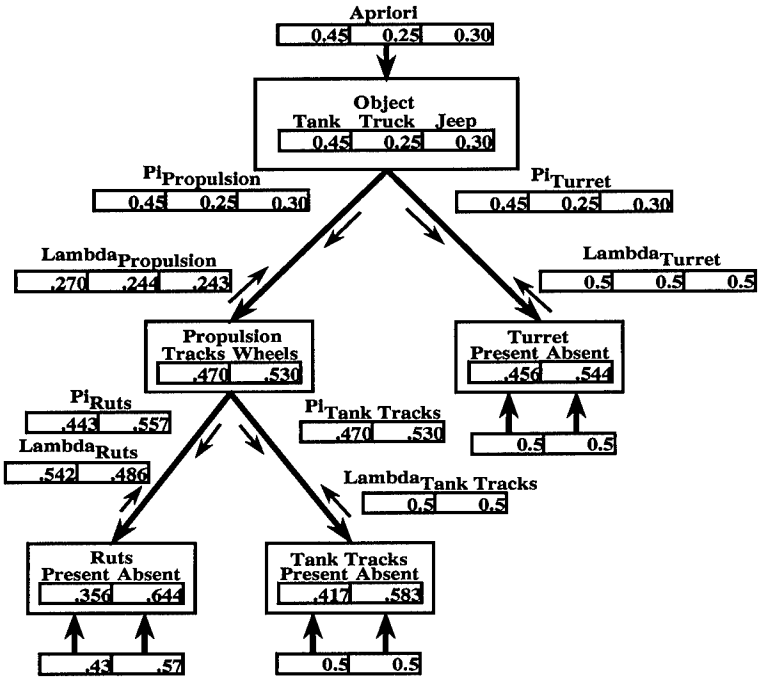


Figure A.10 Update of the Propulsion Node

Figure A.10 displays the network after the update. A quick comparison with Figure 2.12 shows that the weak evidential support for the presence of ruts now manifests itself as weakened belief that the object is a wheeled vehicle. Consequently, the π vector message to the Tank Tracks node shows increased anticipatory support for the presence of tank tracks and the λ message vector to the Object node reflects increased diagnostic support that the object is a tank. This second assertion may not be initially clear because the element values in the $\lambda_{Propulsion}$ message vector actually decrease. Pearl's algorithm for calculating λ message vectors does not call for normalizing them as it does for belief vectors or π message vectors. This makes

it more difficult to assess at a glance the relative strengths of the diagnostic support for the hypotheses of the targeted node. However, elements with larger values indicate stronger support for the associated hypothesis.

Message vectors to the Tank Tracks and Object nodes stimulate both to update. The Tank Tracks node is updated first. This is an arbitrary decision as the network may update any node requiring updating at any time. The network follows the six steps for updating the node. As every calculation required for this update appears previously, only the final vector values are presented along with analysis to their meaning.

Step 1: Inspect all messages received from the parent and child nodes.

The $Pi_{TankTracks}$ field of Figure A.10 provides the π message vector from the parent node. The λ message vector is merely the input to the node.

Step 2: Calculate the new λ vector via Equation 2.22

Using an earlier observation, the λ vector is simply the input vector. Thus the λ vector is [.5 .5].

Step 3: Calculate the new π vector via Equation 2.23

The required conditional probabilities reside in Table 2.4. With a new π message vector from the parent, the π vector becomes [.442 .558].

Step 4: Calculate the new belief vector via Equation 2.21.

As noted previously, if the elements of the λ vector are equivalent and the π vector elements sum to one, then the belief vector is equivalent to the π vector. Thus, the new belief vector is [.442 .558].

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent node.

The message stimulating the node to update originated from its parent node. By the rules of propagating information through the network, no λ message vector is required. As the Tank Tracks node is an evidence node, no child messages are required either.

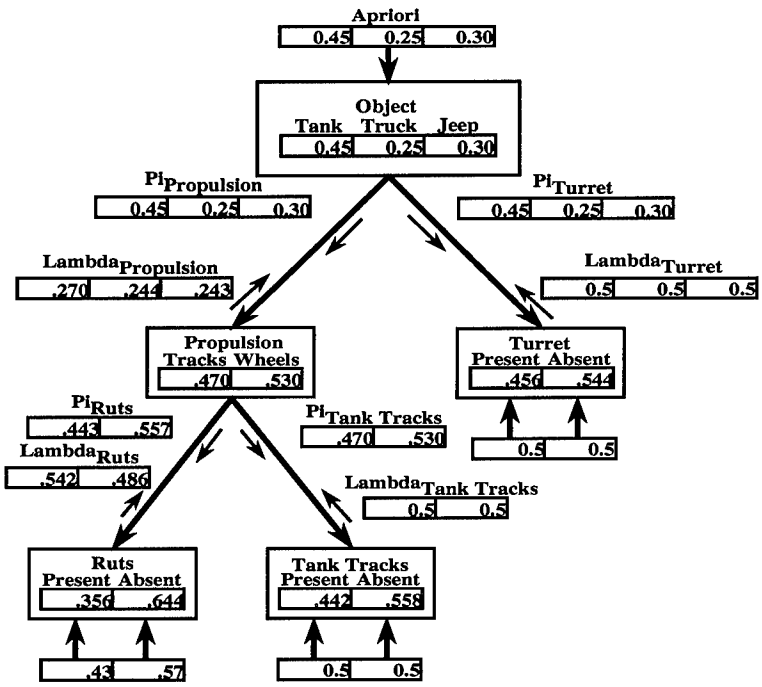


Figure A.11 Update of the Tank Tracks Node

Figure A.11 illustrates the update to the Tank Tracks node. A quick comparison with Figure A.10 shows the belief in the presence of tank tracks increases slightly. Thus, the weak support for the presence of ruts actually strengthens the network's belief in the presence of competing evidence.

The update of the Tank Tracks node does not produce any message vectors and does not dirty any nodes. However, the Object node still requires updating. The network follows the required six steps.

Step 1: Inspect all messages received from the parent and child nodes.

The π message vector for the Object node is the a priori vector. There are two λ message vectors, $\lambda_{Propulsion}$ and λ_{Turret}.

Step 2: Calculate the new λ vector via Equation 2.22

The new λ vector becomes [.135 .122 .1215].

Step 3: Calculate the new π vector via Equation 2.23

For root nodes, the π vector is just the π message vector.

Step 4: Calculate the new belief vector via Equation 2.21.

The new belief vector is [.479 .239 .282]

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent node.

No λ message vectors are required for root nodes.

Step 6: Calculate a new π message vector for each child node via Equation 2.25. Transmit these to the appropriate child nodes.

The Propulsion node sent the message that stimulated the Object node to update. Thus it expects no π message vector in return. A π message vector is required for the Turret node. The Object node receives only two λ message vectors and one of these has equal elements. Thus, by an earlier result, the π_{Turret} message vector is equal to the belief vector or [.479 .239 .282].

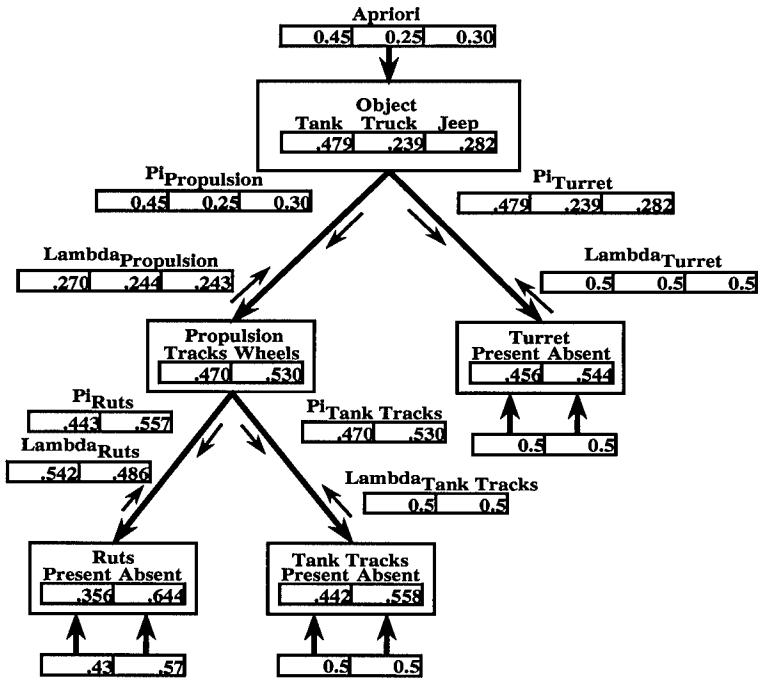


Figure A.12 Update of the Object Node

Figure A.12 illustrates the update to the Object node. Comparing Figure A.11 shows that the weak evidence for the presence of Ruts manifests itself as a slight increase to the belief the object is a tank. This increased belief comes at the expense of the beliefs in wheeled vehicles. Thus the network, combining a priori information that anticipates tanks as the most prevalent class and weak diagnostic evidence for wheeled vehicles, makes a preliminary assertion that the target is a tank. With so little information collected, however, this assertion is, as indicated by a belief value less than .5, still very tentative and uncertain.

The propagation of information throughout the network is not yet complete. A π message vector from the Object node dirties the Turret node and invokes an update. The node runs quickly through the six update steps.

Step 1: Inspect all messages received from the parent and child nodes.

The Pi_{Turret} field of Figure A.12 provides the π message vector from the parent node. The λ message vector is merely the input to the node.

Step 2: Calculate the new λ vector via Equation 2.22

The λ vector for an evidence node is the input vector. Thus the λ vector is [.5 .5].

Step 3: Calculate the new π vector via Equation 2.23

The required conditional probabilities reside in Table 2.2. With a new π message vector from the parent, the π vector becomes [.482 .518].

Step 4: Calculate the new belief vector via Equation 2.21.

The elements of the λ vector are equivalent and the π vector elements sum to one. Thus, the belief vector is equivalent to the π vector.

Step 5: Calculate a new λ message vector via Equation 2.24. Transmit it to the parent node.

The message stimulating the node to update originated from its parent node. By the rules of propagating information through the network, no λ message vector is required. As the Turret node is an evidence node, no child messages are required either.

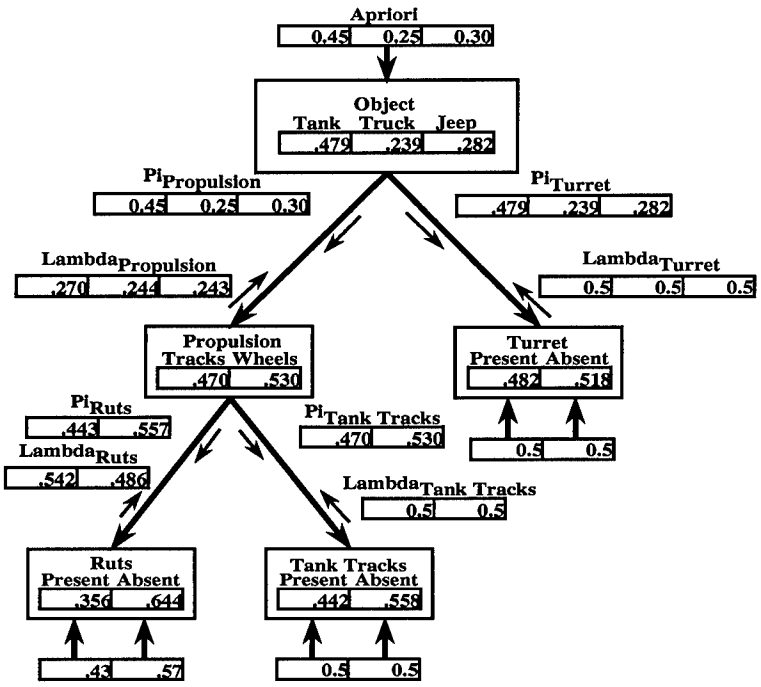


Figure A.13 Update of the Turret Node

Figure A.13 illustrates the update to the Turret node. A quick comparison with Figure A.12 shows the increased anticipation in tanks increases the belief in turret.

The update to the Turret node generated no λ or π messages. As the network has updated every node, the propagation of information linked to the introduction of evidence at the Ruts node ends here. The network in Figure A.13 is now at equilibrium and will not change until new evidence is introduced.

The preceding network updates caused by the presentation of evidence to the Ruts node proceeded in a very painstaking and step by step fashion. This approach showed the internal workings of the network as it updated. Hopefully, this knowledge facilitates a better understanding of how Bayesian networks operate and what the values associated with the network mean. The network of Figure A.13 still anticipates further evidence which it shall receive shortly. These next paragraphs do not illustrate the node updates in the same detail. The nodes employ the same six steps for updating, but the text presents and discusses only the resulting new values.

Figure A.14 shows new evidence presented to the Tank Tracks node. The new input vector is [.91 .09]

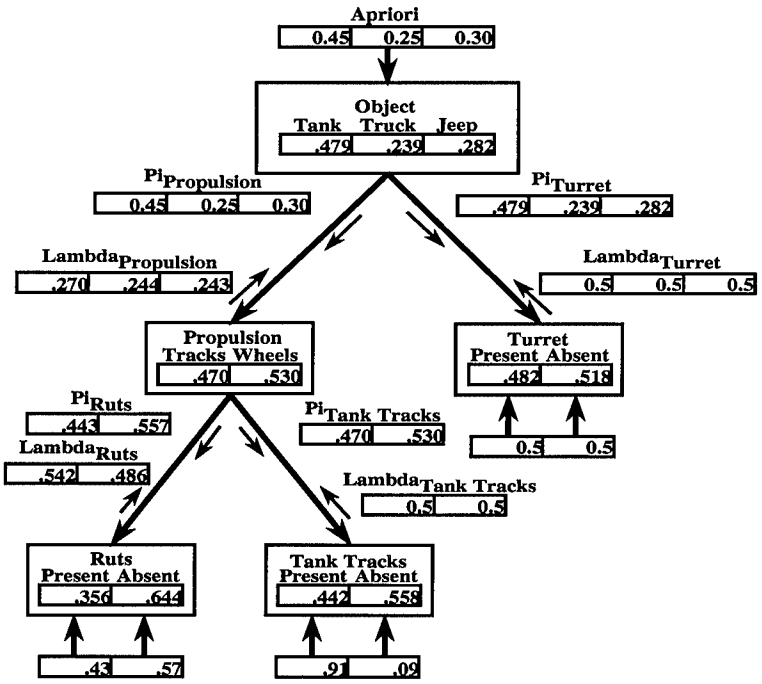


Figure A.14 New Evidence Introduced to the Tank Tracks Node

The new evidence stimulates the Tank Tracks node to update. The update calculates a new belief vector and generates a λ message vector for the Propulsion node. The new belief vector becomes [.889 .111] and the new $\lambda_{TankTracks}$ message vector is [.861 .091]. The diagnostic support for tank tracks is strong but is tempered slightly by weak anticipatory support. Strong evidential support for tank tracks results in strong diagnostic support for tank tracks in the λ message vector to the propulsion node. Figure A.15 displays the results of the Tank Tracks node update.

The $\lambda_{TankTracks}$ message vector stimulates the Propulsion node to update. The update produces a new belief vector, a new $\lambda_{Propulsion}$ message vector to the Object node, and a π_{Ruts} message vector to the Ruts node. The new belief vector is [.894 .106]. The strong diagnostic support for tank tracks, the weak diagnostic support for ruts, and the anticipatory support for tanks combine in the Propulsion node to produce a strong belief that tracks provide the object's

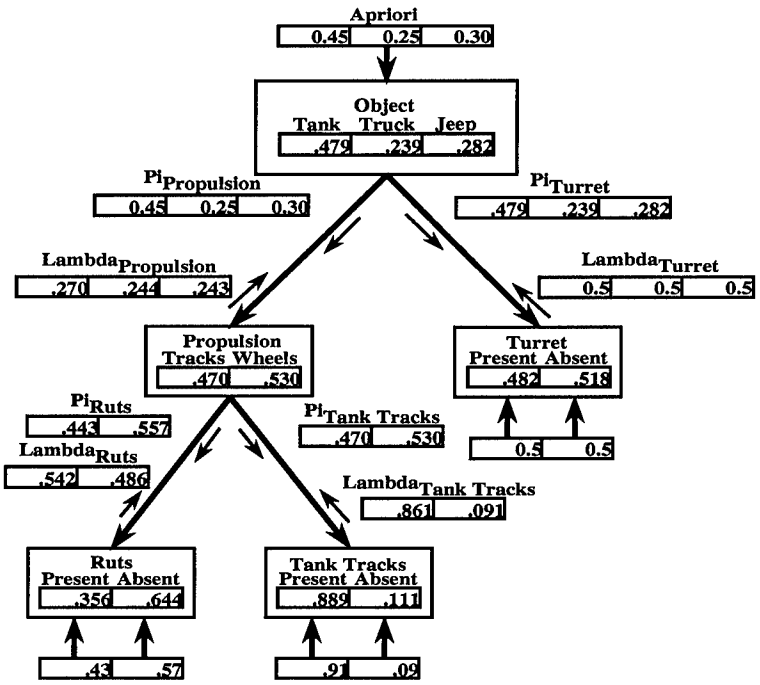


Figure A.15 Update to the Tank Tracks Node

propulsion. The new $\lambda_{Propulsion}$ message vector is [.446 .065 .048]. The strong diagnostic support for tracks produces a substantial increase in diagnostic support for tanks.

Because the elements of neither λ message vector are equal, the π_{Ruts} message vector is not equal to the belief vector as it was in previous updates. The node must calculate the π_{Ruts} message vector using Equation 2.25. The new π_{Ruts} message vector is [.883 .117]. The strong diagnostic support for tank tracks produces weak anticipatory support for the presence of ruts. Figure A.16 displays the results of the Propulsion node update.

The vector messages produced by the update to the Propulsion node stimulate the Ruts node and the Object node to update. The Ruts node update is handled first. Updating this node produces a new belief vector. The update generates no new message vectors due to the rules for propagating information through the network. The new belief vector is [.198 .802]. The weak anticipatory support for ruts coupled with the weak diagnostic evidence combines to decrease the belief in ruts to its lowest level.

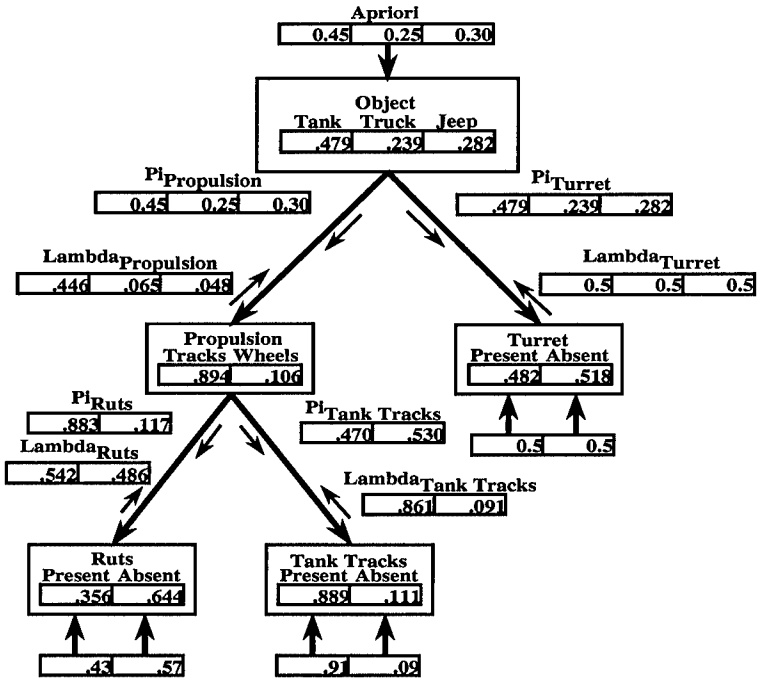


Figure A.16 Update to the Propulsion Node

The network next updates the Object node. The update to this node produces a new belief vector and a new π_{Turret} message vector. The new belief vector is [.868 .07 .062]. The strength of the strong evidential support for tank tracks, which the network converts into strong diagnostic support for tracked vehicles and then strong diagnostic support for tanks, produces a dramatic increase in the belief the object is a tank. The anticipatory support in terms of the a priori values further strengthens this belief.

The strong diagnostic support for tanks combines with anticipatory support to substantially strengthen the tank entry in the anticipatory support sent to Turret. Because one of the Object node's λ messages has equal elements, the new π_{Turret} message vector is equal to the belief vector. Figure A.17 displays the results of the updates to the nodes Ruts and Objects.

The π_{Turret} message vector stimulates the Turret node to update. The update produces a new belief vector but no message vectors. Thus, the propagation of information resulting from new evidence presented at the Tank Tracks node ends here. The new belief vector for the Turret node is [.861 .131]. The strong anticipatory support stemming from the strong

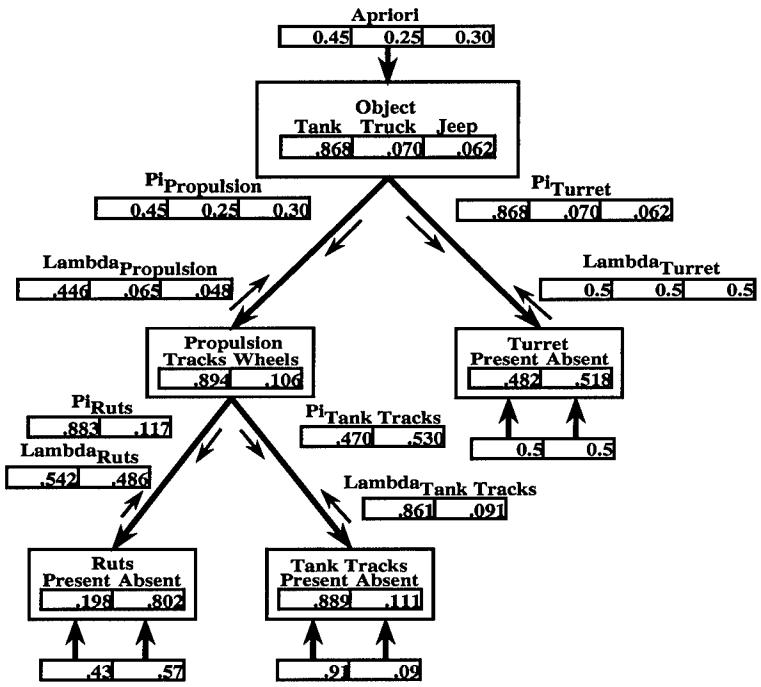


Figure A.17 Update to the Nodes Ruts and Object

diagnostic support for tank pushes the belief that a turret is present up dramatically. With this final update, the network is again at equilibrium. Figure A.18 displays this steady state.

With strong diagnostic support for tank tracks, the hypothesis Tank emerges highly favored. The large belief value for the Tank hypothesis indicates current evidence strongly supports the conclusion the object under consideration is a tank. To continue, the network receives the last piece of evidence. This evidence is an input vector of [.77 .23] received at the Turret node. Figure A.19 shows the evidence presented to the Turret node.

The presented evidence is in support of a turret. Combined with already strong anticipatory support, it further increases the associated belief value in the Turret node and produces strong diagnostic support for the tank hypothesis in the Object node. The Object node converts the diagnostic support to anticipated support and sends it via π message vectors to the Propulsion, Ruts, and Tank Tracks nodes. The result increases the belief vector elements for tracks and the presence of tank tracks, while decreasing the belief in the presence of ruts. Thus all the nodes in the network are effected by this new evidence. As the steps involved in propa-

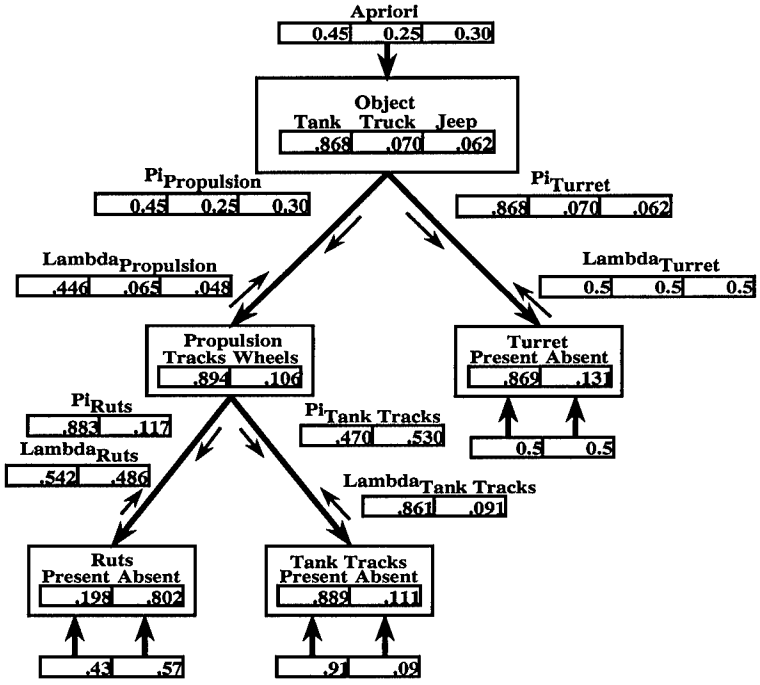


Figure A.18 The Bayesian Network in a Steady State

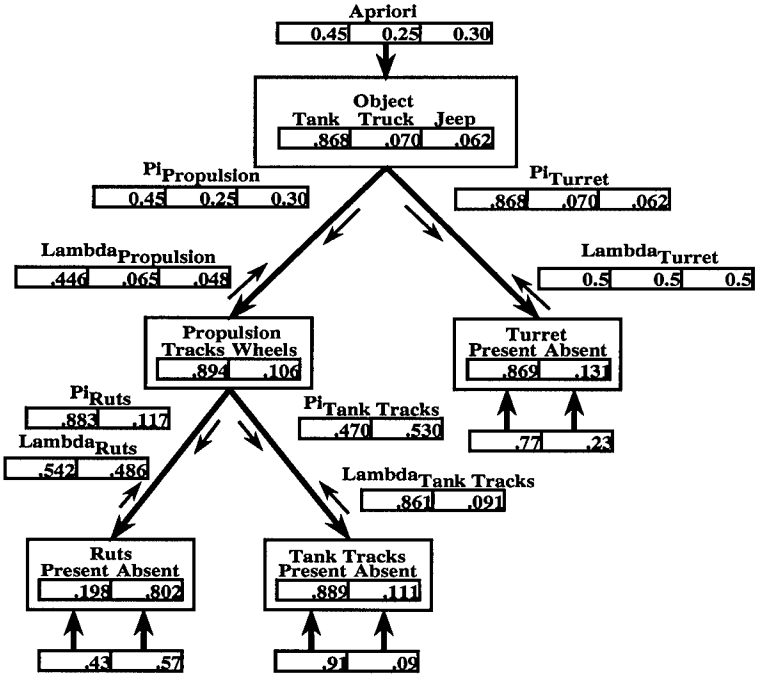


Figure A.19 Evidence Presented to Turret Node

gating the new information through the network have appeared previously, the information is allowed to propagate and all nodes are updated accordingly. Figure A.20 shows the network in a steady state after the new information has propagated through. A quick comparison with Figure A.19 shows how the new information changes the belief vectors.

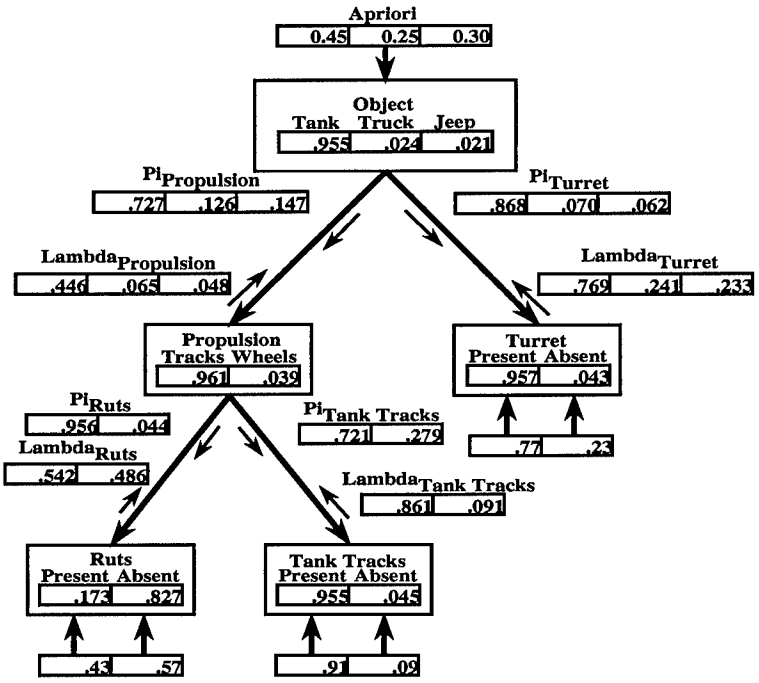


Figure A.20 Bayesian Network in a Steady State After Receiving All the Evidence

With all evidence received, the network clearly comes to the conclusion that the object under consideration is a tank. Note that the belief vectors of all the evidence nodes currently contain belief vector element values closer to the extremes of zero and one then the associated input vectors. This is due to anticipatory support influencing the belief in what the network expects to encounter. Support at the evidence nodes for hypotheses that complement one another, for example hypotheses Turret Present, Tank Tracks Present, and Ruts Absent, produce anticipatory support that further leverage the beliefs that this evidence exists. In this regard, Bayesian networks see what they believe.

The evidence presented to the network of Figure A.20 foreshadows the network's tank conclusion. The evidence vectors clearly supported the tank hypothesis. Figure A.21 and

Figure A.22 show the same Bayesian network presented with different evidence vectors. The networks are shown in their steady state after all information has propagated through. For these examples, the evidence nodes receive the evidence simultaneously. The method for calculating vectors when the network receives all the evidence in parallel is as follows. First, calculate all the λ message vectors starting at the evidence nodes and proceeding level by level to the root node. Second, calculate the belief vector at the root node. Third, calculate all the π vector messages starting at the root node and proceeding level by level to the evidence nodes. Finally, calculate the belief vectors for all nodes other than the root.

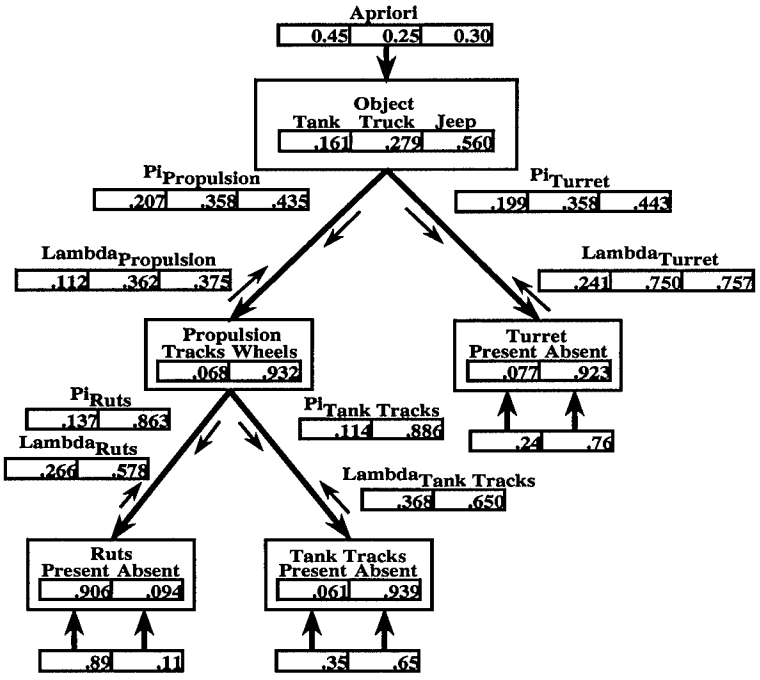


Figure A.21 Bayesian Network After Receiving Evidence Favoring a Jeep Conclusion

The evidence vectors for Figure A.21 support the hypotheses for wheeled vehicles. The evidence results in strong diagnostic support for the conclusion that the target is a wheeled vehicle. This strong diagnostic support is complementary. Combining it with the information stored in the conditional probabilities serves to leverage the belief that the observations represented by the evidence vectors are true. This leverage manifests itself in belief vectors at the evidence nodes that are higher in value than the actual evidence vectors. In other words,

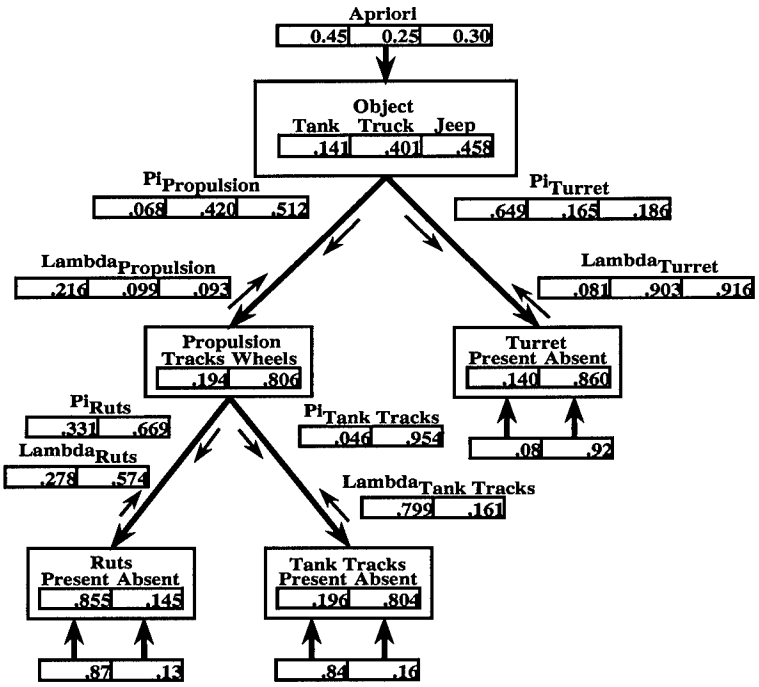


Figure A.22 Bayesian Network After Receiving Contradictory Evidence

the diagnostic support for the lack of a turret reinforces the belief that ruts are present and tank tracks are absent. The diagnostic support for ruts flows upward in the network where it combines with a lack of diagnostic support for tank tracks to produce a conclusion at the Propulsion node that the object is a wheeled vehicle. This conclusion is further strengthened by anticipatory support stemming from the lack of evidence in a turret. The wheel conclusion sends diagnostic support for wheeled vehicles to the Object. It combines with the lack of diagnostic support for turret and the a priori values to produce a conclusion that the object is a jeep.

Actually, the evidence involved serves only to support a wheeled vehicle hypothesis over the tracked vehicle hypothesis. The a priori values and small differences in the conditional probabilities caused the network to favor the Jeep hypothesis over the Truck hypothesis. This observation demonstrates a small flaw in this example. The evidence provided to the network contains no information to distinguish between a Truck hypothesis and a Jeep hypothesis. The network really just distinguishes between tracked and wheeled vehicles. The network

must rely wholly on the conditional probability matrices and the a priori values to distinguish between Truck and Jeep hypotheses. As these values are mostly static, the network would rarely if ever select the Truck hypothesis over the Jeep. To alleviate this flaw, a designer could add another evidence node receiving evidence for a feature that distinguishes between trucks and jeeps. Examples include a node receiving evidence on the size of the object or a node receiving evidence for the presence of a truck bed. The current lack of ability to distinguish between Truck and Jeep objects is why the belief in the Jeep hypothesis is relatively low. As the evidence presented supports both the Jeep and Truck hypotheses, the network makes its decision only with reservations. Essentially the network makes an educated guess.

For the next example, the presented evidence is ambiguous. The evidence vector for tank tracks conflicts with the values of the other two vectors. In this case, the network can produce no clear winner and even the most favored hypotheses have belief values that indicate the network is more inclined to believe the object is none of the above. The network does support the wheeled vehicle hypotheses over the tracked vehicle hypothesis. This is due to the strong dependence of the Tank hypothesis to the presence of a turret. Evidence to the contrary results in a low belief in the Tank hypothesis. The high conditional probability linking the Tank hypothesis to the presence of a turret has a strong effect throughout the network. As an example, the strong lack of evidence in a turret results in strong anticipatory support for the absence of tank tracks. This support results in elements of the Tank Tracks belief vector approaching the opposite of the evidence vectors. Essentially, the network suspects the feature extractor responsible for supplying the tank tracks evidence of lying.

A.2 Final Remarks

This Bayesian network tutorial attempted to demystify Bayesian networks as applied to classification problems. It detailed the internal operations required to calculate the belief vectors at each node and attempted to explain the semantic meaning of the numbers. With this knowledge in hand, Chapter III investigates characteristics of Bayesian networks designed for classification. The primary characteristic addressed is how the belief vector changes based on

changes to the parameters of the network. Chapter III also discusses the issue of where the conditional probability tables arise from. This topic is further examined in Chapter IV.

Appendix B. Matlab Script Files

This appendix holds all the headers for all the Matlab script files referenced in this work. Each header thoroughly describes the operation of each script file. They include detailing each program's expected inputs and outputs. The programs themselves are available on electronic media from the Air Force Institute of Technology, Department of Engineering, 2950 P St, Wright-Patterson AFB, OH 45433.

Each section serves to partition the script files into related sets. Section B.4 contains all the script files necessary to implement the Bayesian Network Classifier Algorithm for classification problems discriminating between two classes using one or two features.

B.1 Rank 2 Bayesian Network

Matlab script files associated with Rank 2 Bayesian Networks

```
%************************************************************************
% FUNCTION make2D.m
%************************************************************************
% Purpose: Creates the 2-D matrix necessary to show the decision regions
% of a rank 2 bayesnet with two inputs in two space.
%
% Called by: movie2D
%
% Calls: regions2D(confuse, ap, n) or
% spregions2D(confuse, ap, n)
%
% Quirks: None
%
% Usage: make2D(matrix, vector, positive int, positive int);
%
% Example: make2D(confuse, ap, n, fig);
%************************************************************************

function make2D(confuse, ap, n, fig)

% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
```

```
%
% n: Number of points to evaluate per axis
%
% fig: Figure handle of where to draw the image
%************************************************************************

%************************************************************************
% FUNCTION netbelief2D.m
%************************************************************************
% Purpose: Simulates the operation of a rank 2, 2 input bayesnet and
% displays a 3-D matrix representing the belief values based on
% input evidence.
%
% Called by: None
%
% Quirks: None
%
% Usage: netbelief2D(matrix, array, positive int, positive int, positive int)
%
% Example: netbelief2D(confuse, ap, n, fig)
%************************************************************************

function netbelief(confuse, ap, n, fig, c)

%
% confuse: The confusion matrix
%
% ap: The list of a priori values
%
% fig: The figure window for drawing the resulting 3-D graph
%
% n: Number of points to evaluate per axis
%************************************************************************

%************************************************************************
% FUNCTION movie2D.m
%************************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% decision regions of a rank 2 bayesnet with two inputs change
% over the changes in the conditional probability matrices.
%
% Called by: None
```

```
%
% Calls: make2D(confuse, ap, n, fig)
%
% Quirks: None
%
% Usage: [matrixm] = movie2D(int, int, matrix, vector, int, int);
%
% Example: [M] = movie2D(i, j, confuse, ap, n, frames);
%
% frames must be odd!
%*************************************************************************

function [M] = movie2D(i, j, confuse, ap, n, frames)

% M: The array holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
%
% i,j: Index into the confusion matrix.This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%*************************************************************************

%*************************************************************************
% FUNCTION belmovie2D.m
%*************************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% belief vector of a rank 2 bayesnet with two inputs change over
% the changes in the conditional probability matrices.
%
% Called by: None
%
% Calls: netbelief2D(confuse, ap, n, fig)
```

```
%
% Quirks: None
%
% Usage: [matrixm] = belmovie2D(int, int, matrix, vector, int, int);
%
% Example: [M] = belmovie2D(i, j, confuse, ap, n, frames);
%*********************************************************************

function [M] = belmovie2D(i, j, confuse, ap, n, frames)

% M: The array holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
%
% i,j:Index into the confusion matrix.This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%*********************************************************************

%*********************************************************************
% FUNCTION movieab2D.m
%*********************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% decision regions of a rank 2 bayesnet with two inputs change over
% the changes in the conditional probability matrices. This script
% file differs from movie2D.m in that it step confuse(i,j) from 1
% to confuse(j,i) and then steps confuse(j,i) to one thus ranging
% the movie among the extremes.
%
% Called by: None
%
% Calls: make2D(confuse, ap, n, fig)
%
```

```
% Quirks: frames must be odd!
%
% Usage: [matrixm] = movieab2D(int, int, matrix, vector, int, int);
%
% Example: [M] = movieab2D(i, j, confuse, ap, n, frames);
%*********************************************************************

function [M] = movieab2D(i, j, confuse, ap, n, frames)

% M: The array holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
%
% i,j:Index into the confusion matrix.This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%*********************************************************************

%*********************************************************************
% FUNCTION regions2D.m
%*********************************************************************
% Purpose: Simulates the operation of a rank 2, 2 input regions2D and
% returns a 3-D matrix representing the subsequent decision
% regions.
%
% Called by: None
%
% Quirks: None
%
% Usage: [matrixx matrixy matrixc] = regions2D(matrix, vector, int);
%
% Example: [X Y C] = regions2D(confuse, ap, n);
%*********************************************************************
```

```
function [X, Y, C] = regions2D(confuse, ap, n)

% X: Indexing on the x axis
% Y: Indexing on the y axis
% C: Mapping of color codes indicating decision regions
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%************************************************************************

%************************************************************************
% FUNCTION spregions2D.m
%************************************************************************
% Purpose: Simulates the operation of a rank 2, 2 input bayesnet.
% Returns 3 matrices, X and Y values representing evidence
% levels on each input and a color-coded decision region matrix.
% This function makes use of assumptions about the parameters of
% the equations to simplify the calculations as much as possible
%
% Called by: make2D.m
%
% Calls: None
%
% Quirks: None
%
% Usage: [matrixx matrixy matrixc] = spregions3D(matrix, vector,int);
%
% Example: [X Y C] = spregions3D(confuse, ap, n);
%************************************************************************

function [X, Y, C] = spregions2D(confuse, ap, n)

% X: Indexing on the x axis
% Y: Indexing on the y axis
% C: Mapping of color codes indicating decision regions
%
% confuse: Confusion matrix
%
```

```
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%**************************************************************************
```

B.2 Rank 3 Bayesian Network

Matlab script files associated with Rank 3 Bayesian Networks

```
%**************************************************************************
% FUNCTION make3D.m
%**************************************************************************
% Purpose: Creates the 3-D matrix necessary to show the decision regions
% of a rank 3 bayesnet with three inputs in three space.
%
% Called by: movie3D, movie3D
%
% Calls: regions3D(zaxis, z, confuse, ap, n) or
% spregions3D(z, confuse, ap, n)
%
% Quirks: None
%
% Usage: make3D(matrix, vector, positive int, positive int);
%
% Example: make3D(confuse, ap, n, fig);
%**************************************************************************

function make3D(confuse, ap, n, fig)

% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% fig: Figure handle of where to draw the image
%**************************************************************************

%**************************************************************************
% FUNCTION slice3D.m
%**************************************************************************
```

```
% Purpose: Creates the 2-D matrix showing the decision regions of a
% rank 3 bayesnet with three inputs at specified evidence values.
%
% Called by: None
%
% Calls: regions3D(zaxis, zaxisval, confuse, ap, n);
%
% Quirks: None
%
% Usage: slice3D('char',real, matrix, vector, int, int)
% where 'char' can be one of 'x','y','z'.
%
% Example: slice3D('z', zaxisval, confuse, ap, n, fig);
%************************************************************************

function slice3D(zaxis, zaxisval, confuse, ap, n, fig)

% zaxis: The evidence (represented as either w,x,y, or z) assigned
% to the z axis.
%
% zaxisval: The z axis value currently being evaluated.
%
% confuse: The confusion matrix.
%
% ap: The a priori vector
%
% n: Number of points to evaluate on an axis. The total number
% of points will be (n + 1)^2.
%
% fig: Figure number to print to.
%************************************************************************

%************************************************************************
% FUNCTION netbelief3D.m
%************************************************************************
% Purpose: Simulates the operation of a rank 3, 3 input bayesnet and
% displays a 3-D matrix representing the belief values based
% on input evidence.
%
% Called by: None
%
% Quirks: None
```

```
%
% Usage: netbelief3D(real, matrix, array, positive int, positive int)
%
% Example: netbelief3D(z, confuse, ap, n, fig)
%**************************************************************************

function netbelief3D(z, confuse, ap, n, fig)

%
% confuse: The confusion matrix
%
% ap: The list of a priori values
%
% fig: The figure window for drawing the resulting 3-D graph
%
% n: Number of points to evaluate per axis
%
% z: Value of z at which to evaluate the belief values
%**************************************************************************

%**************************************************************************
% FUNCTION movie3D.m
%**************************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% decision regions of a rank 3 bayesnet with three inputs change
% over the changes in the conditional probability matrices.
%
% Called by: None
%
% Calls: make3D(confuse, ap, n, fig)
%
% Quirks: None
%
% Usage: [matrixm matrixmbs] = movie3D(int, int, matrix, vector, int, int);
%
% Example: [M MBS] = movie3D(i, j, confuse, ap, n, frames);
%
% frames must be odd!
%**************************************************************************

function [M, MBS] = movie3D(i, j, confuse, ap, n, frames)
```

```
% M: The array holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
%
% MBS: The array holding frames of the movie. This array will
% specifically hold backside views of the 3-D decision space.
%
% i,j: Index into the confusion matrix.This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%************************************************************************

%************************************************************************
% FUNCTION belmovie3D.m
%************************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% belief vector of a rank 3 bayesnet with three inputs change over
% the changes in the conditional probability matrices.
%
% Called by: None
%
% Calls: netbelief3D(confuse, ap, n, fig)
%
% Quirks: None
%
% Usage: [matrixm] = belmovie3D(int, int, matrix, vector, int, int);
%
% Example: [M] = belmovie3D(i, j, confuse, ap, n, frames);
%************************************************************************

function [M] = belmovie3D(i, j, confuse, ap, n, frames)

% M: The array holding frames of the movie. This array will
```

```
% specifically hold forward views of the 3-D decision space.
%
% i,j:Index into the confusion matrix. This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%**************************************************************************

%**************************************************************************
% FUNCTION movieab3D.m
%**************************************************************************
% Purpose: Creates the matrix necessary to show a movie on how the
% decision regions of a rank 3 bayesnet with 3 inputs change over
% the changes in the conditional probability matrices. This script
% file differs from movie3D.m in that it step confuse(i,j) from 1
% to confuse(j,i) and then steps confuse(j,i) to one thus ranging
% the movie among the extremes.
%
% Called by: None
%
% Calls: make3D(confuse, ap, n, fig)
%
% Quirks: frames must be odd!
%
% Usage: [matrix, matrix] = movieab3D(int, int, matrix, vector, int, int);
%
% Example: [M,MBS] = movieab3D(i, j, confuse, ap, n, frames);
%**************************************************************************

function [M, MBS] = movieab2D(i, j, confuse, ap, n, frames)

% M: The matrix holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
```

```
%
% MBS: Matrix holding the back side of the movie
%
% i,j:Index into the confusion matrix. This will be the value that will vary
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%*********************************************************************

%*********************************************************************
% FUNCTION regions3D.m
%*********************************************************************
% Purpose: Simulates the operation of a rank 3, 3 input bayesnet.
% Returns 4 matrices, X Y and Z representing evidence levels
% on each input and a color-coded decision region matrix.
%
% Called by: make3D.m
%
% Calls: None
%
% Quirks: None
%
% Usage: [matrixx matrixy matrixz matrixc] =
% regions3D(char, real, matrix, vector,int);
%
% Example: [X Y Z C] = regions3D(zaxis, zval, confuse, ap, n);
%*********************************************************************

function [I, J, K, C] = regions3D(zaxis, zval, confuse, ap, n)

% X: Indexing on the x axis
% Y: Indexing on the y axis
% Z: Indexing on the z axis
```

```
% C: Mapping of color codes indicating decision regions
%
% zaxis: Specifies which axis to hold constant.
%
% zval: Z axis value on which to evaluate the function.
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%************************************************************************

%************************************************************************
% FUNCTION spregions3D.m
%************************************************************************
% Purpose: Simulates the operation of a rank 3, 3 input bayesnet.
% Returns 4 matrices, X Y and Z values representing evidence
% levels on each input and a color-coded decision region matrix.
% This function makes use of assumptions about the parameters of
% the equations to simplify the calculations as much as possible.
%
% Called by: make3D.m
%
% Calls: None
%
% Quirks: None
%
% Usage: [matrixx matrixy matrixz matrixc] =
% spregions3D(real, matrix, vector,int);
%
% Example: [X Y Z C] = spregions3D(z, confuse, ap, n);
%************************************************************************

function [X, Y, Z, C] = spregions3D(z, confuse, ap, n)

% X: Indexing on the x axis
% Y: Indexing on the y axis
% Z: Indexing on the z axis
% C: Mapping of color codes indicating decision regions
%
% z: Z axis value on which to evaluate the function.
```

```
%
% confuse: Confusion matrix
%
% ap: Vector containing a priori pi values
%
% n: Number of points to evaluate per axis
%************************************************************************
```

B.3 Rank 4 Bayesian Network

Matlab script files associated with Rank 4 Bayesian Networks

```
%************************************************************************
% FUNCTION make4D.m
%************************************************************************
% Purpose: Creates the 3-D matrix necessary to show the decision regions
% of a rank 4 bayesnet with four inputs in three space.
%
% Called by: movie4D
%
% Calls: regions4D(zaxis, time, timeval, zaxisval, confuse, n)
%
% Quirks: None
%
% Usage: make4D('char','char', real, matrix, int)
% where 'char' can be one of 'w','x','y','z'.
% The characters input for time and zaxis must differ.
%
% Example: make4D('y','z', timeval, confuse, ap, n);
%************************************************************************

function make4D(zaxis, time, timeval ,confuse, ap ,n)

% time: The evidence (represented as either w,x,y or z) that
% will be varied over time
%
% zaxis: The evidence (represented as either w,x,y, or z) assigned
% to the z axis.
%
% timeval: The time axis value currently being evaluated.
%
% confuse: The confusion matrix
```

```
%
% n: Number of points to evaluate on an axis. The total number
% of points will be (n + 1)^3.
%*************************************************************************

%*************************************************************************
% FUNCTION slice4D.m
%*************************************************************************
% Purpose: Creates the 2-D matrix showing the decision regions of a rank
% 4 bayesnet with four inputs at specified evidence values.
%
% Called by: None
%
% Calls: regions4D(zaxis, time, zaxisval, timeval, confuse, n);
%
% Quirks: None
%
% Usage: slice4D('char','char', real, real, matrix, int)
% where 'char' can be one of 'w','x','y','z'.
% The characters input for time and zaxis must differ.
%
% Example: slice4D('y','z', zaxisval, timeval, confuse, ap, n);
%*************************************************************************

function slice4D(zaxis, time, zaxisval, timeval ,confuse, ap, n)

% time: The evidence (represented as either w,x,y or z) that
% will be varied over time
%
% zaxis: The evidence (represented as either w,x,y, or z) assigned
% to the z axis.
%
% timeval: The time axis value currently being evaluated.
%
% zaxisval: The z axis value currently being evaluated.
%
% confuse: The confusion matrix.
%
% ap: The a priori vector
%
% n: Number of points to evaluate on an axis. The total number
% of points will be (n + 1)^2.
```

```
%*************************************************************************

%*************************************************************************
% FUNCTION movie4D.m
%*************************************************************************
% Purpose: Creates matrices necessary to show movies on how the
% decision regions of a rank 4 bayesnet with four inputs change
% over a change in the evidence of the fourth hypothesis. Two
% movies are captured. The first shows the front surfaces of a
% 3-D representation of the decision regions. The second
% captures the back surfaces.
%
% Called by: None
%
% Calls: make4D(zaxis, time, timeval, confuse, n)
%
% Quirks: None
%
% Usage: [array1 array2] = movie4D('char', 'char', matrix, int, int)
% where 'char' can be one of 'w','x','y','z'.
% The characters input for time and zaxis must differ.
%
% Example: movie4D('y','z', confuse, n, frames);
%*************************************************************************

function [M, MBS] = movie4D(zaxis, time, confuse, n, frames)

% M: The array holding frames of the movie. This array will
% specifically hold forward views of the 3-D decision space.
%
% MBS: The array holding frames of the movie. This array will
% specifically hold backside views of the 3-D decision space.
%
% time: The evidence (represented as either w,x,y or z) that
% will be varied over time.
%
% w maps to evidence in Hypothesis 1
% x maps to evidence in Hypothesis 2
% y maps to evidence in Hypothesis 3
% z maps to evidence in Hypothesis 4
%
% zaxis: The evidence (represented as either w,x,y, or z) assigned
```

```
% to the z axis.
%
% confuse: The confusion matrix.
%
% n: Number of points to evaluate on an axis. The total number
% of points will be (n + 1)^3.
%
% frames: Number of frames to shoot for the movie. This also
% determines the number of points we plot on the time axis.
%
% The movies can be viewed by entering "movie(X)" at the matlab prompt.
% X represents an array returned by this function.
%*************************************************************************

%*************************************************************************
% FUNCTION regions4D.m
%*************************************************************************
% Purpose: Simulates the operation of a rank 4, 4 input bayesnet.
% Returns 4 matrices of X Y and Z values representing evidence
% levels on each input. Creates and returns a color-coded
% decision region matrix. This function makes use of assumptions
% about the parameters of the equations to simplify the
% calculations as much as possible (version 3).
%
% Called by: make4D
%
% Calls: None
%
% Quirks: The characters input for time and zaxis must differ.
%
% Usage: [arrayx,arrayy,arrayz,arrayc] =
% regions4D('char','char', real, real, matrix, vector,int)
% where 'char' can be one of 'w','x','y','z'.
% % Example: [X Y Z C] = regions4D('y','z', zaxisval, timeval, con, ap, n);
%*************************************************************************

function [I, J, K, C] = regions4D(zaxis,time,zaxisval,timeval,con, ap, n)

% I: Points on the x axis.
% J: Points on the y axis.
% K: Points on the z axis.
% C: Matrix of color values.
```

```
%
% time: The evidence (represented as either w,x,y or z) that
% will be varied over time
%
% zaxis: The evidence (represented as either w,x,y, or z) assigned
% to the z axis.
%
% timeval: The time axis value currently being evaluated.
%
% zaxisval: The z axis value currently being evaluated.
%
% con: The confusion matrix.
%
% ap: The a priori vector
%
% n: Number of points to evaluate on an axis. The total number
% of points will be (n + 1)^2.
%*************************************************************************
```

B.4 Bayesian Network Classifier Algorithm

Matlab script files necessary to implement the Bayesian Network Classifier Algorithm

```
%*************************************************************************
% FUNCTION classify2D.m
%*************************************************************************
% Purpose: Receives two vectors of data, computes the mean and standard
% deviation of the data. Assumes a gaussian distribution and plots
% plots the distributions of the data based on the computed means
% and standard deviations. Receives two limits on decision
% boundaries. Plots these relative to the data distributions.
% Program then calculates the probabilities that the data of the
% two classes falls on either side of the decision boundaries and
% uses this information to populate a confusion matrix. Normalizes
% the data to their respective probability curves. Applies the
% confusion matrix and normalized data to calculate the belief
% values of a Bayesian network. Based on the belief values
% determines the number of data points misclassified.
%
% Called by: None
%
% Quirks: None
```

```
%
% Usage: classify2D(array, array, array, array, real, real,vector)
%
% Example: classify2D(Data1m, Data2m, Data1, Data2, bound1, bound2,ap)
%*********************************************************************

function classify2D(Data1m, Data2m, Data1, Data2, bound1, bound2, ap)

% Data1m: Data set to train the classifier, first feature.
%
% Data2m: Data set to train the classifier, second feature.
%
% Data1: Test Data set for first feature.
%
% Data2: Test Data set for second feature.
%
% bound1: Rightmost boundary
%
% bound2: Leftmost boundary
%
% ap: a priori vector
%*********************************************************************

%*********************************************************************
% FUNCTION classify4D.m
%*********************************************************************
% Purpose: Receives two sets of two vectors of data representing features of
% two classes, computes the mean and standard deviation of all the
% data vectors. Assumes a gaussian distribution and plots the
% distributions of the data based on the computed means and standard
% deviations. Receives four limits on decision boundaries. Plots
% these relative to the data distributions. Program then calculates
% the probabilities that the data of the two classes falls on either
% side of the decision boundaries and uses this information to
% populate a confusion matrix. Normalizes the data to their
% respective probability curves. Applies the confusion matrix and
% normalized data to calculate the belief values of a Bayesian
% network. Based on the belief values determines the number of data
% points misclassified.
%
%
% Called by: None
```

```
%
% Quirks: None
%
% Usage: [vector] = classify4D(matrix, matrix, matrix, matrix, vector,vector)
%
% Example: [r] = classify4D(Data1tr, Data1test, Data2tr, Data2test ,bounds, ap)
%*********************************************************************

function [r] = classify4D(Data1tr, Data1test, Data2tr, Data2test, bounds, ap)

% r: Vector for communicating the results of the classification effort
%
% Data1tr: Data set for class one. Consists of an array of two element
% vectors. This array holds the training data for class one.
%
% Data1test: Data set for class one. Consists of an array of two element
% vectors. This array holds the test data. All vectors in this array
% should classify to class 1.
%
% Data2tr: Data set for class two. Consists of an array of two element
% vectors. This array holds the training data for class two.
%
% Data2test: Data set for class two. Consists of an array of two element
% vectors. This array holds the test data. All vectors in this array
% should classify to class 2.
%
% bounds: vector of boundaries required to establish the conditional
% probabilities of the confusion matrix.
%
% ap: a priori vector
%*********************************************************************

%*********************************************************************
% FUNCTION gauss.m
%*********************************************************************
% Purpose: Accepts a mean and a standard deviation and returns a array of
% numbers approximating a gaussian curve.
%
% Called by: None
%
% Quirks: None
%
```

```
% Usage: [array, array] = gauss(real, real, positive int, positive int)
%
% Example: [x,y] = gauss(mu, sigma, n, r)
%************************************************************************

function [X,Y] = gauss(mu, sigma, n, r)

% X: Indexing array. Returned so user can graph the result
%
% Y: Name of the array to hold the result.
%
% mu: The mean of the distribution.
%
% sigma: The variance of the distribution.
%
% n: Number of points to evaluate.
%
% r: Range of points to evaluate
%************************************************************************

%************************************************************************
% FUNCTION cdfgauss.m
%************************************************************************
% Purpose: Accepts a mean, variance and value x and returns the probability
% distribution function of a normal distribution at that point.
%
% Called by: None
%
% Quirks: None
%
% Usage: real = cdfgauss(real, real, real, real)
%
% Example: t = cdfgauss(mu, sigma, x, tol)
%************************************************************************

function [y] = cdfgauss(mu, sigma, x, tol)

% mu: The mean of the distribution.
%
% sigma: The variance of the distribution.
%
% x: Point to evaluate
```

```
%
% tol: point to quit
%*************************************************************************

%*************************************************************************
% FUNCTION pgauss.m
%*************************************************************************
% Purpose: Accepts a mean, standard deviation and vector x and returns the
% probability density function of a normal distribution at that point.
%
% Called by: None
%
% Quirks: None
%
% Usage: real = pgauss(real, real, vector)
%
% Example: t = gauss(mu, sigma, x)
%*************************************************************************

function [y] = pgauss(mu, sigma, x)

%
% mu: The mean of the distribution.
%
% sigma: The standard deviation of the distribution.
%
% x: Set of points to evaluate
%*************************************************************************

%*************************************************************************
% FUNCTION ngauss.m
%*************************************************************************
% Purpose: Accepts a mean, standard deviation and vector x and returns the
% normalized probability density function of a normal distribution at
% the points in the vector.
%
% Called by: None
%
% Quirks: None
%
% Usage: real = ngauss(real, real, vector)
%
```

```
% Example: t = ngauss(mu, sigma, x)
%**************************************************************************

function [y] = ngauss(mu, sigma, x)

% mu: The mean of the distribution.
%
% sigma: The standard deviation of the distribution.
%
% x: Set of points to evaluate
%**************************************************************************

%**************************************************************************
% FUNCTION ldf.m
%**************************************************************************
% Purpose: Accepts a covariance matrix, a mean vector, and an input and
% calculates the value of the linear discriminate function.
%
% Called by: bayeserror2D
%
% Quirks: None
%
% Usage: real = ldf(matrix, vector, vector)
%
% Example: t = ldf(sigma1, mu1, X)
%**************************************************************************

function [y] = ldf(sigma1, mu1, X)

% sigma1: Covariance matrix.
%
% mu1: Mean vector.
%
% X: Feature vector.
%**************************************************************************

%**************************************************************************
% FUNCTION bayeserror.m
%**************************************************************************
% Purpose: Accepts a mean and standard deviation of two distributions and
% and computes the points at which they cross.
%
```

```
% Called by: None
%
% Quirks: None
%
% Usage: real = bayeserror(real, real, real, real)
%
% Example: t = bayeserror(mu1, sigma1, mu2, sigma2)
%************************************************************************

function [y] = bayeserror(mu1, sigma1, mu2, sigma2)

% mu1: The mean of the first distribution.
%
% sigma1: The standard deviation of the first distribution.
%
% mu2: The mean of the second distribution.
%
% sigma2: The standard deviation of the second distribution.
%************************************************************************

%************************************************************************
% FUNCTION bayeserror2D.m
%************************************************************************
% Purpose: Accepts means and standard deviations for a two class, two
% feature problem and estimates the bayes error rate.
%
% Called by: None
%
% Quirks: None
%
% Usage: vector = bayeserror2D(vector, matrix, vector, matrix, matrix, matrix)
%
% Example: t = bayeserror2D(mu1, sigma1, mu2, sigma2, data1, data2)
%************************************************************************

function [y] = bayeserror2D(mu1, sigma1, mu2, sigma2, data1, data2)

% mu1: Mean vector for class 1.
%
% sigma1: Covariance matrix for class 1.
%
% mu2: Mean vector for class 2.
```

```
%
% sigma2: Covariance matrix for class 2.
%
% data1: Data points for class 1.
%
% data2: Data points for class 2.
%************************************************************************
```

B.5 Miscellaneous

Miscellaneous Matlab Script Files

```
%************************************************************************
% FUNCTION apriori.m
%************************************************************************
% Purpose: Returns an array of a priori values for use with the bayesian
% network simulation programs
% Called by: None
%
% Quirks: None
%
% Usage: array = apriori(int, positive int)
%
% Example: ap = apriori(option, size)
%
% Notes: Typing a positive value in option will return an array of 'size'
% elements, each equal in value and the sum of the entries equal to
% one. Typing a negative value for option returns an array of 'size'
% random numbers between zero and one
%************************************************************************

function [C] = apriori(offset, s)

% C: Name of the array to hold the result. This is normally
% considered an array of a priori values to the root node of a
% Bayesian network
%
% option: A positive value will result in equal value entries summing to
% one. A negative value results in an array of random numbers
% between zero and one
%
% s: Size of the array desired
```

```
%
% If offset is less than zero, populate a array of length 'size' with
% random numbers.
%*************************************************************************

%*************************************************************************
% FUNCTION makematrix.m
%*************************************************************************
% Purpose: Makes a confusion matrix for use with the bayesian network
% simulation programs
% Called by: None
%
% Usage: matrix = makematrix(real, positive int)
%
% Example: confuse = makematrix(offset, size)
%
% Notes: Typing a real value between 0 and 1 will cause the off-diagonal
% elements of the resulting matrix to be set to that value.
% Typing a -1 will set all off-diagonal matrix values to random
% values. All on-diagonal entries will be 1.
%*************************************************************************

function [C] = makematrix(offset, size)

% C: Name of the matrix to hold the result. This is normally
% considered a confusion matrix
%
% offset: Value of the off-diagonal matrix entries. Setting offset to
% -1 will populate these entries with random numbers.
%
% size: Size of the matrix desired
%*************************************************************************
```

Appendix C. Datasets

This appendix list the mammography data used in the real world classifier example in Chapter IV. Table C.1 and Table C.2 lists the data used to train the Bayesian network classifier for benign and malignant tumors respectively. Table C.3 and Table C.4 lists the test data used to test the classifier for benign and malignant tumors respectively.

Table C.1 Malignant Training Data for Mammography Example

$(f_1, Malignant)$	$(f_2, Malignant)$
1.7624961e+02	4.2944116e+01
1.1888495e+02	5.7715399e+01
2.3357889e+02	1.6636353e+02
9.5713551e+01	5.6172877e+01
1.9541159e+02	2.1482684e+02
1.7296469e+02	6.2885669e+01
1.7550827e+02	1.3228989e+02
2.6579533e+02	2.5742566e+02
1.6965656e+02	1.4265584e+02
4.2442964e+02	1.4105638e+02
2.9164415e+02	4.6981340e+01
1.1118129e+02	8.1909374e+01
1.1956997e+02	7.3467235e+01
1.1230076e+02	6.4492521e+01
7.4219298e+01	1.5866503e+02

Table C.2 Benign Training Data for Mammography Example

$(f_1, Benign)$	$(f_2, Benign)$
1.2013797e+02	6.9345202e+01
1.0431807e+02	7.4468617e+01
1.1497022e+02	1.3905650e+02
2.8420768e+02	1.3673853e+02
1.1009324e+02	9.7938660e+01
1.5103374e+02	3.5661412e+02
8.0789721e+01	5.1558006e+01
1.4175762e+02	9.6351045e+01
9.0571981e+01	1.8157858e+02
1.0612803e+02	8.7320156e+01
1.9376015e+02	2.8316206e+02
2.7142025e+02	1.9202644e+02
1.5248323e+02	1.3793253e+02
1.2574700e+02	1.4194384e+02
1.7275039e+02	1.9778419e+02
1.3066745e+02	9.8642897e+01
1.1114740e+02	9.1636569e+01
1.2354682e+02	6.2556057e+01
9.9036420e+01	1.6641247e+02
1.3229269e+02	1.3532825e+02
1.2353619e+02	2.8565141e+02
8.0801851e+01	4.5494448e+01
1.2189089e+02	1.9832251e+02
1.8295637e+02	3.2185236e+02
3.2418483e+02	5.2584625e+02
4.1285233e+02	4.8001244e+02
8.2579812e+01	3.8464836e+01
8.7356814e+01	1.8685933e+01
1.1615421e+02	1.2796722e+02
1.6120116e+02	1.6923314e+02
1.3792958e+02	2.0848039e+02

Table C.3 Malignant Test Data for Mammography Example

$(f_1, Malignant)$	$(f_2, Malignant)$
1.5237856e+02	3.6369017e+01
1.1776806e+02	8.7190073e+01
1.5204244e+02	2.3946198e+02
1.5564867e+02	1.9525022e+02
1.5124102e+02	1.5645095e+02
2.5594100e+02	2.4681651e+02
3.0647640e+02	1.1830565e+02
1.2925181e+02	8.4745706e+01
2.0324425e+02	1.3544319e+02
1.1123064e+02	1.5458157e+01
9.5067179e+01	5.9777877e+01
7.2844785e+01	8.1226507e+01
1.7494260e+02	8.8714377e+01
1.1417605e+02	5.3784090e+01
2.4613896e+02	3.0760256e+01
1.5789300e+02	9.1189128e+01

Table C.4 Benign Test Data for Mammography Example

$(f_1, Benign)$	$(f_2, Benign)$
2.9174000e+02	3.6529498e+02
2.3257728e+02	2.2862542e+02
1.3166690e+02	1.7496502e+02
1.1870359e+02	5.0969301e+02
7.2711338e+01	4.1927394e+01
2.4733973e+02	2.1747379e+02
2.4011101e+02	3.1001527e+02
1.0131698e+02	3.5360643e+02
8.0917077e+01	8.2396424e+01
1.1952520e+02	4.0331121e+02
8.9373017e+01	2.2756904e+02
9.4563000e+01	7.2846709e+01
8.1769251e+01	1.1037617e+02
1.0929160e+02	1.1818777e+02
1.2173062e+02	5.9074956e+01
8.5181315e+01	6.0314814e+01
7.9911788e+01	8.7386034e+01
1.5263840e+02	2.7236306e+02
1.5008203e+02	2.2485094e+02
8.6458238e+01	6.4825922e+01
7.3566532e+01	2.5186009e+02
1.0194168e+02	1.7141348e+02
1.0646932e+02	1.7167583e+02
1.1613752e+02	5.6050556e+01
1.1011331e+02	1.6470250e+02
1.3198001e+02	1.3586039e+02
1.1673852e+02	3.0970826e+01
2.1462981e+02	1.0695719e+02
1.1981655e+02	1.7695010e+02
1.1636293e+02	2.1113255e+02
1.6943333e+02	1.2804013e+02
1.4428670e+02	4.0775495e+02

Bibliography

1. Cherniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, *12*(4):50 – 63 (Winter 1991).
2. Cooper, G.F. *Probabilistic Inference Using Belief Networks is NP-Hard.* Technical Report KSL-87-27, Stanford University: Medical Computer Science Group, 1987.
3. Dagum, Paul and Michael Luby. "Approximating probabilistic inference in Bayesian belief networks is NP-hard (Research Note)," *Artificial Intelligence*, *60*:141 – 153 (1993).
4. Dennis W. Ruck, Capt, USAF, "Personal Interview," May 1994. Assistant Professor, Air Force Institute of Technology, Wright-Patterson AFB OH.
5. Giarratano, Joseph and Gary Riley. *Expert Systems: Principles and Programming.* PWS Publishing Company, 1994.
6. Kocur, Katherine. *Computer-Aided Breast Cancer Diagnosis.* MS thesis, School of Engineering, Air Force Institute of Technology, 2950 P St, Wright-Patterson AFB, OH 45433, December 1994.
7. Kocur, Katherine and et. al. "Computer-Aided Breast Cancer Diagnosis." Submitted to IEEE Engineering in Medicine and Biology, November 1994.
8. Levitt, Tod S., Thomas O. Binford and Gil J. Ettinger. "Intelligent Probabilistic Inference." *Uncertainty in Artificial Intelligence* edited by L. N. Kanal R. D. Shachter, T. S. Levitt and J. F. Lemmer, North-Holland, 1990.
9. Musman, S. A., L. W. Chang and L. B. Booker. "A Real Time Control Strategy for Bayesian Belief Networks with Application to Ship Classification Problem Solving." *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence.* 738 – 744. New York, NY: IEEE, 1990.
10. Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann, 1988.
11. Pearl, Judea. "Fusion, Propagation, and Structuring in Belief Networks," *Artificial Intelligence*, *29*:241 – 288 (1990).
12. Raney, Steven, et al. "ARAGTAP ATR System Overview." Tech report limited to Government and authorized Government contractors only. General publication forthcoming, 1993.
13. Rimey, Raymond D. and Christopher M. Brown. "Task Specific Utility in a General Bayes Net Vision System." *IEEE Computer Vision and Pattern Recognition.* 142–147. New York, NY: IEEE, 1992.
14. Sarkar, Sudeep and Kim L. Boyer. "Perceptual Organization Using Bayesian Networks." *IEEE Computer Vision and Pattern Recognition.* 251–256. New York, NY: IEEE, 1992.
15. Shachter, Ross D. "Intelligent Probabilistic Inference." *Uncertainty in Artificial Intelligence* edited by L. N. Kanal and J. F. Lemmer, North-Holland, 1986.

Vita

Captain Greg Ahlquist [redacted]. He graduated valedictorian in 1984 from Malad High School in Malad, Idaho. Captain Ahlquist attended the University of Utah in Salt Lake City under an AFROTC scholarship and graduated from that institution in March of 1989 with a Bachelor of Science degree in Electrical Engineering. Upon graduation, he accepted a commission in the United States Air Force and reported to his first duty station at Rome Laboratory, previously the Rome Air Development Center, at Griffiss AFB, NY in June of 1989. From that time until April of 1993, Captain Ahlquist worked in Rome Laboratory's Intelligence and Reconnaissance and Plans and Programs Directorates. He entered the Air Force Institute of Technology in May 1993 seeking a Master of Science degree in Computer Engineering. Capt Ahlquist is a member of the Tau Beta Pi engineering honor society.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE
An Analysis of Bayesian Networks as Classifiers

5. FUNDING NUMBERS

6. AUTHOR(S)
Gregory Clyde Ahlquist, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCE/ENG/94D-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr Martin Justice
Wright Laboratory Target Recognition Branch, WL/AARA
Bldg 23, 2010 5th St
Wright-Patterson AFB, OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An analysis of Bayesian networks as classifiers is presented. This analysis results in an algorithm and several tools related to Bayesian network classifiers. The tools calculate and display the decision regions for two level Bayesian network classifiers. They collectively provide an approach to analyze the effects of changing network parameters on the network's decision regions. The algorithm defines a Bayesian network classifier to solve traditional classification problems. The algorithm is data driven, meaning that the resulting Bayesian network classifier is uniquely tuned to the classification problem at hand. Also, the algorithm contains procedures for defining the topology of a Bayesian network classifier and for precisely deriving the required conditional probabilities. A brief tutorial on Bayesian networks is also presented.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Bayesian Networks, Classifiers, Classification	195
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

★U.S.GPO:1990-0-273-271