Bayesian Networks for Cardiovascular Monitoring

by

Jennifer Roberts
B. S., Electrical Engineering
B. S., Computer Science
University of Maryland, College Park (2004)

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2006

Author .....
Department of Electrical Engineering
and Computer Science
May 12, 2006

Certified by.....
George Verghese
Professor of Electrical Engineering
Thesis Supervisor

Certified by.....
Thomas Heldt
Postdoctoral Research Associate
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Bayesian Networks for Cardiovascular Monitoring

by

Jennifer Roberts

Submitted to the Department of Electrical Engineering
and Computer Science
on May 12, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

In the Intensive Care Unit, physicians have access to many types of information when treating patients. Physicians attempt to consider as much of the relevant information as possible, but the astronomically large amounts of data collected make it impossible to consider all available information within a reasonable amount of time. In this thesis, I explore Bayesian Networks as a way to integrate patient data into a probabilistic model. I present a small Bayesian Network model of the cardiovascular system and analyze the network's ability to estimate unknown patient parameters using available patient information. I test the network's estimation capabilities using both simulated and real patient data, and I discuss ways to exploit the network's ability to adapt to patient data and learn relationships between patient variables.

Thesis Supervisor: George Verghese
Title: Professor of Electrical Engineering

Thesis Supervisor: Thomas Heldt
Title: Postdoctoral Research Associate

Acknowledgments

I would like to thank Dr. Thomas Heldt for his support, encouragement, and guidance. He has been a wonderful mentor, and his continual feedback has helped me throughout my time at MIT. I also thank Professor George Verghese for his gentle support and thoughtful guidance. I appreciate the way he always looks out for his students. I thank Tushar Parlikar for supplying me with simulated and real patient data and for contributing a section to the thesis. Without his help, this work would not have been possible. I also thank my parents, Tom and Valerie; my sister, Becca; my grandmother, Dolores; and my boyfriend, Chen, for their continued love and support. I graciously thank the Fannie and John Hertz Foundation and the National Science Foundation for their financial support. This work is made possible by grant RO1 EB001659 from the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bayesian Networks provide a flexible way of incorporating different types of information into a single probabilistic model. In a medical setting, one can use these networks to create a patient model that incorporates lab test results, clinician observations, vital signs, and other forms of qualitative and quantitative patient data. This thesis will outline a mathematical framework for incorporating Bayesian Networks into cardiovascular models for the Intensive Care Unit and explore effective ways to assimilate new patient information into the Bayesian models.

1.1 Using Models to Integrate Patient Data

Physicians have access to many types of information when treating patients. For example, they can examine real-time waveform data like blood pressures and electrocardiogram (ECG) recordings; data trends like time-averaged heart rate; intermittent measurements like body temperature and lab results; and qualitative observations like reported dizziness and nausea.

Within the Intensive Care Unit (ICU), physicians attempt to consider as much of the relevant information as possible, but the astronomically large amounts of data collected make it impossible to consider all available information within a reasonable amount of time. In addition, not all of the data collected is helpful in its raw form, but sufficient statistics taken from such data might help physicians gain a more

thorough understanding of recent changes in the patient's state. Because of this, we are exploring ways to integrate different types of patient data into more synthesized forms (see http://mimic.mit.edu/).

Models provide one way of synthesizing multiple observations of the same complex system, and Bayesian Networks provide a probabilistic framework for developing patient models. With Bayesian Networks, we can incorporate different data types into a single model and use Bayesian inference to probabilistically estimate variables of interest, even when data is missing.

By using models to synthesize data, we hope to improve monitoring capabilities and move patient monitoring in new directions by generating more accurate alarms, making predictive diagnoses, providing clinical hypotheses, predicting treatment outcomes, and helping doctors make more informed clinical decisions. As an example, alarms generated by current monitoring systems have extremely high false alarm rates, and because they go off so frequently, nurses learn to ignore them. Often these alarms are generated by observing only a single patient variable at a time. By synthesizing more of the available patient data, systems could generate more accurate alarms that would convey more reliable information.

Similarly, current monitoring systems measure and display patient vital signs, but they only perform a limited amount of analysis on the measured waveforms. By using models to synthesize patient data, monitoring systems might one day perform analyses that could diagnose problems before they become critical, perhaps predicting a patient's impending hemodynamic crisis minutes in advance so that doctors can perform preventive treatments instead of reactive ones. Advanced systems might further use patient models to simulate treatment options and use the simulation results to guide doctors toward more effective treatments. Systems might also learn to identify the true source of a medical problem so that doctors could address problem sources instead of treating symptoms. This discussion touches on just a few of the ways that a more synthesized approach might aid both clinicians and patients.

1.2 Traditional Cardiovascular Models

As mentioned before, models provide one way of synthesizing different observations of the same complex system. Many traditional cardiovascular models are derived from physiology by using computational models of the heart and blood vessels. Such computational models provide a set of differential equations that can be mapped to an equivalent set of circuit equations. Within the equivalent circuit models, voltage represents pressure, current represents blood flow, resistance represents blood vessel resistance, and capacitance represents compliance, a measure of vascular distensibility. These circuit models are used to simulate cardiovascular dynamics and gain a thorough understanding of how components within the cardiovascular system interact.

A number of cardiovascular circuit models exist with varying levels of complexity. The Windkessel model, for example, describes the arterial portion of the circulatory system with only two parameters. In contrast, other models like those described in [4, 6, 14] contain many vascular compartments representing the right and left ventricles, the systemic veins and arteries, the pulmonary veins and arteries, and sometimes even other organs in the body. Some models simulate pulsatile behavior [2, 4], while others track trend behavior [12]. Timothy Davis's thesis contains a short survey of several of the early lumped-parameter hemodynamic models including CVSIM, CVSAD, and the Windkessel model [3].

While these physiological models are complex enough to track real patient data, using these models to do so in real time is extremely challenging. Within the ICU, only a limited number of patient vital signs are consistently available. These signals do not provide enough information to accurately estimate more than a small number of model parameters [14]. This means that many parameters in the larger physiological models cannot be estimated accurately using commonly available patient data. In addition, real data is often extremely noisy. Signals frequently disappear or become overwhelmed by artifacts when patients move or become excited. These artifacts make it even more difficult to robustly track patient trends using physiological models.

In addition, conventional physiological models do not use all available information when attempting to track patient state. They do not incorporate qualitative observations like "the patient is dizzy" or "the patient is bleeding heavily." They also fail to incorporate information about the patient's diagnosis. Similarly, physiological models fail to assimilate intermittent data like that obtained through lab reports, nurses notes, and medication charts, into their calculations. These types of information convey important information about the patient's state and should be taken into account.

In order to integrate both qualitative and quantitative information into modeling efforts, we use Bayesian Network models to complement the traditional physiological ones. With Bayesian Networks, we can incorporate non-numeric, discrete, and continuous information into the same model. Bayesian Networks can robustly use intermittent data when it is available and function reliably when such data is not available. These networks provide a stochastic framework for estimating physiological model parameters, a framework we can rely on when patient measurements become noisy or unreliable.

1.3 Bayesian Networks in Medicine

Within the medical field, Bayesian networks have frequently been used for diagnosis [8, 16, 11, 10], patient monitoring [1], and therapy planning [13]. For instance, the Heart Disease Program [8] uses a several-hundred-node Bayesian Network to create a list of most probable cardiovascular diagnoses, given patient symptoms. While accurate, the network is difficult to maintain because of the large number of network parameters set by expert opinion.

VentPlan [13], a ventilator therapy planner, attempted to use a Bayesian Network to incorporate qualitative data into a mathematical model of the pulmonary system. By exploiting the Bayesian Network, the program used qualitative patient information to initialize parameters of the mathematical model. The mathematical model then ran simulations to determine the best ventilator treatment plan, given the current

patient state. Overall, the system proposed treatment plans that changed ventilator settings in the correct direction, but experts disagreed about the proper magnitude of change needed.

Using Bayesian Networks, Berzuini et al. [1] propose a methodology that uses information about previous patients to monitor current patients. Within this methodology, each patient is described using the same parameterized model. Each set of patient parameters is assumed to be drawn from the same unknown probability distribution. The current patient receives an additional parameter that rates how typical his or her response is in comparison with the general population. If the current patient's response seems typical, information about previous patient parameters is used to set the current patient's parameters. Otherwise, current patient parameters are set using only information about the current patient. The relative weight assigned to information from previous patients versus information from the current patient changes based on the current patient's typicality. With a Bayesian Network, Berzuini et al. explicitly described how current patient parameters depend information about both current and previous patients. They then used this methodology to model how patients' white blood counts respond to chemotherapy. By conditioning on information obtained from previous patients, these researchers more accurately tracked white blood count for patients deemed typical of the general population.

Recently, much work has been done on sequentially learning Bayesian Network structures from data [5]. Algorithms that perform this type of learning are often used in biological contexts to identify large biological networks, but have also been used in other contexts to learn models directly from data. The book by Neapolitan contains a summary of some of these applications and research within the field of Learning Bayesian Networks [10].

In this thesis, I focus on developing relatively simple Bayesian Network models of the cardiovascular system that capture necessary patient dynamics without adding extraneous model parameters that are difficult to estimate. Unlike the Heart Disease Program, I explore much simpler models whose parameters can be learned and updated based on available patient data. The current goal is not to diagnose every

type of cardiovascular disease, but instead to estimate unavailable information about a patient's state based on data generally available in the ICU. We aim to develop a Bayesian model for the cardiovascular system similar to the Bayesian model used by VentPlan, while exploring ways to balance population data with current patient history, as in [1].

1.4 Goals

In this thesis, I will describe a methodology for using Bayesian Networks for cardiovascular monitoring in the Intensive Care Unit. Specifically, I will

- describe a mathematical framework for incorporating Bayesian Network models into cardiovascular patient monitoring systems for the ICU;
- describe a methodology for assimilating new patient information into Bayesian models without disregarding or overemphasizing previously acquired information;
- describe simulation results that explore whether
  - Bayesian Networks can provide good estimates of hidden variables;
  - Bayesian Networks can learn and track changes in patient state;
- describe future research directions based on simulation results.

1.5 Outline

In Chapter 2, I give an overview of Bayesian Networks and important terminology. Within this chapter, I introduce a simple Bayesian Network model of the cardiovascular system, a model later used to predict unknown patient parameters using available patient data. In Chapter 3, I discuss how to model the network probability distribution parameters, and I describe various ways of using patient data to set and update these parameters. I then apply the Bayesian Network model to both simulated and

real patient data in Chapter 4. In this chapter, I analyze the network's ability to use Bayesian inference to predict unknown patient parameters, given generally available patient information. Chapter 5 concludes with a discussion of the network's capabilities and a discussion of future research directions.

Chapter 2

Bayesian Networks

This chapter provides an overview of Bayesian Networks. It describes how these networks are derived and the types of mathematical issues one needs to consider when creating and implementing a Bayesian Network. In this chapter, I introduce a small Bayesian Network model of the cardiovascular system that I will explore more deeply later in the thesis.

2.1 What is a Bayesian Network?

Before introducing Bayesian Networks, I would like to briefly discuss networks in a more general context. A network, or graph, consists of a set of nodes and a set of edges which connect the nodes. The edges can be either directed or undirected, depending on whether they point from one node to the other or simply indicate a link between nodes.

In a directed graph, like the one shown in Figure 2-1, the edges are represented by arrows. The triangular end of each arrow is called the head, while the other end is called the tail. In Figure 2-1, nodes X and W are connected by a directed edge, with node X at the edge's tail and node W at its head. When two nodes are connected by a directed edge, the one at the tail is called the parent, while the one at the head is called the child. Node X, then, has two children, W and Y.

Paths through the graphs are formed by following arrows, tail to head, from one

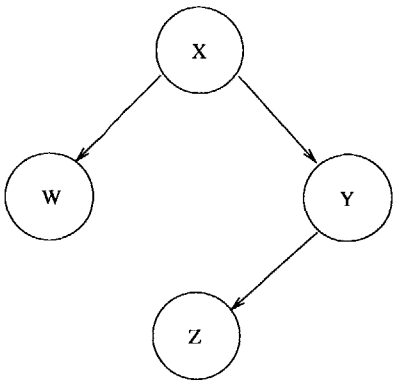


Figure 2-1: A directed acyclic graph.

node to another. Since arrows can be followed, tail to head, from node X to node Z, a path exists from X to Z. Since arrows can not be followed in this manner from node W to node Z, no path exists from W to Z. If a path exists from one node to another, the latter node is called a descendant of the former. Thus, W, Y, and Z are descendants of X, but X, Y, and Z are not descendants of W. Because of this, X, Y, and Z are known as non-descendants of W.

When a path exists from a node back to itself, that path is called a cycle. The graph in Figure 2-1 contains no cycles, but if an arrow was added from Z to X, it would contain a cycle. A directed graph containing no cycles is called a directed acyclic graph.

Networks are used to represent many different types of relationships. In computer networks, the nodes may represent computers while the edges represent wires between the computers. Within a cellular network, nodes represent cell phones or cell towers and directed edges represent down-links or up-links between nodes. In social networks, nodes signify people, while edges signify interactions or relationships between those people.

In a Bayesian Network, nodes represent random variables, and edges represent conditional dependencies between those variables. In this manner, a Bayesian Network uses a directed acyclic graph to represent a joint probability distribution. More specifically, a particular graph structure represents a *class* of probability distributions that factors in a certain way. These factorizations in turn imply a certain set of

conditional independencies. The Markov Property, a property shared by all Bayesian Networks, summarizes the most fundamental set of these implied conditional independencies. This property states that child nodes are conditionally independent of their non-descendants, given their parents. If a node does not have parents, the node is simply independent of its non-descendants.

Due to the Markov Property, a Bayesian Network's joint probability distribution relates to its graph structure in a straightforward manner. If the network uses N nodes to represent the random variables $\{x_1, \ldots, x_n\}$, the joint probability distribution of $\{x_1, \ldots, x_n\}$ can be expressed as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | \text{the parents of } x_i).$$

As an example, the *class* of Markov Chains all factor as

$$P(x_1, \ldots, x_n) = P(x_1) \cdot \prod_{i=2}^{n} P(x_i | x_{i-1}),$$

and all exhibit the same set of conditional independencies, namely that the past, x_{i-1}, is conditionally independent of the future, x_{i+1}, given the present, x_i. Thus, all Markov chains with the same number of random variables have a Bayesian network with the same graphical form.

Figure 2-2 displays a simple Bayesian Network, its conditional probability distributions, and an equation for its joint probability distribution. Due to the Markov Property, this network structure implies that heart rate (HR) is independent of net volume inflow (NVI) because heart rate has no parents, and net volume inflow is not a descendant of heart rate.

The full set of conditional independencies implied by the graph structure can be derived from the set delineated by the Markov Property [10]. When the graph structure does not imply that a conditional independence exists within a set of nodes, that set of nodes often exhibits a conditional dependence. Implied conditional dependencies of this sort, however, do not exist within every probability distribution

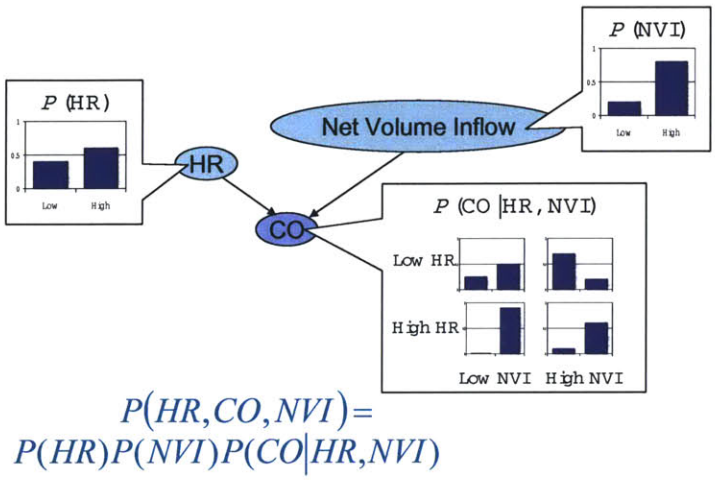


Figure 2-2: A Bayesian Network example, where HR denotes heart rate, CO denotes cardiac output, and NVI denotes net volume inflow.

that factors in a manner consistent with the graph structure. To see this, suppose that in Figure 2-2 the top row of the conditional distribution for cardiac output (CO) equaled the bottom row, so CO no longer depended on HR. Then the distribution for CO would be the same regardless of whether HR was low or high, and the conditional dependence implied by the arrow from HR to CO would not be present within the underlying probability distribution.

A Bayesian Network is called well-posed if and only if all conditional dependencies implied by the network structure are present in the probability distribution. In other words, a well-posed probability distribution contains only those conditional independencies implied by the network structure and no additional independencies not implied by the structure. When the underlying distribution is not well-posed, it exhibits conditional independencies not reflected in the network structure. In this case, one can always find a different network structure that would better describe the underlying distribution. In the example in the previous paragraph, we could obtain a well-posed network structure by removing the arrow between HR and CO. Unless otherwise specified, all Bayesian Networks discussed in this thesis are well-posed.

Bayesian Networks are often used to make inferences about unknown information,

given known observations. This involves calculating the *a posteriori* distribution of the unknown variable, given all available information. The *a posteriori* distribution can then be used to calculate a best estimate of the unknown variable, given the known data. Because the joint probability distribution is represented as a Bayesian Network, algorithms can efficiently calculate the *a posteriori* distribution by exploiting the network structure [17, 10].

2.2 Important Terminology

A Bayesian Network model has three interrelated parts: the network structure, the form of the probability distribution, and the distribution parameters. The network structure, i.e., the directed acyclic graph structure, defines a set of conditional independencies that the underlying distribution must exhibit, but the form of the distribution is otherwise unconstrained. The structure specifies only the manner in which the distribution must factor.

Given a network structure, each node can be assigned a different type of discrete or continuous distribution, conditional on the values of its parents. The form of these conditional node distributions determines the form of the underlying joint probability distribution. For example, if the node variables are jointly Gaussian, the conditional distributions will all be Gaussian.

The distribution form further constrains the underlying distribution defined by the network structure, but one must also set the conditional distribution parameters to have a fully-defined model. These network parameters form the final part of the Bayesian Network model. In the jointly Gaussian case, the parameter set consists of the conditional means and variances for each network node. Much of the thesis is devoted to setting and updating these network parameters.

To create a Bayesian Network, one can deduce a probabilistic model from prior experience, learn model structure and distributions from real data, or use some combination of these two approaches. In our treatment of Bayesian Networks, we assume that an appropriate network structure and an appropriate probability distribution

form have been derived by experts. This network structure and distribution form remain constant throughout our simulations. We then explore ways of setting and updating the parameters based on prior experience and patient history. Specifically, we are interested in the following issues:

- Parameter Initialization – Initially setting the distribution parameters based on both expert opinion and general population data by performing some combination of
  - Pre-Training Initialization – Setting the distribution parameters based on previous expectations or expert opinions.
  - Batch Learning – Learning distribution parameters from previously acquired patient data sets.
- Inference – Using both the initialized network and available patient information to estimate the values of unknown variables.
- Sequential Parameter Learning – Incrementally updating the distribution parameters based on new information about the patient.

Many Bayesian Network models are initialized using a set of training data and then employed to make inferences about new data sets, without any type of feedback mechanism to update the probability distributions based on incoming data. In the ICU setting this type of approach seems flawed, because the model will be trained on general population data but applied to a particular patient whose characteristics may or may not reflect those of the general population. In order to make the model robust, the network must dynamically adapt to new information obtained from the patient.

Perpetually using the same Bayesian Network model for a given patient also assumes that the patient's response is stationary. It assumes we are modeling a stationary probability distribution. During a hospital stay, however, the patient's response may change based on un-modeled variables. Assuming that the underlying probabil-

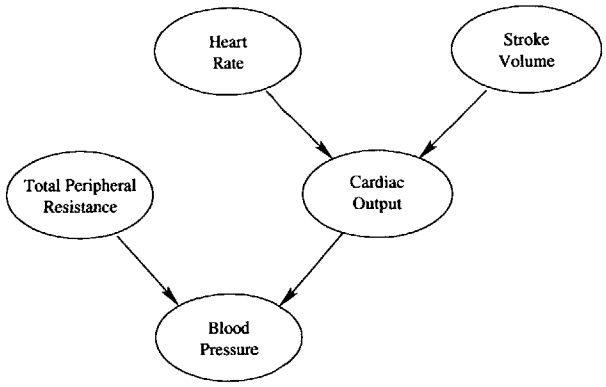


Figure 2-3: A simple Bayesian Network model of the cardiovascular system.

ity distribution is not stationary, we must continually update parameters to capture changes in the underlying probability distribution.

2.3 A Bayesian Network Model of the Cardiovascular System

As mentioned previously, I set out to create a simple Bayesian Network model of the cardiovascular system that captures necessary patient dynamics without adding extraneous model parameters that are difficult to estimate. With the help of medical experts, I developed the Bayesian Network model pictured in Figure 2-3. This model describes a probabilistic relationship among the random variables stroke volume (SV), mean arterial blood pressure (BP), total peripheral resistance (TPR), cardiac output (CO), and heart rate (HR) such that

$$P(\mathrm{SV}, \mathrm{BP}, \mathrm{TPR}, \mathrm{CO}, \mathrm{HR}) = P(\mathrm{TPR})P(\mathrm{HR})P(\mathrm{SV})P(\mathrm{BP}|\mathrm{TPR}, \mathrm{CO})P(\mathrm{CO}|\mathrm{HR}, \mathrm{SV}). \quad (2.1)$$

The model reflects relationships between cardiac output and the other random variables, namely that $CO = HR \times SV$ and $CO = \frac{BP}{TPR}$. The network represents a preliminary model upon which one could build in the future.

As mentioned in Section 2.1, the Bayesian Network structure implies that a set of

conditional dependencies and independencies exist between the node variables. Again, these independencies can be derived from the fact that each node is conditionally independent of its non-descendants, given its parents [10]. Assuming well-posedness within Figure 2-3, TPR and CO are conditionally dependent, given BP, and BP is conditionally independent of both HR and SV, given CO. Intuitively, this means that if BP is fixed, knowing the value of TPR yields additional information about CO. Similarly, if CO is fixed, knowing BP yields no additional information about HR or SV.

Although the random variables in the Bayesian Network can have probability distributions of an arbitrary form, unless the random variables are Gaussian, in practice continuous distributions are approximated by discrete probability mass functions (PMFs). Within a medical setting, many patient variables are non-Gaussian. For instance, continuous-valued variables like heart rate and blood pressure can only take values within finite-length intervals, so these variables are best modeled by non-Gaussian random variables. Similarly, discrete valued data and non-numeric data realistically take only a finite number of discrete values. Measurement noise may be Gaussian, but the actual patient parameters being measured generally are not.

Within this model, we use a Bayesian Network to model relationships between heart rate, blood pressure, cardiac output, total peripheral resistance, and stroke volume. Realistically, these patient parameters can only take values in a finite range, and median filtering removes much of the Gaussian noise. Because of this, we choose to approximate their continuous probability distributions using discrete probability mass functions. Specifically, we model the random variables in Figure 2-3 using discrete probability mass functions that assign probabilities to five quantized random variable values. Each quantized value represents a range of values that a continuous version of the same random variables might take. Because of this, the PMFs approximate continuous distributions.

Ultimately, the model depicted in Figure 2-3 could be expanded to incorporate many different types of patient data, but for the research presented in this thesis, this simple model seems appropriate. I previously explored an alternate model which

incorporated treatment information, but due to a lack of acceptable training data (see Appendix A), I chose to focus on the model presented in this section instead. Throughout the remainder of the thesis, I will explore how well this model can capture patient dynamics.

Chapter 3

Learning Algorithms

When creating a Bayesian Network for patient monitoring, one would like a model that relies on both general population information and current patient data. In this chapter, I explain several methodologies for training a Bayesian Network that use both previously available data and data that is gradually presented to the network. These learning algorithms view the Bayesian Network parameters as random variables and update their distributions based on incoming data. I first describe how we model parameters using what are known as Dirichlet distributions and then describe how we exploit these distributions to train the Bayesian Network using patient data. I finally introduce the learning algorithms explored in later chapters.

3.1 Modeling the Bayesian Network Parameters using Dirichlet Distributions

When a patient visits the hospital, his or her response often changes with time based on un-modeled variables like medications. This means that the probability distributions modeled by the Bayesian Network, and thus the Bayesian Network parameters, also change with time. In a sense, the Bayesian Network parameters become random variables themselves. When performing parameter learning, or training, we in fact view the network parameters as random variables. This allows us to use Bayesian

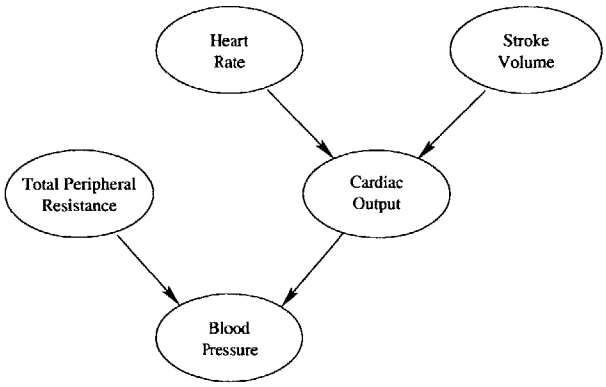


Figure 3-1: A simple Bayesian Network model of the cardiovascular system.

methods to set and update the patient parameters based on patient data.

To treat the Bayesian Network parameters as random variables, we must first define a probabilistic model from them. A good model will reveal how the parameter distributions relate to observed patient training data. This section develops how we model probability mass function (PMF) parameters using Dirichlet distributions. First, we use multinomial distributions to examine how probability parameters relate to observed patient data sets when the probability parameters remain constant. We then expand upon this idea to develop a probabilistic model for the parameters based on relationships between the parameters and observed data sets.

3.1.1 Introduction to Dirichlet Distributions

Before viewing the Bayesian Network parameters as random variables, let us first examine the relationship between a fixed set of patient parameters and multiple sets of observed patient data. Toward this end, let us take a closer look at the HR node of the cardiovascular model reproduced in Figure 3-1. Again, we model HR as a random variable whose values fall into one of five quantization bins. When an observed value falls into bin i, we will say that HR equals i. Thus, HR takes values within the set $\{1, 2, \ldots, 5\}$. Assuming that HR takes the value i with probability f_i, the numbers $\{f_1, f_2, \ldots, f_5\}$ form the set of network parameters that we need to estimate.

Assume for a moment that we have a training data set which contains N independent samples of the random variable HR. Let b_i be number of times HR takes the

value i during the N independent trials. If we then consider many different training sets containing N independent samples, the number of times HR takes the value i in a particular training set can be viewed as a random variable, B_i. Looking over the many different training sets, the random variables $\{B_1, \ldots, B_5\}$ have a multinomial probability mass function of the form:

$$P(B_1 = b_1, \ldots, B_5 = b_5) = \begin{cases} \frac{n!}{\prod_{i=1}^{5} b_i!} f_1^{b_1} \cdots f_5^{b_5} & \text{if } \sum_{i=1}^{5} b_i = N \text{ and } b_i \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This multinomial distribution is a simple extension of a binomial distribution, which describes the probability of observing b_1 successes in N independent Bernoulli trials. The multinomial distribution instead describes the probability of observing $\{b_1, \ldots, b_5\}$ occurrences of the values $\{1, \ldots, 5\}$ in N independent HR trials.

If we look at a particular data set in which the values $\{1, \ldots, 5\}$ are observed $\{b_1, \ldots, b_5\}$ times during the N trials, then $\frac{b_i}{N}$, the relative frequency with which i is observed, is the maximum likelihood estimate of f_i. This means that when f_i equals $\frac{b_i}{N}$, the probability of value i occurring b_i times during N trials is maximized.

During the above analysis, $\{B_1, \ldots, B_5\}$, the number of times that values are observed in the training data sets, are seen as random variables. The HR probability parameters, $\{f_1, \ldots, f_5\}$, and the number of observed trials, N, are seen as constants. Using this approach, we obtain what seems to be a logical estimate for the probability parameter f_i. Intuitively, however, $\{f_1, \ldots, f_5\}$ should be the random variables and $\{B_1, \ldots, B_5\}$ should be fixed, because we generally obtain only a single training data set with fixed $\{b_1, \ldots, b_5\}$. The probability parameters, on the other hand, are unknown and may change with time. Thus, these probability parameters are better modeled as random variables themselves.

In order to view the HR probability parameters as random variables, let F_i be a random variable representing the probability, or 'frequency,' with which HR takes the value i. We then model this set of PMF parameters, $\{F_1, \ldots, F_5\}$, as having what is known as a Dirichlet distribution. The reasons for choosing this distribu-

tion model should become apparent as we describe the Dirichlet distribution and its characteristics.

Using this model, the HR probability parameters $\{F_1, \ldots, F_5\}$ share the following Dirichlet probability density function [10]:

$$p(F_1 = f_1, \ldots, F_4 = f_4) = \begin{cases} \frac{\Gamma(N)}{\prod_{i=1}^{5} \Gamma(a_i)} f_1^{a_1 - 1} f_2^{a_2 - 1} \cdots f_5^{a_5 - 1} & \text{if } 0 \leq f_i \leq 1 \text{ and } \sum_{i=1}^{5} f_i = 1; \\ 0 & \text{otherwise;} \end{cases}$$

where $a_1, \ldots, a_5$ are non-negative parameters, $\sum_{i=1}^{5} a_i = N$, and $\Gamma(n)$ is the gamma function, an interpolation of the factorial function where $\Gamma(n+1) = n!$. For convenience, we use $Dir(f_1, \ldots, f_4, a_1, \ldots, a_5)$ to denote this distribution. The Dirichlet distribution parameters $a_1, \ldots, a_5$ are called Dirichlet counts.

When the HR probability parameters $\{F_1, F_2, \ldots, F_5\}$ are modeled using this Dirichlet distribution, the expected value of random parameter F_i equals $\frac{a_i}{N}$. In mathematical notation, $E[F_i] = \frac{a_i}{N}$, for valid values of i. If we interpret the Dirichlet parameter a_i to be the number of times value i is observed in N independent trials, this gives us the estimate of F_i that we would expect.

The Dirichlet parameter interpretation is actually a bit more complicated, but the initial interpretation of a_i as the number of times i is observed yields good intuition. Due to the form of the Dirichlet distribution, a_i parallels $b_i + 1$, or one plus the number of times i is observed in a particular training set. Thus, when $a_i = 1$ for all i, all of the b_i equal zero, indicating that we have not yet observed any HR information. In this case, the Dirichlet distribution simplifies to a uniform distribution, indicating we have no prior information about the HR probability parameters, $\{F_i\}$, and thus, all valid combinations of $\{F_1, F_2, \ldots, F_5\}$ are equally likely.

As new information becomes available, we can use the Dirichlet distribution to compute Bayesian updates for the probability distribution parameters. The update process is computationally simple because the Dirichlet distribution is a conjugate prior. This means that if we model the HR probability parameters using a Dirichlet distribution and compute an *a posteriori* distribution based on new data, the resulting

a posteriori is another Dirichlet distribution. More specifically, assume that we have observed one training data set in which HR values $\{1, \ldots, 5\}$ are observed $\{b_1, \ldots, b_5\}$ times during N trials. Then, if we initially view all valid combinations of HR parameter values as equally likely, we would let $a_i = b_i + 1$, let $N = \sum_{i=1}^{5} a_i$, and model the probability parameters as having a $Dir(f_1, f_2, \ldots, f_4, a_1, a_2, \ldots, a_5)$ distribution. If we then received an additional training data set, d, in which the HR values $\{1, \ldots, 5\}$ are observed $\{s_1, \ldots, s_5\}$ times during M new trials, the *a posteriori* HR parameter distribution, given the new data, is equal to

$$p(F_1 = f_1, \ldots, F_4 = f_4|d) = Dir(f_1, f_2, \ldots, f_4, a_1 + s_1, a_2 + s_2, \ldots, a_5 + s_5).$$

Based on the updated HR parameter distribution, $E[F_i|d] = (a_i + s_i)/(N + M)$, which again makes intuitive sense. In other words, the new expected HR parameter values equal the relative frequencies observed in the combined data set of $N + M$ points. For a more thorough treatment of Dirichlet distributions, please refer to [10].

3.1.2 Modeling Nodes with Multiple Parents

When a node has parents, its parameters are modeled using several different Dirichlet distributions. To understand how this works, let us first examine the PMF of an arbitrary node variable, X, that has N parents, $\{Y^1, Y^2, \ldots Y^n\}$. If X can take n_X possible values, and Y^i can take n_{Y^i} possible values, the conditional PMF for X can be parameterized by a set of $n_X \cdot \prod_i n_{Y^i}$ parameters. These PMF parameters can be indexed according to the values taken by the child and parent nodes. For instance, let x represent a particular value taken by X, and let y^i represent a particular value taken by Y^i. Then, the conditional PMF of X can be parameterized by a set of probabilities of the form $F_{x,y^1,\ldots,y^n}$, where $F_{x,y^1,\ldots,y^n}$ represents the probability that X takes value x when its set of parents, $\{Y^1, Y^2, \ldots Y^n\}$, takes the set of values $\{y^1, y^2, \ldots y^n\}$. Within our model, BP has two parents, CO and TPR, and its conditional distribution can be parameterized by probabilities of the form F_{bp_i,co_j,tpr_k}, where F_{bp_i,co_j,tpr_k} is a random variable describing the probability that BP $= bp_i$, given that CO $= co_j$ and

TPR = tpr_k. Expanding upon the notation used in the last section, HR has no parents, so its parameters simplify to the set $\{F_{hr_1}, \ldots, F_{x_{hr_5}}\}$, where F_{hr_i} describes the probability that HR takes value hr_i.

A node's parameters can be grouped into sets which describe conditional PMFs, given a certain set of parent values. If we assume that X can only take values within the set $\{x_1, \ldots, x_{n_X}\}$, the set of PMF parameters $\{F_{x_1,y^1,\ldots,y^n}, \ldots, F_{x_{n_x},y^1,\ldots,y^n}\}$ defines the PMF for X, given that $\{Y^1, Y^2, \ldots Y^n\}$ equals $\{y^1, y^2, \ldots y^n\}$. Since these parameters form a PMF, they must sum to one, i.e., $\sum_{i=1}^{x_{n_X}} f_{x_i,y^1,\ldots,y^n} = 1$. Since each parameter set of this form creates a single PMF, the parameters within such a set share a Dirichlet distribution. For example, the BP parameters for which CO = 1 and TPR = 1 together form the conditional PMF described by $P(\text{BP}|\text{CO} = 1, \text{TPR} = 1)$, and this set of parameters share the same Dirichlet distribution.

3.1.3 Model Complexity

The number of PMF parameters determine the complexity of our model, since each of these parameters should ideally be learned using patient data. If the number of parameters becomes too large, or large numbers of PMF parameters correspond to node variable value combinations that do not occur within the training data sets, it becomes difficult to confidently define the Bayesian Network model using patient information.

The number of PMF parameters assigned to a particular node depends on how many parents that node has, how many values it takes, and how many values its parents take. If node X takes n_X possible values, and its parents $\{Y^1, Y^2, \ldots Y^n\}$ take $\{n_{Y^1}, \ldots, n_{Y^n}\}$ values apiece, X has $n_X \cdot \prod_i n_{Y^i}$ PMF parameters. Thus, by limiting the number of parents a particular node has, we can keep the number of parameters at a manageable level.

3.2 Setting and Updating the Dirichlet Distributions

By using Dirichlet distributions to stochastically model the conditional PMF parameters, we create a probabilistic model with multiple levels. At the top level, the Bayesian Network represents a discrete probability distribution stored as a set of conditional node distributions. In our model, this top-level distribution is the joint probability distribution of HR, TPR, BP, CO, and SV, and it is stored as a collection of conditional PMFs, one for each node variable. The PMF parameters like $\{F_{hr_1}, F_{hr_2}, \ldots, F_{hr_5}\}$ describe the frequencies with which the node variables take particular values. At the next level, the PMF parameters are modeled as random variables with Dirichlet distributions. These Dirichlet distributions depend on other parameters called Dirichlet counts.

Each PMF parameter has a corresponding Dirichlet count that stores information relevant to that parameter. The Dirichlet count corresponding to the PMF parameter F_{x_i,y^1,y^2} intuitively represents the number of times that X equals x_i, Y^1 equals y^1, and Y^2 simultaneously equals y^2 within the available data set. To initialize or train the Bayesian Network, one adjusts the Dirichlet counts. The PMF parameters are then taken to be the expected values of the Dirichlet distributions. These expected values can be calculated directly from the Dirichlet counts as explained in Section 3.1. Thus, by updating the Dirichlet counts, we use Dirichlet distributions to set PMF parameters based on incoming data.

The Dirichlet counts are initially set to values based on prior belief about the node variable distributions. For example, if X has no parents and one feels strongly that X is likely to take the value 1 and unlikely to take the value 2, one sets the Dirichlet count corresponding to $X = 1$ to a high value and the Dirichlet count corresponding to $X = 2$ to a low value. When training the network on data, the Dirichlet counts are updated based on the number of times the corresponding set of parent and child nodes simultaneously take a particular set of values within the data set. Essentially, the learning procedures compute histograms for each set of parent and child node

values and use these histograms to update the Dirichlet counts.

When initializing the Dirichlet counts, one must ensure that the Dirichlet counts for different network nodes do not conflict with one another. Consider the toy example pictured in Figure 3-2. In this Bayesian Network, the Dirichlet functions reduce to beta functions since each function only has two parameters. F_{x_1}, the parameter corresponding to the probability that $X = 1$, has a beta(1, 1) distribution, which is similar to observing $X = 1$ once and $X = 2$ once during a total of two observations. F_{x_2}, the parameter corresponding to $X = 2$, is defined be $1 - F_{x_1}$ because the sum of the PMF parameters must equal one. F_{y_1} has a beta(1, 1) distribution when $X = 1$ and a beta(1, 0) distribution when $X = 2$, which is similar to observing $(X, Y) = (1, 1)$, $(X, Y) = (1, 2)$, and $(X, Y) = (2, 1)$ in three observations. F_{y_2} again equals $1 - F_{y_1}$. In this case, the number of observations used to set beta counts for the X and Y parameters differ. This discrepancy can cause undesirable results as the Dirichlet counts are updated based on incoming information [10]. To avoid these complications, one must initialize the Dirichlet counts so that they are self-consistent, i.e., so that the sum of the counts assigned to each node is the same, and the counts can all be drawn from the same set of observations. When a Bayesian Network has self-consistent Dirichlet counts, the network is said to have an equivalent sample size equal to the sum of the Dirichlet counts assigned to a particular node. For example, if F_{x_1} received a beta(2, 1) distribution in Figure 3-2, then the conflict would be removed, and the network would have an equivalent sample size of $2 + 1 = 3$. For a more rigorous exploration of equivalent sample sizes, refer to [10].

In addition to ensuring that the Dirichlet counts are initialized consistently, one should consider how their initial values will affect the speed with which the Bayesian Network adapts to new information. Since the PMF parameters are set to the expected values of the underlying Dirichlet distributions, many different Dirichlet distributions can result in the same PMF. If we again consider Dirichlet distributions with two parameters, all of the distributions pictured in Figure 3-3a result in the same PMF since they all have the same expected value. More specifically, suppose Z is akin to a Bernoulli random variable, and Z equals 1 with probability $f_1 = E[F_1]$ and

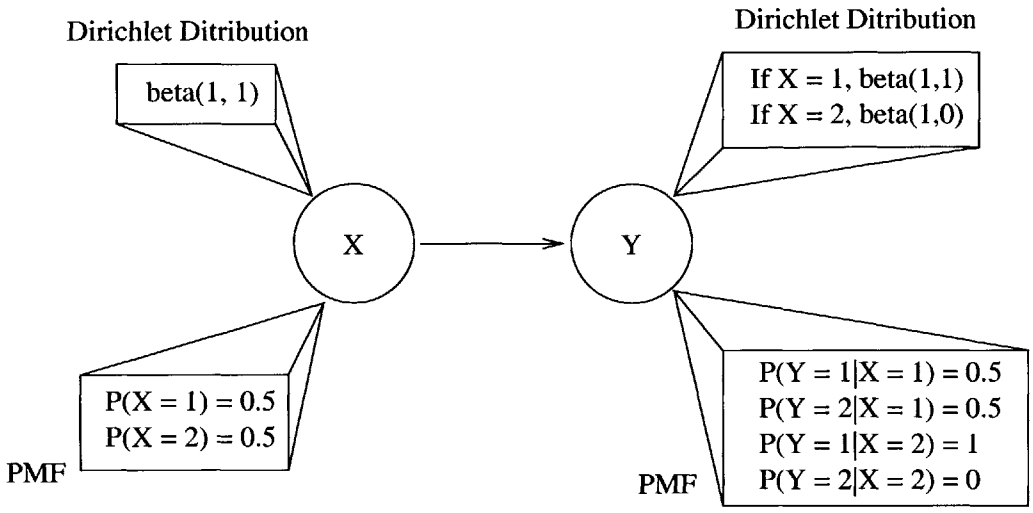
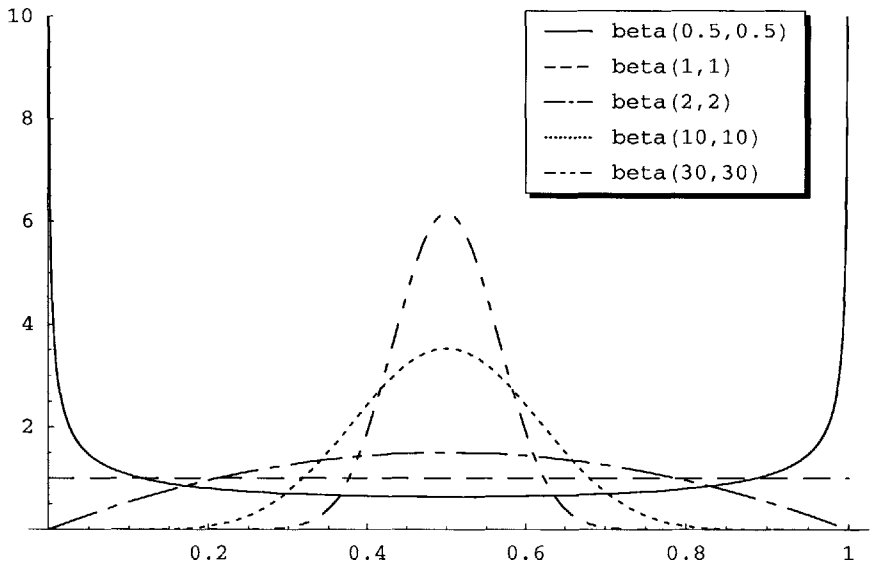
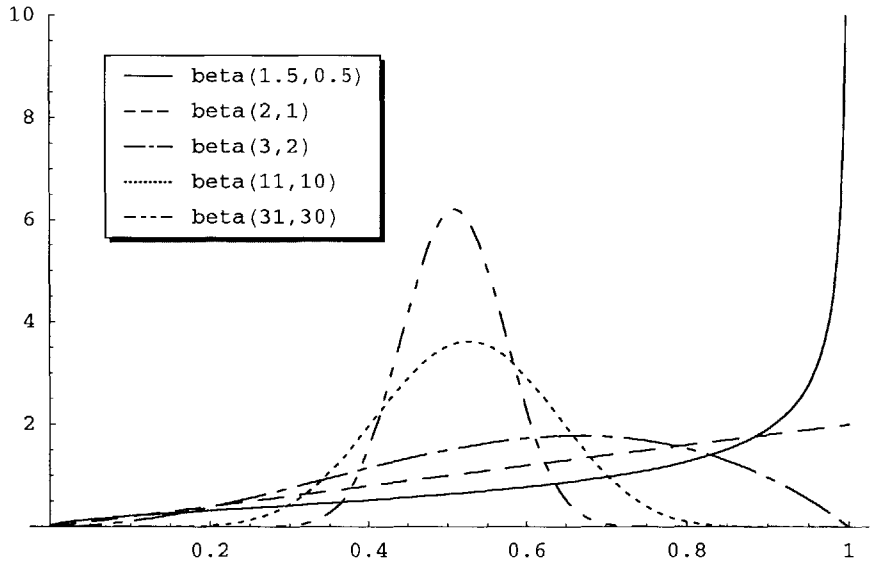


Figure 3-2: A Bayesian Network example that does not exhibit an equivalent sample size.

2 with probability $f_2 = E[F_2] = 1 - f_1$. Assume that the beta(a_1, a_2) plots in Figure 3-3a now represent probability distribution for the random variable F_1. In all cases, $f_1 = f_2 = 0.5$, since the beta distributions all have an expected value of 0.5. Each beta distribution, however, results in a different posterior distribution when certain data observations are made. Suppose we take each of the beta distributions in Figure 3-3a to be a prior distribution of F_1, and we calculate the posterior distributions after observing a new sample of Z. If we observe that $Z = 1$, the resulting posterior distributions are pictured in Figure 3-3b. When the original beta parameters were less than 1, observing that $Z = 1$ makes f_1 very close to 1 and $f_2 = 1 - f_1$ very close to 0. Alternatively, when the original beta parameters were both one, the prior is uniform and f_1 becomes $\frac{2}{2+1} = 0.67$, a value much closer to its *a priori* value of 0.5. For the final two cases, f_1 stays very close to 0.5, because a more significant amount of prior evidence indicates that f_1 and f_2 are equal. Thus, by setting the *a priori* Dirichlet counts in different ways, we can change how quickly the PMF parameters adapt to new data. The learning algorithms presented in the next section experiment with ways of initializing and changing the Dirichlet counts in this manner.



(a)



(b)

Figure 3-3: Examples of two-parameter Dirichlet distributions, otherwise known as beta distributions.

3.3 Learning Algorithms

In the medical setting, one wants to incorporate knowledge gained from medical experts, general population statistics, and current patient data. An ideal Bayesian Network should rely on well-established information about the general population, while still adapting to the current patient. The learning, or training, algorithms presented in this section perform distribution parameter learning by relying on different ways of setting and constraining Dirichlet parameters to affect how quickly a Bayesian network adapts to new information and how strongly it depends on previously obtained history.

We can set a Bayesian Network's probability parameters using some combination of pre-training initialization, batch learning, and sequential learning. During pre-training initialization, we set the Dirichlet counts based on expert opinion and previous experience. In batch learning, we use information from previously acquired patient data to set or update the Dirichlet counts all at once. During sequential learning, we use incoming patient data, either one point at a time or several points at a time, to continuously update the Dirichlet counts based on recently acquired patient data.

The training algorithms presented in this section represent different ways of combining batch and sequential learning approaches to vary the amount and type of information stored in the network's probability distributions. We us these methodologies to create networks that incorporate both persistent and transient memory components.

3.3.1 Batch Learning

During batch learning, the network receives all of the training data at once, calculates a single set of posterior Dirichlet distributions, and uses the same resulting PMFs and conditional PMFs every time that the network performs inference in the future. In other words, the network memory is persistent, and the Bayesian Network's probability distribution remains stationary.

Whenever we perform batch learning in this thesis, we precede the batch learning by a pre-training initialization step. During this step, the Dirichlet counts are initialized uniformly so that the network has an equivalent sample size of 1. Since all nodes are modeled using five-level multinomial random variables, Dirichlet counts for nodes with no parents are initially set to 0.2, and Dirichlet counts for nodes with two parents initially equal 0.008. This means that all *a priori* PMFs and conditional PMFs are uniform, but as soon as the network receives training data, it immediately assigns an extremely low probability to values not appearing in the observed data set. This mirrors the beta(0.5, 0.5) case presented in Figure 3-3a.

During learning, the batch algorithm computes histograms for each set of child and parent nodes to count the number of times that different value combinations appear within the training data set. In the Bayesian Network model pictured in Figure 3-1, the learning algorithm independently computes histograms for TPR, HR, and SV, the nodes with no parents. The algorithm then adds the histogram results to the initial Dirichlet counts to obtain the updated Dirichlet counts. It then computes joint histograms for each set of {TPR, CO, BP} and {HR, SV, CO} values, and adds the appropriate histogram count to each of the corresponding initial Dirichlet counts. In this manner, it obtains updated Dirichlet counts for BP and CO, respectively. The algorithm sets each PMF parameter to the expected value of the appropriate updated Dirichlet distribution as explained in Sections 3.1 and 3.2. Once the updated PMF parameters are calculated, the training is complete. Each time the network is subsequently used for inference, the network uses this same set of PMF parameters.

3.3.2 Sequential Learning with Transient Memory

During sequential learning, the posterior distributions are recalculated every time that the network obtains a new piece of training data. In pure sequential learning, histograms are first computed on the new piece of training data in the same way as described above. The histogram values are again added to the Dirichlet priors to obtain the updated Dirichlet counts. The updated Dirichlet counts then become the Dirichlet priors used the next time the network receives another set of training

data. This process is repeated each time the network receives data. In a sense, sequential learning is batch learning repeated over and over again on smaller data sets. Frequently each data point is presented to the network individually, but the data can be segmented in many different ways during sequential learning.

When the network performs inference, it uses the PMFs obtained from the most recent Dirichlet counts. Thus, the same Bayesian Network may use different probability distributions when performing inference at different times, if it has been sequentially trained during the interim.

To perform sequential learning with a transient memory of size n, the network is again initialized using the same pre-training initialization procedure described in the batch learning section. The nodes without parents train sequentially on the first n training data points, but this time they store a copy of the data history. When the network receives the $(n+1)^{st}$ training data point, the nodes increment the appropriate Dirichlet counts and then subtract out the Dirichlet counts obtained from the first data point. In general, when the parentless nodes receive information from the $(n+k)^{th}$ training data point, where k is a positive integer, they increment their Dirichlet counts based on the $(n+k)^{th}$ data point and decrement the Dirichlet counts based on values from the k^{th} data point. Thus, the new Dirichlet counts equal the initial counts plus the counts obtained from the last n data points seen.

The nodes with parents follow a similar procedure, but they store n most recent *relevant* data points instead. Considering the conditional PMF for BP when CO = 1 and TPR = 1, the parameters for this PMF train only on data points for which CO and TPR both equal 1. Thus, these parameters initially train on the first n data points for which CO = TPR = 1. We will refer to these data points as *relevant* data points. When the network receives its $(n+1)^{st}$ relevant training data point, i.e., the $(n+1)^{st}$ point for which CO = TPR = 1, the $P(\text{BP}|\text{CO}=1,\text{TPR}=1)$ parameters increment the appropriate Dirichlet count and then subtract out the Dirichlet count obtained from the first data point. Similarly, when these parameters receive the $(n+k)^{th}$ relevant training data point, where k is a positive integer, they increment the appropriate Dirichlet count based on the $(n+k)^{th}$ data point and decrement the

Dirichlet count based on the k^{th} data point. Other conditional PMFs respond to data in the same way. Each conditional PMF, conditioned on the parent random variables $\{Y^1, Y^2\}$ taking values $\{y_i^1, y_j^2\}$, is set using the n most recent relevant data points, or the n most recent data points for which the parents Y^1 and Y^2 take the values y_i^1 and y_j^2, respectively.

3.3.3 Combinational Learning

Combinational learning combines aspects of both sequential and batch learning. In combinational learning, the Dirichlet counts are initialized using the same pre-training initialization procedure as before. Then, the network receives a batch of data containing m data points to be stored in persistent memory. The network is batch trained on these m data points. The resulting network is then sequentially trained, with a transient memory size of n, on the incoming data. The Dirichlet counts at any point in time equal the initial counts plus counts from the m data points stored in persistent memory plus counts from the last n relevant data points seen. The network's probability distribution then has both stationary and non-stationary components.

This type of learning is similar to that described in [1]. The relative sizes of the persistent (batch) and transient (sequential) memories determine how much weight is assigned to each type of information. Within the current algorithm, the persistent memory size is determined by the number of points used to batch train the network. When this number of points is large relative to the sequential transient memory size, the information used to batch train the network is weighted heavily. When number of points used to batch train the network is small relative to the transient memory size, the incoming patient data is weighted heavily. In future versions of this algorithm, an additional typicality parameter could be added to more explicitly control the relative weight assigned to these two types of information.

Chapter 4

Application to Patient Data

In this chapter, we test our Bayesian Network model of the cardiovascular system using both real patient data and simulated patient data. We then analyze how accurately the model can estimate a patient's cardiovascular parameters, given generally available patient information. The first section reviews the Bayesian Network model and gives an overview of the general procedure used during the test trials. The second section describes procedures used and results obtained when testing the network with simulated patient data, and the third section describes procedures used and results obtained when testing the network with real patient data.

4.1 The Bayesian Network

The Bayesian Network model of the cardiovascular system introduced in Section 2.3, and reproduced in Figure 4-1, describes probabilistic relationships between cardiac output (CO), stroke volume (SV), total peripheral resistance (TPR), heart rate (HR), and blood pressure (BP). The nodes again represent random variables, and the arrows represent conditional dependencies. The node variables are modeled as multinomial random variables which take one of five quantized values, and the five quantization levels are chosen to cover the patient data sets described below.

The number of quantization levels determines the number of PMF parameters stored by the Bayesian Network nodes. Within Figure 4-1, the HR, SV, and TPR

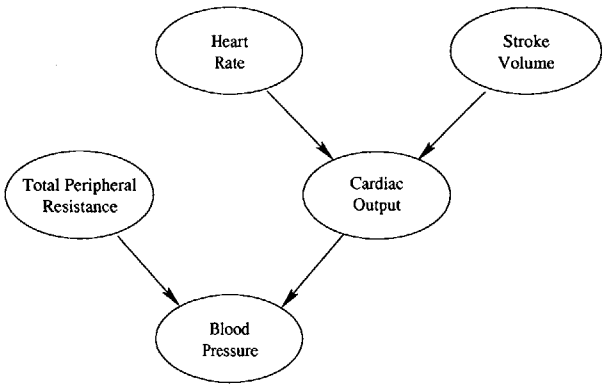


Figure 4-1: A simple Bayesian Network model of the cardiovascular system.

nodes contain five probabilities, one for each of the five valid quantization levels. Since the CO and BP nodes have two parents, they contain more probability parameters. Due to their two parents, the nodes store twenty-five conditional PMFs, one for each set of parent node values. Thus, these nodes both contain one-hundred twenty-five probabilities apiece, five for each of the twenty-five conditional PMFs. Each group of five PMF parameters is modeled as having a joint Dirichlet distribution, and these Dirichlet distributions are defined in terms of Dirichlet counts. With one Dirichlet count per PMF parameter, the Dirichlet counts store how frequently data combinations described by their corresponding PMF parameters occur within the training data sets.

In this chapter, we train the model in Figure 4-1 using the learning algorithms described in Section3.3, and then we test the network's ability to estimate SV, CO, and TPR, given BP and HR. At the beginning of each trial, we initialize the Bayesian Network probability distributions using the pre-training initialization procedure described in Section 3.3.1, so both the conditional distributions and the underlying Dirichlet distributions are initially uniform, and the Dirichlet distributions have an equivalent sample size of one. We then train the network using one set of quantized training data and if possible, test the network by performing inference on a second quantized data set. During sequential learning, we use the same data set for both training and inference due to the sequential nature of the algorithm. During testing,

the network uses its current probability distributions to estimate current quantized values of CO, SV, and TPR, given current quantized values of HR and BP. We then compare the actual quantized patient data to the quantized estimates to evaluate the network's estimation accuracy.

4.2 Preliminary Tests using Simulated Data

We first tested the Bayesian Network using simulated patient data to ensure that the network could learn the types of relationships found in a patient setting. When dealing with actual patients, measurements may be inaccurate and noisy, and many internal patient parameters cannot be measured. In a simulated environment, however, one can completely control all patient parameters, and one knows the exact actual value of all internal and external patient parameters. Thus, when we ask the Bayesian Network to estimate internal patient parameters, we know with complete certainty what the output should be.

In this section, we batch train the network on portions of simulated data and test how accurately the network can estimate CO, SV, and TPR, given current values of HR and BP. We find that the network accurately estimates unknown variables at a rate far higher than that obtained through random chance.

4.2.1 Simulated Data

Data Generation

For the preliminary work reported in this section, the data used to train the Bayesian Network was obtained by simulation of the simple pulsatile cardiovascular model introduced in [12]. Figure 4-2 illustrates the circuit representation for the model. C_a is the arterial compliance, C_v is the venous compliance, $C_h(t)$ is the time-varying compliance of a single ventricular chamber, R_1 is the inflow resistance to the ventricle, R_2 is the outflow resistance from the ventricle, and R_3 is the total peripheral resistance. The pressure V_h is the ventricular pressure, V_v is the central venous pressure, and V_a

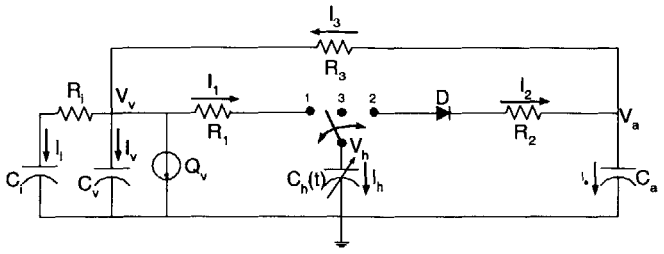


Figure 4-2: The simple pulsatile cardiovascular model. The model has a single ventricular compartment (R_1, $C_h(t)$, R_2), an arterial compartment (C_a, R_3), a venous compartment (C_v), an interstitial fluid compartment (C_i, R_i), and a blood infusion or hemorrhage source Q_v.

is the arterial blood pressure. The ventricular volume is Q_h.

The elastance function $E_h(t)$=$1/C_h(t)$ for the ventricle is represented as a piecewise linear function [12] given by:

$$E_h(t) = \begin{cases} \frac{3(E_s-E_d)}{T}t + E_d & \text{for} \quad 0 \leq t \leq \frac{T}{3} \\ \frac{6(E_s-E_d)}{T}(\frac{T}{3} - t) + E_s & \text{for} \quad \frac{T}{3} \leq t \leq \frac{T}{2} \\ E_d & \text{for} \quad \frac{T}{2} \leq t \leq T \end{cases} \tag{4.1}$$

where T is the duration of the cardiac cycle, E_s is the end-systolic elastance, and E_d ($\ll E_s$) is the end-diastolic elastance. State equations for this model can be derived as in [12], but with an additional equation representing the interstitial compartment dynamics. The nominal parameters used in the model are shown in Table B.1 in the appendix.

To generate testing and training data, we used the model to run twenty 500-second-long simulations during which one or more patient parameters deviated from their nominal values. These deviations are described in Table 4.1. The cardiovascular responses, sampled at a rate of 10 Hz, were used to compute the beat-to-beat averaged values of the arterial and venous blood pressures, as well as the average flow across the total peripheral resistance. These variables were then used to calculate cardiac output. Using heart rate (HR=$\frac{60}{T}$) and CO, we calculated SV. These five variables,

Data Segment	Description of parameter changes
1	Nominal Parameters
2	TPR increases from 1 to 5
3	Q_v=4 ml/s
4	Q_v=-4 ml/s
5	TPR decreases from 1 to 0.3
6	C_a decreases from 2 to 0.6
7	HR is stepped up from 60 to 300 bpm
8	HR is stepped down from 60 to 30 bpm
9	C_v decreases from 50 to 20
10	R_2 increases from 0.01 to 0.5
11	R_2 decreases from 0.01 to 0.001
12	E_s increases from 2.5 to 5
13	E_d decreases from 0.1 to 0.05
14	E_s decreases from 2.5 to 1
15	E_d increases from 0.1 o 0.2
16	Q_v=4 ml/s and TPR increases from 1 to 3
17	Q_v=-4 ml/s and TPR decreases from 1 to 0.5
18	HR is stepped up from 60 to 300 bpm TPR decreases from 1 to 0.5
19	HR is stepped down from 60 to 30 bpm TPR decreases from 1 to 3
20	HR is stepped up from 60 to 300 bpm TPR increases from 1 to 3

Table 4.1: Description of the training data segments.

HR, SV, CO, BP, and TPR, were saved for each of the twenty simulations. The first ten samples of each simulation were discarded to remove unrealistic transient simulation effects, and the resulting data was then used to train and test the Bayesian Network.

Training and Test Sets

The data was segmented into ten second intervals. Half of the intervals were chosen uniformly at random to be included in the training set, while the remaining intervals were placed in the test set. This random selection process was repeated fifty times to obtain fifty different training and test sets. This procedure was repeated for interval sizes of 100 seconds and 250 seconds.

Data was then quantized by placing values into five equally sized bins that cover

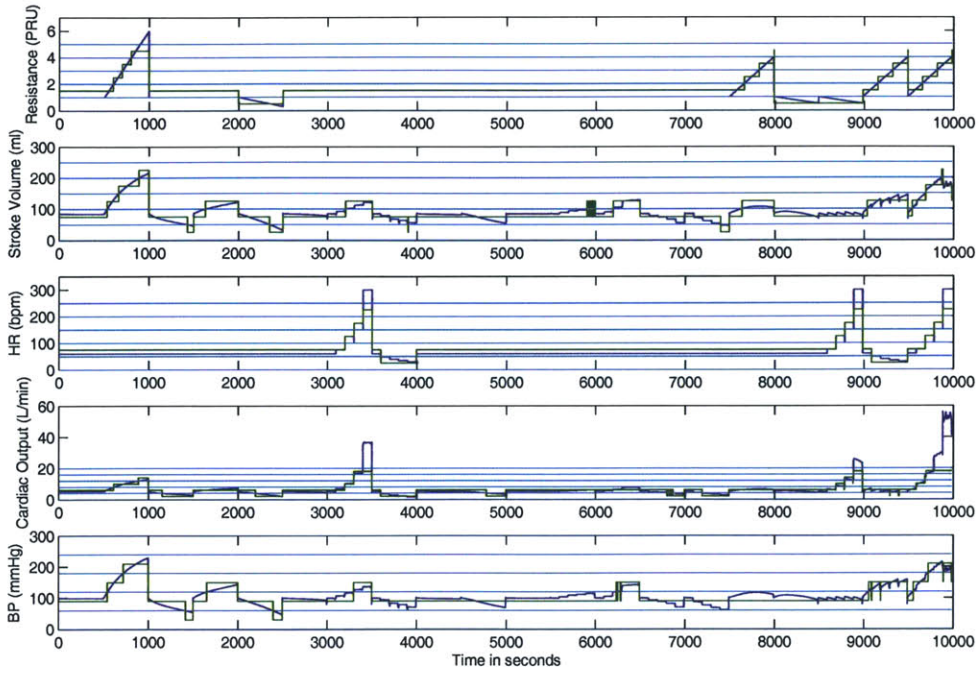


Figure 4-3: Simulated data

an acceptable dynamic range for each variable of interest. For example, HR was quantized to values of 25, 75, 125, 175, and 225 beats per minute. These quantized values were mapped to bin numbers, with the smallest quantization range mapping to bin 1, and the largest mapping to bin 5. The quantized values used when training on simulated data are shown in Table B.2).

Figure 4-3 shows the data obtained from each of the twenty simulations. The simulation results are concatenated, so the large discontinuities indicate the end of one simulation and the beginning of another. The blue waveforms show the simulation outputs, while the green quantized waveforms show the quantized versions of the simulation outputs that are used to train and test the network. The horizontal lines indicate the quantization thresholds. Any values larger than the highest quantization threshold are quantized to the midpoint of the highest quantization range, while any values lower than the lowest quantization threshold are quantized to the midpoint of the lowest quantization range.

4.2.2 Network Training

After pre-training initialization, we first batch trained the network model on the entire data set to obtain a baseline for comparison. Throughout batch training, the network received data for all five node variables at each time step. After training on the whole data set, the network used the resulting distribution to perform inference each time it computed estimates. Estimates obtained from the network trained in this manner reveal the optimal estimates for our data set. We then batch trained additional networks on each of the randomly segmented training sets described above.

4.2.3 Testing the Trained Network

We provided each trained network with quantized BP and HR values from the corresponding test set and used the network to obtain estimates of CO, TPR, and SV, given the HR and BP values. We computed both quantized minimum mean square error (MMSE) estimates and maximum *a posteriori* (MAP) estimates as follows:

$$\begin{aligned} MMSE(\mathrm{X}|\mathrm{BP},\mathrm{HR}) &= \{E[\mathrm{X}_q|\mathrm{BP}_q,\mathrm{HR}_q]\}_q \\ &= \{\Sigma x_q P(x_q|\mathrm{BP}_q,\mathrm{HR}_q)\}_q, \end{aligned} \tag{4.2}$$

and

$$MAP(\mathrm{X}|\mathrm{BP},\mathrm{HR}) = \arg\max_{x_q}\{P(x_q|\mathrm{BP}_q,\mathrm{HR}_q)\}, \tag{4.3}$$

where X is either CO, TPR, or SV; the subscript q indicates that the corresponding value is quantized; and x_q ranges over the set of quantized values taken by the random variable X. We then mapped the quantized estimates to their corresponding bins and compared the binned estimates with the actual binned values from the corresponding test set.

4.2.4 Error Analysis

Any time an estimate did not fall in the same bin as the actual value, the estimate was counted as an error. Bins were used to calculate the errors since identifying the relative range of the variable of interest is normally acceptable in a clinical setting. We examined two types of error rates, the absolute error rate, i.e., the number of errors divided by the total number of samples, and the root mean square error. The bin numbers were used to calculate the root mean square error as $\sqrt{\frac{(estimated_b - actual_b)^2}{numsamples}}$, where $estimated_b$ is the estimated bin number, $actual_b$ is the actual value's bin number, and *numsamples* is the total number of points in the test data set. By using the bin numbers, the error rate for each variable was normalized to the same scale.

4.2.5 Results

Error rates and standard deviations increased as the segment size increased, since data combinations not seen in the training sets were more likely to appear in the test sets when the segment size was large. Analysis is presented for the 250-second segment case since those error rates are the most conservative.

Error rates for both types of estimates were comparable. As seen in Table 4.2, the MAP estimates provided a slightly smaller absolute error rate, and the MMSE estimates provided a slightly smaller root mean square error, as expected, but the differences do not seem statistically relevant. If left to random chance, the expected

Estimator	Mean Absolute Error Rates (STD) MMSE	MAP	Mean Root Mean Square Error (STD) MMSE	MAP
Resistance	0.33 (0.06)	0.31 (0.06)	0.73 (0.14)	0.77 (0.17)
Cardiac Output	0.30 (0.07)	0.27 (0.07)	0.72 (0.16)	0.78 (0.23)
Stroke Volume	0.26 (0.08)	0.29 (0.06)	0.68 (0.16)	0.71 (0.14)

Table 4.2: A comparison of the mean error rates and standard deviations for the data sets with 250 second segments, where MMSE denotes the minimum mean square error estimates and MAP denotes the MAP estimates.

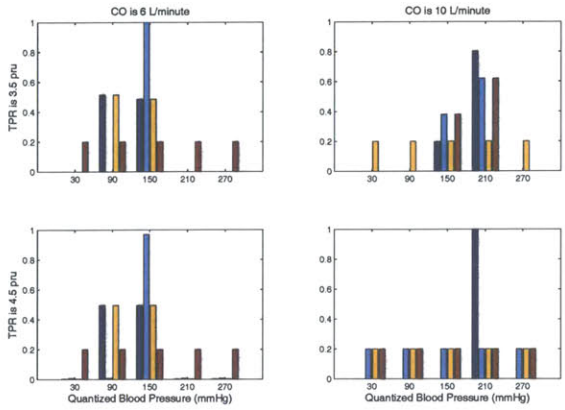


Figure 4-4: Conditional probability mass functions for BP, given CO and TPR. Each plot shows the conditional PMF of BP for a particular set of CO and TPR values.

absolute error rate would be approximately eighty percent, so the Bayesian Network is performing far better than random chance. The 25% to 33% absolute error rates indicate that the estimates fall in the correct bin more than two-thirds of the time. The small root mean square errors indicate that when the estimate does not fall in the same bin as the actual value, it tends to fall in a nearby or adjacent bin.

Errors more frequently occur when a combination of values appears in the test set but not the training set. Figure 4-4 shows how such occurrences affect a subset of the learned conditional probability mass functions (PMFs). In Figure 4-4, each set of bars of a given shading displays the PMF obtained when the network was trained using a different training set. The first bar in each group was obtained by training on the entire data set, while the other bars were obtained using partial training sets. When a particular set of BP-CO-TPR value combinations was not found in the training set, the PMF remained uniform. The plot on the bottom right displays such a case.

4.3 Tests using Real Patient Data

While simulated data provides exact information about all internal patient parameters, simulated data cannot predict how a Bayesian Network model will respond to patients in the ICU. By training and testing the network using real patient data, however, we can gain insight into how a Bayesian Network model might perform in an ICU

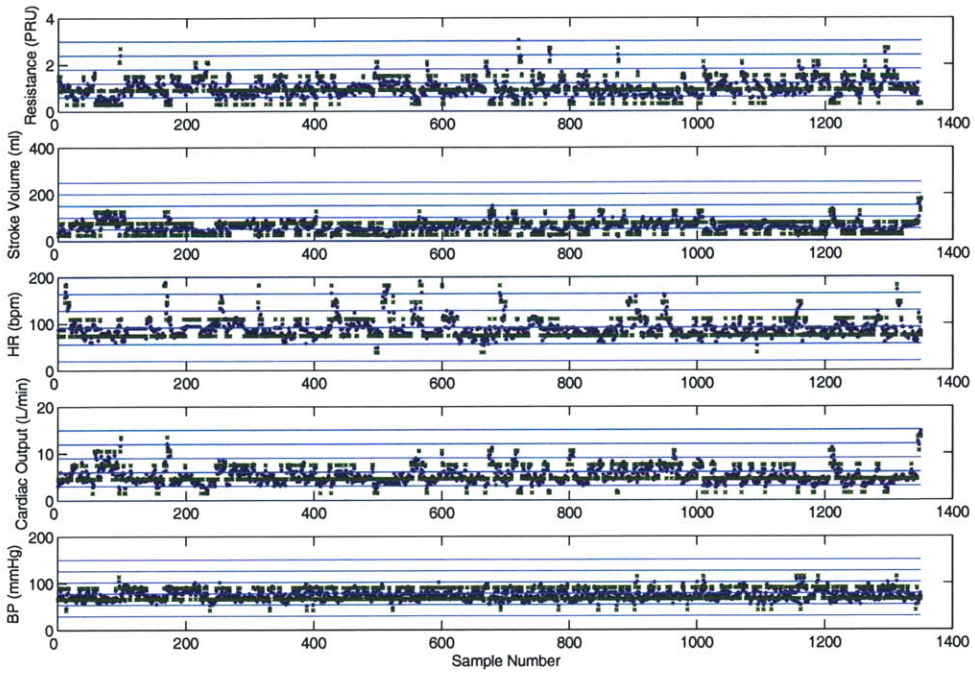


Figure 4-5: Data samples containing thermodilution measurements. The stars indicate actual values while the crosses represent quantized values. The data samples are plotted versus the sample set numbers.

setting. The following section explores how our Bayesian Network model performs when presented with ICU patient data from the Multi-parameter Intelligent Monitoring for Intensive Care (MIMIC) patient database (http://mimic.mit.edu/index.html).

4.3.1 Patient Data

All of the patient data used in this section came from a database of one-hundred-twenty de-identified ICU patient records, all of which contain thermodilution cardiac output measurements. The data sets used to test and train the network fall into two classes. The first class contains the actual thermodilution cardiac output measurements and measurements of the concurrent heart rates and mean arterial blood pressures. The HR and BP values were obtained by averaging one minute of the patient waveform data prior to the time at which the cardiac output measurements were taken. Corresponding values of SV and TPR were then calculated using $SV = \frac{CO}{HR}$

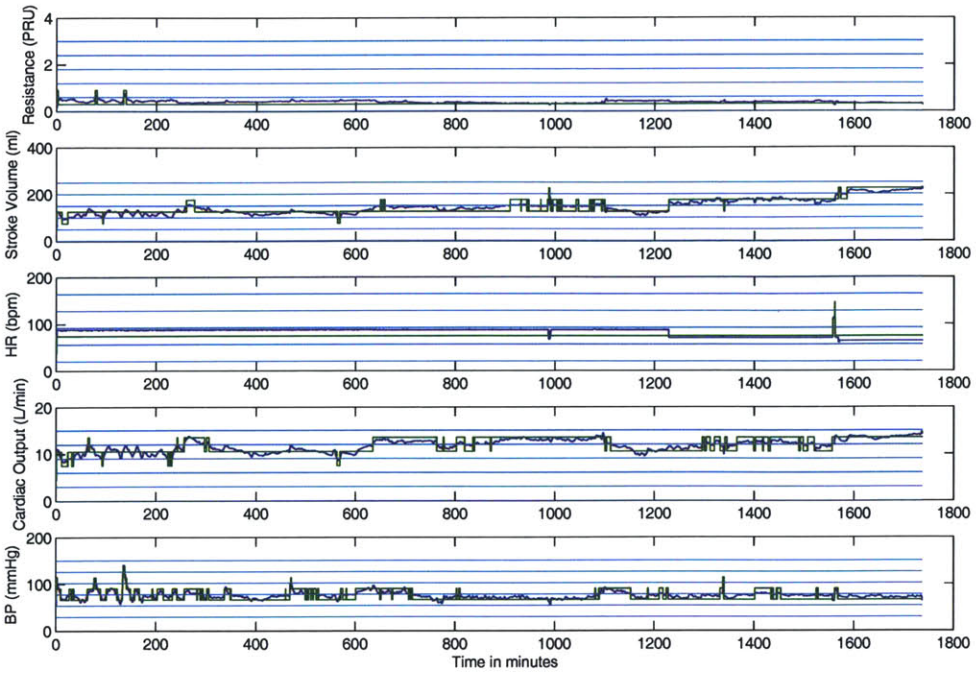


Figure 4-6: Data from patient 11007. Original waveforms are shown in blue, quantized waveforms are shown in green, and the quantization thresholds are shown in aqua.

and $TPR = \frac{BP}{CO}$. Any set of CO-HR-BP-TPR-SV measurements containing missing values was discarded. The resulting data set, denoted as the Gold Standard Data, contains 1351 sets of measurements taken from 120 different patients.

The second data set consists of full patient records extracted from the 120-patient database mentioned above. Using the data from each patient record, beat-to-beat estimates of cardiac output were calculated using Liljestrand's method [7]. Beat-to-beat averaged BP waveforms were obtained from the 125-Hz blood pressure waveforms from the database. We used the beat-to-beat BP, HR, and CO values to calculate beat-to-beat values for SV and TPR. All of the waveforms were then median filtered to reduce noise. The resulting waveforms were taken as the actual values of HR, BP, CO, TPR, and SV when training and testing the Bayesian network. For the most part, we restrict our attention to a particular patient from the database, Patient 11007.

Both the Gold Standard Data and full patient record data were quantized by

placing values into five equally sized bins that cover an acceptable dynamic range for each variable of interest. These quantized values were again mapped to bin numbers, with the smallest quantization range mapping to bin 1, and the largest mapping to bin 5. The quantized values used when training on real patient data are shown in Table B.3). Any values above or below the most extreme thresholds were mapped to the nearest quantization bin. Figure 4-5 shows that data samples contained in the Gold Standard Data set as well as their quantized values and the quantization thresholds. Figure 4-6 shows the waveform data, quantized waveforms, and quantization thresholds for Patient 11007.

4.3.2 Batch Training and Prediction

First we batch trained the Bayesian Network on the Gold Standard Data, and tested it using the data from Patient 11007. This means that we used the Gold Standard Data to set the probability distribution once and used the same distribution to perform inference on the entire Patient 11007 data set. After training the network once on all of the Gold Standard Data, we successively presented it with each set of the patient's HR and BP values and let the network compute the MMSE and MAP estimates of the corresponding CO, SV, and TPR values, following the same procedure used in Section 4.2.3. Since the network was batch trained, the same probability distribution was used to compute each of the estimates. We again compared the binned estimates to the binned versions of the actual SV, CO, and TPR values, i.e., binned versions of the values obtained using Liljestrand's method, and determined that the network failed to accurately estimate CO and SV nearly one hundred percent of the time. Figure 4-7 shows the actual CO, SV, and TPR waveforms, the quantized waveforms, the MAP estimates, and the MMSE estimates obtained using batch training. As shown in this figure, the MAP and MMSE estimates fail to accurately track the CO and SV waveforms.

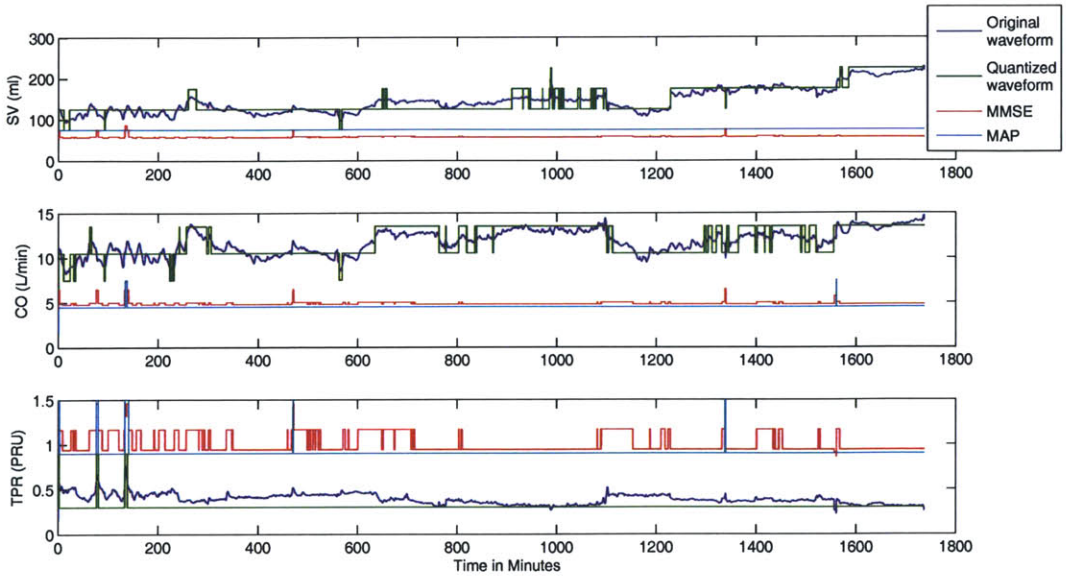


Figure 4-7: Comparison of actual Patient 11007 TPR, CO, and SV waveforms to estimated waveforms obtained using a network solely trained on the Gold Standard Data.

Comparison of MMSE and MAP Estimators

Because the MAP estimates can only take the quantized values represented in the probability distribution, the MAP estimates almost always equal quantized versions of the corresponding MMSE estimates. In the rare instances where this is not the case, the marginal distribution used to calculate the estimates has more than one significant peak. The MMSE estimate reflects the bi-modal nature of the distribution, while the MAP estimate does not. In general, the MMSE estimates are more interesting because they can take unquantized values. Throughout the remaining trials, I focus my analysis on the MMSE estimates for simplicity. Because the MAP estimates tend to equal the quantized MMSE estimates, the analysis for MAP estimates follows as an easy extension.

4.3.3 Single Beat Prediction using Sequential Training

Since batch training alone resulted in unsatisfactory performance, I explored sequential training methods, that is, methods which allowed the network to continually

update its probability distributions based on incoming patient data. The networks described in this section were sequentially trained using finite memories of various sizes.

Because the patient record data is averaged on a beat-to-beat basis, each set of data values presented to the network represents information from a single beat. Thus, a finite memory of one thousand points indicates that the current probability distributions depend on information from the most recent thousand heart beats relevant to the probability distribution in question. For example, the probability distributions for HR, SV, and TPR, variables in our network with no parents, depend on the most recent thousand heart beats. On the other hand, distributions for CO and BP, nodes in our network with two parents, depend on the last thousand heart beats for which the parent random variables took a particular set of values. In other words, $P(CO|HR_b = 1, SV_b = 1)$, where HR_b and SV_b represent the bin numbers for HR and SV, respectively, depends on the last thousand beats for which $HR_b = 1$ and $SV_b = 1$, and sets of beats of this sort frequently extend further back in time than the most recent thousand heart beats.

When tested, the sequentially trained network uses the probability distribution learned from beats 0 through t to calculate its $(t+1)^{st}$ estimates. After updating its probability distributions based on information from beat t, the network receives HR and BP information from beat $t+1$ and uses this information to estimate the CO, SV, and TPR associated with the $(t+1)^{st}$ beat. Since the network uses information from beat t to predict information about beat $t+1$, these estimates are called single beat predictions. To compute error rates, we again compare the binned single beat-predictions with the binned actual values.

Sequential Memory Size Variation

To test the effects of finite memory size on the Bayesian Network's estimation accuracy, I sequentially trained the Bayesian Network model four times using finite memory lengths of 100, 1000, 5000, and 8000 heart beats. These memory lengths approximately correspond to time intervals of one minute, ten minutes, fifty-five min-

utes, and an hour and a half, respectively. Each time, the network was first reinitialized and then sequentially trained and tested using the Patient 11007 data. The results are shown in Figure 4-8.

With a memory size of 100 beats, the estimates become noisy. They tend to spike to new values whenever the HR or BP presented to the network changes. As the memory size increases, this type of behavior occurs less frequently. Spikes occur when one of the node distributions returns to its initial uniform distribution value. This situation is described in more detail in Section 5.2.1.

If the data remains within the same set of quantization bins for long enough, the estimates settle to the appropriate quantized values. This makes intuitive sense, because the probability distributions can only distinguish between this set of quantized values. When the MMSE estimate takes an unquantized value, it indicates that more than one quantization bin has been assigned a non-negligible probability. When a large percentage of the memory window contains identical data points, however, the Bayesian Network assigns a high probability to the corresponding quantization bins, causing the MMSE estimates to approach quantized values. When the memory size is small, the MMSE estimates move quickly from one quantized value to another, but as the memory size grows larger, the estimates take unquantized values for longer. For memory sizes of 5000 and 8000 points, this type of behavior enables the network to accurately track intermediate values when the data fluctuates between different quantization bins. This type of estimation can be seen in the SV plots of Figure 4-8c and Figure 4-8d at minute 1000. Here, the quantized SV fluctuates between values, but the MMSE estimate closely approximates the actual SV waveform.

As the memory size increases, the estimates respond more slowly to abrupt changes in the patient data. Just after minute 1200, as seen in Figure 4-6, SV increases abruptly without a corresponding change in HR or BP. Because they move quickly from one quantized value to another, the networks with memory sizes of 100 and 1000 respond quickly to this abrupt change, but the other networks take longer to respond.

The sudden increase in SV at minute 1200 shows how the network responds to changes in the estimated variables that are not accompanied by changes in HR or

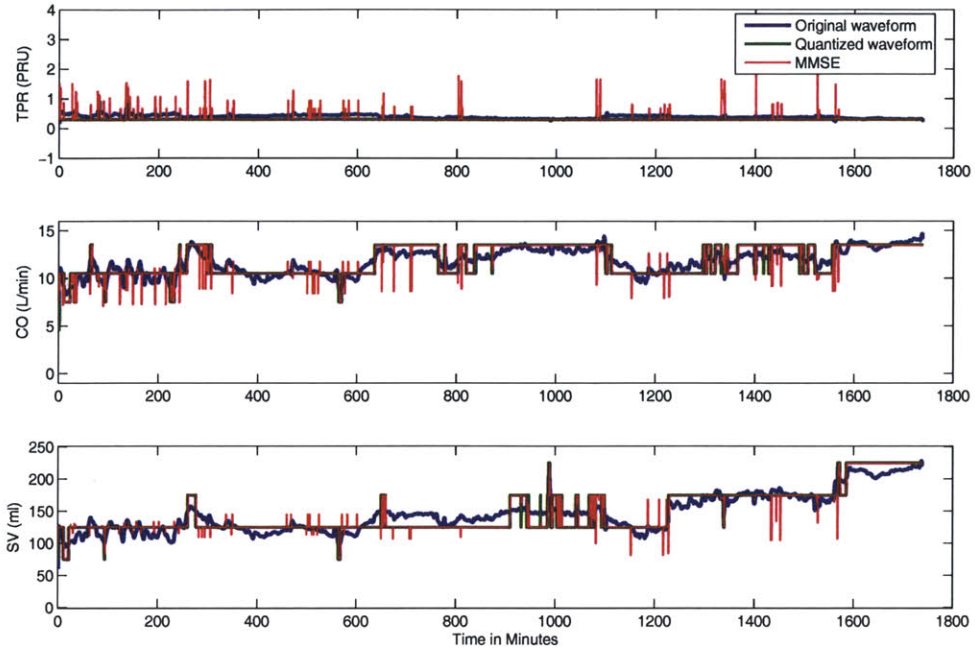


Figure 4-8a

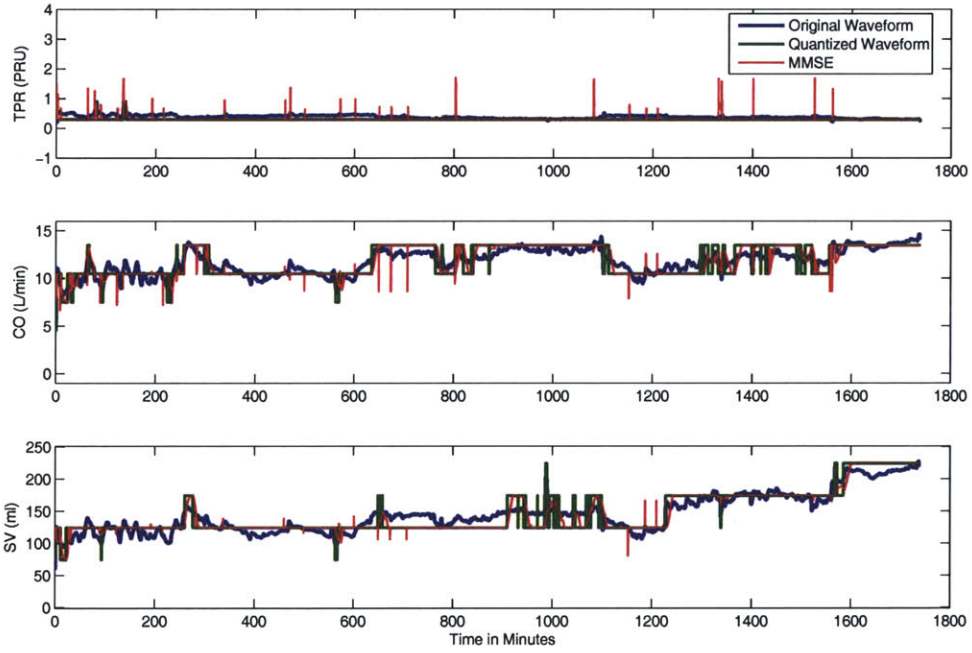


Figure 4-8b

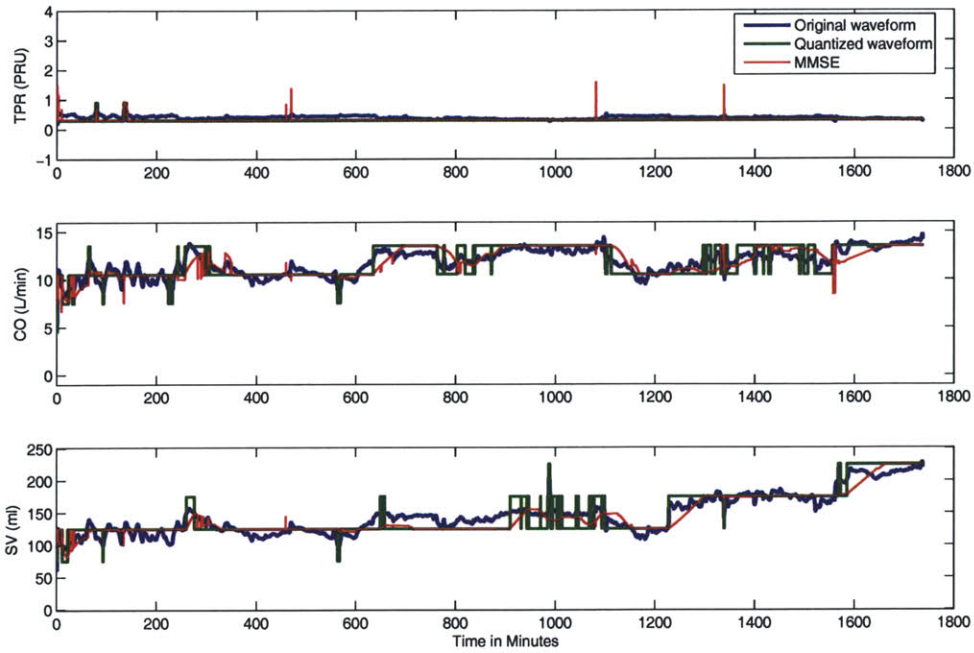


Figure 4-8c

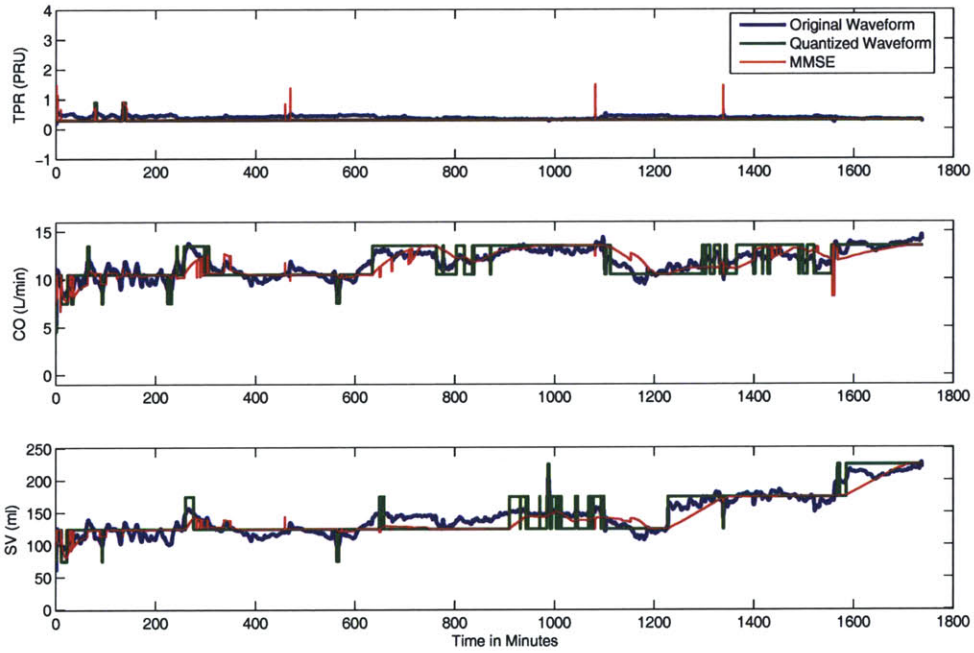


Figure 4-8d

Figure 4-8: Estimates when the Bayesian Network is sequentially trained with various finite memory sizes and tested on the Patient 11007 data. In a, b, c, and d, the memory sizes are 100, 1000, 5000, and 8000 beats, respectively.

Memory Size	100	1000	5000	8000
Resistance	0.21%	0.30%	0.29%	0.26%
Cardiac Output	1.96%	8.32%	18.00%	21.04%
Stroke Volume	1.53%	5.34%	12.55%	13.10%

Table 4.3: Absolute error rates for the MMSE estimates when the network is sequentially trained on Patient 11007 data using finite memory sizes of 100, 1000, 5000, and 8000 beats.

BP, the variables available to the network during estimation. Without some type of indication from either HR or BP that something is changing, the network cannot predict a sudden change in SV. Thus, the network adapts to the change in SV in a delayed fashion as the network updates its probability distributions based upon the new SV data. Because of the change in SV after minute 1200, the network successfully tracks a slight increase in the CO waveform, even though the quantized CO, HR, BP, and TPR values do not change. Here, the network relies on the changing probabilistic information about SV to change the CO estimate.

Within the Patient 11007 data set, changes in SV and CO often occur without corresponding changes in BP and HR. Thus for this data set, the network frequently relies on information that the probability distributions extract from recent SV and CO trends to estimate future CO and SV values.

Table 4.3 shows the absolute error rates obtained by quantizing the MMSE estimates and comparing the quantized estimates to the quantized waveform. The absolute error rates measure the number of times the quantized waveforms do not equal the quantized estimates, divided by the total number of samples in the waveform. The error rates in Table 4.3 are expressed as percentages.

The error rates obtained using sequential learning are better than those obtained using the simulated data analyzed in Section 4.2.5. Thus, according to this metric, the sequentially trained networks perform better than anticipated.

Table 4.3 shows that the absolute error rates increase as the memory sizes increase. This reflects the fact that the estimates move quickly from one quantized value to

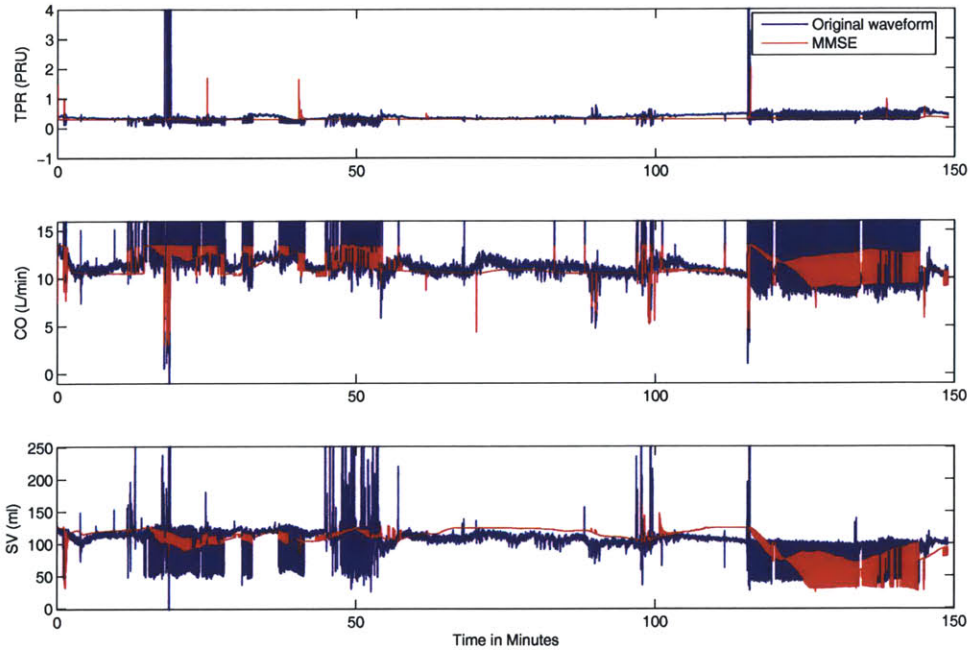


Figure 4-9: Comparison of the actual non-median filtered Patient 6696 waveforms with the MMSE estimated waveforms obtained by first sequentially training the Bayesian Network on the Gold Standard Data and then sequentially training and testing the network on Patient 6696 data with a history size of 1000 data points.

the next when the memory size is small. Since this type of behavior is not necessarily desirable, comparing quantized estimates to the quantized waveform does not prove to be a satisfactory means of evaluating network performance. This issue is addressed in the Section 4.3.4.

Robustness to Noise

To test how the network, training, and inference algorithms would respond to noise, I tested a sequentially trained network with a memory size of 1000 points using a non-median filtered version of data from Patient 6696, another ICU patient. Overall, the network performed much better without noise, but the MMSE estimates provided some robustness against noise. As seen in Figure 4-9, which compares the actual TPR, CO, and BP waveforms with the MMSE estimates obtained using the sequentially trained network, the MMSE estimates frequently produce a trend that

follows the actual CO, TPR, and SV waveforms. The MAP estimates, which are not pictured, perform far worse because they can only take quantized values. When the noise extends across quantization boundaries, the MAP estimates flip back and forth between adjacent quantized values, creating noisy estimated trends, while the MMSE estimates frequently produces a less noisy version that balances the two quantized extremes. When the noise lasts for a short period of time, a period less than a window length, the estimates appear to dampen the noise slightly.

4.3.4 Multiple Beat Prediction using Sequential Training

In Section 4.3.3, we performed single beat predictions, which used the probability distributions obtained by sequentially training on first t time steps to predict patient parameters at time step $t+1$. In a hospital setting, however, it seems likely that the network would need to use its current probability distributions to estimate patient parameters five to ten minutes in the future. Thus, this section explores the network's ability to use probability distributions obtained by sequentially training on first t time steps to estimate CO, TPR, and SV at time step $t+n$, given the HR and BP measurements at time step $t+n$. Since the network uses distributions from beat t to predict information about beat $t+n$, these estimates are called n-beat predictions.

Previously, we quantitatively evaluated network performance by comparing quantized estimates to quantized versions of the actual waveforms. As mentioned in Section 4.3.3, this type of metric favors estimates that move quickly from one quantized value to another. As a better metric of network performance, we can compare the network's estimates to estimates obtained using a naive approach that requires minimal computation. In this section, we evaluate networks using such a metric. In this case, we use the last available observation as the naive estimate. In other words, we use the CO, TPR, and SV observation from time step t as the naive estimates for time step $t+n$ when performing n-step prediction. We then compare this naive estimate this the n-step estimate produced by the Bayesian Network and note the percent of the time that the Bayesian Network estimate is closer to the actual waveform than the naive estimate. In other words, we evaluate how frequently the Bayesian Network

Finite Memory Size	Prediction Gap Size (beats)	Mean Frequency with which Network Outperformed Naive Approach (%)
1000	1	0.1
1000	416	20.4
1000	831	27.6
5000	416	22.3
5000	831	30.9
5000	1661	38.2
8000	416	23.2
8000	831	31.2
8000	1661	36.9

Table 4.4: Average percent of the time that the Bayesian Network estimates were more accurate than the naive estimates for CO and SV.

produces more accurate estimates than the naive approach.

To test the network's ability to perform n-beat predictions, I first calculated single-beat, 416-beat (five minute), and 831-beat (ten minute) predictions using Bayesian Networks that were sequentially trained using finite memory sizes of 1000 beats. I then computed 416-beat (five minute), 831-beat (ten minute), and 1661-beat (twenty minute) predictions using sequentially trained Bayesian Networks with 5000 and 8000-beat finite memories. Each time, the network was first reinitialized and then sequentially trained and tested using the Patient 11007 data. The estimates were then compared with estimated obtained using the naive approach mentioned above.

Since TPR remains essentially constant, the naive approach estimates for TPR were extremely accurate throughout all trials. On average, the Bayesian Network produced a superior estimate only 7.3% of the time. Because the trends for CO and SV contain similar transients, the network estimated these variables with similar levels of accuracy. Taking the performance rate as the percent of the time that the MMSE estimates were closer to the actual waveforms than naive estimates, the performance rates for CO and SV were extremely similar. Due to their similarity, these rates were averaged to obtain a mean performance rate for each trial. The mean CO-SV performance rates are shown in Table 4.4.

In the single-beat case, the naive approach produced more accurate estimates

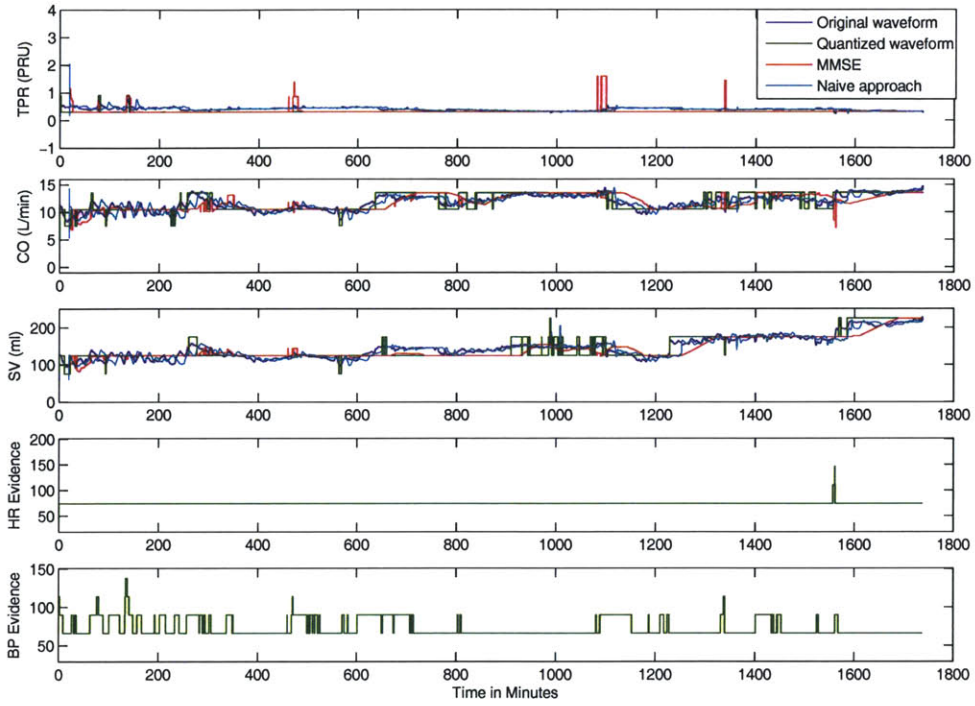


Figure 4-10: 1661-beat MMSE and naive estimates obtained when the network is trained using a sequential memory size of 5000 beats. The bottom two plots show the HR and BP evidence presented to the network during training and inference.

nearly one hundred percent of the time. In general, the naive approach outperformed the Bayesian Network approach, but as the number of prediction beats increased, the network performance improved. When the time gap was small, the naive approach performed quite well because the data had not had very much time to change. As the prediction gap became larger, however, the last observed data point was more likely to significantly differ from the current actual value. Thus, as the prediction gap increased, the naive took longer to adapt to changes in the patient data. As the prediction gap increased, the Bayesian Network also took longer to adapt to changes in patient data, but its adaption delays grew more slowly than those associated with the naive approach. The naive approach delays always grew linearly with n, while the Bayesian Network delays sometimes became shorter. Specifically, when the network relied on previously stored information to perform its prediction, its time delays did not grow as quickly.

Figure 4-10 shows results from the trial in which the MMSE estimates exhibited the best performance. During this trial, the network was sequential trained using a memory size of 5000 beats and produced 1661-beat (20 minute) predictions. The MMSE estimates produced using 1661-beat prediction are qualitatively similar to those produced using smaller prediction gaps. The majority of the improvement in relative Bayesian Network performance as the prediction gap increased is due to degraded naive approach accuracy.

4.3.5 Combinational Learning

Previously, we displayed results obtained when a sequentially trained network was used to estimate values of CO, SV, and TPR that occurred twenty minutes after the final training data point occurred. In this case, the network probability distributions were trained using only data from the current patient. If the network had access to information from other patients, perhaps this additional knowledge would help the network produce better estimates. This type of approach, which we call combinational learning, creates a network with a probability distribution that relies on both persistent and transient memories (see Section 3.3).

Combinational learning is accomplished by first batch training the network on previously available data, and then sequentially training and testing the network on incoming patient data. In this section, we first batch trained the Bayesian Network model using the Gold Standard Data. We then sequentially trained and tested the network on the Patient 11007 data using a finite memory size of 5000 beats. This created a persistent memory containing information from the 1351 beats of the Gold Standard Data set and a transient memory containing data from 5000 beats of recent Patient 11007 history. Due to the relative memory sizes, the incoming patient data was weighted more heavily.

The results shown in Figure 4-11 reveal that batch training the network before performing sequential training caused the network performance to degrade. Comparing these results with those seen in Figure 4-10, the network produced better estimates of CO and SV when it relied only on current patient data. Comparing

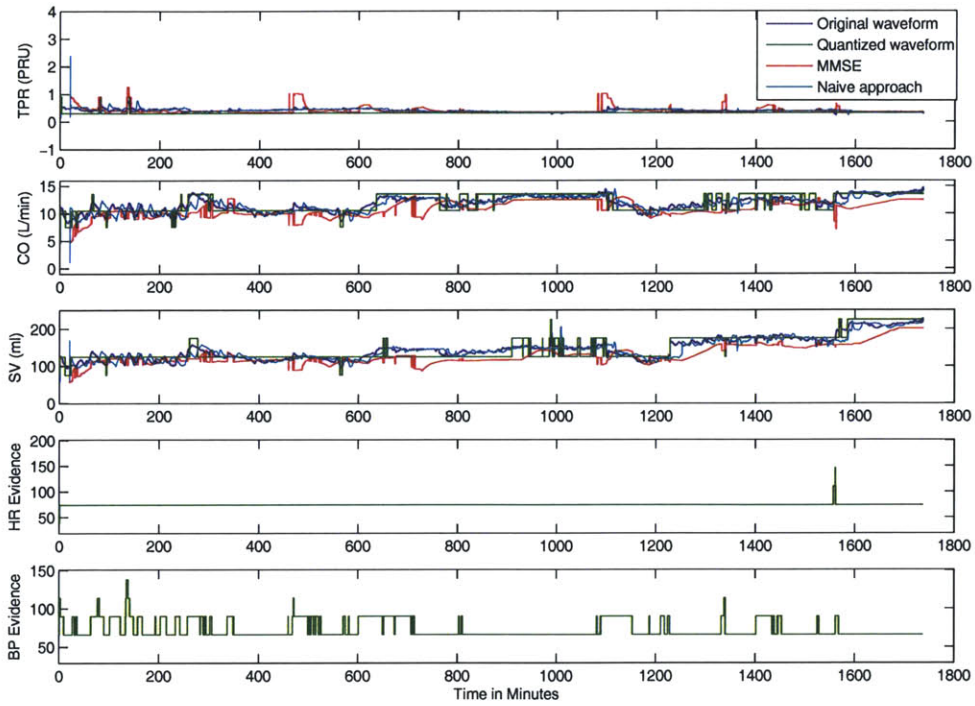


Figure 4-11: 1661-beat MMSE and naive estimates obtained when the network is first batch trained on the Gold Standard Data and then sequentially trained on the Patient 11007 data using a finite memory size of 5000 beats. The bottom two plots show the HR and BP evidence presented to the network during training and inference.

the Gold Standard Data with the Patient 11007 data, this is not surprising because the Patient 11007 does not behave in a manner consistent with the Gold Standard Data set. Specifically, Gold Standard Data CO and SV values typically remained low, while those in the Patient 11007 trends took higher values.

Due to the batch training, the network's ability to estimate TPR improved somewhat arbitrarily. Essentially, the Gold Standard Data added some variability to the TPR distribution so that the TPR estimate did not equal a quantized value as frequently. This enabled it to come closer to the actual TPR value more often than it did when the network was only sequentially trained.

Although using information from other patients did not improve estimation capabilities in this example, using data from additional patients probably would help in other instances. Specifically, previous patient information would probably improve estimates if previous patients shared response characteristics with the current patient.

Chapter 5

Discussion and Future Work

5.1 Discussion

The Bayesian Network can successfully track TPR, CO, and SV by sequentially learning from patient data. The network creates estimates that fall within the same quantization range as the actual waveforms eighty to one hundred percent of the time. Despite this capability, the naive approach of taking the most recently observed point as the current estimate outperforms the Bayesian Network approach sixty-three to eighty percent of the time when there is a gap between training and estimation and nearly one hundred percent of the time when there is no gap. Since the naive approach performs well when the data is stationary, but performs poorly when the data changes abruptly, future work must enhance the Bayesian Network's ability to predict abrupt changes in patient state and to predict how an abrupt change in one variable will affect the other variables' values. Overall, the Bayesian Network's strengths lie in its ability to consider all network variables at once and learn probabilistic relationships between variables. This ability to synthesize information must be exploited to enhance future performance.

5.2 Future Work

This section discusses future research directions, exploring possible ways to improve the network's performance and enhance its abilities to detect and respond to patient data.

5.2.1 Further Analysis of the Current Network Model and Algorithms

In this subsection, I outline unresolved issues encountered while working with the current Bayesian Network model. I discuss algorithmic improvements and suggest ways to enhance our understanding of how our network model responds to real patient data.

Sequential Algorithm Revision

As explained in Section 3.3, the sequential learning algorithm with an n-point memory size uses information from both initial probability distributions and recent patient history to set the current probability distributions. The algorithm sets the current distribution probabilities based on normalized histograms of the n most recent points relevant to the probability distributions of interest plus the relevant initial Dirichlet counts. During the trials examined in this section, the probability distributions were initialized using small uniform Dirichlet counts whose influence quickly becomes overwhelmed by the incoming data. Under certain conditions, however, the initial Dirichlet counts become significant.

As shown in Figure 4-8a, changes in the quantized BP or HR values sometimes cause momentary noise in the estimated CO, SV, and TPR waveforms. Essentially, changes in the random variables presented as evidence to the network can abruptly change the estimated values obtained via Bayesian inference. This type of noise occurs more frequently when the window size is small, but also occurs when the data used to train the network remains constant for long periods of time. This behavior seems to emerge when the data remains constant for a long time relative to

the size of the sequential algorithm's sliding memory window. When the network is initialized differently, the shapes of the noise spikes change, indicating that the noise spikes coincide with times when the network relies heavily on initial distributions to compute its estimates. This occurs when the histograms computed from recent patient history contain no information about a particular combination of HR-BP-SV-CO-TPR values.

Further analysis reveals several problems with the current algorithm. First, the learning algorithm does not maintain an equivalent sample size. To see this, note that the HR, SV, and TPR probability distributions depend on the n most recent data points. CO and BP, however, have twenty-five conditional probability distributions, one conditional distribution per set of parent-node values, and each conditional distribution depends on the most recent n data points that co-occur with the correct set of parent-node values. This means that the HR, SV, and TPR distributions depend on n data points, while the CO and BP distributions depend on $25 \times n$ data points. Since the conditional node distributions do not all depend on the same number of points, the *a posteriori* distributions calculated using the network may have undesirable characteristics. Future work should address this issue by analyzing the effects of the non-equivalent sample sizes and if necessary, developing a sequential learning scheme that maintains an equivalent sample size.

In addition to drawing probability distributions from a non-equivalent sample size, the current algorithm allows the probability distributions to become completely overrun by stationary data. If for instance, HR falls into quantization bin number 3 so that $\mathrm{HR_b} = 3$ for more than n successive points, the algorithm changes the heart rate PMF to a distribution where the probability assigned to $\mathrm{HR_b} = 3$ is close to one and the other probabilities are close to zero. Thus, the PMF learns that HR almost always takes the same value, which is not true in general. This problem could be partially addressed through combinational learning, i.e., by batch training the network on *relevant* data before beginning the sequential training. Then, the network would have persistent memory that would significantly affect the sequentially learned portion of the distributions. In effect, the batch training would provide persistent memory of

long-term HR variability, while sequential training would increase the probability assigned to recently observed values. This would hopefully keep the network from memorizing stationary data. To address this problem in another way, one could create an algorithm that would not learn from constant patient data for more than a specified period of time. Such an algorithm would be more selective when choosing the patient data points used to set the probability distributions, so that the probability distributions do not simply learn constant patient values. Then when the patient data does change, the network will retain memory of other past value combinations so that the network will not have to relearn old data. This type of approach is discussed in [15], where the Bayesian Network probability distributions are updated using a table of values carefully selected from the incoming data. Since not all incoming data was added to the table, the probability distributions did not simply memorize current data values.

Experimentation with Model Parameters

Further analysis should be done to determine how the number of quantization levels assigned to node variables affects the Bayesian Network's performance. Changing the valid quantized values as well as incorporating non-uniform quantization ranges could affect network performance as well. Adding additional quantization levels might allow for more accurate estimates, but might also make it difficult to obtain enough data to confidently set all of the probability distribution parameters.

More work should be done to determine how different combinations of batch and sequential learning affect the network's prediction capabilities. By using combinational learning, or different combinations of batch and sequential learning, one can vary the size and relative influence of data stored in persistent and transient memories. Varying the memory sizes both together and relative to one another should vary how quickly the network adapts to new information and how much it relies on general population statistics. It would be useful to explore how these memory size relationships affect network performance.

In addition to varying memory sizes, training data can be weighted based on its

apparent relevance. During sequential learning, one could add exponentially decaying weights to the incoming data so that recent data is weighted more heavily than the distant past. As in [1], the persistent memory and transient memory could be assigned weights that vary based on the current data trends. Perhaps in steady state, the transient memory containing recent patient history could be weighted more heavily, while the persistent memory containing variability information could be weighted more heavily when the patient data changes abruptly. In the future, one could explore whether methodologies of this type would enhance our Bayesian Network's performance.

Explicitly Modeling Inter-patient Dependencies

Berzuini et al. [1] explicitly define a probabilistic model that describes how patient parameters depend on other variables. They assume that each set of patient parameters is drawn from the same distribution when patients behave in a similar manner. When patients appear to fall into different classes, each class of patients has its own patient parameter distribution. Patients within a given class are assumed to draw their parameters from the same distribution. The parameter model presented in [1] explicitly delineates how patient parameters relate to patient observations as well. Designing a model that describes how we believe the current patient parameters relate to observed data and to other patient parameters would help guide our algorithm design. Explicitly modeling how patient parameters change with time would guide future work as well.

Probability Distribution Analysis

To gain additional insight into how the probability distributions adapt to current patient data, it would be useful to examine how the network probability distributions change with time in response to the training data. It would also be useful to analyze how significantly each conditional node distribution affects the *a posteriori* distributions and estimation outputs.

Reevaluation of Estimation Goals

Further thought must be put into determining exactly what we would like the network to learn and predict. It appears that when patient variable values remain constant, the naive approach of using the most recently observed value as the next estimate works extremely well. On the other hand, when the patient data changes abruptly, the naive approach obviously fails. In addition, as the network learns from stationary data, it forgets information about value variability, but does not necessarily outperform the naive approach due to the quantized nature of the network. A superior scheme might use the naive approach when patient data remains stationary and use the network to predict new values when the patient data does change abruptly. An alternative scheme might use two networks of varying resolutions, one low-resolution network to predict responses to abrupt patient changes and another high-resolution network to perform estimation when the patient data remains stationary. In such a scheme, quantization levels within the low-resolution network could cover the entire dynamic range of interest, while high-resolution quantization levels would cover only a small dynamic range containing the stationary patient data observed most recently. Regardless, explicitly stating which type of network performance we desire would help direct future work.

5.2.2 Real Patient Data

As seen in Figure 4-7, information contained in certain patient data training sets may be irrelevant to the patient of interest. Thus, when creating persistent memory within the Bayesian Network model, one must ensure that the data used is relevant to the current patient. Thus, it seems desirable to have several different patient databases to chose from when initializing the persistent memory, perhaps one database for each type of common cardiovascular disorder. A specific database would then correspond to a particular disorder and would contain data from patients who shared a similar medical history. Then, if the current patient was diagnosed with disorder A, the Bayesian Network's probability distributions could be initialized using data from other

patients with the same disorder. Similarly, this approach would keep distributions from being contaminated by data from patients with very different medical histories. If enough patient data can be gathered it would be useful to determine whether this type of approach would yield more accurate Bayesian estimates.

Learning with Missing Data

The current algorithms assume complete training data sets. This means that each training data point presented to the network contains values for all of the node variables. When performing sequential learning in a real patient setting, however, complete data sets will not always be available. The EM algorithm, presented in [9], performs Bayesian learning when portions of the training data set are missing. Currently, the sequential algorithm cannot learn from an incomplete data set, so this algorithm should be extended to include the EM capabilities.

5.2.3 Expansion of the Current Model

Dynamic Bayesian Models

While the current network model can adapt to changes in patient data, it cannot learn time dependencies within patient data trends. Thus, it can sense the current HR value, but it cannot identify whether HR is increasing, decreasing, or remaining constant. Dynamic Bayesian Networks, however, learn from both current variable values and from the relationships between variable values adjacent in time. A Dynamic Bayesian Network expansion might provide more accurate estimates when the data changes abruptly since Dynamic Bayesian Networks store information about how the node variables respond to increasing or decreasing patient parameters.

Adding Additional Network Nodes

By adding new network nodes, the Bayesian Network model could be expanded to incorporate information about medications, lab reports, nurses notes, diagnoses, and medical treatments. Including such information could help the network better pre-

dict changes in patient state. Incorporating these additional nodes into a Dynamic Bayesian Network structure might also allow the network to take into account treatment administration time delays. Through intelligent network design, this additional information might help the network more accurately predict abrupt changes in patient state and produce more accurate estimates of patient parameters.

When adding new network nodes, structural learning algorithms [10] can be used to learn underlying probabilistic dependencies from the data. This approach ensures that the probabilistic dependencies implied by the network are present in the patient data used to create the network structure, giving the network structure an additional degree of credibility. The structural algorithms also set the probability parameters based on the patient training data, ensuring that the probability distributions can be learned from patient data as well.

5.3 Conclusion

While the Bayesian Network model shows promise, much remains to be done to enhance its performance. Chiefly, further thought must be put into the learning algorithms' design and intent. In addition, the network must learn to respond more robustly to abrupt changes in patient data, through the use of better training data sets, Dynamic Bayesian models, improved training algorithms, expanded Bayesian Network designs, or some combination there of. Future work should explore whether changes like those discussed within this chapter would in fact enhance network performance.

Appendix A

Alternate Bayesian Network Model which Incorporates Treatment Information

Initially, we hoped to use a Bayesian Network model to capture the effects of medication and fluid levels on vital signs and internal patient parameters. We wanted to develop a way of incorporating treatment information into a Windkessel model by using estimates obtained from the Bayesian Network to set Windkessel model parameters.

With the help of medical experts, I developed the Bayesian Network model pictured in Figure A-1, where *Levophed* is a medication, *HR* denotes heart rate, *Net Volume Inflow* denotes the net change in the total blood volume, *BP* is mean arterial blood pressure, *CO* is cardiac output, and *TPR* is total peripheral resistance.

I then used the patient data pictured in Figure A-2 to initialize the network parameters. For test purposes, I allowed each random variable to take three values: low, mid-range, and high. I then discretized the patient data based on the thresholds indicated in Figure A-2.

After using this data to set the parameter values, I arrived at the conditional probability mass functions partially pictured in Figure A-3. I then performed inference on the network by giving the network different values for Levophed and Net Volume

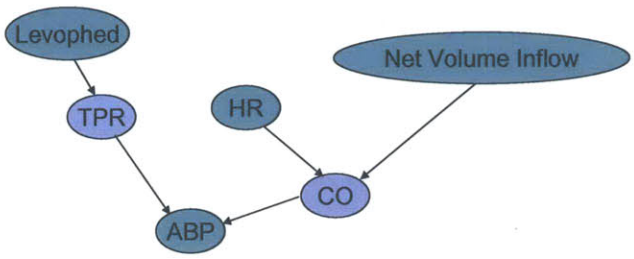


Figure A-1: A simple model of the cardiovascular system

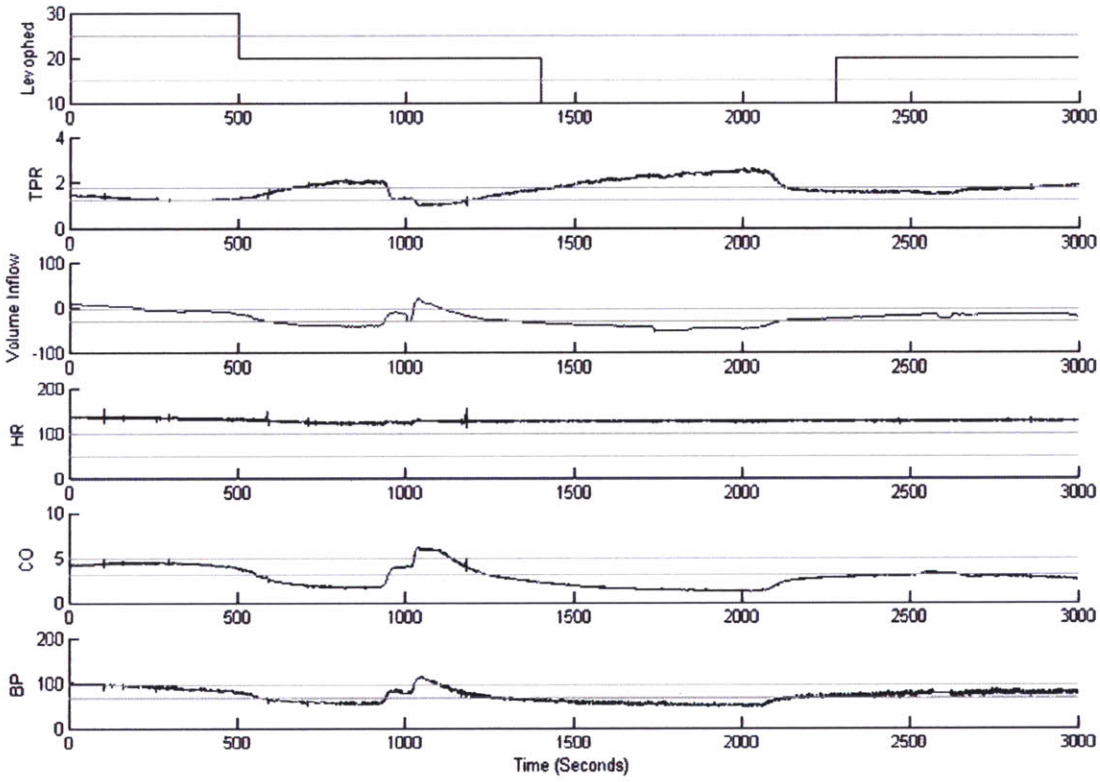


Figure A-2: Patient data used to initialize the network parameters

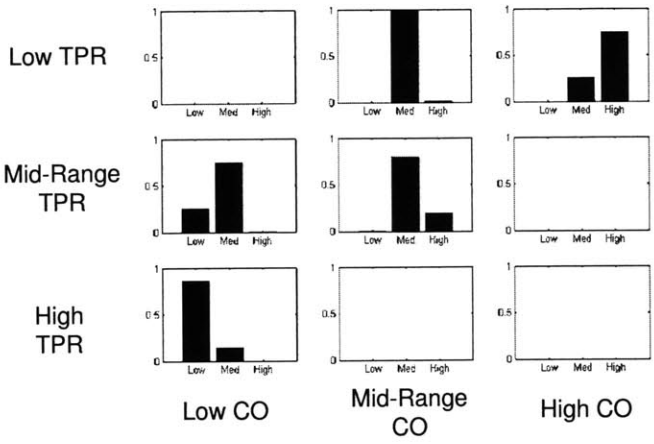


Figure A-3: Conditional PMF from a trained network: Conditional PMF of BP given TPR and CO

Inflow and computing the marginal distributions for blood pressure. The resulting marginal distributions are pictured in Figure A-4.

Although the distributions pictured in Figures A-3 and A-4 are plausible, overall the distributions were not. Specifically, Levophed, a peripheral vasoconstrictor, is supposed to constrict veins and arteries, and thus increase TPR. Looking at the training data in Figure A-2, TPR reaches its peak value when the patient is no longer receiving Levophed. This discrepancy caused the network to learn counterintuitive probability distributions.

In the real patient database available to us, the medication administration times are only accurate to within a twenty or thirty minute window. When the other available information fluctuates on a much faster time scale, this level of temporal accuracy is not acceptable. While the Bayesian Network in Figure A-1 may yield useful results when initialized correctly, we would need to obtain a much larger, more comprehensive data set with accurate medication administration times to test the validity and estimation capabilities of this model.

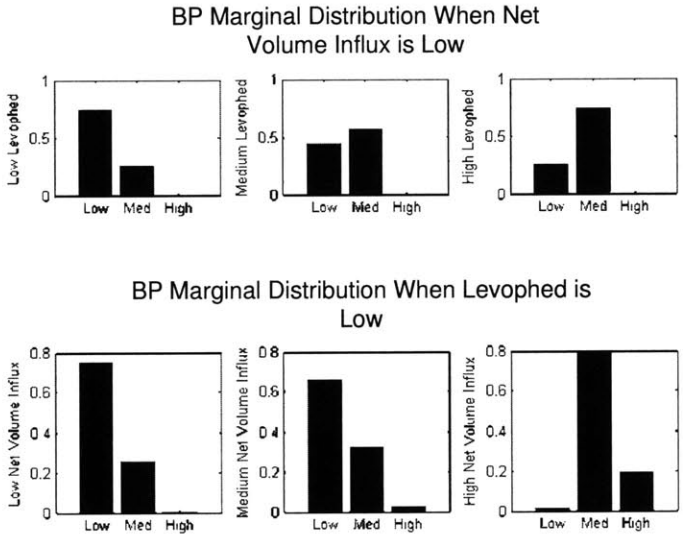


Figure A-4: Network results

Appendix B

Simulation Parameters and Bayesian Network Quantization Values

Parameter	Value in pulsatile model
R_1	0.01 mmHg/(ml/s)
R_2	0.03 mmHg/(ml/s)
R_3	1 mmHg/(ml/s)
C_a	2 mmHg/ml
C_v	100 mmHg/ml
Q_v	0 ml/s
C_i	300 ml/s
R_i	1 mmHg/(ml/s)
E_d	0.1 ml/mmHg
E_s	2.5 ml/mmHg
$V_a(0)$	91.2281 mm Hg
$V_v(0)$	15.0337 mm Hg
$Q_h(0)$	127.383 mm Hg
T	1 s

Table B.1: Nominal parameters used to create the simulated data.

Bin Number	Quantized Value for				
	TPR	SV	HR	CO	BP
1	0.5	25	25	2	30
2	1.5	75	75	6	90
3	2.5	125	125	10	150
4	3.5	175	175	14	210
5	4.5	225	225	18	270

Table B.2: Bin numbers and corresponding quantization values used when training the Bayesian Network on simulated patient data.

Bin Number	Quantized Value for				
	TPR	SV	HR	CO	BP
1	0.3	25	38	1.5	42
2	0.9	75	74	4.5	66
3	1.5	125	110	7.5	90
4	2.1	175	146	10.5	114
5	2.7	225	182	13.5	138

Table B.3: Bin numbers and corresponding quantization values used when training the Bayesian Network on real patient data.

Bibliography

[1] C. Berzuini, R. Bellazzi, S. Quaglini, and D. J. Spiegelhalter. Bayesian networks for patient monitoring. *Artificial Intelligence in Medicine*, 4:243–260, May 1992.

[2] J. J. S. Chen, T. Heldt, G. C. Verghese, and R. G. Mark. Analytical solution to a simplified circulatory model using piecewise linear elastance function. *Computers in Cardiology*, 30:45–48, 2003.

[3] T. L. Davis. Teaching physiology through interactive simulation of hemodynamics. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1991.

[4] T. L. Davis and R. G. Mark. Teaching physiology through interactive simulation of hemodynamics. In *Proc. of the IEEE Computers in Cardiology*, pages 649–652, September 1990.

[5] N. Friedman and M. Goldszmidt. Sequential update of Bayesian network structure. In D. Geiger and P. Shanoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 165–174, 1997.

[6] T. Heldt, E. B. Shim, R. D. Kamm, and R. G. Mark. Computational model of cardiovascular function during orthostatic stress. *Computers in Cardiology*, 27:777–780, 2000.

[7] G. Liljestrand and E. Zander. Vergleichende bestimmungen des minutenvolumens des herzens beim menschen mittels der stickoxydulmethode und durch blutdruckmessung. *Z Ges Exp Med*, 59:105–122, 1928.

[8] W. Long. Temporal reasoning for diagnosis in a causal probabilistic knowledge base. *Artificial Intelligence in Medicine*, 8:193–215, 1996.

[9] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* John Wiley and Sons, New York, 1997.

[10] R. E. Neapolitan. *Learning Bayesian Networks.* Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

[11] O. Ogunyemi. Methods for reasoning from geometry about anatomic structures injured by penetrating trauma. *Journal of Biomedical Informatics*, (in press).

[12] T. Parlikar and G. C. Verghese. A simple cycle-averaged model of cardiovascular dynamics. In *Proceedings of the 35th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, September 2005.

[13] G. Rutledge, G. Thomsen, I. Beinlich, B. Farr, B. Sheiner, and L. Fagan. Combining qualitative and quantitative computation in a ventilator therapy planner. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 315–319, Washington, D.C., 1989. The American Medical Informatics Association.

[14] Z. Samar, T. Heldt, G. C. Verghese, and R. G. Mark. Model-based cardiovascular parameter estimation in the intensive care unit. *Computers in Cardiology*, 32:635–638, 2005.

[15] K. Torabi, S. Sayad, and S. T. Balke. On-line adaptive bayesian classification for in-line particle image monitoring in polymer film manufacturing. *Computers and Chemical Engineering*, 30:18–27, 2005.

[16] A. Tucker, V. Vinciotti, X. Lui, and D. Garway-Heath. A spatio-temporal bayesian network classifier for understanding visual field deterioration. *Artificial Intelligence in Medicine*, 34:163–177, 2005.

[17] L. C. van der Gaag. Bayesian Belief Networks: Odds and Ends. Technical report, Utrecht University, Department of Computer Science.