Probabilistic Network Models for Cardiovascular Monitoring

by

Shirley X. Li

B.S., Electrical Engineering
Massachusetts Institute of Technology (2006)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007
[June 2007]

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2007

Certified by .....
George C. 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

Probabilistic Network Models for Cardiovascular Monitoring

by

Shirley X. Li

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

While treating patients during their hospital stay, physicians must frequently take into consideration massive amounts of clinical data. This data can come in many forms, such as continuous blood pressure tracings, intermittent laboratory results, or simple qualitative observations on the patient's appearance. Although access to such a rich collection of information is beneficial for making diagnoses and treatment decisions, it can sometimes be difficult for clinicians to mentally keep track of everything, especially in hectic environments such as hospital intensive care units (ICUs). In addition, there are certain physiological variables that cannot be measured noninvasively, but are critical indicators of a patient's state of health. One such example in cardiology is cardiac output - the mean flow rate of blood from the heart.

In this thesis, we explore probabilistic networks as a method for integrating different types of clinical data into a single model, and as a vehicle for summarizing population statistics from medical databases. These networks can then be used to estimate unobservable variables of interest. We propose and test several networks of varying complexity on both a set of experimental porcine data, and a set of real ICU patient data. We find that continuous estimation of cardiac output is possible using probabilistic networks, and that the errors produced are comparable to those obtained from deterministic methods that employ the same information. Furthermore, since this technique is purely statistical in nature, it can be easily reformulated for applications where deterministic methods do not exist.

Thesis Supervisor: George C. Verghese
Title: Professor of Electrical Engineering

Thesis Supervisor: Thomas Heldt
Title: Postdoctoral Research Associate

Acknowledgments

I would like to express my sincerest gratitude towards Professor George Verghese for giving me the opportunity to work under his supervision, and for teaching me so much along the way. My interactions with you as a student, an advisee, and simply as a member of the Course 6 community have truly been a highlight of my MIT experience.

I would also like to thank Professor Roger Mark for introducing to me the wonders of physiology and medicine. Taking your class was a life-changing experience, and TAing it has been a privilege.

I owe so much to Dr. Thomas Heldt, who has been there for me in every way. Thank you for your guidance as a mentor and your kindness as a friend. You are a gifted teacher, and I hope that many others will have the opportunity to benefit from your dedication, as I have.

To Dr. Tushar Parlikar, this work would not have been possible without your help, and this year would not have been nearly as enjoyable without your friendship. Thank you for your humor, warmth, and infinite patience.

I would also like to acknowledge and thank Said Francis, Lynne Salameh, and Faisal Kashif for their contributions to this thesis, and Professor Tommi Jaakkola for providing me with very helpful resources. Special notes of gratitude to Said for his company during the days and nights that were spent in the reading room, and Gireeja Ranade for providing me with motivation to work and an excuse for mischief.

To all my wonderful friends, you are the reasons why the past five years have been some of the happiest of my life. Special thanks to Amy Tang, who has been with me every step of the way, Christian Deonier, for pushing me when I hesitate, and Joseph Cheng, for sharing with me his intellectual curiosity. And to my special sisters, I've treasured every moment that we've spent together, and I look forward to what the future brings.

Finally, I sincerely thank my parents for their unconditional love and support, and for their foresight in raising me.

This work was supported in part by the National Aeronautics and Space Administration (NASA) through the NASA Cooperative Agreement NCC 9-58 with the National Space Biomedical Research Institute and in part by grant R01 EB001659 from the National Institute of Biomedical Imaging and Bioengineering of the United States Institutes of Health.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

When making diagnosis and treatment decisions on patients during their hospital stay, physicians must frequently take into consideration massive amounts of clinical data. This data can come in many forms, such as continuous blood pressure tracings, intermittent laboratory results, or simple qualitative observations on the patient's appearance. The availability of information is critical to a physician's ability to make informed decisions, and as much of it as possible should be used. However, due to the sheer amount of data that is routinely collected in modern day hospitals, it can sometimes be difficult for clinicians to mentally keep track of everything. In addition, there are certain physiological variables that cannot be measured noninvasively, but yield valuable insight into a patient's state of health.

An important example in cardiology of such a variable is the rate of blood flow from the heart - cardiac output. Cardiac output has well-understood physiological relationships with many routinely recorded signals, such as arterial blood pressure, but it itself cannot be measured without invasive or expensive procedures. Knowledge of cardiac output and other unobservable variables of interest would be especially helpful to physicians in hectic working environments, such as hospital intensive care units (ICUs). Cardiac output is also used in the ICU to assess the criticality of a patient's condition.

The objective of this research project is to develop probabilistic models of the cardiovascular system that can be used to estimate cardiac output and other unobservable variables of interest. We pay particular attention to solidifying the methodology behind using prob-

abilistic networks, and establishing a modeling framework that can be easily extended to encompass more complex problems. We validate our models using experimental porcine data and actual ICU patient data.

1.2 Motivation for Probabilistic Networks

An important class of cardiovascular models is based on lumped-parameter representations of the hemodynamic system, where physical concepts such as vessel resistance and vascular distensibility are modeled by electrical circuit elements such as resistors and capacitors, respectively. These models can be designed at a level of detail appropriate for the application at hand and therefore track the dynamics of the system in corresponding detail.

However, this class of models has two major shortcomings. First, it cannot easily incorporate qualitative or non-continuous data, such as descriptive observations or effects of medication. Such intermittent and qualitative data can contain very important information about the patient and ideally should not be discounted in modeling.

Secondly, circuit models require clean, robust, and very specific signals as inputs. Such measurements are often difficult to obtain in real-time, or are simply unavailable, since vitals signs are extremely noisy and can disappear completely with patient movement or instrumentation error. Furthermore, as the complexity of these models increase, more signals are needed to perform parameter estimation. Thus, the power of circuit models is severely limited by the quality and quantity of available data.

Probabilistic modeling is an alternative framework that could remedy the weaknesses of and serve as a complement to existing circuit models. Using probabilistic distributions to describe relationships between physiological parameters gives us the ability to incorporate qualitative, discrete, and continuous data into a single model, while providing a natural way to handle uncertainty and noise. In addition, with the recent emergence of extensive medical databases that contain commonly encountered pathological conditions, probabilistic networks can be used as a vehicle for summarizing population statistics. Bayesian networks and its variants in particular have established and computationally efficient methods for inference and estimation, providing a convenient theoretical base to build upon.

1.3 Overview of Thesis

This thesis is divided into two main chapters.

Chapter 2 discusses the theory behind probabilistic networks, with focus on Bayesian networks, Hidden Markov models, and Dynamic Bayesian networks. Specific techniques for parameter learning and inference in each of the three types of networks are introduced, as well as general concepts in probability and graph theory.

Chapter 3 explains how probabilistic networks can be used to solve our particular problem of estimating cardiovascular health and patient state. Prior work will be summarized and results will be stated.

The thesis concludes with a summary of the contributions made, and recommendations for future work.

Chapter 2

Probabilistic Network Theory

The probabilistic relationship between multiple random variables is fully characterized by their joint probability distribution (JPD). This is often specified using algebraic expressions such as Gaussian mixtures, or multidimensional arrays if the variables are discrete. The disadvantage of expressing JPDs in this manner is that any independencies existing between variables become buried. In addition, these expressions can quickly become intractable if there are more than just a few variables involved. Probabilistic networks, on the other hand, are graphical representations of JPDs that explicitly indicate the existence of independence relationships, thereby reducing the complexity of the mathematical description.

One important application of probabilistic networks is the estimation of random variables of interest given knowledge of other variables that are probabilistically related. This process is called *inference*, and many algorithms exist that exploit the graphical structure of probabilistic networks to efficiently perform it.

In this chapter, we present the theory behind three closely related types of probabilistic networks: Bayesian networks, Hidden Markov Models (HMMs), and Dynamic Bayesian networks (DBNs). We first focus on the process of obtaining network parameters from a data set, called *parameter learning*, and then turn to methods for performing inference on a learned network. We limit our discussion to discrete random variables, since that is the domain in which our models in Chapter 3 lie.

2.1 Bayesian Networks

Bayesian networks, also called Bayes nets, are directed acyclic graphs (DAGs) whose nodes represent random variables and whose edges indicate probabilistic dependencies. A Bayesian network is defined by its graphical structure and a set of conditional probability distributions (CPDs) on each of its nodes. Together, they determine the network's equivalent JPD.

The applications of Bayesian networks range from modeling signal pathways in gene networks to image recognition in artificial intelligence. In the context of medicine, Bayesian networks are commonly used in building decision support systems to enhance clinical diagnosis and disease classification. For instance, Berzuini et al. [6] proposed a methodology that uses Bayesian networks and a population database to predict the effects of drugs on chronically ill patients.

There are two distinct steps to using a Bayesian network for modeling and estimation. First, its graphical structure and associated CPDs must be determined. Both the structure and CPDs can be learned from data or directly set by "expert" knowledge, meaning that they are derived from one's holistic understanding of the system being modeled. Often times, the network structure will be chosen to reflect the causal relationships between variables in the physical system (e.g., "rain" causes "wet grass"). However, it is important to emphasize that although this is a valid modeling approach, the arrows in a Bayes net denote probabilistic dependence and not causal dependence.

Structural learning from data usually involves choosing a network structure to maximize a particular information criterion; this topic is not explored in this thesis, but more information can be found in Heckerman et al. [7].

As mentioned above, the CPDs of a Bayesian network can be set from expert or prior knowledge, although it is more common to derive them from a data set, or from a combination of data and prior knowledge. Sections 2.1.2 and 2.1.3 explain how this can be done. Note that although Bayesian networks are fundamentally just a method for representing JPDs, it is easier in modeling and inference to directly manipulate the network rather than the JPD itself.

The second step in using a Bayes net is to perform inference over a defined network, given observations, or *evidence*, on a subset of the nodes. Sections 2.1.4 and 2.1.5 describe inference via variable elimination and the junction tree algorithm, respectively.

2.1.1 Structure and Properties of Bayesian Networks

Before diving into the specific properties of Bayesian networks, we first introduce some common terminology regarding generic DAGs. In Figure 2-1:

- X is a *parent* of W and Y.
- W and Y are the *children* of X.
- Z is a *descendant* of X, because there exists a directed path from X to Z.
- X is an *ancestor* of Z.
- An *ancestral set* is a set of nodes that also contains the ancestors of all its elements. For instance, nodes X, Y, and Z form an ancestral set. Nodes X, Y, Z, and W also form an ancestral set.
- X, Y, and Z are the *non-descendants* of W, because there does not exist a directed path from W to X, Y or Z.

In addition, a DAG cannot contain any directed cycles, as its name suggests.

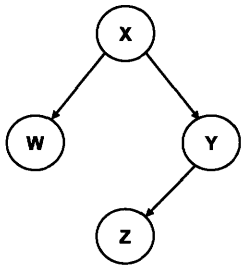

Figure 2-1: A directed acyclic graph (DAG).

The graphical structure of a Bayesian network explicitly indicates the independence between variables in its associated JPD. Nodes that are connected by an edge are possibly, but not necessarily, dependent. For instance, a node is independent of its parent if it has the same CPDs for every outcome of its parent. Thus, a fully connected Bayes net is always a consistent representation of its JPD. In addition, Bayesian networks possess the *Markov condition*, which states that each node is independent of the set of all its non-descendants, given the values of all its parents. Together, these properties restrict the equivalent JPD of

an n-node network to a specific and relatively simple form:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{ the parents of } X_i). \tag{2.1}$$

Thus, a JPD that is represented by a Bayesian network can be factored simply by looking at its graph.

A network's independence relationships are altered by the incorporation of evidence. For instance, Figure 2-2 describes the scenario in which there are two independent fair coin tosses, X_1 and X_2, and a comparison variable X_3. Conditioned on its parents, $P(X_3 = 1|X_1 = X_2) = 1$ and $P(X_3 = 1|X_1 \neq X_2) = 0$. That is, X_3 compares the outcome of the tosses and returns 1 if they are equal and 0 if they are not. Without knowledge of X_3, X_1 and X_2 are clearly independent. However, if the value of X_3 is observed, then we have information relating X_1 and X_2. This is called *induced dependence* [1].

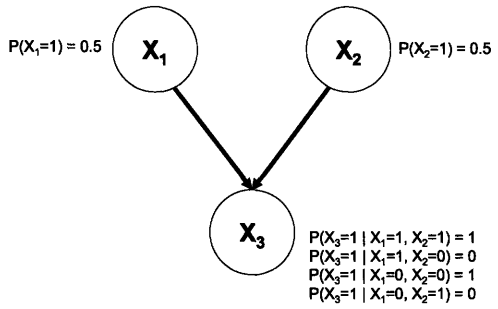

Figure 2-2: Induced dependence in Bayesian networks. Each node is a binary random variable with sample space $\{0, 1\}$. When X_3 is observed, X_1 and X_2 are no longer independent [1].

A method for determining if two sets of nodes in a Bayesian network are independent, conditioned on a third set of nodes is to check for *d-separation.* Before we can define d-separation, we introduce two important concepts in graph theory. First, the *moral* graph of a DAG is defined to be the undirected graph that results from connecting together ("marrying") in the DAG every pair of nodes with a common child, and then dropping the directionality of all the edges. The process of creating a moral graph is called *moralization.* Secondly, a set of nodes $\mathbf{Z}$ is said to *separate* the sets $\mathbf{X}$ and $\mathbf{Y}$ in an undirected graph if and only if a node in $\mathbf{Z}$ is on every path connecting nodes in $\mathbf{X}$ with nodes in $\mathbf{Y}$.

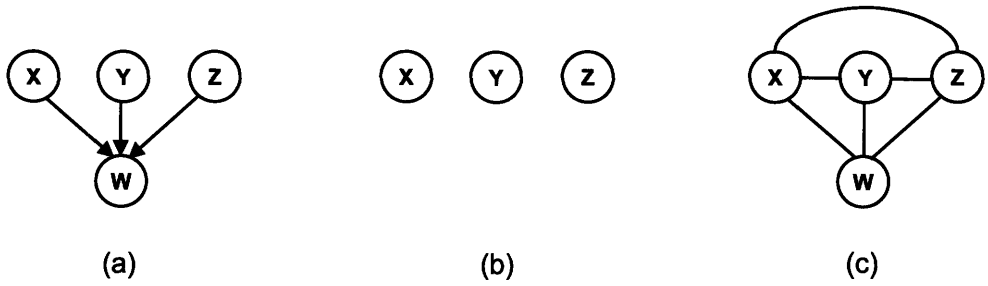

Figure 2-3: Illustration of d-separation. Shown in (b) is the moralized smallest ancestral set containing X, Y, and Z of the Bayes net in (a). Moralizing a larger ancestral set will not give the correct independence statement, as shown in (c).

The definition of d-separation for disjoint sets **X**, **Y**, and **Z** in a Bayesian network is as follows:

> Set **Z** *d-separates* sets **X** and **Y** if and only if **Z** separates **X** and **Y** in the moral graph of the smallest ancestral set containing **X**, **Y**, and **Z**. Furthermore, if **Z** d-separates **X** and **Y**, then **X** and **Y** are independent, conditioned on **Z** [8].

The example in Figure 2-3 shows the importance of only moralizing the *smallest* ancestral set. The smallest ancestral set containing nodes X, Y, and Z of the Bayes net in Figure 2-3(a) is shown in Figure 2-3(b). Notice that this graph is already moral. Since there are no edges between any of the three nodes, X and Y are independent, conditioned on Z, according to the definition of d-separation (they are in fact independent regardless of Z). However, if the entire network (also an ancestral set) is moralized, then node Z no longer separates X and Y.

2.1.2 Concepts of Bayesian Probability and Estimation

There are two schools of thought on the interpretation of probability and estimation: Bayesian and classical. In this section, we briefly discuss the differences between the two in the context of parameter learning, with the primary objective of introducing the Dirichlet distribution.

Suppose we have a biased die with r faces. The outcome of a roll of the die can be modeled as a discrete random variable X with sample space $\{1, \ldots, r\}$. Furthermore, let $\mathbf{D} = [D_1, \ldots, D_r]$ be the number of times our die lands on each face after a sequence of N tosses. Let the bias on the i^{th} face of the die be θ_i and $\mathbf{\Theta} = [\theta_1, \ldots, \theta_{r-1}]$. The θ_i's are called the parameters of X because they specify the probability that the outcome of X will be i: $P(X = i|\mathbf{\Theta}) = \theta_i$. Notice that $\mathbf{\Theta}$ only has $r - 1$ independent elements, since $\theta_r = 1 - \sum_{i=1}^{r-1} \theta_i$. The distribution of X is therefore a probability mass function (PMF) with histograms of heights $[\theta_1 \ldots \theta_r]$. The distribution of $\mathbf{D}$ is then multinomial, which is just the multivariate generalization of the binomial distribution. Recall that the canonical example of a binomial random variable is the number of heads obtained in a fixed number of independent coin tosses. In many modeling problems, $\mathbf{\Theta}$ is unknown and is the target of estimation.

In the classical interpretation of this example, $\mathbf{\Theta}$ is a deterministic but unknown quantity. If $\mathbf{d}$ is a particular outcome of $\mathbf{D}$, then the classical approach estimates $\mathbf{\Theta}$ by its relative frequency of occurrence in a data set:

$$\hat{\theta}_{i_{\mathrm{ML}}} = \frac{d_i}{\sum_{i=1}^{r} d_i}. \tag{2.2}$$

This is called the maximum likelihood (ML) estimate because it maximizes the likelihood function, $\mathfrak{L}(\mathbf{\Theta}) = P(\mathbf{d}|\mathbf{\Theta})$. However, this estimate of $\mathbf{\Theta}$ is just a single number without an associated measure of uncertainty. The uncertainty should be related to the size of the data set; that the estimate does not reflect the amount of data used is a major shortcoming of the classical approach.

In Bayesian estimation, $\mathbf{\Theta}$ is considered to be a random variable and is often assumed to have a Dirichlet distribution (see Figure 2-4) [9]. Thus, we do not have a single number as an estimate for $\mathbf{\Theta}$, but rather a distribution that tells us how likely each outcome of $\mathbf{\Theta}$ is.

The Dirichlet distribution, denoted $Dir(\boldsymbol{\Theta}; \alpha_1, \ldots, \alpha_r)$, has the following form,

$$Dir(\boldsymbol{\Theta}; \alpha_1, \ldots, \alpha_r) = \frac{\Gamma(\sum_{i=1}^{r} \alpha_i)}{\prod_{i=1}^{r} \Gamma(\alpha_i)} \prod_{i=1}^{r} \theta_i^{\alpha_i - 1}, \tag{2.3}$$

where Γ is the gamma function,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt,$$

which, for integers, reduces to

$$\Gamma(z) = (z-1)!.$$

The α_i's are nonnegative real numbers and are called hyperparameters, since they parameterize the distribution of the 'parameters' $\boldsymbol{\Theta}$ (now taken as random variables).

One important reason for using the Dirichlet distribution is that it is the conjugate prior of the multinomial distribution. This means that the *a posteriori* probability of $\boldsymbol{\Theta}$ given the multinomial sample $\mathbf{d}$ remains a Dirichlet distribution with new hyperparameters:

$$P(\boldsymbol{\Theta}|\mathbf{d}) \sim Dir(\boldsymbol{\Theta}; \alpha_1 + d_1, \ldots, \alpha_r + d_r). \tag{2.4}$$

Thus, as new data is observed, the hyperparameters are updated and $\boldsymbol{\Theta}$'s distribution changes (see Figure 2-4). This property forms the basis of how new data is used to update the CPDs of Bayesian networks, as will be explained in Section 2.1.3.

There are two useful point estimators for $\boldsymbol{\Theta}$ that can be derived from its distribution: the maximum a posteriori (MAP) estimate and the minimum mean squared error (MMSE) estimate. The MAP estimator maximizes $P(\boldsymbol{\Theta}|\mathbf{D})$ and is therefore equal to the mode of $P(\boldsymbol{\Theta}|\mathbf{D})$. For a Dirichlet PDF, the mode is given by:

$$\hat{\theta}_{i_{\text{MAP}}} = \frac{\alpha_i - 1}{\left(\sum_{i=1}^{r} \alpha_i\right) - r}. \tag{2.5}$$

Recall from Bayes' rule that $P(\boldsymbol{\Theta}|\mathbf{d}) \propto \mathfrak{L}(\boldsymbol{\Theta}) P(\boldsymbol{\Theta})$. Thus, the ML and MAP estimates are equal if the prior is uniform.

The MMSE estimate minimizes the mean squared error, and can be shown to equal the

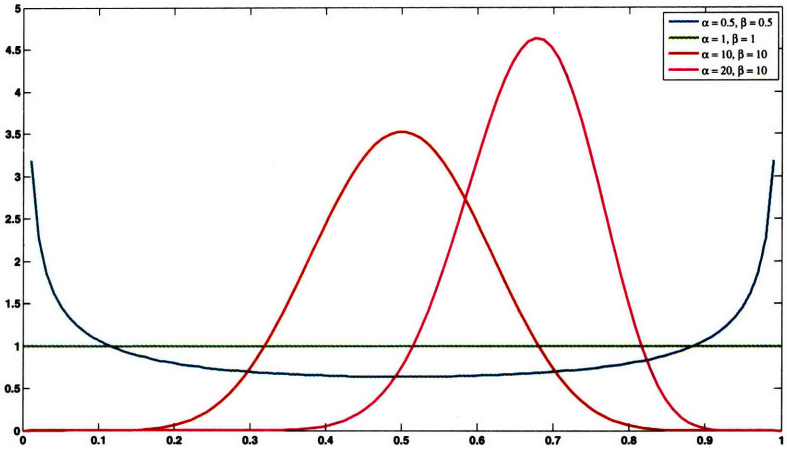

Figure 2-4: The Beta PDF with hyperparameters α and β is the univariate case of the Dirichlet PDF. Plotted here are Beta distributions for varying values of α and β. When $\alpha = \beta = 1$, the Beta PDF is uniform. As $\alpha + \beta$ increases, the distribution becomes more and more peaky around its mode, suggesting more certainty on the outcome of the random variable.

expectation of $P(\boldsymbol{\Theta}|\mathbf{D})$. For a Dirichlet PDF, this is given by:

$$\hat{\theta}_{i_{\text{MMSE}}} = E[\theta_i] = \frac{\alpha_i}{\sum_{i=1}^{r} \alpha_i}. \tag{2.6}$$

From the discussion above, one can see that the Bayesian framework provides us with a richer picture of the estimation problem than the classical approach, since the uncertainty in our estimate is quantified by the distribution of $\boldsymbol{\Theta}$. More importantly for applications, Bayesian probability allows us to define a nonuniform prior distribution that biases the estimates towards our prior beliefs or expert opinions.

2.1.3 Parameter Learning from Complete Data and Prior Knowledge

In this section, we discuss how the CPDs of a Bayesian network can be learned from a combination of data and prior knowledge. The CPDs are often referred to as the parameters of a network and are denoted collectively by $\boldsymbol{\Theta}$. As shown in Figure 2-5, the distribution of a parentless node X_i is simply a PMF with histogram heights $[\theta_{i,1}, \ldots, \theta_{i,r-1}]$, if that particular node has r possible outcomes. This is analogous to the die example in Section 2.1.2. Furthermore, it is assumed that the CPD of a child node can be independently specified for each combination of values taken by its parents. This *parameter independence* assumption, introduced by Spiegelhalter and Lauritzen [10] [11], means that the CPD of a child node X_i is also just a PMF with histogram heights $[\theta_{i,j,1}, \ldots, \theta_{i,j,r-1}]$, where j represents a particular parental configuration. Thus, learning the parameters of a Bayesian network, or *training* the network, amounts to determining $[\theta_{i,j,1}, \ldots, \theta_{i,j,r-1}]$ for each node and each combination of parental outcomes.

Taking the Bayesian approach to probability that was discussed in the previous section, $\boldsymbol{\Theta}_{\mathbf{ij}} = [\theta_{i,j,1}, \ldots, \theta_{i,j,r-1}]$ is then a Dirichlet random variable. Let $[\alpha_1^o, \ldots, \alpha_r^o]$ be the initial hyperparameters of $\boldsymbol{\Theta}_{\mathbf{ij}}$. Then $[\alpha_1^o, \ldots, \alpha_r^o]$ quantifies our prior knowledge or belief of what the parameters of the Bayes net should be before observing any evidence. The larger $\sum_{i=1}^{r} \alpha_i^o$ is, the more certain we are about our guess, since this corresponds to a more peaky distribution. $[\alpha_1^o, \ldots, \alpha_r^o]$ is a parameter that we can freely change, and is especially important in problems where the amount of available data is limited.

To update $[\theta_{i,j,1}, \ldots, \theta_{i,j,r-1}]$ for a node after observing evidence on that node, we use Equation 2.4: Let $\mathbf{d} = [d_1, \ldots, d_r]$ be an observed data set for node i and parental configuration j. Then the Dirichlet hyperparameters changes from $[\alpha_1^o, \ldots, \alpha_r^o]$ to $[\alpha_1^o + d_1, \ldots, \alpha_r^o + d_r]$. To obtain a point estimate for $[\theta_{i,j,1}, \ldots, \theta_{i,j,r-1}]$, we can either maximize its PDF using Equation 2.5 or take its expected value using Equation 2.6. Under the assumption that the data set is complete, learning the parameters of a Bayes net entails performing this process over all the nodes and parental configurations in the network. For a discussion on learning with missing data, refer to the explanation of the EM algorithm in Dempster et al. [12] or the tutorial by Bilmes [13].

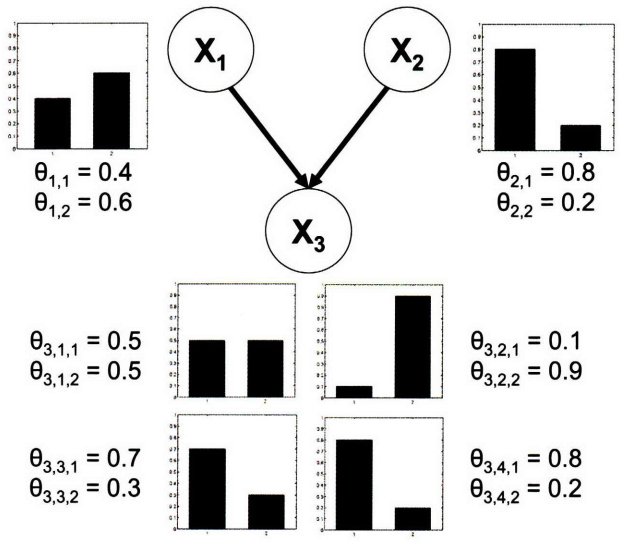

Figure 2-5: Each combination of outcomes for X_1 and X_2 gives rise to an independent CPD for X_3.

2.1.4 An Introduction to Inference

Once a Bayesian network is constructed, it can be used to infer the distributions of variables of interest, called *query* variables. The basic step in inference is marginalizing or conditioning the JPD of a network over those variables that are respectively inconsequential or observed. For instance, if we were interested in finding the marginal density of X_i from $P(X_1, \ldots, X_n)$, then

$$P(X_i) = \sum_{\mathbf{X} \setminus X_i} P(X_1, \ldots, X_n), \tag{2.7}$$

where $\mathbf{X} = \{X_1, \ldots, X_n\}$ and $\mathbf{X} \setminus X_i$ denotes the subset of $\mathbf{X}$ excluding X_i.

Marginalization is a concept from probability; it is not unique to probabilistic networks. The existence of a graphical representation of the JPD simply provides a visual understanding of its decomposition into a product of CPDs. Inference is a simple concept; the difficulty stems from the computational complexity of performing multiple summations over large multivariate functions. Thus, the goal of any inference algorithm is to reduce the complexity of this calculation by exploiting simplifying relationships between the variables.

One of the simplest and most general inference algorithms is *variable elimination*, whose basic idea is to eliminate factors of a JPD by distributing sums into products [14]. For instance, suppose we had the Bayesian network in Figure 2-6(a). From its structure, we can immediately say that its corresponding JPD is:

$$P(X_1, \ldots, X_5) = P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_2)P(X_5|X_3, X_4). \tag{2.8}$$

If we were interested in finding $P(X_4)$, then we would marginalize $P(X_1, \ldots, X_5)$ over all the variables except for X_4:

$$\begin{aligned} P(X_4) &= \sum_{X_1}\sum_{X_2}\sum_{X_3}\sum_{X_5} P(X_1, \ldots, X_5) \\ &= \sum_{X_1}\sum_{X_2}\sum_{X_3}\sum_{X_5} P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_2)P(X_5|X_3, X_4) \\ &= \sum_{X_1} P(X_1)\Big(\sum_{X_2} P(X_2)P(X_4|X_2)\Big(\sum_{X_3} P(X_3|X_1, X_2)\Big(\sum_{X_5} P(X_5|X_3, X_4)\Big)\Big)\Big) \end{aligned}$$

By factoring out terms that do not depend on certain summations, we are left with a series

of operations that are each over a function with fewer variables than in the original. This is desirable because the computational work required is bounded by the largest summation term [14]. This technique becomes inefficient for multiple query variables, but the concept of distribution of sums over products is present in other more efficient and specialized algorithms. The next section describes one such example: the junction tree algorithm.

For inference in the presence of evidence, a similar procedure is used to evaluate the conditional (rather than marginal) probability of the variable of interest, given the observations.

2.1.5 The Junction Tree Algorithm

The junction tree algorithm is a commonly used method for inference in Bayesian networks. Rather than acting directly on a Bayes net, it employs a message-passing routine on a secondary graphical structure called a *junction tree.* Thus, this algorithm can be applied to any type of graph that is convertible to a junction tree, making it a very versatile and general purpose inference technique. In this section, we describe the process of converting a Bayesian network to a junction tree, and then the subsequent message-passing routine that is used to compute the marginal distributions [15]. We demonstrate the steps on the example in Figure 2-6(a) as we progress through the algorithm.

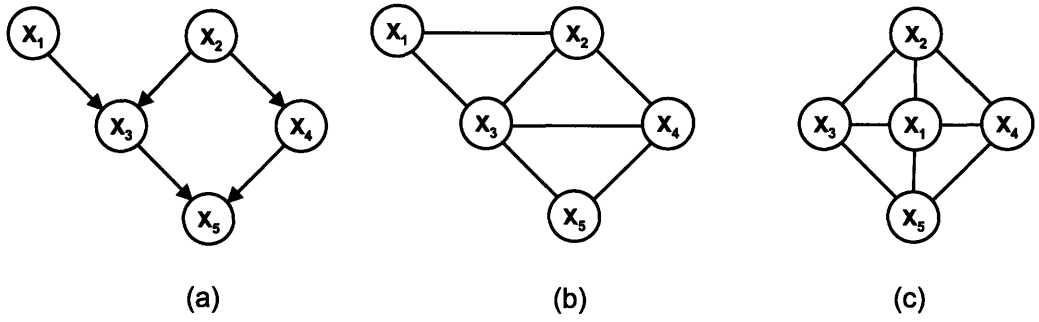

Figure 2-6: An example Bayesian network is shown in (a). The triangulated moral graph of (a) is shown in (b). The MRF in (c) is an example of a non-triangulated graph [2].

The first step is to convert the Bayes net into a undirected graph through *moralization,* which we defined in Section 2.1.1. Figure 2-6(b) shows the moral graph of Figure 2-6(a). Undirected graphs, called Markov random fields (MRFs), are themselves a commonly used modeling framework, and the remaining steps of this discussion are directly applicable to inference in MRFs.

The second step is to *triangulate* the moral graph by adding a chord to every cycle with four or more nodes. The moral graph in Figure 2-6(b) is already triangulated. Figure 2-6(c) is an example of a non-triangulated graph. It can be shown that an undirected graph can be converted to a junction tree if and only if it is triangulated [16]. Two nodes in an undirected graph are independent, conditioned on a third node, if and only if the removal

of the third node results in their separation. Thus, by adding extra chords, triangulation does not preserve all the independence relationships that existed in the original MRF. Also note that triangulation is not in general a unique process. The order in which the chords are added is associated with an *elimination order*, and there is an optimal elimination order that results in a junction tree with the least computational complexity; see Huang and Darwiche [15] for a discussion on how to choose the optimal elimination order.

The next step is to form a *junction tree* from the triangulated graph. The nodes of the junction tree are the *cliques* of the moralized triangulated graph. A clique is a maximally complete subgraph of an undirected graph, meaning that each node in a clique is connected to every other node in the clique, and no additional node can be included in it such that this is still true. In terms of notation, we will not distinguish between a clique C_i of the graph and the set of random variables associated with the clique.

We require the junction tree to satisfy the *running intersection property*, which states that for every pair of nodes i and j of the junction tree, i.e., for every pair of cliques C_i and C_j, all cliques on the unique path between C_i and C_j must contain $C_i \cap C_j$. Figure 2-7(a) shows the cliques of the moral graph in Figure 2-6(b), and Figure 2-7(b) shows a junction tree. The existence of a junction tree is guaranteed for any triangulated graph, as mentioned previously. In general, multiple junction trees can be derived from a given triangulated graph; a method for choosing the optimal one is described in Huang and Darwiche [15].

The last step is to explicitly draw out the *separator sets* between each node pair, as shown in Figure 2-7(c). A separator set S_k between cliques C_i and C_j is a node that contains the variables common to both cliques: $S_k = C_i \bigcap C_j$.

This junction tree that we have created is just another factorization of our original JPD, except that the factors have changed from CPDs to *potential functions.* A potential function, denoted by Ψ_{C_i} or Ψ_{S_i}, is simply a mapping from the variables in C_i to the nonnegative numbers. It can in general have arbitrary form and need not be a probability distribution. There is a potential function associated with each clique and separator set in a junction tree, and the product of the clique potentials divided by the product of the separator potentials must equal the JPD of the network:

$$P(\mathbf{U}) = \frac{\prod_i \Psi_{C_i}}{\prod_j \Psi_{S_j}}, \tag{2.9}$$

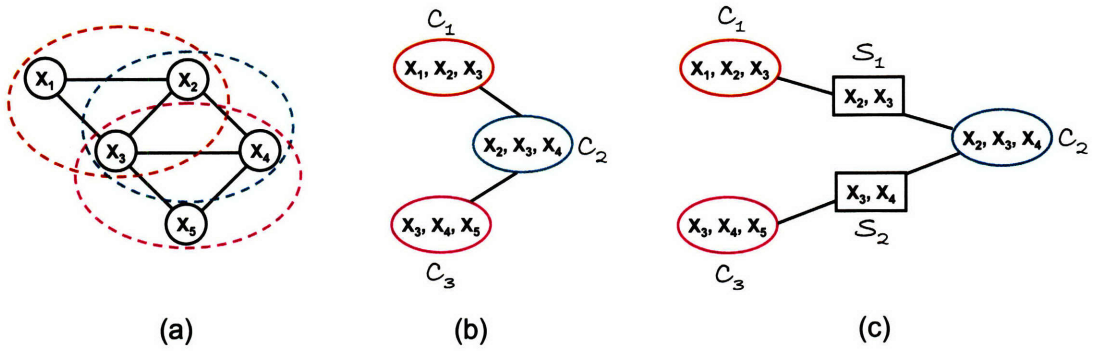

Figure 2-7: The cliques of a moralized triangulated graph are circled in (a). Its junction tree is shown in (b). The separators sets are explicitly drawn in (c) [2].

where $\mathbf{U}$ is the set of all the variables in the network, Ψ_{C_i} is the potential of the i^{th} clique, and Ψ_{S_j} is the potential of the j^{th} separator set.

We now explain how manipulation of these potential functions leads to the marginals of our query variables. The potential functions must first be initialized. Then they are rearranged via the aforementioned message-passing routine such that at its termination, the potential at each clique and separator set is equal to the marginal distribution over the variables contained by that node.

The initialization of a junction tree's potentials is done via the following procedure (see Figure 2-8):

1. Assign the potential of each separator set to 1:

$$\Psi_{S_j} = 1.$$

2. Assign each term of the original JPD factorization (see for example Equation 2.8) to a clique that contains the variables appearing in that CPD factor. If a clique is assigned more than one CPD factor, multiply them together:

$$\Psi_{C_i} = \prod_k P(X_k | \text{the parents of } X_k),$$

where X_k, and the parents of X_k, are contained in C_i.

Note that by assigning the potentials in this manner, the condition in Equation 2.9 holds.

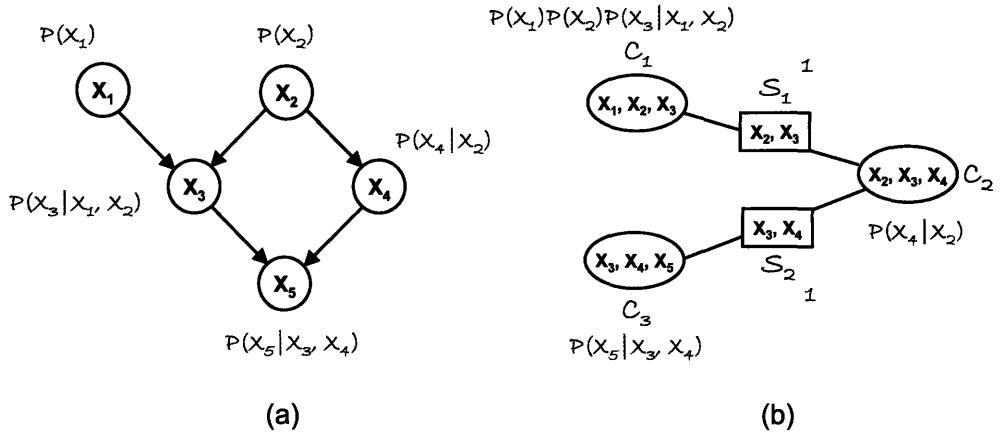

Figure 2-8: The CPD factors of the Bayesian network in (a) are assigned to cliques in the junction tree in (b) in the initialization step. The separator potentials are set to 1.

A message pass from an arbitrary clique C_a to its neighboring clique C_b, through their connecting separator set S, is defined as follows:

1. Create an updated separator potential Ψ_S^* that is equal to the summation of the potential of C_a over all the variables in C_a except those that are also in the separator set.

$$\Psi_S^* = \sum_{C_a \setminus S} \Psi_{C_a}$$

2. The potential of clique C_b is updated by multiplying its old potential by the ratio of the new separator potential to the old separator potential.

$$\Psi_{C_b}^* = \frac{\Psi_S^*}{\Psi_S} \Psi_{C_b}$$

If more than one branch intersects at C_b, then the separator ratios of each branch are multiplied together.

$$\Psi_{C_b}^* = \Big(\prod_i \frac{\Psi_{S_i}^*}{\Psi_{S_i}} \Big) \Psi_{C_b}$$

As a simple example, we show below that if the two node network in Figure 2-9 is

initialized such that Equation 2.9 holds, then Equation 2.9 remains true after two message passes in opposite directions.

Initialization:

$$\begin{aligned} \Psi_S &= 1 \\ P(C_1 \cup C_2) &= \frac{\Psi_{C_1}\Psi_{C_2}}{\Psi_S} \\ &= \Psi_{C_1}\Psi_{C_2} \end{aligned}$$

Message pass from C_1 to C_2:

$$\begin{aligned} \Psi^*_S &= \sum_{C_1 \setminus S} \Psi_{C_1} \\ \Psi^*_{C_2} &= \frac{\Psi^*_S}{\Psi_S}\Psi_{C_2} \\ &= \sum_{C_1 \setminus S} \Psi_{C_1}\Psi_{C_2} \end{aligned}$$

Message pass from C_2 to C_1:

$$\begin{aligned} \Psi^{**}_S &= \sum_{C_2 \setminus S} \Psi^*_{C_2} \\ &= \sum_{C_2 \setminus S} \sum_{C_1 \setminus S} \Psi_{C_1}\Psi_{C_2} \\ &= \sum_{C_2 \setminus S} \Psi_{C_2} \sum_{C_1 \setminus S} \Psi_{C_1} \\ \Psi^*_{C_1} &= \frac{\Psi^{**}_S}{\Psi^*_S}\Psi_{C_1} \\ &= \frac{\sum_{C_2 \setminus S} \Psi_{C_2} \sum_{C_1 \setminus S} \Psi_{C_1}}{\sum_{C_1 \setminus S} \Psi_{C_1}}\Psi_{C_1} \\ &= \sum_{C_2 \setminus S} \Psi_{C_2}\Psi_{C_1} \end{aligned}$$

Find the ratio of new clique potentials to separator potential:

$$\begin{aligned}\frac{\Psi^*_{C_1}\Psi^*_{C_2}}{\Psi^{**}_S} &= \frac{\sum_{C_2\setminus S}\Psi_{C_2}\Psi_{C_1}\sum_{C_1\setminus S}\Psi_{C_1}\Psi_{C_2}}{\sum_{C_2\setminus S}\Psi_{C_2}\sum_{C_1\setminus S}\Psi_{C_1}} \\ &= \Psi_{C_1}\Psi_{C_2} \\ &= P(C_1 \cup C_2)\end{aligned}$$

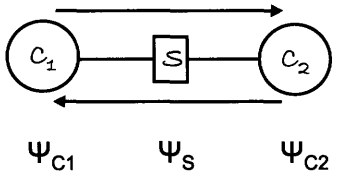

Figure 2-9: Two iterations of message-passing occur in opposite directions.

Now that we have defined message-passing between two cliques, we generalize it for an entire junction tree. The process has two stages called *Collect Evidence* and *Distribute Evidence*. These two stages correspond to the two iterations of message-passing discussed in the previous paragraph. If the junction tree potentials are initialized such that Equation 2.9 holds, then Equation 2.9 will still hold after running Collect Evidence and Distribute Evidence.

To perform Collect Evidence, first chose a clique in the junction tree to call the root. This choice can be arbitrary, but it is beneficial in terms of computation time to choose a root that is equally far from all the terminal cliques. Message-passing during Collect Evidence begins at the terminal cliques of the junction tree and propagates towards the root clique. A nonterminal clique node must wait until it has been updated by a message pass before initiating its own message. If a clique is connected to multiple branches, and thus receives multiple messages, then it must wait until all the messages have been received before propagating it forward. Collect Evidence terminates once the root node has been updated. Figure 2-10(a) shows the updated potentials of our example network after performing Collect Evidence with C_2 as the root.

Distribute Evidence is the same procedure as Collect Evidence, except that the message-

passing begins at the root. The routine ends when all the terminal cliques have been updated. At the end of Collect Evidence and Distribute Evidence, the potential function at every clique and separator is equal to the node's marginal distribution, $P(C_i)$ or $P(S_j)$, as shown in Figure 2-10(b). This is the main result of the junction tree algorithm. Recall that the purpose of the junction tree algorithm is to efficiently calculate summations over a JPD in order to find the marginals of our query variables. Now, instead of performing summations over the JPD of an entire network, we need only choose the smallest node that contains our variable of interest and perform marginalization on its potential function.

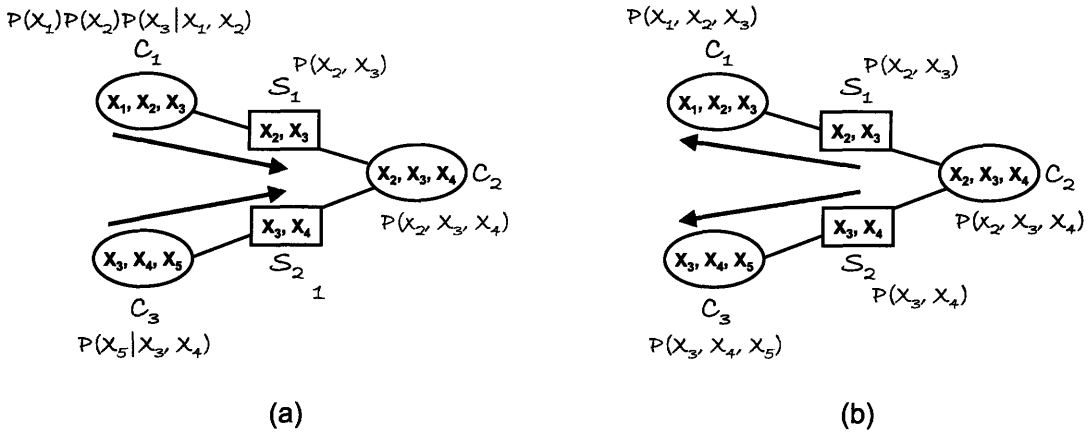

Figure 2-10: Collect Evidence propagates messages towards the root, as shown in (a). Distribute Evidence propagates messages towards the terminal nodes, as shown in (b).

The discussion above describes inference on a network where all the nodes are unobserved; there is no explicit treatment of evidence. Evidence changes the CPD factors of the original Bayesian network. For discrete random variables, CPDs are represented as tables whose entries are the probabilities of each possible outcome. Introducing evidence replaces the corresponding table entries with either 1 or 0, since that particular outcome is now known. Thus, each time new evidence is incorporated, the potential functions must be altered and Collect Evidence and Distribute Evidence must be rerun. However, the structure of the junction tree stays the same throughout.

2.2 Hidden Markov Models

Hidden Markov models (HMMs) are a type of probabilistic network that is commonly used in the modeling of dynamic systems. They are an extension of Markov chains and a close relative of Bayesian networks. Like Bayesian networks, nodes in an HMM represent random variables and directed edges represent probabilistic dependencies. Unlike Bayesian networks, all standard HMMs have the specific network structure shown in Figure 2-11. At each time instant, an unobservable *state* node X_t emits an observed *symbol* Y_t, also called the output node. The state variable models a first-order discrete-time Markov random process, which satisfies the Markov property. Similar to the Markov condition for Bayesian networks (Equation 2.1), the Markov property states that:

$$P(X_{t+1}|X_t, X_{t-1}, \ldots) = P(X_{t+1}|X_t). \tag{2.10}$$

In other words, the future state in an HMM is independent of the past states if conditioned on the present. Furthermore, the emitted symbol Y_t is independent of all other nodes if conditioned on X_t.

An HMM is defined by its transition, emission, and initial state probabilities: $P(X_{t+1}|X_t)$, $P(Y_t|X_t)$, and $P(X_0)$. If the nodes of the HMM are discrete, then these probabilities are defined by the matrices, $\mathbf{A} = [a_{ij}]$, $\mathbf{B} = [b_{ij}]$, and $\mathbf{\Pi} = [\pi_i]$, respectively, which are collectively referred to as the model parameters and denoted by $\lambda(\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$. The entry a_{ij} is the probability of a transition to state j in the next time step, given that the state is currently i,

$$a_{ij} = P(X_{t+1} = j|X_t = i). \tag{2.11}$$

The entry b_{ij} is the probability that state i will emit symbol j,

$$b_{ij} = P(Y_t = j|X_t = i). \tag{2.12}$$

The entry π_i is the probability that the Markov process begins in state i at time $t = 1$,

$$\pi_i = P(X_0 = i). \tag{2.13}$$

Notice that each row of $\mathbf{A}$ and $\mathbf{B}$ must sum to 1.

The canonical example of a situation that can be modeled by an HMM is the ball-and-urn problem [17]. Suppose we had M urns, each containing a different selection of balls of N possible colors. The urns are hidden from view and a sequence of balls is drawn for us to see. In this example, the urns are the states and the ball colors are the emitted symbols. The transition probabilities govern the sequence of urns from which the balls are drawn, and the emission probabilities correspond to the probability of choosing a certain ball color from each different urn. In this case, $\mathbf{A}$ is $M \times M$ and $\mathbf{B}$ is $M \times N$.

There are three basic problems associated with HMMs that are frequently encountered:

1. Given the model parameters λ, find the probability of a particular observation sequence, $\mathbf{y}$.

2. Given λ and an observation sequence $\mathbf{y}$, find the underlying sequence of states $\mathbf{x}^*$ that maximizes the probability of generating $\mathbf{y}$.

3. Given a set of observation sequences $\boldsymbol{\Upsilon} = \{\mathbf{y}\}$, find the λ that maximizes the probability of generating $\boldsymbol{\Upsilon}$.

The first problem involves evaluating $P(\mathbf{Y} = \mathbf{y}|\lambda) = \sum_{\text{all possible X}} P(\mathbf{Y} = \mathbf{y}, \mathbf{X}|\lambda)$, which can be accomplished efficiently using the *forward-backward* algorithm. This question is not relevant to our applications in Chapter 3 and will not be discussed in this thesis. More information on the forward-backward algorithm can be found in Alpaydin [17].

The second problem is essentially an inference problem, where the symbols are the evidential nodes and the states are the query nodes (see Section 2.1.4). It can be efficiently solved using the *Viterbi* algorithm, which is explained in Section 2.2.1. Note that in this problem formulation, inference occurs offline since the observations for the entire time interval must be known in advance. This is called fixed interval smoothing [18].

The third problem describes parameter learning from incomplete data, and can be solved using the Baum-Welch algorithm, an expectation maximization technique. For a detailed description of Baum-Welch, see Alpaydin [17]. If a complete data set is available for training, then the model parameters can easily be obtained via the method introduced in Section 2.1.3 for Bayesian networks. This is further discussed in Section 2.2.2.

HMMs are used for modeling in many areas of research. One well-known application is in the area of automatic speech recognition (ASR). Additional information on ASR can be found in Rabiner's tutorial on HMMs [19].

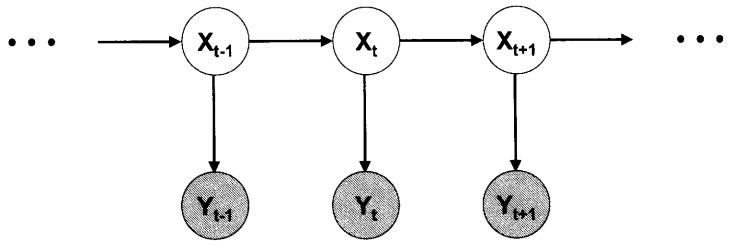

Figure 2-11: A standard Hidden Markov Model.

2.2.1 The Viterbi Algorithm

The Viterbi algorithm is a dynamic programming method for inferring the most likely sequence of hidden states, given a particular observation sequence and the model parameters. In other words, we are looking for $\mathbf{x}^*$ such that,

$$\mathbf{x}^* = \underset{\mathbf{x}=\{x_1,\ldots,x_M\}}{\arg\max} P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}, \lambda). \tag{2.14}$$

Let there be M possible states $\{S_1, \ldots, S_M\}$ in an HMM of length T with model parameters $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{\Pi}$. Let $\delta_t(j)$ equal the joint probability of the most likely sequence of states that ends in state j at time t, and the emission of all the observations up to time t:

$$\delta_t(j) = \max_{x_1,\ldots,x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t = j, y_1, \ldots, y_t|\lambda). \tag{2.15}$$

Finally, let $\psi_t(j)$ equal the most likely state at time $t-1$ such that the process is in state j at time t. In other words, $\psi_t(j)$ is the state at $t-1$ in the sequence of states associated with probability $\delta_t(j)$. As shown below, the Viterbi algorithm recursively calculates $\delta_t(j)$, while keeping track of the most probable penultimate state in $\psi_t(j)$. The most likely state sequence is then calculated as the algorithm backtracks through the recursion.

Initialization:

$$\begin{aligned} \delta_1(j) &= \pi_j b_{jy_1} \\ \psi_1(j) &= 0 \end{aligned}$$

Recursion:

$$\begin{aligned} \delta_t(j) &= \max_i \delta_{t-1}(i) a_{ij} \cdot b_{jy_t} \\ \psi_t(j) &= \arg\max_i \delta_{t-1}(i) a_{ij} \end{aligned}$$

Termination:

$$x_T^* = \arg\max_i \delta_T(i)$$

Backtrack:

$$x_t^* = \psi_{t+1}(x_{t+1}^*), \text{ for } t = T-1, T-2, \ldots, 1$$

In the initialization step, $\delta_1(i)$ is equal to the probability that the process will initially begin in state i and emit y_1. Since there is only one possible path for the process to take to reach i at $t = 1$, $\delta_1(i)$ is also the probability of the most likely path that terminates in i and yields y_1. This is consistent with the definition in Equation 2.15. $\psi_1(i)$ is arbitrarily initialized to 0 since there is not yet a penultimate state.

The recursive step assumes that we already have the probability of the most likely path that ends in i at $t-1$: $\delta_{t-1}(i)$. Notice that this probability is a function of the state at $t-1$. In other words, the state at $t-1$ is fixed by us, but all the states previous to that are chosen to be the most probable given the observations up to that point. $\delta_{t-1}(i)$ is then multiplied by the transition probability from i to j and the emission probability of generating symbol y_t from state j, thereby obtaining the probability of the state path that is in i at time $t-1$ and j at time t, and that emits the observed symbols. This expression is maximized over all the possible i's so that $\delta_t(j)$ is once again only a function of the latest state. The i that achieves this maximization is stored in $\psi_t(j)$. Thus, $\psi_t(j)$ has a different value for each possible current state j.

Once the recursion reaches the end of the HMM, $\delta_T(i)$ is the probability of the entire observation sequence and the most likely state sequence that ends in state i. The terminating step is to maximize $\delta_T(i)$ over the i's. The state that achieves this maximization, x_T^*, is the most likely state at time T.

After x_T^* is determined, we backtrack through the $\psi_t(j)$'s to find the rest of the sequence. Recall that $\psi_t(j)$ stores the most likely state at time $t-1$ given that the state at t is j. Thus, x_t^* is equal to $\psi_{t+1}(x_{t+1}^*)$ for $t = T-1, T-2, \ldots, 1$.

2.2.2 Parameter Learning from Complete Data and Prior Knowledge

The model parameters of an HMM are its transition, emission, and initial state probabilities, and the process of learning them is exactly the same as that in a Bayesian network. The transition probability corresponds to the CPD between node X_t and its only parent, X_{t-1}. The emission probability corresponds to the CPD between node Y_t and its only parent, X_t. As we work through the training data set, we increment the appropriate matrix element in $\mathbf{A}$ for each combination of X_{t-1} and X_t that is observed. Similarly, we increment the appropriate elements of $\mathbf{B}$ each time we observe a combination of $X(t)$ and $Y(t)$. The matrices can be initialized with nonuniform entries to bias the model parameters towards prior knowledge, functioning like the Dirichlet priors in Bayesian networks. Note that the matrices must be normalized over each row to obtain a proper stochastic matrix.

2.2.3 Multiple Output HMMs

There are many variations of the standard HMM structure that can be used to model more complex systems. Figure 2-12 shows two such examples [18]. One of the simplest extensions that can be made is to have multiple independent observed nodes for each state node, as shown in Figure 2-13(a). If the emission probabilities of Y_t and Z_t are $P(Y_t|X_t)$ and $P(Z_t|X_t)$, respectively, then $P(Y_t, Z_t|X_t) = P(Y_t|X_t)P(Z_t|X_t)$.

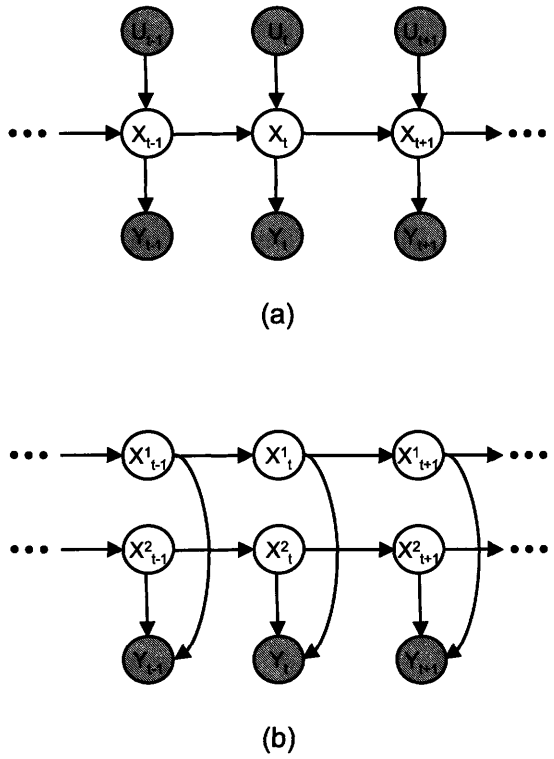

Figure 2-12: The transition probability of an input-output HMM, shown in (a), is conditioned on both the previous state and an input variable: $P(X_t|X_{t-1}, U_t)$. The emission probability of a factorial HMM, shown in (b), is conditioned on two or more independent states: $P(Y_t|X_t^1, X_t^2)$.

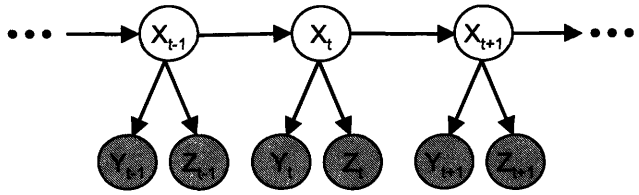

Figure 2-13: The state of a multiple output HMM emits two symbols at each time instance.

2.2.4 Autoregressive HMMs

In an autoregressive HMM, the assumption that an observation is only dependent on the current state is relaxed so that the observed variable also models a time series [20] [21], see Figure 2-14.

A discrete autoregressive HMM model with M states $\{1, \ldots, M\}$ and N symbols $\{1, \ldots, N\}$ is defined by its transition, emission, and initial state matrices, as follows:

1. A probability distribution for the first state, $\mathbf{\Pi} = [\pi_1, \ldots, \pi_M]$.

2. A two-dimensional state transition matrix, $A = [a_{ij}]$, that specifies the probability of transition to state j in the next time step, given that the state is currently i.

3. A three-dimensional transition array, $\mathbf{C} = [c_{ijs}]$, that specifies the probability of emitting symbol j at time t, given that the emitted symbol at time $t-1$ was i and that the state at t is s. $\mathbf{C}$ can also be written as a collection of two-dimensional matrices, $\mathbf{C}^s = [c^s_{ij}]$, to explicitly show that it is a transition matrix between successive symbol nodes. The rows of $\mathbf{C}^s$ should sum to 1.

Notice that an autoregressive HMM reduces to a standard HMM if each $\mathbf{C}^s$ has identical rows.

The algorithm for finding the most likely sequence of states in an autoregressive HMM is almost identical to the standard Viterbi algorithm, except $\mathbf{B}$ is replaced by $\mathbf{C}$ [21]:

Initialization:

$$\begin{aligned} \delta_1(j) &= P(x_1 = j, y_0, y_1) \\ &= \pi_j c^j_{y_0 y_1} \end{aligned}$$

Recursion:

$$\begin{aligned} \delta_t(j) &= \max_{x_1, \ldots, x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t = j, y_0, \ldots, y_t) \\ &= \max_i \delta_{t-1}(i) a_{ij} \cdot c^j_{y_{t-1} y_t} \end{aligned}$$

Termination:

$$x_T^* = \arg\max_i \delta_T(i)$$

Backtrack:

$$x_t^* = \arg\max_i \delta_t(i) a^*_{i x_{t+1}}, \text{ for } t = T-1, T-2, \ldots, 1$$

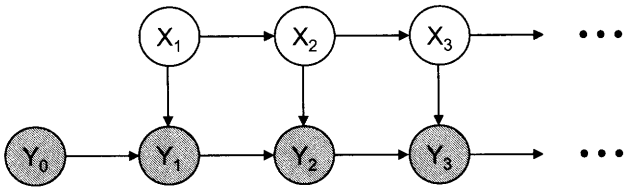

Figure 2-14: An autoregressive HMM.

2.3 Dynamic Bayesian Networks

In Sections 2.2.3 and 2.2.4, we extended the standard HMM structure to include more complex networks in each time-slice. Dynamic Bayesian networks (DBNs) are an even further extension, where the network in each time-slice can be any arbitrarily complex Bayes net, and the node are allowed to represent higher-order Markov random processes, i.e. to have a dependence on the past that extends beyond one time-slice. Thus, the concepts of state transition and emission probabilities no longer apply. Instead, a DBN is defined by:

1. A set of intra-slice CPDs that describes the factorization of the local Bayes net.

2. A set of inter-slice transition CPDs that governs the temporal relationships between slices. If all the nodes are first-order Markov, then the temporal probabilities need only be specified between adjacent time points. If there are higher-order temporal dependencies, then multiple sets of inter-slice probabilities must be specified (see Figure 2-15(b)).

The notion of emitted symbols and hidden states is now replaced by designating observed and unobserved nodes.

The process of parameter learning is essentially the same for DBNs as it is for static Bayesian networks and HMMs. Only two sets of probabilities need to be determined for first-order Markov DBNs: one set to characterize the Bayes net in the initial time slice, and one to relate the networks at times t and $t+1$.

Since there are minimal restrictions on the structure of DBNs, they can be powerful and versatile tools for modeling and estimation of sequential processes. However, the generality of the definition also leads to even greater computational challenges in performing inference. The most straightforward way to perform inference in a DBN is to *unroll* the network over the entire time interval. The unrolled network is then just an equivalent static Bayes net on which standard Bayesian network inference techniques, such as the junction tree algorithm, can be applied. However, one can easily see that even the simplest DBN (such as a standard HMM) can become intractable if unrolled over a long time interval. Also, since both the intra-slice and inter-slice probabilities do not change with time, there is a great deal of redundancy in completely unrolling a DBN. Another approach would be to convert the

DBN into an HMM and use the forward-backward algorithm. An in-depth explanation of these approaches, as well as other efficient DBN inference algorithms, can be found in Murphy [18].

In general, inference methods are either *exact* or *approximate*. The goal of exact inference algorithms is to obtain an exact expression for the marginal distribution over the query variables. The junction tree algorithm, for instance, is an exact inference technique. However, since DBNs can easily become very complex, exact inference is often computationally intractable. In contrast, approximate inference methods, such as the Boyen-Koller algorithm, settle for finding approximations to the marginal distributions. Further information on approximate inference can also be found in Murphy [18].

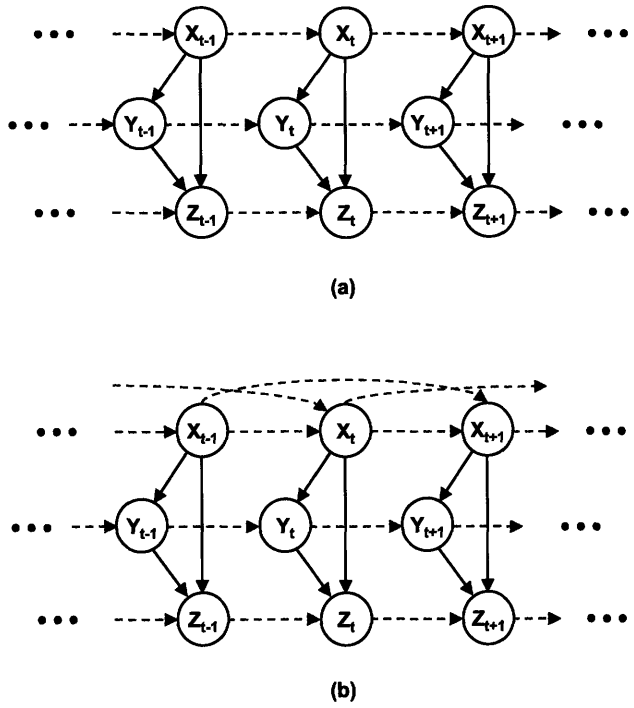

Figure 2-15: Three slices of an unrolled first-order Markov DBN are shown in (a). The blue dashed arrows indicate the inter-slice probabilistic relationships, while the black solid arrows represent the intra-slice dependencies. Shown in (b) is a DBN that also has a second-order Markov node, X.

Chapter 3

Applications to Cardiovascular Monitoring

In this chapter, we apply the concepts of probabilistic networks toward solving the problem of patient state estimation in the ICU. We use these networks as a method for capturing physiological relations in medical databases, and subsequently as a tool for estimating unobservable variables of interest in individual patients. Rather than trying to produce a diagnosis or a categorical result, like in many other medical applications of machine learning, we aim to derive an actual time series estimate of the unknown physiological parameter. Our focus is on cardiac output. The hope is that time series estimates will provide physicians with a more complete picture of a patient's medical condition than a binary decision variable. We also emphasize here that this modeling framework can in principle be easily extended to other selections of query and evidential variables.

In the first section of this chapter, definitions of important physiological terms and abbreviations will be stated. The origins and limitations of the two data sets that we will be using are described in Section 3.2. Section 3.3 introduces the various software toolboxes that were used in the implementation of the code. Sections 3.4, 3.5, and 3.6 present the different networks that were tried, and the results of each approach. Finally, Section 3.7 provides a summary of the results and a discussion of the major points of the chapter.

3.1 Terminology and Abbreviations

There are a few terms from cardiovascular physiology that will be used repeatedly in the remainder of this thesis. They include heart rate (HR), mean arterial blood pressure (ABP), cardiac output (CO), stroke volume (SV), and total peripheral resistance (TPR).

The definition of HR is self-evident and its values are presented in units of beats per minute (bpm).

ABP is the mean pressure in the systemic arteries (the arteries leading away from the left ventricle), and is usually sampled from the femoral artery.

CO is the average flow rate of blood that is pumped out of the left ventricle. It is a quantity that cannot be continuously measured without attaching a flowmeter to the aorta. A clinical method for obtaining sporadic CO measurements is thermodilution, which involves tracking the thermal profile of the bloodstream after an injection of a bolus of cold fluid. Thermodilution is generally only performed a few times during a patient's ICU stay, as it requires the insertion of a catheter by a physician, and careful execution. CO is a critical measurement of cardiovascular health because it directly affects the amount of oxygen delivered to the body.

SV is the amount of blood that the heart pumps out per cardiac cycle. SV, CO and HR are related by the following equation:

$$SV = \frac{CO}{HR}. \tag{3.1}$$

TPR is the effective resistance to flow of the entire systemic vasculature. It is analogous to electrical resistance if a pressure drop is thought of as a voltage drop, and fluid flow is thought of as current flow:

$$TPR = \frac{ABP}{CO}. \tag{3.2}$$

We will also be looking at the effects of three drugs: Dobutamine (DBM), Esmolol (ESM), and Nitroglycerin (NTR). Dobutamine is a β_1 agonist, which means that its effects on the cardiovascular system are to increase contractility (a measure of the strength of cardiac contractions) and HR. Thus, an infusion of Dobutamine will increase the subject's CO. Esmolol is a β_1 blocker and has the opposite effects of Dobutamine. Nitroglycerin is a vasodilator, which causes TPR, and consequently ABP, to drop.

3.2 Description of the Data Sets

The networks presented in this chapter were validated using two different sets of data. The first set was derived from experimental procedures conducted on nine Yorkshire swine [22], five of which were appropriate for our purposes. Each file in this data set contains continuous measurements of HR, CO, and ABP. Time stamps and dosage records of drug infusions are also available for each swine. The main advantage of this data set is the availability of true, invasively measured CO. The main disadvantage is the small number of subjects.

The second data set comes from the MIMIC II ICU database [23], which is a collection of recordings from actual ICU patients. This data set contains continuous measurements of HR and ABP, but only sporadic CO measurements via thermodilution (only 1351 points in total across 120 patients). Since such a small number of noncontinuous CO data is insufficient for our purposes, we supplement these 120 MIMIC II patients with continuous CO estimates from Parlikar [24]. The prior work described in Section 3.4.1 also uses CO estimates obtained by Liljestrand's method [25]. We consider these estimates to be the truth when training and testing our networks, because they are our best alternative. It is reasonable for us to test our CO estimates using another estimated value as reference, because methods such as those described by Liljestrand and Parlikar are derived from detailed models of the cardiovascular system, and require knowledge of variables that our methodology does not employ, such as actual blood pressure waveforms. Thus, the dynamics that we are trying to capture with network models are not on the same accuracy scale as those described by the aforementioned methods.

In addition to the lack of true measured CO, another disadvantage of the MIMIC II data set is that the physiological conditions of the patients generally do not fluctuate significantly. The lack of variability is not ideal for training data because it translates to unpopulated entries in the conditional probability matrices.

Records of administered medications also exist for this data set; however, extracting this information is nontrivial and was not done for this thesis.

3.3 Software

The Bayesian network and DBN code that was written for this chapter employs K. P. Murphy's implementation of the junction tree algorithm, which can be found in his Bayes

Net Toolbox for Matlab [26]. The Matlab HMM toolbox was used for the most-likely-state-sequence calculations in Section 3.5.

In addition, we extensively explored the GeNIe/SMILE software package for Bayesian networks and DBNs [5], although we do not present any results from it. SMILE is a C++ library of tools for developing and manipulating Bayesian networks. It contains implementations of several approximate inference algorithms, and has been updated recently to support DBNs. It is a good software package to use if approximate inference is necessary (such as in a large network), but is currently lacking in documentation with regards to DBNs. GeNIe is a graphical user interface that can be used in conjunction with SMILE. Its functionalities are not completely caught up with those of SMILE, but it is an accessible tool for anyone interested in Bayesian networks to use.

3.4 Bayesian Network Models

We first use static Bayesian networks to compute estimates for cardiac output. It is important to emphasize that although the estimates in this section are time series, simple Bayesian networks do not contain information on the temporal behavior of their variables. In other words, the cardiac output times series are simply obtained by piecing together point estimates made at each time instance.

3.4.1 Prior Work and Results

The background for the work in this thesis is primarily derived from a thesis and paper by J. M. Roberts [3] [4], who tackled the same problem of estimating unobservable cardiovascular characteristics from measurable hemodynamic signals. The Bayesian network structure developed by Roberts is shown in Figure 3-1. Following the notational convention of HMMs, the shaded nodes are observed and the white nodes are unobserved. Thus, in addition to CO, TPR and SV are also estimated. All the nodes are discrete and have sample spaces of size 5.

Roberts used the same MIMIC II data set described in Section 3.2 for training and testing her network. In her error analysis, Liljestrand's estimate of CO was used as the "true" CO value [25]. The true SV and TPR were calculated using Liljestrand's CO and Equations 3.1 and 3.2, respectively. Porcine data was not used in that study.

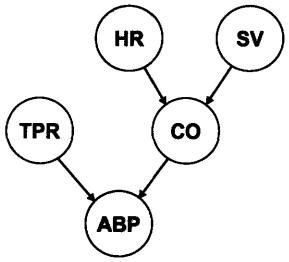

Figure 3-1: Bayesian network from Roberts et al [3]. The shaded nodes are unobserved. "HR" is heart rate, "CO" is cardiac output, "SV" is stroke volume, "TPR" is total peripheral resistance, and "ABP" is arterial blood pressure.

Two methods of parameter learning were explored in Roberts for determining the CPDs of the Bayes net: batch learning and sequential learning. Both methods employ the theory presented in Sections 2.1.2 and 2.1.3 in that the training data was used to increment the Dirichlet hyperparameters of each CPD. In both methods, the Dirichlet priors were initialized to be uniform.

In batch learning, the network was trained using all 1351 thermodilution CO measurements, their concurrent measurements for ABP and HR, and the calculated values for SV and TPR (again using Equations 3.1 and 3.2). Since the 1351 sets of points are taken across all 120 patients, they represent the physiological behavior of the "general population". However, these points do not capture any kind of temporal relationships between the variables.

This network was tested on one particular patient by producing estimates of CO, SV, and TPR at each time point, given concurrent measurements of ABP and HR. It is evident from the resulting plot, shown in Figure 3-2, that this method fails to capture the desired relationships between model variables.

In sequential learning, the network CPDs were computed from data ranging from time $t - N$ to t from one particular patient, and used to estimate that patient's CO, TPR, and SV at time $t + 1$. The Dirichlet priors were once again initialized to be uniform, and ABP and HR values at time $t + 1$ were given as evidence. This process was repeated along the entire length of the data, creating a time series for CO, TPR, and SV. Figure 3-3 shows the resulting plot. The low errors produced by this method are somewhat misleading; since there is such little variability in the MIMIC II data, a simple sample-and-hold of the last

training point produces estimates of comparable quality.

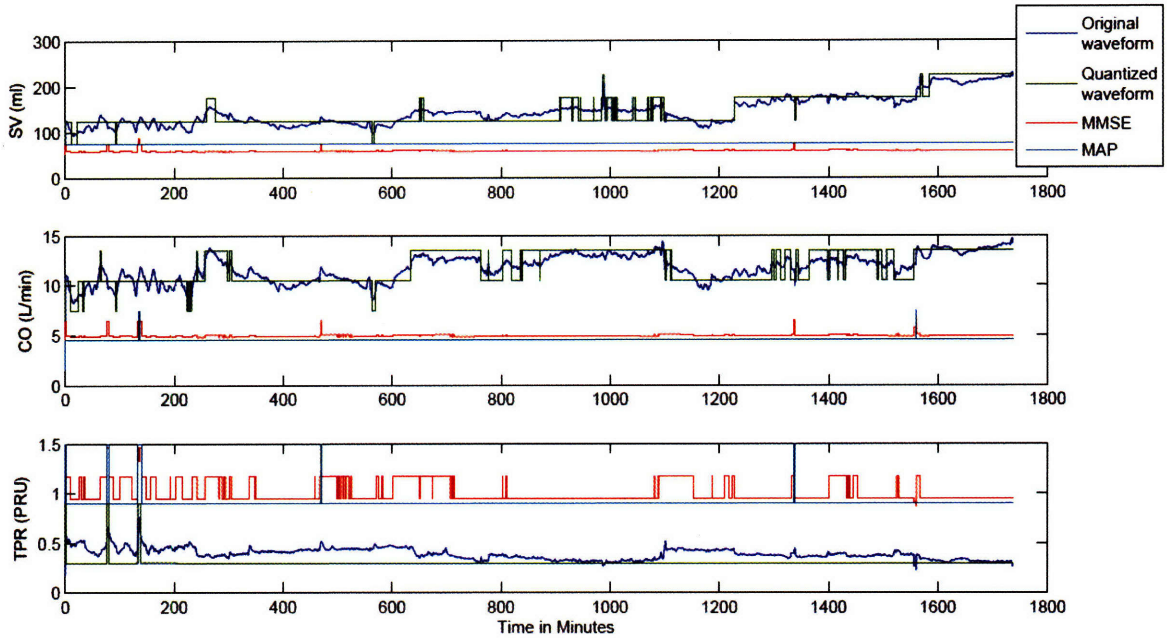

Figure 3-2: The results of batch training, taken from Roberts [4]. Shown in blue are the "true" values of CO, TPR, and SV derived from Liljestrand's method. Shown in green are the quantized versions of the blue waveforms. The red and cyan plots are the Bayesian network MMSE and MAP estimates, respectively.

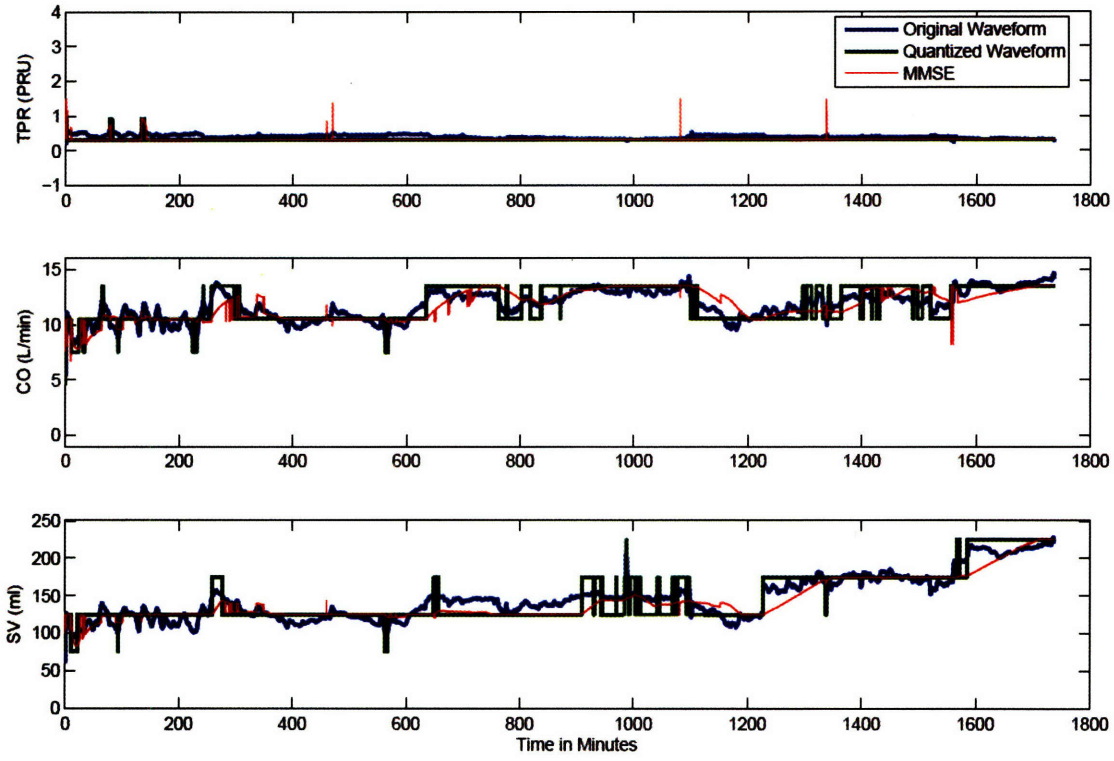

Figure 3-3: The results of sequential training, taken from Roberts [4]. Shown in blue are the "true" values of CO, TPR, and SV derived from Liljestrand's method. Shown in green are the quantized versions of the blue waveforms, and shown in red are the Bayesian network MMSE estimates. The training window size N is 90 minutes.

3.4.2 Current Models and Results

The first alteration that we made to the approach presented in Roberts is to eliminate the SV and TPR nodes in Figure 3-1. Not only are SV and TPR never observed in the data, they are deterministically related to HR, ABP, and CO via Equations 3.1 and 3.2. Thus, they do not add any value to the modeling problem. The new network is shown in Figure 3-4. Its structure was chosen to be as general as possible; we do not assume independence between any of the variables. Since this network is so small, there is no need to reduce its structural complexity for computational purposes.

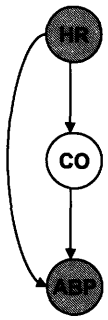

Figure 3-4: Current static Bayesian network model.

The second alteration that we made to Roberts' approach is to focus on batch learning rather than sequential learning. In a realistic setting, it is unlikely that a training value for CO (or another unobservable variable) will be consistently available shortly before the time of estimation.

With batch learning, a network is used to estimate a particular patient's unobservable physiological parameters based on what is known about the general population. However, rather than using the set of thermodilution points for training like in Figure 3-2, we turned to porcine data that contain continuous true measurements of CO. We computed the CPDs of the Bayes net from concurrent measurements of ABP, HR, and CO from four of the five animals. The data was sampled at approximately half-minute intervals to be closer to the time scale of physiological change, rather than that of noise, and quantized into 35 bins. The sampling interval and quantization levels are kept constant throughout the rest of this chapter, so that valid comparisons can be made between the different networks.

Once the CPDs were learned, the Bayesian network was given ABP and HR values from the fifth swine, referred to as "Pig 9" in Mukkamala et al. [22], and inference via the junction tree algorithm was performed to obtain the PMF of CO. From this distribution, the MAP value for CO was found for each ABP and HR pair. We do not use the MMSE estimate because it is not guaranteed to be discrete; it makes more sense to compare consistently quantized waveforms when doing error analysis than to compare a discrete waveform with a continuous one.

Figure 3-5 shows the Bayesian network results for Pig 9. The estimated values track the measured values quite nicely for the first 30 minutes, indicating that the network is capable of determining the baseline CO value for Pig 9 without any knowledge of its actual CO behavior - not even a calibration factor. However, the quality of the estimates decreases when the data fluctuates outside the nominal CO range. These fluctuations are due to the experimental interventions that were performed on the swine. Since all the animals were stimulated using different combinations of drugs and procedures (such as hemorrhaging), the training data for CO outside the nominal range is sufficiently different from the test data that large errors are produced.

Similarly for the MIMIC II data set, we tested the network on one randomly chosen subject, and trained with the remaining 119 subjects. Figure 3-6 shows the results for Patient b68062. Figure 3-7 shows the same plot, but zoomed in to the physiological range of CO. Again, without any information on the CO behavior of this specific patient, the Bayes net is able to produce estimates that on average track the true values.

In order to compare the different models that are introduced throughout this chapter, we use two criteria to quantify error: mean absolute normalized error (MANE) and root mean square normalized error (RMSNE). They are defined as follows:

$$MANE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{True_i - Est_i}{True_i}\right| \tag{3.3}$$

$$RMSNE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{True_i - Est_i}{True_i}\right)^2} \tag{3.4}$$

We use MANE in addition to RMSNE because RMSNE penalizes heavily for singular occurrences of large error. However, since our waveforms are quantized into nonuniform

bins (the bins at extreme values are larger) and we use the midpoints of those bins for calculating error (to preserve the measurement units), we expect to see larger errors at extreme values, and do not want to bias our calculations unfairly based on that. In general, normalized errors larger than 1 are considered to be bad.

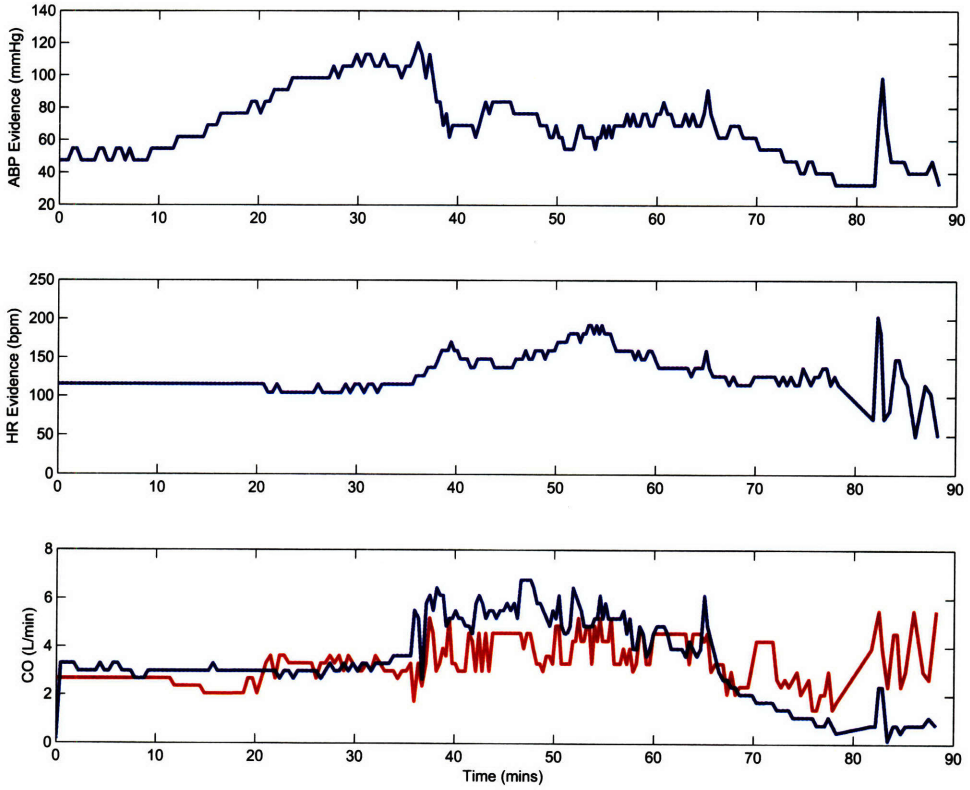

Figure 3-5: Estimated CO for Pig 9 using the current static Bayes net is shown in red. Experimentally measured CO is shown in blue. The MANE is 0.63 and the RMSNE is 1.80.

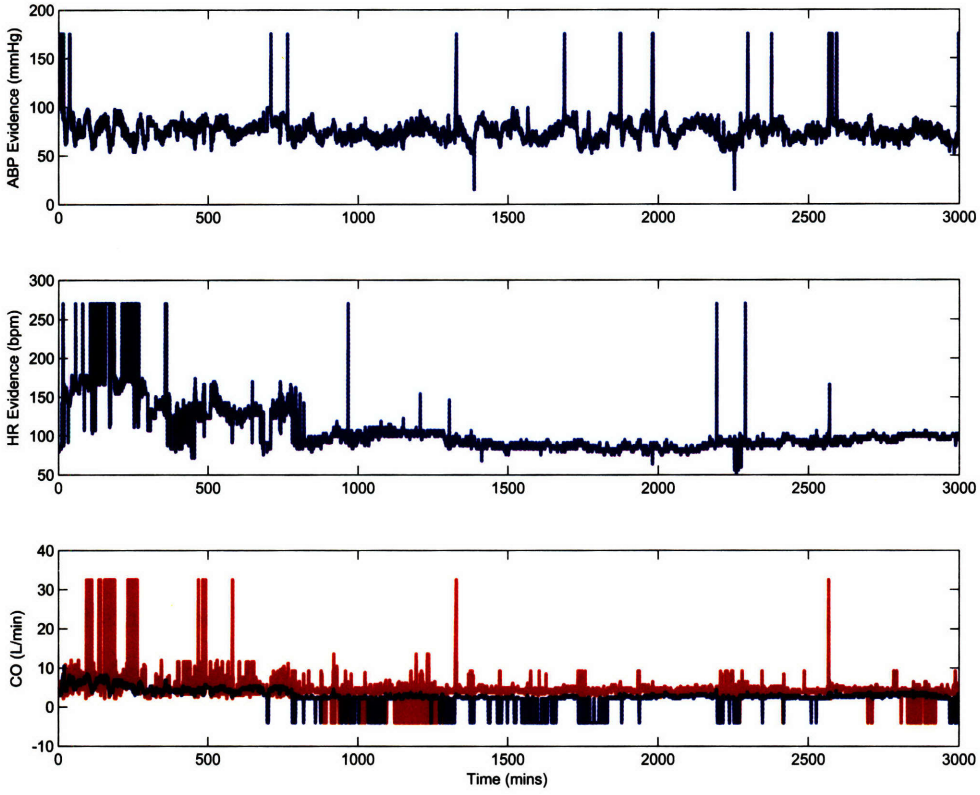

Figure 3-6: Estimated CO for Patient b68062 using the current static Bayes net is shown in red. Parlikar's CO estimates are shown in blue. The MANE is 0.72 and the RMSNE is 1.09.

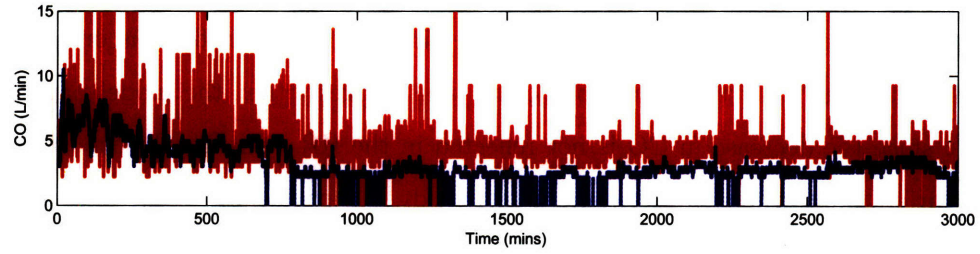

Figure 3-7: Magnified version of Figure 3-6 showing only CO that is in the physiological range.

3.5 Hidden Markov Models

In this section, we present the results of CO estimation using various types of HMMs for both the porcine and MIMIC II data sets. In addition, we show estimates of CO obtained from established deterministic methods that employ the same variables. We move from using static networks to dynamic networks in order to better capture the temporal correlations in the data. As in the previous section, Pig 9 and Patient b68062 were used for testing, and the remaining four pigs and 119 patients were used for training.

3.5.1 Single-Chain HMM

One of the simplest dynamic probabilistic networks that can be used is a standard single-chain HMM relating CO and HR, as shown in Figure 3-8. The transition and emission matrices were obtained from the training set and an initial uniform prior, and the most likely CO sequence was calculated using the Viterbi algorithm. The results for Pig 9 and Patient b68062 are shown in Figures 3-9 and 3-10, respectively. Also plotted for comparison are CO estimates calculated from the following simple deterministic method, as described in Parlikar [24]:

$$CO_{est} = k \cdot HR, \tag{3.5}$$

where k is a constant calibration factor that is equal to the ratio of the mean CO of the training data to the mean HR of the training data.

The HMM estimate for Pig 9 nicely captures the increase and decrease in CO at around the 35 and 65 minute marks, respectively. However, the transient fluctuations in the data are barely picked up at all. In addition, the estimated waveform flattens out at the end of the plot, while the true value dips down. This suggests that the model might be too simple, since the decrease in CO is not reflected in the HR evidence. Incorporating more input variables into the network would help remedy this problem.

Surprisingly, a simple scaling of HR also performs quite well as a CO estimator. This means that SV stays fairly constant across all the data. Notice that the estimated CO waveform is not exactly a scaled version of the HR evidence; this is because the scaling was done on the continuous waveforms prior to quantization.

In the case of patient data, the HMM performs terribly. Large artifacts in the training data and HR evidence caused our estimate to jump to values that are not physiologically

possible. Processing the data to remove such obvious artifacts beforehand would surely improve the HMM's performance, and is suggested for future work.

A constant scaling of HR does a decent job of tracking the CO for this patient, except for a persistent offset that actually contributes significantly to the error criteria. However, since there is so little variability in the patient data, it is not surprising that such a method works.

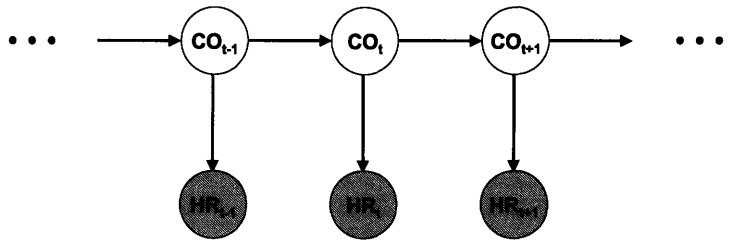

Figure 3-8: Single-chain HMM.

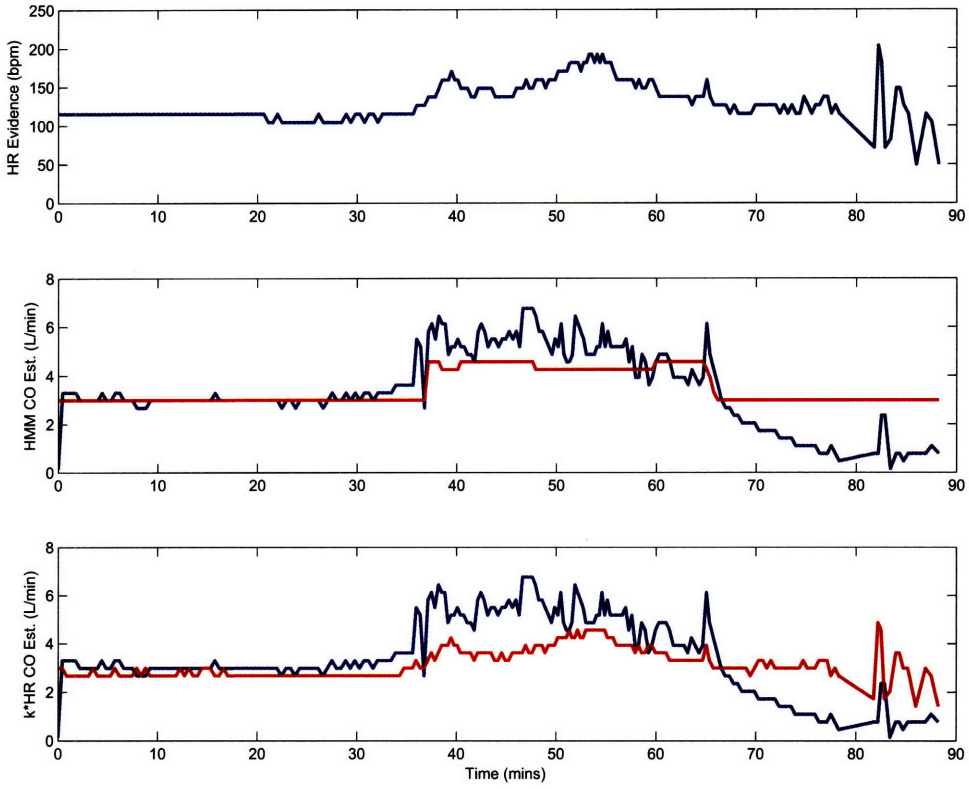

Figure 3-9: Estimated CO for Pig 9 using a single-chain HMM is shown in the second panel in red. Estimated CO obtained from Equation 3.5 is shown in the third panel in red. Experimentally measured CO is shown in blue. The observed sequence of HR is shown in the first panel. In the second panel, the MANE is 0.58 and the RMSNE is 1.96. In the third panel, the MANE is 0.61 and the RMSNE is 1.76.

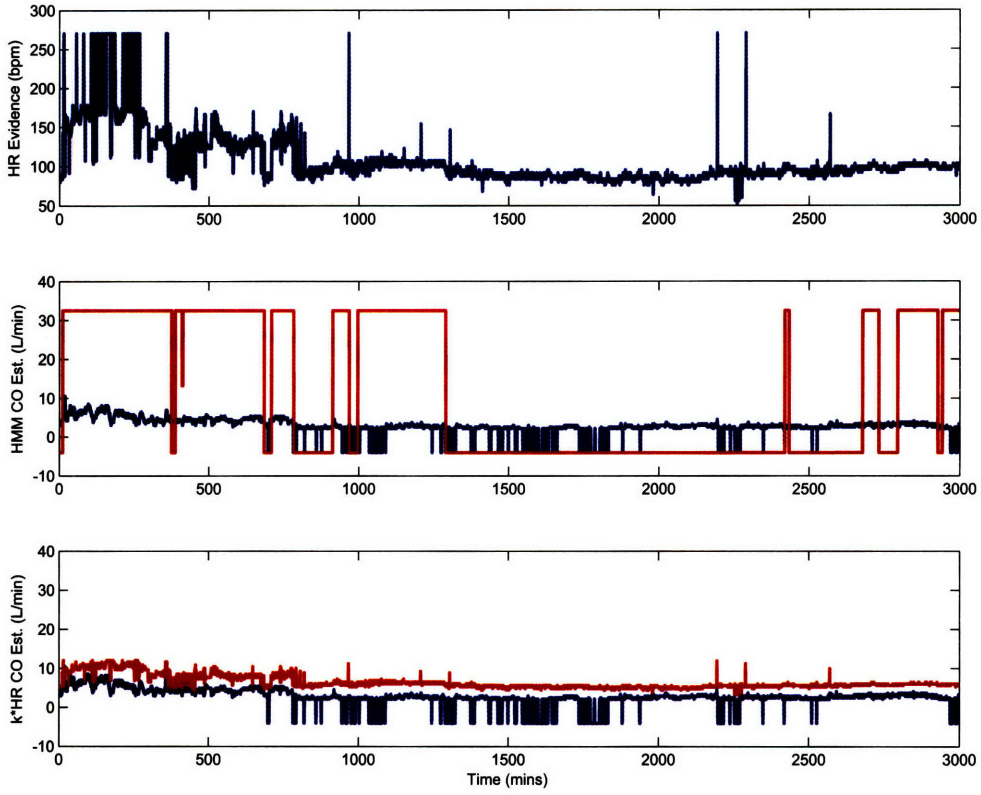

Figure 3-10: Estimated CO for Patient b68062 using a single chain HMM is shown in the second panel in red. Estimated CO obtained from Equation 3.5 is shown in the third panel in red. Parlikar's CO estimates are shown in blue. The observed sequence of HR is shown in the first panel. In the second panel, the MANE is 5.08 and the RMSNE is 6.07. In the third panel, the MANE is 1.11 and the RMSNE is 1.22.

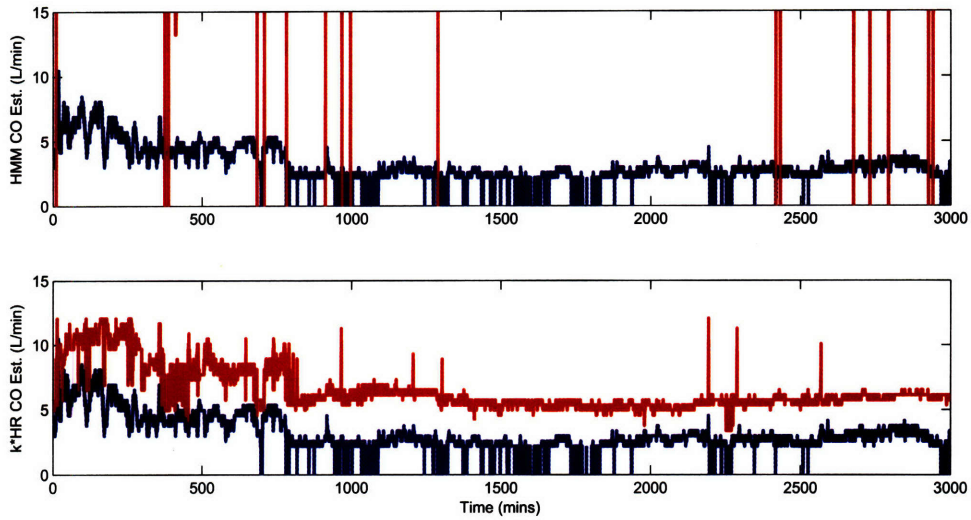

Figure 3-11: Magnified version of Figure 3-10 showing only CO that is in the physiological range.

3.5.2 Multiple Output HMM

A slightly more complex HMM model is one that also incorporates drug information. Figure 3-12 shows a HMM where CO is the hidden state, and HR and Dobutamine are both emitted symbols. The Dobutamine node is binary and indicates the time period during which the drug was being infused. The results of using this network on Pig 9 is shown in Figure 3-13. We do not apply this model to the MIMIC II data because patient medication records were difficult to extract, as previously mentioned.

The estimated waveform produced by this network is essentially the same as the one obtained from a single-chain HMM, suggesting that information on Dobutamine administration does not add value to the model.

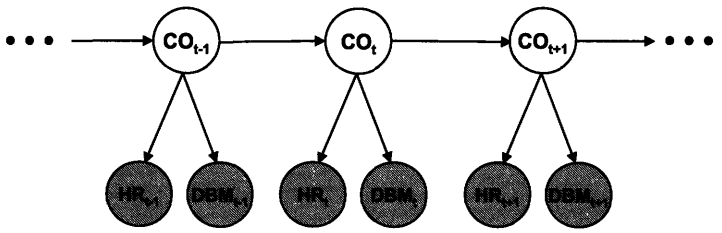

Figure 3-12: Multiple output HMM with hidden state CO, and emissions HR and Dobutamine.

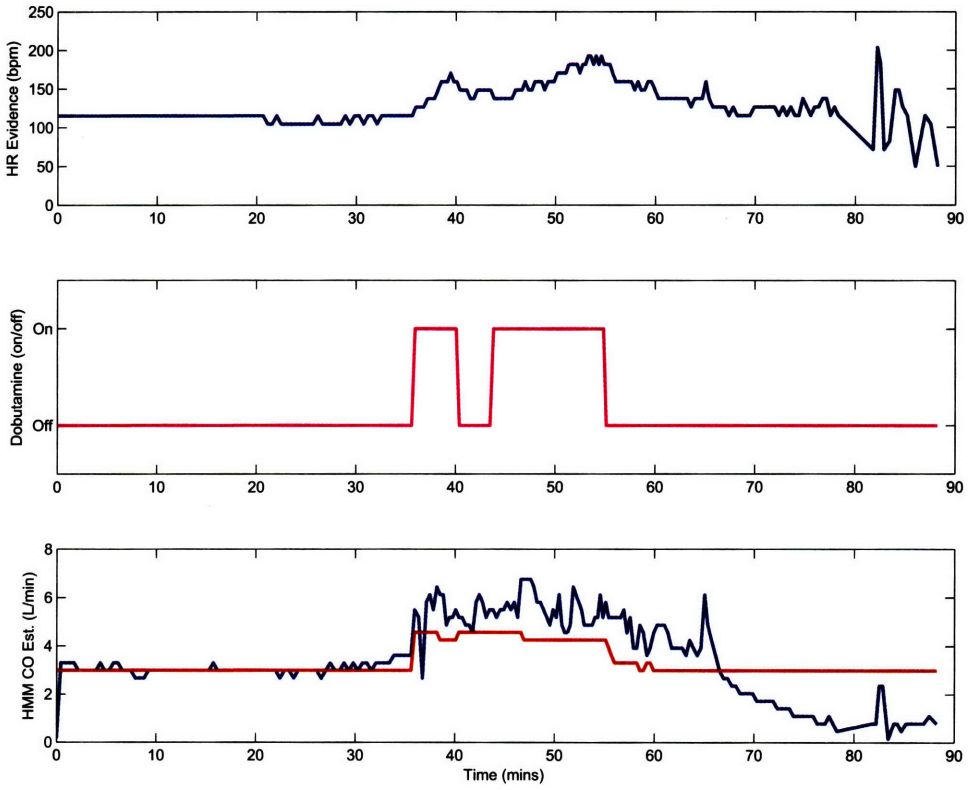

Figure 3-13: Estimated CO for Pig 9 using a multiple output HMM is shown in the third panel in red. The first panel again shows the observed sequence of HR. The second panel shows the periods of Dobutamine infusion. The MANE is 0.60 and the RMSNE is 1.97.

3.5.3 Autoregressive HMM

Since HR is clearly itself a random process with temporal correlations, the next step we take is to fit our data onto an autoregressive HMM. Figure 3-14 shows the network structure, and Figures 3-15 and 3-16 show the results for Pig 9 and Patient b68062, respectively.

The autoregressive HMM results for both Pig 9 and Patient b68062 differ minimally from the results obtained from a single-chain HMM, suggesting again that the relationship between HR and CO is dominant compared to other correlations in the data.

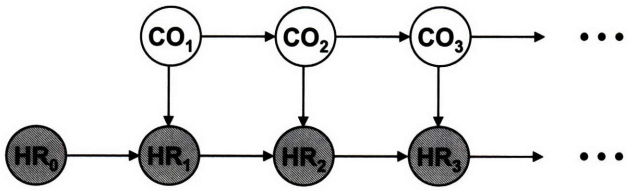

Figure 3-14: Autoregressive HMM relating CO and HR.

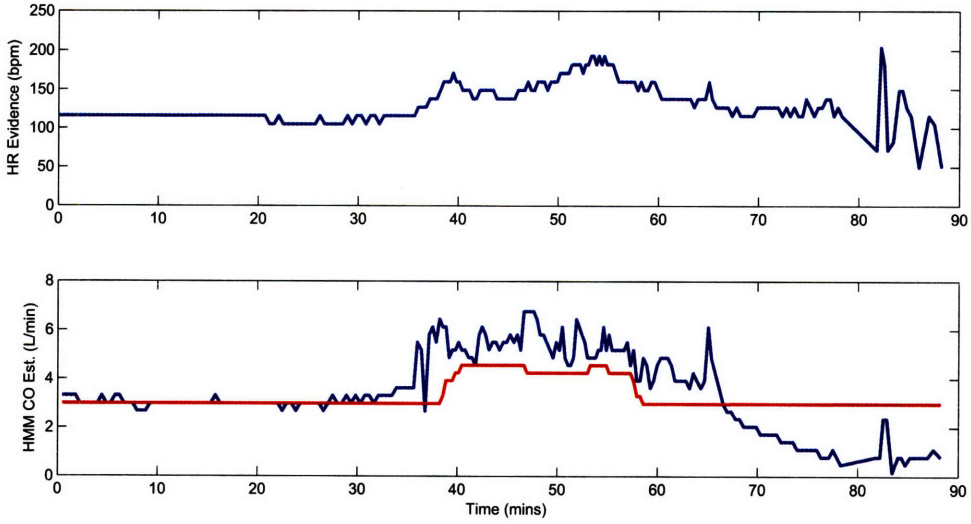

Figure 3-15: Estimated CO for Pig 9 using an autoregressive HMM. The MANE is 0.52 and the RMSNE is 1.54.

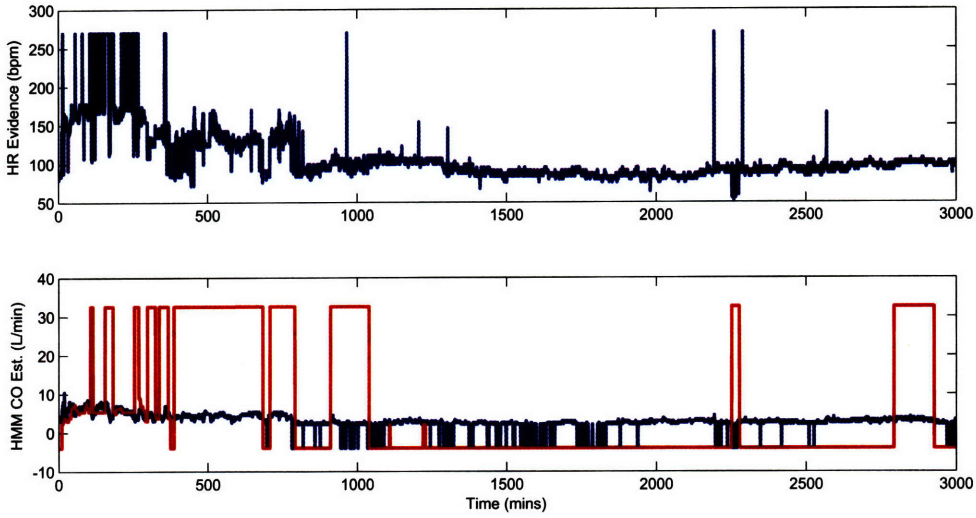

Figure 3-16: Estimated CO for Patient b68062 using an autoregressive HMM. The MANE is 3.63 and the RMSNE is 4.76.

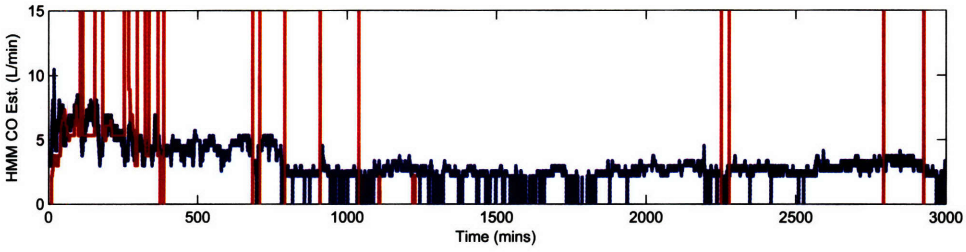

Figure 3-17: Magnified version of Figure 3-16 showing only CO that is in the physiological range.

3.6 Dynamic Bayesian Network Models

Lastly, we present the results obtained from DBNs. Although the networks in this section do not seem to differ much from the HMMs presented previously, there is in fact a significant jump in generality between the two methods. This is because the implementations of HMM models all use some variation of the Viterbi algorithm, which requires a particular network structure. In contrast, the DBNs in this section use the junction tree algorithm for inference. Thus, these models can be expanded to arbitrary complexity, although computation time and space remain a limiting factor.

3.6.1 Prior Work and Results

A prior study that explored the use of DBNs in the estimation of cardiovascular variables is J. Hulst's Master of Science thesis [5]. In it, he develops the DBN shown in Figure 3-18, and uses it to infer unobservable variables of interest before and after the onset of cardiogenic shock - a low blood pressure condition that results from heart failure. Since his DBN is quite involved and contains many physiological variables that are realistically unmeasurable, only simulated data was used to validate the model. Cardiogenic shock was simulated by abruptly decreasing the maximum elastance of the left ventricle. The evidential nodes are HR, systolic, and diastolic arterial blood pressure; all the other nodes were unobserved. Figure 3-19 shows the results given by Hulst. It is evident that the DBN performs very well, as the onset of shock is is clearly captured by the binary shock decision variable. However, it is important to note that the network was trained and tested on simulated data that were all perturbed in the same manner. It is unclear how well it would perform with experimental data.

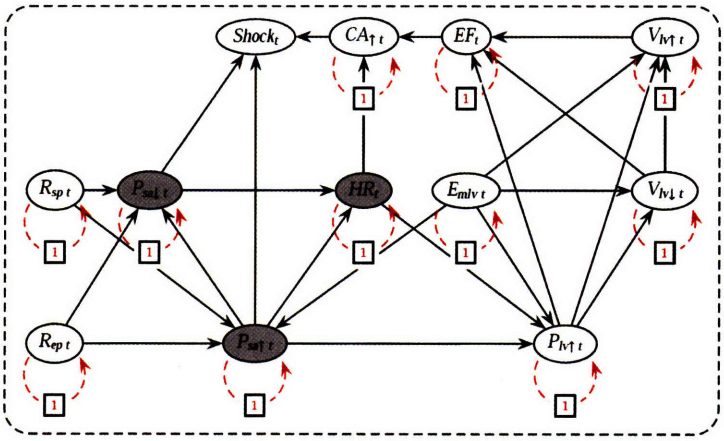

Figure 3-18: DBN model of the cardiovascular system as presented in Hulst [5]. CA is cardiac output, EF is ejection fraction, E_{mlv} is maximum elastance of the left ventricle, P_{lv} is the pressure in the left ventricle, P_{sa} is the pressure in the systemic arteries, V_{lv} is the volume of the left ventricle, and the R's are resistances of various parts of the circulation.

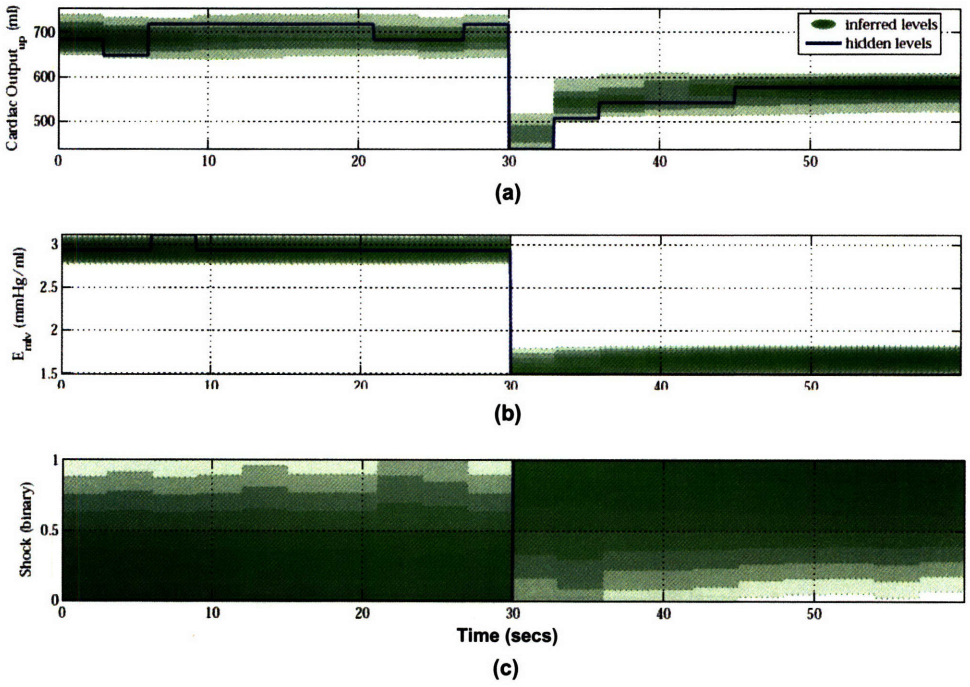

Figure 3-19: Shown in (a) is the CO estimate of a particular simulation, as presented in Hulst [5]. The perturbation of the elastance variable is shown in (b); this is done to simulate cardiogenic shock. Panel (c) shows the binary shock variable itself.

3.6.2 Current Models and Results

The first DBN that was tried is shown in Figure 3-20, where now both HR and ABP are evidential nodes. We once again sample both data sets at approximately half-minute intervals, and quantize the waveforms into 35 bins. The resulting estimates for Pig 9 and Patient b68062 are shown in Figures 3-22 and 3-24, respectively. For comparison, CO estimates calculated from the following equation [24] are also shown:

$$CO_{est} = k \cdot HR \cdot ABP, \tag{3.6}$$

where k is a constant calibration factor that is equal to the ratio of the mean CO of the training data to the mean of the product of HR and ABP of the training data.

The estimated waveforms here are very similar to those obtained from a static Bayesian network, suggesting that ABP contributes significant information to the model. The temporal contributions of the DBN are particularly noticeable in the first 30 minutes of the Pig 9 plot; the DBN estimate is less sensitive to fluctuations in the input signals than the static Bayes net estimate. The simple scaling method also works well here, especially since large spikes in the data often occur concurrently across all three variables, and are thus picked up nicely by a scaling estimate.

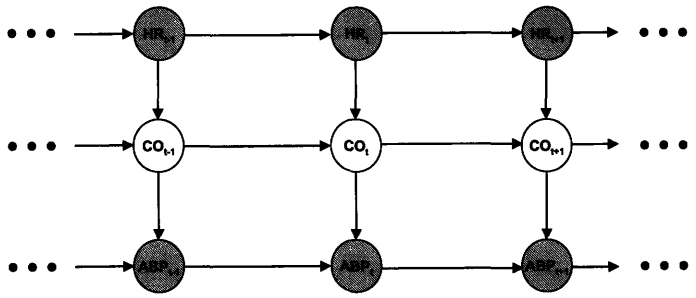

Figure 3-20: DBN relating CO, ABP, and HR. CO is a query node, and ABP and HR are evidential nodes.

Slightly modifying this model to also include drug information leads to the network in Figure 3-26. The evidence supplied to this DBN is shown in Figure 3-27, and the resulting estimated CO is shown in Figure 3-28. Once again, since the MIMIC II database does not have easily extractable medication records, only porcine results are given. The three

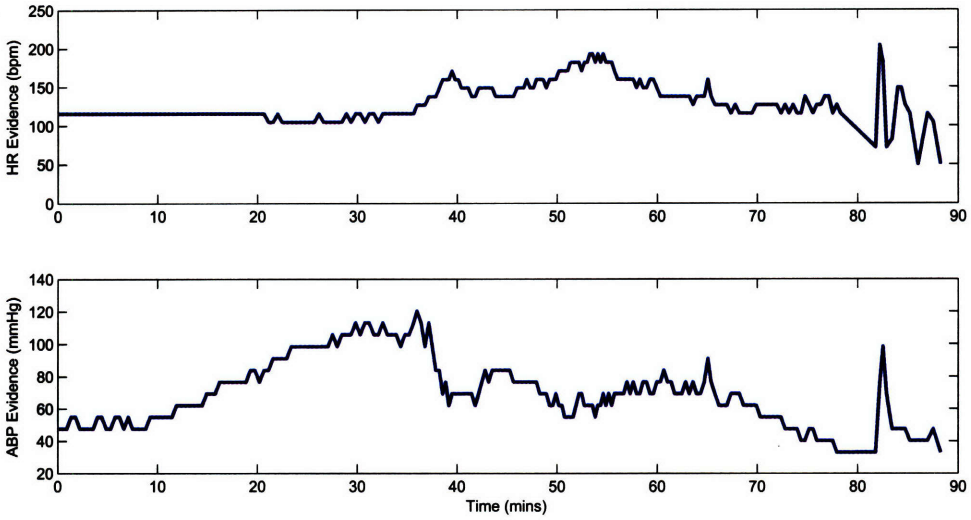

Figure 3-21: The observed sequences of HR and ABP that were used to obtain the CO estimates shown in Figure 3-22.

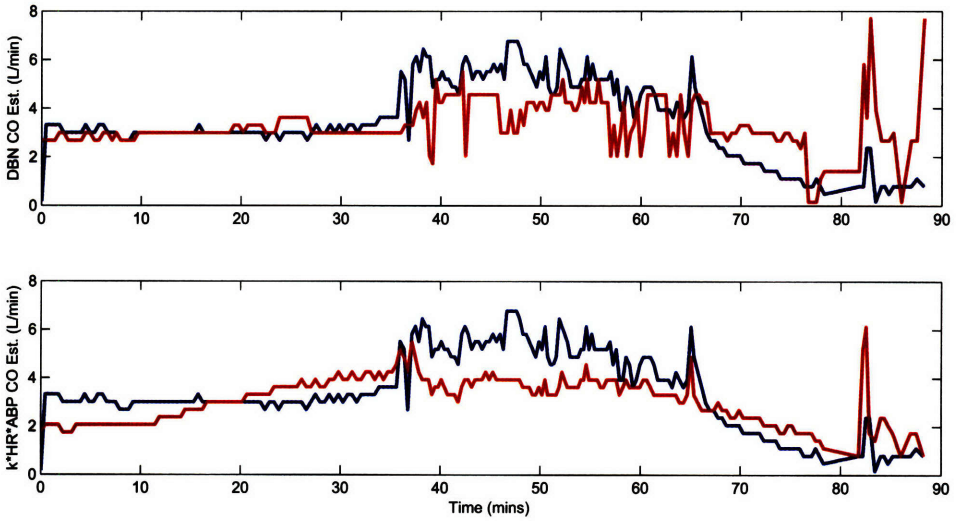

Figure 3-22: Estimated CO for Pig 9 using the DBN in Figure 3-20 is shown in the top panel in red. Estimated CO obtained from Equation 3.6 is shown in the bottom panel in red. Experimentally measured CO is shown in blue. In the top panel, the MANE is 0.61 and the RMSNE is 2.20. In the bottom panel, the MANE is 0.43 and the RMSNE is 1.06.

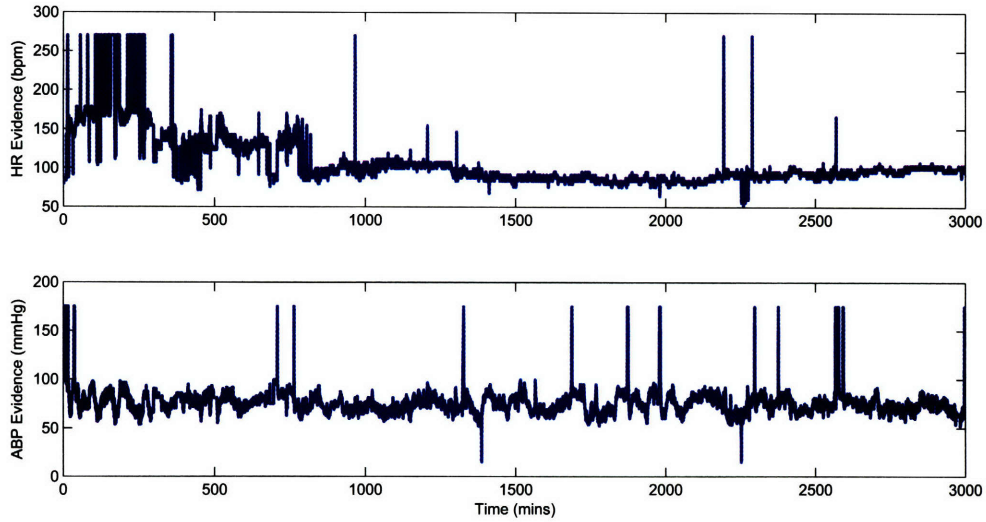

Figure 3-23: The observed sequences of HR and ABP that were used to obtain the CO estimates shown in Figure 3-24.

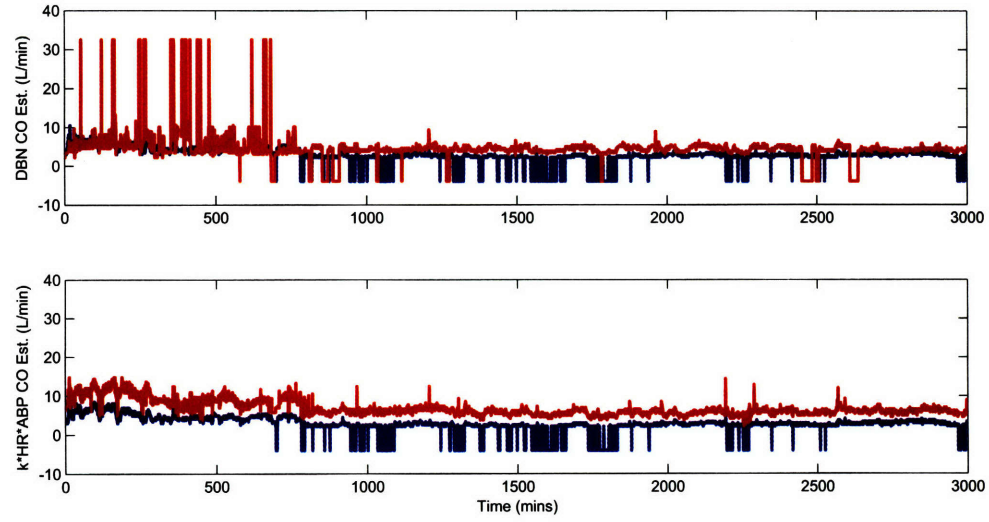

Figure 3-24: Estimated CO for Patient b68062 using the DBN in Figure 3-20 is shown in the top panel in red. Estimated CO obtained from Equation 3.6 is shown in the bottom panel in red. Parlikar's CO estimate is shown in blue. In the top panel, the MANE is 0.79 and the RMSNE is 1.25. In the bottom panel, the MANE is 1.17 and the RMSNE is 1.25.

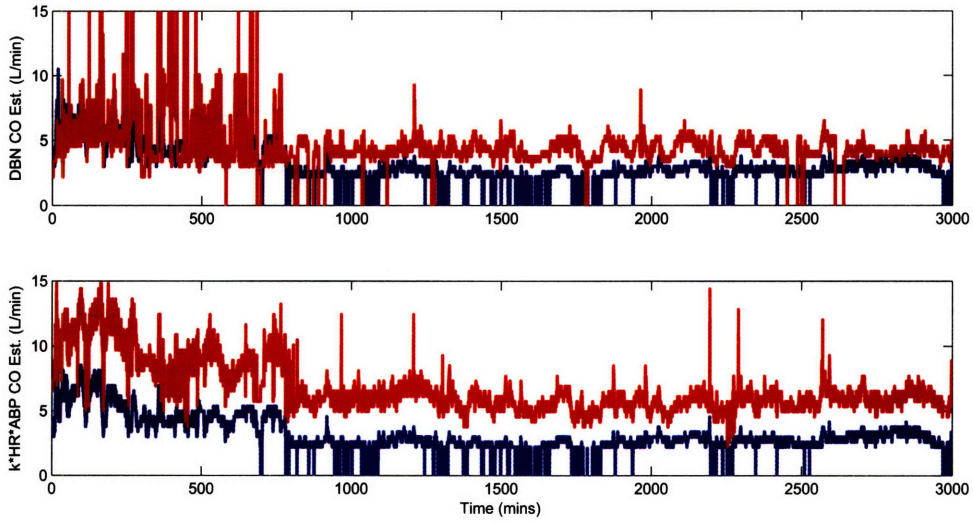

Figure 3-25: Magnified version of Figure 3-24 showing only CO that is in the physiological range.

drugs that we account for are Dobutamine, Esmolol, and Nitroglycerin, although others were given as well.

Once again, the incorporation of drug information does not significantly improve the quality of the estimates, although the errors do go down. In future work, one could consider using the actual dosages, rather than just a binary variable for the drugs.

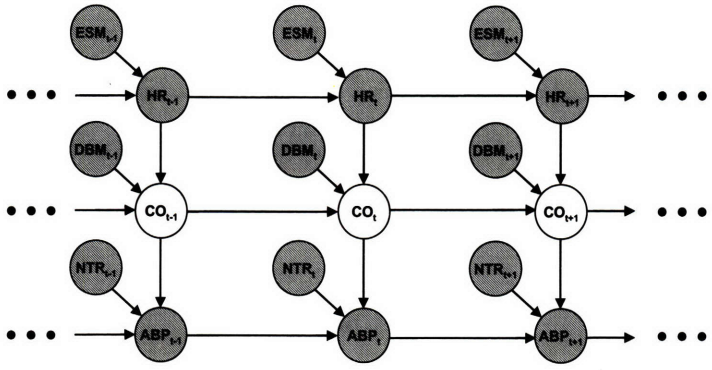

Figure 3-26: DBN relating CO, ABP, HR, and infusions of Esmolol, Dobutamine, and Nitroglycerin.

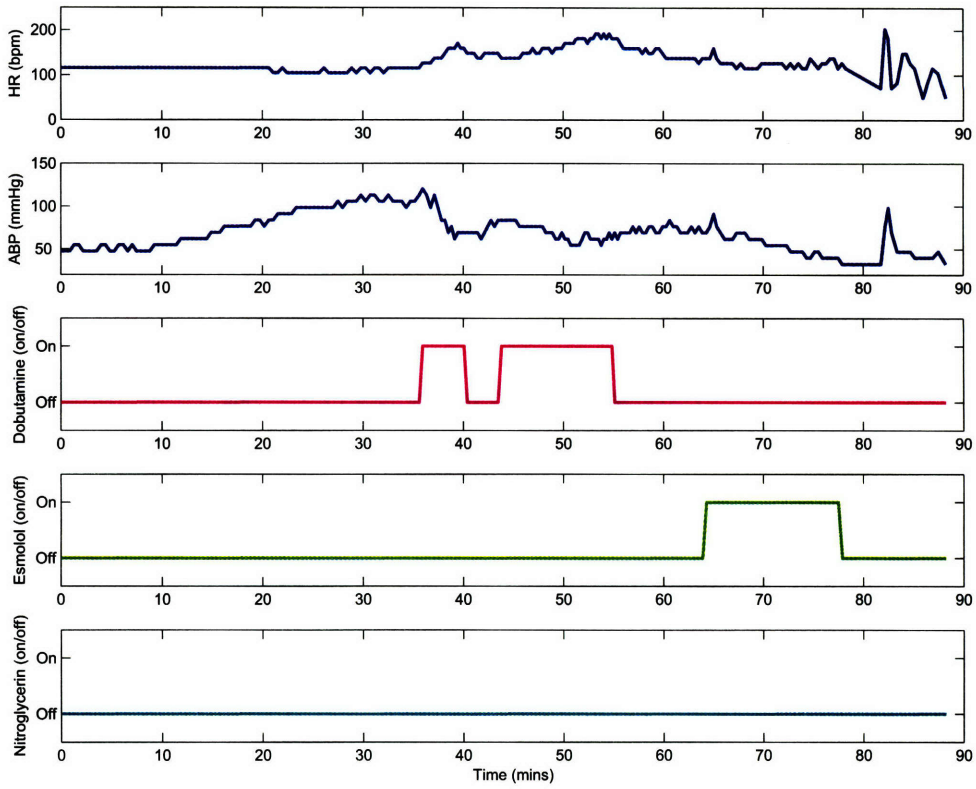

Figure 3-27: Evidence supplied to the DBN in Figure 3-26 to obtain the results shown in Figure 3-28. Nitroglycerin was actually not administered to Pig 9, but was given to the other animals.

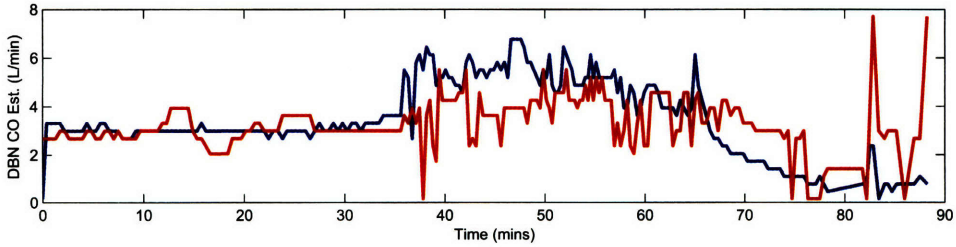

Figure 3-28: Estimated CO for Pig 9 using the DBN in Figure 3-26 is shown in red. Experimentally measured CO is shown in blue. The MANE is 0.59 and the RMSNE is 1.88.

3.7 Discussion of Results

The errors from applying each method presented in this chapter to Pig 9 and Patient b68062 are summarized in Tables 3.1 and 3.2, respectively. There is no clear pattern in the results for Pig 9, but it is clear from Patient b68062's results that ABP is a very important variable to consider when estimating CO. As for comparing the performance of our networks with that of deterministic scaling methods, the errors are in the same ball park.

In order to obtain more representative errors for the entire data set, we test our network models on 12 randomly chosen patients from the MIMIC II database, after training the networks on the other 108 patients. These results are stated in Table 3.3. We do not repeat this analysis for the porcine data set because it is too small. The mean errors once again suggest that ABP is a critically important variable to consider, much more so than the temporal correlations in the data.

We also point out that the quality of the estimates varies a great deal between subjects. For the 12 patients in Table 3.3, the DBN MANE values range from 0.25 to 1.00. This suggests that the training data may not be rich enough, since large errors often occur when certain outcome combinations are not encountered during parameter learning. Using training data with higher variability, or intelligently choosing the CPD priors, would aid in forming more robust networks, and consequently, in producing more consistent results.

Model	**Evidence Variables**	**MANE**	**RMSNE**
Static Bayes Net	HR, ABP	0.63	1.80
Single Chain HMM	HR	0.58	1.96
Multiple Output HMM	HR, DBM	0.60	1.97
Autoregressive HMM	HR	0.52	1.54
DBN	HR, ABP	0.61	2.20
DBN with Medications	HR, ABP, DBM, ESM, NTR	0.59	1.88
k-HR	HR	0.61	1.76
k-HR-ABP	HR, ABP	0.43	1.06

Table 3.1: Summary of results for Pig 9.

Model	Evidence Variables	MANE	RMSNE
Static Bayes Net	HR, ABP	0.72	1.09
Single Chain HMM	HR	5.08	6.07
Autoregressive HMM	HR	3.63	4.76
DBN	HR, ABP	0.79	1.25
k-HR	HR	1.11	1.22
k-HR-ABP	HR, ABP	1.17	1.25

Table 3.2: Summary of results for Patient b68062.

Patient No.	Static BN	S.C. HMM	A.R. HMM	DBN	k-HR	k-HR-ABP
b71758	0.18	1.49	1.88	0.63	0.16	0.15
b65508	0.47	0.73	1.60	0.48	0.34	0.49
b67092	0.57	0.88	1.36	0.66	0.51	0.53
b61862	0.83	4.87	2.68	1.00	1.09	1.22
b68134	0.46	1.57	1.63	0.56	0.26	0.31
b61947	0.55	1.27	1.32	0.72	0.42	0.44
b60734	0.30	0.76	2.32	0.25	0.29	0.23
b62761	0.29	1.91	2.92	0.38	0.33	0.27
b70776	0.26	0.57	2.04	0.38	0.14	0.30
b73616	0.35	1.14	1.69	0.65	0.17	0.22
b73687	0.30	2.04	2.19	0.39	0.16	0.16
b74034	0.59	1.79	3.49	0.79	0.66	0.82
Min Error	0.18	0.57	1.32	0.25	0.14	0.15
Max Error	0.83	4.87	3.49	1.00	1.09	1.22
Mean Error	0.43	1.58	2.09	0.57	0.38	0.43

Table 3.3: Comparison of MANE in MIMIC II data. The networks used here are shown in Figures 3-4, 3-8, 3-14, and 3-20, respectively.

Chapter 4

Conclusion

4.1 Summary

In this thesis, we presented probabilistic networks as a method for capturing physiological correlations in the parameters of the cardiovascular system.

In particular, Chapter 2 summarized in a unifying manner the theory behind three types of probabilistic networks: Bayes nets, HMMs, and DBNs. The semantics of each type of network was defined, and methods for parameter learning and probabilistic inference were explained. We hope that this chapter can serve as a self-contained tutorial for anyone interested in learning about this topic.

In Chapter 3, we developed several networks, with varying degrees of complexity, for computing time series estimates of CO. We tested the networks on both a set of experimental porcine data and a set of real ICU patient data. We found that ABP and HR are critical evidence nodes for CO estimation, and that models with higher complexity were more capable of following rapid fluctuations in the data than simpler models with only one input variable. We also compared the results of our networks with those calculated using a simple scaling method, and found that the errors were comparable.

The primary contributions of this thesis were to provide a detailed explanation of probabilistic networks as a modeling technique, and to demonstrate their performance on real data. Although these networks did not significantly outperform simpler, physiologically-based formulas for estimating CO, it is a purely statistical framework that holds great promise in applications where good deterministic models do not yet exist.

4.2 Future Work

A natural extension of the work presented in this thesis is to explore more complex network models of the cardiovascular system. Such networks could include additional variables, such as electrocardiogram readings and lab results, or higher order temporal correlations. Additional exploration of parameters, such as the sampling rate and number of quantization levels, would also be beneficial.

Another topic that could yield interesting results is parameter learning with missing data. It is often unrealistic to ask for a complete set of training data, as values for variables such as CO are frequently missing. Training the networks with missing data, which involves using the expectation maximization (EM) algorithm, could be a possible solution.

A topic that is related to missing data is the use of informative priors. In this thesis, we always initialized the network CPDs with uniform prior distributions. If the data set is sparse, or contains missing data, then it is critically important that a good prior is chosen.

Lastly, it would be highly advantageous to explore these networks in the context of larger, richer data sets, which are admittedly hard to come by. Databases that are differentiated by certain diseases or conditions would be especially interesting and helpful.

Bibliography

[1] T. Jaakkola. 6.867 Machine Learning Lecture 21. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, URL: http://courses.csail.mit.edu/6.867/lectures/lecture-21.pdf, Fall 2006.

[2] T. Jaakkola. Slides on the junction tree algorithm, April 2007. Personal communication.

[3] J. M. Roberts, T. A. Parlikar, T. Heldt, and G. C. Verghese. Bayesian networks for cardiovascular monitoring. In *Proceedings of the 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 205–209, August 2006.

[4] J. M. Roberts. Bayesian networks for cardiovascular monitoring. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 2006.

[5] J. Hulst. Modeling physiological processes with dynamic Bayesian networks. Master's thesis, Delft University of Technology, Faculty of Electrical Engineering, Mathematics, and Computer Science, August 2006.

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

[7] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[8] E. Castillo, J. M. Gutierrez, and A. Hadi. *Expert Systems and Probabilistic Network Models.* Springer-Verlag, New York, NY, 1997.

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

[10] D. Spiegelhalter and S. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

[11] D. Heckerman. A tutorial on learning with Bayesian networks. *Microsoft Research Technical Report MSR-TR-95-06*, November 1996.

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

[13] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. Technical Report TR-97-021, International Computer Science Institute, April 1998.

[14] K. P. Murphy. A brief introduction to graphical models. University of California, Berkeley, URL: http://www.cs.ubc.ca/ murphyk/Bayes/bayes.html, May 2001.

[15] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 11:1–158, 1994.

[16] M. I. Jordan. The junction tree algorithm: Lecture notes. University of California, Berkeley, November 2004.

[17] E. Alpaydin. *Introduction to Machine Learning.* MIT Press, Cambridge, MA, 2004.

[18] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, Computer Science Division, Fall 2002.

[19] L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, February 1989.

[20] T. Xuan. Autoregressive Hidden Markov Model with application in an El Niño study. Master's thesis, University of Saskatchewan, Department of Mathematics and Statistics, December 2004.

[21] A. Berchtold. The double chain Markov model. Technical Report 348, University of Washington, Department of Statistics, February 1999.

[22] R. Mukkamala, A. T. Reisner, H. M. Hojman, R. G. Mark, and R. J. Cohen. Continuous cardiac output monitoring by peripheral blood pressure waveform analysis. *IEEE Transactions on Biomedical Engineering*, 53:459–467, March 2006.

[23] M. Saeed, C. Lieu, G. Raber, and R. G. Mark. MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in Cardiology*, pages 641– 644, September 2002.

[24] T. A. Parlikar. *Modeling and Monitoring of Cardiovascular Dynamics for Patients in Critical Care.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2007.

[25] 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.

[26] K. P. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics: Proceedings of the Interface*, 33, 2001.