Learning Dynamic Bayesian Networks

Zoubin Ghahramani

Department of Computer Science
University of Toronto
Toronto, ON M5S 3H5, Canada
zoubin@cs.toronto.edu
http://www.cs.utoronto.ca/~zoubin/

August, 1997
DRAFT

Abstract. Bayesian networks are directed acyclic graphs that represent dependencies between variables in a probabilistic model. Many time series models, including the hidden Markov models (HMMs) used in speech recognition and Kalman filter models used in filtering and control applications, can be viewed as examples of dynamic Bayesian networks. We first provide a brief tutorial on learning and Bayesian networks. We then present some dynamic Bayesian networks that can capture much richer structure than HMMs and Kalman filters, including spatial and temporal multiresolution structure, distributed hidden state representations, and multiple switching linear regimes. While exact probabilistic inference is intractable in these networks, one can obtain tractable variational approximations which call as subroutines the forward-backward and Kalman filter recursions. These approximations can be used to learn the model parameters by maximizing a lower bound on the likelihood.

Table of Contents

1 Introduction 2

2 A Bayesian network tutorial 2

3 Dynamic Bayesian networks 5
3.1 Example 1: State-space models 6
3.2 Example 2: Hidden Markov models 7

4 Learning and Inference 8
4.1 ML Estimation with Complete Data 9
4.2 ML Estimation with Hidden Variables: The EM algorithm . . . 9
4.3 Example 1: Learning state-space models 11
4.4 Kalman smoothing 12
4.5 Example 2: Learning hidden Markov models 13
4.6 The forward–backward algorithm 14

5 Beyond Tractable Models 15

5.1 Example 3: Factorial HMMs 16
5.2 Example 4: Tree structured HMMs 17
5.3 Example 5: Switching State space models 18

6 Inference and Intractability 20
6.1 Gibbs sampling 20
6.2 Variational Methods 21
6.3 Example: Mean field for factorial HMMs 22
6.4 Example: Structured approximation for factorial HMMs 24
6.5 Convex duality 26

7 Conclusion 28

1 Introduction

Suppose we wish to build a model of data from a finite sequence of ordered observations, $\{Y_1, Y_2, \ldots, Y_t\}$. In most realistic scenarios, the observations, whether physiological data or stock prices, are not related deterministically. We can express the uncertainty in our beliefs about the future outcome Y_{t+1} via a probability density $P(Y_{t+1}|Y_1, \ldots, Y_t)$. Such a probability density can then be used to make predictions, define error bars, or make decisions that are expected to minimize some loss function.

In this paper we present a probabilistic framework for learning models of temporal data. We express these models using the Bayesian network formalism (a.k.a. probabilistic graphical models or belief networks)—a marriage of probability theory and graph theory in which dependencies between variables are expressed graphically. The graph not only allows the user to understand which variables affect which other ones, but also serves as the backbone for efficiently computing marginal and conditional probabilities that may be required for inference and learning.

The next section provides a brief tutorial of Bayesian networks. Section 3 demonstrates the use of Bayesian networks for modeling time series, including some well-known examples such as the Kalman filer and the hidden Markov model. Section 4 focuses on the problem of learning the parameters of a Bayesian network using the Expectation–Maximization (EM) algorithm [3, 10]. Section 5 describes some richer models appropriate for time series with nonlinear or multiresolution structure. Inference in such models may be computationally intractable. However, in section 6 we present several tractable methods for approximate inference which can be used as the basis for learning.

2 A Bayesian network tutorial

A Bayesian network is simply a graphical model for representing conditional independencies between a set of random variables. Consider four variables, W,

X, Y, and Z. From basic probability theory we know that we can factor the joint probability as a product of conditional probabilities:

$$P(W, X, Y, Z) = P(W)P(X|W)P(Y|W, X)P(Z|W, X, Y).$$

This factorization does not tell us anything useful about the joint probability distribution: each variable can potentially depend on every other variable. However, consider the following factorization:

$$P(W, X, Y, Z) = P(W)P(X)P(Y|W)P(Z|X, Y). \tag{1}$$

The above factorization implies a set of conditional independence relations. The variable (or set of variables) A is *conditionally independent* from B given C if $P(A, B|C) = P(A|C)P(B|C)$ for all A,B and C such that $P(C) \neq 0$. From the above factorization we can show that given the values of X and Y, Z and W are independent:

$$\begin{aligned} P(Z, W|X, Y) &= \frac{P(W, X, Y, Z)}{P(X, Y)} \\ &= \frac{P(W)P(X)P(Y|W)P(Z|X, Y)}{\int P(W)P(X)P(Y|W)P(Z|X, Y)\ dW\ dZ} \\ &= \frac{P(W)P(Y|W)P(Z|X, Y)}{P(Y)} \\ &= P(W|Y)P(Z|X, Y) \end{aligned}$$

A Bayesian network is a graphical way to represent a particular factorization of a joint distribution. Each variable is represented by a node in the network. A directed arc is drawn from node A to node B if B is conditioned on A in the factorization of the joint distribution. For example, to represent the factorization (1) we would draw an arc from W to Y but not from W to Z. The Bayesian network representing the factorization **??** is shown in Figure 1.

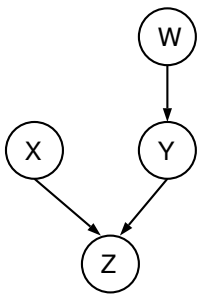

Fig. 1. A directed acyclic graph (DAG) consistent with the conditional independence relations in $P(W, X, Y, Z)$.

Some basic definitions from graph theory will be necessary at this point. The node A is a *parent* of another node B if there is a directed arc from A to B; if so, B is a *child* of A. The *descendents* of a node are its childen, children's childen, and so on. A *directed path* from A to B is a sequence of nodes starting from A and ending in B such that each node in the sequence is a parent of the following node in the sequence. An *undirected path* from A to B is a sequence of nodes starting from A and ending in B such that each node in the sequence is a parent *or child* of the following node.

The semantics of a Bayesian network are simple: each node is conditionally independent from its non-descendents given its parents.[1] More generally, two disjoint sets of nodes A and B are conditionally independent given C, if C *d-separates* A and B, that is, if along every undirected path between a node in A and a node in B there is a node D such that: (1) D has converging arrows[2] and neither D nor its descendents are in C, or (2) D does not have converging arrow and D is in C [38]. From visual inspection of the graphical model it is therefore easy to infer many independence relations without explicitly grinding through Bayes rule. For example, W is conditionally independent from X given the set $C = \{Y, Z\}$, since $Y \in C$ is along the only path between W and X, and Y does not have converging arrows. However, we cannot infer from the graph that W is conditionally independent from X given Z.

Notice that since each factorization implies a strict ordering of the variables, the connections obtained in this manner define a directed *acyclic* graph[3]. Furthermore, there are many ways to factorize a joint distribution, and consequently there are many Bayesian networks consistent with a particular joint. A Bayesian network G is said to be an independency map *I-map* for a distribution P if every d-separation displayed in G corresponds to a valid conditional independence relation in P. G is a *minimal I-map* if no arc can be deleted from G without removing the *I-map* property.

The absence of arcs in a Bayesian networks implies conditional independence relations which can be exploited to obtain efficient algorithms for computing marginal and conditional probabilities. For *singly connected* networks, in which the underlying undirected graph has no loops, there exists a general algorithm called *belief propagation* [29, 38]. For *multiply connected* networks, in which there can be more than one undirected path between any two nodes, there exists a more general algorithm known as the *junction tree algorithm* [30, 23]. I will provide the essence of the belief propagation algorithm (since the exact methods used throughout this paper are based on it) and refer the reader to relevant textbooks [38, 22] for details.

[1] Since there is a one-to-one correspondence between nodes and variables, we will often talk about conditional independence relations between nodes meaning conditional independence relations between the variables associated with the nodes.

[2] That is, D is a child of both the previous and following nodes in the path.

[3] Undirected graphical models (Markov networks) are another important tool for representing probability distributions, and have a different set of semantics [5, 13]. We will deal exclusively with directed graphical models in this paper.

Assume we observe some *evidence*: the value of some variables in the network. The goal of belief propagation is to update the marginal probabilities of all the variables in the network to incorporate this new evidence. This is achieved by local message passing: each node, n sends a message to its parents and to its children. Since the graph is singly connected, n separates the graph into two mutually exclusive sets: $e^+(n)$, consisting of the parents of n, the nodes connected to n through its parents[4], and n itself, and $e^-(n)$ consisting of the children of n and the nodes connected to n through its children. The message from n to each of its children is the probability of each setting of n given the evidence observed in the set $e^+(n)$. The message from n to each of its parents is the probability, given every setting of the parent, of the evidence observed in the set $e^-(n) \cup \{n\}$. The marginal probability of a node is proportional to the product of the messages obtained from its parents, weighted by the conditional probability of the node given its parents, and the message obtained from its children. If the parents of n are $\{p_1, \ldots, p_k\}$ and the childen of n are $\{c_1, \ldots, c_\ell\}$, then

$$P(n|e) \propto \left[\sum_{\{p_1,\ldots,p_k\}} P(n|p_1,\ldots,p_k) \prod_{i=1}^{k} P(p_i|e^+(p_i)) \right] \prod_{j=1}^{\ell} P(c_j, e^-(c_j)|n) \quad (2)$$

where the summation (or more generally the integral) extends over all settings of $\{p_1, \ldots, p_k\}$. For example, given the evidence $e = \{X = x, Z = z\}$,

$$P(Y|X = x, Z = z) \propto \left[\int P(Y|W)P(W)\, dW \right] P(Z = z, X = x|Y) \quad (3)$$

$$\propto P(Y)\ P(Z = z|X = x, Y)\ P(X = x) \quad (4)$$

where $P(W)$ is the message passed from W to Y since $e^+(W) = \emptyset$, and $P(Z = z, X = x|Y)$ is the message passed from Z to Y.

It is useful to distinguish between *observable* variables, which constitute the evidence, and *hidden* variables, whose probability must be inferred from the evidence. Often a Bayesian network is constructed by combining *a priori* knowledge about conditional independences between the variables, perhaps from an expert in a particular domain, and a data set of observations. A natural way in which this *a priori* knowledge can be elicited from the expert is by asking questions regarding "causality:" a variable that has a direct causal effect on another variable will be its parent in the network. This notion therefore plays a central role in the design of dynamic Bayesian networks.

3 Dynamic Bayesian networks

In time series modeling, we observe the values of certain variables at different points in time. The assumption that an event can cause another event in the future, but not vice-versa, simplies the design of Bayesian networks for time

[4] That is, the nodes for which the undirected path to n goes through a parent of n.

series: directed arcs should flow forward in time. Assigning a time index t to each variable, one of the simplest causal models for a sequence of data $\{Y_1, \ldots, Y_T\}$ is a *first-order Markov model*, in which each variable is directly influenced only by the previous variable (Figure 2):

$$P(Y_1, Y_2, \ldots, Y_t) = P(Y_1)P(Y_2|Y_1) \cdots P(Y_T|Y_{T-1})$$

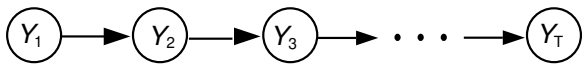

Fig. 2. A Bayesian network representing a first-order Markov process.

These models are impoverished in that they cannot capture dependencies between observables over more than one time step. One simple way of extending Markov models is to allow higher order interactions between variables. For example, a τ^{th}-order Markov model allows arcs from $\{Y_{t-\tau}, \ldots, Y_{t-1}\}$ to Y_t. Another way to extend Markov models is to posit that the observations are dependent on a hidden variable, which we will call the *state*, and that the sequence of *states* is a Markov process (Figure 3). A classic model of this kind is the linear-Gaussian state-space model, also known as the Kalman filter.

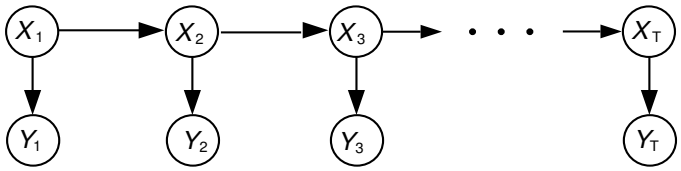

Fig. 3. A Bayesian network specifying conditional independence relations for a state-space model.

3.1 Example 1: State-space models

In state-space models, a sequence of D-dimensional real-valued observation vectors $\{Y_1, \ldots, Y_T\}$, is modeled by assuming that at each time step Y_t was generated from a K-dimensional real-valued hidden state variable X_t, and that the sequence of X's define a first-order Markov process. Using the short-hand notation $\{Y_t\}$ to denote sequences from $t = 1$ to $t = T$:

$$P(\{X_t, Y_t\}) = P(X_1)P(Y_1|X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})P(Y_t|X_t). \tag{5}$$

The state transition probability $P(X_t|X_{t-1})$ can be decomposed into deterministic and stochastic components:

$$X_t = f_t(X_{t-1}) + w_t$$

where f_t is the deterministic transition function determining the mean of X_t given X_{t-1}, and w_t is a zero-mean random noise vector. Similarly, the observation probability $P(Y_t|X_t)$ can be decomposed as

$$Y_t = g_t(X_t) + v_t.$$

If both the transition and output functions are linear and time-invariant and the distribution of the states and observation noise variables is Gaussian, the model becomes a linear-Gaussian state-space model:

$$X_t = AX_{t-1} + w_t \tag{6}$$

$$Y_t = CX_t + v_t \tag{7}$$

where A is the state transition matrix and C is the observation matrix.

Often, the observations can be divided into a set of input (or predictor) variables and output (or response) variables. Again, assuming linearity and Gaussian noise we can write the state transition function as

$$X_t = AX_{t-1} + BU_t + w_t, \tag{8}$$

where U_t is the input observation vector and B is the input matrix. The Bayesian network corresponding to this model would include a sequence of nodes $\{U_t\}$ each of which is a parent of the corresponding X_t. Linear-Gaussian state-space models are used extensively in all areas of control and signal processing.

3.2 Example 2: Hidden Markov models

In a hidden Markov model (HMM), the sequence of observations $\{Y_t\}$ is modeled by assuming that each observation depends on a *discrete* hidden state S_t, and that the sequences of hidden states are distributed according to a Markov process. The joint probability for the sequences of states and observations, can be factored in exactly the same manner as equation (5), with S_t taking the place of X_t:

$$P(\{S_t, Y_t\}) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t). \tag{9}$$

Consequently, the conditional independences in an HMM can also be expressed graphically using the Bayesian network shown in Figure 3. The state is represented by a single multinomial variable that can take one of K discrete values, $S_t \in \{1, \ldots, K\}$. The state transition probabilities, $P(S_t|S_{t-1})$, for a time-invariant HMM can be specified by a single $K \times K$ transition matrix. If the observables are discrete symbols taking on one of L values, the emission probabilities $P(Y_t|S_t)$ can be fully specified by a $K \times L$ observation matrix. For real-valued observation vectors, $P(Y_t|S_t)$ can be modeled in many different forms,

such as a Gaussian, mixture of Gaussians, or a neural network. Like state-space models, HMMs can be augmented to allow for input variables [7, 4, 33]. The system then models the conditional distribution of a sequence of output observations given a sequence of input observations. HMMs have been applied extensively to problems in speech recognition [26], computational biology [2], and fault detection [45].

4 Learning and Inference

A Bayesian approach to learning starts with some *a priori* knowledge about the model structure—the set of arcs in the Bayesian network—and model parameters. This initial knowledge is represented in the form of a prior probability distribution over model structures and parameters, and updated using the data to obtain a posterior probability distribution over models and parameters. More formally, assuming a prior distribution over models structures $P(\mathcal{M})$ and a prior distribution over parameters for each model structure $P(\theta|\mathcal{M})$, a data set $\mathcal{D}$ is used to form a posterior distribution over models using Bayes rule

$$P(\mathcal{M}|\mathcal{D}) = \frac{\int P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})\ d\theta\ P(\mathcal{M})}{P(\mathcal{D})}.$$

For a given model structure, we can compute the posterior distribution over the parameters:

$$P(\theta|\mathcal{M},\mathcal{D}) = \frac{\int P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})\ d\theta}{P(\mathcal{D}|\mathcal{M})}.$$

If the data set is some sequence of observations $\mathcal{D} = \{Y_1,\ldots,Y_T\}$ and we wish to predict the next observation, Y_{T+1} based on our data and models, then the Bayesian prediction

$$P(Y_{T+1}|\mathcal{D}) = \int P(Y_{T+1}|\theta,\mathcal{M},\mathcal{D})P(\theta|\mathcal{M},\mathcal{D})P(\mathcal{M}|\mathcal{D})\ d\theta\ d\mathcal{M}$$

integrates out the uncertainty in the model structure and parameters.

We obtain a somewhat impoverished by nonetheless useful limiting case of the Bayesian approach to learning if we assume a single model structure $\mathcal{M}$ and we estimate the parameters $\hat{\theta}$ that maximize the likelihood $P(\mathcal{D}|\theta,\mathcal{M})$ under that model. In the limit of a large data set and an uninformative (i.e. uniform) prior over the parameters, the posterior $P(\theta|\mathcal{M},\mathcal{D})$ will be sharply peaked around the maxima of the likelihood, and therefore the predictions of a single maximum likelihood (ML) model will be similar to those obtained by Bayesian integration over the parameters.

We focus in this paper on the problem of estimating ML parameters for a model given the model structure. Although in principle this is an only approximate Bayesian learning, in practice a full-fledged Bayesian analysis is often impractical[5]. Furthermore, in many application areas there is strong a priori

[5] Two approximate methods for integrating over the posterior in the case of neural network models are described in [32] and [35].

knowledge about the model structure and a single estimate of the parameters provides a more parsimonious and interpretable model than a distribution over parameters.

4.1 ML Estimation with Complete Data

Assume a data set of independent and identically distributed observations $\mathcal{D} = \{Y^{(1)}, \ldots, Y^{(N)}\}$, each of which can be a vector or time series of vectors, then the likelihood of the data is:

$$P(\mathcal{D}|\theta, \mathcal{M}) = \prod_{i=1}^{N} P(Y^{(i)}|\theta, \mathcal{M})$$

For notational convenience we henceforth drop the implicit conditioning on the model structure, $\mathcal{M}$. The ML parameters are obtained by maximing the likelihood, or equivalently the log likelihood:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log P(Y^{(i)}|\theta).$$

If the observation vector includes all the variables in the Bayesian network, then each term in the likelihood further factors as:

$$\log P(Y^{(i)}|\theta) = \log \prod_j P(Y_j^{(i)}|Y_{\mathrm{pa}(j)}^{(i)}, \theta_j) \tag{10}$$

$$= \sum_j \log P(Y_j^{(i)}|Y_{\mathrm{pa}(j)}^{(i)}, \theta_j), \tag{11}$$

where j indexes over the nodes in the Bayesian network, $\mathrm{pa}(j)$ is the set of parents of j, and θ_j are the parameters that define the conditional probability of Y_j given its parents. The likelihood therefore decouples into local terms involving each node and its parents, simplifying the ML estimation problem. For example, if the Y variables are discrete and θ_j is the conditional probability table for Y_j given its parents, then the ML estimate of θ_j is simply a normalized table containing counts of each setting of Y_j given each setting of its parents in the data set.

4.2 ML Estimation with Hidden Variables: The EM algorithm

With hidden variables the log likelihood cannot be decomposed as in (11). Rather, we find:

$$\mathcal{L}(\theta) = \log P(Y|\theta) = \log \sum_X P(Y, X|\theta) \tag{12}$$

where X is the set of hidden variables, and $\sum_X$ is the sum (or integral) over X required to obtain the marginal probability of the data. (We have dropped the superscript (i) in (12) by evaluating the likelihood for a single observation.)

Using any distribution Q over the hidden variables, we can obtain a lower bound on $\mathcal{L}$:

$$\log \sum_X P(Y, X|\theta) = \log \sum_X Q(X) \frac{P(Y, X|\theta)}{Q(X)} \tag{13}$$

$$\geq \sum_X Q(X) \log \frac{P(X, Y|\theta)}{Q(X)} \tag{14}$$

$$= \sum_X Q(X) \log P(X, Y|\theta) - \sum_X Q(X) \log Q(X) \tag{15}$$

$$= \mathcal{F}(Q, \theta) \tag{16}$$

where the middle inequality is known as Jensen's inequality and can be proven using the concavity of the log function. If we define the *energy* of a global configuration (X, Y) to be $\log P(X, Y|\theta)$, then some readers may notice that the lower bound $\mathcal{F}(Q, \theta) \leq \mathcal{L}(\theta)$ is the negative of a quantity known in statistical physics as the *free energy*: the expected energy under Q minus the entropy of Q [36]. The Expectation-Maximization (EM) algorithm [3, 10] alternates between maximizing $\mathcal{F}$ with respect to Q and θ, respectively, holding the other fixed. Starting from some initial parameters θ_0:

$$\textbf{E step:} \qquad Q_{k+1} \leftarrow \arg\max_{\mathrm{Q}} \mathcal{F}(Q, \theta_k) \tag{17}$$

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \mathcal{F}(Q_{k+1}, \theta) \tag{18}$$

It is easy to show that the maximum in the E step results when $Q_{k+1}(X) = P(X|Y, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(Q_{k+1}, \theta_k) = \mathcal{L}(\theta_k)$. The maximum in the M step is obtained by maximizing the the first term in (15), since the entropy of Q does not depend on θ:

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \sum_X P(X|Y, \theta_k) \log P(X, Y|\theta).$$

This is the expression most often associated with the EM algorithm, but it obscures the elegant interpretation of EM as coordinate ascent in $\mathcal{F}$. Since $\mathcal{F} = \mathcal{L}$ at the beginning of each M step, and since the E step does not change θ, we are guaranteed not to decrease the likelihood after each combined EM step.

It is worthwhile to point out that it is usually not necessary to explicitly evaluate the posterior distribution $P(X|Y, \theta_k)$. Since $\log P(X, Y|\theta)$ contains both hidden and observed variables in the network, it can be factored as before as the sum of log probabilities of each node given its parents. Consequently, the quantities required for the M step are the expected values, under the posterior distribution $P(X|Y, \theta_k)$, of the analogous quantities required for ML estimation in the complete data case.

4.3 Example 1: Learning state-space models

Using equation (5), the log probability of the hidden states and observations for linear-Gaussian state-space models can be written as

$$\log P(\{X_t, Y_t\}) = \log P(X_1) + \sum_{t=1}^{T} \log P(Y_t|X_t) + \sum_{t=2}^{T} \log P(X_t|X_{t-1}). \quad (19)$$

Each of the above probability densities is Gaussian, and therefore the overall expression is a sum of quadratics. For example, using equation (7):

$$\log P(Y_t|X_t) = -\frac{1}{2}(Y_t - CX_t)'R^{-1}(Y_t - CX_t) - \frac{1}{2}|R| + const$$

where R is the covariance of the observation noise v_t, $'$ is the matrix transpose, and $|\cdot|$ is the matrix determinant.

If the all the random variables were observed, then the ML parameters could be solved for by maximizing (19). Taking derivatives of (19) we obtain a linear systems of equations. The ML estimate of the matrix C, for example, would be

$$C \leftarrow \left(\sum_t Y_t X_t'\right) \left(\sum_t X_t X_t'\right)^{-1}.$$

Since the states are in fact hidden, in the M step we use expected values wherever we don't have access to the actual observed values. Let us denote the expected value of some quantity $f(X)$ with respect to the posterior distribution of X by $\langle f(X) \rangle$,

$$\langle f(X) \rangle = \int_X f(X)\ P(X|Y, \theta_k)\ dX. \quad (20)$$

Then, the M step for C is

$$C \leftarrow \left(\sum_t Y_t \langle X_t \rangle'\right) \left(\sum_t \langle X_t X_t' \rangle\right)^{-1}.$$

Similar M steps can be derived for all the other parameters by taking derivatives of the expected log probability [44, 11, 15].[6] In general we require all terms of the kind $\langle X_t \rangle$, $\langle X_t X_t' \rangle$ and $\langle X_t X_{t-1}' \rangle$. These terms can be computed using the Kalman smoothing algorithm.

[6] The parameters of a linear-Gaussian state-space model can also be estimated using methods from on-line recursive identification [31].

4.4 Kalman smoothing

The Kalman smoother solves the problem of estimating the state at time t of a linear-Gaussian state-space model given the model parameters and a sequence of observations $\{Y_1, \ldots, Y_t, \ldots, Y_T\}$. It consists of two parts: a forward recursion which uses the observations from Y_1 to Y_t, known as the *Kalman filter* [27], and a backward recursion which uses the observations from Y_T to Y_{t+1} [40].[7] We have already seen that in order to compute the marginal probability of a variable in a Bayesian network one must take into account both the evidence above and below the variable. In fact, the Kalman smoother is simply a special case of the belief propagation algorithm we have already encountered for Bayesian networks.

The Gaussian marginal density of the hidden state vector is completely specified by its mean and covariance matrix. To describe the smoothing algorithm it will be useful to define the following quantities: X_t^τ and V_t^τ are respectively the mean and covariance matrix of X_t given observations $\{Y_1, \ldots Y_\tau\}$. The Kalman filter consists of the following forward recursions:

$$X_t^{t-1} = AX_{t-1}^{t-1} \tag{21}$$

$$V_t^{t-1} = AV_{t-1}^{t-1}A' + Q \tag{22}$$

$$K_t = V_t^{t-1}C'(CV_t^{t-1}C' + R)^{-1} \tag{23}$$

$$X_t^t = X_t^{t-1} + K_t(Y_t - CX_t^{t-1}) \tag{24}$$

$$V_t^t = V_t^{t-1} - K_t C V_t^{t-1} \tag{25}$$

where X_1^0 and V_1^0 are initialized as the prior mean and covariance of the state, which are model parameters. Equations (21) and (22) describe the forward propagation of the state mean and variance before having accounted for the observation at time t. The mean evolves according to the known dynamics A which also affects the variance. In addition the variance also increases by Q, the state noise. The observation Y_t has the effect of shifting the mean by an amount proportional to the prediction error $Y_t - CX_t^{t-1}$, where the proportionality term K_t is known as the *Kalman gain matrix*. Observing Y_t also has the effect of reducing the variance of X_t. These equations can all be derived (perhaps laboriously) by analytically evaluating the Gaussian integrals that result when belief propagation is applied to the Bayesian network corresponding to state-space models.

At the end of the forward recursions we have the values for X_T^T and V_T^T. We now need to proceed backwards and evaluate the influence of future observations on our estimate of states in the past:

$$J_{t-1} = V_{t-1}^{t-1}A'(V_t^{t-1})^{-1} \tag{26}$$

$$X_{t-1}^T = X_{t-1}^{t-1} + J_{t-1}(X_t^T - AX_{t-1}^{t-1}) \tag{27}$$

$$V_{t-1}^T = V_{t-1}^{t-1} + J_{t-1}(V_t^T - V_t^{t-1})J_{t-1}' \tag{28}$$

[7] The forward and backward recursions together are also known as the Rauch-Tung-Streibel (RTS) smoother. Thorough treatments of Kalman filtering and smoothing can be found in [1, 18].

where J_t is a gain matrix with a similar role to the Kalman gain matrix. Again, equation (27) shifts the mean by an amount proportional to the prediction error $X_t^T - AX_{t-1}^{t-1}$. We can also compute the covariance across two time steps [44]

$$V_{t,t-1}^T = V_t^t J_{t-1}' + J_t(V_{t+1,t}^T - AV_t^t)J_{t-1}'$$

which is initialized $V_{T,T-1}^T = (I - K_T C)AV_{T-1}^{T-1}$. The expectations required for EM can now be readily computed:

$$\langle X_t \rangle = X_t^T \tag{29}$$

$$\langle X_t X_t' \rangle = X_t^T X_t^{T'} + V_t^T \tag{30}$$

$$\langle X_t X_{t-1}' \rangle = X_t^T X_{t-1}^{T'} + V_{t,t-1}^T. \tag{31}$$

4.5 Example 2: Learning hidden Markov models

The log probability of the hidden variables and observations for an HMM can be written as:

$$\log P(\{S_t, Y_t\}) = \log P(S_1) + \sum_{t=1}^{T} \log P(Y_t|S_t) + \sum_{t=2}^{T} \log P(S_t|S_{t-1}). \tag{32}$$

Let us represent the K-valued discrete state S_t using K-dimensional unit column vectors, e.g. the state at time t taking on the value "2" is represented as $S_t = [010\ldots0]'$. Each of the terms in (32) can be decomposed into summations over S_t. For example, the transition probability is

$$P(S_t|S_{t-1}) = \prod_{i=1}^{K}\prod_{j=1}^{K}(P_{ij})^{S_{t,i}S_{t-1,j}}$$

where P_{ij} is the probability of transitioning from state j to state i, arranged in a $K \times K$ matrix P. Then

$$\log P(S_t|S_{t-1}) = \sum_{i=1}^{K}\sum_{j=1}^{K} S_{t,i}S_{t-1,j} \log P_{ij} \tag{33}$$

$$= S_t'(\log P)S_{t-1} \tag{34}$$

using matrix notation. Similarly, if we assume a vector of initial state probabilities, $\boldsymbol{\pi}$, then

$$\log P(S_1) = S_1' \log \boldsymbol{\pi}.$$

Finally, the emission probabilities depend on the form of the observations. If Y_t is a discrete variable which can take on D values, then we again represent it using D-dimensional unit vectors and obtain

$$\log P(Y_t|S_t) = Y_t'(\log E)S_t$$

where E is a $D \times K$ emission probability matrix.

Since the state variables are hidden we cannot compute (32) directly. The EM algorithm, which in the case of HMMs is known as the Baum-Welch algorithm [3], allows us to circumvent this problem by computing the expectation of (32) under the posterior distribution of the hidden states given the observations. This expectation can be expressed as a function of terms of the kind $\langle S_t \rangle$ and $\langle S_t S'_{t-1} \rangle$. The first term, $\langle S_t \rangle$, is a vector containing the probability that the HMM was in each of the K states at time t given its current parameters and the entire sequence of observations[8]. The second term, $\langle S_t S'_{t-1} \rangle$, is a matrix containing the joint probability that the HMM was in each state at times $t-1$ and t. In the HMM notation of [39], $\langle S_t \rangle$ corresponds to γ_t and $\langle S_t S'_{t-1} \rangle$ corresponds to ξ_t. Given these expectations, the M step is straightforward: we take derivatives of (32) with respect to the parameters, set to zero, and solve subject to the sum-to-one constraints that ensure valid transition, emission and initial state probabilities. For example, for the transition matrix we obtain

$$P_{ij} \propto \sum_{t=2}^{T} \langle S_{t,i} S_{t-1,j} \rangle \tag{35}$$

$$= \frac{\sum_{t=2}^{T} \langle S_{t,i} S_{t-1,j} \rangle}{\sum_{t=2}^{T} \langle S_{t-1,j} \rangle}. \tag{36}$$

The necessary expectations are computed using the forward–backward algorithm.

4.6 The forward–backward algorithm

The forward–backward algorithm is simply belief propagation applied to the Bayesian network corresponding to a hidden Markov model (see [46] for a recent treatment). The forward pass computes for each t the joint probability of S_t and the sequence of observations Y_1 to Y_t, defined as α_t:

$$\alpha_t = P(S_t, Y_1, \ldots, Y_t) \tag{37}$$

$$= \left[\sum_{S_{t-1}} P(S_{t-1}, Y_1, \ldots, Y_{t-1}) P(S_t | S_{t-1}) \right] P(Y_t | S_t) \tag{38}$$

$$= \left[\sum_{S_{t-1}} \alpha_{t-1} P(S_t | S_{t-1}) \right] P(Y_t | S_t). \tag{39}$$

The backward pass computes the conditional probability of the observations Y_{t+1} to Y_T given S_t:

$$\beta_t = P(Y_{t+1}, \ldots, Y_T | S_t) \tag{40}$$

[8] When learning from a data set containing multiple sequences, this quantity has to be computed separately for each sequence. For clarity, we will describe the single sequence case only.

$$= \sum_{S_{t+1}} P(Y_{t+2}, \ldots, Y_T | S_{t+1}) P(S_{t+1}|S_t) P(Y_{t+1}|S_{t+1}) \tag{41}$$

$$= \sum_{S_{t+1}} \beta_{t+1} P(S_{t+1}|S_t) P(Y_{t+1}|S_{t+1}). \tag{42}$$

From these it is easy to compute the expectations needed for EM:

$$\langle S_{t,i} \rangle = \gamma_{ti} = \frac{\alpha_{t,i}\beta_{t,i}}{\sum_i \alpha_{t,i}\beta_{t,i}} \tag{43}$$

$$\langle S_{t,i} S_{t-1,j} \rangle = \xi_{tij} = \frac{\alpha_{t-1,j} P_{ij} P(Y_t|S_{t,i}) \beta_{t,i}}{\sum_{k,\ell} \alpha_{t-1,k} P_{k\ell} P(Y_t|S_{t,\ell}) \beta_{t,\ell}}. \tag{44}$$

Notice that the Kalman smoothing algorithm and the forward–backward algorithm are conceptually identical. Occasionally, it is also useful to compute the single most probable state sequence. The solution to this problem is given by the *Viterbi algorithm* [**?**], which is also very similar to the forward–backward algorithm except that some of the summations are replaced by maximizations (see [39] for a tutorial on HMMs, especially as applied to speech recognition).

5 Beyond Tractable Models

Linear-Gaussian state-space models and hidden Markov models provide an interesting starting point for designing dynamic Bayesian networks. However, they suffer from important limitations when it comes to modeling real world time series. In the case of linear-Gaussian state-space models the limitation is advertised in the name: in many realistic applications, both the state dynamics and the relation between states and observations can be nonlinear, and the noise can be non-Gaussian. For hidden Markov models, the situation is more subtle. HMMs are a dynamical extension of mixture models, and unconstrained mixture models can be used to model any distribution in the limit of an infinite number of mixture components. Furthermore, if the state transition matrix is unconstrained, any arbitrary nonlinear dynamics can also be modeled. So where does the limitation lie?

Consider the problem of modeling the movement of several objects in a sequence of images. If there are M objects, each of which can occupy K positions and orientations in the image, there are K^M possible states of the system underlying an image. A hidden Markov model would require K^M distinct states to model this system. This representation is not only inefficient but difficult to interpret. We would much rather if our "HMM" could capture the underlying state space by using M different K-dimensional variables. More seriously, an unconstrained HMM with K^M states has of order K^{2M} parameters in the transition matrix. Unless the data set captures all these possible transitions, severe over-fitting may result.

In this section, we describe three ways in which HMMs and state-space models can be extended to overcome some of these limitations. The first of these

represents the hidden state of an HMM using a set of distinct state variables. We can this HMM with a distributed state representation, a *factorial hidden Markov model* [17].

5.1 Example 3: Factorial HMMs

We generalize the HMM by representing the state using a collection of discrete state variables

$$S_t = S_t^{(1)}, \ldots S_t^{(m)}, \ldots, S_t^{(M)}, \tag{45}$$

each of which can take on $K^{(m)}$ values. The state space of this model consists of the cross product of these state variables. For simplicity, we will assume that $K^{(m)} = K$, for all m, although the algorithms we present can be trivially generalized to the case of differing $K^{(m)}$. Given that the state space of this factorial HMM consists of all K^M combinations of the $S_t^{(m)}$ variables, placing no constraints on the state transition structure would result in a $K^M \times K^M$ transition matrix. Such an unconstrained system is uninteresting for several reasons: it is equivalent to an HMM with K^M states; it is unlikely to discover any interesting structure in the K state variables, as all variables are allowed to interact arbitrarily; and both the time complexity and sample complexity of the estimation algorithm are exponential in M.

We therefore focus on factorial HMMs in which the underlying state transitions are constrained. A natural structure to consider is one in which each state variable evolves according to its own dynamics, and is *a priori* uncoupled from the other state variables:

$$P(S_t|S_{t-1}) = \prod_{m=1}^{M} P(S_t^{(m)}|S_{t-1}^{(m)}). \tag{46}$$

A Bayesian network representing this model is shown in Figure 4. The transition structure for this model can be parametrized using M distinct $K \times K$ matrices.

As shown in Figure 4, the observation at time step t can depend on all the state variables at that time step in a factorial HMM. For real-valued observations, one simple form for this dependence is linear-Gaussian; that is, the observation Y_t is a Gaussian random vector whose mean is a linear function of the state variables. We represent the state variables as $K \times 1$ vectors, where each of the K discrete values corresponds to a 1 in one position and 0 elsewhere. The resulting probability density for a $D \times 1$ observation vector Y_t is

$$P(Y_t|S_t) = |C|^{-1/2} (2\pi)^{-D/2} \exp\left\{-\frac{1}{2} (Y_t - \mu_t)' C^{-1} (Y_t - \mu_t)\right\}, \tag{47}$$

where

$$\mu_t = \sum_{m=1}^{M} W^{(m)} S_t^{(m)}. \tag{48}$$

Each $W^{(m)}$ matrix is a $D \times K$ matrix whose columns are the contributions to the means for each of the settings of $S_t^{(m)}$, C is the $D \times D$ covariance matrix, $'$ denotes matrix transpose.

One way to understand the observation model in equations (47) and (48) is to consider the marginal distribution for Y_t, obtained by summing over the possible states. There are K settings for each of the M state variables, and thus there are K^M possible mean vectors obtained by forming sums of M columns where one column is chosen from each of the $W^{(m)}$ matrices. The resulting marginal density of Y_t is thus a Gaussian mixture model, with K^M Gaussian mixture components each having a constant covariance matrix C. This static mixture model, without inclusion of the time index and the Markov dynamics, is a factorial parameterization of the standard mixture of Gaussians model that has interest in its own right [48, 19, 14]. The model we are considering here extends this by allowing Markov dynamics in the discrete state variables underlying the mixture.

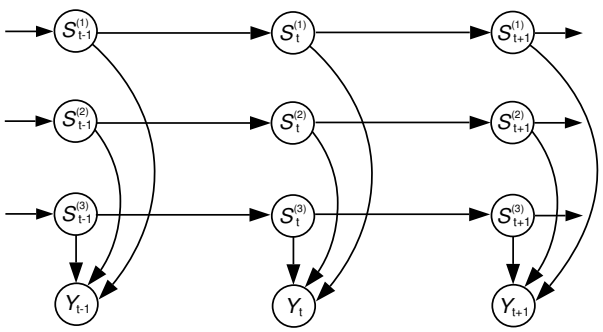

Fig. 4. A Bayesian network representing the conditional independence relations in a factorial HMM with $M = 3$ underlying Markov chains.

5.2 Example 4: Tree structured HMMs

In factorial HMMs, the state variables at one time step are assumed to be *a priori* independent given the state variables at the previous time step. This assumption can be relaxed in many ways by introducing coupling between the state variables in a single time step [42]. One interesting way to couple the variables is to order them, such that $S_t^{(m)}$ depends on $S_t^{(n)}$ for $1 \leq n < m$. Furthermore, if all the state variables and the output also depend on an observable input variable, X_t, we obtain the Bayesian network shown in Figure 5.

This architecture can be interpreted as a probabilistic decision tree with Markovian dynamics linking the decision variables. Consider how this model

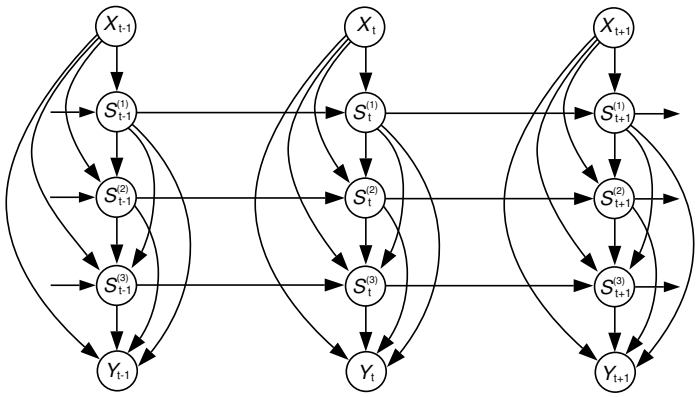

Fig. 5. Tree structured hidden Markov models.

would generate data at the first time step, $t = 1$. Given input X_1, the top node $S_1^{(1)}$ can take on K values. This stochastically partitions X-space into K decision regions. The next node down the hierarchy, $S_1^{(2)}$, subdivides each of these regions into K subregions, and so on. The output Y_1 is generated from the input X_1 and the K-way decisions at each of the M hidden nodes. At the next time step, a similar procedure is used to generate data from the model, except that now each decision in the tree is dependent on the decision taken at that node in the previous time step. This model therefore generalizes the "hierarchical mixture of experts" [25] and other related decision tree models such as CART [6] and MARS [12] by giving the decisions Markovian dynamics. Tree structured HMMs provide a useful starting point for modeling time series with both temporal and spatial structure at multiple resolutions. We have explored this generalization of factorial HMMs in [24].

5.3 Example 5: Switching State space models

Both factorial HMMs and tree-structured HMMs use discrete hidden state representations. To model time series with continuous but nonlinear dynamics, it is possible to combine the real-valued hidden state of linear-Gaussian state-space models and the discrete state of HMMs. One natural way to do this is the *switching state-space model* [16].

In switching state-space models, the sequence of observations $\{Y_t\}$ is modeled using a hidden state space comprising M real-valued state vectors, $X_t^{(m)}$, and one discrete state vector S_t. The discrete state, S_t, is a multinomial variable that can take on M values: $S_t \in \{1, \ldots, M\}$; for reasons that will become obvious we refer to it as the *switch* variable. The joint probability of observations and hidden states can be factored as

$$P(\{S_t, X_t^{(1)}, \ldots, X_t^{(M)}, Y_t\}) = P(S_1) \prod_{t=2}^{T} P(S_t|S_{t-1}) \prod_{m=1}^{M} P(X_1^{(m)}) \prod_{t=2}^{T} P(X_t^{(m)}|X_{t-1}^{(m)})$$

$$\times \prod_{t=1}^{T} P(Y_t | X_t^{(1)}, \ldots, X_t^{(M)}, S_t), \tag{49}$$

which corresponds graphically to the conditional independences represented by Figure 6. Conditioned on a setting of the switch state, $S_t = m$, the observable is multivariate Gaussian with output equation given by state-space model m. The probability of the observation vector Y_t is therefore

$$P(Y_t | X_t^{(1)}, \ldots, X_t^{(M)}, S_t = m) = (2\pi)^{-\frac{D}{2}} |R|^{-\frac{1}{2}} \times \exp\left\{ -\tfrac{1}{2} \left(Y_t - C^{(m)} X_t^{(m)} \right)' R^{-1} \left(Y_t - C^{(m)} X_t^{(m)} \right) \right\} \tag{50}$$

where D is the dimension of the observation vector, R is the observation noise covariance matrix, and $C^{(m)}$ is the output matrix for state-space model m (cf. equation (7) for a single linear-Gaussian state-space model). Each real-valued state vector evolves according to the linear Gaussian dynamics of a state-space model with differing initial state, transition matrix, and state noise (equation (6)). The switch state itself evolves according to the discrete Markov dynamics specified by initial state probabilities $P(S_1)$ and an $M \times M$ state transition matrix $P(S_t | S_{t-1})$.

This model can be seen as an extention of the "mixture of experts" architecture for modular learning in neural networks [21, 7, ?]. Each state-space model is a linear expert with Gaussian output noise and linear-Gaussian dynamics. The switch state "gates" the outputs of the M state-space models, and therefore plays the role of a gating network with Markovian dynamics [7, 33].

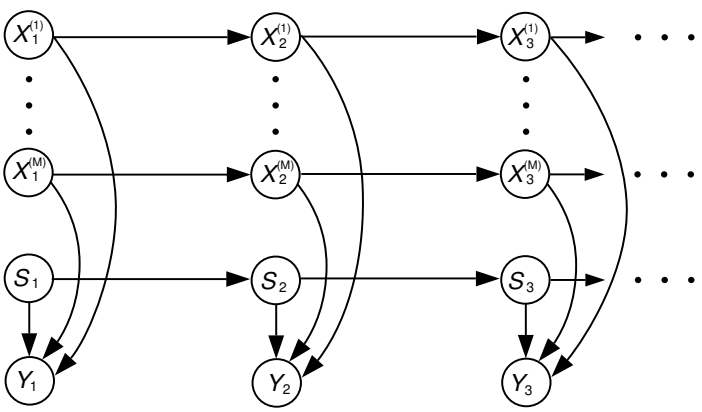

Fig. 6. Bayesian network representation for switching state-space models. S_t is the discrete switch variable and $X_t^{(m)}$ are the real-valued state vectors.

6 Inference and Intractability

The problem with all the extensions of hidden Markov models and state-space models presented in the previous section is that, given a sequence of observations, most probabilities of interest are intractable to compute.

Consider, for example, computing the likelihood of a factorial HMM—the marginal probability of a sequence of observations given the parameters, $P(\{Y_t\}|\theta)$, where $\{Y_t\}$ denotes $\{Y_1, \ldots, Y_T\}$. This is the sum over all possible hidden state sequences of the joint probability of the sequence and the observations:

$$P(\{Y_t\}|\theta) = \sum_{\{S_t\}} P(\{S_t, Y_t\}|\theta).$$

There are K^M possible states at each time step, and therefore K^{MT} hidden state sequences of length T, assuming none of the transition probabilities is exactly 0. The brute-force approach of evaluating all such sequences can be avoided by making use of the conditional independences represented in the Bayesian network. For example, directly applying the forward pass of the forward–backward algorithm outlined in section 4.6, we can compute the likelihood by summing the α's at the last time step

$$P(\{Y_t\}|\theta) = \sum_{S_T} P(S_T, Y_1, \ldots, Y_T|\theta) \tag{51}$$

$$= \sum_{S_T} \alpha_T. \tag{52}$$

For the factorial HMM, α_t is a vector of size equal to the full state space at time t, i.e. it has K^M elements. This results in a recursive algorithm that computes the likelihood using $O(TK^{2M})$ operations. This can be further improved upon by using the fact that the state transitions are defined via M matrices of size $K \times K$ rather than a single $K^M \times K^M$ matrix, resulting in a recursive algorithm using $O(TMK^{M+1})$ operations (see [17], appendix B). Unfortunately, this time complexity cannot be improved upon. Given the observation at time t, the K-valued state variables become coupled in an M^{th} order interaction. It is not possible to sum over each variable independently. Like the likelihood, computing the posterior probability of a single state variable given the observation sequence, $P(S_t^{(m)}|Y_1, \ldots, Y_T)$, is also exponential in M. Similar exponential time complexity results hold for the likelihoods and posterior probabilities of tree-structured HMMs and switching state-space models.

6.1 Gibbs sampling

One approach to computing approximate marginal probabilities is to make use of Monte Carlo integration. Since the log likelihood can be expressed as

$$\log P(\{Y_t\}|\theta) = \sum_{\{S_t\}} P(\{S_t\}|\{Y_t\}, \theta) \left[\log P(\{S_t\}, \{Y_t\}, \theta) - \log P(\{S_t\}|\{Y_t\}, \theta)\right],$$

by sampling from the posterior distribution, $P(\{S_t\}|\{Y_t\},\theta)$, the log likelihood can be approximated using the above expression, which is just the negative of the free energy (15). To learn the parameters of the model, samples from the posterior are used do evaluate the expectations required for EM. While for intractable models, sampling directly from the posterior distributions is computationally prohibitive it is often easy to set up a Markov chain that will converge to samples from the posterior. One of the simplest such methods is Gibbs sampling (for a review of Gibbs sampling and other Markov chain Monte Carlo methods, see [34]).

For a given observation sequence $\{Y_t\}$, Gibbs sampling starts with a random setting of the hidden states $\{S_t\}$. At each step of the sampling process, each state variable is updated stochastically according to its probability distribution conditioned on the setting of all the other state variables. The graphical model is again useful here, as each node is conditionally independent of all other nodes given its Markov blanket, defined as the set of children, parents, and parents of the children of a node. To sample from a typical state variable $S_t^{(m)}$ in a factorial HMM, for example, we only need to examine the states of a few neighboring nodes:

$$S_t^{(m)} \sim P(S_t^{(m)}|\{S_t^{(n)} : n \neq m\}, S_{t-1}^{(m)}, S_{t+1}^{(m)}, Y_t) \tag{53}$$

$$\propto P(S_t^{(m)}|S_{t-1}^{(m)})\ P(S_{t+1}^{(m)}|S_t^{(m)})\ P(Y_t|S_t^{(1)}, \ldots, S_t^{(m)}, \ldots, S_t^{(M)}), \tag{54}$$

where $\sim$ denotes "sampled from". Sampling once from each of the TM hidden variables in the model results in a new sample of the hidden state of the model and requires $O(TMK)$ operations. The sequence of overall states resulting from each pass of Gibbs sampling defines a Markov chain over the state space of the model. This Markov chain is guaranteed to converge to the posterior probabilities of the states given the observations [13] as long as none of the probabilities in the model is exactly zero [9]. Thus, after some suitable time, samples from the Markov chain can be taken as approximate samples from the posterior probabilities. The first and second-order statistics needed to estimate $\langle S_t^{(m)} \rangle$, $\langle S_t^{(m)} S_t^{(n)\prime} \rangle$ and $\langle S_{t-1}^{(m)} S_t^{(m)\prime} \rangle$ are collected using the states visited and the probabilities estimated during this sampling process are used in the approximate E step of EM.[10] Monte Carlo methods for learning in dynamic Bayesian networks have been explored by [9, 28, 8, 17].

6.2 Variational Methods

Another approach to approximating a probability distribution P is to define a parametrized distribution Q and vary its parameters so as to minimize the

[9] Actually, the weaker assumption of ergodicity will suffice to ensure convergence

[10] A more Bayesian treatment of the learning problem, in which the parameters are also considered hidden random variables, can be handled by Gibbs sampling by replacing the "M step" with sampling from the conditional distribution of the parameters given the other hidden variables (for example, see [47]).

distance between Q and P. In the context of the EM algorithm, we have already seen that the likelihood $\mathcal{L}(\theta)$ is lower bounded by the free energy $\mathcal{F}(Q, \theta)$. The difference between $\mathcal{L}$ and $\mathcal{F}$ is given by the Kullback-Leibler divergence between Q and the posterior distribution of the hidden variables:

$$\mathcal{L}(\theta) - \mathcal{F}(Q, \theta) = \mathrm{KL}\left(Q(\{S_t\}|\phi) \,\|P(\{S_t\}|\{Y_t\}, \theta)\right) \tag{55}$$

$$= \sum_{\{S_t\}} Q(\{S_t\}|\phi) \, \log\left[\frac{Q(\{S_t\}|\phi}{P(\{S_t\}|\{Y_t\}, \theta)}\right] \tag{56}$$

where ϕ are the parameters of the variational distribution Q.

The complexity of exact inference in the approximation given by Q is determined by its conditional independence relations, not by its parameters. Thus, we can chose Q to have a tractable structure—a Bayesian network that eliminates some of the dependencies in P. Given this structure, we are free to vary the parameters of Q so as to obtain the tightest possible bound by minimizing (56). We will refer to the general strategy of using a parameterized approximating distribution as a *variational approximation* and refer to the free parameters of the distribution as *variational parameters*.

6.3 Example: Mean field for factorial HMMs

We illustrate this approach using the simplest variational approximation to the posterior distribution in factorial HMMs: the state variables are assumed independent (Figure 7 (a)) which means that

$$Q(\{S_t\}|\phi) = \prod_{t=1}^{T} \prod_{m=1}^{M} Q(S_t^{(m)}|\phi_t^{(m)}). \tag{57}$$

The variational parameters, $\phi = \left\{\phi_t^{(m)}\right\}$, are the means of the state variables, where, as before, a state variable $S_t^{(m)}$ is represented as a K-dimensional vector with a 1 in the k^{th} position and 0 elsewhere, if the m^{th} Markov chain is in state k at time t. The elements of the vector $\phi_t^{(m)}$ therefore define the state occupation probabilities for the multinomial variable $S_t^{(m)}$ under the distribution Q:

$$Q(S_t^{(m)}|\phi_t^{(m)}) = \prod_{k=1}^{K} \left(\phi_{t,k}^{(m)}\right)^{S_{t,k}^{(m)}} \qquad \text{where} \quad S_{t,k}^{(m)} \in \{0, 1\}; \quad \sum_{k=1}^{K} S_{t,k}^{(m)} = 1. \tag{58}$$

A completely factorized approximation of this kind is often used in statistical physics, where it provides the basis for simple yet powerful *mean field approximations* to statistical mechanical systems [37].

To make the bound as tight as possible we vary ϕ separately for each observation sequence so as to minimize the KL divergence. Taking the derivatives

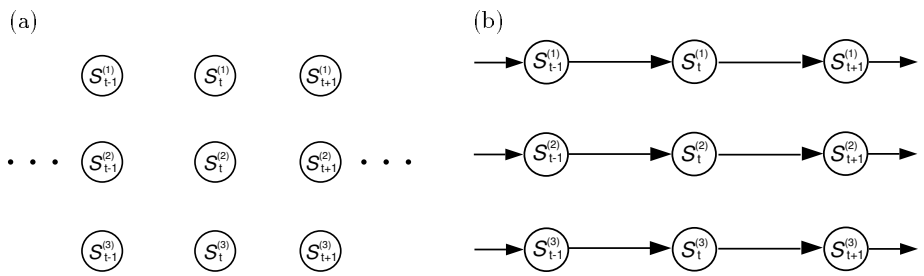

Fig. 7. (a) The completely factorized variational approximation assumes that all the state variables are independent (conditional on the observation sequence). (b) A structured variational approximation assumihg that the state variables retain their Markov structure within each chain, but are independent across chains.

of (56) with respect to $\phi_t^{(m)}$ and setting them to zero, we obtain the set of fixed point equations defined by

$$\phi_t^{(m)\text{ new}} = \varphi\left\{ W^{(m)'} C^{-1} \tilde{Y}_t^{(m)} - \frac{1}{2}\Delta^{(m)} + (\log P^{(m)})\, \phi_{t-1}^{(m)} + (\log P^{(m)})'\, \phi_{t+1}^{(m)} \right\} \tag{59}$$

where $\tilde{Y}_t^{(m)}$ is the residual error in Y_t given the predictions from all the state variables not including m:

$$\tilde{Y}_t^{(m)} \equiv Y_t - \sum_{\ell \neq m}^{M} W^{(\ell)} \phi_t^{(\ell)}, \tag{60}$$

$\Delta^{(m)}$ is the vector of diagonal elements of $W^{(m)'} C^{-1} W^{(m)}$, and $\varphi\{\cdot\}$ is the softmax operator, which maps a vector A into a vector B of the same size, with elements

$$B_i = \frac{\exp\{A_i\}}{\sum_j \exp\{A_j\}}, \tag{61}$$

and $\log P^{(m)}$ denotes the elementwise logarithm of the transition matrix $P^{(m)}$ (see appendix C in [17] for details of the derivation).

The first term of (59) is the projection of the error in reconstructing the observation onto the weights of state vector m—the more a particular setting of a state vector can reduce this error, the larger its associated variational parameter. The second term arises from the fact that the second order correlation $\langle S_t^{(m)} S_t^{(m)} \rangle$ evaluated under the variational distribution is a diagonal matrix composed of the elements of $\phi_t^{(m)}$. The last two terms introduce dependencies forward and backward in time.[11] Therefore, although the posterior distribution

[11] The first term is replaced by $\log \pi^{(m)}$ for $t = 1$ the second term does not appear for $t = T$.

over the hidden variables is approximated with a completely factorized distribution, the fixed point equations couple the parameters associated with each node with the parameters of its Markov blanket. In this sense, the fixed point equations propagate information along the same pathways as those defining the exact algorithms for probability propagation.

The following may provide an intuitive interpretation of the approximation being made by this distribution. Given a particular observation sequence, the hidden state variables for the M Markov chains at time step t are stochastically coupled. This stochastic coupling is approximated by a system in which the hidden variables are uncorrelated but have coupled means. The variational or "mean-field" equations solve for the deterministic coupling of the means that best approximates the stochastically coupled system.

Each hidden state vector is updated in turn using (59), with a time complexity of $O(TMK^2)$ per iteration. Convergence is determined by monitoring the KL divergence in the variational distribution between successive time steps; in practice convergence is very rapid (about 2 to 10 iterations of (59)). Once the fixed point equations have converged, the expectations required for the E step can be obtained as a simple function of the parameters [17].

6.4 Example: Structured approximation for factorial HMMs

The approximation presented in the previous section factors the posterior probability such that all the state variables are statistically independent. Here we present another approximation that is tractable and preserves much of the probabilistic structure of the original system. In this scheme, the factorial HMM is approximated by M uncoupled HMMs as shown in Figure 7 (b). Within each HMM, efficient and exact inference is implemented via the forward–backward algorithm. Since the arguments presented in the previous section did not hinge on the the form of the approximating distribution, each distribution Q provides a lower bound on the log likelihood and can be used to obtain a learning algorithm. The approach of exploiting such tractable substructures was first suggested in the machine learning literature by Saul and Jordan (1996).

We write the structured variational approximation as

$$Q(\{S_t\}|\phi) = \frac{1}{Z_Q} \prod_{m=1}^{M} Q(S_1^{(m)}|\phi) \prod_{t=2}^{T} Q(S_t^{(m)}|S_{t-1}^{(m)}, \phi), \tag{62}$$

where Z_Q is a normalization constant ensuring that Q integrates to one. The parameters of Q are $\phi = \{\pi^{(m)}, P^{(m)}, h_t^{(m)}\}$—the original priors and state transition matrices of the factorial HMM and a time-varying bias for each state variable. Using these parameters the prior and transition probabilities are

$$Q(S_1^{(m)}|\phi) = \prod_{k=1}^{K} \left(h_{1,k}^{(m)} \pi_k^{(m)} \right)^{S_{1,k}^{(m)}} \tag{63}$$

$$Q(S_t^{(m)}|S_{t-1}^{(m)},\phi) = \prod_{k=1}^{K}\left(h_{t,k}^{(m)}\sum_{j=1}^{K}P_{k,j}^{(m)}S_{t-1,j}^{(m)}\right)^{S_{t,k}^{(m)}}$$
$$= \prod_{k=1}^{K}\left(h_{t,k}^{(m)}\prod_{j=1}^{K}\left(P_{k,j}^{(m)}\right)^{S_{t-1,j}^{(m)}}\right)^{S_{t,k}^{(m)}}, \qquad (64)$$

where the last equality follows from the fact that $S_{t-1}^{(m)}$ is a vector with a 1 in one position and 0 elsewhere. Comparing equations (62)–(64) to equation (9), we can see that the $K \times 1$ vector $h_t^{(m)}$ plays the role of the probability of an observation ($P(Y_t|S_t)$ in (9)) for each of the K settings of $S_t^{(m)}$. For example, $Q(S_{1,j}^{(m)} = 1|\phi) = h_{1,j}^{(m)}\ P(S_{1,j}^{(m)} = 1|\phi)$ corresponds to having an observation at time $t = 1$ that under state $S_{1,j}^{(m)} = 1$ has probability $h_{1,j}^{(m)}$.

Intuitively, this approximation uncouples the M Markov chains and attaches to each state variable a distinct fictitious observation. The probability of this fictitious observation can be varied so as to minimize the KL divergence between Q and P.

Applying the same arguments as before, we obtain a set of fixed point equations for $h_t^{(m)}$ that minimize $\mathrm{KL}(Q\|P)$:

$$h_t^{(m)\ \mathrm{new}} = \exp\left\{W^{(m)'}C^{-1}\tilde{Y}_t^{(m)} - \frac{1}{2}\Delta^{(m)}\right\}, \qquad (65)$$

where $\Delta^{(m)}$ is defined as before, and where we redefine the residual error to be

$$\tilde{Y}_t^{(m)} \equiv Y_t - \sum_{\ell\neq m}^{M} W^{(\ell)}\langle S_t^{(\ell)}\rangle. \qquad (66)$$

The parameter $h_t^{(m)}$ obtained from these fixed point equations is the observation probability associated with state variable $S_t^{(m)}$ in hidden Markov model m. Using these probabilities, the forward–backward algorithm is used to compute a new set of expectations for $\langle S_t^{(m)}\rangle$, which are fed back into (65) and (66). The forward–backward algorithm is therefore used as a subroutine in the minimization of the KL divergence.

Notice the similarity between equations (65)–(66) and equations (59)–(60) for the completely factorized system. In the completely factorized system, since $\langle S_t^{(m)}\rangle = \phi_t^{(m)}$, the fixed point equations can be written explicitly in terms of the variational parameters. In the structured approximation, the dependence of $\langle S_t^{(m)}\rangle$ on $h_t^{(m)}$ is computed via the forward–backward algorithm. Also, the fixed point equations (65) do not contain terms involving the prior, $\pi^{(m)}$, or transition matrix, $P^{(m)}$. These terms have cancelled by our choice of approximation.

The other intractable dynamic Bayesian networks we have presented are also amenable to structured variational approximations. In the case of tree-structured HMMs there are two natural choices for the substructures to retain in

the approximation. One choice is to remove the arc within a time step and retain the temporal dependences, resulting in the Bayesian network shown in Figure 7 (b). The other choice is to retain the arcs *within* a time step and eliminate the arcs between consecutive time steps. Both of these approximations, along with an approximation based on the Viterbi algorithm are pursued in [24].

For switching state-space models, the natural approximation is to uncouple the M state-space models (SSMs) from the discrete Markov process controlling the switch variable. Of course, through the variational parameters all the models become deterministically coupled, but for the purposes of computing posterior probabilities, it suffices to apply Kalman smoothing to each state-space models and the forward–backward algorithm to the switch process. The variational parameters can be thought of as the real-valued "responsibilities" of each state-space model for each observation in the sequence. To determine the best variational parameters we start from some responsibilities and compute the posterior probability of the state in each SSM using Kalman smoothing, with the data *weighted by the responsibilities*. A weighting of 1 corresponding to applying the normal Kalman smoothing equations, whereas a weighting of 0 corresponds to assuming that the data was not observed at all; intermediate weighting can be implemented by dividing the R matrix in (23) by the responsibility. We then recompute responsibilities by running the forward–backward algorithm on the switch process using the predicted error of each SSM. This procedure is iterated until the responsibilities converge. Details of this structured variational approximation for switching state-space models are provided in [16].

6.5 Convex duality

The framework for obtaining lower bounds on log likelihoods is a special case of more general variational methods based on convex duality. In this section, we provide a brief tutorial of these methods closely following Jaakkola (1997) who introduced these methods to problems in Bayesian network learning. A more general treatment can be found in Rockafellar (1970). But before delving into convex duality we will motivate the reader by making the following two remarks. First, we have presented lower bounds and suggested maximizing lower bounds on likelihoods as an objective for learning; however, it is also clearly desirable to complete the picture by deriving upper bounds. Second, we have not dealt with networks in which there are complex nonlinear interactions. Methods from convex duality can, in principle, be used to solve these problems. We present only a brief tutorial here and refer the reader to [20] for examples of how this approach can be used to define upper bounds and deal with certain nonlinearities.

A convex function $f(x)$ is characterized by the property that the set of points $\{(x, y) : y \geq f(x)\}$ is convex. This set is called the *epigraph* of f and denoted $epi(f)$. Now, convex sets can be represented as the intersection of all half-spaces that contain them. We parametrize these half-spaces to obtain the dual of f. Consider one such half-space

$$y \geq \xi^T x - \mu.$$

Since it contains $epi(f)$, $y \geq f(x)$ implies $y \geq \xi^T x - \mu$, therefore

$$f(x) \geq \xi^T x - \mu$$

at every x, which implies

$$f(x) \geq \max_x \{\xi^T x - \mu\} \tag{67}$$

It follows that

$$\mu \geq \max_x \{\xi^T x - f(x)\} \equiv f^*(\xi)$$

where we have defined $f^(\xi)$ as the dual function of $f(x)$.

An intuitive way to think about the dual function is that for every point x there is a linear function with slope ξ and intercept μ that touches f at x and is a lower bound for $f(x)$. The dual $f^*(\xi)$ is a function of these slopes that evaluates to the corresponding y-intercept of f at the point at which f has slope ξ.[12] Simply put, we have shown that a convex function of x can be lower-bounded by a linear function of x parametrized by ξ.

This simple result has important consequences. We now show that the lower bound on the log likelihood can be seen as a special case of this bound.

The log likelihood can be written

$$\log P(Y) = \log \sum_S P(Y, S) = \log \sum_S \exp\{\phi(S)\}$$

where $\phi(S) = \log P(Y, S)$ is a "potential" over the hidden states. The log partition function $f(\phi) = \log \sum_S \exp\{\phi(S)\} = \log P(Y)$ is a convex function over potentials $\phi(S)$. The dual to the log partition function $f(\phi)$ is the negative entropy function, $f^*(Q) = -H(Q) = \sum_S Q(S) \log Q(S)$, which itself is a convex function over probability distributions Q. The duality between f and f^* can be verified by taking the derivatives of $f(\phi)$ with respect to ϕ, remembering that the dual is a function of the slopes that evaluates to the corresponding intercepts. Therefore, using (67)

$$\log P(Y) = f(\phi) \geq \max_Q \{Q^T \phi + H(Q)\} \tag{68}$$

$$= \max_Q \{\sum_S Q(S) \log P(Y, S) - \sum_S Q(S) \log Q(S)\} \tag{69}$$

which is the usual lower bound $\mathcal{F}$.

[12] For strictly convex functions

7 Conclusion

Bayesian networks are a concise graphical formalism for describing probabilistic models. We have provided a brief tutorial of methods for learning and inference in dynamic Bayesian networks. In many of the interesting models, beyond the simple linear dynamical system or hidden Markov model, the calculations required for inference are intractable. Two different approaches for handling this intractability are Monte Carlo methods such as Gibbs sampling, and variational methods. An especially promising variational approach is based on exploiting tractable substructures in the Bayesian network.

Acknowledgements

Much of the work reviewed in the paper arose from collaborations between the author and Geoffrey E. Hinton, Michael I. Jordan, and Lawrence K. Saul. The author was supported by a fellowship from the Ontario Information Technology Research Centre.

References

1. B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Prentice-Hall, Englewood Cliffs, NJ, 1979.
2. P. Baldi, Y. Chauvin, T. Hunkapiller, and M.A. McClure. Hidden Markov models of biological primary sequence information. *Proc. Nat. Acad. Sci. (USA)*, 91(3):1059–1063, 1994.
3. L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970.
4. Y. Bengio and P. Frasconi. An input–output HMM architecture. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 427–434. MIT Press, Cambridge, MA, 1995.
5. J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J. Royal Stat. Soc. B*, 36:192–326, 1974.
6. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.
7. T. W. Cacciatore and S. J. Nowlan. Mixtures of controllers for jump linear and non-linear plants. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 719–726. Morgan Kaufmann Publishers, San Francisco, CA, 1994.
8. C. K. Carter and R. Kohn. Markov chain Monte Carlo in conditionally Gaussian state space models. *Australian Graduate School of Management, University of New South Wales*, 1996.
9. T. Dean and K. Kanazawa. A model for reasoning about persitence and causation. *Computational Intelligence*, 5(3):142–150, 1989.
10. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, 39:1–38, 1977.

11. V. Digalakis, J. R. Rohlicek, and M. Ostendorf. ML estimation of a Stochastic Linear System with the EM Algorithm and its Application to Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442, 1993.
12. J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.
13. S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.
14. Z. Ghahramani. Factorial learning and the *EM* algorithm. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 617–624. MIT Press, Cambridge, MA, 1995.
15. Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2 [ftp://ftp.cs.toronto.edu/pub/zoubin/tr-96-2.ps.gz] , 1996.
16. Z. Ghahramani and G. E. Hinton. Switching state-space models. Technical Report CRG-TR-96-3 [ftp://ftp.cs.toronto.edu/pub/ zoubin/switch.ps.gz], 1996.
17. Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. *Machine Learning*, 1997.
18. G.C. Goodwin and K.S. Sin. *Adaptive filtering prediction and control*. Prentice-Hall, 1984.
19. G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and Helmholtz free energy. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann Publishers, San Francisco, CA, 1994.
20. T. S. Jaakkola. Variational methods for Inference and estimation in graphical models. Technical Report Ph.D. Thesis, Cambridge, MA, 1997.
21. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.
22. F. V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag, New York, 1996.
23. F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.
24. M. I. Jordan, Z. Ghahramani, and L. K. Saul. Hidden Markov decision trees. In M.C. Mozer, M.I Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*. MIT Press, Cambridge, MA, 1997.
25. M. I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.
26. B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33:251–272, 1991.
27. R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction. *Journal of Basic Engineering (ASME)*, 83D:95–108, 1961.
28. K. Kanazawa, D. Koller, and S. J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In P. Besnard and S. Hanks, editors, *Uncertainty in Artificial Intelligence. Proceedings of the Eleventh Conference.*, pages 346–351. Morgan Kaufmann Publishers, San Francisco, CA, 1995.
29. J. H. Kim and J. Peal. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 190–193. 1983.

30. S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, pages 157–224, 1988.
31. L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA, 1983.
32. D. J. C. MacKay. Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995.
33. M. Meila and M. I. Jordan. Learning fine motion by Markov mixtures of experts. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.
34. R. M. Neal. Probabilistic inference using Markov chain monte carlo methods. Technical Report CRG-TR-93-1, 1993.
35. R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, New York, 1996.
36. R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. *Submitted*, 1993.
37. G. Parisi. *Statistical Field Theory*. Addison-Wesley, Redwood City, CA, 1988.
38. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.
39. L. R. Rabiner and B. H. Juang. An Introduction to hidden Markov models. *IEEE Acoustics, Speech & Signal Processing Magazine*, 3:4–16, 1986.
40. H. E. Rauch. Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, 8:371–372, 1963.
41. R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
42. L. K. Saul and M. I. Jordan. Mixed memory Markov models. In D. Madigan and P. Smyth, editors, *Proceedings of the 1997 Conference on Artificial Intelligence and Statistics*. Ft. Lauderdale, FL, 1997.
43. L.K. Saul and M. I. Jordan. Exploiting tractable substructures in Intractable networks. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.
44. R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis*, 3(4):253–264, 1982.
45. P. Smyth. Hidden Markov models for fault detection in dynamic systems. *Pattern Recognition*, 27(1):149–164, 1994.
46. P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9:227–269, 1997.
47. M. A. Tanner and W. H. Wong. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550, 1987.
48. R. S. Zemel. *A minimum description length framework for unsupervised learning*. Ph.D. Thesis, Dept. of Computer Science, University of Toronto, Toronto, Canada, 1993.

This article was processed using the LaTeX macro package with LLNCS style