24 Inference with Hidden Markov Models

Our development of belief propagation, i.e., the sum-product algorithm, focused on tree models. A very important class of tree models are Markov chains. In these notes, we specialize the sum-product algorithm to Markov chains, and see that they have additional structure that can be exploited. In the process, we'll develop probabilistic interpretations of message-passing that provide additional insight into belief propagation.

24.1 Markov Chains

A Markov chain

$$x_1 \leftrightarrow x_2 \leftrightarrow \cdots \leftrightarrow x_N$$

has the representation as an undirected graph depicted in Fig. 1. Moreover, one of many possible equivalent representations for this chain as a directed graph is depicted in Fig. 2.

We have already encountered such models: in our introduction to directed graphs, where we discussed simple 3-node Markov chains; in our introduction to undirected graphical models, where the Isiing models we discussed are instances of such chains having variables over binary alphabets; and in our development of Gaussian graphical models, where they corresponded to linear dynamical systems with Gaussian inputs.

More generally, Markov chains are useful for modeling a surprisingly wide range of natural and man-made phenomena. Frequently, they are used for modeling the temporal evolution of such phenomena, in which case the node index represents a discretization of time. But in other applications the index could represent space or any number of other possibilities.

24.2 Hidden Markov Models

Frequently, the variables in a Markov chain of interest are *latent*, i.e., not directly observed. In this case, we are interested in calculating posterior beliefs conditioned on some set of observed variables that are related to those in the chain. For such scenarios, the model of Fig. 3 is widely used, where $y_1, \ldots, y_N$ are the observations. This is referred to as a *hidden Markov model (HMM)*. Note that with this model, given the chain $x_1, \ldots, x_N$, the observations $y_1, \ldots, y_N$ are independent of one another. Fig. 4 shows one of many possible directed graphs that are equivalent representations for an HMM.

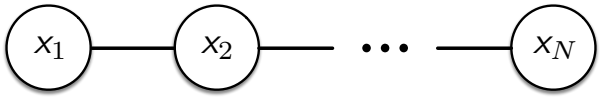


Figure 1: The undirected graph representing a Markov chain.

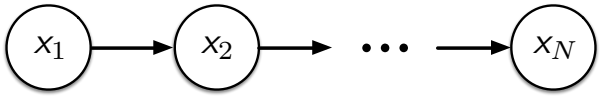


Figure 2: A directed graph representing a Markov chain.

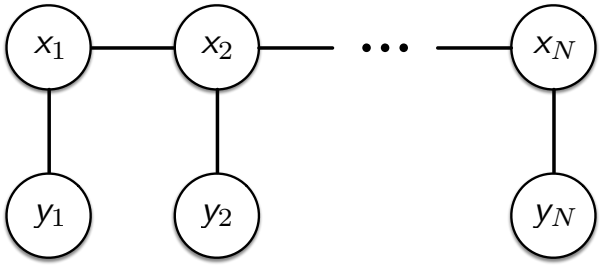


Figure 3: The undirected graph representing a hidden Markov model.

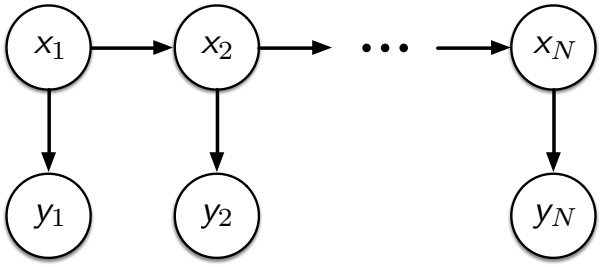


Figure 4: An equivalent directed graph representing a hidden Markov model of Fig. 3.

Our choice of directed graph representation for HMMs yields the following factorization of posterior distribution

$$\begin{aligned} p_{x_1,\ldots,x_N|y_1,\ldots,y_N}(x_1,\ldots,x_N|y_1,\ldots,y_N) & \\ &\propto p_{x_1,\ldots,x_N,y_1,\ldots,y_N}(x_1,\ldots,x_N,y_1,\ldots,y_N) \\ &= p_{x_1}(x_1)\prod_{i=2}^{N} p_{x_i|x_{i-1}}(x_i|x_{i-1})\prod_{j=1}^{N} p_{y_i|x_i}(y_j|x_j). \\ &= \underbrace{p_{x_1,y_1}(x_1,y_1)}_{\triangleq\phi_1(x_1)}\prod_{i=2}^{N}\underbrace{p_{x_i|x_{i-1}}(x_i|x_{i-1})}_{\triangleq\psi_{i-1,i}(x_{i-1},x_i)}\prod_{j=2}^{N}\underbrace{p_{y_j|x_j}(y_j|x_j)}_{\triangleq\phi_j(x_j)}. \end{aligned} \tag{1}$$

As (1) reflects, the associated reduced graph is the Markov chain depicted in Fig. 1, but where the relevant potentials with the observations folded in are

$$\phi_1(x_1) = p_{x_1,y_1}(x_1,y_1) \tag{2a}$$
$$\phi_i(x_i) = p_{y_i|x_i}(y_i|x_i), \qquad i=2,\ldots,N \tag{2b}$$
$$\psi_{i,i+1}(x_i,x_{i+1}) = p_{x_{i+1}|x_i}(x_{i+1}|x_i), \qquad i=1,\ldots,N-1. \tag{2c}$$

24.3 The Forward-Backward Algorithm

When we specialize the sum-product algorithm to this class of trees, we obtain what is referred to as the *forward-backward algorithm*, which we now develop and interpret.

First, note that the corresponding messages are

$$m_{1\to2}(x_2) = \sum_{x_1}\phi_1(x_1)\,\psi_{12}(x_1,x_2) \tag{3a}$$
$$m_{i\to i+1}(x_{i+1}) = \sum_{x_i}\phi_i(x_i)\,\psi_{i,i+1}(x_i,x_{i+1})\,m_{i-1\to i}(x_i), \qquad i=2,3,\ldots,N \tag{3b}$$
$$m_{N\to N-1}(x_{N-1}) = \sum_{x_N}\phi_N(x_N)\,\psi_{N-1,N}(x_{N-1},x_N) \tag{3c}$$
$$m_{i\to i-1}(x_{i-1}) = \sum_{x_i}\phi_i(x_i)\,\psi_{i-1,i}(x_{i-1},x_i)\,m_{i+1\to i}(x_i), \qquad i=1,2,\ldots,N-1, \tag{3d}$$

and that the posterior beliefs are

$$p_{x_1|y_1,\ldots,y_N}(x_1|y_1,\ldots,y_N) \propto \phi_1(x_1)\,m_{2\to1}(x_1) \tag{3e}$$
$$p_{x_i|y_1,\ldots,y_N}(x_i|y_1,\ldots,y_N) \propto \phi_i(x_i)\,m_{i-1\to i}(x_i)\,m_{i+1\to i}(x_i), \qquad i=2,\ldots,N-1 \tag{3f}$$
$$p_{x_N|y_1,\ldots,y_N}(x_N|y_1,\ldots,y_N) \propto \phi_N(x_N)\,m_{N-1\to N}(x_N). \tag{3g}$$

Eqs. (3) constitute what is referred to as the *forward-backward algorithm* or the *BCJR algorithm* (after the authors of the paper in which it was described).[1] More specifically, (3a)–(3b) are referred to as the "forward" messages, which are computed starting from node 1 and proceeding to node N, and (3c)–(3d) are referred toa s the "backward" messages, which are subsequently computed starting from node N and progressing back to node 1. This algorithm has long history and is extremely widely used in a myriad of applications, even today.

Note, too, that the computational complexity of the forward-backward algorithm scales exactly as the general sum-product algorithm, i.e., $O(N|\mathcal{X}|^2)$.

24.3.1 Interpretation of Forward-Backward Messages

Using (2) and the conditional independencies expressed by graph separation in Fig. 3, we can interpret the messages in the forward-backward algorithm as probabilities.

Claim 1. *The forward messages in the forward-backward algorithm have the probabilistic interpretation*

$$m_{i\to i+1}(x_{i+1}) = p_{y_1,\ldots,y_i,x_{i+1}}(y_1,\ldots,y_i,x_{i+1}), \qquad i = 1,\ldots,N-1 \tag{4}$$

Proof. Let us use induction. Assume that for some i we have

$$m_{i-1\to i}(x_i) = p_{y_1,\ldots,y_{i-1},x_i}(y_1,\ldots,y_{i-1},x_i). \tag{5}$$

Then using (3b) with (2) and (5) we have

$$\begin{aligned}
m_{i\to i+1}(x_{i+1}) &= \sum_{x_i} \phi_i(x_i)\,\psi_{i,i+1}(x_i,x_{i+1})\,m_{i-1\to i}(x_i) \\
&= \sum_{x_i} p_{y_i|x_i}(y_i|x_i)\,p_{x_{i+1}|x_i}(x_{i+1}|x_i)\,p_{y_1,\ldots,y_{i-1},x_i}(y_1,\ldots,y_{i-1},x_i) \\
&= \sum_{x_i} p_{x_{i+1}|x_i}(x_{i+1}|x_i)\,p_{y_i|x_i}(y_i|x_i)\,p_{y_1,\ldots,y_{i-1}|x_i}(y_1,\ldots,y_{i-1}|x_i)\,p_{x_i}(x_i) \\
&= \sum_{x_i} p_{x_{i+1}|x_i}(x_{i+1}|x_i)\,p_{y_1,\ldots,y_i|x_i}(y_1,\ldots,y_i|x_i)\,p_{x_i}(x_i) \qquad (6) \\
&= \sum_{x_i} p_{y_1,\ldots,y_i,x_{i+1}|x_i}(y_1,\ldots,y_i,x_{i+1}|x_i)\,p_{x_i}(x_i) \qquad (7) \\
&= \sum_{x_i} p_{y_1,\ldots,y_i,x_i,x_{i+1}}(y_1,\ldots,y_i,x_i,x_{i+1}) \\
&= p_{y_1,\ldots,y_i,x_{i+1}}(y_1,\ldots,y_i,x_{i+1}), \qquad (8)
\end{aligned}$$

[1]The algorithm appears in L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974, which emphasized the application to error-control decoding.

where to obtain (6) we have used that $(\mathsf{y}_1, \ldots, \mathsf{y}_{i-1}) \perp\!\!\!\perp \mathsf{y}_i \mid \mathsf{x}_i$, and where to obtain (7) we have used that $(\mathsf{y}_1, \ldots, \mathsf{y}_i) \perp\!\!\!\perp \mathsf{x}_{i+1} \mid \mathsf{x}_i$.

It remains only to verify (5) for $i = 2$, which we obtain from (3a) as follows

$$\begin{aligned} m_{1\to 2}(x_2) &= \sum_{x_1} \phi_1(x_1)\, \psi_{12}(x_1, x_2) \\ &= \sum_{x_1} p_{\mathsf{x}_1,\mathsf{y}_1}(x_1, y_1)\, p_{\mathsf{x}_2|\mathsf{x}_1}(x_2|x_1) \\ &= \sum_{x_1} p_{\mathsf{x}_1,\mathsf{y}_1,\mathsf{x}_2}(x_1, y_1, x_2) \\ &= p_{\mathsf{y}_1,\mathsf{x}_2}(y_1, x_2). \end{aligned}$$

□

Claim 2. *The backward messages in the forward-backward algorithm have the probabilistic interpretation*

$$m_{i\to i-1}(x_{i-1}) = p_{\mathsf{y}_i,\ldots,\mathsf{y}_N|\mathsf{x}_{i-1}}(y_i, \ldots, y_N|x_{i-1}), \qquad i = 2, \ldots N. \tag{9}$$

Proof. Let us again use induction. Assume that for some i we have

$$m_{i+1\to i}(x_i) = p_{\mathsf{y}_{i+1},\ldots,\mathsf{y}_N|\mathsf{x}_i}(y_{i+1}, \ldots, y_N|x_i), \qquad i = 1, \ldots N-1. \tag{10}$$

Then using (3d) with (2) and (10) we have

$$\begin{aligned} m_{i\to i-1}(x_{i-1}) &= \sum_{x_i} \phi_i(x_i)\, \psi_{i-1,i}(x_{i-1}, x_i)\, m_{i+1\to i}(x_i) \\ &= \sum_{x_i} p_{\mathsf{y}_i|\mathsf{x}_i}(y_i|x_i)\, p_{\mathsf{x}_i|\mathsf{x}_{i-1}}(x_i|x_{i-1})\, p_{\mathsf{y}_{i+1},\ldots,\mathsf{y}_N|\mathsf{x}_i}(y_{i+1}, \ldots, y_N|x_i) \\ &= \sum_{x_i} p_{\mathsf{x}_i|\mathsf{x}_{i-1}}(x_i|x_{i-1})\, p_{\mathsf{y}_i,\ldots,\mathsf{y}_N|\mathsf{x}_i}(y_i, \ldots, y_N|x_i) \qquad (11) \\ &= \sum_{x_i} \frac{p_{\mathsf{x}_{i-1},\mathsf{x}_i}(x_{i-1}, x_i)}{p_{\mathsf{x}_{i-1}}(x_{i-1})}\, p_{\mathsf{y}_i,\ldots,\mathsf{y}_N|\mathsf{x}_{i-1},\mathsf{x}_i}(y_i, \ldots, y_N|x_{i-1}, x_i) \qquad (12) \\ &= \sum_{x_i} p_{\mathsf{y}_i,\ldots,\mathsf{y}_N,\mathsf{x}_i|\mathsf{x}_{i-1}}(y_i, \ldots, y_N, x_i|x_{i-1}) \\ &= p_{\mathsf{y}_i,\ldots,\mathsf{y}_N|\mathsf{x}_{i-1}}(y_i, \ldots, y_N|x_{i-1}), \qquad (13) \end{aligned}$$

where to obtain (11) we have used that $\mathsf{y}_i \perp\!\!\!\perp (\mathsf{y}_{i+1}, \ldots, \mathsf{y}_N) \mid \mathsf{x}_i$, and where to obtain (12) we have used that $(\mathsf{y}_i, \ldots, \mathsf{y}_N) \perp\!\!\!\perp \mathsf{x}_{i-1} \mid \mathsf{x}_i$.

It remains only to verify (10) for $i = N-1$, which we obtain from (3c) as follows

$$\begin{aligned}
m_{N\to N-1}(x_{N-1}) &= \sum_{x_N} \phi_N(x_N)\, \psi_{N-1,N}(x_{N-1}, x_N) \\
&= \sum_{x_N} p_{y_N|x_N}(y_N|x_N)\, p_{x_N|x_{N-1}}(x_N|x_{N-1}) \\
&= \sum_{x_N} p_{y_N|x_N,x_{N-1}}(y_N|x_N, x_{N-1})\, p_{x_N|x_{N-1}}(x_N|x_{N-1}) \qquad (14) \\
&= \sum_{x_N} p_{x_N,y_N|x_{N-1}}(x_N, y_N|x_{N-1}) \\
&= p_{y_N|x_{N-1}}(y_N|x_{N-1}), \qquad (15)
\end{aligned}$$

where to obtain (14) we have used that $\mathsf{y}_N \perp\!\!\!\perp \mathsf{x}_{N-1} \mid \mathsf{x}_N$. □

In turn, the posterior belief computation (3e)–(3g) is naturally interpreted in terms of these messages. In particular, using (9) we see (3e) becomes

$$\begin{aligned}
p_{x_1|y_1,\ldots,y_N}(x_1|y_1,\ldots,y_N) &\propto \phi_1(x_1)\, m_{2\to 1}(x_1) \\
&= p_{x_1,y_1}(x_1, y_1)\, p_{y_2,\ldots,y_N|x_1}(y_2,\ldots,y_N|x_1) \\
&= p_{x_1}(x_1)\, p_{y_1|x_1}(y_1|x_1)\, p_{y_2,\ldots,y_N|x_1}(y_2,\ldots,y_N|x_1) \\
&= p_{x_1}(x_1)\, p_{y_1,\ldots,y_N|x_1}(y_1,\ldots,y_N|x_1) \qquad (16) \\
&= p_{y_1,\ldots,y_N,x_1}(y_1,\ldots,y_N,x_1),
\end{aligned}$$

where to obtain (16) we have used that $\mathsf{y}_1 \perp\!\!\!\perp (\mathsf{y}_2,\ldots,\mathsf{y}_N) \mid \mathsf{x}_1$.

Similarly, using (4) and (9) we see (3f) becomes, for $i = 2,\ldots,N-1$,

$$\begin{aligned}
&p_{x_i|y_1,\ldots,y_N}(x_i|y_1,\ldots,y_N) \\
&\quad\propto \phi_i(x_i)\, m_{i-1\to i}(x_i)\, m_{i+1\to i}(x_i) \\
&\quad= p_{y_i|x_i}(y_i|x_i)\, p_{y_1,\ldots,y_{i-1},x_i}(y_1,\ldots,y_{i-1},x_i)\, p_{y_{i+1},\ldots,y_N|x_i}(y_{i+1},\ldots,y_N|x_i) \\
&\quad= p_{y_1,\ldots,y_{i-1},x_i}(y_1,\ldots,y_{i-1},x_i)\, p_{y_i,\ldots,y_N|x_i}(y_i,\ldots,y_N|x_i) \qquad (17) \\
&\quad= p_{x_i}(x_i)\, p_{y_1,\ldots,y_{i-1}|x_i}(y_1,\ldots,y_{i-1}|x_i)\, p_{y_i,\ldots,y_N|x_i}(y_i,\ldots,y_N|x_i) \\
&\quad= p_{x_i}(x_i)\, p_{y_1,\ldots,y_N|x_i}(y_1,\ldots,y_N|x_i) \qquad (18) \\
&\quad= p_{y_1,\ldots,y_N,x_i}(y_1,\ldots,y_N,x_i),
\end{aligned}$$

where to obtain (17) we have used that $\mathsf{y}_i \perp\!\!\!\perp (\mathsf{y}_{i+1},\ldots,\mathsf{y}_N) \mid \mathsf{x}_i$, and where to obtain (18) we have used that $(\mathsf{y}_1,\ldots,\mathsf{y}_{i-1}) \perp\!\!\!\perp (\mathsf{y}_i\ldots,\mathsf{y}_N) \mid \mathsf{x}_i$.

Finally, using (4) we see (3g) becomes

$$\begin{aligned}
p_{x_N|y_1,\ldots,y_N}(x_N|y_1,\ldots,y_N) &\propto \phi_N(x_N)\, m_{N-1\to N}(x_N) \\
&= p_{y_N|x_N}(y_N|x_N)\, p_{y_1,\ldots,y_{N-1},x_N}(y_1,\ldots,y_{N-1},x_N) \\
&= p_{y_N|x_N}(y_N|x_N)\, p_{y_1,\ldots,y_{N-1}|x_N}(y_1,\ldots,y_{N-1}|x_N)\, p_{x_N}(x_N) \\
&= p_{y_1,\ldots,y_N|x_N}(y_1,\ldots,y_N|x_N)\, p_{x_N}(x_N) \qquad (19) \\
&= p_{y_1,\ldots,y_N,x_N}(y_1,\ldots,y_N,x_N),
\end{aligned}$$

where to obtain (19) we have used that $(\mathsf{y}_1, \ldots, \mathsf{y}_{N-1}) \perp\!\!\!\perp \mathsf{y}_N \mid \mathsf{x}_N$.

As these computations reflect, consistent with (1), the constant Z by which we ultimately divide to normalize these posterior beliefs has the probabilistic interpretation

$$Z = p_{\mathsf{y}_1,\ldots,\mathsf{y}_N}(y_1, \ldots, y_N). \tag{20}$$

It is worth noting that this normalization does *not* depend on i, so it can be precomputed at any node, and used at all other nodes.

24.3.2 (α, β)-Form of the Forward-Backward Algorithm

The forward-backward algorithm we have derived and interpreted in the preceding sections is a computationally efficient procedure for computating posterior beliefs in HMMs. As we discussed in developing the general sum-product algorithm, much of the efficiency results by recognizing that many computations can be shared in generating different posterior beliefs.

This insight can be applied in even richer ways, as we now illustrate in the context of HMMs. In particular, we have focused on computing posterior beliefs of the form

$$p_{\mathsf{x}_i|\mathsf{y}_1,\ldots,\mathsf{y}_N}(\cdot|y_1, \ldots, y_N). \tag{21}$$

However, when the node index represents time, for example, there are applications where one is interested also in posterior beliefs of the form

$$p_{\mathsf{x}_i|\mathsf{y}_1,\ldots,\mathsf{y}_i}(\cdot|y_1, \ldots, y_i). \tag{22}$$

These can be viewed as *causal* beliefs, since they correspond to beliefs of the present state, based on observations up to (and including) the current time. By contrast, (21) are more naturally viewed as *noncausal* beliefs, since they correspond to beliefs of the present state based on all past, present, and future observations. While noncausal beliefs are important on off-line inference, for on-line inference the causal beliefs (22), which can be computed in real-time, are often of greater inference.[2]

In this section, we show that the computations of the forward-backward algorithm can be restructured in relatively straightforward ways so as to produce *both* causal and noncausal beliefs *without* incurring any increase in computational complexity.

To develop the rearrangement of interest, it suffices to refactor (1) into an equivalent but different set of potentials. Specifically, we use [cf. (1)]

$$\begin{aligned} &p_{\mathsf{x}_1,\ldots,\mathsf{x}_N|\mathsf{y}_1,\ldots,\mathsf{y}_N}(x_1, \ldots, x_N|y_1, \ldots, y_N) \\ &\qquad \propto \underbrace{p_{\mathsf{x}_1,\mathsf{y}_1}(x_1, y_1)}_{\triangleq \phi'_1(x_1)} \prod_{i=2}^{N} \underbrace{p_{\mathsf{x}_i|\mathsf{x}_{i-1}}(x_i|x_{i-1})\, p_{\mathsf{y}_i|\mathsf{x}_i}(y_i|x_i)}_{\triangleq \psi'_{i-1,i}(x_{i-1},x_i)} \end{aligned} \tag{23}$$

[2]Causal beliefs correspond to what are sometimes referred to as "filtered" estimates of the state, while noncausal beliefs correspond to what are sometimes referred to as "smoothed" estimates of the state. This terminology is more common in the case of Gaussian HMMs, which we will return to later in the subject.

As (23) reflects, the associated reduced graph is still the Markov chain depicted in Fig. 1, but where the new potentials are

$$\phi'_1(x_1) = p_{x_1,y_1}(x_1, y_1) \tag{24a}$$
$$\phi'_i(x_i) = 1, \qquad i = 2, \ldots, N \tag{24b}$$
$$\psi'_{i,i+1}(x_i, x_{i+1}) = p_{x_{i+1}|x_i}(x_{i+1}|x_i)\, p_{y_{i+1}|x_{i+1}}(y_{i+1}|x_{i+1}), \qquad i = 1, \ldots, N-1. \tag{24c}$$

So required sum-product algorithm has the same form as (3), but with modified potentials and, hence, messages:

$$m'_{1\to 2}(x_2) = \sum_{x_1} \phi'_1(x_1)\, \psi'_{12}(x_1, x_2) \tag{25a}$$
$$m'_{i\to i+1}(x_{i+1}) = \sum_{x_i} \phi'_i(x_i)\, \psi'_{i,i+1}(x_i, x_{i+1})\, m'_{i-1\to i}(x_i), \qquad i = 2, 3, \ldots, N \tag{25b}$$
$$m'_{N\to N-1}(x_{N-1}) = \sum_{x_N} \phi'_N(x_N)\, \psi'_{N-1,N}(x_{N-1}, x_N) \tag{25c}$$
$$m'_{i\to i-1}(x_{i-1}) = \sum_{x_i} \phi'_i(x_i)\, \psi'_{i-1,i}(x_{i-1}, x_i)\, m'_{i+1\to i}(x_i), \qquad i = 1, 2, \ldots, N-1, \tag{25d}$$

and posterior belief computation

$$p_{x_1|y_1,\ldots,y_N}(x_1|y_1, \ldots, y_N) \propto \phi'_1(x_1)\, m'_{2\to 1}(x_1) \tag{25e}$$
$$p_{x_i|y_1,\ldots,y_N}(x_i|y_1, \ldots, y_N) \propto \phi'_i(x_i)\, m'_{i-1\to i}(x_i)\, m'_{i+1\to i}(x_i), \qquad i = 2, \ldots, N-1 \tag{25f}$$
$$p_{x_N|y_1,\ldots,y_N}(x_N|y_1, \ldots, y_N) \propto \phi'_N(x_N)\, m'_{N-1\to N}(x_N). \tag{25g}$$

Via analysis analogous to that used in Section 9.3.1, we can interpret the new messages; we leave their derivations as exercises. First, we interpret the new forward messages as follows.

Claim 3. *The new forward messages have the probabilistic interpretation*

$$\alpha_{i+1}(x_{i+1}) \triangleq m'_{i\to i+1}(x_{i+1}) = p_{y_1,\ldots,y_{i+1},x_{i+1}}(y_1, \ldots, y_{i+1}, x_{i+1}), \qquad i = 1, \ldots, N-1 \tag{26}$$

Evidently, the new forward messages, which we have relabeled $\alpha_i(x_i)$, are proportional to the causal belief of interest. Note that the old forward messages are related to the new ones via

$$
\begin{aligned}
m_{i\to i+1}(x_{i+1}) &= p_{\mathsf{y}_1,\ldots,\mathsf{y}_i,\mathsf{x}_{i+1}}(y_1,\ldots,y_i,x_{i+1}) \\
&= \sum_{x_i} p_{\mathsf{y}_1,\ldots,\mathsf{y}_i,\mathsf{x}_i,\mathsf{x}_{i+1}}(y_1,\ldots,y_i,x_i,x_{i+1}) \\
&= \sum_{x_i} p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i)\, p_{\mathsf{y}_1,\ldots,\mathsf{y}_i,\mathsf{x}_i}(y_1,\ldots,y_i,x_i) \\
&= \sum_{x_i} p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i)\, \alpha_i(x_i), \qquad (27)
\end{aligned}
$$

where we have used that $\mathsf{y}_1,\ldots,\mathsf{y}_i \perp\!\!\!\perp \mathsf{x}_{i+1} \mid \mathsf{x}_i$.

Second, we interpret the new backward messages as follows.

Claim 4. *The new backward messages have the probabilistic interpretation*

$$
\beta_i(x_i) \triangleq m'_{i+1\to i}(x_i) = m_{i+1\to i}(x_i) = p_{\mathsf{y}_{i+1},\ldots,\mathsf{y}_N|\mathsf{x}_i}(y_{i+1},\ldots,y_N|x_i), \quad i = 1,\ldots N-1. \qquad (28)
$$

Evidently, the new backward messages, which we have relabeled $\beta_i(x_i)$, are identical to the original backward messages; our refactorization has not changed them.

In terms of our new notation, and defining, for additional convenience,

$$
\alpha_1(x_1) \triangleq \phi'_1(x_1) = p_{\mathsf{x}_1,\mathsf{y}_1}(x_1,y_1) \qquad (29)
$$

$$
\beta_N(x_N) \triangleq 1, \qquad (30)
$$

we can rewrite (25) in the form

$$
\alpha_{i+1}(x_{i+1}) = \sum_{x_i} p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i)\, p_{\mathsf{y}_{i+1}|\mathsf{x}_{i+1}}(y_{i+1}|x_{i+1})\, \alpha_i(x_i), \qquad i = 1,\ldots,N-1 \qquad (31a)
$$

$$
\beta_i(x_i) = \sum_{x_{i+1}} p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i)\, p_{\mathsf{y}_{i+1}|\mathsf{x}_{i+1}}(y_{i+1}|x_{i+1})\, \beta_{i+1}(x_{i+1}), \qquad i = 1,\ldots,N-1, \qquad (31b)
$$

with the noncausal beliefs in the form

$$
\gamma_i(x_i) \triangleq p_{\mathsf{x}_i|\mathsf{y}_1,\ldots,\mathsf{y}_N}(x_i|y_1,\ldots,y_N) \propto \alpha_i(x_i)\, \beta_i(x_i), \qquad i = 1,\ldots,N. \qquad (31c)
$$

Eqs (31) are referred to as the (α,β)-form of the the forward-backward algorithm, which is among the most widely used versions, and among the earliest developed ones.

24.3.3 (α, γ)-Form of the Forward-Backward Algorithm

Finally, it should be noted that other versions of the forward-backward algorithm are sometimes used in practice, to obtain yet further advantages.

As an example, we begin by noting that both versions of the forward-backward algorithm we have developed so far use each observation y_i twice—once during the construction of message $m_{i \to i+1}$ (or α_i) as the forward pass (3b) reveals, and once during the construction of message $m_{i \to i-1}$ (or β_{i-1}), as the backward pass (3d) reveals. In particular, this observation enters via $\phi_i(x_i)$.

However, it is possible to restructure the computation in the forward-backward algorithm so that the data can be discarded after the forward pass is complete, i.e., so that the backward pass does not require further access to the data. Moreover, in the process, we can arrange to have the backward messages directly correspond to the noncausal beliefs of interest, as a further simplification.

To obtain this version of the forward-backward algorithm, use use the α_i messages developed above for the forward pass, then use the following backward recursion to obtain the noncausal posterior beliefs

$$\gamma_i(x_i) = \sum_{x_{i+1}} \left[\frac{p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i)\, \alpha_i(x_i)}{\sum_{x_i'} p_{\mathsf{x}_{i+1}|\mathsf{x}_i}(x_{i+1}|x_i')\, \alpha_i(x_i')} \right] \gamma_{i+1}(x_{i+1}). \tag{32}$$

This is referred to as the (α, γ) version of the forward-backward algorithm. We omit the derivation, which is straightforward. Details can be found, e.g., Chapter 12 of Jordan's notes.

24.4 The Viterbi Algorithm

The specialization of the max-product (or min-sum) algorithm to an HMM is referred to as the *Viterbi algorithm*. While now used to address a wide range of problems in many fields, Viterbi's development of the algorithm was originally motivated by a problem of reliable communication: that of decoding what are referred to as convolutional codes, which are naturally modeled using HMMs. Accordingly, we'll consider such an example.

In our convolutional code, the message m consists of random bits N bits. The coded message b consist of $2N-1$ bits, alternating between the following:

- The odd-numbered bits are the message bits, i.e., $\mathsf{b}_{2i-1} = \mathsf{m}_i$.
- The even-numbered bits are the exclusive-OR of adjacent message bits, i.e., $\mathsf{b}_{2i} = \mathsf{m}_i \oplus \mathsf{m}_{i+1}$.

During transmission, each of the bits of the coded message is independently flipped with probability ϵ; the corrupted coded bits are denoted using $\mathsf{y}_1, \mathsf{y}_2, \ldots$. The goal of the decoder is to infer the $\mathsf{m}_1, \mathsf{m}_2, \ldots$ from the $\mathsf{y}_1, \mathsf{y}_2, \ldots$.

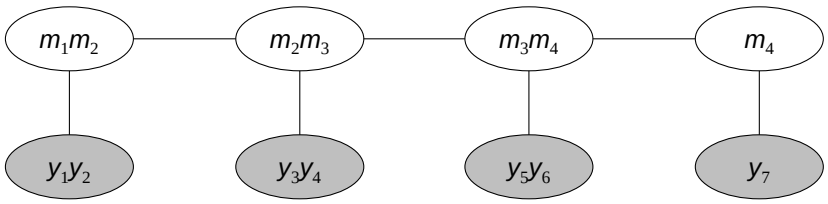


Figure 5: Representation of a convolutional code as an HMM via node aggregation.

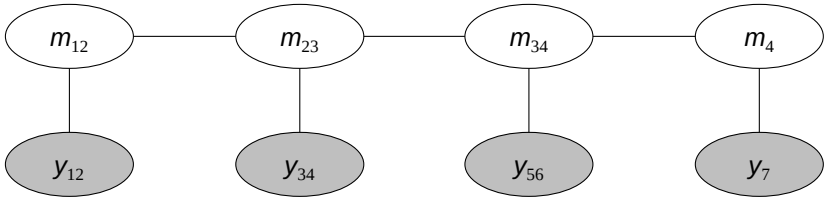


Figure 6: A relabeling of our convolutional code variables.

While this code has a natural represented in terms of a directed graph, the obvious undirected representation via moralization does not result in a tree. However, we can obtain an HMM representation via a node aggregation process. In particular, we combine all of the adjacent message bits into variables $m_i m_{i+1}$, and combine pairs of adjacent received message bits $y_{2i-1}y_{2i}$.

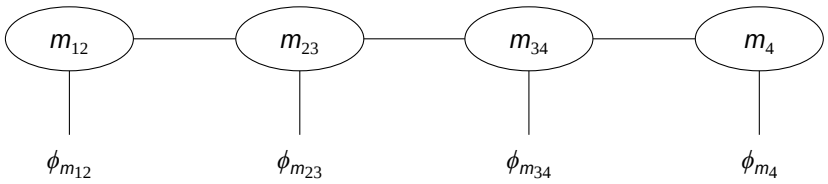
In the sequel, we restrict our attention to a block of 4 consecutive "superstates", as depicted in Fig. 5. To ease the notation a bit for what's to follow, we use $m_i m_{i+1} \triangleq m_{i,i+1}$ and $y_j y_{j+1} \triangleq y_{j,j+1}$, so Fig. 5 becomes Fig. 6.

Fix $\epsilon \in (0, 1/2)$. Let $w_\epsilon = \log\left(\frac{1-\epsilon}{\epsilon}\right) > 0$. Then the potential functions are given by:

$$\begin{aligned}
\psi_{m_{i,i+1},m_{i+1,i+2}}(\text{“}ab\text{”}, \text{“}cd\text{”}) &= \mathbb{1}_{b=c} && \text{for } i \in \{1,2\},\\
\psi_{m_{34},m_4}(\text{“}ab\text{”}, \text{“}c\text{”}) &= \mathbb{1}_{b=c},\\
\psi_{m_{i,i+1},y_{2i-1,2i}}(\text{“}ab\text{”}, \text{“}uv\text{”}) &= \exp\left\{\mathbb{1}_{a=u}w_\epsilon + \mathbb{1}_{a\oplus b=v}w_\epsilon\right\} && \text{for } i \in \{1,2,3\},\\
\psi_{m_4,y_7}(\text{“}a\text{”}, \text{“}u\text{”}) &= \exp\left\{\mathbb{1}_{a=u}w_\epsilon\right\},
\end{aligned}$$

where $a, b, c, d, u, v \in \{0, 1\}$ and "ab" denotes a length-2 bit-string.

Suppose we observe $\mathbf{y} = (0, 0, 0, 1, 0, 0, 0)$ and want to infer the MAP configuration for $\mathbf{m} = (m_1, m_2, m_3, m_4)$. Let's solve this using min-sum. The first thing to do is to incorporate the data into the model by plugging in the observed values; this will cause the resulting graphical model to just be a line:



The new singleton potentials are:

$$\begin{aligned}
\phi_{\boldsymbol{m}_{12}}(\text{“}ab\text{”}) &= \exp\left\{\mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=0}w_\epsilon\right\} \\
\phi_{\boldsymbol{m}_{23}}(\text{“}ab\text{”}) &= \exp\left\{\mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=1}w_\epsilon\right\} \\
\phi_{\boldsymbol{m}_{34}}(\text{“}ab\text{”}) &= \exp\left\{\mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=0}w_\epsilon\right\} \\
\phi_{\boldsymbol{m}_{4}}(\text{“}a\text{”}) &= \exp\left\{\mathbb{1}_{a=0}w_\epsilon\right\}
\end{aligned}$$

The pairwise potentials remain the same.

Since we're going to use min-sum, we write out the log-potentials:

$$\begin{aligned}
\log\phi_{\boldsymbol{m}_{12}}(\text{“}ab\text{”}) &= \mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=0}w_\epsilon \\
\log\phi_{\boldsymbol{m}_{23}}(\text{“}ab\text{”}) &= \mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=1}w_\epsilon \\
\log\phi_{\boldsymbol{m}_{34}}(\text{“}ab\text{”}) &= \mathbb{1}_{a=0}w_\epsilon + \mathbb{1}_{a\oplus b=0}w_\epsilon \\
\log\phi_{\boldsymbol{m}_{4}}(\text{“}a\text{”}) &= \mathbb{1}_{a=0}w_\epsilon \\
\log\psi_{\boldsymbol{m}_{i,i+1},\boldsymbol{m}_{i+1,i+2}}(\text{“}ab\text{”},\text{“}cd\text{”}) &= \begin{cases} 0 & \text{if } b=c \\ -\infty & \text{otherwise} \end{cases} \qquad \text{for } i\in\{1,2\} \\
\log\psi_{\boldsymbol{m}_{34},\boldsymbol{m}_{4}}(\text{“}ab\text{”},\text{“}c\text{”}) &= \begin{cases} 0 & \text{if } b=c \\ -\infty & \text{otherwise} \end{cases}
\end{aligned}$$

Using the min-sum message update equation,

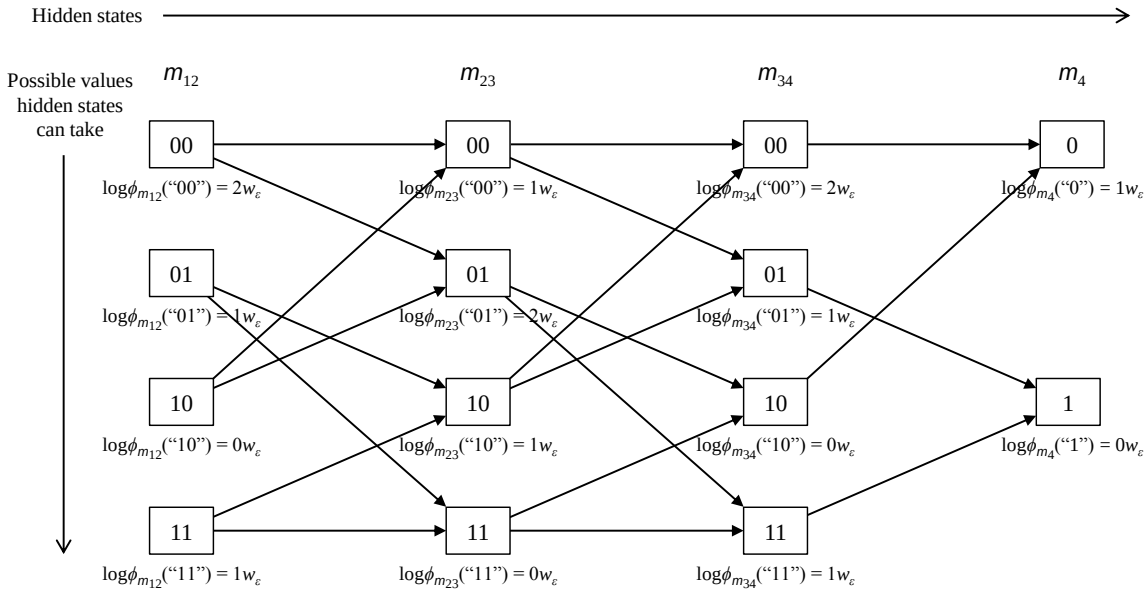
$$\begin{aligned}
&-\log m_{(i,i+1)\to(i+1,i+2)}(\text{“}cd\text{”}) \\
&= \min_{\text{“}ab\text{”}\in\{0,1\}^2}\left\{-\log\phi_{\boldsymbol{m}_{i,i+1}}(\text{“}ab\text{”}) - \log\psi_{\boldsymbol{m}_{i,i+1},\boldsymbol{m}_{i+1,i+2}}(\text{“}ab\text{”},\text{“}cd\text{”}) - \log m_{(i-1,i)\to(i,i+1)}(\text{“}ab\text{”})\right\} \\
&= \min_{a\in\{0,1\}}\left\{-\log\phi_{\boldsymbol{m}_{i,i+1}}(\text{“}ac\text{”}) - \log m_{(i-1,i)\to(i,i+1)}(\text{“}ac\text{”})\right\},
\end{aligned}$$

where the last equality uses the fact that what the pairwise potentials really are saying is that we only allow transitions where the last bit of the previous state is the first bit of the next state. This means that if we know the next state is “cd”, then the previous state must end in “c”, so we shouldn't bother optimizing over all bit-strings of length 2.

Note that if we pass messages from the left-most node to the right-most node, then what $\log m_{(i,i+1)\to(i+1,i+2)}(\text{“}cd\text{”})$ represents is the highest “score” achievable that ends in $\boldsymbol{m}_{i+1,i+2} = \text{“}cd\text{”}$ but that does not include the contribution of the observation

associated with $m_{i+1,i+2}$. The goal of max-sum would then be to find the MAP configuration with the highest overall score, with contributions from all nodes' singleton potentials and pairwise potentials.

We can visualize this setup using a *trellis diagram*:



For each hidden state's possible value, incoming edges correspond to which possible previous states are possible. The MAP configuration is obtained by summing across scores (i.e., the log singleton potential values in this case) along a path that goes from one of the left-most nodes to the right-most node of the trellis diagram. If we keep track of which edges we take that maximizes[3] the score along the path we traverse, then we can backtrack once we're at the end to figure out what the best configuration is, as was done in the MAP elimination algorithm from last lecture.

The key idea for the min-sum algorithm is that we basically iterate through the vertical layers (i.e., hidden states) in the trellis diagram one at a time from left to right. At each vertical layer, we store the highest possible score achievable that arrives at each possible value that the hidden state can take. Note that for this example, the edges do not have scores associated with them; in general, edges may have scores as well.

With this intuition, we run the max-sum with the modification that we keep track of which optimizing values give rise to the "highest score so far" at each hidden state's possible value. At the end, we backtrack to obtain the global MAP configuration $(00, 00, 00, 0)$, which achieves a overall score of $6w_\epsilon$.

[3]Since we have left off the negation from our log-potentials, the minimization becomes a maximization.