Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.438 Algorithms For Inference
Fall 2014

9 Forward-backward algorithm, sum-product on factor graphs

The previous lecture introduced belief propagation (sum-product), an efficient inference algorithm for tree structured graphical models. In this lecture, we specialize it further to the so-called *hidden Markov model (HMM)*, a model which is very useful in practice for problems with temporal structure.

9.1 Example: convolution codes

We motivate our discussion of HMMs with a kind of code for communication called a *convolution code*. In general, the problem of communication is that the sender would like to send a message $\boldsymbol{m}$, represented as a bit string, to a receiver. The message may be corrupted along the way, so we need to introduce redundancy into the message so that it can be reconstructed accurately even in the presence of noise. To do this, the sender sends a coded message $\boldsymbol{b}$ over a noisy channel. The channel introduces some noise (e.g. by flipping random bits). The receiver receives the "received message" $\boldsymbol{y}$ and then applies a decoding procedure to get the decoded message $\hat{\boldsymbol{m}}$. A schematic is shown in Figure 1. Clearly, we desire a coding scheme where $\hat{\boldsymbol{m}} = \boldsymbol{m}$ with high probability, $\boldsymbol{b}$ is not much larger than $\boldsymbol{m}$, and $\hat{\boldsymbol{m}}$ can be efficiently computed from $\boldsymbol{y}$.

We now discuss one example of a coding scheme, called a convolution code. Suppose the message $\boldsymbol{m}$ consists of N bits. The coded message $\boldsymbol{b}$ will consist of $2N-1$ bits, alternating between the following:

- The odd-numbered bits $\boldsymbol{b}_{2i-1}$ repeat the message bits $\boldsymbol{m}_i$ exactly.
- The even-numbered bits $\boldsymbol{b}_{2i}$ are the XOR of message bits $\boldsymbol{m}_i$ and $\boldsymbol{m}_{i+1}$, denoted $\boldsymbol{m}_i \oplus \boldsymbol{m}_{i+1}$.

The ratio of the lengths of $\boldsymbol{m}$ and $\boldsymbol{b}$ is called the *rate* of the code, so this convolution code is a rate $\frac{1}{2}$ code, i.e. for every coded message bit, it can convey $\frac{1}{2}$ message bit. We assume an error model called a *binary symmetric channel*: each of the bits of the coded message is independently flipped with probability ε. We can represent this as a directed graphical model as shown in Figure 2. Note that from the receiver's perspective, only the $\boldsymbol{y}_i$'s are observed, and the task is to infer the $\boldsymbol{m}_i$'s.

In order to perform inference, we must convert this graph into an undirected graphical model. Unfortunately, the straightforward construction, where we moralize the graph, does not result in a tree structure, because of the cliques over $\boldsymbol{m}_i$, $\boldsymbol{m}_{i+1}$, and $\boldsymbol{b}_{2i}$. Instead, we coarsen the representation by combining nodes into "supernodes." In particular, we will combine all of the adjacent message bits into variables $\boldsymbol{m}_i\boldsymbol{m}_{i+1}$, and

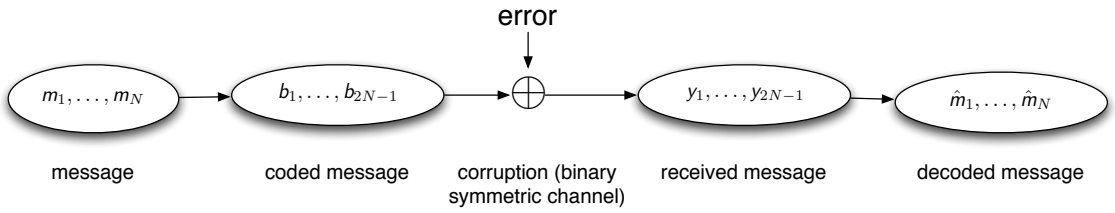

Figure 1: A schematic representation of the problem setup for convolution codes.

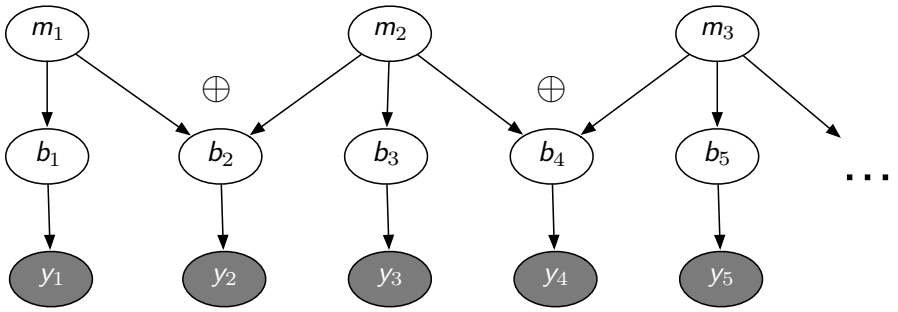

Figure 2: Our convolution code can be represented as a directed graphical model.

we will combine pairs of adjacent received message bits $y_{2i-1}y_{2i}$, as shown in Figure 3. This results in a tree-structured directed graph, and therefore an undirected tree graph — now we can perform sum-product.

9.2 Hidden Markov models

Observe that the graph in Figure 3 is *Markov* in its hidden states. More generally, a *hidden Markov model (HMM)* is a graphical model with the structure shown in Figure 4. Intuitively, the variables x_i represent a state which evolves over time and which we don't get to observe, so we refer to them as the *hidden state*. The variables y_i are signals which depend on the state at the same time step, and in most applications are observed, so we refer to them as *observations*.

From the definition of directed graphical models, we see that the HMM represents the factorization property

$$\mathbf{P}(x_1, \ldots, x_N, y_1, \ldots, y_N) = \mathbf{P}(x_1) \prod_{i=2}^{N} \mathbf{P}(x_i|x_{i-1}) \prod_{j=1}^{N} \mathbf{P}(y_j|x_j). \tag{1}$$

Observe that we can convert this to the undirected representation shown in Figure 4 (b) by taking each of the terms in this product to be a potential. This allows us to

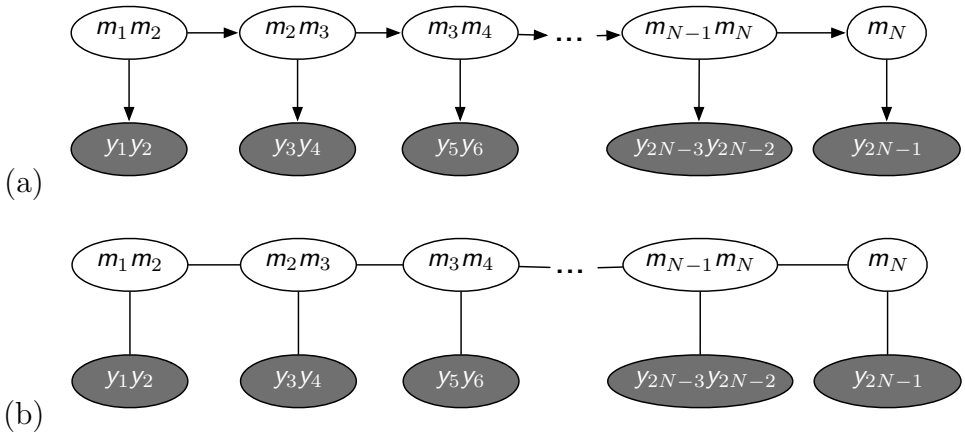

Figure 3: (a) The directed graph from figure 2 can be converted to a tree-structured graph with combined variables. (b) The equivalent undirected graph.

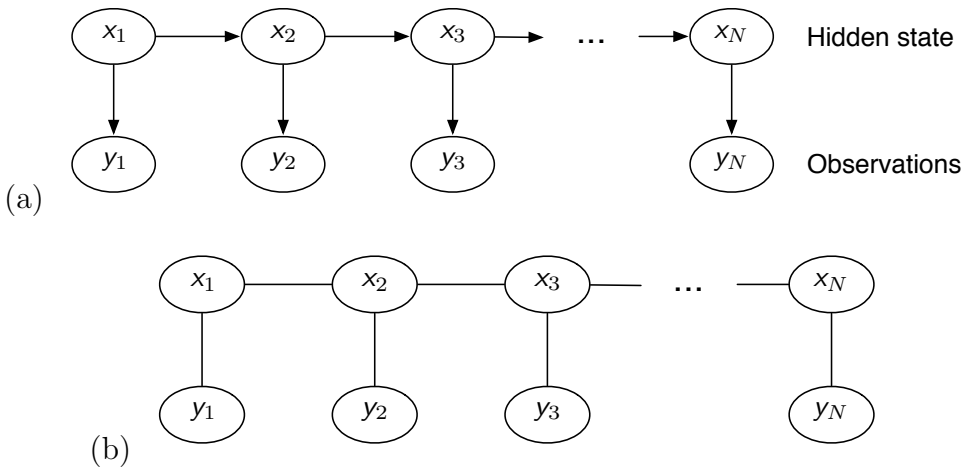

Figure 4: (a) a hidden Markov model, and (b) its undirected equivalent.

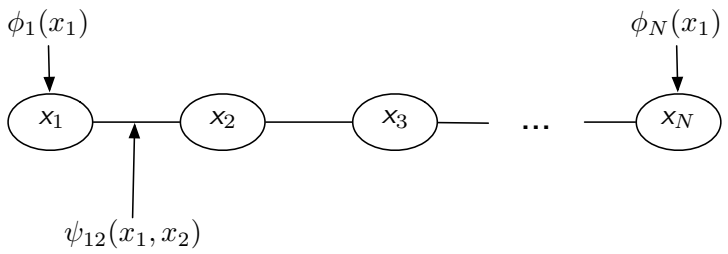

Figure 5: Representation of an HMM as a chain graph.

perform inference using sum product on trees. In particular, our goal is typically to infer the marginal distribution for each of the hidden states given all of the observations. By plugging in the values of the observations, we can convert the HMM to a chain graph, as shown in Figure 5. This graphical model has two sets of potentials:

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

The resulting sum-product messages are:

$$m_{1\to 2}(x_2) = \sum_{x_1} \phi_1(x_1)\psi_{12}(x_1, x_2) \tag{3}$$
$$m_{i\to i+1}(x_{i+1}) = \sum_{x_i} \phi_i(x_i)\psi_{i,i+1}(x_i, x_{i+1})m_{i-1\to i}(x_i) \qquad 2 \le i \le N \tag{4}$$
$$m_{N\to N-1}(x_{N-1}) = \sum_{x_N} \phi_N(x_N)\psi_{N-1,N}(x_{N-1}, x_N) \tag{5}$$
$$m_{i\to i-1}(x_{i-1}) = \sum_{x_i} \phi_i(x_i)\psi_{i-1,i}(x_{i-1}, x_i)m_{i+1\to i}(x_i) \qquad 1 \le i \le N-1 \tag{6}$$

Belief propagation on HMMs is also known as the *forward-backward algorithm.*

You can easily show that

$$\begin{aligned} m_{1\to 2}(x_2) &= \sum_{x_1} \phi_1(x_1)\psi_{12}(x_1, x_2) = \sum_{x_1} \mathbf{P}(x_1, y_1)\mathbf{P}(x_2|x_1) \\ &= \sum_{x_1} \mathbf{P}(x_1, y_1, x_2) = \mathbf{P}(y_1, x_2). \\ m_{2\to 3}(x_3) &= \sum_{x_2} \phi_2(x_2)\psi_{23}(x_2, x_3)m_{1\to 2}(x_2) = \sum_{x_2} \mathbf{P}(y_2|x_2)\mathbf{P}(x_3|x_2)\mathbf{P}(y_1, x_2) \\ &= \sum_{x_2} \mathbf{P}(x_3, x_2, y_2, y_1) = \mathbf{P}(y_1, y_2, x_3). \end{aligned}$$

Continuing in this fashion, we can show that:

$$m_{i-1\to i}(x_i) = \mathbf{P}(y_1, y_2, \ldots, y_{i-1}, x_i).$$

Similarly:

$$\begin{aligned}
m_{N\to N-1}(x_{N-1}) &= \sum_{x_N} \phi_N(x_N)\psi_{N-1,N}(x_{N-1}, x_N) = \sum_{x_N} \mathbf{P}(y_N|x_N)\mathbf{P}(x_N|x_{N-1}) \\
&= \sum_{x_N} \mathbf{P}(x_N, y_N|x_{N-1}) = \mathbf{P}(y_N|x_{N-1}). \\
m_{N-1\to N-2}(x_{N-2}) &= \sum_{x_{N-1}} \phi_{N-1}(x_{N-1})\psi_{N-2,N-1}(x_{N-2}, x_{N-1}) \\
&= \sum_{x_{N-1}} \mathbf{P}(y_{N-1}|x_{N-1})\mathbf{P}(x_{N-1}|x_{N-2})\mathbf{P}(y_N|x_{N-1}) \\
&= \sum_{x_{N-1}} \mathbf{P}(y_{N-1}, y_N, x_{N-1}|x_{N-2}) = \mathbf{P}(y_{N-1}, y_N|x_{N-2}).
\end{aligned}$$

Continuing in this fashion, we get:

$$m_{i+1\to i}(x_i) = \mathbf{P}(y_{i+1}, y_{i+2}, \ldots, y_N|x_i).$$

9.2.1 α, β Forward-Backward Algorithms and Probabilistic interpretation of messages

As we have seen, belief propagation on HMMs takes the form of a two-pass algorithm, consisting of a forward pass, and a backward pass. In fact there is considerably flexibility on how computation is structured in the algorithm, even with the two-pass structure. As a result, we refer to this algorithm and its variants collectively as the forward-backward algorithm.

To illustrate how the computation can be rearranged in useful ways, in this section we highlight one variant of the forward-backward algorithm, termed the α, β version for reasons that will become apparent. The α, β forward-backward algorithm was also among the earliest versions developed.

To see the rearrangement of interest, first note that each posterior marginal of interest $p_{x_i|y_1,\ldots,y_N}$ is proportional to $p_{x_i,y_1,\ldots,y_N}$ and that from the graphical model in Figure 4, $(y_1, \ldots, y_i)$ are d-separated from $(y_{i+1}, \ldots, y_N)$ given x_i. Therefore, the joint distribution factorizes as:

$$\begin{aligned}
p_{y_1,\ldots,y_N,x_i} &= p_{y_1,\ldots,y_i,x_i} p_{y_{i+1},\ldots,y_N|x_i,y_1,\ldots,y_i} &(7)\\
&= p_{y_1,\ldots,y_i,x_i} p_{y_{i+1},\ldots,y_N|x_i}. &(8)
\end{aligned}$$

Now we derive recursive update rules for computing each of the two parts. First, the forward messages α:

$$\begin{aligned}
\alpha_i(x_i) &= \mathbf{P}(y_1, \ldots, y_i, x_i) && (9)\\
&= \sum_{x_{i-1}} \mathbf{P}(y_1, \ldots, y_i, x_i, x_{i-1}) && (10)\\
&= \sum_{x_{i-1}} \mathbf{P}(y_1, \ldots, y_{i-1}, x_{i-1}) \mathbf{P}(x_i|x_{i-1}) \mathbf{P}(y_i|x_i) && (11)\\
&= \sum_{x_{i-1}} \alpha_{i-1}(x_{i-1}) \mathbf{P}(x_i|x_{i-1}) \mathbf{P}(y_i|x_i), \forall i \in \{2, \ldots, N\}, && (12)
\end{aligned}$$

where $\alpha_1(x_1) = \mathbf{P}(x_1, y_1)$.

Then, the backward messages β:

$$\begin{aligned}
\beta_i(x_i) &= \mathbf{P}(y_{i+1}, \ldots, y_N|x_i) && (13)\\
&= \sum_{x_{i+1}} \mathbf{P}(x_{i+1}, y_{i+1}, \ldots, y_N|x_i) && (14)\\
&= \sum_{x_{i+1}} \mathbf{P}(y_{i+1}|x_{i+1}) \mathbf{P}(x_{i+1}|x_i) \mathbf{P}(y_{i+2}, \ldots, y_N|x_{i+1}) && (15)\\
&= \sum_{x_{i+1}} \mathbf{P}(y_{i+1}|x_{i+1}) \mathbf{P}(x_{i+1}|x_i) \beta_{i+1}(x_{i+1}), \forall i \in \{1, \ldots, N-1\}, && (16)
\end{aligned}$$

where $\beta_N(x_N) = 1$.

Now consider the relationship between α and β and our sum-product messages. The forward message corresponds to part of the formula for α,

$$\begin{aligned}
m_{i \to i+1}(x_{i+1}) &= \mathbf{P}(y_1, \ldots, y_i, x_{i+1}) = \sum_{x_i} \mathbf{P}(y_1, \ldots, y_i, x_i, x_{i+1})\\
&= \sum_{x_i} \mathbf{P}(x_{i+1}|x_i) \mathbf{P}(y_1, \ldots, y_i, x_i) = \sum_{x_i} \mathbf{P}(x_{i+1}|x_i) \alpha_i(x_i). && (17)
\end{aligned}$$

Observe that the α messages and the forward messages in belief propagation perform exactly the same series of sums and products, but they divide up the steps differently. Following an analogous line of reasoning, we find that the backwards messages are identical, i.e. $\beta_i(x_i) = m_{i+1 \to i}(x_i)$. This gives a useful probabilistic interpretation to the sum-product messages.

We can see directly that the marginal computation corresponding to

$$\gamma_i(x_i) \triangleq p_{\mathsf{x}_i}(x_i|y_1, \ldots, y_N) \propto p_{\mathsf{x}_i}(x_i, y_1, \ldots, y_N) = \phi_i(x_i) m_{i-1 \to i}(x_i) m_{i+1 \to i}(x_i)$$

in the sum-product algorithm is, in terms of the and α, β messages,

$$\gamma_i(x_i) = \frac{\alpha_i(x_i) \beta_i(x_i)}{\mathbf{P}(y_1, \ldots, y_N)}.$$

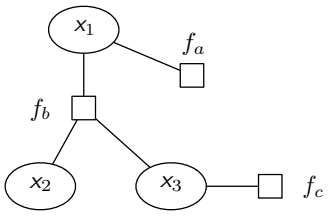

Figure 6: An example of factor graph belief propagation.

In turn, we note that the likelihood $\mathbf{P}(y_1, \ldots, y_N)$ is obtained from the messages at any node:

$$\mathbf{P}(y_1, \ldots, y_N) = \sum_{x_i} \alpha_i(x_i)\beta_i(x_i), \forall i.$$

One appealing feature of this form of the forward-backward algorithm is that the α messages are, themselves, marginals that are often directly useful in applications.

In particular,

$$\alpha_i(x_i) \propto \mathbf{P}(x_i|y_1, \ldots, y_i),$$

i.e., $\alpha_i(x_i)$ represents the marginal at state node i using data up to observation node i, and as such represents a "causal" marginal of particular interest in real-time applications. Sometimes these are referred to as "filtered" marginals, for reasons that we will come back to when we revisit the forward-backward algorithm in the context of Gaussian HMMs.

Finally, it should be noted that several other closely related variants of the forward-backward algorithm are also possible. One is the so-called α, γ version, whereby the recursion for β is replaced with a recursion for γ, which has the benefit of producing the desired marginals directly as messages, and the additional benefit that each piece of data y_i is used only once, in the forward pass. Specifically, the recursion for the γ messages can be shown to be (see, e.g., Jordan's notes, Chapter 12):

$$\gamma_i(x_i) = \sum_{x_{i+1}} \left[\frac{\mathbf{P}(x_{i+1}|x_i)\alpha_i(x_i)}{\sum_{x_i'} \mathbf{P}(x_{i+1}|x_i')\alpha_i(x_i')} \right] \gamma_{i+1}(x_{i+1}).$$

9.3 Sum-product on factor graphs

We now consider how to perform the sum product algorithm on a slightly more general class of graphical models, tree-structured factor graphs. Observe that this is a strictly larger set of graphs than undirected or directed trees. In either of these two cases, there is one factor for each edge in the original graph, so their equivalent factor graph representation is still a tree. However, some non-tree-structured directed or undirected graphs may have tree structured factor graph representations. One

important example is *polytrees*: recall that these are directed graphs which are tree-structured when we ignore the directions of the edges. For instance, observe that the convolution code directed graph shown in Figure 2 is a polytree, even though its undirected representation (obtained by moralizing) is not a tree. Using factor graph belief propagation, it is possible to perform inference in this graph without resorting to the supernode representation of Section 9.1.

In factor graph belief propagation, messages are sent between variable nodes and factor nodes. As in undirected belief propagation, each node sends messages to one neighbor by multiplying and/or summing messages from its other neighbors. Factor nodes a multiply incoming messages by their factor and sum out all but the relevant variable i:

$$m_{a\to i}(x_i) \;=\; \sum_{x_{N(a)\setminus\{i\}}} f_a(x_i, x_{N(a)\setminus\{i\}}) \prod_{j\in N(a)\setminus\{i\}} m_{j\to a}(x_j) \tag{18}$$

The variable nodes simply multiply together their incoming messages:

$$m_{i\to a}(x_i) \;=\; \prod_{b\in N(i)\setminus\{a\}} m_{b\to i}(x_i) \tag{19}$$

For instance, consider Figure 6. The messages required to compute $p_{x_1}(x_1)$ are:

$$\begin{aligned}
m_{a\to 1}(x_1) &= f_a(x_1) && (20)\\
m_{2\to b}(x_2) &= 1 && (21)\\
m_{c\to 3}(x_3) &= f_c(x_3) && (22)\\
m_{3\to b}(x_3) &= m_{c\to 3}(x_3) && (23)\\
m_{b\to 1}(x_1) &= \sum_{x_2,x_3} f_b(x_1,x_2,x_3) m_{2\to b}(x_2) m_{3\to b} m(x_3) && (24)
\end{aligned}$$

6.438 Algorithms for Inference

Fall 2014