11.1 SP algorithm on trees

We introduce the Sum Product algorithm on trees. For a tree graph, the SP algorithm eliminates the orders based on depth-first traversal. The elimination of each node can be considered as message passing directly along the tree branches in the directed tree graphs. An example with the tree model in Fig. 11.1, where we start from the bottom ($x_4$ and $x_5$) to the top ($x_1$). The goal is to find $P_{X_1}(x_1)$.

\[ P_{X_1,X_2,X_3,X_4,X_5}(x_1,x_2,x_3,x_4,x_5) = \prod_{i=1}^{5} \phi_i(x_i) \prod_{i=1}^{5} f_i(x_i,x_{π(i)}) \]

\[ = \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_5(x_5)f_4(x_2,x_4)f_5(x_2,x_5)f_2(x_2,x_1)f_3(x_3,x_1). \]

Denote $m_{i\rightarrow j}(x_j)$ the factor resulting from eliminating variables from below up to $x_j$. We have

\[ m_{4\rightarrow 2}(x_2) = \sum_{x_4} \phi_4(x_4)f_4(x_4,x_2), \]

\[ m_{5\rightarrow 2}(x_2) = \sum_{x_5} \phi_5(x_5)f_5(x_5,x_2). \]

Thus, for $x_1$, $x_2$, and $x_3$:

\[ P_{X_1,X_2,X_3}(x_1,x_2,x_3) = \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)f_3(x_3,x_1)f_2(x_2,x_1)m_{4\rightarrow 2}(x_2)m_{5\rightarrow 2}(x_2). \]
Algorithm 1 Iterative Sum-Product algorithm.

1: Initialize: \( m_{i \rightarrow j}^{(0)} = 1, \forall (i, j) \).
2: for \( \text{iter}=1:\text{maxiter} \) do
3: For all edges
4: \( m_{i \rightarrow j}^{(\text{iter})}(x_j) = \sum_{x_i} \phi_i(x_i) f_{i \rightarrow j}(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_k^{(\text{iter}-1)}(x_i) \)
5: end for

Then, for the first layer, we have

\[
m_{2 \rightarrow 1}(x_1) = \sum_{x_2} \phi_2(x_2) f_2(x_2, x_1) m_{4 \rightarrow 2}(x_2) m_{5 \rightarrow 2}(x_2),
\]

\[
m_{3 \rightarrow 1}(x_1) = \sum_{x_3} \phi_3(x_3) f_3(x_3, x_1).
\]

Finally we get \( P_{X_1}(x_1) = \phi_1(x_1) m_{2 \rightarrow 1}(x_1) m_{3 \rightarrow 1}(x_1) \). According to this procedure, we can conclude that the complexity is \( \mathcal{O}(N|\mathcal{X}|^2) \), where \( \mathcal{X} \) represents the alphabet corresponding to the random variables.

In general, let \( N(i) \) denote the neighbors of \( x_i \),

\[
m_{i \rightarrow j}(x_j) = \sum_{x_i} \phi_i(x_i) f_i(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_k^{(\text{iter})}(x_i).
\]

Note that the above algorithm finds the marginal, not just for \( x_1 \), but for all nodes in the system. That is

\[ P_{X_i}(x_i) = \Phi_i(x_i) \prod_{j \in N(i)} m_{j \rightarrow i}(x_i) \]

The detail of the algorithm can be generalized in 1. It converges in \( d \) iterations, where \( d \) is the diameter of the graph. This algorithm has been used and shown to work well in practice, even if the graph is not tree-like, especially in applications to decoding.

11.2 Hidden Markov Model

11.2.1 Hidden Markov Model

A Hidden Markov Model (HMM) is a model in which you observe a sequence of emissions \( Y_1, Y_2, \ldots, Y_N \) without knowing the sequence of states \( X_1, X_2, \ldots, X_N \) the model went through to generate the emissions; the sequence of states \( X_1, X_2, \ldots, X_N \) forms a Markov Process. This is mathematically represented as:

\[
P_{Y_1, Y_2, \ldots, Y_N, X_1, X_2, \ldots, X_N}(y_1, \ldots, y_N, x_1, \ldots, x_N) = \prod_{i=1}^{N} P_{Y_i|X_i}(y_i|x_i) \times P_{X_1}(x_1) \times \prod_{i=2}^{N} P_{X_i|X_{i-1}}(x_i|x_{i-1})
\]

See Fig. 11.2. In the sequel, we represent \( \bar{X} = (X_1, X_2, \ldots, X_N), \bar{Y} = (Y_1, Y_2, \ldots, Y_N) \). The quantities \( \bar{x}, \bar{y} \) are also defined similarly.

In the HMM, given observations \( Y_i = y_i, i = 1, 2, \ldots, N \), we are typically interested in obtaining (i) \( P_{X_i|Y_1, Y_2, \ldots, Y_N}(x_i|y_1, y_2, \ldots, y_N) \), or (ii) \( \max_{\bar{x}} P_{\bar{X}|\bar{Y}}(\bar{x}|\bar{y}) \) or (iii) \( P_{X_{i+1}|Y_i, Y_{i-1}, \ldots, Y_1}(x_{i+1}|y_i, y_{i-1}, \ldots, y_1) \).

Quantities (i),(iii) are obtained via the sum-product algorithm whereas quantity (ii) is obtained via the Viterbi algorithm. In coding theory, Hidden Markov models are useful in the context of convolutional codes.
11.2.2 Convolutional codes

The encoder of a convolutional codes has memory and has \( N \) outputs, where the \( i \)-th output bit depends on the \( i \)-th input bit and \( \ell \) previous input bits. Consider the example of Fig. 11.3, where \( N = 2k - 1 \), and

\[
b_i = \begin{cases} 
  m_i/2 + m_{i+2}/2 & \text{if } i \text{ is even} \\
  m_{i+1}/2 & \text{if } i \text{ is odd}
\end{cases}
\]

where \( \vec{m} = (m_1, m_2, \ldots, m_k) \) are the message bits.

This convolutional code can be thought of as a Hidden Markov model where, \( X_1 = m_1 \), \( X_2 = (m_1, m_2) \), \( X_3 = m_2, X_4 = m_2, m_3, X_5 = m_3, X_6 = (m_3, m_4), X_7 = m_4 \). In general, the output depends on the input and the previous inputs, as shown in Fig.11.4

If \( (b_1, b_2, \ldots, b_N) \) are input to a memoryless channel \( P_{Y|X} \), then the outputs \( Y_1, Y_2, \ldots, Y_N \) can be represented as Hidden Markov model as follows:

\[
P_{X,Y}(\vec{x}, \vec{y}) = \prod_{i=1}^{N} P_{Y|X}(y_i|x_i) \times P_{X_1}(x_1) \times \prod_{i=2}^{N} P_{X_{i+1}|X_i}(x_{i+1}|x_i)
\]

\[
= \prod_{i=1}^{N} \phi_i(x_i) \prod_{i=2}^{N} f_i(x_i, x_{i-1}),
\]
where \( \phi_1(x_1) = P_{Y,X}(y_1, x_1) \) and \( \phi_i(x_i) = P_{Y|X}(y_i| x_i), i = 2, 3, \ldots, N \) are the potentials, and \( f_i(x_i, x_{i-1}) = P_{X_i|X_{i-1}}(x_i|x_{i-1}) \).

In the sum-product algorithm, the messages can be represented as:

\[
\begin{align*}
    m_{i-1}^{i+1} \rightarrow i(x_i) &= \sum_{x_{i-1}} m_{i-1} \rightarrow i(x_i) \phi_i(x_i)
m_{i}^{i+1} \rightarrow i(x_i) &= \sum_{x_{i+1}} m_{i} \rightarrow i+1(x_{i+1}) f(x_i, x_{i+1})
\end{align*}
\]

In the context of HMMs and convolutional codes, the sum-product algorithm above is often referred to as the BCJR algorithm, or the forward-backward algorithm.

### 11.2.3 Trellis

A HMM can be represented by a Trellis, which is a state diagram being expanded across the states to show the passage of time. In the Trellis diagram, every possible code word is represented by a single unique path. Fig.11.5 shows an example of a Trellis diagram from a 4 states HMM.

![Trellis example](image)

The ML codeword can be found by drawing a Trellis and then using the Viterbi algorithm over the trellis (exercise for the student to figure how this works.).