18-661 Introduction to Machine Learning

Graphical Models II - Message-Passing (Belief Propagation) Algorithms

Fall 2020

ECE – Carnegie Mellon University

Outline

1. Review of Probabilistic Graphical Models
2. Review of Bayes Ball Theorem (d-separation)
3. Sum-Product Message-Passing to Find Marginals on Trees
4. Sum-Product Message-Passing to Find Posteriors
5. Forward-backward Algorithm for Hidden Markov Models

Midterm Information

Midterm will be on Tuesday, 10/20 in-class.

- Conducted as an online exam on Gradescope, with multiple-choice and short-answer questions
- Closed-book except for one double-sided letter-size handwritten page of notes that you can prepare as you wish.
- We will provide formulas for relevant probability distributions.
- You will not need a calculator. Only pen/pencil and scratch paper are allowed.

Will cover all topics up to and including Nearest Neighbors (10/15)

- (1) point estimation/MLE/MAP, (2) linear regression, (3) naive Bayes, (4) logistic regression, (5) SVMs, (6) Graphical Models, (7) Nearest Neighbors.
- Practice Midterm exam has been posted on Gradescope

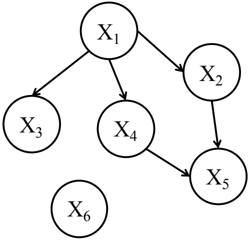
Homework

- Hw4 released – due on Oct 25th, after the midterm
- To give you additional flexibility with the homeworks, your lowest homework score of the semester will not be considered in the final grade

Review of Probabilistic Graphical Models

Directed Graphical Models (also called Bayesian Networks)

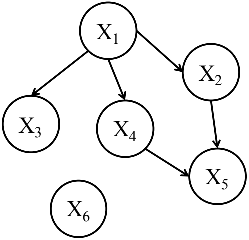
- **Nodes** represent random variables
- **Edges** represent conditional dependencies
- Directed acyclic graph – no loops



Advantages

1. Compact way of describing a family of joint dist. (last lecture)
2. Enable us to visualize conditional dependencies (last lecture)
3. Enable us to perform inference using observed data (this lecture)

Compact Way of Writing the Joint Distribution



Method to write the joint dist. described by any directed acyclic graph

$$
\begin{aligned}
p(x_1, x_2, \ldots, x_n) &= \prod_{i=1}^{n} p(x_i | x_{\pi_i}) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_1)p(x_5|x_2, x_4)p(x_6)
\end{aligned}
$$

where x_{π_i} is the set of parents of node i. For example,

- $x_{\pi_1} = \{\}$, $x_{\pi_6} = \{\}$
- $x_{\pi_2} = \{x_1\}$, $x_{\pi_3} = \{x_1\}$, $x_{\pi_4} = \{x_1\}$
- $x_{\pi_5} = \{x_2, x_4\}$

Storage Complexity of the Joint Distribution

- Due to conditional independencies, storage complexity is reduced
- Each node with d parents needs to store a $d + 1$ dimensional table

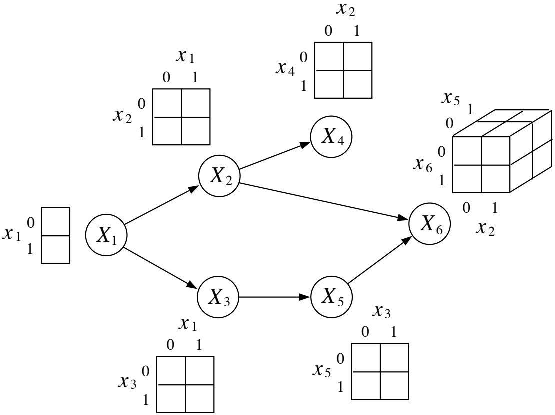
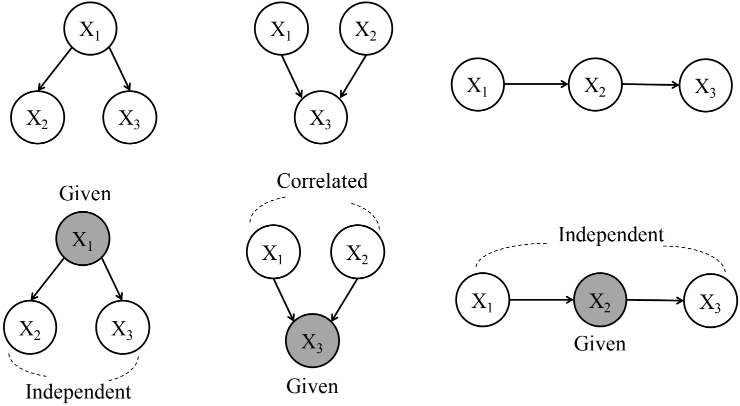


Image Source: Prof. Michael Jordan's lecture notes on Probabilistic Graphical Models

Review of Bayes Ball Theorem (d-separation)

Finding Variable Dependencies from a Graphical Model

- For the three canonical graphs, we inferred the conditional independences



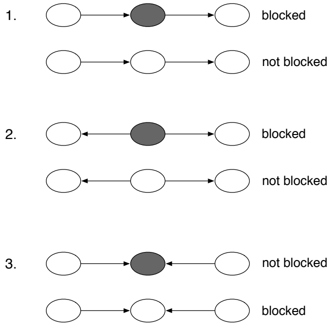
$X_2 \perp\!\!\!\perp X_3 | X_1$ $\quad$ $X_2 \not\perp\!\!\!\perp X_3 | X_1$ $\quad$ $X_1 \perp\!\!\!\perp X_3 | X_2$

- How do you identify these for a general graph?

Bayes Ball Theorem (also called d-separation)

Checking conditional dependencies between two nodes i and j given the observed values of nodes in set $\mathcal{S}$ (can also be an empty set).

- Shade the set of observed nodes $\mathcal{S}$ in grey
- Imagine a ball placed at node i. We want to move it to j
- The ball's movement along each edge is governed by the rules



- If the ball does not reach X_j, then $X_i \perp\!\!\!\perp X_j | X_{\mathcal{S}}$. Else $X_i \not\perp\!\!\!\perp X_j | X_{\mathcal{S}}$

Example of using the Bayes Ball algorithm

Graph 1

- $X_1 \perp\!\!\!\perp X_4 | \{X_2, X_3\}$
- $X_5 \not\perp\!\!\!\perp X_6$
- Ques: Is $X_1 \perp\!\!\!\perp X_6 | \{X_2, X_3\}$?

Graph 2

- $X_2 \not\perp\!\!\!\perp X_3 | X_1, X_6$
- $X_2 \not\perp\!\!\!\perp X_4 | X_1, X_6$
- Ques: Is $X_2 \perp\!\!\!\perp X_3 | X_1$?

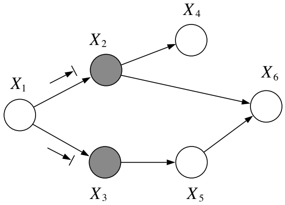


Figure 2.16: A ball cannot pass through X_2 to X_6 nor through X_3.

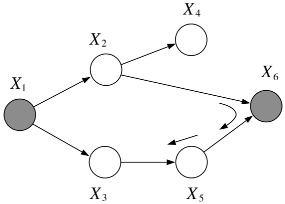
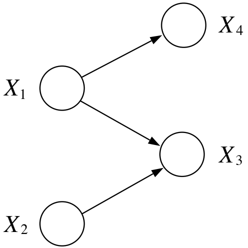


Figure 2.17: A ball can pass from X_2 through X_6 to X_5, and thence to X_3.

Example of using the Bayes Ball algorithm

List of all conditional independencies for this graph



$$X_1 \perp\!\!\!\perp X_2$$

$$X_2 \perp\!\!\!\perp X_4$$

$$X_2 \perp\!\!\!\perp X_4 \mid X_1$$

$$X_3 \perp\!\!\!\perp X_4 \mid X_1$$

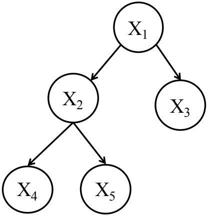
$$X_2 \perp\!\!\!\perp X_4 \mid \{X_1, X_3\}$$

$$\{X_2, X_3\} \perp\!\!\!\perp X_4 \mid X_1$$

Sum-Product Message-Passing to Find Marginals on Trees

Tree Graphical Models

- Each node (except for the root node) has exactly one parent. An N-node graph will have $N-1$ edges
- What is the joint distribution?



$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_2)$$

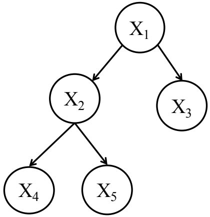
- What are the size of the conditional distribution tables at each node? $O(|\mathcal{X}|^2)$

Sum-Product Algorithm to Find the Marginal Distribution

- Suppose we want to evaluate the marginal distribution of a node, say $p(x_2)$ given the joint distribution $p(x_1, x_2, x_3, x_4, x_5)$
- The naive approach is to sum over all other variables:

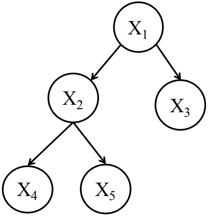
$$p(x_2) = \sum_{x_1, x_3, x_4, x_5 \in \mathcal{X}} p(x_1, x_2, x_3, x_4, x_5)$$

- How many operations does this take? $O(|\mathcal{X}|^5)$ – for each $x_2 \in \mathcal{X}$ sum over $|\mathcal{X}|^4$ possible values
- Complexity is exponential in the number of nodes
- Can we do better?



Sum-Product Algorithm to Find the Marginal Distribution

- Want to evaluate the marginal dist. $p(x_2)$
- Let us substitute the joint distribution of the tree to see if we can reduce the complexity

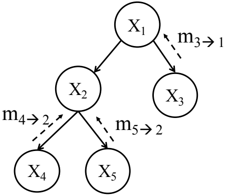


$$p(x_2) = \sum_{x_1, x_3, x_4, x_5 \in \mathcal{X}} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_2)$$

$$= \sum_{x_1 \in \mathcal{X}} p(x_1)p(x_2|x_1) \left(\sum_{x_3 \in \mathcal{X}} p(x_3|x_1) \right) \left(\sum_{x_4 \in \mathcal{X}} p(x_4|x_2) \right) \left(\sum_{x_5 \in \mathcal{X}} p(x_5|x_2) \right)$$

- Observe that the last three sums (corresponding to summing over the leaf nodes x_3, x_4, x_5) can be evaluated separately

Sum-Product Algorithm to Find the Marginal Distribution



- Want to evaluate the marginal dist. $p(x_2)$
- Simplifying the expression further

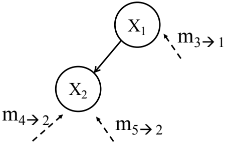
$$p(x_2) = \sum_{x_1 \in \mathcal{X}} p(x_1) p(x_2|x_1) \underbrace{\left(\sum_{x_3 \in \mathcal{X}} p(x_3|x_1) \right)}_{m_{3\to1}(x_1)} \underbrace{\left(\sum_{x_4 \in \mathcal{X}} p(x_4|x_2) \right)}_{m_{4\to2}(x_2)} \underbrace{\left(\sum_{x_5 \in \mathcal{X}} p(x_5|x_2) \right)}_{m_{5\to2}(x_2)}$$

$$= \sum_{x_1 \in \mathcal{X}} p(x_1) p(x_2|x_1) m_{3\to1}(x_1) m_{4\to2}(x_2) m_{5\to2}(x_2)$$

- Can think of $m_{5\to2}(x_2)$ as a "message" vector sent by node 5 to 2
- For each x_2, $m_{5\to2}(x_2)$ is the sum of $p(x_5|x_2)$ over all possible $x_5 \in \mathcal{X}$
- Similarly for $m_{3\to1}(x_1)$ and $m_{4\to2}(x_2)$

Sum-Product Algorithm to Find the Marginal Distribution

- Now we can remove nodes 3, 4, 5 because their information is already captured in their messages
- Now group the terms containing x_1



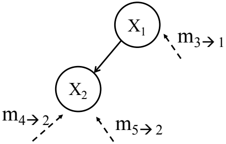
$$p(x_2) = m_{4\to 2}(x_2) m_{5\to 2}(x_2) \underbrace{\left(\sum_{x_1 \in \mathcal{X}} p(x_1) p(x_2|x_1) m_{3\to 1}(x_1) \right)}_{m_{1\to 2}(x_1)}$$

$$= m_{4\to 2}(x_2) m_{5\to 2}(x_2) m_{1\to 2}(x_1)$$

- The marginal distribution $p(x_2)$ is simply the product of incoming messages from all neighbors

Sum-Product Algorithm to Find the Marginal Distribution

- To find other marginals $p(x_3)$, $p(x_4)$, etc., we can reuse many of these messages
- In fact, if we pre-compute the two-way messages $m_{i\to j}$ and $m_{j\to i}$ for each edge (i,j) of the tree, then we can evaluate any marginal in terms of them



- Now group the terms containing x_1

$$p(x_2) = m_{4\to 2}(x_2)m_{5\to 2}(x_2)\underbrace{\left(\sum_{x_1\in\mathcal{X}} p(x_1)p(x_2|x_1)m_{3\to 1}(x_1)\right)}_{m_{1\to 2}(x_1)}$$

$$= m_{4\to 2}(x_2)m_{5\to 2}(x_2)m_{1\to 2}(x_1), \text{ the product of incoming messages}$$

- Why is it called sum-product? The message $m_{i\to j}$ is the product of incoming messages from all neighbors of i except j

Computational Complexity of the Sum-Product Algorithm

$$p(x_2) = \sum_{x_1 \in \mathcal{X}} p(x_1) p(x_2|x_1) \underbrace{\left(\sum_{x_3 \in \mathcal{X}} p(x_3|x_1) \right)}_{m_{3\to1}(x_1), O(|\mathcal{X}|)} \underbrace{\left(\sum_{x_4 \in \mathcal{X}} p(x_4|x_2) \right)}_{m_{4\to2}(x_2), O(|\mathcal{X}|)} \underbrace{\left(\sum_{x_5 \in \mathcal{X}} p(x_5|x_2) \right)}_{m_{5\to2}(x_2), O(|\mathcal{X}|)}$$

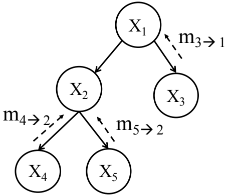
$$= \underbrace{\sum_{x_1 \in \mathcal{X}} p(x_1) p(x_2|x_1) m_{3\to1}(x_1) m_{4\to2}(x_2) m_{5\to2}(x_2))}_{O(|\mathcal{X}|)\text{operations}}$$

- For each x_2, we need $(n-1)O(|\mathcal{X}|)$ operations, where n is the number of nodes. Thus, just $nO(|\mathcal{X}|^2)$ in total
- Much smaller than $O(|\mathcal{X}|^n)$ with the brute-force approach, where n is the number of nodes

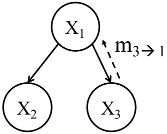
General Sum-Product Procedure of any n-node Tree

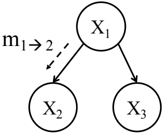
To find the marginal $p(x_i)$ of any node i

- Decide an elimination ordering of all other nodes, starting from the leaves, and moving towards node i
- To remove node i, compute its outgoing message by summing the joint distribution over x_i
- Continue removing nodes until only node i remains



Example: Sum-Product Algorithm





- X_1, X_2, X_3 are binary, $p(x_1) = [0.5, 0.5]$, and $p(x_3 = 0|x_1 = 0) = 0.3$, $p(x_3 = 0|x_1 = 1) = 0.6$, $p(x_2 = 0|x_1 = 0) = 0.6$, $p(x_2 = 0|x_1 = 1) = 0.2$.
- Find the marginal distribution $p(x_2)$

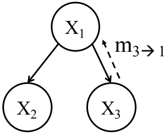
$$p(x_2) = \sum_{x_1, x_3 \in \mathcal{X}} p(x_1)p(x_2|x_1)p(x_3|x_1) \quad \text{Eliminate node 3}$$

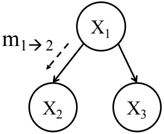
$$= \sum_{x_1 \in \mathcal{X}} p(x_1)p(x_2|x_1) \underbrace{\sum_{x_3 \in \mathcal{X}} p(x_3|x_1)}_{m_{3\to 1}(x_1)}$$

$$= \sum_{x_1 \in \mathcal{X}} p(x_1)p(x_2|x_1)m_{3\to 1}(x_1) \quad \text{Eliminate node 1}$$

$$= m_{1\to 2}(x_2)$$

Example: Sum-Product Algorithm





- X_1, X_2, X_3 are binary, $p(x_1) = [0.5, 0.5]$, and $p(x_3 = 0|x_1 = 0) = 0.3$, $p(x_3 = 0|x_1 = 1) = 0.6$, $p(x_2 = 0|x_1 = 0) = 0.6$, $p(x_2 = 0|x_1 = 1) = 0.2$.
- Find the marginal distribution $p(x_2)$

$$p(x_2) = \sum_{x_1 \in \mathcal{X}} p(x_2|x_1)p(x_1) \underbrace{\sum_{x_3 \in \mathcal{X}} p(x_3|x_1)}_{m_{3\to 1}(x_1)}$$

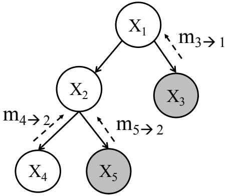
$$= \begin{bmatrix} 0.6 & 0.2 \\ 0.4 & 0.8 \end{bmatrix} \left(\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} p(x_2 = 0) \\ p(x_2 = 1) \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}$$

Sum-Product Message-Passing to Find Posteriors

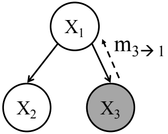
Sum-Product Algorithm to Find Posteriors

- Instead of finding marginals, suppose we want to find the posterior distribution $p(x_i | x_j = a, x_k = b)$ of a node i, given observed values of nodes x_j and x_k
- We can use the same sum-product algorithm. But instead of summing over all possible values of x_j and x_k, we just replace them by their observed values a and b



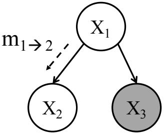
Example: Sum-Product Algorithm to Find Posteriors

- X_1, X_2, X_3 are binary, $p(x_1) = [0.5, 0.5]$, and $p(x_3 = 0|x_1 = 0) = 0.3$, $p(x_3 = 0|x_1 = 1) = 0.6$, $p(x_2 = 0|x_1 = 0) = 0.6$, $p(x_2 = 0|x_1 = 1) = 0.2$.
- Find the posterior distribution $p(x_2)$ given the observation $x_3 = 0$



$$p(x_2) \propto \sum_{x_1, x_3 \in \mathcal{X}} p(x_1)p(x_2|x_1)p(x_3|x_1) \quad \text{Eliminate node 3}$$

$$= \sum_{x_1 \in \mathcal{X}} p(x_1)p(x_2|x_1) \underbrace{p(x_3 = 0|x_1)}_{m_{3\to1}(x_1)}$$



$$= \sum_{x_1 \in \mathcal{X}} p(x_1)p(x_2|x_1)m_{3\to1}(x_1) \quad \text{Eliminate node 1}$$

$$= m_{1\to2}(x_2)$$

Example: Sum-Product Algorithm to Find Posteriors

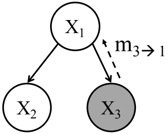
- X_1, X_2, X_3 are binary, $p(x_1) = [0.5, 0.5]$, and $p(x_3 = 0|x_1 = 0) = 0.3$, $p(x_3 = 0|x_1 = 1) = 0.6$, $p(x_2 = 0|x_1 = 0) = 0.6$, $p(x_2 = 0|x_1 = 1) = 0.2$.
- Find the posterior distribution $p(x_2)$ given the observation $x_3 = 0$

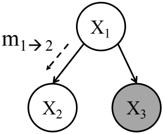
$$p(x_2) \propto \sum_{x_1 \in \mathcal{X}} p(x_2|x_1)p(x_1)\underbrace{p(x_3 = 0|x_1)}_{m_{3\to1}(x_1)}$$

$$= \begin{bmatrix} 0.6 & 0.2 \\ 0.4 & 0.8 \end{bmatrix} \left(\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \cdot \begin{bmatrix} 0.3 \\ 0.6 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.6 & 0.2 \\ 0.4 & 0.8 \end{bmatrix} \begin{bmatrix} 0.15 \\ 0.3 \end{bmatrix}$$

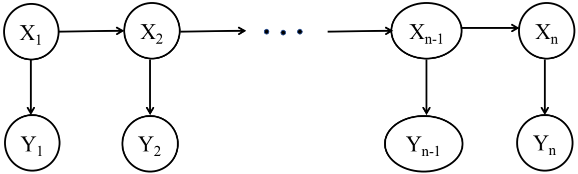
$$\begin{bmatrix} p(x_2 = 0) \\ p(x_2 = 1) \end{bmatrix} \propto \begin{bmatrix} 0.09 + 0.06 \\ 0.06 + 0.24 \end{bmatrix} \propto \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$$





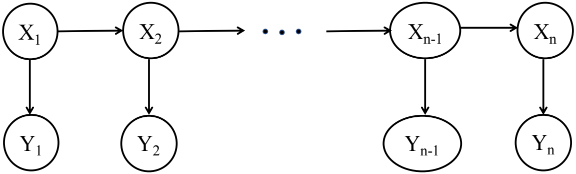
Forward-backward Algorithm for Hidden Markov Models

Hidden Markov Model (HMM)



- Eg. Suppose there are two forms of exercise a person A does: $Y =$ running (outdoor) or $Y =$ yoga (indoor) each day. Their choice is governed by the weather, which can be $X =$ rainy or sunny.
- Given that the weather X_i on the i-th day is rainy, A is more likely to do yoga ($P(Y_i = \text{ yoga}|X_i)$ is larger)
- Tomorrow's weather depends on today's weather

Joint Distribution of the HMM

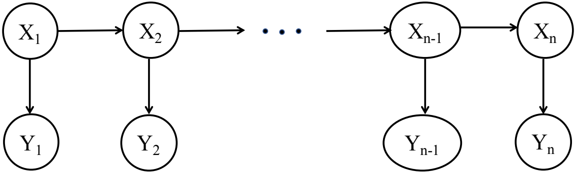


Joint distribution of the HMM is

$$p(x_1, \ldots, x_n, y_1, \ldots, y_n) = p(x_1) \prod_{j=1}^{n} p(y_j|x_j) \prod_{i=2}^{n} p(x_i|x_{i-1})$$

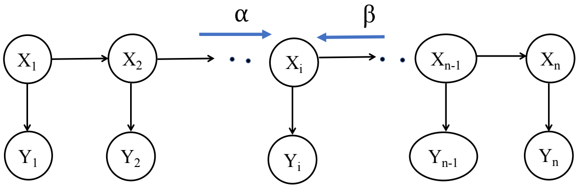
GOAL of the Forward-Backward Algorithm: Find the posterior distribution $p(x_i|y_1, y_2, \ldots y_n)$ of a state x_i given all the observations $y_1, y_2, \ldots y_n$

Simplifying the Posterior Expression



$$
\begin{aligned}
p(x_i|y_1,\dots,y_n) &= \frac{p(x_i,y_1,\dots,y_n)}{p(y_1,\dots,y_n)} \\
&= \frac{p(x_i,y_1,\dots y_i)p(y_{i+1},\dots y_n|x_i)}{p(y_1,\dots,y_n)} \\
&= \frac{p(x_i,y_1,\dots y_i)p(y_{i+1},\dots y_n|x_i)}{\sum_{x_i} p(x_i,y_1,\dots y_i)p(y_{i+1},\dots y_n|x_i)} \\
&= \frac{\alpha_i(x_i)\beta_i(x_i)}{\sum_{x_i}\alpha_i(x_i)\beta_i(x_i)}
\end{aligned}
$$

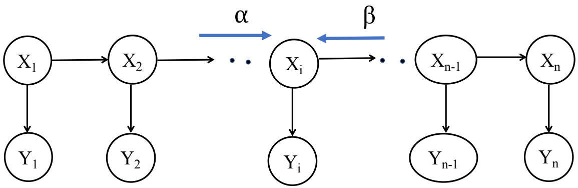
Simplifying the Posterior Expression



$$
\begin{aligned}
p(x_i|y_1,\ldots,y_n) &= \frac{p(x_i, y_1, \ldots y_i)p(y_{i+1}, \ldots y_n|x_i)}{\sum_{x_i} p(x_i, y_1, \ldots y_i)p(y_{i+1}, \ldots y_n|x_i)} \\
&= \frac{\alpha_i(x_i)\beta_i(x_i)}{\sum_{x_i} \alpha_i(x_i)\beta_i(x_i)}
\end{aligned}
$$

- Define $\alpha_i(x_i) \triangleq p(x_i, y_1, \ldots y_i)$, the forward messages
- Define $\beta_i(x_i) \triangleq p(y_{i+1}, \ldots y_n|x_i)$, the backward messages
- The Forward-Backward Algorithm is an efficient (recursive) method to compute $\alpha_i(x_i)$ and $\beta_i(x_i)$

Visualizing the Messages and the Posterior



- Define $\alpha_i(x_i) \triangleq p(x_i, y_1, \ldots y_i)$, the forward messages
- Define $\beta_i(x_i) \triangleq p(y_{i+1}, \ldots y_n | x_i)$, the backward messages
- The Forward-Backward Algorithm is an efficient (recursive) method to compute $\alpha_i(x_i)$ and $\beta_i(x_i)$

Recursively computing the forward messages $\alpha_i(x_i)$

The first message $\alpha_1(x_1) = p(x_1, y_1)$. Now let us compute $\alpha_i(x_i)$ in terms of $\alpha_{i-1}(x_{i-1})$ for all $i = 2, \ldots, n$:

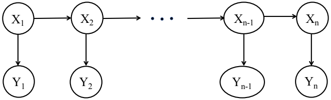
$$
\begin{aligned}
\alpha_i(x_i) &= p(x_i, y_1, \ldots y_i) \\
&= \sum_{x_{i-1}} p(x_{i-1}, x_i, y_1, \ldots y_i) \\
&= \sum_{x_{i-1}} p(x_{i-1} y_1, \ldots y_{i-1}) p(x_i, y_i | x_{i-1}, y_1, \ldots y_{i-1}) \\
&= \sum_{x_{i-1}} p(x_{i-1} y_1, \ldots y_{i-1}) p(x_i, y_i | x_{i-1}) \\
&= \sum_{x_{i-1}} p(x_{i-1}, y_1, \ldots y_{i-1}) p(x_i | x_{i-1}) p(y_i | x_i) \\
&= \sum_{x_{i-1}} \alpha_{i-1}(x_{i-1}) p(x_i | x_{i-1}) p(y_i | x_i)
\end{aligned}
$$

Recursively computing the backward messages $\beta_i(x_i)$

The last message $\beta_n(x_n) = 1$. Now let us compute $\beta_i(x_i)$ in terms of $\beta_{i+1}(x_{i+1})$ for all $i = 1, \ldots, n-1$:

$$
\begin{aligned}
\beta_i(x_i) &= p(y_{i+1}, \ldots, y_n | x_i) \\
&= \sum_{x_{i+1}} p(x_{i+1}, y_{i+1}, \ldots, y_n | x_i) \\
&= \sum_{x_{i+1}} p(y_{i+1}, \ldots y_n | x_i, x_{i+1}) p(x_{i+1} | x_i) \\
&= \sum_{x_{i+1}} p(y_{i+2}, \ldots y_n | x_{i+1}) p(x_{i+1} | x_i) p(y_{i+1} | x_{i+1}) \\
&= \sum_{x_{i+1}} \beta_{i+1}(x_{i+1}) p(x_{i+1} | x_i) p(y_{i+1} | x_{i+1})
\end{aligned}
$$

Putting it all together: Forward-Backward Algorithm



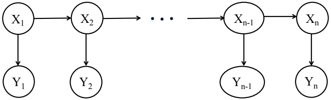
Given the following:

$$p(x_1)$$
$$p(x_i|x_{i-1}) \text{ for all } i = 2, \dots n$$
$$p(y_i|x_i) \text{ for all } i = 1, \dots n$$

the goal is to find

$$p(x_i|y_1, y_2, \dots y_n) \text{ for some } i$$

Putting it all together: Forward-Backward Algorithm



1. Find all the forward messages

$$\alpha_1(x_1) = p(x_1)p(y_1|x_1)$$
$$\alpha_i(x_i) = \sum_{x_{i-1}} \alpha_{i-1}(x_{i-1})p(x_i|x_{i-1})p(y_i|x_i) \text{ for } i = 2, \ldots, n$$

2. Find all the backward messages

$$\beta_n(x_n) = 1$$
$$\beta_i(x_i) = \sum_{x_{i+1}} \beta_{i+1}(x_{i+1})p(x_{i+1}|x_i)p(y_{i+1}|x_{i+1}) \text{ for } i = n-1, \ldots, 1$$

3. Output the posterior probability

$$p(x_i|y_1, y_2, \ldots y_n) = \frac{\alpha_i(x_i)\beta_i(x_i)}{\sum_{x_i} \alpha_i(x_i)\beta_i(x_i)}$$

Mini-summary of Graphical Models

You should know

- How to write the joint distribution corresponding to a given graphical model
- How to draw the graphical model corresponding to a joint dist.
- Checking conditional independence of variables using the Bayes Ball algorithm
- Sum-product algorithm on trees to find marginals and posteriors
- Forward-backward algorithm for hidden Markov models