Exact Inference: Elimination and Sum Product (and hidden Markov models)

David M. Blei
Columbia University

October 13, 2015

The first sections of these lecture notes follow the ideas in Chapters 3 and 4 of *An Introduction to Probabilistic Graphical Models* by Michael Jordan. In addition, many of the figures are taken these chapters.

The inference problem

Consider two sets of nodes E and F. We want to calculate the conditional distribution $p(x_F \mid x_E)$. This is the inference problem.

It amounts to three computations. First we marginalize out the set of variables x_R, where R contains the variables except x_E and x_F,

$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R). \tag{1}$$

From the result, we then marginalize x_F to obtain the marginal of x_E,

$$p(x_E) = \sum_{x_F} p(x_E, x_F). \tag{2}$$

Finally we take the ratio to compute the conditional distribution

$$p(x_F \mid x_E) = p(x_E, x_F)/p(x_E). \tag{3}$$

Our goal is to efficiently compute these quantities.

What is the problem? Suppose R contains many nodes, each taking on one of k values. Then marginalizing them out, in the first calculation, requires summing over $k^{|R|}$ configurations. This will usually be intractable.

Elimination

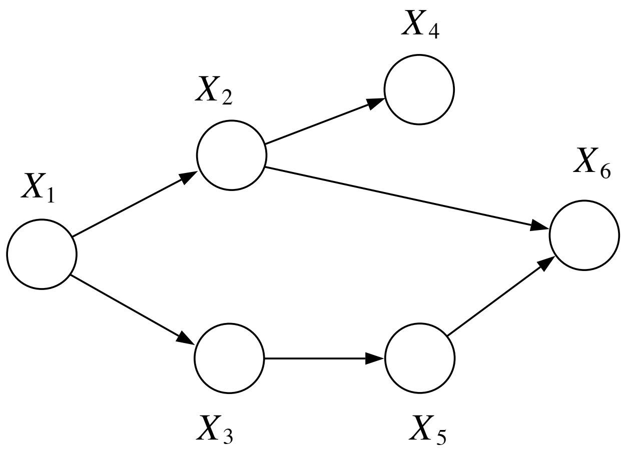
We first discuss *elimination*. It is a stepping stone to a more useful algorithm.

Before describing the algorithm, we introduce some notation. We will need to differentiate between two types of variables. Some we sum over or are arguments to a function; others are clamped at specific values, e.g., because they are part of the evidence.

So, $\bar{x}_6$ will refer to a specific value that the variable x_6 can take on. We will use a delta function, what is called an *evidence potential*, $\delta_{\bar{x}_6}(x_6)$ as a function whose value is equal to one if $x_6 = \bar{x}_6$ and zero otherwise. You will see later how this is useful.

An example

We demonstrate the elimination algorithm by an example. Consider the example graphical model from the last lecture,



Let us compute $p(x_1 \mid \bar{x}_6)$. While you are getting used to the notation, this is $P(X_1 = x_1 \mid X_6 = \bar{x}_6)$. To compute this, we take the ratio of $p(x_1, \bar{x}_6)$ and $p(\bar{x}_6)$.

We first compute the joint of x_1 and $\bar{x}_6$. To do this, we multiply the evidence potential at the end of the joint (to clamp x_6) and then summing out all the variables except x_1.

$$p(x_1, \bar{x}_6) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)\delta_{\bar{x}_6}(x_6) \tag{4}$$

Normally, one would sum out all except x_1 and x_6, but we use the summation to harness the evidence potential and select the right value of x_6 from the table.

The reason for this might seem mysterious, but it will come into focus as we continue to think about inference. Here, our goal is to have $p(\bar{x}_6 \mid x_5, x_2)$ in the expression. This is achieved with $\sum_{x_6} p(x_6 \mid x_5, x_2)\delta_{\bar{x}_6}(x_6)$. Thus we can treat all of the non-query node variables identically; we do not have to differentiate between evidence and non-evidence.

Thanks to the factorization, we can move sums inside the expression,

$$p(x_1, \bar{x}_6) = p(x_1) \sum_{x_2} p(x_2 \mid x_1) \sum_{x_3} p(x_3 \mid x_1) \sum_{x_4} p(x_4 \mid x_2) \sum_{x_5} p(x_5 \mid x_3) \sum_{x_6} p(x_6 \mid x_2, x_5) \delta_{\bar{x}_6}(x_6) \tag{5}$$

Let's make an intermediate factor involving the summation over x_6,

$$m_6(x_2, x_5) \triangleq \sum_{x_6} p(x_6 \mid x_2, x_5) \delta_{\bar{x}_6}(x_6) \tag{6}$$

This is a function of x_2 and x_5 because they are involved in the terms we summed over. We now rewrite the joint

$$p(x_1, \bar{x}_6) = p(x_1) \sum_{x_2} p(x_2 \mid x_1) \sum_{x_3} p(x_3 \mid x_1) \sum_{x_4} p(x_4 \mid x_2) \sum_{x_5} p(x_5 \mid x_3) m_6(x_2, x_5). \tag{7}$$

We have *eliminated* x_6 from the RHS calculation.

Let's do the same for the summation over x_5,

$$m_5(x_2, x_3) \triangleq \sum_{x_5} p(x_5 \mid x_3) m_6(x_2, x_5). \tag{8}$$

This is a function of x_2 and x_3. We rewrite the joint,

$$p(x_1, \bar{x}_6) = p(x_1) \sum_{x_2} p(x_2 \mid x_1) \sum_{x_3} p(x_3 \mid x_1) \sum_{x_4} p(x_4 \mid x_2) m_5(x_2, x_3). \tag{9}$$

Notice the intermediate factor $m_5(x_2, x_3)$ does not depend on x_4. We rewrite the joint

$$p(x_1, \bar{x}_6) = p(x_1) \sum_{x_2} p(x_2 \mid x_1) \sum_{x_3} p(x_3 \mid x_1) m_5(x_2, x_3) \sum_{x_4} p(x_4 \mid x_2). \tag{10}$$

We continue in this fashion, defining intermediate functions, moving them to the right place in the summation, and repeating. Note that the next function $m_4(x_2)$ actually equals one. But, to keep things programmatic, we will include it anyway.

$$\begin{aligned}
p(x_1, \bar{x}_6) &= p(x_1) \sum_{x_2} p(x_2 \mid x_1) \sum_{x_3} p(x_3 \mid x_1) m_5(x_2, x_3) m_4(x_2) && (11)\\
&= p(x_1) \sum_{x_2} p(x_2 \mid x_1) m_4(x_2) \sum_{x_3} p(x_3 \mid x_1) m_5(x_2, x_3) && (12)\\
&= p(x_1) \sum_{x_2} p(x_2 \mid x_1) m_4(x_2) m_3(x_1, x_2) && (13)\\
&= p(x_1) m_2(x_1) && (14)
\end{aligned}$$

We can further marginalize out x_1 to find the probability $p(\bar{x}_6)$,

$$p(\bar{x}_6) = \sum_{x_1} p(x_1, \bar{x}_6). \tag{15}$$

And finally we can take the ratio to find the conditional probability.

Discussion of the example

Question: What drives the computational complexity of the calculation we just made?

In each iteration we form a function by summing over a variable. That function is defined on some number of the other variables. The complexity has to do with forming that function. Functions of few variables are easier to form; functions of many variables are more difficult.

In our example, none of the intermediate functions had more than two arguments. If, for example, each variable is binary then we would never be manipulating a table of more than four items.

Consider this alternative, where we eliminate x_2 first

$$p(x_1, \bar{x}_6) = \sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\sum_{x_2} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)\delta_{\bar{x}_6}(x_6) \tag{16}$$

$$= \sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6} p(x_1)p(x_3 \mid x_1)p(x_5 \mid x_3)\delta_{\bar{x}_6}(x_6)\sum_{x_2} p(x_2 \mid x_1)p(x_4 \mid x_2)p(x_6 \mid x_2, x_5) \tag{17}$$

Here the intermediate function is

$$m_2(x_1, x_4, x_5, x_6) \triangleq \sum_{x_2} p(x_2 \mid x_1)p(x_4 \mid x_2)p(x_6 \mid x_2, x_5) \tag{18}$$

It is still a summation over x_2, but contains many more cells than the first intermediate function from the first example. This suggests that the complexity of the algorithm depends on the order in which we eliminate variables.

The elimination algorithm

We will now generalize this calculation, and describe the elimination algorithm. We are given a graphical model G, query node F, and evidence nodes E. The goal of the elimination algorithm is to compute $p(x_F \mid x_E)$.

Summary of the algorithm. It is an iterative algorithm where, in each iteration, we take the *sum* of a *product* of functions. These functions can be (a) conditional probability tables $p(x_i \mid x_{\pi_i})$ (b) delta functions on evidence $\delta_{\bar{x}_i}(x_i)$ or (c) intermediate functions $m_i(x_{S_i})$ that were created in previous iterations.

The algorithm maintains an *active list* of functions currently in play. At each iteration, it takes some functions off of this active list and uses them to compute a new intermediate function. At the end of the algorithm, the last function remaining can be used to compute the query of interest.

Details. Our goal is to compute $p(x_F \mid x_E)$.

1. Set an *elimination ordering* I, such that the query node F is last.
2. Set an *active list* of functions. Initialize with
   - Each conditional probability table $p(x_i \mid x_{\pi_i})$
   - Evidence potentials $\delta_{\bar{x}_i}(x_i)$ for each evidence node
3. Eliminate each node i in order:
   (a) Remove functions from the active list that involve x_i.
   (b) Set $\phi_i(T_i)$ equal to the product of these functions, where T_i is the set of all variables involved in them.

   Sum over x_i to compute the intermediate function

$$m_i(S_i) = \sum_{x_i} \phi_i(T_i). \tag{19}$$

   The arguments to the intermediate function are S_i (i.e., $T_i = S_i \cup i$).

   (c) Put $m_i(S_i)$ on the active list.
4. In the penultimate step, we have the unnormalized joint distribution of the query node and the evidence. Marginalizing out the query node gives us the marginal probability of the evidence.

This algorithm uses a graph to perform an inference. We defined it for one query node, but it generalizes to computing the conditional distribution of multiple nodes.

Undirected graphs

Recall the semantics of an undirected graph,

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi(x_C). \tag{20}$$

The elimination algorithm works on undirected graphs as well. Rather than placing conditional probability tables on the active list, we place the potential functions. See the book for a good example.

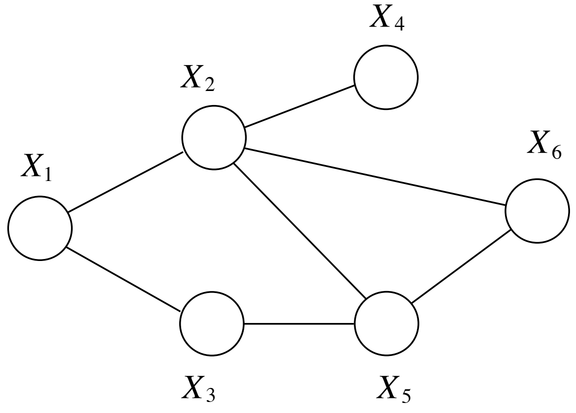
One nuance is the normalizing constant. Reusing the example above, we first compute $p(x_1, \bar{x}_6) = (1/Z)m_2(x_1)$. (This m_2 is different from above, calculated using the undirected elimination algorithm.) We than compute $p(\bar{x}_6) = \sum_{x_1}(1/Z)m_2(x_1)$. Taking the ratio gives the conditional of interest. Note we did not need to compute the normalizing constant Z because it cancels in the ratio. (Question: If we were interested in Z, how would we compute it?)

Graph eliminate

As we saw, the complexity of the elimination algorithm is controlled by the number of arguments in the intermediate factors. This, in turn, has to do with the number of parents of the nodes and the elimination ordering.

We can reason about the complexity of eliminate using only the graph. Initially, the graph represents the number of arguments in the active functions. As we eliminate nodes, we need to ensure that it represents the arguments of the intermediate functions too.

Let's do the directed case. (The undirected case is easier.) First create an undirected version of the directed graph where we fully connect the parents of each node. (This captures that arguments to $p(x_i \mid x_{\pi_i})$. In our example, here is the so-called "moralized" graph,



We can run an elimination algorithm graphically. Set the elimination ordering and consider the moralized undirected version of the graphical model. At each iteration, remove the next node and connect the nodes that were connected to it. Repeat.

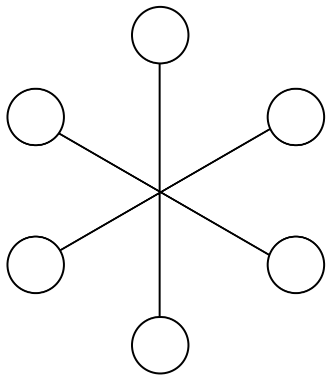
The *elimination cliques* are the collection of nodes that are the neighbors of x_i at the iteration when it is eliminated. Of course, these are the arguments to the intermediate functions when doing inference. If we record the elimination cliques for an ordering then we see that the complexity of the algorithm is driven by the largest one.

(Formally, the complexity of the elimination algorithm is exponential in the smallest achievable value, over orderings, of the largest elimination clique. However, finding the ordering is an NP-hard problem.)

Run graph eliminate with these examples, and record the elimination cliques:

- $\{6, 5, 4, 3, 2, 1\}$
- $\{2, 3, 4, 5, 6, 1\}$

Example: Consider this hub and spoke graph,



[We need a node in the center.] What happens when we remove the center node first; what happens when we remove the leaf nodes first?

Tree propagation

The elimination algorithm gives insight and generalizes to any (discrete) graphical model. But it is limited too, especially because it only computes a single query. We might have multiple inferences to calculate. Further, finding the elimination ordering is a hard problem and has large computational consequences.

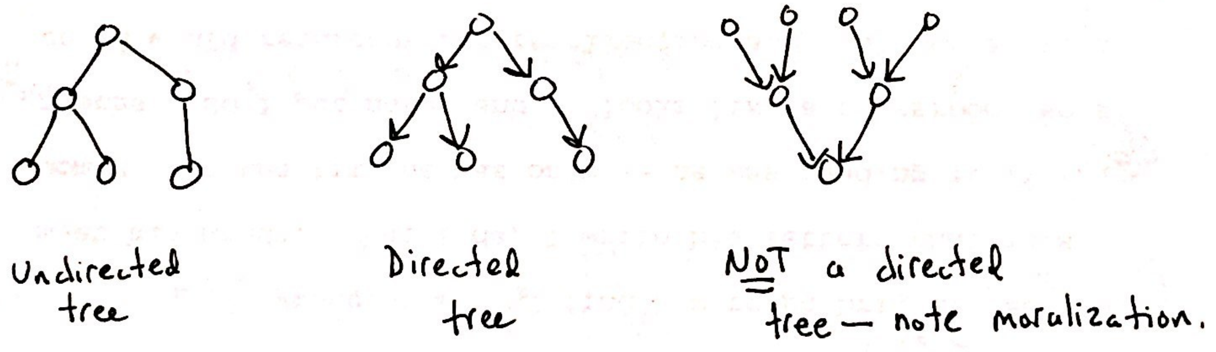
We will next discuss the *sum product* algorithm, an inference algorithm for trees.

1. Trees are important; many modern graphical models are trees.
2. This is the basis of the *junction tree* algorithm, a general exact inference method.
3. It relates to belief propagation, a method of approximate inference.

Tree graphical models

An undirected graphical model is a tree if there is only one path between any pair of nodes. A directed graphical model is a tree if its moralization is an undirected tree, i.e., there are no v-structures in the graph.

Here are some examples



Parameterization. Let's consider undirected trees. We will use a parameterization on singleton "cliques" and pairs (i.e., maximal cliques),

$$p(x) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi(x_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j). \tag{21}$$

This is parameterized by *singleton* and *pair-wise* potential functions.

Now consider directed trees. For trees, undirected trees and directed trees are the same class of graphical models. Define the root node to be x_r. The joint in a directed tree is

$$p(x) = p(x_r) \prod_{(i,j) \in \mathcal{E}} p(x_j \,|\, x_i). \tag{22}$$

We can set potentials equal to these conditional probabilities,

$$\psi(x_r) \triangleq p(x_r) \tag{23}$$

$$\psi(x_i) \triangleq 1 \quad \text{for } i \neq r \tag{24}$$

$$\psi(x_i, x_j) \triangleq p(x_j \,|\, x_i). \tag{25}$$

Thus the directed tree is a special case of the undirected tree. Any algorithm for undirected trees can be used for the directed tree.

We will only consider undirected trees.

Evidence. Just as we can consider undirected trees without loss of generality, we also will not need to pay special attention to evidence. Consider the evidence nodes x_E. Define

$$\psi^E(x_i) \triangleq \begin{array}{ll} \psi(x_i)\delta_{\bar{x}_i}(x_i) & i \in E \\ \psi(x_i) & i \notin E \end{array}. \tag{26}$$

Notice that

$$p(x \,|\, \bar{x}_E) = \frac{1}{Z^E} \prod_{i \in \mathcal{V}} \psi^E(x_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j) \tag{27}$$

This has the exact same form as the unconditional case.

We will not need to consider if the distribution is conditional or unconditional.

(Note: this trick only works with undirected models; the reason is that clamping the evidence only changes the normalizing constant. If we append a factorized joint with a delta function then the corresponding factorization does not normalize.)

Elimination on trees

As a precursor to the general algorithm, we derive the elimination algorithm for this restricted class of graphical models. First we set an *elimination ordering* of the nodes such that x_1 is the last element. (We will discuss the implications of this ordering later.)

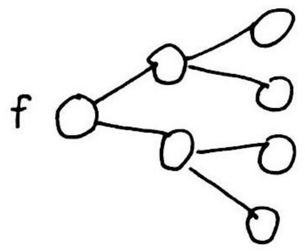
Recall the elimination algorithm:

```
1. Choose an ordering $I$ such that the query node is last.
2. Place all potentials on the active list.
3. Eliminate each node $i$
    (a) Take the product of active functions that reference $x_i$.
        Pop them off the active list.
    (b) Form the intermediate function $m_i(\cdot)$ by summing over $x_i$.
    (c) Put the intermediate function on the active list
```

Now we consider elimination on a tree. Set up the following ordering:

- Treat the query node f as the root.
- Direct all edges to point away from f. (Note this is *not* a directed GM.)
- Order each node by depth first; each node is eliminated only after its children.

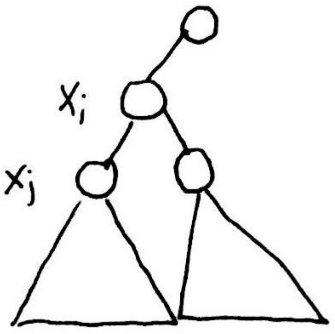
Let's look at a tree and consider the complexity of the elimination algorithm (by running the graph elimination algorithm).



Working backwards from the leaves, we see that the elimination cliques have a maximum size of two. Thus this is an efficient algorithm. (For contrast, consider removing a child of the root node first and note the elimination cliques; this is inefficient.)

Though we considered a specific query, we will see that elimination leads to a general algorithm for computing the (conditional) probability of any query node on a tree.

Let's look at the elimination step in more detail. Consider a pair of nodes (i, j), where i is closer to the root than j.



According to our algorithm, j will be eliminated first.

What is the intermediate function that we create when j is eliminated? To answer this, consider what we know about the potentials that will be on the active list when j is eliminated:

- $\psi(x_j)$
- $\psi(x_i, x_j)$
- no functions with x_k as an argument, a descendant of j
- no functions with x_ℓ as an argument, outside of the subtree of j

When we eliminate x_j, we add an intermediate function that is *only* a function of x_i. We call this function $m_{j \to i}(x_i)$ as a "message" from j to i.

What will this message be? It will involve

- the singleton potential $\psi(x_j)$
- the pair-wise potential $\psi(x_i, x_j)$
- and the other messages $m_{k \to j}(x_j)$ for other neighbors $k \in \mathcal{N}(j) \backslash i$

The message is

$$m_{j \to i}(x_i) \triangleq \sum_{x_j} \psi(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{k \to j}(x_j). \tag{28}$$

Once we compute messages up the tree, we compute the probability of the query f,

$$p(x_f \mid \bar{x}_E) \propto \psi(x_f) \prod_{k \in \mathcal{N}(f)} m_{k \to f}(x_f) \tag{29}$$

Note there is no pair-wise potential because f is the root

Equation (28) and Equation (29) are the elimination algorithm for an undirected tree. As the reading points out, inference involves solving a system of equations where the variables are the messages. The depth-first ordering of the messages ensures that each one is only a function of the messages already computed.

From elimination on trees to the sum-product algorithm

Let's do an example. We have a four-node tree. [Draw the tree: $X_1 - X_2, X_2 - \{X_3, X_4\}$.] Our goal is to compute $p(x_1)$. Note there is no evidence.

We compute the following messages, in this order:

$$\begin{aligned}
m_{3\to 2}(x_2) &= \sum_{x_3} \psi(x_3)\psi(x_2, x_3) && (30)\\
m_{4\to 2}(x_2) &= \sum_{x_4} \psi(x_4)\psi(x_2, x_4) && (31)\\
m_{2\to 1} &= \sum_{x_2} \psi(x_2)\psi(x_2, x_1) m_{3\to 2}(x_2) m_{4\to 2}(x_2) && (32)\\
p(x_1) &\propto \psi(x_1) m_{2\to 1}(x_1) && (33)
\end{aligned}$$

Now let's change our query and compute $p(x_2)$. This becomes the root of the tree and we compute messages $m_{1\to 2}$, $m_{3\to 2}$, $m_{4\to 2}$, and finally $p(x_2)$. *Notice that $m_{4\to 2}$ is the same as above.*

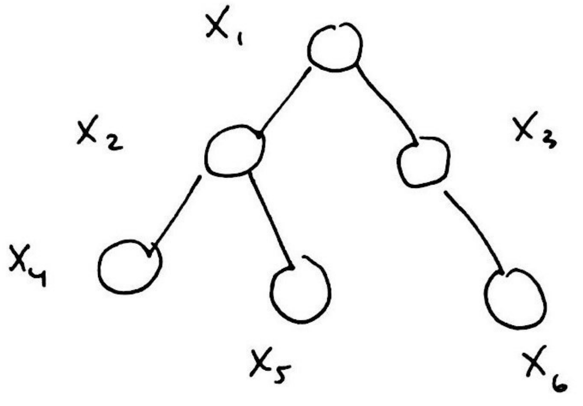
Let's change our query again and compute $p(x_4)$. It becomes the root and we compute messages $m_{1\to 2}$, $m_{3\to 2}$, $m_{2\to 4}$. Again, the messages are reused.

Key insight: *We are reusing computation from other queries.*

The sum-product algorithm is based on Equation (28) and Equation (29) and a message passing protocol. "A node can send a message to its neighbors when (and only when) it has received messages from all its other neighbors."(Jordan, 2003)

This lets us compute any marginal on the graph. It only requires computing 2 messages per node, and each is a function of just one argument.

Consider a six-node tree.



Sum-product propagates messages up and then down the tree. We compute:

- $m_{4\to 2}, m_{5\to 2}, m_{6\to 3}$
- $m_{2\to 1}, m_{3\to 1}$
- $m_{1\to 2}, m_{1\to 3}$
- $m_{2\to 4}, m_{2\to 5}, m_{3\to 6}$

Now we can easily compute any marginal.

The max-product algorithm

Let's consider a different problem, computing the configuration of the random variables that has highest probability. When we condition on evidence, this is called the *maximum a posteriori* problem. It arises in many applied settings.

First, let's look at the simpler problem of finding the maximal achievable probability, max $p(x)$. Like summation, maximization distributes over addition. Consider (briefly) our old graphical model,

$$\max_x p(x) = \tag{34}$$
$$\max_{x_1} p(x_1) \max_{x_2} p(x_2 \,|\, x_1) \max_{x_3} p(x_3 \,|\, x_1) \max_{x_4} p(x_4 \,|\, x_2) \max_{x_5} p(x_5 \,|\, x_3) \max_{x_6} p(x_6 \,|\, x_2, x_5). \tag{35}$$

Notice this is similar to computing a marginal. It turns out that all of the derivations that we've made with sums—the elimination algorithm, the sum-product algorithm—can be made with maximums.

Thus we can define the max elimination algorithm on trees just as we defined the marginal elimination algorithm. (Again, undirected trees subsume directed trees and trees with evidence.) First set a root node. Then define the messages in a depth-first ordering,

$$m^{\max}_{j\to i}(x_i) = \max_{x_j} \psi(x_j)\psi(x_i, x_j) \prod_{k\in\mathcal{N}(j)\backslash i} m^{\max}_{k\to j}(x_j). \tag{36}$$

The last message gives us the maximal probability,

$$\max p(x) = \max_{x_r} \psi(x_r) \prod_{k\in\mathcal{N}(r)} m^{\max}_{k\to r}(x_r) \tag{37}$$

Notice that we do not need to worry about reusing messages here. The maximal probability is the same regardless of the root node.

(Aside: When computing maximums of probabilities, underflow is a problem. The reason is that when we have many random variables we are multiplying many numbers smaller than one. It's always easier to work in log space. Happily, we can consider $\max \log p(x)$. We define the max-sum algorithm—max takes the place of sum, and sum takes the place of product—and use the log of the potential functions to compute messages.)

We have only computed the maximum value, not the configuration of variables that achieves it. Each message $m_{j\to i}(x_i)$ tells us a maximum over x_j, depending on the value of x_i. To compute the configuration that achieves the maximum, we must also store, for each value of x_i, which value of x_j achieves that maximum. We define another function,

$$\delta_{j\to i}(x_i) \triangleq \arg\max_{x_j} \psi(x_j)\psi(x_i, x_j) \prod_{k\in\mathcal{N}(j)\backslash i} m^{\max}_{k\to j}(x_j). \tag{38}$$

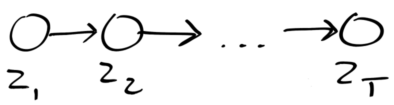
To obtain the maximizing configuration, first compute the messages and δ's going up the tree. Then consider the value of the root that achieves the maximum in Equation (37). Propagate that value as the argument to δ down the tree. This gives the maximizing configuration.

Example: Discrete hidden Markov model

We can now look at a real example, the discrete hidden Markov model. This is the first real model we discuss.

Markov model

A Markov model, as we've discussed, is a chain of random variables.



The parameters $p(z_{t+1} \mid z_t)$ give the probability of the next item given the previous item.

A famous example is a *language model*, which gives a distribution of English. The random variables are terms in a vocabulary. The Markov model specifies the probability of the next term given the previous term. This is called a bigram model. (When there is no connection it is called a unigram model; when there are connections beyond one step back it is called an n-gram model.) Note the parameterization: We must specify a $V \times V$ matrix of probabilities. For each term there is a distribution over the next term.

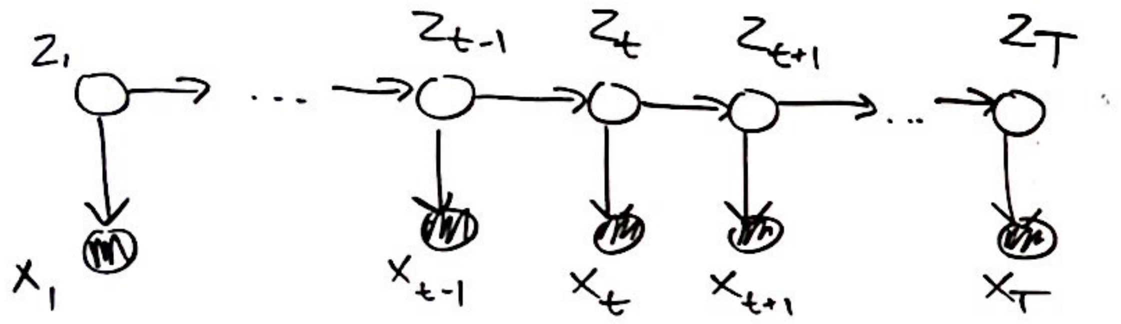
Aside: Claude Shannon's bigram. Claude Shannon invented information theory in a famous paper (Shannon, 1948). Incidentally, he also invented (and sampled from) the bigram model in that paper.

To start, he sampled z_0. He chose a random book from his shelf, opened to a random page, chose a random word, and wrote it down. To sample z_1, he chose another random book, opened to a random page, and read it until he found z_0. He then wrote down the *next* word in that book. Given z_t, he repeated this process for z_{t+1}.

Question: Why does this sample from a Markov model?

Hidden Markov model

The *hidden Markov model* is a widely used model that builds on the Markov models assumptions. The idea is that there is a Markov model of hidden random variables—variables that we cannot observe—each of which governs an observation. The hidden variables are called the hidden states.



In the HMM, the observations are a sequence of items. Recall that we discussed the conditional independence and marginal independence assumptions that we are making. What are they? Observations are conditionally independent given the hidden state; they are marginally dependent. You can see this with the Bayes ball algorithm.

We need to specify the probabilities that form the joint distribution. They are

- the transition probabilities $p(z_{t+1} \mid z_t)$
- the observation probabilities $p(x_t \mid z_t)$
- an initial state probability $p(z_1)$

These are the parameters to a hidden Markov model.

In our general discussion, we have been imagining that each random variable in a graphical model has its own probability table. Here, as is typical, the nodes share their probability tables across t. Thus $x_t \mid z_t$ is distributed in the same way as $x_{t+1} \mid z_{t+1}$. Note this is a conditional distribution—in a realization from this model they still might have different values, especially if they are associated with different values of the latent state.

Further, for this discussion we will take the probabilities as fixed and given. Later, when we talk about models and data, we will discuss how to handle the (typical) case where the parameters are unknown. In short, we usually find an estimate of the parameters that maximizes the likelihood of the data. But more on this later.

Sum-Product for an HMM

The inference problem is to compute the marginal of the hidden state or the maximizing sequence of hidden states given a set of observations. Have you seen this before? What are some applications that you know about?

HMMs are everywhere. Here are some examples (Murphy, 2013):

- **Speech recognition.** The hidden states are words, governed by a bigram distribution. The observations are features of the audio signal (let's suppose they are discrete), governed by an observation model.
- **Part of speech tagging.** The hidden states are parts of speech, governed by a distribution of the next POS given the previous one. Observations are words.

  Q: Suppose there are 10,000 vocabulary words and 5 parts of speech. How does this compare, in terms of the number of probabilities to specify, to the bigram model?
- **Gene finding.** The observations are a sequence of nucleotides (A,G,C,T) on a genome; the hidden states encode whether we are in a gene-encoding region. The observation probabilities and transition matrix come from biological knowledge and data.
- **Others.** Neuroscience (spike train sorting); handwriting recognition; others?

In all these cases we are interested in computing the marginal probability of a hidden variable (e.g., "What word did she say at time step 4?", "Is the nucleotide at position 600 part of a gene-coding region?") or the maximizing sequence ("What sentence did he most likely just utter?"). These are inference problems, just as we have been discussing.

The HMM is a tree. We turn it into an undirected tree with the following potentials:

$$\psi(z_{t-1}, z_t) \triangleq p(z_t \mid z_{t-1}) \tag{39}$$
$$\psi(z_t, x_t) \triangleq p(x_t \mid z_t) \tag{40}$$
$$\psi(z_1) \triangleq p(z_1) \tag{41}$$
$$\psi(z_t) \triangleq 1, \quad t > 1 \tag{42}$$
$$\psi(x_t) \triangleq \delta_{\bar{x}_t}(x_t) \tag{43}$$

Our goal is to compute the marginal of any hidden state.

Observation messages. There are a few kinds of messages. Let's start (arbitrarily) at the first time step,

$$m_{x_1 \to z_1}(z_1) = \sum_{x_1} \psi(x_1)\psi(x_1, z_1) \tag{44}$$
$$= \sum_{x_1} \delta_{\bar{x}_1}(x_1) p(x_1 \mid z_1). \tag{45}$$

In general,

$$m_{x_t \to z_t}(z_t) = \sum_{x_t} \psi(x_t)\psi(x_t, z_t) \tag{46}$$
$$= \sum_{x_t} \delta_{\bar{x}_t}(x_t) p(x_t \mid z_t). \tag{47}$$

Each of these messages simply selects the appropriate observation probability,

$$m_{x_t \to z_t}(z_t) = p(\bar{x}_t \mid z_t) \tag{48}$$

Forward messages. Now we compute messages between hidden states. Beginning with the first time step,

$$m_{z_1 \to z_2}(z_2) = \sum_{z_1} \psi(z_1)\psi(z_1, z_2) m_{x_1 \to z_1}(z_1) \tag{49}$$
$$= \sum_{z_1} p(z_1) p(z_2 \mid z_1) p(\bar{x}_1 \mid z_1). \tag{50}$$

Recall the message passing protocol. Computing this message lets the algorithm move on to the next time step. In general, the message computed from $t-2$ to $t-1$ allows the algorithm to go *forward* to the message for $t-1$ to t,

$$m_{z_{t-1} \to z_t}(z_t) = \sum_{z_{t-1}} \psi(z_{t-1})\psi(z_{t-1}, z_t) m_{x_{t-1} \to z_{t-1}}(z_{t-1}) m_{z_{t-2} \to z_{t-1}}(z_{t-1}) \tag{51}$$
$$= \sum_{z_{t-1}} p(z_t \mid z_{t-1}) p(\bar{x}_{t-1} \mid z_{t-1}) m_{z_{t-2} \to z_{t-1}}(z_{t-1}). \tag{52}$$

Let's pause to consider the probabilistic interpretation of these messages. The message in Equation (49) is

$$m_{z_1 \to z_2}(z_2) = \sum_{z_1} p(z_1) p(z_2 \mid z_1) p(\bar{x}_1 \mid z_1) \tag{53}$$

$$= \sum_{z_1} p(z_1, z_2, \bar{x}_1) \tag{54}$$

$$= p(\bar{x}_1, z_2) \tag{55}$$

We assert that the forward message is

$$m_{z_{t-1} \to z_t}(z_t) = p(\bar{x}_1, \ldots, \bar{x}_{t-1}, z_t). \tag{56}$$

This is the joint probability of the of the observations up to time $t - 1$ and the hidden state at time t.

We can see this by induction,

$$m_{z_t \to z_{t+1}}(z_{t+1}) = \sum_{z_t} p(z_{t+1} \mid z_t) p(\bar{x}_t \mid z_t) p(\bar{x}_1, \ldots, \bar{x}_{t-1}, z_t) \tag{57}$$

$$= p(\bar{x}_1, \ldots, \bar{x}_t, z_{t+1}). \tag{58}$$

Backward messages. To run sum-product, we also need the messages that go backward from $t + 1$ to t. We go backward with this message,

$$m_{z_{t+1} \to z_t}(z_t) = \sum_{z_{t+1}} \psi(z_{t+1}) \psi(z_t, z_{t+1}) m_{x_{t+1} \to z_{t+1}}(z_{t+1}) m_{z_{t+2} \to z_{t+1}}(z_{t+1}) \tag{59}$$

$$= \sum_{z_{t+1}} p(z_{t+1} \mid z_t) p(\bar{x}_{t+1} \mid z_{t+1}) m_{z_{t+2} \to z_{t+1}}(z_{t+1}). \tag{60}$$

For completion, the boundary case is

$$m_{z_T \to z_{T-1}}(z_{T-1}) = \sum_{z_T} \psi(z_T) \psi(z_{T-1}, z_T) m_{x_T \to z_T}(z_T) \tag{61}$$

$$= \sum_{z_T} p(z_T \mid z_{T-1}) p(\bar{x}_T \mid z_T). \tag{62}$$

This message also has a probabilistic interpretation,

$$m_{z_T \to z_{T-1}}(z_{T-1}) = p(\bar{x}_T \mid z_{T-1}). \tag{63}$$

Again by induction, we can show that

$$m_{z_{t+1} \to z_t}(z_t) = p(\bar{x}_T, \ldots, \bar{x}_{t+1} \mid z_t). \tag{64}$$

Forward-Backward algorithm. These two types of messages can compute any marginal,

$$p(z_t) \propto m_{z_{t-1} \to z_t}(z_t) m_{z_{t+1} \to z_t}(z_t) m_{x_t \to z_t}(z_t). \tag{65}$$

This is called the forward-backward algorithm or the alpha-beta algorithm, and was derived independently for the hidden Markov model. Here we see that it's an instance of sum-product, a special case of a more general algorithm.

Further, especially in speech recognition, we are often interested in the maximizing sequence of hidden states. (E.g., in speech recognition these are the words that maximize the probability of the full utterance.) The max-product algorithm (or max-sum on the log probabilities) gives us this. In the HMM literature, it is known as the Viterbi algorithm. Again, we see that it is a special case of a more general algorithm.

References

Jordan, M. (2003). *An Introduction to Probabilistic Graphical Models*.

Murphy, K. (2013). *Machine Learning: A Probabilistic Approach*. MIT Press.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.