Reading for Lecture 10

Release v10

Christopher Lee

October 19, 2011

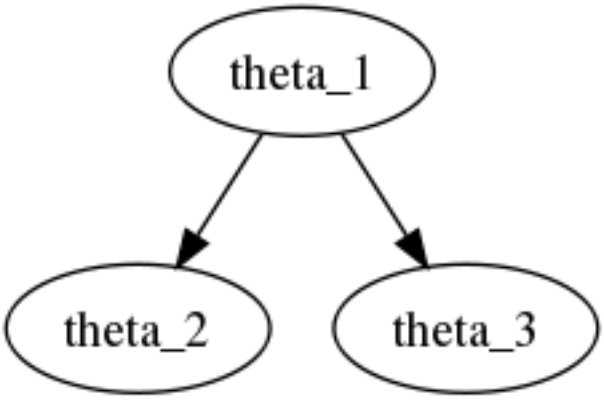
Contents

1 Generalized Markov Models

1.1 Branching Structures

So far we have discussed *Markov chains*, a linear sequence of variables that each depend only on the previous variable. This can be represented by an information graph in which each node (variable) has a single outgoing edge to the next node. However, we can generalize most of the methods we've developed to more complicated information graph structures.

- *conditionally independent branches*: say θ_2 and θ_3 both depend on θ_1 but not on each other. This can be represented by drawing a single edge from $\theta_1 \rightarrow \theta_2$ and another edge from $\theta_1 \rightarrow \theta_3$.

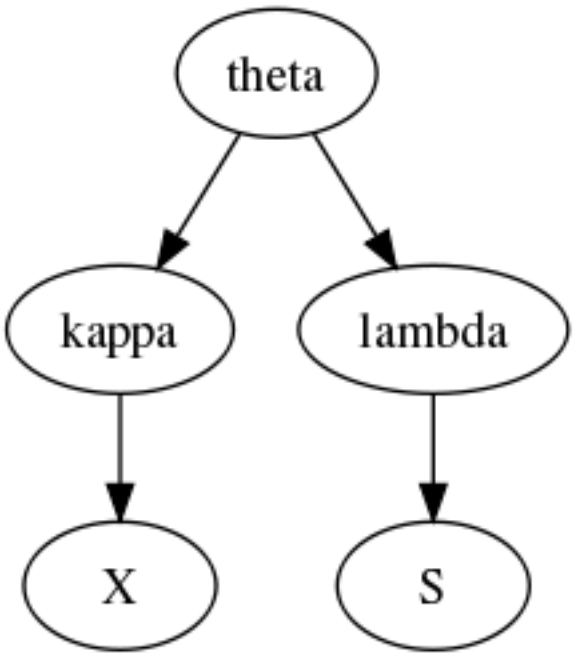


Note that both these edges obey a standard first-order Markov property. On the other hand, if we tried to compute the forward probability in the usual way, we'd run into a problem: the forward probabilities θ_2 and θ_3 both contain contributions from θ_1 (typically, from its associated observation X_1). We cannot just combine the two forward probabilities as if the branches were independent, first of all because that would "double-count" the contribution from θ_1, and second of all because they are *not* independent, only *conditionally* independent given θ_1. Below we will investigate how to combine them properly.

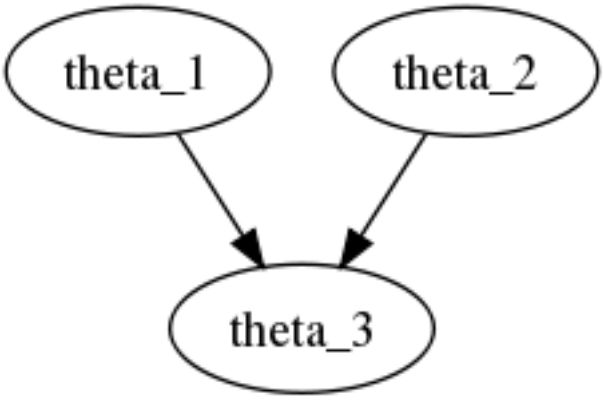
We will refer to this kind of structure as "multiple dependents", i.e. one node (variable) has multiple outgoing edges.

Example: Forensic Test Mismatch Model

The SNP frequency variable θ affects both the crime scene variable κ and the suspect variable λ.



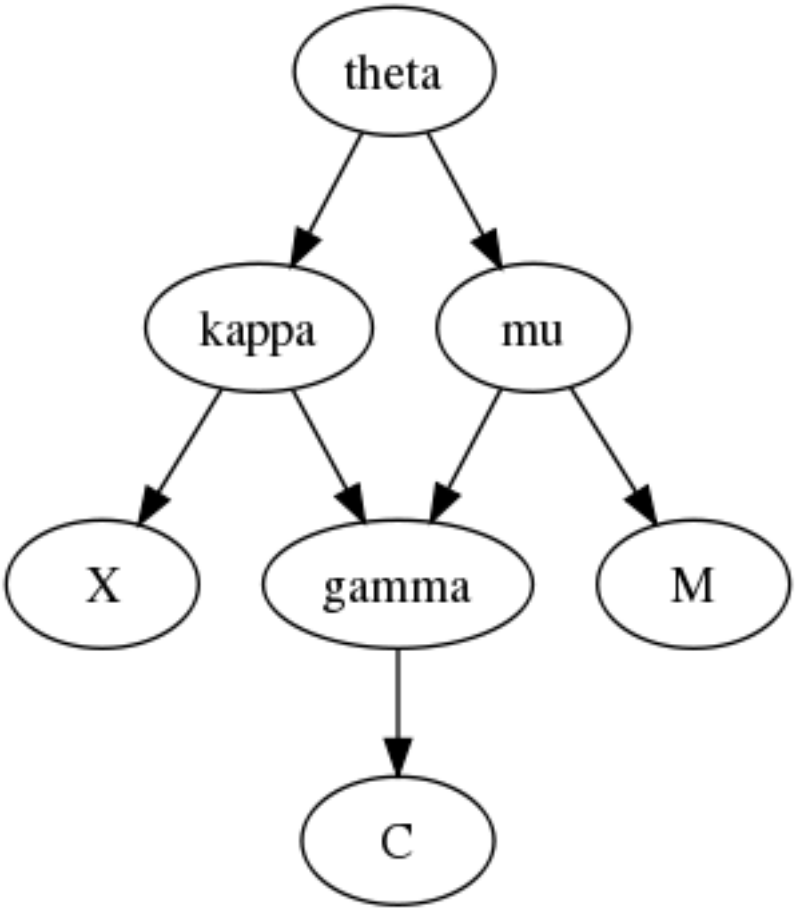
- *multiple, independent conditions*: say θ_3 depends on both θ_1 and θ_2, but neither of them depend on each other.



Again, the standard forward probability calculation will not handle this properly, because θ_3 does not obey the standard first-order Markov property.

Example: Paternity Test DAD Model

The variable γ representing the child depends on both the variables κ, μ representing the father and mother. If θ is considered a constant, then κ, μ have no other connection (apart from the fact they both affect γ).



- *branch closure*: If θ is considered a variable, the situation changes, because now we have one node θ with multiple dependents which in turn re-connect at another node γ with multiple conditions. In the information graph this is represented by a pair of nodes where there is *more than one path* (traversing one or more variables) from the first node to the second. This creates extra complications in probability calculations, because everything on

the branches between the "origin" and "destination" must be treated as coupled to *both* the origin and destination variables. We will investigate how to handle this case properly below.

As should be evident from these simple examples, these kinds of branching structures occur commonly in real-world problems, so we need to understand how to handle them correctly.

1.2 Higher Order Markov Chains

One special case merits discussion: a Markov chain where each variable in the sequence depends on the previous *two* (or more) variables. A Markov chain where each variable depends on the previous two variables is called "second-order"; if it depends on the previous three variable that is "third order", etc. Note first that this does not fit our definition of *branch closure*: although there are multiple paths from one node to another, one of those paths is just a single edge, and thus does not traverse any other variable, contrary to our definition of coupled branches.

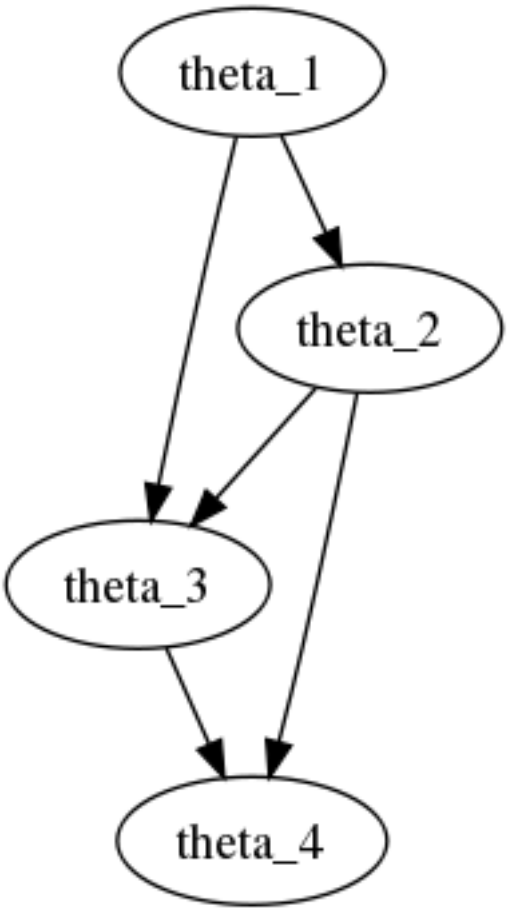


Figure 1: **A second-order Markov chain**

Higher order Markov chains can be handled by simple modifications of the standard forward-backward computations; we will examine this in detail below.

2 The Posterior Cut: Using a Subject Variable to Divide the Information Graph

2.1 The Posterior Cut Method

There is a simple way to find an efficient algorithm for computing the posterior probability across a branched information graph.

- We consider a variable (or group of variables) whose posterior probability we wish to calculate, e.g. θ_t. We call this the *subject variable(s)*.
- We wish to find "a cut" of the information graph that uses our subject variable(s) to divide it into two or more pieces connected only by our subject variable(s).
- We draw a prospective cut on the information graph, by simply drawing a line that separates all the variables in the graph into two disjoint subsets that we will compute separately.
- In the simple, linear case these "partitions" simply correspond to the forward vs. backward probability components.
- For branched information graphs, the partitions may be more complicated.
- We label the two partitions created by this cut as follows:
  - we choose a symbol for each partition. For example for "forward" and backward, we can call them "f" and "b".
  - we add index subscripts that uniquely identify this specific cut: a *subject index* that specifies the subject variable (in this example, that's just t); and one or more *state indices* to specify the specific state that it's computed for (in this example, f_t only needs to track the value of θ_t, so we can just use the index i specifying state s_i of θ_t).
  - The number of state indices needed is dictated by the number of conditions the next variable depends on. Remember that we are computing f_t etc. not only to compute the posterior probability of θ_t but also to calculate the next such partition (i.e. f_{t+1}). If θ_{t+1} depends on *two* preceding variables (i.e. a second-order Markov chain), we would accordingly need *two* state indices representing θ_t, θ_{t-1}, i.e. f_{tij}.

  In this example, the forward partition would be labeled f_{ti}, and the backwards partition b_{ti}.
- We now wish to find a recursion for each of these partitions that defines it in terms of one or more sub-partitions of itself. Concretely, in our example, we would like to a way to compute f_{ti} from its sub-partition $f_{t-1,j}$, and b_{ti} from its sub-partition $b_{t+1,k}$.
- We therefore draw the equivalent cuts for representing $f_{t-1,j}$ and $b_{t+1,k}$, and label the regions corresponding to $f_{t-1,j}$ and $b_{t+1,k}$.
- Note that we assign them unique index labels j, k so that we can identify which variable we are referring to by simply specifying its associated index. This makes writing the formulas much simpler. To make this completely explicit, we write the index labels next to their associated nodes on the information graph.
- Now we focus on a specific partition, say f_{ti}. We want to identify a list of probability components that will give us f_{ti} from $f_{t-1,j}$. All we have to do is mark the probability components that are contained in f_{ti} but not in $f_{t-1,j}$. These are simply the conditional probabilities (edges) that are contained in the f_{ti} partition but not in the smaller $f_{t-1,j}$ partition. We can do so by simply circling those components on the info graph:
  - We start by circling our first ingredient, $f_{t-1,j}$.
  - We then circle the transition from $\theta_{t-1} \to \theta_t$. That gives us $p(\theta_t|\theta_{t-1})$.
  - We then circle the transition from $\theta_t \to X_t$. That gives us $p(X_t|\theta_t)$.

  - That captures everything needed to compute f_{ti}, so we're done!
- We now write the recursion formula for f_{ti} as follows:
  - We simply write a product of the list of terms we circled on the info graph.
  - For simplicity, we can write this formula using just the unique indices for each variable, i.e. i, j, k representing $\theta_t, \theta_{t-1}, \theta_{t+1}$ respectively.
  - We specify explicit summation over whatever state index(es) must be eliminated. In this example, we wish to compute f_{ti}, which contains index i only. However, our list of terms includes both indexes i and j. So we must sum over all values of j.

  In this case, our formula would be:

$$f_{ti} = \sum_j f_{t-1,j} p(i|j) p(X_t|i)$$

  which is shorthand for

$$f_{ti} = \sum_j f_{t-1,j} p(\theta_t = s_i|\theta_{t-1} = s_j) p(X_t|\theta_t = s_i)$$

- We do the same thing for the other partition, in this case b_{ti}.

2.2 Guidelines for How To Think About Challenging Cases

If you find that there seems to be no valid way to compute what you think want from what you think is its appropriate predecessor, that tells you that your assumptions are not adequate for the problem. You will have to rethink one or more of the following:

- the number of indexes required for one of your partitions. For example, in the first-order Markov chain case described above, f_{ti} was chosen to depend on only a single variable (indexed by i). This is adequate, because for a first order Markov chain we can compute the conditional probability of the next variable knowing the state of just a single variable (i.e. all we need to know is i, not j as well). However, for a higher order Markov chain that clearly is not adequate: e.g. for a second-order Markov chain we would have to index on two variables, i.e. f_{tij}.
- the choice of "predecessor": again, for higher-order cases we will probably have to look "further back" (rather than just the previous variable θ_{t-1}) according to how many previous variables the Markov property is conditioned on. In other words, the appropriate choice of predecessor would not be f_{t-1} but f_{t-2} or earlier.
- In general there is little ambiguity about how to choose a cut: it is dictated by the choice of subject variable(s). This creates two disjoint partitions coupled only by the subject variable(s).
- However, for a given partition, it may not be possible to define a recursion that forms it in terms of a single "predecessor" sub-partition. Specifically, in branched structures (e.g. see the first example below), we may have to combine two or more sub-partitions to cover our desired partition with *no duplications*. The key principle is that we must combine strictly disjoint partitions whose union constitutes the partition we are seeking to obtain. Every variable must be counted exactly once!

3 Deriving Forward Backward Rules from Info Graph Factoring

The best way to understand these guidelines is to see them applied to specific examples.

3.1 Example: Conditionally Independent Branches

Say we are interested in computing the posterior probability of κ.

- We therefore cut at κ to obtain $f_{\kappa,k}$ and $b_{\kappa,k}$ partitions.
- Note that the $f_{\kappa,k}$ partition cannot be defined solely in terms of its predecessor $f_{\theta,i}$ – that would leave out the entire λ branch! Since the $f_{\kappa,k}$ partition must cover everything not contained in the $b_{\kappa,k}$ partition, it needs to include the λ branch.
- Looking at the diagram, this looks straightforward. We can combine the $f_{\theta,i}$ and $b_{\lambda,l}$ partitions to construct the $f_{\kappa,k}$ partition.
- The list of "ingredients" for $f_{\kappa,k}$ is just: $f_{\theta,i}$, $p(k|i)$, $b_{\lambda,l}$ and $p(l|i)$.
- We construct the final recursion formula by summing to eliminate the unwanted state indices:

$$f_{\kappa,k} = \sum_{i,l} f_{\theta,i} p(k|i) p(l|i) b_{\lambda,l}$$

- The same logic applies to computing $f_{\lambda,l}$.
- The only additional (slight) complication is finding a recursion for $b_{\theta,i}$. It can easily be assembled by combining $b_{\kappa,k}$ and $b_{\lambda,l}$, and again $p(k|i)p(l|i)$. The final recursion formula is then

$$b_{\theta,i} = \left(\sum_{k} p(k|i) b_{\kappa,k} \right) \left(\sum_{l} p(l|i) b_{\lambda,l} \right)$$

3.2 Example: Independent Conditions

Again, we take κ as our subject variable. The only difference vs. the previous example is that the directions are turned upside down.

- We cut at κ to obtain $f_{\kappa,k}$ and $b_{\kappa,k}$ partitions.
- Note that the $b_{\kappa,k}$ partition cannot be defined solely in terms of its successor $b_{\gamma,g}$ – that would leave out the entire μ branch! Since the $b_{\kappa,k}$ partition must cover everything not contained in the $f_{\kappa,k}$ partition, it needs to include the μ branch.
- As before, we construct $b_{\kappa,k}$ by combining two partitions, $b_{\gamma,g}$ and $f_{\mu,m}$.
- The list of "ingredients" for $b_{\kappa,k}$ is just: $b_{\gamma,g}$, $f_{\mu,m}$ and $p(g|k,m)$.
- We construct the final recursion formula by summing to eliminate the unwanted state indices:

$$b_{\kappa,k} = \sum_{m,g} b_{\gamma,g} f_{\mu,m} p(g|k,m)$$

- The same logic applies to computing $b_{\mu,m}$.
- Finally, we note that $f_{\gamma,g}$ of course must take both branches into account:

$$f_{\gamma,g} = \sum_{k,m} f_{\kappa,k} f_{\mu,m} p(g|k,m)$$