CS 188   Introduction to Artificial Intelligence

Fall 2023   Note 18

Author (all other notes): Nikhil Sharma

Author (Bayes' Nets notes): Josh Hug and Jacky Liang, edited by Regina Wang

Author (Logic notes): Henry Zhu, edited by Peyrin Kao

Markov Models

In previous notes, we talked about Bayes' nets and how they are a wonderful structure used for compactly representing relationships between random variables. We'll now cover a very intrinsically related structure called a **Markov model**, which for the purposes of this course can be thought of as analogous to a chain-like, infinite-length Bayes' net. The running example we'll be working with in this section is the day-to-day fluctuations in weather patterns. Our weather model will be **time-dependent** (as are Markov models in general), meaning we'll have a separate random variable for the weather on each day. If we define W_i as the random variable representing the weather on day i, the Markov model for our weather example would look like this:
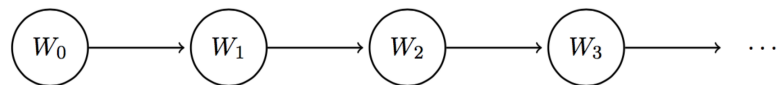
What information should we store about the random variables involved in our Markov model? To track how our quantity under consideration (in this case, the weather) changes over time, we need to know both it's **initial distribution** at time $t = 0$ and some sort of **transition model** that characterizes the probability of moving from one state to another between timesteps. The initial distribution of a Markov model is enumerated by the probability table given by $P(W_0)$ and the transition model of transitioning from state i to $i+1$ is given by $P(W_{i+1}|W_i)$. Note that this transition model implies that the value of W_{i+1} is conditionally dependent only on the value of W_i. In other words, the weather at time $t = i+1$ satisfies the **Markov property** or **memoryless property**, and is independent of the weather at all other timesteps besides $t = i$.

Using our Markov model for weather, if we wanted to reconstruct the joint between W_0, W_1, and W_2 using the chain rule, we would want:

$$P(W_0, W_1, W_2) = P(W_0)P(W_1|W_0)P(W_2|W_1, W_0)$$

However, with our assumption that the Markov property holds true and $W_0 \perp\!\!\!\perp W_2|W_1$, the joint simplifies to:

$$P(W_0, W_1, W_2) = P(W_0)P(W_1|W_0)P(W_2|W_1)$$

And we have everything we need to calculate this from the Markov model. More generally, Markov models make the following independence assumption at each timestep: $W_{i+1} \perp\!\!\!\perp \{W_0, ..., W_{i-1}\}|W_i$. This allows us

to reconstruct the joint distribution for the first $n+1$ variables via the chain rule as follows:

$$P(W_0, W_1, ..., W_n) = P(W_0)P(W_1|W_0)P(W_2|W_1)...P(W_n|W_{n-1}) = P(W_0)\prod_{i=0}^{n-1}P(W_{i+1}|W_i)$$

A final assumption that's typically made in Markov models is that the transition model is **stationary**. In other words, for all values of i (all timesteps), $P(W_{i+1}|W_i)$ is identical. This allows us to represent a Markov model with only two tables: one for $P(W_0)$ and one for $P(W_{i+1}|W_i)$.

The Mini-Forward Algorithm

We now know how to compute the joint distribution across timesteps of a Markov model. However, this doesn't explicitly help us answer the question of the distribution of the weather on some given day t. Naturally, we can compute the joint then **marginalize** (sum out) over all other variables, but this is typically extremely inefficient, since if we have j variables each of which can take on d values, the size of the joint distribution is $O(d^j)$. Instead, we'll present a more efficient technique called the **mini-forward algorithm**.

Here's how it works. By properties of marginalization, we know that

$$P(W_{i+1}) = \sum_{w_i} P(w_i, W_{i+1})$$

By the chain rule we can re-express this as follows:

$$\boxed{P(W_{i+1}) = \sum_{w_i} P(W_{i+1}|w_i)P(w_i)}$$

This equation should make some intuitive sense — to compute the distribution of the weather at timestep $i+1$, we look at the probability distribution at timestep i given by $P(W_i)$ and "advance" this model a timestep with our transition model $P(W_{i+1}|W_i)$. With this equation, we can iteratively compute the distribution of the weather at any timestep of our choice by starting with our initial distribution $P(W_0)$ and using it to compute $P(W_1)$, then in turn using $P(W_1)$ to compute $P(W_2)$, and so on. Let's walk through an example, using the following initial distribution and transition model:

W_0	$P(W_0)$
sun	0.8
rain	0.2

W_{i+1}	W_i	$P(W_{i+1}|W_i)$
sun	sun	0.6
rain	sun	0.4
sun	rain	0.1
rain	rain	0.9

Using the mini-forward algorithm we can compute $P(W_1)$ as follows:

$$
\begin{aligned}
P(W_1 = sun) &= \sum_{w_0} P(W_1 = sun|w_0)P(w_0) \\
&= P(W_1 = sun|W_0 = sun)P(W_0 = sun) + P(W_1 = sun|W_0 = rain)P(W_0 = rain) \\
&= 0.6 \cdot 0.8 + 0.1 \cdot 0.2 = \boxed{0.5} \\
P(W_1 = rain) &= P(W_1 = rain|w_0)P(w_0) \\
&= P(W_1 = rain|W_0 = sun)P(W_0 = sun) + P(W_1 = rain|W_0 = rain)P(W_0 = rain) \\
&= 0.4 \cdot 0.8 + 0.9 \cdot 0.2 = \boxed{0.5}
\end{aligned}
$$

Hence our distribution for $P(W_1)$ is

W_1	$P(W_1)$
sun	0.5
rain	0.5

Notably, the probability that it will be sunny has decreased from 80% at time $t = 0$ to only 50% at time $t = 1$. This is a direct result of our transition model, which favors transitioning to rainy days over sunny days. This gives rise to a natural follow-up question: does the probability of being in a state at a given timestep ever converge? We'll address the answer to this problem in the following section.

Stationary Distribution

To solve the problem stated above, we must compute the **stationary distribution** of the weather. As the name suggests, the stationary distribution is one that remains the same after the passage of time, i.e.

$$P(W_{t+1}) = P(W_t)$$

We can compute these converged probabilities of being in a given state by combining the above equivalence with the same equation used by the mini-forward algorithm:

$$P(W_{t+1}) = P(W_t) = \sum_{w_t} P(W_{t+1}|w_t)P(w_t)$$

For our weather example, this gives us the following two equations:

$$
\begin{aligned}
P(W_t = sun) &= P(W_{t+1} = sun|W_t = sun)P(W_t = sun) + P(W_{t+1} = sun|W_t = rain)P(W_t = rain) \\
&= 0.6 \cdot P(W_t = sun) + 0.1 \cdot P(W_t = rain) \\
P(W_t = rain) &= P(W_{t+1} = rain|W_t = sun)P(W_t = sun) + P(W_{t+1} = rain|W_t = rain)P(W_t = rain) \\
&= 0.4 \cdot P(W_t = sun) + 0.9 \cdot P(W_t = rain)
\end{aligned}
$$

We now have exactly what we need to solve for the stationary distribution, a system of 2 equations in 2 unknowns! We can get a third equation by using the fact that $P(W_t)$ is a probability distribution and so must sum to 1:

$$
\begin{aligned}
P(W_t = sun) &= 0.6 \cdot P(W_t = sun) + 0.1 \cdot P(W_t = rain) \\
P(W_t = rain) &= 0.4 \cdot P(W_t = sun) + 0.9 \cdot P(W_t = rain) \\
1 &= P(W_t = sun) + P(W_t = rain)
\end{aligned}
$$

Solving this system of equations yields $P(W_t = sun) = 0.2$ and $P(W_t = rain) = 0.8$. Hence the table for our stationary distribution, which we'll henceforth denote as $P(W_\infty)$, is the following:

W_∞	$P(W_\infty)$
sun	0.2
rain	0.8

To verify this result, let's apply the transition model to the stationary distribution:

$$
\begin{aligned}
P(W_{\infty+1} = sun) &= P(W_{\infty+1} = sun|W_\infty = sun)P(W_\infty = sun) + P(W_{\infty+1} = sun|W_\infty = rain)P(W_\infty = rain) \\
&= 0.6 \cdot 0.2 + 0.1 \cdot 0.8 = \boxed{0.2} \\
P(W_{\infty+1} = rain) &= P(W_{\infty+1} = rain|W_\infty = sun)P(W_\infty = sun) + P(W_{\infty+1} = rain|W_\infty = rain)P(W_\infty = rain) \\
&= 0.4 \cdot 0.2 + 0.9 \cdot 0.8 = \boxed{0.8}
\end{aligned}
$$

As expected, $P(W_{\infty+1}) = P(W_\infty)$. In general, if W_t had a domain of size k, the equivalence

$$P(W_t) = \sum_{w_t} P(W_{t+1}|w_t)P(w_t)$$

yields a system of k equations, which we can use to solve for the stationary distribution.

Hidden Markov Models

With Markov models, we saw how we could incorporate change over time through a chain of random variables. For example, if we want to know the weather on day 10 with our standard Markov model from above, we can begin with the initial distribution $P(W_0)$ and use the mini-forward algorithm with our transition model to compute $P(W_{10})$. However, between time $t = 0$ and time $t = 10$, we may collect new meteorological evidence that might affect our belief of the probability distribution over the weather at any given timestep. In simpler terms, if the weather forecasts an 80% chance of rain on day 10, but there are clear skies on the night of day 9, that 80% probability might drop drastically. This is exactly what the **Hidden Markov Model** helps us with - it allows us to observe some evidence at each timestep, which can potentially affect the belief distribution at each of the states. The Hidden Markov Model for our weather model can be described using a Bayes' net structure that looks like the following:
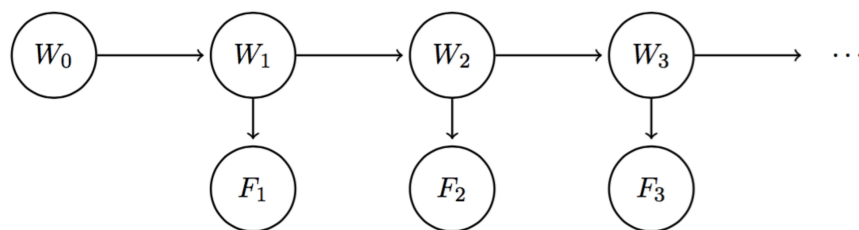
Unlike vanilla Markov models, we now have two different types of nodes. To make this distinction, we'll call each W_i a **state variable** and each weather forecast F_i an **evidence variable**. Since W_i encodes our belief of the probability distribution for the weather on day i, it should be a natural result that the weather forecast for day i is conditionally dependent upon this belief. The model implies similar conditional indepenence relationships as standard Markov models, with an additional set of relationships for the evidence variables:

$$
\begin{aligned}
F_1 &\perp\!\!\!\perp W_0|W_1 \\
\forall i &= 2,\ldots,n; \qquad W_i \perp\!\!\!\perp \{W_0,\ldots,W_{i-2},F_1,\ldots,F_{i-1}\}|W_{i-1} \\
\forall i &= 2,\ldots,n; \qquad F_i \perp\!\!\!\perp \{W_0,\ldots,W_{i-1},F_1,\ldots,F_{i-1}\}|W_i
\end{aligned}
$$

Just like Markov models, Hidden Markov Models make the assumption that the transition model $P(W_{i+1}|W_i)$ is stationary. Hidden Markov Models make the additional simplifying assumption that the **sensor model** $P(F_i|W_i)$ is stationary as well. Hence any Hidden Markov Model can be represented compactly with just three probability tables: the initial distribution, the transition model, and the sensor model.

As a final point on notation, we'll define the **belief distribution** at time i with all evidence $F_1,\ldots,F_i$ observed up to date:

$$B(W_i) = P(W_i|f_1,\ldots,f_i)$$

Similarly, we'll define $B'(W_i)$ as the belief distribution at time i with evidence $f_1,\ldots,f_{i-1}$ observed:

$$B'(W_i) = P(W_i|f_1,\ldots,f_{i-1})$$

Defining e_i as evidence observed at timestep i, you might sometimes see the aggregated evidence from timesteps $1 \leq i \leq t$ reexpressed in the following form:

$$e_{1:t} = e_1, \ldots, e_t$$

Under this notation, $P(W_i|f_1, \ldots, f_{i-1})$ can be written as $P(W_i|f_{1:(i-1)})$. This notation will become relevant in the upcoming sections, where we'll discuss time elapse updates that iteratively incorporate new evidence into our weather model.

The Forward Algorithm

Using the conditional probability assumptions stated above and marginalization properties of conditional probability tables, we can derive a relationship between $B(W_i)$ and $B'(W_{i+1})$ that's of the same form as the update rule for the mini-forward algorithm. We begin by using marginalization:

$$B'(W_{i+1}) = P(W_{i+1}|f_1, \ldots, f_i) = \sum_{w_i} P(W_{i+1}, w_i|f_1, \ldots, f_i)$$

This can be reexpressed then with the chain rule as follows:

$$B'(W_{i+1}) = P(W_{i+1}|f_1, \ldots, f_i) = \sum_{w_i} P(W_{i+1}|w_i, f_1, \ldots, f_i)P(w_i|f_1, \ldots, f_i)$$

Noting that $P(w_i|f_1, \ldots, f_i)$ is simply $B(w_i)$ and that $W_{i+1} \perp\!\!\!\perp \{f_1, \ldots f_i\}|W_i$, this simplies to our final relationship between $B(W_i)$ to $B'(W_{i+1})$:

$$\boxed{B'(W_{i+1}) = \sum_{w_i} P(W_{i+1}|w_i)B(w_i)}$$

Now let's consider how we can derive a relationship between $B'(W_{i+1})$ and $B(W_{i+1})$. By application of the definition of conditional probability (with extra conditioning), we can see that

$$B(W_{i+1}) = P(W_{i+1}|f_1, \ldots, f_{i+1}) = \frac{P(W_{i+1}, f_{i+1}|f_1, \ldots, f_i)}{P(f_{i+1}|f_1, \ldots, f_i)}$$

When dealing with conditional probabilities a commonly used trick is to delay normalization until we require the normalized probabilities, a trick we'll now employ. More specifically, since the denominator in the above expansion of $B(W_{i+1})$ is common to every term in the probability table represented by $B(W_{i+1})$, we can omit actually dividing by $P(f_{i+1}|f_1, \ldots, f_i)$. Instead, we can simply note that $B(W_{i+1})$ is proportional to $P(W_{i+1}, f_{i+1}|f_1, \ldots, f_i)$:

$$B(W_{i+1}) \propto P(W_{i+1}, f_{i+1}|f_1, \ldots, f_i)$$

with a constant of proportionality equal to $P(f_{i+1}|f_1, \ldots, f_i)$. Whenever we decide we want to recover the belief distribution $B(W_{i+1})$, we can divide each computed value by this constant of proportionality. Now, using the chain rule we can observe the following:

$$B(W_{i+1}) \propto P(W_{i+1}, f_{i+1}|f_1, \ldots, f_i) = P(f_{i+1}|W_{i+1}, f_1, \ldots, f_i)P(W_{i+1}|f_1, \ldots, f_i)$$

By the conditional independence assumptions associated with Hidden Markov Models stated previously, $P(f_{i+1}|W_{i+1}, f_1, \ldots, f_i)$ is equivalent to simply $P(f_{i+1}|W_{i+1})$ and by definition $P(W_{i+1}|f_1, \ldots, f_i) = B'(W_{i+1})$. This allows us to express the relationship between $B'(W_{i+1})$ and $B(W_{i+1})$ in it's final form:

$$\boxed{B(W_{i+1}) \propto P(f_{i+1}|W_{i+1})B'(W_{i+1})}$$

Combining the two relationships we've just derived yields an iterative algorithm known as the **forward algorithm**, the Hidden Markov Model analog of the mini-forward algorithm from earlier:

$$B(W_{i+1}) \propto P(f_{i+1}|W_{i+1}) \sum_{w_i} P(W_{i+1}|w_i)B(w_i)$$

The forward algorithm can be thought of as consisting of two distinctive steps: the **time elapse update** which corresponds to determining $B'(W_{i+1})$ from $B(W_i)$ and the **observation update** which corresponds to determining $B(W_{i+1})$ from $B'(W_{i+1})$. Hence, in order to advance our belief distribution by one timestep (i.e. compute $B(W_{i+1})$ from $B(W_i)$), we must first advance our model's state by one timestep with the time elapse update, then incorporate new evidence from that timestep with the observation update. Consider the following initial distribution, transition model, and sensor model:

W_0	$B(W_0)$
sun	0.8
rain	0.2

W_{i+1}	W_i	$P(W_{i+1}|W_i)$
sun	sun	0.6
rain	sun	0.4
sun	rain	0.1
rain	rain	0.9

F_i	W_i	$P(F_i|W_i)$
good	sun	0.8
bad	sun	0.2
good	rain	0.3
bad	rain	0.7

To compute $B(W_1)$, we begin by performing a time update to get $B'(W_1)$:

$$
\begin{aligned}
B'(W_1 = sun) &= \sum_{w_0} P(W_1 = sun|w_0)B(w_0) \\
&= P(W_1 = sun|W_0 = sun)B(W_0 = sun) + P(W_1 = sun|W_0 = rain)B(W_0 = rain) \\
&= 0.6 \cdot 0.8 + 0.1 \cdot 0.2 = \boxed{0.5} \\
B'(W_1 = rain) &= \sum_{w_0} P(W_1 = rain|w_0)B(w_0) \\
&= P(W_1 = rain|W_0 = sun)B(W_0 = sun) + P(W_1 = rain|W_0 = rain)B(W_0 = rain) \\
&= 0.4 \cdot 0.8 + 0.9 \cdot 0.2 = \boxed{0.5}
\end{aligned}
$$

Hence:

W_1	$B'(W_1)$
sun	0.5
rain	0.5

Next, we'll assume that the weather forecast for day 1 was good (i.e. $F_1 = good$), and perform an observation update to get $B(W_1)$:

$$
\begin{aligned}
B(W_1 = sun) &\propto P(F_1 = good|W_1 = sun)B'(W_1 = sun) = 0.8 \cdot 0.5 = \boxed{0.4} \\
B(W_1 = rain) &\propto P(F_1 = good|W_1 = rain)B'(W_1 = rain) = 0.3 \cdot 0.5 = \boxed{0.15}
\end{aligned}
$$

The last step is to normalize $B(W_1)$, noting that the entries in table for $B(W_1)$ sum to $0.4 + 0.15 = 0.55$:

$$
\begin{aligned}
B(W_1 = sun) &= 0.4/0.55 = \frac{8}{11} \\
B(W_1 = rain) &= 0.15/0.55 = \frac{3}{11}
\end{aligned}
$$

Our final table for $B(W_1)$ is thus the following:

W_1	$B'(W_1)$
sun	8/11
rain	3/11

Note the result of observing the weather forecast. Because the weatherman predicted good weather, our belief that it would be sunny increased from $\frac{1}{2}$ after the time update to $\frac{8}{11}$ after the observation update.

As a parting note, the normalization trick discussed above can actually simplify computation significantly when working with Hidden Markov Models. If we began with some initial distribution and were interested in computing the belief distribution at time t, we could use the forward algorithm to iteratively compute $B(W_1), \ldots, B(W_t)$ and normalize only once at the end by dividing each entry in the table for $B(W_t)$ by the sum of it's entries.

Viterbi Algorithm

In the Forward Algorithm, we used recursion to solve for $P(X_N|e_{1:N})$, the probability distribution over states the system could inhabit given the evidence variables observed so far. Another important question related to Hidden Markov Models is: *What is the most likely sequence of hidden states the system followed given the observed evidence variables so far?* In other words, we would like to solve for $\arg\max_{x_{1:N}} P(x_{1:N}|e_{1:N}) = \arg\max_{x_{1:N}} P(x_{1:N}, e_{1:N})$. This trajectory can also be solved for using dynamic programming with the **Viterbi algorithm**.

The algorithm consists of two passes: the first runs forward in time and computes the probability of the best path to each (state, time) tuple given the evidence observed so far. The second pass runs backwards in time: first it finds the terminal state that lies on the path with the highest probability, and then traverses backward through time along the path that leads into this state (which must be the best path).

To visualize the algorithm, consider the following **state trellis**, a graph of states and transitions over time:
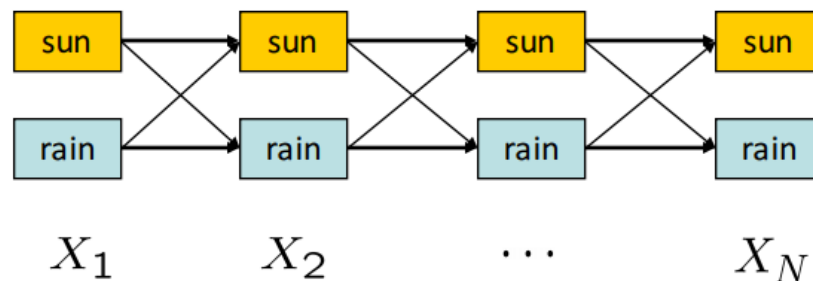
In this HMM with two possible hidden states, sun or rain, we would like to compute the highest probability path (assignment of a state for every timestep) from X_1 to X_N. The weights on an edge from X_{t-1} to X_t is equal to $P(X_t|X_{t-1})P(E_t|X_t)$, and the probability of a path is computed by taking the *product* of its edge weights. The first term in the weight formula represents how likely a particular transition is and the second term represents how well the observed evidence fits the resulting state.

Recall that:

$$P(X_{1:N}, e_{1:N}) = P(X_1)P(e_1|X_1)\prod_{t=2}^{N} P(X_t|X_{t-1})P(e_t|X_t)$$

The Forward Algorithm computes (up to normalization)

$$P(X_N, e_{1:N}) = \sum_{x_1,..,x_{N-1}} P(X_N, x_{1:N-1}, e_{1:N})$$

In the Viberbi Algorithm, we want to compute

$$\arg\max_{x_1,..,x_N} P(x_{1:N}, e_{1:N})$$

to find the maximum likelihood estimate of the sequence of hidden states. Notice that each term in the product is exactly the expression for the edge weight between layer $t-1$ to layer t. So, the product of weights along the path on the trellis gives us the probability of the path given the evidence.

We could solve for a joint probability table over all of the possible hidden states, but this results in an exponential space cost. Given such a table, we could use dynamic programming to compute the best path in polynomial time. However, because we can use dynamic programming to compute the best path, we don't necessarily need the whole table at any given time.

Define $m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t}, e_{1:t})$, or the maximum probability of a path starting at any x_0 and the evidence seen so far to a given x_t at time t. This is the same as the highest weight path through the trellis from step 1 to t. Also note that

$$m_t[x_t] = \max_{x_{1:t-1}} P(e_t|x_t)P(x_t|x_{t-1})P(x_{1:t-1}, e_{1:t-1}) \tag{1}$$

$$= P(e_t|x_t)\max_{x_{t-1}} P(x_t|x_{t-1})\max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1}) \tag{2}$$

$$= P(e_t|x_t)\max_{x_{t-1}} P(x_t|x_{t-1})m_{t-1}[x_{t-1}]. \tag{3}$$

This suggests that we can compute m_t for all t recursively via dynamic programming. This makes it possible to determine the last state x_N for the most likely path, but we still need a way to backtrack to reconstruct the entire path. Let's define $a_t[x_t] = P(e_t|x_t)\arg\max_{x_{t-1}} P(x_t|x_{t-1})m_{t-1}[x_{t-1}] = \arg\max_{x_{t-1}} P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$ to keep track of the last transition along the best path to x_t. We can now outline the algorithm.

Result: Most likely sequence of hidden states $x^*_{1:N}$

```
/* Forward pass                                                          */
for t = 1 to N do
    for x_t ∈ 𝒳 do
        if t = 1 then
            | m_t[x_t] = P(x_t)P(e_0|x_t)
        else
            a_t[x_t] = arg max_{x_{t-1}} P(x_t|x_{t-1})m_{t-1}[x_{t-1}];
            m_t[x_t] = P(e_t|x_t)P(x_t|a_t[x_t])m_{t-1}[a_t[x_t]];
        end
    end
end
/* Find the most likely path's ending point                             */
x*_N = arg max_{x_N} m_N[x_N];
/* Work backwards through our most likely path and find the hidden
    states                                                              */
for t = N to 2 do
    | x*_{t-1} = a_t[x*_t];
end
```

Notice that our a arrays define a set of N sequences, each of which is the most likely sequence to a particular end state x_N. Once we finish the forward pass, we look at the likelihood of the N sequences, pick the best one, and reconstruct it in the backwards pass. We have thus computed the most likely explanation for our evidence in polynomial space and time.