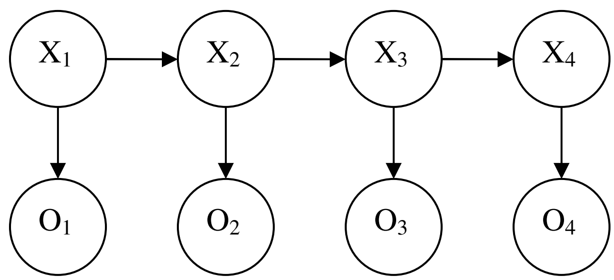
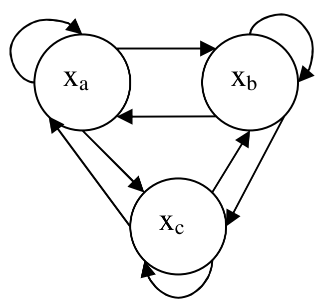
Hidden Markov Model Inference

A Hidden Markov Model (HMM) has hidden states $X_1, \ldots, X_n$ and observed states $O_1, \ldots, O_n$ that have values $x_1, \ldots, x_n$ and $o_1, \ldots, o_n$. These values come from some set (or alphabet) $x_i \varepsilon \{x_a, x_b, \ldots\}$, $o_i \varepsilon \{o_a, o_b, \ldots\}$.

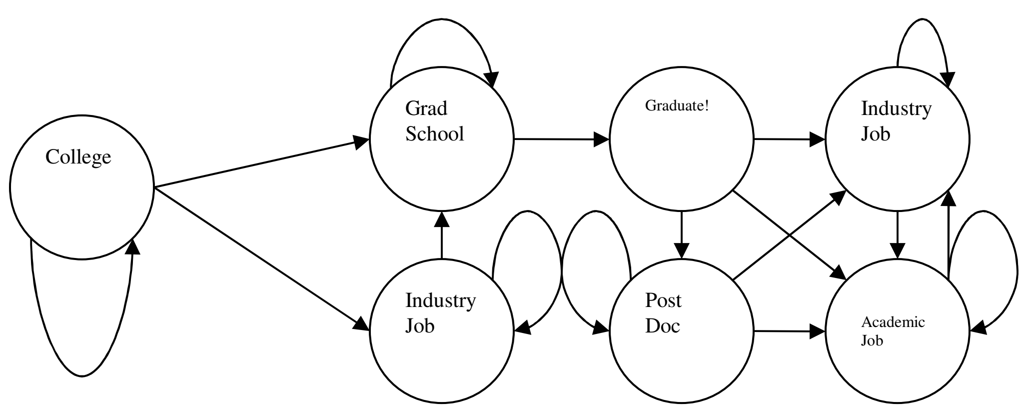


An HMM is specified by the initial state probability $P(X_1)$ for each $\{x_a, x_b, \ldots\}$, the state transition probability $P(X_{i+1}|X_i)$ for pairs of $\{x_a, x_b, \ldots\}$, and the output probability $P(O_i|X_i)$ for $X_i = \{x_a, x_b, \ldots\}$ and $O_i = \{o_a, o_b, \ldots\}$. For the purpose of inference, these probability tables are known. One main assumption for Hidden Markov models is that the state and out probabilities are time-invariant, holding equivalently for timestep 2 and 4.

A (sometimes) useful visualization for the state transition model is to draw the graph with edges weighted by the conditional probability of traveling from the origin state to the destination state.



For (an entirely unrealistic) example, life after college



Some useful observations for this HMM might be "what hour you go to sleep," "how much of your diet consists of Ramen noodles," "how many papers you read," etc. ☺

Remember, Bayes Nets factor (using the Chain Rule) according to their parents, so for a HMM:

$P(X_1, \dots X_n, O_1, \dots O_n) = P(X_1) \prod_{i=2 \text{ to } n} P(X_i|X_{i-1}) \prod_{i=1 \text{ to } n} P(O_i|X_i)$.

Viterbi Decoding

We want to find the choice of hidden variable assignments that has the highest probability (given the known observations).

$\text{Argmax}_{x1,x2,\dots xN} P(X_1=x_1, X_2=x_2, \dots X_n=x_n|O_1=o_1,\dots O_n=o_n)$
$= \text{Argmax}_{x1,x2,\dots xN} P(X_1=x_1, X_2=x_2, \dots X_n=x_n, O_1=o_1,\dots O_n=o_n)$
By Bayes' rule and because $P(O_1=o_1,\dots O_n=o_n)$ is just a constant since $o_1..o_n$ known

We can "eliminate" variables in either direction along the chain.
This is just simple algebra for what order we distribute the maximization. First use the chain rule for the variables and "push" the maximums as far right as possible:

$$\begin{aligned}
= \text{Argmax}_{x4}\ & P(O_4 = o_4|X_4 = x_4) * \\
& \max_{x3} P(X_4 = x_4|X_3 = x_3)\, P(O_3 = o_3|X_3 = x_3) * \\
& \max_{x2} P(X_3 = x_3|X_2 = x_2)\, P(O_2 = o_2|X_2 = x_2) * \\
& \max_{x1} P(X_2 = x_2|X_1 = x_1)\, P(X_1 = x_1)\, P(O_1 = o_1|X_1 = x_1)
\end{aligned}$$

Let $\alpha_{i+1}(X_{i+1} = x_{i+1}) = \max_{xi} P(X_{i+1} = x_{i+1}|X_i = x_i)\, P(O_i = o_i|X_i = x_i)\, \alpha_i(X_i = x_i)$
Base case: $\alpha_0(\dots) = 1$

Then:

$\text{Argmax}_{x1,x2,\dots xN} P(X_1=x_1, X_2=x_2, \dots X_n=x_n, O_1=o_1,\dots O_n=o_n)$
$= \text{Argmax}_{x4} P(O_4 = o_4|X_4 = x_4)\ \alpha_4(X_4 = x_4)$
$= \text{Argmax}_{x4} P(O_4 = o_4|X_4 = x_4) \max_{x3} P(X_4 = x_4|X_3 = x_3)\, P(O_3 = o_3|X_3 = x_3)\, \alpha_3(X_3 = x_3)$
$= \dots$

Algorithm
Set: $\alpha_0(\dots) = 1$
Then compute $\alpha_1(x_1)$ for each choice of x_1
Then compute $\alpha_2(x_2)$ for each choice of x_2
Once you have $\alpha_n(x_n)$, find the x_n^* that maximizes $P(O_n = o_n|X_n = x_n)\, \alpha_n(X_n = x_n)$.
Now you can go backwards and find the x_{n-1}^* such that:
$\alpha_n(X_n = x_n^*) = P(X_n = x_n^*|X_{n-1} = x_{n-1})\, P(O_n = o_n|X_n = x_n^*)\, \alpha_{n-1}(X_{n-1} = x_{n-1})$
Then find x_{n-2}^* in the same way until you've found all $x_1^* \dots x_n^*$.

Or we can eliminate in the other direction as well:

$= \text{Argmax}_{x1}\ P(X_1 = x_1)\ P(O_1 = o_1|X_1 = x_1)\ *$
$\quad \max_{x2}\ P(X_2 = x_2|X_1 = x_1)\ P(O_2 = o_2|X_2 = x_2)\ *$
$\quad \max_{x3}\ P(X_3 = x_3|X_2 = x_2)\ P(O_3 = o_3|X_3 = x_3)\ *$
$\quad \max_{x4}\ P(X_4 = x_4|X_3 = x_3)\ P(O_4 = o_4|X_4 = x_4)$

Let $\beta_{i-1}(X_{i-1} = x_{i-1}) = \max_{xi}\ P(X_i = x_i|X_{i-1} = x_{i-1})\ P(O_i = o_i|X_i = x_i)\ \beta_i(X_i = x_i)$
Base case: $\beta_n(\ldots) = 1$

Then:

$\text{Argmax}_{x1,x2,\ldots xN}\ P(X_1=x_1, X_2=x_2, \ldots X_n=x_n, O_1=o_1,\ldots O_n=o_n)$
$= \text{Argmax}_{x1}\ P(X_1 = x_1)\ P(O_1 = o_1|X_1 = x_1)\ *\ \beta_1(X_1=x_1)$
$= \text{Argmax}_{x1}\ P(X_1 = x_1)\ P(O_1 = o_1|X_1 = x_1)\ *$
$\quad \max_{x2}\ P(X_2 = x_2|X_1 = x_1)\ P(O_2 = o_2|X_2 = x_2)\ \beta_2(X_2=x_2)$
$= \ldots$

Algorithm
Set: $\beta_n(\ldots) = 1$
Then compute $\beta_{n-1}(x_{n-1})$ for each choice of x_{n-1}
Then compute $\beta_{n-1}(x_{n-2})$ for each choice of x_{n-2}
Once you have $\beta_1(x_1)$, find the x_1^* that maximizes $P(X_1 = x_1)\ P(O_1 = o_1|X_1 = x_1)\ \beta_1(x_1)$
Then find x_2^* such that $\beta_2(x_2^*) = P(X_2 = x_2^*|X_1 = x_1)\ P(O_2 = o_2|X_2 = x_2^*)\ \beta_1(x_1^*)$
Then find x_3^* similarly…

Forward-Backward Algorithm

We want to find the marginal probability of each hidden variable X_j.

$P(X_j|O_1=o_1,\ldots, O_n=o_n) = P(X_j, O_1=o_1,\ldots, O_n=o_n) * C$
Some constant C, by Bayes' rule and because $P(O_1=o_1,\ldots O_n=o_n)$ is just a constant since $o_1..o_n$ known
$P(X_j, O_1=o_1,\ldots, O_n=o_n) = \sum_{x1,\ x2,\ \ldots\ x(i-1),\ x(i+1),\ \ldots,\ xn} P(X_1, \ldots X_n, O_1=o_1,\ldots, O_n=o_n)$
We are just summing over all other hidden variables except for the one we want the marginal of. This is the definition of marginalization.

Bayes Nets factor according to parents, so…
$P(X_1,\ldots X_n, O_1, \ldots, O_n) = P(X_1) \prod_{i=2\text{ to }n} P(X_i|X_{i-1}) \prod_{i=1\text{ to }n} P(O_i|X_i)$

$= P(X_1) \prod_{i=2\text{ to }j} P(X_i|X_{i-1}) \prod_{i=1\text{ to }j} P(O_i|X_i) \prod_{i=(j+1)\text{ to }n} P(X_i|X_{i-1}) \prod_{i=(j+1)\text{ to }n} P(O_i|X_i)$
Splitting up the products
$= P(X_1, \ldots, X_j, O_1, \ldots, O_i) * P(X_{j+1}, \ldots X_n, O_{i+1}, \ldots O_n|X_j)$

So now to get the marginal of X_j

$\sum_{x1,\, x2,\, \ldots\, x(j-1),\, x(j+1),\, \ldots,\, xn} P(X_1, \ldots X_n, O_1=o_1, \ldots, O_n=o_n)$
$= \sum_{x1,\, x2,\, \ldots\, x(j-1)} P(X_1, \ldots, X_j, O_1, \ldots, O_j) \sum_{x(j+1),\, \ldots,\, xn} P(X_{j+1}, \ldots X_n, O_{j+1}, \ldots O_n|X_j)$
$= \alpha_j(x_j)\, \beta_j(x_j)$

Let $\alpha_j(x_j) = \sum_{x1,\, x2,\, \ldots\, x(j-1)} P(X_1, \ldots, X_j, O_1, \ldots, O_i)$
Let $\beta_j(x_j) = \sum_{x(j+1),\, \ldots,\, xn} P(X_{j+1}, \ldots X_n, O_{j+1}, \ldots O_n|X_j)$

Can we compute these efficiently? Replace every "max" with a "sum" in the Viterbi computations of $\alpha_i(x_i)$ and $\beta_i(x_i)$ and run the same algorithms. Voila!

Discussion

When would you want to find the best "string" of hidden states versus finding the distribution for each individual hidden state?

This depends on the application. If you are trying to make predictions based on having the whole string (e.g., if the string is a word), then Viterbi makes more sense since than taking the most likely choice for each marginal probability individually, which may produce gibberish. This is because the single most likely sequence of states could differ greatly from the sum of a number of possible sequences with a different value in one of the hidden variables.

A simple example
"aaa" 30% probability
"abb" 20% probability
"bab" 25% probability
"bbb" 25% probability

"aaa" is the most likely sequence
"a" is the most likely first character
"a" is the most likely second character
"b" is the most likely third character, … but the string "aab" has 0 probability

For other applications, knowing the probability of the hidden state at a particular instant is more important (e.g., P(fire)) since there may be many very unlikely paths that led to that marginal probability overall being reasonably large (maybe enough to call the fire department?)