Hidden Markov Models Fundamentals

Daniel Ramage
CS229 Section Notes

December 1, 2007

Abstract

How can we apply machine learning to data that is represented as a sequence of observations over time? For instance, we might be interested in discovering the sequence of words that someone spoke based on an audio recording of their speech. Or we might be interested in annotating a sequence of words with their part-of-speech tags. These notes provides a thorough mathematical introduction to the concept of Markov Models — a formalism for reasoning about states over time — and Hidden Markov Models — where we wish to recover a series of states from a series of observations. The final section includes some pointers to resources that present this material from other perspectives.

1 Markov Models

Given a set of states $S = \{s_1, s_2, ... s_{|S|}\}$ we can observe a series over time $\vec{z} \in S^T$. For example, we might have the states from a weather system $S = \{sun, cloud, rain\}$ with $|S| = 3$ and observe the weather over a few days $\{z_1 = s_{sun}, z_2 = s_{cloud}, z_3 = s_{cloud}, z_4 = s_{rain}, z_5 = s_{cloud}\}$ with $T = 5$.

The observed states of our weather example represent the output of a random process over time. Without some further assumptions, state s_j at time t could be a function of any number of variables, including all the states from times 1 to $t-1$ and possibly many others that we don't even model. However, we will make two MARKOV ASSUMPTIONS that will allow us to tractably reason about time series.

The LIMITED HORIZON ASSUMPTION is that the probability of being in a state at time t depends only on the state at time $t-1$. The intuition underlying this assumption is that the state at time t represents "enough" summary of the past to reasonably predict the future. Formally:

$$P(z_t | z_{t-1}, z_{t-2}, ..., z_1) = P(z_t | z_{t-1})$$

The STATIONARY PROCESS ASSUMPTION is that the conditional distribution over next state given current state does not change over time. Formally:

$$P(z_t|z_{t-1}) = P(z_2|z_1);\ t \in 2...T$$

As a convention, we will also assume that there is an initial state and initial observation $z_0 \equiv s_0$, where s_0 represents the initial probability distribution over states at time 0. This notational convenience allows us to encode our belief about the prior probability of seeing the first real state z_1 as $P(z_1|z_0)$. Note that $P(z_t|z_{t-1}, ..., z_1) = P(z_t|z_{t-1}, ..., z_1, z_0)$ because we've defined $z_0 = s_0$ for any state sequence. (Other presentations of HMMs sometimes represent these prior believes with a vector $\pi \in \mathbb{R}^{|S|}$.)

We parametrize these transitions by defining a state transition matrix $A \in \mathbb{R}^{(|S|+1)\times(|S|+1)}$. The value A_{ij} is the probability of transitioning from state i to state j at any time t. For our sun and rain example, we might have following transition matrix:

$$A = \begin{array}{ccccc} & s_0 & s_{sun} & s_{cloud} & s_{rain} \\ s_0 & 0 & .33 & .33 & .33 \\ s_{sun} & 0 & .8 & .1 & .1 \\ s_{cloud} & 0 & .2 & .6 & .2 \\ s_{rain} & 0 & .1 & .2 & .7 \end{array}$$

Note that these numbers (which I made up) represent the intuition that the weather is self-correlated: if it's sunny it will tend to stay sunny, cloudy will stay cloudy, etc. This pattern is common in many Markov models and can be observed as a strong diagonal in the transition matrix. Note that in this example, our initial state s_0 shows uniform probability of transitioning to each of the three states in our weather system.

1.1 Two questions of a Markov Model

Combining the Markov assumptions with our state transition parametrization A, we can answer two basic questions about a sequence of states in a Markov chain. What is the probability of a particular sequence of states $\vec{z}$? And how do we estimate the parameters of our model A such to maximize the likelihood of an observed sequence $\vec{z}$?

1.1.1 Probability of a state sequence

We can compute the probability of a particular series of states $\vec{z}$ by use of the chain rule of probability:

$$\begin{aligned} P(\vec{z}) &= P(z_t, z_{t-1}, ..., z_1; A) \\ &= P(z_t, z_{t-1}, ..., z_1, z_0; A) \\ &= P(z_t|z_{t-1}, z_{t-2}, ..., z_1; A)P(z_{t-1}|z_{t-2}, ..., z_1; A)...P(z_1|z_0; A) \\ &= P(z_t|z_{t-1}; A)P(z_{t-1}|z_{t-2}; A)...P(z_2|z_1; A)P(z_1|z_0; A) \end{aligned}$$

$$
\begin{aligned}
&= \prod_{t=1}^{T} P(z_t|z_{t-1};A) \\
&= \prod_{t=1}^{T} A_{z_{t-1}\, z_t}
\end{aligned}
$$

In the second line we introduce z_0 into our joint probability, which is allowed by the definition of z_0 above. The third line is true of any joint distribution by the chain rule of probabilities or repeated application of Bayes rule. The fourth line follows from the Markov assumptions and the last line represents these terms as their elements in our transition matrix A.

Let's compute the probability of our example time sequence from earlier. We want $P(z_1 = s_{sun}, z_2 = s_{cloud}, z_3 = s_{rain}, z_4 = s_{rain}, z_5 = s_{cloud})$ which can be factored as $P(s_{sun}|s_0)P(s_{cloud}|s_{sun})P(s_{rain}|s_{cloud})P(s_{rain}|s_{rain})P(s_{cloud}|s_{rain}) = .33 \times .1 \times .2 \times .7 \times .2$.

1.1.2 Maximum likelihood parameter assignment

From a learning perspective, we could seek to find the parameters A that maximize the log-likelihood of sequence of observations $\vec{z}$. This corresponds to finding the likelihoods of transitioning from sunny to cloudy versus sunny to sunny, etc., that make a set of observations most likely. Let's define the log-likelihood a Markov model.

$$
\begin{aligned}
l(A) &= \log P(\vec{z};A) \\
&= \log \prod_{t=1}^{T} A_{z_{t-1}\, z_t} \\
&= \sum_{t=1}^{T} \log A_{z_{t-1}\, z_t} \\
&= \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}
\end{aligned}
$$

In the last line, we use an indicator function whose value is one when the condition holds and zero otherwise to select the observed transition at each time step. When solving this optimization problem, it's important to ensure that solved parameters A still make a valid transition matrix. In particular, we need to enforce that the outgoing probability distribution from state i always sums to 1 and all elements of A are non-negative. We can solve this optimization problem using the method of Lagrange multipliers.

$$
\max_{A} \quad l(A)
$$

$$
\begin{aligned}
s.t. \qquad & \sum_{j=1}^{|S|} A_{ij} = 1, \;\; i = 1..|S| \\
& A_{ij} \geq 0, \;\; i,j = 1..|S|
\end{aligned}
$$

This constrained optimization problem can be solved in closed form using the method of Lagrange multipliers. We'll introduce the equality constraint into the Lagrangian, but the inequality constraint can safely be ignored — the optimal solution will produce positive values for A_{ij} anyway. Therefore we construct the Lagrangian as:

$$
\mathcal{L}(A,\alpha) = \sum_{i=1}^{|S|}\sum_{j=1}^{|S|}\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} + \sum_{i=1}^{|S|} \alpha_i (1 - \sum_{j=1}^{|S|} A_{ij})
$$

Taking partial derivatives and setting them equal to zero we get:

$$
\begin{aligned}
\frac{\partial \mathcal{L}(A,\alpha)}{\partial A_{ij}} &= \frac{\partial}{\partial A_{ij}} (\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}) + \frac{\partial}{\partial A_{ij}} \alpha_i (1 - \sum_{j=1}^{|S|} A_{ij}) \\
&= \frac{1}{A_{ij}} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \alpha_i \equiv 0 \\
&\Rightarrow \\
A_{ij} &= \frac{1}{\alpha_i} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}
\end{aligned}
$$

Substituting back in and setting the partial with respect to α equal to zero:

$$
\begin{aligned}
\frac{\partial \mathcal{L}(A,\beta)}{\partial \alpha_i} &= 1 - \sum_{j=1}^{|S|} A_{ij} \\
&= 1 - \sum_{j=1}^{|S|} \frac{1}{\alpha_i} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \equiv 0 \\
&\Rightarrow \\
\alpha_i &= \sum_{j=1}^{|S|} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\
&= \sum_{t=1}^{T} 1\{z_{t-1} = s_i\}
\end{aligned}
$$

Substituting in this value for α_i into the expression we derived for A_{ij} we obtain our final maximum likelihood parameter value for $\hat{A}_{ij}$.

$$\hat{A}_{ij} = \frac{\sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{t=1}^{T} 1\{z_{t-1} = s_i\}}$$

This formula encodes a simple intuition: the maximum likelihood probability of transitioning from state i to state j is just the number of times we transition from i to j divided by the total number of times we are in i. In other words, the maximum likelihood parameter corresponds to the fraction of the time when we were in state i that we transitioned to j.

2 Hidden Markov Models

Markov Models are a powerful abstraction for time series data, but fail to capture a very common scenario. How can we reason about a series of states if we cannot observe the states themselves, but rather only some probabilistic function of those states? This is the scenario for part-of-speech tagging where the words are observed but the parts-of-speech tags aren't, and for speech recognition where the sound sequence is observed but not the words that generated it. For a simple example, let's borrow the setup proposed by Jason Eisner in 2002 [1], "Ice Cream Climatology."

> The situation: You are a climatologist in the year 2799, studying the history of global warming. You can't find any records of Baltimore weather, but you do find my (Jason Eisner's) diary, in which I assiduously recorded how much ice cream I ate each day. *What can you figure out from this about the weather that summer?*

A Hidden Markov Model (HMM) can be used to explore this scenario. We don't get to observe the actual sequence of states (the weather on each day). Rather, we can only observe some outcome generated by each state (how many ice creams were eaten that day).

Formally, an HMM is a Markov model for which we have a series of *observed* outputs $x = \{x_1, x_2, ..., x_T\}$ drawn from an output alphabet $V = \{v_1, v_2, ..., v_{|V|}\}$, i.e. $x_t \in V$, $t = 1..T$. As in the previous section, we also posit the existence of series of states $z = \{z_1, z_2, ..., z_T\}$ drawn from a state alphabet $S = \{s_1, s_2, ...s_{|S|}\}$, $z_t \in S$, $t = 1..T$ but in this scenario the values of the states are *unobserved*. The transition between states i and j will again be represented by the corresponding value in our state transition matrix A_{ij}.

We also model the probability of generating an output observation as a function of our hidden state. To do so, we make the OUTPUT INDEPENDENCE ASSUMPTION and define $P(x_t = v_k | z_t = s_j) = P(x_t = v_k | x_1, ..., x_T, z_1, ..., z_T) = B_{jk}$. The matrix B encodes the probability of our hidden state generating output v_k given that the state at the corresponding time was s_j.

Returning to the weather example, imagine that you have logs of ice cream consumption over a four day period: $\vec{x} = \{x_1 = v_3, x_2 = v_2, x_3 = v_1, x_4 = v_2\}$

where our alphabet just encodes the number of ice creams consumed, i.e. $V = \{v_1 = 1\,ice\,cream, v_2 = 2\,ice\,creams, v_3 = 3\,ice\,creams\}$. What questions can an HMM let us answer?

2.1 Three questions of a Hidden Markov Model

There are three fundamental questions we might ask of an HMM. What is the probability of an observed sequence (how likely were we to see $3, 2, 1, 2$ ice creams consumed)? What is the most likely series of states to generate the observations (what was the weather for those four days)? And how can we learn values for the HMM's parameters A and B given some data?

2.2 Probability of an observed sequence: Forward procedure

In an HMM, we assume that our data was generated by the following process: posit the existence of a series of states $\vec{z}$ over the length of our time series. This state sequence is generated by a Markov model parametrized by a state transition matrix A. At each time step t, we select an output x_t as a function of the state z_t. Therefore, to get the probability of a sequence of observations, we need to add up the likelihood of the data $\vec{x}$ given every possible series of states.

$$\begin{aligned} P(\vec{x}; A, B) &= \sum_{\vec{z}} P(\vec{x}, \vec{z}; A, B) \\ &= \sum_{\vec{z}} P(\vec{x}|\vec{z}; A, B) P(\vec{z}; A, B) \end{aligned}$$

The formulas above are true for any probability distribution. However, the HMM assumptions allow us to simplify the expression further:

$$\begin{aligned} P(\vec{x}; A, B) &= \sum_{\vec{z}} P(\vec{x}|\vec{z}; A, B) P(\vec{z}; A, B) \\ &= \sum_{\vec{z}} (\prod_{t=1}^{T} P(x_t|z_t; B)) \, (\prod_{t=1}^{T} P(z_t|z_{t-1}; A)) \\ &= \sum_{\vec{z}} (\prod_{t=1}^{T} B_{z_t\, x_t}) \, (\prod_{t=1}^{T} A_{z_{t-1}\, z_t}) \end{aligned}$$

The good news is that this is a simple expression in terms of our parameters. The derivation follows the HMM assumptions: the output independence assumption, Markov assumption, and stationary process assumption are all used to derive the second line. The bad news is that the sum is over every possible assignment to $\vec{z}$. Because z_t can take one of $|S|$ possible values at each time step, evaluating this sum directly will require $O(|S|^T)$ operations.

Algorithm 1 Forward Procedure for computing $\alpha_i(t)$

1. Base case: $\alpha_i(0) = A_{0\,i}$, $i = 1..|S|$
2. Recursion: $\alpha_j(t) = \sum_{i=1}^{|S|} \alpha_i(t-1) A_{ij} B_{j\,x_t}$, $\;j = 1..|S|$, $t = 1..T$

Fortunately, a faster means of computing $P(\vec{x}; A, B)$ is possible via a dynamic programming algorithm called the FORWARD PROCEDURE. First, let's define a quantity $\alpha_i(t) = P(x_1, x_2, ..., x_t, z_t = s_i; A, B)$. $\alpha_i(t)$ represents the total probability of all the observations up through time t (by any state assignment) and that we are in state s_i at time t. If we had such a quantity, the probability of our full set of observations $P(\vec{x})$ could be represented as:

$$\begin{aligned} P(\vec{x}; A, B) &= P(x_1, x_2, ..., x_T; A, B) \\ &= \sum_{i=1}^{|S|} P(x_1, x_2, ..., x_T, z_T = s_i; A, B) \\ &= \sum_{i=1}^{|S|} \alpha_i(T) \end{aligned}$$

Algorithm 2.2 presents an efficient way to compute $\alpha_i(t)$. At each time step we must do only $O(|S|)$ operations, resulting in a final algorithm complexity of $O(|S| \cdot T)$ to compute the total probability of an observed state sequence $P(\vec{x}; A, B)$.

A similar algorithm known as the BACKWARD PROCEDURE can be used to compute an analogous probability $\beta_i(t) = P(x_T, x_{T-1}, .., x_{t+1}, z_t = s_i; A, B)$.

2.3 Maximum Likelihood State Assignment: The Viterbi Algorithm

One of the most common queries of a Hidden Markov Model is to ask what was the most likely series of states $\vec{z} \in S^T$ given an observed series of outputs $\vec{x} \in V^T$. Formally, we seek:

$$\arg\max_{\vec{z}} P(\vec{z}|\vec{x}; A, B) = \arg\max_{\vec{z}} \frac{P(\vec{x}, \vec{z}; A, B)}{\sum_{\vec{z}} P(\vec{x}, \vec{z}; A, B)} = \arg\max_{\vec{z}} P(\vec{x}, \vec{z}; A, B)$$

The first simplification follows from Bayes rule and the second from the observation that the denominator does not directly depend on $\vec{z}$. Naively, we might try every possible assignment to $\vec{z}$ and take the one with the highest joint probability assigned by our model. However, this would require $O(|S|^T)$ operations just to enumerate the set of possible assignments. At this point, you might think a dynamic programming solution like the Forward Algorithm might save the day, and you'd be right. Notice that if you replaced the $\arg\max_{\vec{z}}$ with $\sum_{\vec{z}}$, our current task is exactly analogous to the expression which motivated the forward procedure.

Algorithm 2 Naive application of EM to HMMs

Repeat until convergence {

(E-Step) For every possible labeling $\vec{z} \in S^T$, set

$$Q(\vec{z}) \quad := \quad p(\vec{z}|\vec{x}; A, B)$$

(M-Step) Set

$$\begin{aligned} A, B \quad &:= \quad \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log \frac{P(\vec{x}, \vec{z}; A, B)}{Q(\vec{z})} \\ s.t. \quad &\sum_{j=1}^{|S|} A_{ij} = 1,\ i = 1..|S|;\ \ A_{ij} \geq 0,\ \ i, j = 1..|S| \\ &\sum_{k=1}^{|V|} B_{ik} = 1,\ i = 1..|S|;\ \ B_{ik} \geq 0,\ \ i = 1..|S|, k = 1..|V| \end{aligned}$$

}

The Viterbi Algorithm is just like the forward procedure except that instead of tracking the total probability of generating the observations seen so far, we need only track the *maximum* probability and record its corresponding state sequence.

2.4 Parameter Learning: EM for HMMs

The final question to ask of an HMM is: given a set of observations, what are the values of the state transition probabilities A and the output emission probabilities B that make the data most likely? For example, solving for the maximum likelihood parameters based on a speech recognition dataset will allow us to effectively train the HMM before asking for the maximum likelihood state assignment of a candidate speech signal.

In this section, we present a derivation of the Expectation Maximization algorithm for Hidden Markov Models. This proof follows from the general formulation of EM presented in the CS229 lecture notes. Algorithm 2.4 shows the basic EM algorithm. Notice that the optimization problem in the M-Step is now constrained such that A and B contain valid probabilities. Like the maximum likelihood solution we found for (non-Hidden) Markov models, we'll be able to solve this optimization problem with Lagrange multipliers. Notice also that the E-Step and M-Step both require enumerating all $|S|^T$ possible labellings of $\vec{z}$. We'll make use of the Forward and Backward algorithms mentioned earlier to compute a set of sufficient statistics for our E-Step and M-Step tractably.

First, let's rewrite the objective function using our Markov assumptions.

$$
\begin{aligned}
A, B &= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log \frac{P(\vec{x}, \vec{z}; A, B)}{Q(\vec{z})} \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log P(\vec{x}, \vec{z}; A, B) \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \log (\prod_{t=1}^{T} P(x_t | z_t; B)) \, (\prod_{t=1}^{T} P(z_t | z_{t-1}; A)) \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} \log B_{z_t \, x_t} + \log A_{z_{t-1} \, z_t} \\
&= \arg\max_{A,B} \sum_{\vec{z}} Q(\vec{z}) \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{k=1}^{|V|} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \log B_{jk} + 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij}
\end{aligned}
$$

In the first line we split the log division into a subtraction and note that the denominator's term does not depend on the parameters A, B. The Markov assumptions are applied in line 3. Line 5 uses indicator functions to index A and B by state.

Just as for the maximum likelihood parameters for a visible Markov model, it is safe to ignore the inequality constraints because the solution form naturally results in only positive solutions. Constructing the Lagrangian:

$$
\begin{aligned}
\mathcal{L}(A, B, \delta, \epsilon) &= \sum_{\vec{z}} Q(\vec{z}) \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{k=1}^{|V|} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \log B_{jk} + 1\{z_{t-1} = s_i \wedge z_t = s_j\} \log A_{ij} \\
&\quad + \sum_{j=1}^{|S|} \epsilon_j (1 - \sum_{k=1}^{|V|} B_{jk}) + \sum_{i=1}^{|S|} \delta_i (1 - \sum_{j=1}^{|S|} A_{ij})
\end{aligned}
$$

Taking partial derivatives and setting them equal to zero:

$$
\begin{aligned}
\frac{\partial \mathcal{L}(A, B, \delta, \epsilon)}{\partial A_{ij}} &= \sum_{\vec{z}} Q(\vec{z}) \frac{1}{A_{ij}} \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} - \delta_i \equiv 0 \\
A_{ij} &= \frac{1}{\delta_i} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}(A, B, \delta, \epsilon)}{\partial B_{jk}} &= \sum_{\vec{z}} Q(\vec{z}) \frac{1}{B_{jk}} \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} - \epsilon_j \equiv 0 \\
B_{jk} &= \frac{1}{\epsilon_j} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\}
\end{aligned}
$$

Taking partial derivatives with respect to the Lagrange multipliers and substituting our values of A_{ij} and B_{jk} above:

$$\begin{aligned}
\frac{\partial \mathcal{L}(A,B,\delta,\epsilon)}{\partial \delta_i} &= 1 - \sum_{j=1}^{|S|} A_{ij} \\
&= 1 - \sum_{j=1}^{|S|} \frac{1}{\delta_i} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \equiv 0 \\
\delta_i &= \sum_{j=1}^{|S|} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\} \\
&= \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i\}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}(A,B,\delta,\epsilon)}{\partial \epsilon_j} &= 1 - \sum_{k=1}^{|V|} B_{jk} \\
&= 1 - \sum_{k=1}^{|V|} \frac{1}{\epsilon_j} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \equiv 0 \\
\epsilon_j &= \sum_{k=1}^{|V|} \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\} \\
&= \sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\}
\end{aligned}$$

Substituting back into our expressions above, we find that parameters $\hat{A}$ and $\hat{B}$ that maximize our predicted counts with respect to the dataset are:

$$\begin{aligned}
\hat{A}_{ij} &= \frac{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i\}} \\
\hat{B}_{jk} &= \frac{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j \wedge x_t = v_k\}}{\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\}}
\end{aligned}$$

Unfortunately, each of these sums is over all possible labellings $\vec{z} \in S^T$. But recall that $Q(\vec{z})$ was defined in the E-step as $P(\vec{z}|\vec{x}; A, B)$ for parameters A and B at the last time step. Let's consider how to represent first the numerator of $\hat{A}_{ij}$ in terms of our forward and backward probabilities, $\alpha_i(t)$ and $\beta_j(t)$.

$$\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_{t-1} = s_i \wedge z_t = s_j\}$$

$$
\begin{aligned}
&= \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1}=s_i \wedge z_t=s_j\}Q(\vec{z}) \\
&= \sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1}=s_i \wedge z_t=s_j\}P(\vec{z}|\vec{x};A,B) \\
&= \frac{1}{P(\vec{x};A,B)}\sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1}=s_i \wedge z_t=s_j\}P(\vec{z},\vec{x};A,B) \\
&= \frac{1}{P(\vec{x};A,B)}\sum_{t=1}^{T} \alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)
\end{aligned}
$$

In the first two steps we rearrange terms and substitute in for our definition of Q. Then we use Bayes rule in deriving line four, followed by the definitions of α, β, A, and B, in line five. Similarly, the denominator can be represented by summing out over j the value of the numerator.

$$
\begin{aligned}
&\sum_{\vec{z}} Q(\vec{z})\sum_{t=1}^{T} 1\{z_{t-1}=s_i\} \\
&= \sum_{j=1}^{|S|}\sum_{\vec{z}} Q(\vec{z})\sum_{t=1}^{T} 1\{z_{t-1}=s_i \wedge z_t=s_j\} \\
&= \frac{1}{P(\vec{x};A,B)}\sum_{j=1}^{|S|}\sum_{t=1}^{T} \alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)
\end{aligned}
$$

Combining these expressions, we can fully characterize our maximum likelihood state transitions $\hat{A}_{ij}$ without needing to enumerate all possible labellings as:

$$
\hat{A}_{ij} = \frac{\sum_{t=1}^{T}\alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)}{\sum_{j=1}^{|S|}\sum_{t=1}^{T}\alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)}
$$

Similarly, we can represent the numerator for $\hat{B}_{jk}$ as:

$$
\begin{aligned}
&\sum_{\vec{z}} Q(\vec{z})\sum_{t=1}^{T} 1\{z_t=s_j \wedge x_t=v_k\} \\
&= \frac{1}{P(\vec{x};A,B)}\sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_t=s_j \wedge x_t=v_k\}P(\vec{z},\vec{x};A,B) \\
&= \frac{1}{P(\vec{x};A,B)}\sum_{i=1}^{|S|}\sum_{t=1}^{T}\sum_{\vec{z}} 1\{z_{t-1}=s_i \wedge z_t=s_j \wedge x_t=v_k\}P(\vec{z},\vec{x};A,B)
\end{aligned}
$$

Algorithm 3 Forward-Backward algorithm for HMM parameter learning

Initialization: Set A and B as random valid probability matrices
where $A_{i0} = 0$ and $B_{0k} = 0$ for $i = 1..|S|$ and $k = 1..|V|$.
Repeat until convergence {
(E-Step) Run the Forward and Backward algorithms to compute α_i and β_i for $i = 1..|S|$. Then set:

$$\gamma_t(i,j) \quad := \quad \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)$$

(M-Step) Re-estimate the maximum likelihood parameters as:

$$\begin{aligned} A_{ij} \quad &:= \quad \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{j=1}^{|S|} \sum_{t=1}^{T} \gamma_t(i,j)} \\ B_{jk} \quad &:= \quad \frac{\sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\}\, \gamma_t(i,j)}{\sum_{i=1}^{|S|} \sum_{t=1}^{T} \gamma_t(i,j)} \end{aligned}$$

}

$$= \quad \frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)$$

And the denominator of $\hat{B}_{jk}$ as:

$$\begin{aligned} &\sum_{\vec{z}} Q(\vec{z}) \sum_{t=1}^{T} 1\{z_t = s_j\} \\ = \quad &\frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} \sum_{\vec{z}} 1\{z_{t-1} = s_i \wedge z_t = s_j\} P(\vec{z}, \vec{x}; A, B) \\ = \quad &\frac{1}{P(\vec{x}; A, B)} \sum_{i=1}^{|S|} \sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1) \end{aligned}$$

Combining these expressions, we have the following form for our maximum likelihood emission probabilities as:

$$\hat{B}_{jk} \quad = \quad \frac{\sum_{i=1}^{|S|} \sum_{t=1}^{T} 1\{x_t = v_k\} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}{\sum_{i=1}^{|S|} \sum_{t=1}^{T} \alpha_i(t) A_{ij} B_{j\,x_t} \beta_j(t+1)}$$

Algorithm 2.4 shows a variant of the FORWARD-BACKWARD ALGORITHM, or the BAUM-WELCH ALGORITHM for parameter learning in HMMs. In the

E-Step, rather than explicitly evaluating $Q(\vec{z})$ for all $\vec{z} \in S^T$, we compute a sufficient statistics $\gamma_t(i,j) = \alpha_i(t)A_{ij}B_{j\,x_t}\beta_j(t+1)$ that is proportional to the probability of transitioning between sate s_i and s_j at time t given all of our observations $\vec{x}$. The derived expressions for A_{ij} and B_{jk} are intuitively appealing. A_{ij} is computed as the expected number of transitions from s_i to s_j divided by the expected number of appearances of s_i. Similarly, B_{jk} is computed as the expected number of emissions of v_k from s_j divided by the expected number of appearances of s_j.

Like many applications of EM, parameter learning for HMMs is a non-convex problem with many local maxima. EM will converge to a maximum based on its initial parameters, so multiple runs might be in order. Also, it is often important to smooth the probability distributions represented by A and B so that no transition or emission is assigned 0 probability.

2.5 Further reading

There are many good sources for learning about Hidden Markov Models. For applications in NLP, I recommend consulting Jurafsky & Martin's draft second edition of *Speech and Language Processing*[1] or Manning & Schütze's *Foundations of Statistical Natural Language Processing.* Also, Eisner's HMM-in-a-spreadsheet [1] is a light-weight interactive way to play with an HMM that requires only a spreadsheet application.

References

[1] Jason Eisner. An interactive spreadsheet for teaching the forward-backward algorithm. In Dragomir Radev and Chris Brew, editors, *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 10–18, 2002.

[1] http://www.cs.colorado.edu/~martin/slp2.html