11-711: Notes on Hidden Markov Model

Fall 2017

1 Hidden Markov Model

Hidden Markov Model (HMM) is a parameterized distribution for sequences of observations. An intuitive way to explain HMM is to go through an example. Suppose that Taylor hears (a.k.a. observes) a sequence of T sounds $o_1, o_2, ..., o_T$ and he wants to reason something about this sequence. He makes the assumption that the sequence of sounds that he heard depends on a sequence of T words $s_1, s_2, ..., s_T$, which he never gets to see and which is why they are called the hidden states. HMM gives Taylor a method which, under certain assumptions, allows him to assign appropriate probablities to sound sequences O's and word sequences S's and to make reasonable deductions about them. Figure 1 illustrates an HMM with $T = 5$.

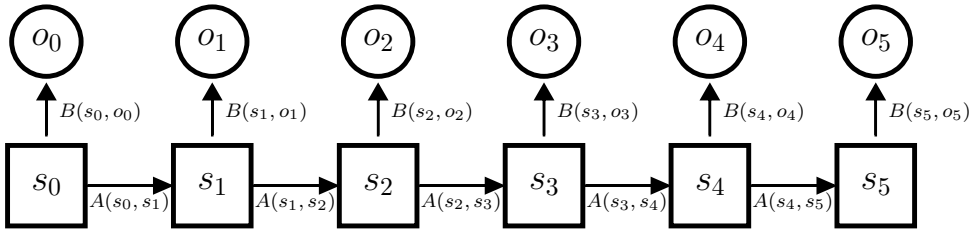

Figure 1: An illustration of a Hidden Markov Model with 5 steps

HMM makes several assumptions about the data it models:

1. The obseverations $o_1, o_2, ..., o_T$ come from a known, finite sets V, called the observation space.
2. The hidden states $s_1, s_2, ..., s_T$ come from a known, finite sets Q, called the hidden state space.
3. The HMM assigns a probability to any given sequence $s_1, s_2, ..., s_T$. The chain rule says that

$$P(s_1, s_2, ..., s_T) = \prod_{t=1}^{T} P(s_t | s_0, s_1, ..., s_{t-1}) = \prod_{t=1}^{T} P(s_t | s_{<t}) \tag{1.1}$$

Here, we are being a little sloppy in terms of notation by assuming that there is a starting state s_0, just so that the indexing is valid. In practice, such state s_0 is sometimes called the starting state, and one can asisgn a probability distribution to it too.
HMM goes further than Equation 1.1 by making the assumption that

$$P(s_t | s_{<t}) = P(s_t | s_{t-1}) \tag{1.2}$$

This is called the **Markov assumption**. With this assumption, Equation 1.1 can be rewritten as

$$P(s_1, s_2, ..., s_T) = \prod_{t=1}^{T} P(s_t|s_{t-1}) \tag{1.3}$$

Note that the Markov assumption generally does not hold. In our example, sequences of words usually have dependencies that go back longer than one immediate step.

4. Each observation o_t only depends on the immediate hidden state s_t. Formally

$$P(o_t|o_{<t}, s_{<t}) = P(o_t|s_t) \tag{1.4}$$

This is called the **independence assumption.**

Under the Markov assumption and the independence assumption, an HMM only needs two set of parameters to model $P(s_t|s_{t-1})$ and $P(o_t|s_t)$. We call these, respectively, the *transition probabilities* and the *emission probabilities.* These probabilities are denoted by matrices A and B, each of whose row is a valid probability distribution, i.e. non-negative numbers that sum up to 1. Table 1 summarizes the components of a general HMM.

Name	Notation	Meaning/Property
State space	$Q = \{q_1, q_2, ..., q_N\}$	Set of N states
Observation space	$V = \{w_1, w_2, ..., w_V\}$	Set of V states
State sequence	$S = \{s_1, s_2, ..., s_T\}$	Sequence of T steps. $s_i \in Q$
Observation sequence	$O = \{o_1, o_2, ..., o_T\}$	Sequence of T steps. $o_i \in V$
Transition probs	$A \in \mathbb{R}^{N \times N}$	Each row is a valid distribution
Emission probs	$B \in \mathbb{R}^{N \times V}$	Each row is a valid distribution

Table 1: Components of a Hidden Markov Model.

There are three important questions regarding HMM.

(1) **Computing the Likelihood.** Given A, B and a sequence of observations O. Compute $P(O|A, B)$.

(2) **Finding the Hidden Sequence.** Given A, B and a sequence of obseveration O. Find the hidden sequence S that is most likely to generate O, that is to find

$$S^* = \underset{S}{\operatorname{argmax}}\, P(O|S, A, B) \tag{1.5}$$

(3) **Estimating the Parameters.** Given a sequence of observation O. Estimate the transition and emission probabilities that are most likely to give O. That is to find

$$A^*, B^* = \underset{A,B}{\operatorname{argmax}}\, P(O|A, B) \tag{1.6}$$

In the following sections, we provide the algorithms to address each of them.

2 Computing the Likelihood: Forward-Pass Algorithm

Given a sequence $O = (o_1, o_2, ..., o_T)$ and known transition and emission probabilities matrices A, B. We want to find $P(O|A, B)$. We use dynammic programming. Specifically, for each step $t = 1, 2, ..., T$ and each hidden state s, we will compute and cache the following value

$$f[t, s] = P(o_1, o_2, ..., o_t, s_t = s|A, B) \tag{2.1}$$

If we know $f[t, s]$, we can compute $P(O|A, B)$ as follows

$$P(o_1, o_2, ..., o_t|A, B) = \sum_s P(o_1, o_2, ..., o_t, s_t = s|A, B) \cdot P(o_t|s_t = s) \tag{2.2}$$

$$= \sum_s f[t, s] \cdot B(s, o_t) \tag{2.3}$$

Now we can compute $f[t, s]$ from previous values $f[t-1, s]$. Specifically

$$f[t, s] = P(o_1, o_2, ..., o_t, s_t = s|A, B) \tag{2.4}$$

$$= \sum_{s'} P(o_1, o_2, ..., o_{t-1}, s_{t-1} = s'|A, B) \cdot P(s_t = s|s_{t-1} = s')P(o_t|s) \tag{2.5}$$

$$= \sum_{s'} f[t-1, s'] \cdot A(s', s)B(s, o_t) \tag{2.6}$$

Putting together Equation 2.3 and Equation 2.6, we have the so-called *Forward-Pass* algorithm to compute the likelihood in an HMM.

(1) Initialize:

- For each hidden state s:

$$f[1, s] = P(o_1, s_1 = s|A, B) \leftarrow B(s, o_1) \cdot A(s_0, s) \tag{2.7}$$

(2) For $t = 2$ to T:

- For each hidden state s:

$$f[t, s] \leftarrow \sum_{s'} f[t-1, s'] \cdot A(s', s)B(s, o_t) \tag{2.8}$$

(3) Finally:

$$P(o_1, o_2, ..., o_T|A, B) \leftarrow \sum_s f[T, s]B(s, o_T) \tag{2.9}$$

Its complexity is $O(N^2 \cdot T)$ where N is the size of the hidden state space.

3 Finding the Hidden Sequence: Viterbi Algorithm

Given a sequence $O = (o_1, o_2, ..., o_T)$ and known transition and emission probabilities matrices A, B. For each hidden sequence S, the joint probability that both O and S happen is

$$P(O, S|A, B) = P(O|S, A, B) \cdot P(S|A, B) \tag{3.1}$$

$$= \prod_{i=1}^{T} P(o_t|s_t) \cdot \prod_{i=1}^{T} P(s_t|s_{t-1}) \tag{3.2}$$

$$= \prod_{i=1}^{T} B(s_t, o_t) \cdot \prod_{i=1}^{T} A(s_{t-1}, s_t) \tag{3.3}$$

We want to find the sequence S that maximizes the quantity above, that is

$$S^* = \operatorname*{argmax}_S P(O, S|A, B) \tag{3.4}$$

It turns out that a dynamic programming programming algorithm that is almost identical to the Forward-Pass algorithm in Section 2 is applicable for the task of finding S in Equation 3.4. That is the Viterbi algorithm [Viterbi, 1967].

We define

$$g[t, s] \stackrel{\text{def}}{=} \max_{s_1 \ldots s_{t-1}} P(o_1, o_2, ..., o_t, s_t = s|A, B) \tag{3.5}$$

The key observation is that if we know all values of $g[t, s]$, then we can compute the maximum joint probability $P(O, S|A, B)$ as follows

$$\max_{s_1,\ldots,s_t} P(o_1, o_2, ..., o_t|A, B) = \max_s P(o_1, o_2, ..., o_t, s_t = s|A, B) \cdot P(o_t|s_t = s) \tag{3.6}$$

$$= \max_s g[t, s] \cdot B(s, o_t) \tag{3.7}$$

Please note the similarity between Equations 3.6-3.7 to Equations 2.2-2.3, where the only difference is the change from summation to max. The reason that Viterbi's algorithm is almost identical to Forward-Pass algorithm is that max has the same distributing property as summation, i.e. one can "factor" our the same term.

We can compute $g[t, s]$ similar to the way we compute $f[t, s]$ in Equation 2.6. Specifically

$$g[t, s] = \max_{s_1,\ldots,s_t} P(o_1, o_2, ..., o_{t-1}, o_t, s_t = s|A, B) \tag{3.8}$$

$$= \max_{s'} P(o_1, o_2, ..., o_t, s_{t-1} = s'|A, B) \cdot P(s_t = s|s_{t-1} = s')P(o_t|s) \tag{3.9}$$

$$= \max_{s'} g[t-1, s'] \cdot A(s', s)B(s, o_t) \tag{3.10}$$

Finally, note that in Equation 3.4, we wanted an $\operatorname{argmax}_S P(O, S|A, B)$, not the $\max_S P(O, S|A, B)$. The general idea to go from $\max(\cdot)$ to $\operatorname{argmax}(\cdot)$ in a dynamic programming algorithm is to create a trace-back array. In Equation 3.10, whenever you know which s' leads to the largest value for $g[t-1, s'] \cdot A(s', s)B(s, o_t)$, you can store that s' in an array $h[t, s]$. After you finish with $g[T, s]$ as in Equation 3.7, you can simply follow $h[t, s]$ to trace back the sequence of hidden states that leads to the maximum value of $P(O, S|A, B)$, which gives you the argmax. Putting everything together, we have the Viterbi's algorithm

(1) Initialize:

- For each hidden state s:

$$g[1, s] \leftarrow B(s, o_1) \cdot A(s_0, s) \tag{3.11}$$

(2) For $t = 2$ to T:

- For each hidden state s:

$$g[t, s] \leftarrow \max_{s'} g[t-1, s'] \cdot A(s', s)B(s, o_t) \tag{3.12}$$

$$h[t, s] \leftarrow \operatorname*{argmax}_{s'} g[t-1, s'] \cdot A(s', s)B(s, o_t) \tag{3.13}$$

(3) Follow $h[t,s]$ to find s_T^*, s_{T-1}^*, ..., s_1^*. Starting at $t=T$

$$s_T^* \leftarrow \operatorname*{argmax}_s g[T,s] \tag{3.14}$$

$$s_t^* \leftarrow h[t+1, s_{t+1}^*] \qquad \text{for } t = T-1, T-2, ..., 1 \tag{3.15}$$

Viterbi's algorithm also has the complexity of $O(N^2 \cdot T)$.

4 Estimating the Parameters: Baum-Welch Algorithm

In Section 2 and Section 3, we have seen that if we know the transition probabilities $A(s,s')$ and the emission probabilities $B(s,o)$, then we can efficiently perform the queries on an HMM. In this section, we are concerned with the problem of estimating A and B, given a single sequence of obseverations $O=(o_1,o_2,...,o_T)$.

The algorithm for this task is the Baum-Welch algorithm [Baum, 1972], an instance of a procedure called Expectaction-Maximization (EM) [Dempster et al., 1977]. In Baum-Welch algorithm, one starts by randomly initializing the values for A and B, then repeatedly updating them. Each updating iteration has two steps, the Expectation step (E-step) and the Maximization step (M-step).

E-step. The E-step assumes that we know A, B and computes the following quantities

$$\gamma[t,s] = P(s_t = s|O,A,B) \tag{4.1}$$

$$\xi[t,s',s] = P(s_{t-1}=s', s_t = s|O,A,B) \tag{4.2}$$

Intuitively, $\gamma[t,s]$ counts how many times does the t^{th} states of the hidden sequence equals s and $\xi[t,s,s']$ counts how many times the duo (s',s) happens at the $(t-1)^{st}$ step and the t^{th} step in the hidden sequence, both up to normalization by a partition function. That is why $\gamma[t,s]$ and $\xi[t,s,s']$ are sometimes referred to as the *pseudo counts.*

We now show how to compute $\gamma[t,s]$ and $\xi[t,s',s]$. Since $\gamma[t,s]$ and $\xi[t,s',s]$ are probabilities conditioned on O, we compute via the partial joint probabilities

$$\alpha[t,s] = P(o_1,o_2,...,o_t,s_t=s|A,B) \tag{4.3}$$

$$\beta[t,s] = P(o_{t+1},o_{t+2},...,o_T|s_t=s,A,B) \tag{4.4}$$

If one knows all $\alpha[t,s]$ and $\beta[t,s]$, then $\gamma[t,s]$ and $\xi[t,s',s']$ can be computed as follows

$$\gamma[t,s] = P(s_t=s|O,A,B) = \frac{P(s_t=s,O|A,B)}{P(O|A,B)} \tag{4.5}$$

$$= \frac{\alpha[t,s]\cdot\beta[t,s]}{\sum_{s'}\alpha[t,s']\cdot\beta[t,s']} \tag{4.6}$$

$$\xi[t,s',s] = P(s_{t-1}=s', s_t=s|O,A,B) = \frac{P(s_{t-1}=s', s_t=s, O|A,B)}{P(O|A,B)} \tag{4.7}$$

$$= \frac{\alpha[t-1,s]\cdot A(s',s)\cdot\beta[t,s]}{\sum_{s'}\alpha[t,s']\cdot\beta[t,s']} \tag{4.8}$$

It only remains how to compute $\alpha[t,s]$ and $\beta[t,s]$. Not surprisingly, they can be computed with dynamic programming

$$\alpha[t,s] = \sum_{s'}\alpha[t-1,s']\cdot A(s',s)\cdot B(s,o_t) = B(s,o_t)\sum_{s'}\alpha[t-1,s']\cdot A(s',s) \tag{4.9}$$

$$\beta[t, s] = \sum_{s'} \beta[t+1, s'] \cdot A(s, s') \cdot B(s', o_{t+1}) \tag{4.10}$$

M-step. When these pseudo counts γ and ξ are known, then in the M-step, one can use maximum likelihood estimator to derive the updated values for A and B

$$\hat{A}(s', s) = \frac{\#\text{times } s \text{ follows } s'}{\#\text{times anything follows } s'} = \frac{\sum_{t=1}^{T-1} \xi[t, s', s]}{\sum_{s''} \sum_{t=1}^{T-1} \xi[t, s', s'']} \tag{4.11}$$

$$\hat{B}(s, o) = \frac{\#\text{times } o \text{ is observed given } s}{\#\text{times anything is observed given } s} = \frac{\sum_{t=1}^{T} \mathbf{1}[o_t = o]\gamma[t, s]}{\sum_{t=1}^{T} \gamma[t, s]} \tag{4.12}$$

References

L. E. Baum. An inequality and associated maximization in statistical estimation for probabilistic functions of markov processes. In *Proceedings of the 3rd Symposium on Inequalities, University of California, Los Angeles*, 1972.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society*, 1977.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transactions on Information Theory*, 1967.