6.858/18.428 Machine Learning	**Lecture 25: Decmeber 12, 1994**
Lecturer: Ron Rivest	*Scribe: Jesse Hull*

25.1 Outline

- Learning Hidden Markov Models
- Application to speech recognition

25.2 Description of Hidden Markov Models

In this lecture we describe Hidden Markov Models and give a method for learning them. Their application to speech recognition is also discussed.

Definition 1 *A Hidden Markov Model is a 5-tuple* $\{Q, V, \pi(i), A, B\}$ *where*

Q = the set of states = $\{q_1, q_2, ..., q_n\}$

V = the output alphabet = $\{v_1, v_2, ..., v_m\}$

$\pi(i)$ = probability of being in state q_i at time $t = 0$

A = transition probabilities = $\{a_{ij}\}$,
where $a_{ij} = Pr[\text{entering state } q_j \text{ at time } t+1 \mid \text{in state } q_i \text{ at time } t]$

B = output probabilites = $\{b_j(k)\}$,
where $b_j(k) = Pr[\text{producing } v_k \text{ at time } t \mid \text{in state } q_j \text{ at time } t]$

25.3 Example of a Hidden Markov Model

If we let $\pi(q_1) = 1$ and $\pi(q_2) = 0$ then the following is a example of a possible transition sequence and output sequence for the Hidden Markov Model in figure 25.1:

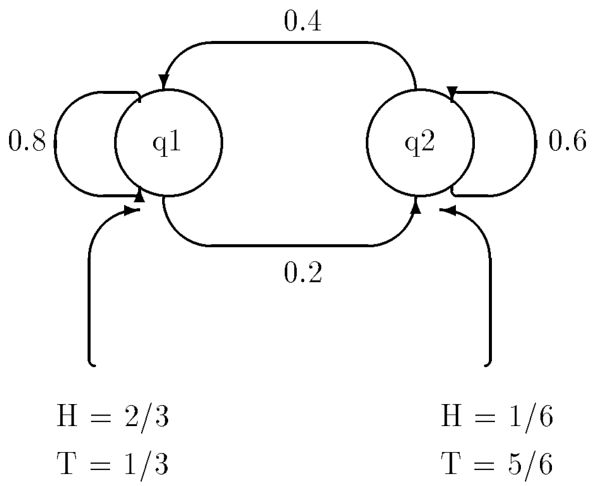

Figure 25.1: A Hidden Markov Model

$$\begin{array}{ccccccccccccc} q_1 & \xrightarrow{0.8} & q_1 & \xrightarrow{0.8} & q_1 & \xrightarrow{0.2} & q_2 & \xrightarrow{0.6} & q_2 & \xrightarrow{0.4} & q_1 & \xrightarrow{0.8} & q_1 \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ \mathrm{H} & & \mathrm{H} & & \mathrm{T} & & \mathrm{T} & & \mathrm{T} & & \mathrm{T} & & \mathrm{H} \end{array}$$

We can easily calculate probabilities for the following events.

1. The probability of the above state transition sequence:

$$Pr[q_1q_1q_1q_2q_2q_1q_1] = \pi(q_1)a_{11}a_{11}a_{12}a_{22}a_{21}a_{11} \approx 0.025$$

2. The probability of the above output sequence given the above transition sequence:

$$Pr[(HHTTTTH)|(q_1q_1q_1q_2q_2q_1q_1)] = \frac{2}{3}\frac{2}{3}\frac{1}{3}\frac{5}{6}\frac{5}{6}\frac{1}{3}\frac{2}{3} \approx 0.23$$

3. The probability of the above output sequence and the above transition sequence:

$$Pr[(HHTTTTH) \wedge (q_1q_1q_1q_2q_2q_1q_1)] \approx (0.025)(0.023) \approx 5.7x10^{-4}$$

25.4 Using Hidden Markov Models to generate speech

Markov models can be used to generate speech by making the output symbols phonemes of speech. A Hidden Markov Model might then capture the pronounciation of a word. By changing the structure and transition probabilies in the Markov model one can modify the legal pronounciation of a word.

25.5 Using Markov Models to recognize words

Suppose one has a set of Hidden Markov Models for a set of words. Then given a speech signal one could ask which model was the most likely to have generated that signal. We need to be able to calculate the probability of a given output sequence for a Hidden Markov Model. So, given a output sequence $O_1O_2...O_T$ we have the following equation:

$$Pr[O_1O_2...O_T] = \sum_{i_1,i_2,...,i_T} Pr[i_1,i_2,...,i_T]Pr[O_1O_2...O_T|i_1,i_2,...,i_T]$$

This is simply the probability of a given state transition sequence times the probability of the output sequence given that transition sequence, summed over all possible state transtions. We can use dynamic programming to calculate this probability efficiently.

Definition 2 *Let* $\alpha_t(i) = Pr[(O_1O_2...O_t) \wedge (i_t = q_i)]$

Then we can use the following formulas to calculate $\alpha_t(i)$ where N is the number of states in the Hidden Markov Model.

$$\begin{aligned}
\alpha_1(i) &= \pi(i)b_i(O_1) \\
\alpha_{t+1}(i) &= \left(\sum_{j=1}^{N} \alpha_t(j)a_{ji}\right) b_i(O_{t+1})
\end{aligned}$$

From these equations it should be clear that $\alpha_T(i)$ can be computed in $O(N^2T)$ time. Furthermore since $\alpha_T(i)$ is the probability that we get the correct output and end in state q_i, the probability that we get the correct output is simply the sum of the $\alpha_T(i)$'s. Namely,

$$Pr[O_1 O_2 ... O_T] = \sum_{i=1}^{N} \alpha_T(i)$$

Hence we can calculate $Pr[O_1 O_2 ... O_T]$ in $O(N^2 T)$ time.

25.6 Using Markov Models to recognize sentences

Now suppose that we not only have models for recognizing words but we have a model that tells us how words should be put together. Namely we have encoded a gramatical model of speech into our Hidden Markov Model. In this case we may ask two questions:

1. For each t, what state is most likely? (Note that the sequence of states computed by this criterion might be impossible.)
2. What single sequence of states has the largest posterior probability?

To answer the first question we define $\beta_t(i)$, which is essentially the time reversal of $\alpha_t(i)$, and we define $\gamma_t(i)$, which is the probability of being in state q_i at time t given the observation sequence.

Definition 3 *Let* $\beta_t(i) = Pr[(O_{t+1} O_{t+2} ... O_T)|(\text{at time } t \text{ in state } q_i)]$

Definition 4 *Let* $\gamma_t(i) = Pr[i_t = q_i | O_1, O_2, \ldots, O_T]$

Then we have the following formulas for $\beta_t(i)$ and $\gamma_t(i)$:

$$\begin{aligned}
\beta_T(i) &= 1 \\
\beta_t(i) &= \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(O_{t+1}) \\
\gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}
\end{aligned}$$

Now we have the answer to question 1: for time t, the state q_i which maximizes $\gamma_t(i)$ is the state with highest probability. To answer question 2, we make one further definition.

Definition 5 *Let* $\delta_t(i) = Pr$[that the most likely single sequence of states ends with state q_i, given observed output]

And the equations for $\delta_t(i)$ are

$$\begin{aligned} \delta_1(i) &= \pi(i) b_i(O_1) \\ \delta_{t+1}(i) &= \left(\max_{j \in [1,N]} \delta_t(j) a_{ji} \right) b_i(O_{t+1}) \end{aligned}$$

Backtracking the calculation $\delta_T(i)$ will allow you to calculate the answer to question 2. This method is called the Viterbi Algorithm.

25.7 Learning Hidden Markov Models

We have shown that given a Hidden Markov Model we can

1. produce output
2. see which sequence of states produces the best fit

Now we need to learn how to build a model. We have to find a method for calculating a_{ij}, b_i, and $\pi(i)$. The algorithm that we present is similiar to the back-propagation algorithm of neural nets in that it is susceptible to local minima and needs lots of data to converge. It does, however, have the advantage that in every step of the algorithm it improves (or at least keeps constant) its accuracy on the test data.

The Baum/Welch training algorithm works as follows: It is given $O_1 O_2 ... O_T$ and initial estimates for parameters a_{ij}, b_i, and $\pi(i)$. We define new parameters $\bar{a}_{ij}$, $\bar{b}_i$, and $\bar{\pi}(i)$. First we define $\Upsilon_t(i,j)$, the estimator of the probabilty of a transition from q_i to q_j at time t given output $O_1 O_2 ... O_T$.

$$\Upsilon_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{Pr[O_1 O_2 ... O_T]}$$

then $\bar{\pi}(i)$, $\bar{a}_{ij}$, and $\bar{b}_i$ are calculated as:

$$\begin{aligned}
\bar{\pi}(i) &= \gamma_1(i) \\
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \Upsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
&= \frac{\text{expected number of transitions from state } q_i \text{to } q_j}{\text{expected number of transitions out of state } q_i} \\
\bar{b}_j(k) &= \frac{\sum_{t=1;O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \\
&= \frac{\text{expected number of times in state } q_j \text{ with observed output } v_k}{\text{expected number of times of being in state } q_j}
\end{aligned}$$

These values $\bar{\pi}(i)$, $\bar{a}_{ij}$, and $\bar{b}_i$ are used for our estimates of a_{ij}, b_i, and $\pi(i)$ for the next iteration. Using this algorithm we can learn Hidden Markov Models. Baum and Welch showed that every iteration of this algorithm will not make the parameters worse. However, the algorithm must still be given reasonable initial parameters. Furthermore, the algorithm says nothing about how many states our Hidden Markov Model should have or how they should be connected.

References

[1] Kai-Fu Lee. Automatic peec recognition. In *Computer Science Research Review*, pages 7-16. School of Computer Science, Carnegie Mellon University, 1989.

[2] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition. In *Bell System Technical Journal*, 62(4):1035-1074, April 1983.

[3] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magzine*, 3(1):4-16, January 1986.