The Hidden Markov Model and Applications in Machine Learning

Joseph Tumulty
Term Paper for Physics 502
Drexel University

Submitted: 3/15/16

1. Motivation

The motivation of this paper is to explore and understand the concept of Hidden Markov Models. These models have been applied to speech and handwriting recognition software and may provide insight into the learning mechanisms in biological neural networks. The idea of a neural network is a complex problem in neuroscience but actually presents a solution to many computational problems. By using the model of the network present in the brain, artificial neural networks can be created and used to solve computational problems. The theoretical framework of these problems can act as a simulation that can give insight into the mechanisms of biological neural networks by comparisons of results with experimental data. Although the focus of artificial neural networks in computing is not primarily focused on the inner workings of the brain, there may be a possibility to apply the work to computational neuroscience research.

The Hidden Markov Model is also a very robust method for determining the nature of any input signal given an output. By the nature of the problem, the model determines the most probable set of parameters dictating input states based on the sequence of output states. This can be applied to many problems where the nature of the input is known but not the specific parameters dictating its behavior. An example of this would be in the sequencing of DNA [1]. There are hidden states (e.g. CpG island or non-CpG island) that change the probability of generating a certain sequence of nucleotides. The mechanisms of these hidden states are not fully understood and so by using the output (nucleotide sequence, ACTGCG…) and a Hidden Markov Model, we can determine the most probable parameters associated with the hidden states.

2. Background

To understand the idea of Hidden Markov Models (HMMs) we start first with the Markov model of a network (this was discussed in class but I will include this for continuity). In the Markov model each node of the network is connected to other nodes with some probability of transition from one node to another. The probability of the system being at the node is dependent only on the previous state. If we label the state where our system is at node 1 as S_1 and let our state at any time t be denoted as q_t then our probability of being at any state j can be denoted by,

$$P(q_t = S_j | q_{t-1} = S_i) = a_{ij},$$

where we have defined a_{ij} to be the transition probability from state S_i to state S_j. These transition probabilities can be represented in a Markov probability matrix with many useful properties. The matrix,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix},$$

has the property that each row sums to 1 because each row represents the probability of any one state i transitioning to any other state j (Note: This convention is swapped from what was discussed in class). It also is useful in calculating the probabilities of being in some state at a later time t (t can be thought of as the number of steps, where there is a state transition at each step). To determine this probability we just raise the matrix to the power of the number of steps, t. So, the probability of being in state, S_j, after t steps having started in state S_i is,

$$P(q_t = S_j | q_1 = S_i) = (A^t)_{ij}$$

where the A^t indicates the matrix raised to the tth power (not the transpose).

Although this is a useful property, we will find that when we transition into the Hidden Markov Model it will be more beneficial to consider a specific sequence of states. In the case of this simple Markov model, this sequence of states is our output, O.

$$O = \{S_3, S_2, S_3, S_1, \ldots\}$$

The probability of this sequence is given by the product of transition probabilities in the sequence,

$$P(O|A) = \pi_3 \cdot a_{32} \cdot a_{23} \cdot a_{31} \cdot \ldots$$

where π_i has been introduced as the probability of starting in the state S_i. The above description of a simple Markov process is not complete but sufficient for transitioning to HMMs. Most importantly it introduced the variables and framework where we will be working.

3. The Hidden Markov Model

The transition to the Hidden Markov Model consists mainly of generating another output from each state; each output having its own probability of occurring. The classic example given is that of a genie in a room with a number of urns. These urns each contain a certain number of balls of different colors. The genie chooses an urn based on some random process. This determines the state (if the output were the urn this would be analogous to the simple Markov process outlined above). The genie then selects a ball from that urn which has a certain probability of being chosen based on the urn. This colored ball is then the output and it is the only thing that the observer sees. The urn on the other hand is hidden, hence the name. The transition probabilities between states (urns) is given by the familiar matrix, A. The probabilities of outputs (color of ball) are given in $B = \{b_j(k)\}$, where

$$b_j(k) = P(v_k \; at \; t | q_t = S_j),$$

is the probability of output v_k given state S_j.

There are three main questions presented in the analysis of a Hidden Markov Model. They are the following:

1. What is the probability of getting a certain observation sequence given a model? Here the model is given by the parameters $\lambda = (A, B, \pi)$
2. What is the most probable state sequence given a certain observation sequence and model?
3. Finally, and most importantly, what is the best set of model parameters $\lambda = (A, B, \pi)$ for a given observation sequence (output)?

The solutions to these problems as well as the source for this notation convention was first presented by L. E. Baum and are compiled and well presented in a paper by L. R. Rabiner [2]. I will summarize the results below.

For the first problem, we could "brute force" it by: generating every possible sequence of states, calculating the probability of generating the given output for each state sequence, and summing the resulting probabilities for all sequences. Computationally this turns out to be unrealistic. The number of computations is on the order of $2T \cdot N^T$. For any reasonably sized network or process this is too large. Instead, we use "forward" and "backward" variables. These are defined respectively as

$$\alpha_t(i) = P(O_1 O_2 O_3 \dots O_t, q_t = S_i | \lambda) \qquad \text{(forward)}$$

$$\beta_t(i) = P(O_{t+1} O_{t+2} \dots O_T | q_t = S_i, \lambda) \qquad \text{(backward)}$$

They can both be used to find a solution to problem 1 but we will focus on the forward variable for simplicity and brevity. The interpretation of the variable is the probability of a certain observation sequence, up until time t, and a state S_i, at time t, given a certain model. To find the total probability of this sequence up until time T for all state sequences, an inductive algorithm is used:

1) $\alpha_1(i) = \pi_i b_i(O_1)$

2) $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \qquad 1 \le t \le T-1$

3) $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

This induction initiates with a state i at $t = 1$ (this is calculated for all states i). Then, at each subsequent t the probabilities of each state are calculated by summing over each probability up until that point, $\alpha_t(i)$, multiplied by the probability of transition to the new state, j. The probability of the output, $b_j(O_{t+1})$, is also included for each new α because we are ultimately looking for the probability of a certain observation sequence. The last step is simply summing over probabilities for every final state i. This tells us the probability of getting an observation sequence given a certain model λ, and it does so with an order of $N^2 T$ calculations.

For the second problem we are looking for the most probable state sequence. One possibility for finding this would be to individually find the most probable state for each observation. This would be valid in a sense but neglects the fact that that states have their own transition probability and this method could generate a state sequence which is either highly unlikely or impossible (in the case where some element of A is 0). Instead of maximizing the probability of each state given an observation, we maximize the overall probability of the state sequence given the observation sequence, i.e. maximize $P(Q|O,\lambda)$. This turns out to be equivalent to maximizing $P(Q,O|\lambda)$. To maximize this probability we invoke the Viterbi Algorithm introduced by A. Viterbi in 1967 [3] and summarized in the paper by Rabiner [2].

To do this we introduce the "best path" probability (named by me but created by others), $\delta_t(i)$, which represents the maximum probability for a path (including the observation sequence up to time t) that ends up in state i. It can be represented as:

$$\delta_t(i) = \max P(q_1 q_2 \dots q_t = i, O_1 O_2 \dots O_t|\lambda)$$

where the "max" is over different sequences $q_1 q_2 \dots q_t$. This variable only tells us the max probability. Therefore, to keep track of the actual sequence we introduce the variable $\psi_t(j)$. This captures the state, i, which maximizes the probability at the transition $i \longrightarrow j$. We can the use these in an inductive algorithm as follows:

1) $\delta_1(i) = \pi_i b_i(O_1) \qquad 1 \le i \le N$

   $\psi_1(i) = 0$

2) $\delta_t(j) = \max_i \left[\delta_{t-1}(i) a_{ij}\right] b_j(O_t) \qquad 2 \le t \le T$

   $\psi_t(j) = \operatorname{argmax}_i \left[\delta_{t-1}(i) a_{ij}\right] \qquad 1 \le j \le N$

3) $P^* = \max_i \left[\delta_T(i)\right]$

   $q_T^* = \operatorname{argmax}_i \left[\delta_T(i)\right]$

4) $q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \dots, 1$

The first two steps of the above algorithm are very similar to the algorithm presented for problem 1 with the added aspect of finding a maximum instead of summing over terms. There is also the added variable that keeps track of the argument that maximizes the probability. The third step simply terminates the recursion and the fourth step backtracks over the arguments, reassigning them to the correct time since each maximizing argument was placed in the index corresponding to the next time, t.

The presented solution to the final problem is the least intuitive but winds up being a sound method. As a reminder, we are now looking for the set of parameters that maximizes the probability of a certain sequence of observations. The one set-back to the method is that it is an iterative search around the neighborhood of the initial parameters. This means it will find local maxima in probability but will not be able to find the global maximum. This leaves room for error when looking for a specific set of parameters but it is effective at finding a characteristic set of parameters or finding the global maximum if the user as a good guess for starting parameters. The method makes use of the forward and backward variables defined earlier, and requires the definition of two new variables. The first new variable is $\gamma_t(i)$. It is the probability of being in a state, S_i at time t given a certain observation sequence and model. It can be represented in terms of the forward and backward variables as:

$$\gamma_t(i) = \frac{\alpha_t(i)\,\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\,\beta_t(i)}$$

This can be reached logically by examining the fact that the forward and backward variables compute the probability up to a point i and after a point i, respectively. The denominator makes it a probability measure that sums to 1.

The other new variable is $\xi_t(i,j)$. It is the probability of being in state S_i at time t and state S_j at time $t+1$. It can also be represented in terms of the forward and backward variables as:

$$\xi_t(i,j) = \frac{\alpha_t(i)\,a_{ij}b_{j+1}(O_{t+1})\,\beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\,a_{ij}b_{j+1}(O_{t+1})\,\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\,a_{ij}b_{j+1}(O_{t+1})\,\beta_{t+1}(j)}$$

The numerator can be logically derived from the fact that the forward variable gives us the first condition, state S_i at time t, and the transition probability, observation probability and backward variable give us the second condition, state S_j at time $t+1$. The denominator as before is a normalizing condition.

Next we examine the respective sums over time from $t = 1$ to $t = T - 1$. For γ this gives us the expected number of transitions from the state S_i. For ξ, it gives us the expected number of transitions from state S_i to state S_j. These sums can then be used to calculate new parameters for our model given a starting configuration, $\lambda = (A, B, \pi)$

First, the parameter π_i is a measure of the probability of starting in a certain state. Therefore we can use the measure $\gamma_1(i)$ to determine a better value $\bar{\pi}_i = \gamma_1(i)$. Secondly we can find a better value for the transition probability between i and j using the sums previously mentioned.

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

This is interpreted as the expected number of transitions from $i \longrightarrow j$ divided by the expected number of transitions from i. Finally we refine our observation probabilities using the expected number of times in state j and observing v_k divided by the expected number of times in state j.

$$\bar{b}_j(k) = \frac{\sum^{T}{}_{\substack{t=1 \\ O_t = v_k}} \; \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

It has been shown that, with every iteration, either the λ is already at a maximum and the values remain unchanged or the probability of the new values increases.

This method answers all three questions presented in Hidden Markov Models. Of the three problems the second and third tell us what we would like to know about the underlying mechanisms creating our output. They tell us the most probably sequence of states as well as the parameters relating those states to each other and to the output. The solution to the first problem acts as a check for a certain set of parameters. Once we find what we believe to be the best set, we can then compute the probability of outputting our original observation sequence and compare it with an initial set of parameters. If the probability is higher we know that we have at least moved towards a local maximum in the parameter space.

4. Applications and Conclusions

As mentioned earlier this method has been used to determine underlying properties of DNA sequencing mechanisms. It has also been very widely used in speech and handwriting recognition. That is because the adjustments of parameters provide an opportunity for a network to "learn" when given training. That is, the programmer does not need to come up with an extremely complicated program which will recognize any letter in any sloppy handwriting. The programmer instead implements one of the higher order (continuous) forms of the Hidden Markov Model and the hidden states are adjusted through the learning process. This basically allows the program to create its own network that will recognize handwriting or speech even with distortions or perturbations from what it has seen in training.

A possible application of this method to computational neuroscience is clear. It provides a method for allowing an artificial network to learn given appropriate training. If we are able to apply a similar method to a neural network which more closely mirrors a biological network it would bring to light possible mechanisms in the formation of biological neural networks. We may be able to uncover the way in which certain neuronal connections form, if only on a statistical level, in the process of learning.

References:

[1] Hao Wu, Brian Caffo, Harris A. Jaffee, Rafael A. Irizarry, and Andrew P. Feinberg. Redefining CpG islands using hidden Markov models. Biostat (2010) first published online March 8, 2010 doi:10.1093/biostatistics/kxq005

[2] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, Feb 1989.

[3] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," in *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260-269, April 1967.

Other Sources

Wai-Ki Ching, Michael K. Ng. 2006. Markov Chains: Models, Algorithms, and Applications. New York, NY: Springer Science+Business Media, Inc.

Robert Gilmore. 2016. Phys 501 Class Notes. Philadelphia, PA: Joseph Tumulty, inc.