Hidden Markov Models

Christopher Yuan

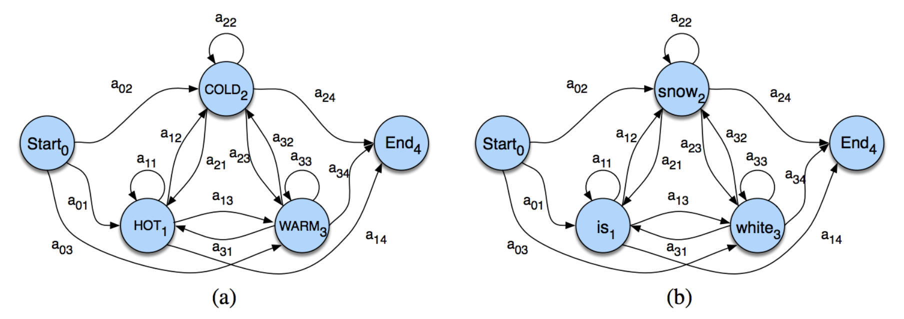
May 2018

1 Introduction

Hidden Markov Models (HMMs) are a type of statistical Markov model used to characterize the observed samples of a discrete-time series. Descendants of the simpler Markov chain, they allow us to talk about both the hidden and observable events in a stochastic model. The basic theory of HMMs was published around 1967 by Baum et al. and are often used to model temporal and sequential data. From their flexibility and computational efficiency, they have found wide application in speech recognition, speech synthesis, music annotation, biological sequence analysis, part-of-speech tagging.

2 Overview of Markovian Systems

The idea behind HMMs is largely derived from simple Markovian systems, namely the Markov Chain. Also called an Observable Markov model, it involves a sequence of states, $S = \{s_1, s_2, ..., s_r\}$ in which the process starts from the first state and moves successively from one state to another. The probability of one state occurring is solely based upon the previous state. If the chain is currently in state s_i, then it moves onto state s_j at the next step. The probability of state s_i giving rise to a specific state s_j is denoted by a_{ij}.



In the figure above, part (a) shows the structure of a Markov Chain describing the probability of a series of weather events, and part (b) one for words. The probability of state s_i giving rise to a specific state s_j is denoted by a_{ij}. These probabilities are known as transition probabilities and are elements of the transition probability matrix A. Note that A is row stochastic, meaning that all the probabilities in each row sum to one:

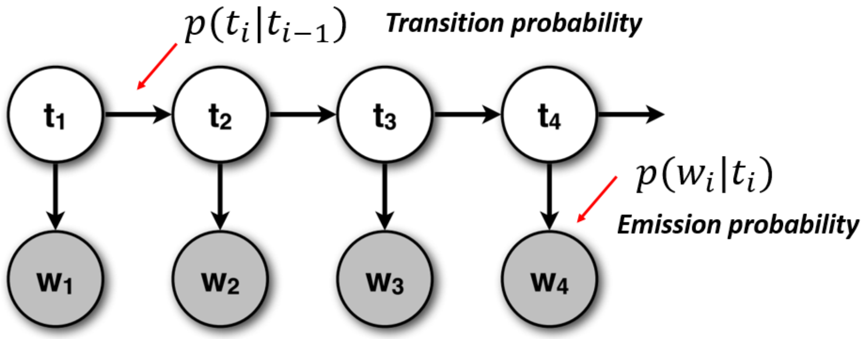
$$\sum_{j=1}^{n} a_{ij} = 1 \forall i$$

Because each state has dependency only on the single state before it, it is called a single order Markovian system, as opposed to one that is dependent on multiple past states. This is why while a Markov Chain is able to emulate the writing style of an author using the probability of a word onto the next, it is essentially memoryless, unable to determine meaning or context.

3 HMM

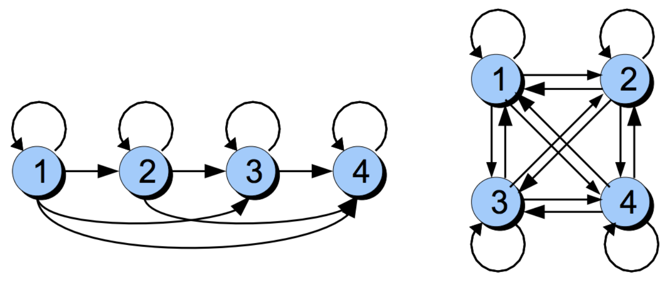
The idea behind the HMM is similar to a Markov Chain in a structural sense, but we are unable to observe the states. Hence, they are hidden. However, upon visiting a state at a time-step, an observation is recorded that is a probabilistic function of the state. For example, when visiting state s_t, we are able to discern a discrete observation o_t from a set of possible observations $\{v_1, v_2, ..., v_M\}$. While we talked about transition probabilities in the previous section, HMMs introduce a new set of parameters, known as emission probabilities, which express the probability of observation o_t being generated from state s_i, denoted by $b_i(o_t)$. Like Markov Chains, the probability of a specific state depends only on the state from the previous timestep. Moreover, the probability of an output observation o_i depends only on the state that produced the observation and not on any other states or any other observations.

Below is a basic left-to-right Hidden Markov Model, with states $Q = t_1 t_2 ... t_N$ and observations $O = w_1 w_2 ... w_N$



HMMs can come in a variety of different structural models. An HMM in which there all transition probabilities are non-zero (meaning it is possible for one state to transition into any other state), is known as a fully-connected or ergodic HMM. In a left-to-right, or Bakis HMM, states transition only from left to right, so no higher numbered state can proceed to a lower numbered state. Bakis HMMs are usually used for modeling temporal operations like speech.

Here are two 4-state hidden Markov models; a left-to-right (Bakis) HMM on the left and a fully connected (ergodic) HMM on the right.



Overall, the uses of Hidden Markov Models can be characterized by three fundamental problems:

3.1 Likelihood

If we are given an observation sequence $O = (o_1 o_2 \dots o_T)$, and a model $\lambda = (A, B)$, how do we determine P(O|λ), the likelihood of the observation sequence occurring given the model?

3.2 Matching/Decoding

If we are given an observation sequence $O = (o_1 o_2 \dots o_T)$, and a model λ, what is the optimal hidden sequence of states that correspond to the observation sequence (i.e. makes the most sense)?

3.3 Training

If we are given an observation sequence $O = (o_1 o_2 \ldots o_T)$, how do we adjust the parameters of an HMM model λ to maximize P(O|λ)?

4 Likelihood: The Forward Algorithm

Recall from Section 3.1 that we are given $O = (o_1 o_2 \ldots o_T)$, and a model λ. The most straightforward answer to find the likelihood of a specific observation sequence is to iterate through the entire state sequence of length T. With a possible state sequence $Q = (q_1 q_2 \ldots q_T)$:

$$\mathcal{P}(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) \tag{1}$$

$$= b_{q_1}(o_1) b_{q_2}(o_2) ... b_{q_T}(o_T) \tag{2}$$

But of course, we don't know what the hidden sequence of states actually is. So the probability of such a sequence Q is:

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} a_{q_2 q_3} ... a_{q_{T-1} q_T}$$

Here the notation π_{q_1} refers to the starting probability of the first state occurring. Multiplying the two, we end up with:

$$P(O \mid \lambda) = \sum_{allQ} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

In the end, the brute force method of solving this problem with a state sequence of length T requires $2T * N^T$ calculations, where T is the number of observations in O and N is the number possible hidden states. Obviously, this is computationally unfeasible.

So here is where the Forward Algorithm comes in. Instead of some exponential algorithm, it is $O(N^2 T)$ efficiency. The idea behind the algorithm is basically that we divide the observation sequence into parts: the first one going from time 1 until time t, and the second one from time t+1 to T. Let us define the forward variable $\alpha_t(i)$ as the probability of observing the first partial observation sequence $o_1 ... o_t$ until time t and being in a specific state s_i at time t, given model λ:

$$\alpha_t(i) \equiv P(o_i ... o_t, q_t = s_i \mid \lambda)$$

Given the 1st-order Markov property, the nice thing about this is that it can be calculated recursively from time 1 to time t.

4.1 Initialization

$$\alpha_1(i) \equiv P(o_1, q_1 = s_i \mid \lambda) \tag{3}$$

$$= P(o_1 \mid q_1 = s_i, \lambda) P(q_1 = s_i \mid \lambda) \tag{4}$$

$$= \pi_i b_i(o_1) \tag{5}$$

4.2 Recursion

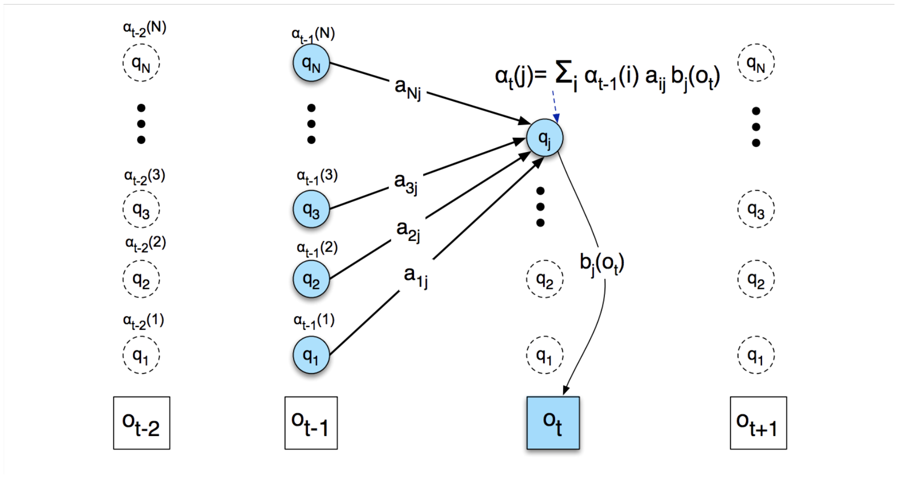
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$$

4.3 Termination

$$\mathcal{P}(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_T = s_i \mid \lambda) \tag{6}$$

$$= \sum_{i=1}^{N} \alpha_T(i) \tag{7}$$

Below, you can see a visualization of the computation of a single element $\alpha_t(i)$ in the structure by summing all the previous values α_{t-1}, weighted by their transition probabilities α, and multiplying by the observation probability $b_i(o_{t+1})$.



5 Decoding: The Viterbi Algorithm

Recall the decoding problem from section 3.2. Given an observation sequence $O = (o_1 o_2 \ldots o_T)$, and a model λ, what is the optimal hidden sequence of states that best corresponds to the observation sequence? At the outset, it may be possible to run the forward algorithm and compute the likelihood of the observation sequence given that hidden state sequence. Then we could choose the hidden state sequence with the maximum observation likelihood. It should be clear from the previous section that we cannot do this because there are an exponentially large number of state sequences.

Instead, the most common decoding algorithm is the Viterbi algorithm, which makes use of dynamic programming. The idea behind the algorithm is to recursively find the best state sequence, taking the transition probabilities into account. The process is very similar to the forward algorithm. Just as in the figure in the previous section we see each cell being filled by the partial probability $\alpha_t(j)$, each cell in this case will be filled by $v_t(j)$. This represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $q_0, q_1, ..., q_{t1}$:

$$v_t(j) = \max_{q_0, q_1, ..., q_{t1}} P(q_0, q_1, ..., q_{t1}, o_1, o_1, ..., o_t, q_t = j \mid \lambda)$$

For a given state q_j at time t, the value $v_t(j)$ is computed as:

$$v_t(j) = \max_{q_0, q_1, ..., q_{t1}}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

The Viterbi algorithm also makes use of backpointers which help keep track of the path of hidden states that lead to each state. The forward algorithm doesn't need this because it is only producing an observation likelihood. It is defined as $\psi_t(j)$, which refers to the best previous state that maximizes $v_t(j)$ of the current state.

5.1 Initialization

$$v_1(j) = a_{0j} b_j(o_1) \tag{8}$$
$$\psi_1(j) = 0 \tag{9}$$

5.2 Recursion

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \tag{10}$$
$$\psi_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \tag{11}$$

5.3 Termination

$$\mathcal{P}* = \max_{i=1}^{N} v_T(i) * a_{iF} \tag{12}$$
$$q_T* = \operatorname*{argmax}_{i=1}^{N} v_T(i) a_{iF} \tag{13}$$

6 Learning: The Baum-Welch Algorithm

The third main problem that HMMs can be used for pertains to expectation maximization. Given an observation sequence $O = (o_1 o_2 \ldots o_T)$, how do we adjust the parameters of an HMM model λ to maximize P(O|λ)? The Baum-Welch, or the Forward-Backward Algorithm, is used to iteratively optimize the probability estimates of the HMM. It works by computing an initial estimate for the probabilities, then using those estimates for computing a better estimate, and so on, iteratively improving the probabilities that it learns.

Similar to what we learned about neural networks and back propagation, you can think of this as a sort of gradient ascent, where the algorithm is trying to climb up the graph and find the model that yields the greatest likelihood. As such, the Baum-Welch Algorithm can get stuck in local maxima. Notice this bears many parallels to using k-means for clustering.

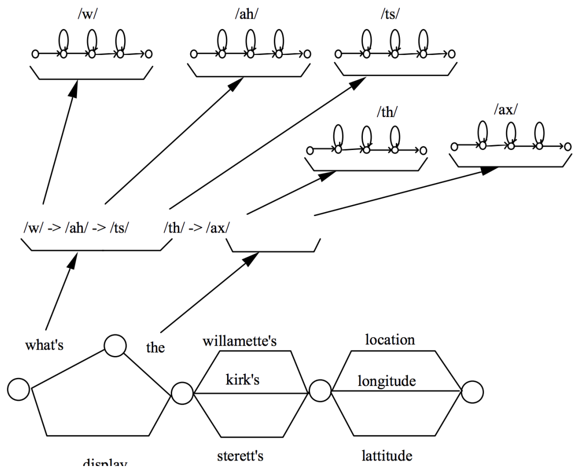
7 Applications

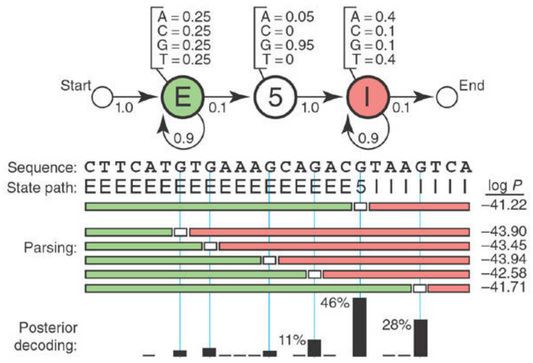
HMMs, being computationally straightforward, provide a good statistical framework for solving a wide range of time-series problems, especially pertaining to classification and pattern recognition. A popular example is speech recognition, and HMMs are used in nearly all such systems. Given an acoustic waveform, HMMs can be used to find a phrase of words that best fit the signal input. Essentially, this is the decoding problem outlined in section 5. What we are trying to find can be expressed like this, where w is a string of words, L is the relevant language, and y is the set of processed acoustic vectors:

$$\operatorname*{argmax}_{w \in L} P(w \mid y)$$

Feature extraction is done to the waveform most commonly through the MFCC, or Mel-Frequency Cepstral Coefficients, encoding scheme. This yields a group of feature vectors and from here we can use these discrete data points as observations in an HMM; Decoding for multiple words is accomplished through multiple layers of HMMs; basic phonemes, or distinct sounds, are constructed into likely syllables, which are then matched to words, and words into a phrase.

HMMs are also prominently used in biology, namely in computational sequence analysis. Below is an example of a site recognition problem, where an exon, intron, and the 5SS consensus nucleotide. The emission probabilities are known based on the frequency of the bases in each component.





8 Disadvantages

HMMs do have some downfalls:

- 1st-order HMMs are limited by the first-order property
- Training an HMM is somewhat expensive
  - use set of seed sequences
  - time needed for convergence
- Although the flexibility of an HMM allows unobserved variables, it may result in less accurate predictions
- Lack the memory, context tracking, and pattern learning of RNNs

9 Conclusion

Overall, HMMs have a strong statistical foundation and are able to flexibly and efficiently interpret sequence data. However, they have been around for decades. With recent technology enabling the increased use of more computationally expensive methods, models like RNNs have been seen to surpass HMMs in specific uses like Speech Recognition and reading handwriting.