

Hidden Markov Model (HMM)

Tutorial

This page will hopefully give you a good idea of what Hidden Markov Models (HMMs) are, along with an intuitive understanding of how they are used. They are related to Markov chains, but are used when the observations don't tell you exactly what state you are in. To fully explain things, we will first cover Markov chains, then we will introduce scenarios where HMMs must be used. There will also be a slightly more mathematical/algorithmic treatment, but I'll try to keep the intuitive understanding front and foremost.

Markov Chains

Markov chains (and HMMs) are all about modelling sequences with discrete states. Given a sequence, we might want to know e.g. what is the most likely character to come next, or what is the probability of a given sequence. Markov chains give us a way of answering this question. To give a concrete example, you can think of text as a sequence that a Markov chain can give us information about e.g. 'THE CAT SA'. What is the most likely next character? Another application is determining the total likelihood of a sequence e.g. which is more likely: 'FRAGILE' or 'FRAMILE'?

Markov chains only work when the states are discrete. Text satisfies this property, since there are a finite number of characters in a sequence. If you have a continuous time series then Markov chains can't be used.

To define a Markov chain we need to specify an alphabet i.e. the possible states our sequence will have e.g. $\mathbf{S} = \{S_1, S_2, \dots, S_K\}$. For text this will be the letters A-Z and punctuation. For proteins this would be the 20 possible amino acids etc. We also need transition probabilities e.g. what is the probability of 'H' following 'T'? Finally we need the probability of characters on their own (the a priori probabilities) e.g. what is the probability of a random character being 'F'.

Once we have these things, we can calculate the probability of a sequence:

$$P(X_1 X_2 X_3 \dots X_N) = P(X_1) \prod_{i=1}^N P(X_{i+1} | X_i)$$

where $P(X_{i+1} | X_i)$ is the probability of character X_{i+1} given that X_i was previous (this is the transition probability). The expression gives the probability of the sequence $X_1 X_2 \dots X_N$, assuming the sequence is a first order markov process i.e. each character only depends on the immediately preceding character. In general this is not true for text, a higher order model will perform better. Using a higher order model will require more transition states, and more training data to estimate the transition probabilities.

To train the transition probabilities for a markov chain you need lots of training data. For English text this will be many sentences. Given this training data we can count the number of times each character follows each other character, then get probabilities by dividing the counts by the number of occurrences. What can happen is that certain transitions will have a count of 0. If this transition occurs in testing, it will give a probability of

0 to the whole sequence, which is generally not wanted. To get around this pseudocounts are added. There are algorithms for determining what all the unseen transitions should be e.g. [Kneser-Ney smoothing](#) among many other algorithms.

Hidden Markov Models

For Markov chains we had a state sequence e.g. characters and we always knew what the states were, i.e. there was no uncertainty. Imagine a scenario where instead of reading the characters from a page, we have a friend in a sound proof booth. He can read the characters out and we have to determine what characters he is saying by lip reading. In this case our states are the same characters as the markov chain case (characters on a page), but now we have an extra layer of uncertainty e.g. it looked like he said 'P', but 'P' looks a lot like 'B', or maybe it was even 'T'. What's important is we can give the probability of a particular observation given that we are in a particular state.

When we speak about HMMs, we still have the transition probabilities between states, but we also have observations (which are not states). Note that we never actually get to see the real states (the characters on the page) we only see the observations (our friend's mouth movements). One of the main uses of HMMs is to estimate the most likely state sequence given the observations i.e. what was written on the page given the sequence of observations. You can imagine the difficulty: if we know the previous state we can multiply the transition probability to the next state by the observation probability, but we don't know the previous state either! So we have to estimate them all at once. We will cover the algorithms for doing this below.

Another example where HMMs are used is in speech recognition. The states are phonemes i.e. a small number of basic sounds that can be produced. The observations are frames of audio which are represented using [MFCCs](#). Given a sequence of MFCCs i.e. the audio, we want to know what the sequence of phonemes was. Once we have the phonemes we can work out words using a phoneme to word dictionary. Determining the probability of MFCC observations given the state is done using [Gaussian Mixture Models \(GMMs\)](#).

The extra layer of uncertainty i.e. not being able to look at the states, makes working with HMMs a little bit more complicated than working with markov chains. There are several algorithms for working with HMMs: for estimating the probability of a sequence of observations given a model the Forward algorithm is used. Given a sequence of observations and a model, to determine the most likely state sequence the Viterbi algorithm is used. Finally, given a lot of observation/state pairs the Baum-Welch algorithm is used to train a model.

A mathematical treatment

The previous sections have been a bit hand wavy to try to convey an intuitive motivation for HMMs. If you want to implement them though, we need to get a little bit more rigorous. In the Markov chain, each state corresponds to an observable event, i.e. given an event we can say with 100% accuracy which state we are in. In the HMM, the observation is a probabilistic function of the state. A HMM is basically a Markov chain where the output observation is a random variable X generated according to an output probabilistic function associated with each state. This means there is no longer a one-to-one correspondence between the observation sequence and the state sequence, so you cannot know for certain the state sequence for a given observation sequence.

A Hidden Markov Model is defined by:

- $\mathbf{O} = \{o_1, o_2, \dots, o_M\}$ – An output observation alphabet. The observation symbols correspond to the physical output of the system being modeled. For speech recognition these would be the MFCCs.
- $\mathbf{\Omega} = \{1, 2, \dots, N\}$ – A set of states representing the state space. s_t is denoted as the state at time t . For speech recognition these would be the phoneme labels, for text it would be the set of characters.
- $\mathbf{A} = \{a_{ij}\}$ – A transition probability matrix, where a_{ij} is the probability of taking a transition from state i to state j .
- $\mathbf{B} = \{b_i(k)\}$ – An output probability distribution, where $b_i(k)$ is the probability of emitting symbol o_k when in state i .
- $\pi = \{\pi_i\}$ – An initial state distribution where $\pi_i = P(s_0 = i) \quad 1 \leq i \leq N$

A complete specification of an HMM includes two parameters N and M , representing the total number of states and the size of the observation alphabet, the observation alphabet \mathbf{O} , and three sets of probability measures $\mathbf{A}, \mathbf{B}, \pi$. A HMM is often denoted by Φ , where $\Phi = (\mathbf{A}, \mathbf{B}, \pi)$. Given the definition above, three basic problems of interest must be addressed before HMMs can be applied to real-world applications:

1. The Evaluation Problem. Given a model Φ and a sequence of observations

$\mathbf{X} = (X_1, X_2, \dots, X_T)$, what is the probability that the model generates the observations, $P(\mathbf{X}|\Phi)$.

2. The Decoding Problem. Given a model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, \dots, X_T)$, what is the most likely state sequence $\mathbf{S} = (s_0, s_1, s_2, \dots, s_T)$ in the model that produces the observations.

3. The Learning Problem. Given a model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, \dots, X_T)$, how can we adjust the model parameter $\hat{\Phi}$ to maximise the joint probability $\prod_X P(\mathbf{X}|\Phi)$ i.e. train the model to best characterise the states and observations.

Evaluating an HMM

The aim of evaluating an HMM is to calculate the probability $P(\mathbf{X}|\Phi)$ of the observation sequence $\mathbf{X} = (X_1, X_2, \dots, X_T)$, given the HMM Φ . The steps involved in calculating $P(\mathbf{X}|\Phi)$ are shown below. The algorithm is called the forward algorithm.

To calculate the likelihood of a sequence of observations, you would expect that the underlying state sequence should also be known (since the probability of a given observation depends on the state). The forward algorithm gets around this by summing over all possible state sequences. A dynamic programming algorithm is used to do this efficiently.

You may wonder why we would want to know the probability of a sequence without knowing the underlying states? It gets used during training e.g. we want to find parameters for our HMM $\Phi = (\mathbf{A}, \mathbf{B}, \pi)$ such that the probability of our training sequences is maximised. The Forward algorithm is used to determine the probability of a an observation sequence given a HMM.

- Step 1: Initialisation

$$\alpha_1(i) = \pi_i b_i(X_1) \quad 1 \leq i \leq N$$

- Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^N \alpha_{t-1}(i) a_{ij} \right] b_j(X_t) \quad 2 \leq t \leq T, 1 \leq j \leq N$$

- Step 3: Termination

$$P(\mathbf{X}|\Phi) = \sum_{i=1}^N \alpha_T(i)$$

If the HMM must end in a particular exit state (final state s_F) then,

$$P(\mathbf{X}|\Phi) = \alpha_T(s_F)$$

Understanding the Forward Algorithm

The forward algorithm can be understood to be working on a matrix of $\alpha_t(i)$ values. The values of $\alpha_t(i)$ represent the likelihood of being in state i at time t given the observations up to time t . You can visualise the alpha values like this:

	t_1	t_2	t_3	...	t_T
A	$\alpha_1(1)$	$\alpha_2(1)$...	$\alpha_T(1)$
B	$\alpha_1(2)$	$\alpha_2(2)$...	
C	$\alpha_1(3)$...	
...
Z	$\alpha_1(26)$...	$\alpha_T(26)$

The lattice of alphas calculated by the forward algorithm.

The blue circles are the values of alpha. $\alpha_1(1)$ is the top left circle etc. The initialisation step from the previous section fills the first column. Since there is no previous state, the probability of being in e.g. state 1 at time 1 is just the probability of starting in state 1 times the probability of emitting observation X_1 : i.e. $\alpha_1(i) = \pi_i b_i(X_1)$. Once we have done that for all the states initialisation is complete.

The induction step takes into account the previous state as well. If we know the previous state is e.g. state 2, then the likelihood of being in state 4 in the current time step is $\alpha_{t+1}(4) = \alpha_t(2) a_{24} b_4(X_t)$ i.e. the likelihood of being in state 2 previously, times the transition probability of going from state 2 to state 4,

times the probability of seeing observation X_t in state 4. Of course, we don't know the previous state, so the forward algorithm just sums over all the previous states i.e. we marginalise over the state sequence.

Remember the job of the forward algorithm is to determine the likelihood of a particular observation sequence, regardless of state sequence. To get the likelihood of the sequence we just sum the final column of alpha values.

The Viterbi algorithm is almost identical to the forward algorithm, except it uses argmax instead of sum. This is because it wants to find the most likely previous state, instead of the total likelihood. To find the best state sequence we backtrack through the lattice, reading off the most likely previous states until we get back to the start.

Decoding an HMM

The aim of decoding an HMM is to determine the most likely state sequence $\mathbf{S} = (s_0, s_1, s_2, \dots, s_T)$ given the observation sequence $\mathbf{X} = (X_1, X_2, \dots, X_T)$ and the HMM Φ . The steps involved in calculating $P(\mathbf{X}|\Phi)$ are shown below. The algorithm is called the Viterbi algorithm.

The viterbi algorithm is very similar to the forward algorithm, except it uses argmax instead of summing over all possible sequences. This is necessary to enable us to pick the most likely sequence. It would seem like argmax is discarding a lot of information and should result in suboptimal results, but in practice it works well.

- Step 1: Initialisation

$$V_1(i) = \pi_i b_i(X_1) \quad 1 \leq i \leq N$$

$$B_1(i) = 0$$

- Step 2: Induction

$$V_t(j) = \max_{1 \leq i \leq N} [V_{t-1}(i) a_{ij}] b_j(X_t) \quad 2 \leq t \leq T, 1 \leq j \leq N$$

$$B_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [V_{t-1}(i) a_{ij}] \quad 2 \leq t \leq T, 1 \leq j \leq N$$

- Step 3: Termination

$$\text{best score} = \max_{1 \leq i \leq N} [V_t(i)]$$

$$s_T^* = \operatorname{argmax}_{1 \leq i \leq N} [B_t(i)]$$

- Step 4: Backtracking

$$s_t^* = B_{t+1}(s_{t+1}^*) \quad t = T-1, T-2, \dots, 1$$

$$\mathbf{S}^* = (s_1^*, s_2^*, \dots, s_T^*) \quad \text{is the best sequence}$$

Estimating HMM Parameters

Estimating the HMM parameters is the most difficult of the three problems, because there is no known analytical method that maximises the joint probability of the training data in a closed form. Instead, the problem can be solved by the iterative Baum–Welch algorithm, also known as the forward backward algorithm. The forward backward algorithm is a type of Expectation Maximisation algorithm.

The idea is to find values for $\Phi = (\mathbf{A}, \mathbf{B}, \pi)$ such that the probability of the training observations (calculated using the forward algorithm) is maximised. We perturb the parameters until they can no longer be improved. I won't go into the details here, an outline is given below.

1. Initialisation. Choose an initial estimate Φ .
2. E-step. Compute auxiliary function $Q(\Phi, \hat{\Phi})$ based on Φ .
3. M-step. Compute $\hat{\Phi}$ according to the re-estimation equations on page 392 of Spoken Language Processing to maximise Q .
4. Iteration. Set $\Phi = \hat{\Phi}$, repeat from step 2 until convergence.