

Hidden Markov Models

Modeling, Inference, Learning

Nishant Mehta

College of Computing
Georgia Institute of Technology
www.cc.gatech.edu/~niche

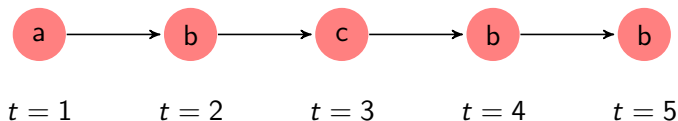
What Are Hidden Markov Models?

Hidden Markov models (HMMs) are discrete Markov processes where every state generates an observation at each time step.

A discrete process is a random variable that, at each time step, takes on a state value.

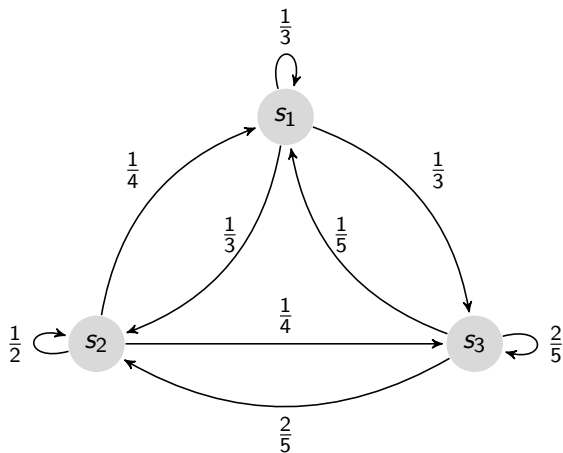
A Markov discrete process is a memoryless process, where the transition probabilities into the next state are entirely dependent upon the current state.

Data generated from a Markov discrete process is a Markov chain. Below is a Markov chain that emits symbols in the set $\{a,b,c\}$.



Discrete Markov Process

An example discrete Markov process would be



Discrete Markov Process Model

What are the parameters of a discrete Markov process model?

With what probability do we start in each state at $t = 0$?

What is the probability that we transition from state s_i to state s_j at time t ?

Discrete Markov Process Model

What are the parameters of a discrete Markov process model?

With what probability do we start in each state at $t = 0$?

- The initial state probabilities π_i

What is the probability that we transition from state s_i to state s_j at time t ?

- The state transition probabilities a_{ij} , stored in a matrix A

Hidden Markov Models

Since HMMs are discrete Markov processes where each state also emits an observation according to some probability distribution, we need to augment our model.

What are the parameters of a hidden Markov model?

- The initial state probabilities π_i
- The state transition probabilities a_{ij} , stored in a matrix A

If we are in some state at time t , what is the density function from which we draw our observations?

Hidden Markov Models

Since HMMs are discrete Markov processes where each state also emits an observation according to some probability distribution, we need to augment our model.

What are the parameters of a hidden Markov model?

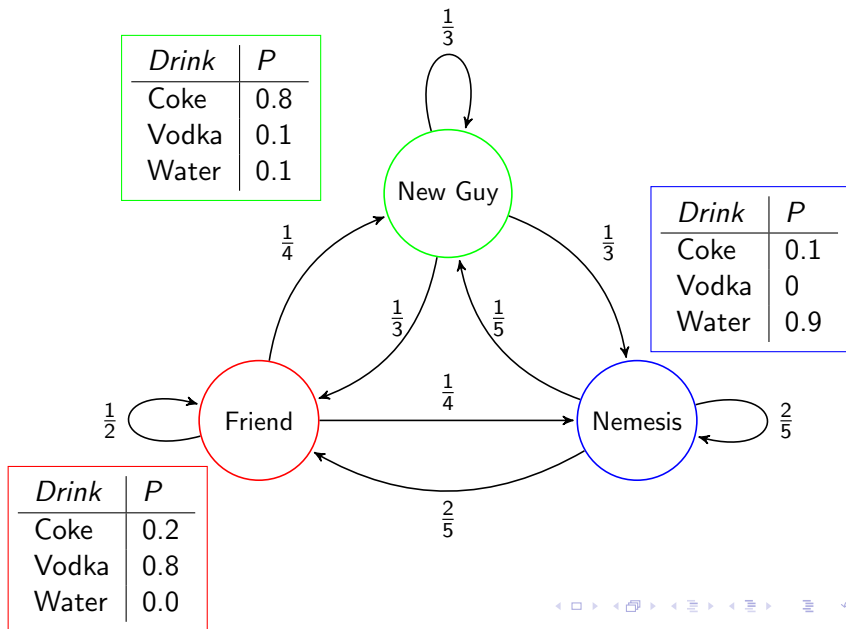
- The initial state probabilities π_i
- The state transition probabilities a_{ij} , stored in a matrix A

If we are in some state at time t , what is the density function from which we draw our observations?

- The probability distribution parameters B for each state. The corresponding probabilities are sometimes called the emission probabilities.

Discrete $b_i(k)$ is the probability that state s_i emits the k^{th} symbol.
Continuous $b_i(x)$ is s_i 's PDF evaluated at x

A Hidden Markov Model Example



Notation

For compactness, we will use the following notation:

$$X_{1:t} = x_1 x_2 \dots x_t$$
$$X = X_{1:T} = x_1 x_2 \dots x_T$$

Three Fundamental Questions

- ① Given an HMM, what is the likelihood of seeing a particular sequence of observations $X_{1:T}$?
- ② Given an HMM and an observation sequence $X_{1:T}$, which state sequence $Q_{1:T}$ best explains the observations?
 - Consider maximum likelihood states at each time step individually
 - Consider single best state sequence to maximize $P(Q | X, \theta)$
- ③ Given a sequence of observations $X_{1:T}$, how do we learn a good HMM for modeling the data?

The first two problems turn out to be variations of the same general dynamic programming algorithm. The first problem can be solved by the forward algorithm, which is an instance of a sum-product algorithm. The second problem can be solved by the Viterbi algorithm, which is an instance of a max-product algorithm.

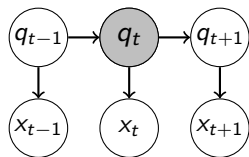
Forward Algorithm

Given an HMM $\theta = \{\pi, A, B\}$, what is the likelihood of seeing a particular sequence of observations $X_{1:T}$?

$$P(X | \theta) = \sum_Q P(X, Q | \theta) = \sum_Q P(X | Q, \theta) P(Q | \theta)$$

Forward Algorithm

Recall the graphical model for an HMM:



We see that the probability distribution of random variable x_t is conditionally independent of $G \setminus \{q_t, x_t\}$, given q_t . That is:

$$x_t \perp\!\!\!\perp \{Q_{1:t-1}, Q_{t+1:T}, X_{1:t-1}, X_{t+1:T}\} \mid q_t$$

We can use this conditional independence relationship to efficiently decompose $P(X \mid Q, \theta)$ into

$$P(X \mid Q, \theta) = \prod_{t=1}^T P(x_t \mid q_t, \theta) = \prod_{t=1}^T b_{q_t}(x_t).$$

Forward Algorithm

$P(Q | \theta)$ decomposes by considering the Markov property:



$$\begin{aligned} P(Q | \theta) &= P(q_1 | \theta) \prod_{t=2}^T P(q_t | Q_{1:t-1}, \theta) \\ &= P(q_1 | \theta) \prod_{t=2}^T P(q_t | q_{t-1}, \theta) \\ &= \pi_{q_1} \prod_{t=1}^T a_{q_{t-1}, q_t} \end{aligned}$$

Forward Algorithm

The previous method involves summing over all N^T possible sequences $Q_{1:T}$, which is exponential in the number of symbols.

Instead, we can exploit the Markov property to use a dynamic programming algorithm. Consider the probability of seeing the first t observations and then ending up in state s_j at time t .

$$\alpha_t(i) = P(x_1 x_2 \dots x_t, q_t = s_i \mid \theta)$$

The variables $\alpha_t(i)$ are called the forward variables.

Forward Algorithm

$$\alpha_t(i) = P(x_1 x_2 \dots x_t, q_t = s_i \mid \theta)$$

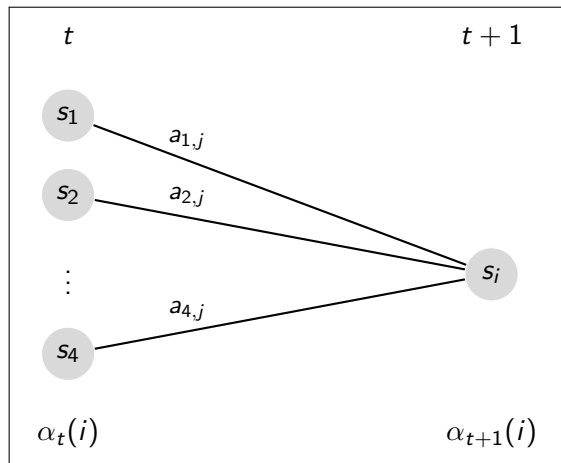
In particular, we have the base case $\alpha_1(i) = \pi_i b_i(x_1)$.

It turns out that in order to calculate this probability for any $t + 1$, we only need to know

- The probability of seeing the first t observations and ending up in each state s_j at time $t - 1$.
- The probability of then transitioning into state s_j and observing x_{t+1} .

Forward Algorithm - Trellis Representation

Being in a particular state s_i at time $t - 1$ provides one path into state s_j at time t , via a transition (s_i, s_j) .



Forward Algorithm

We just sum the probabilities of all N paths and consider the probability of observing x_t in state s_j .

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

Once we have $P(X_{1:T}, q_t = s_i | \theta)$ for $i = 1, \dots, N$, we just marginalize over all s_i to get our answer.

$$\sum_{i=1}^N \alpha_T(i) = \sum_{i=1}^N P(X_{1:T}, q_t = s_i | \theta) = P(X_{1:T} | \theta)$$

Forward Algorithm

The forward algorithm is

- ① $\alpha_1(i) = \pi_i b_i(x_1)$
- ② Repeat for $t = 1 \rightarrow T - 1$:

Repeat for $j = 1 \rightarrow N$:

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

- ③ $P(X_{1:T} | \theta) = \sum_{i=1}^N \alpha_T(i)$

This algorithm as time complexity is $O(N^2 T)$

Backward Algorithm

The forward algorithm gives us the probability of seeing the observations up to time t and ending up at state s_i at time t :
 $\alpha_t(i) = P(X_{1:t}, q_t = s_i)$.

Suppose we wanted to calculate the probability of seeing **all** observations X and being in state s_i at time t : $P(X_{1:T}, q_t = s_i)$.

If we had this probability for all s_i , then we could find the probability of being in a particular state at a given time t , using Bayes rule:

$$\begin{aligned} P(q_t = s_i | X, \theta) &= \frac{P(X, q_t = s_i | \theta)}{P(X | \theta)} \\ &= \frac{P(X, q_t = s_i | \theta)}{\sum_{i=1}^N P(X, q_t = s_i | \theta)} \end{aligned}$$

Backward Algorithm

$$P(X_{1:T}, q_t = s_i) = \underbrace{P(X_{1:t}, q_t = s_i \mid \theta)}_{\alpha_t(i)} \underbrace{P(X_{t+1:T} \mid q_t = s_i, \theta)}_{\beta_t(i)}$$

We can see that this factorization is true from the graphical model of an HMM, where $X_{t+1:T} \perp\!\!\!\perp X_{1:t} \mid q_t$.

The $\beta_t(i)$ variables are called the backward variables.

They can be computed using the backward algorithm, a dynamic programming algorithm similar to the forward algorithm.

Backward Algorithm

$$\beta_t(i) = \mathbb{P}(X_{t+1:T} \mid q_t = s_i, \lambda)$$

$$\beta_t(i) = 1 \forall i$$

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(x_{t+1}) \beta_{t+1}(j) \text{ for } t = T-1, T-2, \dots, 1$$

This takes $O(N^2 T)$ calculations.

Forward-Backward Algorithm

Using the factorization

$$\begin{aligned} P(X_{1:T}, q_t = s_i) &= P(X_{1:t}, q_t = s_i | \theta) P(X_{t+1:T} | q_t = s_i, \theta) \\ &= \alpha_t(i) \beta_t(i) \end{aligned}$$

we now can calculate the probability of being in a state at time t .

$$\begin{aligned} \gamma_t(i) = P(q_t = s_i | X, \theta) &= \frac{P(X, q_t = s_i | \theta)}{\sum_{i=1}^N P(X, q_t = s_i | \theta)} \\ &= \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)} \end{aligned}$$

Viterbi Algorithm

There are different ways to define optimality of the sequence of hidden states.

One optimality criterion would be to maximize the expected number of correct individual states. At every time step, we just select the maximum likely state:

$$\begin{aligned}q_t &= \arg \max_{s_i} P(q_t = s_i \mid X, \theta) \quad \forall t \in \{1, 2, \dots, T\} \\ &= \arg \max_{s_i} \gamma_t(i) \quad \forall t \in \{1, 2, \dots, T\}\end{aligned}$$

Another, more commonly used optimality criterion is to select the single best state sequence, known as the Viterbi path.

While the path consisting of the maximum likely states at every time step may not even be feasible given the state transition probabilities, the Viterbi path maximizes the probability of the state sequence jointly, by maximizing $P(Q, X \mid \theta)$.

Viterbi Algorithm

$V_t(i)$ is the maximum likelihood sequence of states $q_1 q_2 \dots q_t$ terminating at state $q_t = s_i$, given that we observe $X_{1:t}$.

The probability of the path $V_t(i)$ is

$$\delta_t(i) = \max_{Q_{1:t-1}} P(q_1 q_2 \dots q_t = i, X_{1:t} | \theta).$$

The maximum likelihood sequence of t states terminating at some state s_j is simply the mostly likely among all maximum likelihood sequences of $t - 1$ states terminating in some state s_i and then transitioning to state s_j . This leads to a recurrence formula, similar to the $\alpha_t(i)$ in the forward algorithm:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(x_{t+1})$$

This update is precisely the update from the forward algorithm, with the summation operator replaced with a max operator. This is known as a max-product algorithm, and will come up again during MAP estimation for graphical model inference.

Viterbi Algorithm

The Viterbi algorithm consists of the following steps

- ① For $i = 1, 2, \dots, N$
 - $\delta_1(i) = \pi_i b_i(x_1)$
 - $V_1(i) = s_i$
- ② For $t = 1, 2, \dots, T - 1$
 - For $j = 1, 2, \dots, N$
 - $\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(x_{t+1})$
 - $V_{t+1}(j) = [V_t(k) s_j]$ where $k = \arg \max_i \delta_t(i) a_{ij}$

The Viterbi path is then $V_T(k)$ where $k = \arg \max_i \delta_T(i) a_{ij}$

Baum-Welch Algorithm

To learn the parameters (π, A, B) of an HMM, we can use the probability that the HMM is in state s_i at time t followed by state s_j at time $t + 1$, conditioning on the observations X .

$$\begin{aligned}\xi_t(i, j) &= P(q_t = s_i, q_{t+1} = s_j \mid X, \theta) \\ &= \frac{P(q_t = s_i, q_{t+1} = s_j, X \mid \theta)}{P(X \mid \theta)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{P(X \mid \theta)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}\end{aligned}$$

Baum-Welch Algorithm

Note that the $\xi_t(i, j)$ variables allow us to recover the γ_t variables from before by marginalizing over the second state, to yield the probability of being in state s_i at time t , conditioning on X :

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j)$$

Baum-Welch Algorithm

The Baum-Welch algorithm is an EM algorithm that optimizes the maximum likelihood objective $P(X | \theta)$. For HMMs, this objective has many local minima.

This algorithm iteratively does an expectation step and a maximization step until the $P(X | \theta)$ no longer increases.

In the expectation step, we use the forward-backward algorithm to calculate $\gamma_t(i)$ and $\xi_t(i, j)$, by computing the forward variables $\alpha_t(i)$ via the forward algorithm and the backward variables $\beta_t(j)$ via the backward algorithm.

Baum-Welch Algorithm

Given the expectation step, it can be shown that the optimal update to the model, or the maximization step of EM, is:

$$\pi_i = \gamma_1(i) = P(q_1 = s_i | X, \theta)$$

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

$$b_j(k) = \frac{\sum_{t=1}^T \gamma_t(j) I(x_t = v_k)}{\sum_{t=1}^T \gamma_t(j)}$$

Viterbi Training

An inconsistent but computationally easier method for updating the model is to use Viterbi Training.

We first find the maximum likelihood sequence of states given the observations X . This is the Viterbi path.

The modified M step is now

$$\pi_i = I(q_1 = s_i \text{ in Viterbi path})$$

$$a_{ij} = \frac{\text{number of times in state } s_i \text{ immediately followed by state } s_j}{\text{number of times in state } s_i}$$

$$b_j(k) = \frac{\sum_{t=1}^T I(q_t = s_j \text{ in Viterbi path}) I(x_t = v_k)}{\sum_{t=1}^T I(q_t = s_j \text{ in Viterbi path})}$$

HMMs for Continuous Observation Densities

HMMs can be extended for the continuous observation case.

Let's consider states that generate observations according to a multivariate Gaussian distribution where each state $s_j \sim \mathcal{N}(\mu_j, \Sigma_j)$.

The M-step updates for Baum-Welch become

$$\mu_j = \frac{\sum_{t=1}^T \gamma_t(j) x_t}{\sum_{t=1}^T \gamma_t(j)}$$
$$\Sigma_j = \frac{\sum_{t=1}^T \gamma_t(j) (x_t - \mu_j)(x_t - \mu_j)^T}{\sum_{t=1}^T \gamma_t(j)}$$