

Lecture 23–24: Hidden Markov Models

TBA

Contents

1	Overview	1
2	Definition and Example	1
3	Three Basic Problems	2
4	The Evaluation Problem	3
4.1	The Forward Algorithm	3
4.2	The Backward Algorithm	4
4.3	The Forward-Backward Algorithm	5
5	The Decoding Problem	6

1 Overview

Hidden Markov models (HMMs) are used as a modeling tool in a number of application areas, ranging from speech recognition to DNA sequencing to data compression to computer vision. HMMs are Markov processes—described by a Markov matrix (i.e, a transition probability matrix) and an initial distribution—together with an emission probability matrix, which reveals partial state information with each state transition. *Complete state information is hidden.*

2 Definition and Example

A discrete-time *hidden Markov model* is a tuple $\mathcal{H} = \langle X, Y, \Pi, A, B \rangle$, where time is discrete: *i.e.*, $t \in T = \{1, 2, \dots\}$, and

- X is a finite set of states
- Y is a finite alphabet of observations
- $\Pi = \{\pi_i \mid x_i \in X\}$ is an initial probability distribution over start states
- $A = \{a_{ij} \mid x_i, x_j \in X\}$, where $a_{ij} \equiv a(x_i, x_j)$ denotes the probability of transitioning from state x_i to state x_j , is the *transition* probability matrix

N.B.: $\sum_j a_{ij} = 1$ for all states $x_i \in X$.

- $B = \{b_{ijk} \mid x_i, x_j \in X, y_k \in Y\}$, where $b_{ijk} \equiv b(x_i, x_j, y_k)$ denotes the probability of observing y_k upon transitioning from state x_i to state x_j , is called the *emission* probability matrix
- N.B.:** $\sum_k b_{ijk} = 1$ for all states $x_i, x_j \in X$ s.t. $a_{ij} > 0$.

Example: Figure 1 depicts an example of an HMM. The set of states $X = \{1, 2, 3\}$. All transitions between states occur with uniform probability: i.e., $a_{11} = a_{12} = a_{13} = 1/3$; $a_{22} = a_{23} = 1/2$; $a_{33} = 1$. The initial distribution is also uniform. The alphabet of observations $Y = \{H, T\}$, and at state i , $b_{ijH} = 1/i$, for all j s.t. $a_{ij} > 0$.

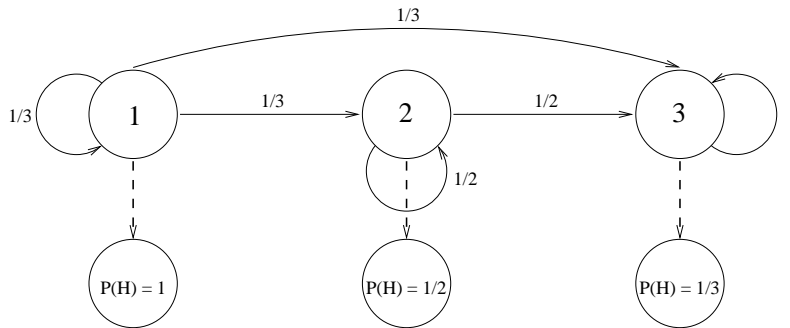


Figure 1: Example: HMM with uniform transition and emission probabilities.

The *trellis* is a visual aid that helps us to envision the state space of an HMM as it evolves over time. The trellis is constructed by listing all the states of an HMM in columns and drawing arrows from each state x_i in one column to all the states x_j in the next column for which $a_{ij} > 0$: i.e., all the states x_j which are reachable from state x_i . A state sequence is a path through the trellis.

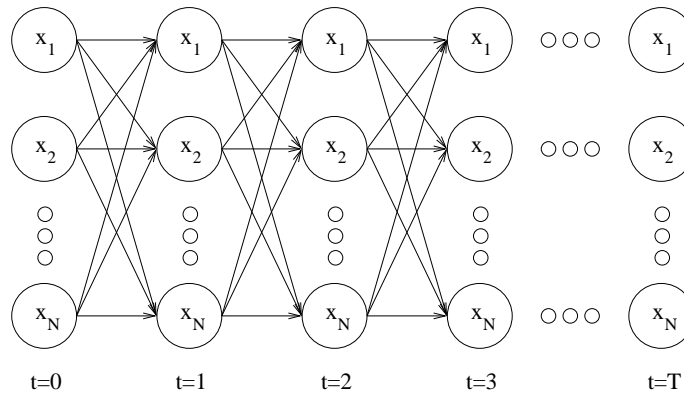


Figure 2: The Trellis. In this example, it is possible to transition from any state to any other state: i.e., for all states x_i, x_j , the transition probability $a_{ij} > 0$.

Exercise: Draw the trellis corresponding to the HMM depicted in Figure 1.

3 Three Basic Problems

Let $x \equiv (x^{0:T})$ denote a state sequence; let $y \equiv (y^{1:T})$ denote an observation sequence.

Evaluate Given an HMM λ and the sequence of observations y , compute the probability $P[y]$: *i.e.*, the probability that the HMM generates the sequence of observations y .

Decode Given an HMM λ and the sequence of observations y , compute x *s.t.* $P[x, y]$ is maximal: *i.e.*, compute x *s.t.* $P[y | x]P[x]$ is maximal: *i.e.*, compute the sequence of states x most likely to generate the sequence of observations y .

Learn Given the sequence of observations y , estimate the parameters Π, A, B of an HMM λ *s.t.* $P[y]$ is maximal: *i.e.*, construct λ that is most likely to generate the sequence of observations y .

4 The Evaluation Problem

Given the sequence of observations y , the probability $P[y]$ can be computed as follows: (i) compute the probability $P[x]$ of path x , (ii) compute the probability $P[y | x]$ of observation sequence y given path x , (iii) sum the product $P[x]P[y | x]$ over all paths x . Formally,

$$P[y] = \sum_x P[x, y] = \sum_x P[x]P[y | x] \quad (1)$$

The RHS of Equation 1 simplifies by applying the Markov and CI assumptions. The probability $P[x]$ of path x equals the probability of state x^0 , times the probability of transitioning from state x^0 to state x^1 , times the probability of transitioning from state x^1 to state x^2 , and so on.

$$P[x] = \pi(x^0) \prod_{t=1}^T a(x^{t-1}, x^t) \quad (2)$$

The probability $P[y | x]$ of observation sequence y given path x is computed by multiplying the probability of observing y^1 upon transitioning from state x^0 to state x^1 , times the probability of observing y^2 upon transitioning from state x^1 to state x^2 , and so on.

$$P[y | x] = \prod_{t=1}^T b(x^{t-1}, x^t, y^t) \quad (3)$$

Hence,

$$P[x, y] = \pi(x^0) \prod_{t=1}^T a(x^{t-1}, x^t) b(x^{t-1}, x^t, y^t) \quad (4)$$

and, moreover,

$$P[y] = \sum_x \pi(x^0) \prod_{t=1}^T a(x^{t-1}, x^t) b(x^{t-1}, x^t, y^t) \quad (5)$$

Direct evaluation of Equation 5 requires $O(TN^T)$ multiplications, where N is the number of states in the HMM, since there are N^T paths through the trellis each of which requires T multiplications.

4.1 The Forward Algorithm

The *forward algorithm* is an efficient alternative for computing the quantity $P[y]$ based on dynamic programming. This algorithm computes the probabilities of partial observation sequences in terms

of the probabilities of shorter observation sequences and visiting intermediate states. It is called the *forward* algorithm because it works from the start of an observation sequence to its end.

Let $\alpha^t(j)$ denote the joint probability at time t of the state x_j and the partial observation sequence $y^{1:t}$: *i.e.*, $\alpha^t(j) = P[y^{1:t}, X^t = x_j]$. The probability $P[y^{1:t}]$ can be computed by marginalizing X^t : *i.e.*, sum the joint probabilities of observing this sequence and visiting state x_j at time t , for all j :

$$\begin{aligned} P[y^{1:t}] &= \sum_j P[y^{1:t}, X^t = x_j] \\ &= \sum_j \alpha^t(j) \end{aligned}$$

In particular, $P[y] = \sum_j \alpha^T(j)$.

The forward algorithm computes α values inductively as follows: for all states j ,

$$\begin{aligned} \alpha^0(j) &= \pi(x_j) \\ \alpha^{t+1}(j) &= \sum_i \alpha^t(i) a_{ij} b_{ij}(y^{t+1}), \quad \text{for } 0 \leq t \leq T-1 \end{aligned}$$

In particular, $\alpha^0(j)$ is defined as the initial probability of state x_j , and $\alpha^{t+1}(j)$ is computed as follows: (i) look up the value $\alpha^t(i)$ for arbitrary state x_i : *i.e.*, the joint probability of observing sequence $(y^{1:t})$ and visiting state x_i at time t ; (ii) look up the probability a_{ij} of transitioning from state x_i to state x_j ; (iii) look up the probability of observing y^{t+1} upon transitioning from state x_i to state x_j ; and (iv) sum the product $\alpha^t(i) a_{ij} b_{ij}(y^{t+1})$ for all i .

Table 1: Forward Algorithm: What is the probability of “THT”?

Time	$\alpha^t(1)$	$\alpha^t(2)$	$\alpha^t(3)$	Sum: $P[y^{1:t}]$
0	1/3	1/3	1/3	1
1	0	0 + 1/12	0 + 1/12 + 2/9	14/36 = 0.3889
2	0	0 + 1/48	0 + 1/48 + 11/108	62/432 = 0.1435
3	0	0 + 1/192	0 + 1/192 + 106/1296	478/5184 = 0.0922

The complexity of the forward algorithm is $O(TN^2)$ (Why?).

4.2 The Backward Algorithm

The *backward algorithm* is another algorithm for computing the quantity $P[y]$. Like the forward algorithm, it is a dynamic programming algorithm. But unlike the forward algorithm, it works from the end of an observation sequence to its start, computing the probabilities of partial observation sequences in terms of the probabilities of shorter observation sequences, *given* intermediate states.

As above the probability $P[y^{t+1:T}]$ can be computed by marginalizing X^t : *i.e.*, sum the joint

probabilities of observing this sequence and visiting state x_i at time t , for all i :

$$\begin{aligned} P[y^{t+1:T}] &= \sum_i P[y^{t+1:T}, X^t = x_i] \\ &= \sum_i P[y^{t+1:T} | X^t = x_i] P[X^t = x_i] \\ &= \sum_i \beta^t(i) P[X^t = x_i] \end{aligned}$$

where $\beta^t(i)$ denotes the probability of the observation sequence $y^{t+1:T}$, given state x_i is visited at time t : *i.e.*, $\beta^t(i) = P[y^{t+1:T} | X^t = x_i]$. But since $P[X^0 = x_i] = \pi(x_i)$, it follows that $P[y] = \sum_i \beta^0(i) \pi(x_i)$.

The β values are computed via backward induction as follows: for all states i ,

$$\begin{aligned} \beta^T(i) &= 1 \\ \beta^{t-1}(i) &= \sum_j a_{ij} b_{ij}(y^t) \beta^t(j), \text{ for } T \geq t \geq 1 \end{aligned}$$

In particular, $\beta^T(i)$ is initialized to 1, and $\beta^t(i)$ is computed as follows: (i) look up the probability a_{ij} of transitioning from state x_i to arbitrary state x_j ; (ii) look up the probability of observing y^t upon transitioning from state x_i to state x_j ; (iii) look up the value $\beta^t(j)$ for state x_j : the probability of observing sequence $y^{t+1:T}$, given state x_j is visited at time t ; and (iv) sum the product $a_{ij} b_{ij}(y^t) \beta^t(j)$ for all j .

The complexity of the backward algorithm is $O(TN^2)$ (Why?).

4.3 The Forward-Backward Algorithm

The forward-backward algorithm caches some combination of forward (α) and backward (β) values.

$$\begin{aligned} P[y, X^t = x_i] &= P[y^{1:t}, X^t = x_i, y^{t+1:T}] \\ &= P[y^{1:t}, X^t = x_i] P[y^{t+1:T} | y^{1:t}, X^t = x_i] \\ &= P[y^{1:t}, X^t = x_i] P[y^{t+1:T} | X^t = x_i] \\ &= \alpha^t(i) \beta^t(i) \end{aligned}$$

Thus, by marginalization, $P[y] = \sum_i P[y, X^t = x_i] = \sum_i \alpha^t(i) \beta^t(i)$, for all t .

Table 2: Backward Algorithm: What is the probability of “THT”?

Time	$\beta^t(1)$	$\beta^t(2)$	$\beta^t(3)$	Average: $P[y^{t+1:t}]$
3	1	1	1	1
2	0 + 0 + 0	1/4 + 1/4	2/3	7/18 = 0.3889
1	0 + 1/6 + 2/9	1/8 + 1/6	2/9	65/216 = 0.3009
0	0 + 0 + 0	7/96 + 1/18	4/27	239/2592 = 0.0922

Table 3: Forward-Backward Algorithm: What is the probability of “THT”?

Time	$\alpha^t(1)\beta^t(1)$	$\alpha^t(2)\beta^t(2)$	$\alpha^t(3)\beta^t(3)$	Sum: $P[y]$
0	(1/3)(0)	(1/3)(.1285)	(1/3)(.1481)	0.0922
1	(0)(0.3889)	(0.0833)(0.2917)	(0.3056)(0.2222)	0.0922
2	(0)(0)	(0.0208)(1/2)	(0.1227)(2/3)	0.0922
3	(0)(1)	(0.0052)(1)	(.0867)(1)	0.0922

5 The Decoding Problem

Recall the statement of the decoding problem: find a path $x \in \arg \max_x P[x, y]$. Equivalently, and perhaps more intuitively, find a path $x \in \arg \max_x P[y|x]P[x]$.

The *Viterbi* algorithm uses dynamic programming to solve the decoding problem by finding longer and longer subpaths: for all $t = 0, \dots, T$,

$$x^{0:t} \in \arg \max_{\{x^{0:t}\}} P[x^{0:t}, y^{1:t}]$$

Define $\sigma^t(j) = \max_{x^{0:t-1}} P[x^{0:t-1}, y^{1:t}, X^t = x_j]$. In words, $\sigma^t(j)$ is the value of the best possible path to state j at time t , where the “best” possible path is that which maximizes the joint probability of $x^{0:t}$ and $y^{1:t}$, terminating at time t at state j . The Viterbi algorithm computes these values inductively as follows:

$$\begin{aligned} \sigma^0(j) &= \pi(x_j) \\ \sigma^{t+1}(j) &= \max_i \sigma^t(i) a_{ij} b_{ij}(y^{t+1}), \quad \text{for } 0 \leq t \leq T-1 \end{aligned}$$

Simultaneously, the algorithm stores the states that comprise a most likely path:

$$\tau^{t+1}(j) = \arg \max_i \sigma^t(i) a_{ij} b_{ij}(y^{t+1}), \quad \text{for } 0 \leq t \leq T-1$$

In words, $\tau^t(j)$ is the state at time $t-1$ on the best possible path to state j at time t . Let x_*^T denote the state j that maximizes $\sigma^T(j)$; that is, x_*^T is the state j at time T that maximizes the the joint probability of $x^{0:T}$ and $y^{1:T}$. We can compute the most likely path to x_*^T , by backtracing through the τ values, as follows:

$$\begin{aligned} x_*^T &\in \arg \max_j \sigma^T(j) \\ x_*^{t-1} &= \tau^t(x_*^t), \quad \text{for } T \geq t \geq 1 \end{aligned}$$

Table 4: Viterbi Algorithm: What is the most likely path, given “THT”?

Time	$\sigma^t(1)$	$\tau^t(1)$	$\sigma^t(2)$	$\tau^t(2)$	$\sigma^t(3)$	$\tau^t(3)$	Backtrace
0	1/3	–	1/3	–	1/3	–	3
1	0	1	$\max\{0, \mathbf{1/12}\}$	2	$\max\{0, 1/12, \mathbf{2/9}\}$	3	3
2	0	1	$\max\{0, \mathbf{1/48}\}$	2	$\max\{0, 1/48, \mathbf{2/27}\}$	3	3
3	0	1	$\max\{0, \mathbf{1/192}\}$	2	$\max\{0, 1/192, \mathbf{4/81}\}$	3	3