

Lecture 35: Hidden Markov Models (Highlights)

Charles A. Bouman

Purdue University

Monday, November 16, 2026

Outline

- 1 The structure of an HMM; the joint density and its log
- 2 The two basic tasks: state estimation and parameter estimation
- 3 MAP state sequence estimation and dynamic programming
- 4 The past/present/future decomposition and the function $L(j, n)$
- 5 The forward-backward algorithm: $\alpha_n(j)$ and $\beta_n(j)$
- 6 Training HMMs with the EM algorithm

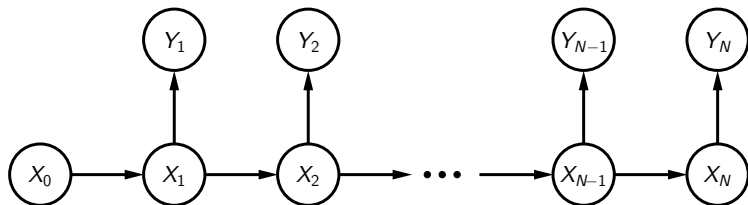
Core idea

An HMM is a Markov chain that is not observed directly: we see observations Y_n whose distribution depends on the hidden states X_n . Both tasks are solved by breaking the problem into **past, present, and future**: that decomposition turns a search over M^{N+1} state sequences into computation of order NM^2 .

Today's Big Ideas

- A hidden Markov model is doubly stochastic, which means that an unobserved Markov chain of discrete states controls the distribution of the observations that we do see.
- Two tasks come up in practice: estimating the hidden states from the observations, and estimating the model parameters from the observations.
- Breaking the state estimation problem into past, present, and future replaces a search over exponentially many sequences with a recursion whose cost grows linearly with the sequence length.
- The same past, present, and future split gives the forward-backward algorithm, which computes the probability of each individual state and of each pair of neighboring states at the same linear cost.
- Training a hidden Markov model uses the EM algorithm, which replaces the sufficient statistics of the complete data problem with the conditional expectations that forward-backward supplies.

The structure of an HMM



This diagram illustrates the dependency of quantities in an HMM. Notice that the values X_n form a Markov chain in time, while the observed values, Y_n , depend on the corresponding labels, X_n .

- The states are $\{X_n\}_{n=0}^N$ and the observations are $\{Y_n\}_{n=1}^N$. The state values determine the distribution of the corresponding observations.

Doubly stochastic, and where HMMs are used

Much like in the case of the Gaussian mixture distribution, the labels X_n are typically not observed in the real application, but we can imagine that their existence explains the changes in behavior of Y_n over long time scales.

Doubly stochastic (same term as in Section 12.2)

The HMM is **doubly stochastic** because the unobserved stochastic process, X_n , controls the observed stochastic process, Y_n .

- HMMs are very useful in applications such as audio or speech processing.
- There, the observations Y_n are typically continuously valued feature vectors describing the local sound properties, and the discrete random variables X_n represent the underlying state of the audio signal.
- In speech applications, X_n might be a label that represents the specific phoneme or other sound that is being voiced in the pronunciation of a word.

Notation, joint density, and log likelihood

Let the density function for Y_n given X_n be

$$f(y|k) = P\{Y_n \in dy | X_n = k\} ,$$

and let the Markov chain X_n be parameterized by $\theta = [\tau_j, P_{i,j}]$ for $i, j \in \Omega$. Then the joint density function for the sequences Y and X is given by

$$p(y, x|\theta) = \tau_{x_0} \prod_{n=1}^N \{f(y_n|x_n)P_{x_{n-1},x_n}\} ,$$

and assuming that no quantities are zero, the log likelihood is given by

$$\log p(y, x|\theta) = \log \tau_{x_0} + \sum_{n=1}^N \{\log f(y_n|x_n) + \log P_{x_{n-1},x_n}\} .$$

(This is actually a mixed probability density/mass function. This is fine as long as one remembers to integrate over y and sum over x .)

The two basic tasks

One typically needs to perform two basic tasks using HMMs.

- 1 **State estimation.** Estimate the unknown states, X_n , from the observed data, Y_n .
- 2 **Parameter estimation.** Estimate the parameters, θ , from the observations, Y_n .

How they are solved

The following sections explain how these problems can be solved using **dynamic programming** and the **EM algorithm** of Chapter 12.

MAP state sequence estimation

When using HMMs, it is often important to be able to estimate the unobserved discrete state sequence, X , from the full set of observations, Y . This problem is known as **state sequence estimation**.

While different estimators present different strengths and weaknesses, one commonly used estimate for the state sequence is the MAP estimate given by

$$\begin{aligned}\hat{x} &= \arg \max_{x \in \Omega^N} p(x|y, \theta) \\ &= \arg \max_{x \in \Omega^N} \log p(y, x|\theta) .\end{aligned}$$

When we expand out the MAP optimization problem using the log likelihood of the HMM, it is not obvious that it can be tractably solved.

Why it looks intractable

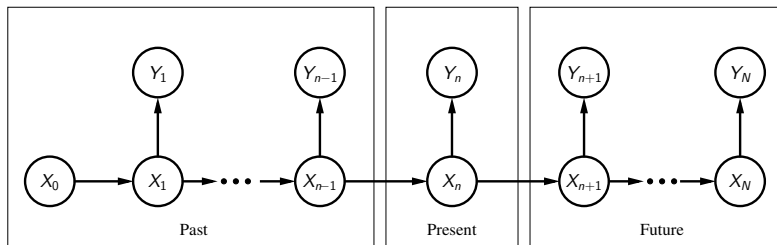
$$\hat{x} = \arg \max_{x \in \Omega^N} \left\{ \log \tau_{x_0} + \sum_{n=1}^N \{ \log f(y_n | x_n) + \log P_{x_{n-1}, x_n} \} \right\} .$$

The headline

On its face, this optimization is an extraordinarily difficult problem, requiring a search over a state space of size M^{N+1} . However, the problem's complexity can be dramatically reduced by decomposing it into components corresponding to the **past, present, and future** values of the states and observations.

The next figure visually represents how these observations and states are related, and includes a table defining some new notation for the problem. This notation will allow us to more compactly express the probabilities of past, present, and future events.

Past, present, and future in an HMM



	Past	Present	Future
Observation	$y_{<n} = \{y_1, \dots, y_{n-1}\}$	y_n	$y_{>n} = \{y_{n+1}, \dots, y_N\}$
State	$x_{<n} = \{x_0, \dots, x_{n-1}\}$	x_n	$x_{>n} = \{x_{n+1}, \dots, x_N\}$

The boxes represent aggregated quantities corresponding to the past, present, or future, while the circles represent individual states or observations. The arrows indicate conditional dependence in our assumed model.

The past/present/future factorization

Using this new notation and the assumed dependencies of the HMM, we can write the joint distribution of the observations and states as the product

$$p(y, x) = p(y_{>n}, x_{>n} | x_n) f(y_n | x_n) p(y_{<n}, x_n, x_{<n}) ,$$

where the conditional distributions have the following intuitive interpretations:

$$p(y_{>n}, x_{>n} | x_n) = P(\text{Future} | \text{present state}) ,$$

$$f(y_n | x_n) = P(\text{Present observation} | \text{present state}) ,$$

$$p(y_{<n}, x_n, x_{<n}) = P(\text{Past along with present state}) .$$

$L(j, n)$ and the dynamic programming recursion

The function $L(j, n)$ is a solution to the future portion of the problem, given an assumed value for $x_n = j$, and it is defined by

$$L(j, n) = \max_{x_{>n}} \{ \log p(y_{>n}, x_{>n} | x_n = j) \} .$$

So the function $L(x_n, n)$ specifies the maximum log probability of the future states and observations given that the present state at time n is known to be x_n . From this definition, we can work out the following recursive relationship for $L(i, n - 1)$ in terms of a maximization involving $L(j, n)$:

$$L(i, n - 1) = \max_{j \in \Omega} \{ \log f(y_n | j) + \log P_{i,j} + L(j, n) \} .$$

Dynamic programming

The application of this recursion, along with the additional boundary constraint that $L(j, N) = 0$, is a special case of a more general optimization strategy known as **dynamic programming**.

Complexity, and recovering the states

Using this approach, the values of $L(i, n)$ can be computed for $0 \leq n \leq N$ and $i \in \Omega$ with order- NM^2 computational complexity. The complexity results because for each value of n and for each value of i , we must maximize over M possible values of j .

Of course, the values of $L(i, n)$ are not what we actually need. What we need to know are the values of the states, x_n , that maximize the likelihood. However, with the stored values of $L(i, n)$ from the dynamic programming recursion, we can easily compute the states using the relationships

$$\begin{aligned}\hat{x}_0 &= \arg \max_{j \in \Omega} \{ \log \tau_j + L(j, 0) \} , \\ \hat{x}_n &= \arg \max_{j \in \Omega} \{ \log P_{\hat{x}_{n-1}, j} + \log f(y_n | j) + L(j, n) \} .\end{aligned}$$

MAP sequence estimation: pseudocode

MAP Sequence Estimation:

For $i \in \Omega$, $L(i, N) = 0$

For $n = N$ to 1 {

For $i \in \Omega$,

$$L(i, n-1) = \max_{j \in \Omega} \{ \log f(y_n | j) + \log P_{i,j} + L(j, n) \}$$

}

For $i \in \Omega$, $\hat{x}_0 = \arg \max_{j \in \Omega} \{ \log \tau_j + L(j, 0) \}$

For $n = 1$ to N {

For $i \in \Omega$,

$$\hat{x}_n = \arg \max_{j \in \Omega} \{ \log P_{\hat{x}_{n-1}, j} + \log f(y_n | j) + L(j, n) \}$$

}

A pseudocode specification of the dynamic programming algorithm used for MAP state sequence estimation of HMMs.

State probabilities: $\alpha_n(j)$ and $\beta_n(j)$

While the MAP estimate of the state sequence, x_n , is useful, in some cases we may need to know the actual distribution of individual states. In particular, we will need to know the joint distribution of pairs of states, (x_{n-1}, x_n) , in order to **train** HMM models.

Again, the trick is to break up the observation and state variables into their past, present, and future components. To do this, we define the two functions

$$\begin{aligned}\alpha_n(j) &= p(x_n = j, y_n, y_{<n}) , \\ \beta_n(j) &= p(y_{>n} | x_n = j) .\end{aligned}$$

The three probability formulas

If we can compute these two functions, then the joint probability of the observations and x_n can be expressed simply as

$$p(x_n = j, y) = \alpha_n(j)\beta_n(j) ,$$

and from this it is easy to compute the conditional density of x_n as

$$p(x_n = j|y) = \frac{\alpha_n(j)\beta_n(j)}{p(y)} ,$$

where $p(y)$ is just a normalizing constant given by

$$p(y) = \sum_{j \in \Omega} \alpha_n(j)\beta_n(j) = \sum_{j \in \Omega} \alpha_0(j)\beta_0(j) .$$

We can similarly compute the joint conditional density of (x_{n-1}, x_n) as

$$p(x_{n-1} = i, x_n = j|y) = \frac{\alpha_{n-1}(i)P_{i,j}f(y_n|j)\beta_n(j)}{p(y)} .$$

The forward and backward recursions

The efficient algorithm for computing the functions $\alpha_n(j)$ and $\beta_n(j)$ is known as the **forward-backward algorithm**. It takes its name from the fact that $\alpha_n(j)$ is computed using a forward recursion and $\beta_n(j)$ is computed using a backward recursion. The overall flow is very similar to that of the dynamic programming technique of the previous section.

$$\begin{aligned}\alpha_n(j) &= \sum_{i \in \Omega} \alpha_{n-1}(i) P_{i,j} f(y_n | j) , & \alpha_0(j) &= \tau_j , \\ \beta_n(i) &= \sum_{j \in \Omega} P_{i,j} f(y_{n+1} | j) \beta_{n+1}(j) , & \beta_N(i) &= 1 .\end{aligned}$$

Complexity

Again, the total computation required for this algorithm is order NM^2 because one must compute the functions for all n and i , as well as the sum of the index j .

Forward-backward algorithm: pseudocode (1 of 2)

Forward-Backward Algorithm:

// forward step

For $j \in \Omega$, $\alpha_0(j) = \tau_j$

For $n = 1$ to N {

For $j \in \Omega$,

$$\alpha_n(j) = \sum_{i \in \Omega} \alpha_{n-1}(i) P_{i,j} f(y_n | j)$$

}

// backward step

For $i \in \Omega$, $\beta_N(i) = 1$

For $n = N$ to 1 {

For $i \in \Omega$,

$$\beta_n(i) = \sum_{j \in \Omega} P_{i,j} f(y_{n+1} | j) \beta_{n+1}(j)$$

}

Forward-backward algorithm: pseudocode (2 of 2)

Forward-Backward Algorithm (continued):

// calculate probabilities

For $n = 1$ to N and $i, j \in \Omega$, {

$$p(y) = \sum_{i \in \Omega} \alpha_0(i) \beta_0(i)$$

$$P\{X_n = i | Y = y, \theta\} = \frac{\alpha_n(i) \beta_n(i)}{p(y)}$$

$$P\{X_n = j, X_{n-1} = i | Y = y, \theta\} = \frac{\alpha_{n-1}(i) P_{i,j} f(y_n | j) \beta_n(j)}{p(y)}$$

}

This algorithm computes the marginal and conditional distributions of states in an HMM.

Training HMMs: why EM is needed

HMMs are only useful if they accurately model the behavior of real physical processes. In practice, this fitting is accomplished by estimating the parameters of the HMM from typical examples. These example sequences are known as **training data**.

- Ideally, the training data should consist of sequences of observations, Y_n , along with the hidden states, X_n . This might be possible in cases where the exact states can be measured at additional cost.
- However, in many cases, the unknown states, X_n , may not be available at any cost. In speech modeling, X_n could represent the phoneme being voiced and Y_n the spectral features of the auditory signal; there is no way to directly measure the phoneme being voiced.

So any training must estimate the parameters of the HMM model while simultaneously estimating the unknown states, X_n . This is a classic example for which the **EM algorithm** is perfectly suited.

The model and its natural sufficient statistics

In a typical application, the observed quantity, Y_n , is a p -dimensional multivariate Gaussian random vector with conditional distribution $N(\mu_{x_n}, R_{x_n})$, so the complete set of parameters is $\theta = [\mu_j, R_j, \tau_j, P_{i,j} \text{ for } i, j \in \Omega]$.

In this case it can be shown that the joint distribution, $p(x, y|\theta)$, is an exponential distribution with natural sufficient statistics given by

$$\begin{aligned}b_j &= \sum_{n=1}^N y_n \delta(x_n - j) , \\S_j &= \sum_{n=1}^N y_n y_n^t \delta(x_n - j) , \\N_j &= \delta(x_0 - j) , \\K_{i,j} &= \sum_{n=1}^N \delta(x_n - j) \delta(x_{n-1} - i) ,\end{aligned}$$

and another useful statistic, K_j , can be computed from these as

$$K_j = \sum_{i \in \Omega} K_{i,j} = \sum_{n=1}^N \delta(x_n - j).$$

The complete-data ML estimates

Referring back to the ML estimates for multivariate Gaussian distributions and the ML estimate derived for Markov chains in Section 13.2, we can write the ML estimate of the entire parameter vector, θ , given both X and Y :

$$\begin{aligned}\hat{\mu}_j &= \frac{b_j}{K_j} , \\ \hat{R}_j &= \frac{S_j}{K_j} - \frac{b_j b_j^t}{K_j^2} , \\ \hat{\tau}_j &= N_j , \\ \hat{P}_{i,j} &= \frac{K_{i,j}}{\sum_{j \in \Omega} K_{i,j}} .\end{aligned}$$

Remember, these are the closed-form ML estimates that we compute when we can observe the states X_n directly. So in this case, the problem is not incomplete.

The E-step: expected sufficient statistics

The rule from Section 12.7 (Lecture 33)

The EM update equations directly result from replacing the natural sufficient statistics of the complete data problem with their expected values.

So to compute the EM update of the parameter θ , we first compute the conditional expectation of the sufficient statistics in the E-step as

$$\begin{aligned}\bar{b}_j &\leftarrow \sum_{n=1}^N y_n P\{X_n = j | Y = y, \theta\} , \\ \bar{S}_j &\leftarrow \sum_{n=1}^N y_n y_n^t P\{X_n = j | Y = y, \theta\} , \\ \bar{N}_j &\leftarrow P\{X_0 = j | Y = y, \theta\} , \\ \bar{K}_{i,j} &\leftarrow \sum_{n=1}^N P\{X_n = j, X_{n-1} = i | Y = y, \theta\} , \\ \bar{K}_j &\leftarrow \sum_{n=1}^N P\{X_n = j | Y = y, \theta\} .\end{aligned}$$

The M-step, and where the posteriors come from

Then the HMM model parameters, $\theta = [\mu_j, R_j, \tau_j, P_{i,j}]$ for $i, j \in \Omega$, are updated using the M-step

$$\begin{aligned}\hat{\mu}_j &\leftarrow \frac{\bar{b}_j}{\bar{K}_j}, \\ \hat{R}_j &\leftarrow \frac{\bar{S}_j}{\bar{K}_j} - \frac{\bar{b}_j \bar{b}_j^t}{\bar{K}_j^2}, \\ \hat{\tau}_j &\leftarrow \bar{N}_j, \\ \hat{P}_{i,j} &\leftarrow \frac{\bar{K}_{i,j}}{\sum_{j \in \Omega} \bar{K}_{i,j}}.\end{aligned}$$

The closing point

Notice that the E-step requires computing the posterior probability of X_n given Y . Fortunately, this may be easily computed using the **forward-backward algorithm** of the previous section.

Summary

- An HMM is doubly stochastic: the unobserved Markov chain X_n controls the observed process Y_n , with $p(y, x|\theta) = \tau_{x_0} \prod_{n=1}^N \{f(y_n|x_n)P_{x_{n-1}, x_n}\}$.
- Two basic tasks: estimate the states, and estimate the parameters. They are solved by dynamic programming and by EM.
- MAP state estimation looks like a search over M^{N+1} sequences, but the past/present/future decomposition and the function $L(j, n)$ reduce it to order NM^2 .
- The forward-backward algorithm computes $\alpha_n(j) = p(x_n = j, y_n, y_{<n})$ and $\beta_n(j) = p(y_{>n}|x_n = j)$, also in order NM^2 , giving the state and state-pair distributions.
- EM training replaces the natural sufficient statistics $b_j, S_j, N_j, K_{i,j}, K_j$ with their conditional expectations; the posteriors needed in the E-step are exactly what forward-backward computes.

Reading: Section 13.3. Next lecture: Section 13.4, Stationary Distributions of Markov Chains.