

5.6 Hidden Markov Models

If the state x_t is **discrete**, then state-space models are often called *hidden Markov models* (HMMs).

In this case, the state-equation is replaced by a *Markov chain* model: if $\{X_t\}$ can take values on $\{1, 2, \dots, S\}$ say, then for $t > 1$

$$p(x_t|x_{1:(t-1)}) \equiv p(x_t|x_{t-1})$$

where

$$\Pr[X_t = l | X_{t-1} = k] = p_{kl}$$

The $S \times S$ transition matrix P , along with the (marginal) distribution of X_1 ($\mathbb{P}[X_1 = k], k = 1, \dots, S$) characterize the nature of the state evolution. Note that P is a *stochastic matrix* whose rows sum to 1.

5.6 Hidden Markov Models

All of the previous types of calculation

- filtering $p(x_t|y_{1:t})$
- prediction $p(x_t|y_{1:(t-1)})$
- smoothing $p(x_t|y_{1:n})$
- parameter estimation

are required. The main difference is that the filtering, prediction and smoothing distributions are **discrete**, and therefore somewhat more straightforward to compute.

The observation variable y_t can be either discrete or continuous.

5.6 Hidden Markov Models

Example ($S = 2$)

Suppose for $t = 1, 2, \dots, n$

$$y_t = \mu(x_t) + \epsilon_t$$

and $\{X_t\}$ follows a Markov chain with transition matrix

$$P = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix}$$

and the distribution of X_1 is the stationary (or equilibrium) distribution, π , of the chain defined as the solution of

$$\pi = \pi P$$

5.6 Hidden Markov Models

Example ($S = 2$)

We compute π (a 1×2 vector) by

$$[1, 1]^\top \pi = \lim_{t \rightarrow \infty} P^t = \begin{bmatrix} 4/11 & 7/11 \\ 4/11 & 7/11 \end{bmatrix}$$

so $\pi = (4/11, 7/11)$.

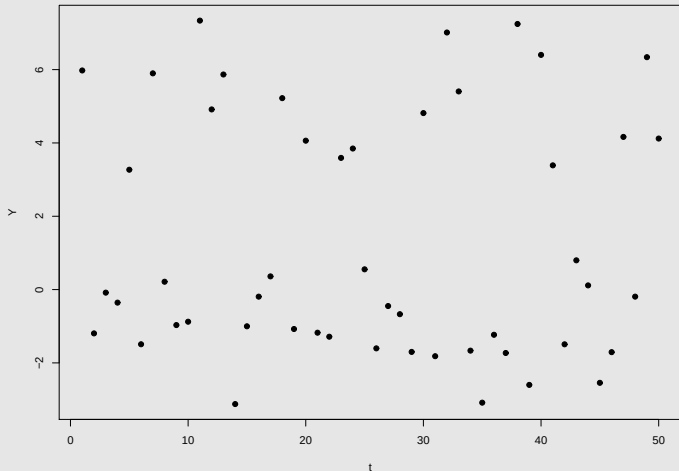
Suppose

$$\mu(x) = \begin{cases} \mu_1 & x = 1 \\ \mu_2 & x = 2 \end{cases}$$

and $\{\epsilon_t\} \sim \mathcal{N}(0, 1)$, IID.

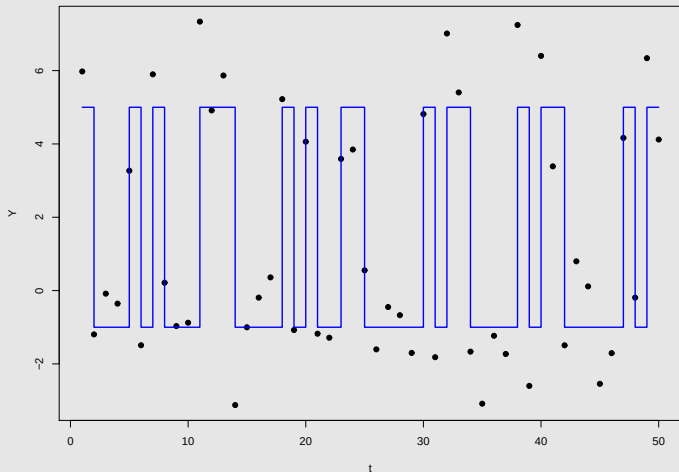
5.6 Hidden Markov Models

Example ($S = 2$)



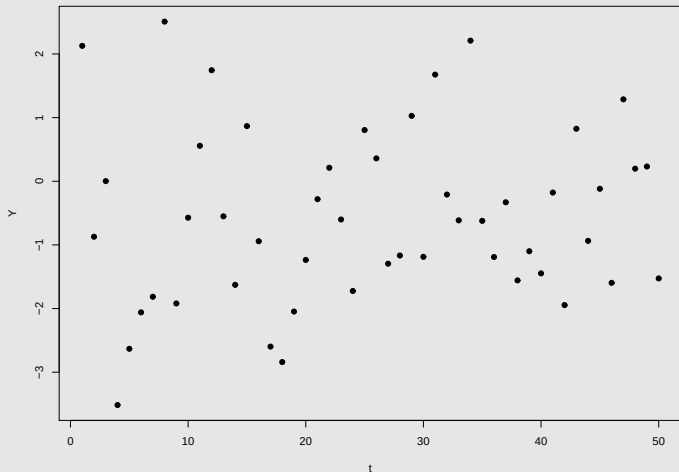
5.6 Hidden Markov Models

Example ($S = 2, \mu_1 = 5, \mu_2 = -1$)



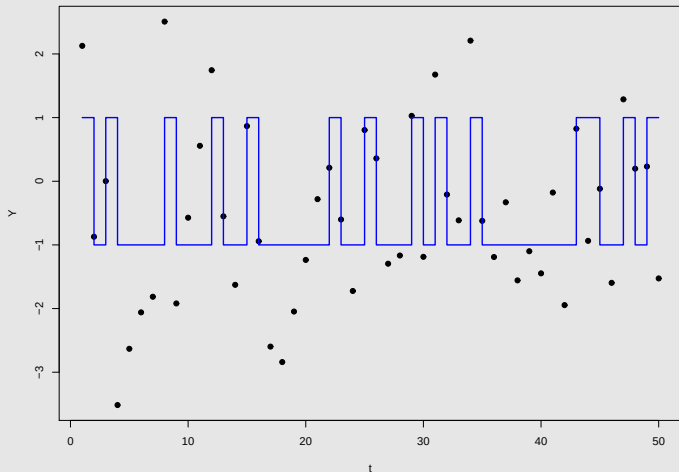
5.6 Hidden Markov Models

Example ($S = 2$)



5.6 Hidden Markov Models

Example ($S = 2, \mu_1 = 1, \mu_2 = -1$)



5.6 Hidden Markov Models

Filtering: By Bayes theorem

$$\begin{aligned} p(x_t|y_{1:t}) &= p(x_t|y_t, y_{1:(t-1)}) = \frac{p(y_t|x_t, y_{1:(t-1)})p(x_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \\ &= \frac{p(y_t|x_t)p(x_t|y_{1:(t-1)})}{\sum_{x=1}^S p(y_t|x)p(x|y_{1:(t-1)})} \\ &\propto p(y_t|x_t)p(x_t|y_{1:(t-1)}) \end{aligned}$$

This is a discrete distribution on $\{1, \dots, S\}$ that we can compute exactly.

5.6 Hidden Markov Models

Prediction: By the theorem of total probability

$$p(x_{t+1}|y_{1:t}) = \sum_{x=1}^S p(x_{t+1}|x)p(x|y_{1:t})$$

- $p(x_{t+1}|x)$ is determined the x th row of P , for $x = 1, 2, \dots, S$
- $p(x|y_{1:t})$ is the previous filtering distribution.

5.6 Hidden Markov Models

Smoothing: By the theorem of total probability, working backwards $t = n, n - 1, \dots, 1$,

$$p(x_t|y_{1:n}) = \sum_{x_{t+1}=1}^S p(x_t|x_{t+1}, y_{1:n})p(x_{t+1}|y_{1:n})$$

$p(x_{t+1}|y_{1:n})$ is the previous smoothing distribution. For the first term

$$\begin{aligned} p(x_t|x_{t+1}, y_{1:n}) \equiv p(x_t|x_{t+1}, y_{1:t}) &= \frac{p(x_t|y_{1:t})p(x_{t+1}|x_t, y_{1:t})}{p(x_{t+1}|y_{1:t})} \\ &= \frac{p(x_t|y_{1:t})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} \end{aligned}$$

which can be computed from the filtering and prediction distributions.

Discrete HMMs

The HMM structure is commonly implemented when the observation is also **discrete**.

As usual let $x_{1:n}$ denote the **hidden** state sequence, and $y_{1:n}$ denote the **observed** data sequence. Then the model is specified by

- **transition probabilities**

$$p_{kl} = \Pr [x_t = l | x_{t-1} = k]$$

with initial distribution $\mathbb{P}[X_1 = k], k = 1, \dots, S$.

- **emission probabilities**

$$e_k(m) = \Pr [y_t = m | x_t = k]$$

$e_k(m)$ can be specified for a finite set ($m \in \{1, 2, \dots, M\}$), or on the integers.

The Viterbi Algorithm:

We seek the most probable state path:

$$\begin{aligned}x_{1:n}^* &= \arg \max_{x_{1:n}} p(x_{1:n}, y_{1:n}) \\ &= \arg \max_{x_{1:n}} p(x_{1:n} | y_{1:n})\end{aligned}$$

That is, we seek the most probable path or sequence of states in light of the data

- $p(x_{1:n}, y_{1:n})$ is the joint probability or joint likelihood
- $p(x_{1:n} | y_{1:n})$ is the posterior probability

Discrete HMMs: The Viterbi Algorithm

- (i) Initialize vector $v_k(0) = \mathbb{P}[X_1 = k], k = 1, \dots, S$.
- (ii) Recursion: for $t = 1, \dots, n$, compute for each $l = 1, 2, \dots, S$.

$$\begin{aligned}v_l(t) &= e_l(y_t) \max_k (v_k(t-1) p_{kl}) \\point_t(l) &= \arg \max_k (v_k(t-1) p_{kl})\end{aligned}$$

- (iii) Termination

$$\begin{aligned}p(y_{1:n}, x_{1:n}^*) &= \max_k (v_k(n)) \\x_n^* &= \arg \max_k (v_k(n))\end{aligned}$$

- (iv) Trace back the optimal path

$$x_{t-1}^* = point_t(x_t^*) \quad t = n, n-1, \dots, 2.$$

Discrete HMMs: The Forward Algorithm

The probability of an observed sequence (or likelihood) is

$$p(y_{1:n}) = \sum_{x_{1:n}} p(y_{1:n}, x_{1:n})$$

However, the direct computation of this sum is not possible, so a recursive calculation is needed.

- (i) Initialize vector $f_0(0) = 1, f_k(0) = 0$ for $k = 1, 2, \dots, S$
- (ii) Recursion: for $t = 1, \dots, n$, and $l = 1, 2, \dots, S$,

$$f_l(t) = e_l(y_t) \sum_{k=1}^S f_k(t-1) p_{kl}$$

- (iii) Final recursion: without loss of generality, assuming that the final state is labelled 1

$$p(y_{1:n}) = \sum_{k=1}^S f_k(n).$$

Discrete HMMs: The Backward Algorithm

We seek the posterior (smoothing) probabilities

$$\begin{aligned} p(x_t = k | y_{1:n}) &= \frac{p(x_t = k, y_{1:n})}{p(y_{1:n})} \\ &= \frac{p(y_1, \dots, y_t, x_t = k) p(y_{t+1}, \dots, y_n | x_t = k)}{p(y_{1:n})} \end{aligned}$$

for $k = 1, 2, \dots, S$.

Direct computation of this sum is not possible, so a recursive calculation is needed, after completing the Forward Algorithm.

Discrete HMMs: The Backward Algorithm

The backward algorithm works in reverse order through the data:

- (i) Initialize element $b_k(n) = f_k(n)$ for $k = 1, \dots, S$.
- (ii) Backwards recursion: for $t = n - 1, \dots, 1$ compute

$$b_k(t) = \sum_{l=1}^S p_{kl} e_l(y_{t+1}) b_l(t+1)$$

for $k = 1, 2, \dots, S$.

Discrete HMMs: The Backward Algorithm

- (iii) It can be shown that the forward and backward probabilities, $f_k(t)$ and $b_k(t)$, are in fact the probabilities

$$\begin{aligned}f_k(t) &= p(y_1, \dots, y_t, x_t = k) \\b_k(t) &= p(y_{t+1}, \dots, y_n | x_t = k)\end{aligned}$$

Thus we have

$$p(x_t = k, y_{1:n}) = f_k(t)b_k(t)$$

and hence, for $k = 1, \dots, S$,

$$p(x_t = k | y_{1:n}) = \frac{f_k(t)b_k(t)}{p(y_{1:n})}$$

Discrete HMMs: Estimation

Parameter Estimation: The Baum-Welch Algorithm:

The algorithms above are all expressed in terms of known transition and emission probabilities. These probabilities can be pre-specified, estimated from observed sequences, or estimated during the state sequence reconstruction.

If the state sequence is **known**, let n_{kl} be the number of transitions from state k to state l , and let $r_k(b)$ be the number of times character b is emitted from state k . Then we may estimate the transition and emission probabilities as follows

$$\hat{p}_{kl} = \frac{n_{kl}}{\sum_{l'} n_{kl'}} \quad \hat{e}_k(b) = \frac{r_k(b)}{\sum_{b'} r_k(b')} \quad (1)$$

Discrete HMMs: The Baum-Welch Algorithm

If the state sequence is **not known**, we must reconstruct the state sequence and perform estimation concurrently. This is achieved using the *Baum-Welch* algorithm, which is a special case of the *Expectation-Maximization* (EM) algorithm.

The Baum-Welch algorithm estimates n_{kl} and $r_k(b)$ as the *expected* number of transitions and emissions, computed using the forward and backward algorithms.

Discrete HMMs: The Baum-Welch Algorithm

The probability of a transition $k \rightarrow l$ at time t is

$$p(x_t = k, x_{t+1} = l | y_{1:n}) = \frac{f_k(t) p_{kl} e_l(y_{t+1}) b_l(t+1)}{p(y_{1:n})}$$

Suppose that N replicate observed sequences

$$y_{1:n}^1, \dots, y_{1:n}^N$$

are available. Then the expected number of times this transition will be observed (in light of the data) is

$$n_{kl} = \sum_{i=1}^N \left\{ \frac{\sum_t f_k^i(t) a_{kl} e_l(y_{t+1}^i) b_l^i(t+1)}{p(y_{1:n}^i)} \right\}$$

Discrete HMMs: Baum-Welch Algorithm

In this formula

- ▶ $y_{1:n}^i$ is the i^{th} observed sequence
- ▶ $f_k^i(t)$ is the forward probability from the forward algorithm implemented for sequence i
- ▶ $b_l^i(t)$ is the backward probability from the backward algorithm implemented for sequence i

Similarly,

$$r_k(b) = \sum_{i=1}^N \left\{ \frac{\sum_{t \in \mathcal{B}_i} f_k^i(t) b_l^i(t)}{p(y_{1:n}^i)} \right\}$$

where $\mathcal{B}_i$ is the set of indices of locations in replicate sequence i where character b is emitted.

Discrete HMMs: Baum-Welch Algorithm

- (i) Initialize parameter values to some arbitrary values
- (ii) Recursion: Set $n_{kl}^{(0)}, r_k^{(0)}(b)$ to be some small “prior” count value, then for $i = 1, \dots, N$:
 - Calculate $f_k^i(t)$ using the Forward Algorithm, and $b_k^i(t)$ using the Backward Algorithm, then set

$$n_{kl}^{(i)} = n_{kl}^{(i-1)} + \frac{1}{p(y_{1:n}^i)} \left\{ \sum_{t=1}^n f_k^i(t) p_{kl} e_l(y_{t+1}^i) b_l^i(t+1) \right\}$$

$$r_k^{(i)}(b) = r_k^{(i-1)}(b) + \frac{1}{p(y_{1:n}^i)} \left\{ \sum_{t \in \mathcal{B}_i} f_k^i(t) b_l^i(t) \right\}$$

- (iii) Estimate p_{kl} and $e_k(b)$ using (1), and likelihood $\prod_{i=1}^n p(y_{1:n}^i)$
- (iv) Terminate the recursion if the computed likelihood value compared to the previous iteration is virtually unchanged.

Computation for continuous response HMMs

Note that most of the above calculations carry through to the continuous emission case, that is if the emission probabilities

$$e_k(m) = \Pr[y_t = m | x_t = k]$$

are replaced by densities

$$e_k(y) = p(y | x_t = k)$$

The forward and backward quantities become unnormalized joint densities

Computation for continuous response HMMs

To avoid underflow in the computation, the computations can either be computed on the log scale, or with continual rescaling of the calculation. For example, in the Viterbi algorithm, the computation of the $v_k(t)$ quantities will underflow quickly without rescaling, but rescaling by

$$c_t = \sum_{k=1}^S v_k(t)$$

will prevent this.

The constants c_t should be stored for future use; if the forward and backward probabilities are computed after rescaling, then this rescaling must be undone when, for example, $p(y_{1:n})$ is computed.