18.3 The backward algorithm and calculating P(x)

The backward algorithm is similar and if we run it to the end, we again calculate $P(x)$. This starts from the end of the sequence and works back to the beginning. Define

$$b_k(i) = P(x_{(i+1):L}|\pi_i = k)$$

the probability observing the last part of a sequence, $x_{i+1}, x_{i+2}, \ldots, x_L$ conditional on starting in state k at time i.

Once again, this is calculated using tabular computation:

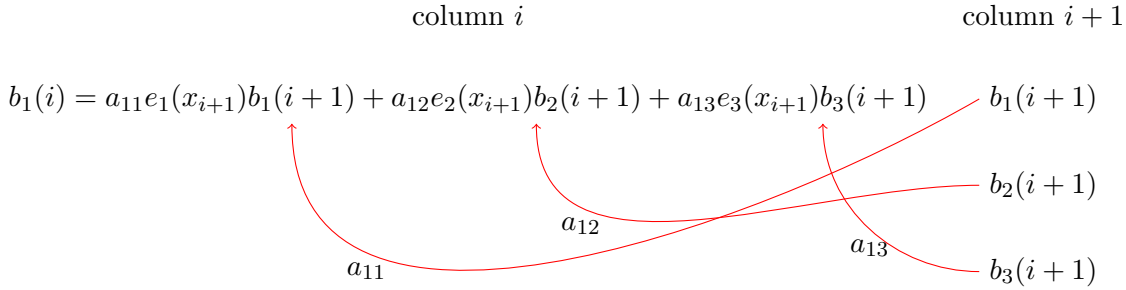
Initialisation $i = L$**:** $b_k(L) = a_{k0}$ for all k.

Recursion $i = L-1, \ldots, 1$**:** $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$.

Termination: $P(x) = \sum_l a_{0l} e_l(x_1) b_l(1)$.

If we don't model the end of the sequence, $a_{k0} = 1$.

Here's a diagram showing how to get the ith column from the $(i+1)$th column in the backward algorithm. The example shown has three states 1, 2 and 3.



The log version of the algorithm is (writing $B_k(i) = \log b_k(i)$):

Initialisation $i = L$**:** $B_k(L) = A_{k0}$ for all k (or $B_k(L) = 0$ if end not modelled)

Recursion $i = L-1, \ldots, 1$**:** $B_k(i) = \log\left[\sum_l \exp(A_{kl} + E_l(x_{i+1}) + B_l(i+1))\right]$.

Termination: $P(x) = \log\left[\sum_l \exp(A_{0l} + E_l(x_1) + B_l(1))\right]$.

18.4 The posterior probability of being in state k at time i $P(\pi_i = k|x)$

The final product of this backward algorithm is not usually what we are interested in (we use the forward algorithm to calculate that) but the combined forward and backward

algorithms allow us to calculate the joint probability of the all observations and the prob that $\pi_i = k$

$$\begin{aligned} P(x, \pi_i = k) &= P(x_{1:i}, \pi_i = k) P(x_{i+1:L} | x_{1:i}, \pi_i = k) \text{ since } P(A, B) = P(A)P(B|A) \\ &= P(x_{1:i}, \pi_i = k) P(x_{i+1:L} | \pi_i = k) \text{ by Markov property} \\ &= f_k(i) b_k(i) \end{aligned}$$

Of more interest is the posterior probability $P(\pi_i = k|x)$ which we obtain directly by

$$P(\pi_i = k|x) = \frac{f_k(i) b_k(i)}{P(x)},$$

where the denominator is calculated either from the forward or backward algorithm.

18.5 What can we do with the posterior estimates?

We saw that the Viterbi path, π^*, is the most likely single path. But usually the most likely path is not very likely at all — there may be many other paths the are nearly as likely. We can use the posterior $P(\pi_i = k|x)$ to get some other likely paths.

The first is $\hat{\pi}$, the maximum posterior path, where

$$\hat{\pi}_i = \arg\max_k P(\pi_i = k|x).$$

Note that $\hat{\pi}$ is often not a legal path through the state space as it may include transitions that are not allowed.

The second is when we are interested in some function of the states, $g(k)$. In these cases, we calculate the the posterior expectation of g at a particular position,

$$G(i|x) = E_k[g(\pi_i|x)] = \sum_k P(\pi_i = k|x) g(k).$$

In particular, if g is an indicator function, that is, g takes the value 1 for some subset of states and 0 for all others, $E_k[g(\pi_i|x)]$ is just the posterior probability that π_i is in the specified subset.

18.6 Estimating the parameters of an HMM

So far we have assumed we know the structure of the HMM and the associated parameter values (the transition probabilities a_{kl} and emission probabilities $e_k(b)$). In general, we don't know either of these. What we usually do is decide on a model (based on our knowledge of the system) and then estimate the parameters of the model. Let $\theta = \{a_{kl}, e_k(b)\}$ be the set of all parameters of the model.

Then we are interested in finding the set of parameters that maximizes the (log) likelihood

$$l(x^1, \ldots, x^n|\theta) = \log P(x^1, \ldots, x^n|\theta) = \sum_{j=1}^{n} \log P(x^j|\theta).$$

The likelihood of the jth sequence, $P(x^j|\theta)$, is just what we have been referring to as $P(x^j)$ up to this point as we had always assumed the parameter values, θ, were known. Writing it as $P(x^j|\theta)$ simply emphasises the fact that we think of it now as a function of the unknown θ.

If we knew the state paths for a long sequence (or many short sequences), we could estimate the parameters simply by using the empirical proportions of transitions and emissions as our probabilities:

$$\hat{a}_{kl} = \frac{A_{kl}}{\sum_i A_{ki}} \text{ and } \hat{e}_k(b)\frac{E_k(b)}{\sum_j E_k(j)}$$

where A and E are empirical counts. Note that, as some transitions or emissions probably wouldn't occur in smaller datasets, it is advisable add a small number of 'pseudo-counts' to the empirical counts so that none are zero). But assuming we know the state paths is unrealistic, so we proceed assuming we have only observed sequences $x^1, x^2, \ldots, x^n$.

18.7 Baum-Welch algorithm for estimating parameters of HMM

The Baum-Welch algorithm is an iterative algorithm that attempts to maximize the (log) likelihood of an HMM. Unlike earlier algorithms we have seen, it is not exact, so the estimate it finds is not guaranteed to be the best. It may also get stuck in local maxima, so different starting points are necessary.

The idea of the algorithm is to pick a starting value for $\theta = (a, e)$. Probable paths for this value of θ are found. From these probable paths, a new value for θ is found by calculating A and E. This process repeats until the likelihood of θ converges on some value (that is, no change or a very small change is seen in $l(x^1, \ldots, x^n|\theta)$ from one step to the next).

In one version of the algorithm, the probable paths used are the Viterbi paths for each sequence and the values A and E are calculated from these paths. This seems reasonable and can produce satisfactory results but it does not converge to the maximum likelihood estimate.

It turns out that we can avoid actually imputing a probable path by directly calculating the probability that the transition from k to l occurs at position i in x:

$$P(\pi_i = k, \pi_{i+1} = l|x, \theta) = \frac{f_k(i)a_{kl}e_l(x_{i+1})b_l(i+1)}{P(x)}.$$

Thus, to get a the expected value of A_{kl}, we simply sum over all possible values of i. A similar argument can be made for E. The expected values for A_{kl} and E_{kl} are then:

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i)a_{kl}e_l(x_{i+1}^j)b_l^j(i+1) \quad (2)$$

$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{i:x_i^j=b} f_k^j(i)b_k^j(i) \quad (3)$$

The Baum-Welch algorithm proceeds as follows:

Initialise: Set starting values for the parameters. Set log-likelihood to $-\infty$.

Iterate:

1. Set A and E to their pseudo count values. For each training sequence x^j:
   (a) Calculate $f_k(i)$ for x^j from forward algorithm
   (b) Calculate $b_k(i)$ for x^j from backward algorithm
   (c) Calculate A^j and E^j and add to A and E using Equations **??** and **??** above.
2. Set new values for a and e, based on A and E
3. Calculate log-likelihood of model
4. If change in log likelihood is small, stop, else, continue.