

Pairwise alignment using HMMs

Begin

Now that we have acquired new technical machinery from hidden Markov model theory, we return for a brief chapter to pairwise sequence alignment. In Chapter 2 we introduced finite state automata with multiple states as a convenient description of more complex dynamic programming algorithms for pairwise alignment. It is also possible to consider them as a basis for a probabilistic interpretation of the gapped alignment process, by converting them into HMMs. One advantage of this approach is that we will be able to use the resulting probabilistic model to explore questions about the reliability of the alignment obtained by dynamic programming, and to explore alternative (suboptimal) alignments. Indeed, by weighting all alternatives probabilistically, we will be able to score the similarity of two sequences independent of any specific alignment. We can also build more specialised probabilistic models out of simple pieces, to model more complex versions of sequence alignment, as discussed previously for FSAs.

Let us first review briefly the finite state automaton that we introduced for pairwise alignment with affine gap penalties. We required three states, M corresponding to a match, and two states corresponding to inserts, which we name here X and Y as shown in Figure 4.1. The recurrence relations for updating the

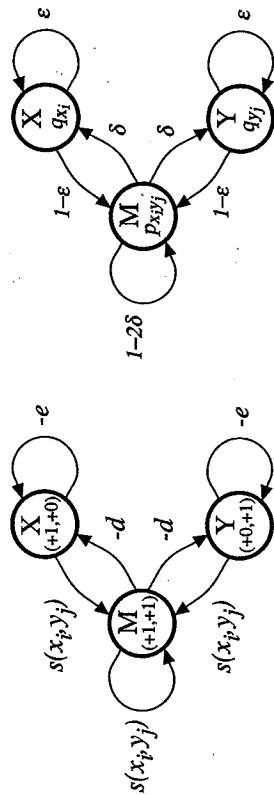


Figure 4.1 A finite state machine diagram for affine gap alignment on the left, and the corresponding probabilistic model on the right.

values of these states in the dynamic programming matrix are

$$\begin{aligned}
 V^M(i, j) &= s(x_i, y_j) + \max \begin{cases} V^M(i-1, j-1), \\ V^X(i-1, j-1), \\ V^Y(i-1, j-1); \end{cases} & (4.1) \\
 V^X(i, j) &= \max \begin{cases} V^M(i-1, j) - d, \\ V^X(i-1, j) - e; \end{cases} \\
 V^Y(i, j) &= \max \begin{cases} V^M(i, j-1) - d, \\ V^Y(i, j-1) - e. \end{cases}
 \end{aligned}$$

These equations are appropriate for global alignment. As previously, we will generally give detailed equations for global alignment, while indicating what changes need to be made for local alignment.

4.1 Pair HMMs

We need to make two sets of changes to an FSA as shown on the left side of Figure 4.1 to turn it into an HMM. First, as shown on the right of Figure 4.1, we must give probabilities both for emissions of symbols from the states, and for transitions between states. For example, state M has emission probability distribution p_{ab} for emitting an aligned pair $a:b$, and states X and Y will have distributions q_a for emitting symbol a against a gap. Because state X emits symbols x_i from sequence x , we write q_{x_i} inside the circle representing state X. We also specify transition probabilities between the states, which must satisfy the requirement that the probabilities for all the transitions leaving each state sum to one. Allowing for symmetry, there are two free parameters for the transition probabilities between the three main states. We denote the transition from M to an insert state (X or Y) by δ , and the probability of staying in an insert state by ϵ .

However, the resulting model shown on the right side of Figure 4.1 does not generate a full model that will provide a probability distribution over all possible sequences. To do that, we need to define a Begin and an End state, as shown in Figure 4.2. In effect these formalise the initialisation and termination conditions that we needed for the dynamic programming algorithms in Chapter 2. We will see below that more complex arrangements of Begin and End states can correspond to local and other types of alignments. Adding an explicit End state introduces the need for another parameter, the probability of a transition into the End state, which we assume for now to be the same from each of M, X and Y; we call it τ . This will in effect determine the average length of an alignment from the model. For now, we will set the transitions from the Begin state to be the same as from the M state (we could have just said that we will start in M, but we wanted to make clear that initialisation can be given independent consideration as well as termination).

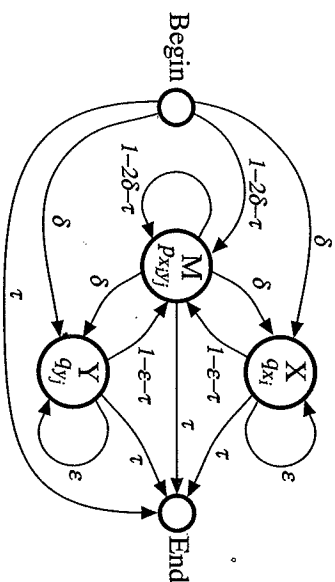


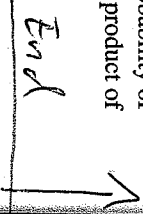
Figure 4.2 The full probabilistic version of Figure 4.1.

This gives us a probabilistic model that is very similar to a hidden Markov model as we defined it in Chapter 3. The difference is that instead of emitting a single sequence it emits a pairwise alignment. We will call this type of model a *pair HMM* to distinguish it from the more standard types of HMMs that emit single sequences. All the algorithms from Chapter 3 carry across to pair HMMs, although they need an extra dimension of search space because of the extra emitted sequence. For example, instead of writing $v_k(i)$ for the Viterbi probabilities, we write $v^k(i, j)$. We will give below the explicit sets of equations for the key algorithms, applied to the basic pair-HMM shown in Figure 4.2.

Just as a standard HMM can generate a sequence, our pair HMM can generate an aligned pair of sequences. This is done by starting in the Begin state, and cycling over the following two steps: (1) pick the next state according to the distribution of transition probabilities leaving the current state; (2) pick a symbol pair to be added to the alignment according to the emission distribution in the new state. The process stops when a transition is made into the End state. Because we have probabilities for each step, we can also keep track of the total probability of generating a particular alignment that we have made. This is just the product of the probabilities of each individual step.

The most probable path is the optimal FSA alignment

The Viterbi algorithm from Chapter 3 will allow us to find the most probable path through a pair HMM given sequences x and y . The correct form for the global pair HMM of Figure 4.2 is as follows. To make the equations simpler, we define the Begin state to be M . As in the previous chapter, we use lower-case symbols $v^*(i, j)$ for probability values, and upper-case $V^*(i, j)$ for log-odds scores. We give the Viterbi algorithm first in terms of probabilities:



Algorithm: Viterbi algorithm for pair HMMs

Initialisation:

$$v^M(0, 0) = 1. \text{ All other } v^*(i, 0), v^*(0, j) \text{ are set to } 0.$$

Recurrence: $i = 1, \dots, n, j = 1, \dots, m;$

$$v^M(i, j) = p_{xy} \max \begin{cases} (1 - 2\delta - \tau)v^M(i - 1, j - 1), \\ (1 - \epsilon - \tau)v^X(i - 1, j - 1), \\ (1 - \epsilon - \tau)v^Y(i - 1, j - 1); \end{cases}$$

$$v^X(i, j) = q_{xi} \max \begin{cases} \delta v^M(i - 1, j), \\ \epsilon v^X(i - 1, j); \end{cases}$$

$$v^Y(i, j) = q_{yj} \max \begin{cases} \delta v^M(i, j - 1), \\ \epsilon v^Y(i, j - 1). \end{cases}$$

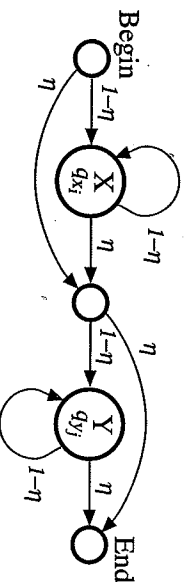
Termination:

$$v^E = \tau \max(v^M(n, m), v^X(n, m), v^Y(n, m)).$$

To find the best alignment, we keep pointers and trace back as usual. Of course, to get the alignment itself we keep track of which residues are emitted at each step in the path during the traceback, as in Chapter 2, as well as (or even in place of) the sequence of states as for the type of HMM described in Chapter 3.

Although it is clear that the recurrence equations of the pair HMM Viterbi algorithm have the same sort of form as those for the state machine version of pairwise alignment (4.1), it is instructive to see the exact form of the correspondence.

First, we have to transform into log-odds ratios with respect to the random model. In fact, now we have a full probabilistic model for our alignment, we should also have one for our random model, with a proper termination condition. Previously we have ignored the fact that our random model could not produce sequences of varying length in a proper probabilistic fashion. Here is a new random model, which is also a pair HMM.



The main states are X and Y , which emit the two sequences in turn, independently of each other. Each has a loop back onto itself with probability $(1 - \eta)$. As well as Begin and End states, there is also a silent state in between X and Y , indicated by a smaller circle. This does not emit any symbols, but is used to gather inputs from both the X and Begin states (see the section on silent states on p. 70 for further information on how these are used). When defined this way the model allows zero-length sequences x or y , just as the pair HMM model in Figure 4.2 does.