Algorithm: Barton–Sternberg multiple alignment

(i) Find the two sequences with the highest pairwise similarity and align them using standard pairwise dynamic programming alignment.

(ii) Find the sequence that is most similar to a profile of the alignment of the first two, and align it to the first two by profile–sequence alignment. Repeat until all sequences have been included in the multiple alignment.

(iii) Remove sequence x^1 and realign it to a profile of the other aligned sequences $x^2,\ldots,x^N$ by profile–sequence alignment. Repeat for sequences $x^2,\ldots,x^N$.

(iv) Repeat the previous realignment step a fixed number of times, or until the alignment score converges. ◁

The ideas of profile alignment and iterative refinement come quite close to the formulation of probabilistic hidden Markov model approaches for the multiple alignment problem. We turn to HMM methods now.

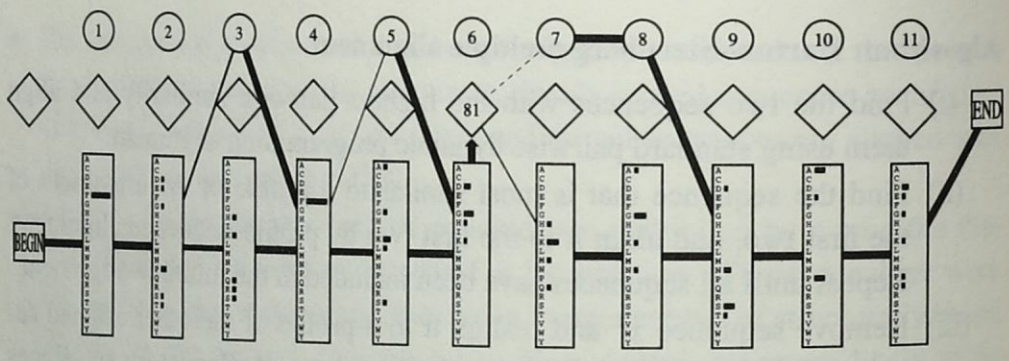
6.5 Multiple alignment by profile HMM training

In Chapter 5 it was shown that sequence profiles could be recast in probabilistic form as profile HMMs. Thus, profile HMMs could simply be used in place of standard profiles in progressive or iterative alignment methods. The use of profile HMM formalisms may have certain advantages. In particular, the essentially *ad hoc* SP scoring scheme can be replaced by the more explicit profile HMM assumption that the sequences are generated independently from a single 'root' probability distribution.

Profile HMMs can also be trained from initially unaligned sequences using the Baum–Welch expectation maximisation algorithm from Chapter 3. These sorts of approaches, drawn from the HMM literature, were in fact the first HMM-based multiple alignment approaches to be applied. If the trained model is used for a final step of Viterbi alignment of each individual sequence, training generates a multiple alignment in addition to a model [Krogh *et al.* 1994].

Multiple alignment with a known profile HMM

Before tackling the problem of estimating a model and a multiple alignment simultaneously from initially unaligned training sequences, we consider the simpler problem of obtaining a multiple alignment from a known model. This problem often arises in sequence analysis, for instance when we have a multiple alignment and a model of a small representative set of sequences in a family, and we wish to use that model to align a large number of other family members together.




```
FPHF-DLS-----HGSAQ
FESFGDLSTPDAVMGNPK
FDRFKHLKTEAEMKASED
FTQFAG-KDLESIKGTAP
FPKFKGLTTADQLKKSAD
FS-FLK-GTSEVPQNNPE
FG-FSG----AS--DPG
```

Figure 6.4 *A model (top) estimated from an alignment (bottom). The residues in the shaded area of the alignment were treated as inserts. See Figure 5.4 for a description of the model drawing.*

We have seen how to align a sequence to a profile HMM: the most probable path through the model is found by the Viterbi algorithm. Constructing a multiple alignment just requires calculating a Viterbi alignment for each individual sequence. Residues aligned to the same profile HMM match state are aligned in columns. This implies an important difference between profile HMM multiple alignments and traditional multiple alignments which will be clearer by example.

Figure 6.4 shows a small profile HMM and the multiple alignment it was derived from. The shaded residues were arbitrarily defined to be insertions for the purposes of this example, and the other ten columns correspond to ten profile HMM match states. The same seven sequences were realigned to the model, giving the optimal Viterbi paths shown in Figure 6.5. These paths result in the multiple alignment shown in Figure 6.6, left, where lower-case residues were assigned to an insert state and upper-case residues were assigned to a match state.

The important observation here is that the original alignment (Figure 6.4) and the new alignment (Figure 6.6, left) are the same alignment. A profile HMM does not attempt to align the lower-case residues assigned to insert states. The choice of how to put the insert residues in the alignment is arbitrary; some profile HMM implementations simply left-justify insert regions, as shown in Figure 6.6. The insert state residues usually represent parts of the sequences which are atypical, unconserved, and not meaningfully alignable. As we discussed earlier, this is a biologically realistic view of multiple alignment. For instance, we expect loops of homologous protein structures often to be structurally different and unalignable.

Position	1	2	3	4	5	6	insert	7	8	9	10	11
	F	P	H	F	–	D	LS	H	G	S	A	Q
	F	E	S	F	G	D	LSTPDAV	M	G	N	P	K
	F	D	R	F	K	H	LKTEAEM	K	A	S	E	D
	F	T	Q	F	A	G	KDLESI	K	G	T	A	P
	F	P	K	F	K	G	LTTADQL	K	K	S	A	D
	F	S	–	F	L	K	GTSEVP	Q	N	N	P	E
	F	G	–	F	S	G	AS	–	–	D	P	G

Figure 6.5 *The most probable paths of the seven sequences through the model. If the path goes through a match state in position i of the model, the corresponding residue is placed in the column labelled i. If it goes through a delete state, a '–' is placed in the table instead, and when it goes through the insert state in position 6 the corresponding residue is placed in the column labelled 'insert'.*

```
FPHF-Dls.....HGSAQ          FS-FLKngvdptaai--NPK
FESFGDlstpdavMGNPK          FPHF-Dls.......HGSAQ
FDRFKHlkteaemKASED          FESFGDlstpdav..MGNPK
FTQFAGkdlesi.KGTAP          FDRFKHlkteaem..KASED
FPKFKGlttadqlKKSAD          FTQFAGkdlesi...KGTAP
FS-FLKgtsevp.QNNPE          FPKFKGlttadql..KKSAD
FG-FSGas.....--DPG          FS-FLKgtsevp...QNNPE
                            FG-FSGas.......--DPG
```

Figure 6.6 *Left: the alignment of the seven sequences is shown with lowercase letters meaning inserts. The dots are just space-filling characters to make the matches line up correctly. Right: the alignment is shown after a new sequence was added to the set. The new sequence is shown at the top, and because it has more inserts more space-filling dots were added.*

In contrast, many other multiple alignment algorithms align the whole sequences, regardless of what parts of the sequence are meaningfully alignable or not.

The alignment on the right in Figure 6.6 shows a new sequence aligned to the same model. This sequence has more inserted residues than any of the other seven sequences in the shaded area assigned to insert state 6, so the alignment of the other seven sequences must be adjusted to allow space for these two new residues. In an implementation, we typically look at all the Viterbi paths and find the maximum number of inserted residues for each insert state before building the multiple alignment, so we know up front how much room we need to leave to accommodate insertions.

Overview of profile HMM training from unaligned sequences

Now we turn to the harder problem of estimating both a model and a multiple alignment from initially unaligned sequences. The method is summarised as follows:

Algorithm: Multiple alignment using profile HMMs

Initialisation: Choose the length of the profile HMM and initialise parameters.

Training: Estimate the model using the Baum–Welch algorithm (p. 64) or the Viterbi alternative (p. 65). It is usually necessary to use a heuristic method for avoiding local optima (see below).

Multiple alignment: Align all sequences to the final model using the Viterbi algorithm (p. 55) and build a multiple alignment as described in the previous section. ◁

We now consider the problems of initialisation and training in detail.

Initial model

A profile HMM is a repeating linear structure of three states (match, delete, and insert). The only decision that must be made in choosing an initial architecture for Baum–Welch estimation is the length of the model M. Here M is the number of *match* states in the profile HMM rather than the total number of states, which is $3M+3$ for the profile HMM architecture of Chapter 5. A commonly used rule is to set M to be the average length of the training sequences (or to set it based on prior knowledge).

Since Baum–Welch estimation finds local optima, not global, it is important to choose initial models carefully. The model should be encouraged to use 'sensible' transitions; for instance, transitions into match states should be large compared to other transition probabilities. At the same time, we want to start Baum–Welch from multiple different points to see if all converge to approximately the same optimum, so we want some randomness in the choice of initial model parameters.

One reasonable approach is to sample the model's initial parameters from the model's Dirichlet prior over parameters (Chapter 11). Alternatively, we can initialise the model with frequencies derived from the prior, use this model to generate a small number of random sequences, and then use these counts as 'data' to estimate an initial model. A further possibility is to estimate the initial model by model construction from an existing guess at the multiple alignment of some or all of the sequences.

Baum–Welch expectation maximisation

The basic parameter estimation is done by a straightforward application of the Baum–Welch algorithm from Chapter 3. Below we give the algorithms in the notation of Chapter 5 for reference.

Algorithm: Forward algorithm for profile HMMs

Initialisation: $f_{\mathrm{M}_0}(0) = 1$.

Recursion:
$$\begin{aligned} f_{\mathrm{M}_k}(i) &= e_{\mathrm{M}_k}(i)[f_{\mathrm{M}_{k-1}}(i-1)a_{\mathrm{M}_{k-1}\mathrm{M}_k} + f_{\mathrm{I}_{k-1}}(i-1)a_{\mathrm{I}_{k-1}\mathrm{M}_k} \\ &\qquad + f_{\mathrm{D}_{k-1}}(i-1)a_{\mathrm{D}_{k-1}\mathrm{M}_k}]; \\ f_{\mathrm{I}_k}(i) &= e_{\mathrm{I}_k}(i)[f_{\mathrm{M}_k}(i-1)a_{\mathrm{M}_k\mathrm{I}_k} + f_{\mathrm{I}_k}(i-1)a_{\mathrm{I}_k\mathrm{I}_k} \\ &\qquad + f_{\mathrm{D}_k}(i-1)a_{\mathrm{D}_k\mathrm{I}_k}]; \\ f_{\mathrm{D}_k}(i) &= f_{\mathrm{M}_{k-1}}(i)a_{\mathrm{M}_{k-1}\mathrm{D}_k} + f_{\mathrm{I}_{k-1}}(i)a_{\mathrm{I}_{k-1}\mathrm{D}_k} + f_{\mathrm{D}_{k-1}}(i)a_{\mathrm{D}_{k-1}\mathrm{D}_k}. \end{aligned}$$

Termination:
$$\begin{aligned} f_{\mathrm{M}_{M+1}}(L+1) &= f_{\mathrm{M}_M}(L)a_{\mathrm{M}_M\mathrm{M}_{M+1}} + f_{\mathrm{I}_M}(L)a_{\mathrm{I}_M\mathrm{M}_{M+1}} \\ &\qquad + f_{\mathrm{D}_M}(L)a_{\mathrm{D}_M\mathrm{M}_{M+1}}. \end{aligned}$$

◁

Algorithm: Backward algorithm for profile HMMs

Initialisation:
$$\begin{aligned} b_{\mathrm{M}_{M+1}}(L+1) &= 1; \\ b_{\mathrm{M}_M}(L) &= a_{\mathrm{M}_M\mathrm{M}_{M+1}}; \\ b_{\mathrm{I}_M}(L) &= a_{\mathrm{I}_M\mathrm{M}_{M+1}}; \\ b_{\mathrm{D}_M}(L) &= a_{\mathrm{D}_M\mathrm{M}_{M+1}}. \end{aligned}$$

Recursion:
$$\begin{aligned} b_{\mathrm{M}_k}(i) &= b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{M}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1}) \\ &\qquad + b_{\mathrm{I}_k}(i+1)a_{\mathrm{M}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{M}_k\mathrm{D}_{k+1}}; \\ b_{\mathrm{I}_k}(i) &= b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{I}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1}) \\ &\qquad + b_{\mathrm{I}_k}(i+1)a_{\mathrm{I}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{I}_k\mathrm{D}_{k+1}}; \\ b_{\mathrm{D}_k}(i) &= b_{\mathrm{M}_{k+1}}(i+1)a_{\mathrm{D}_k\mathrm{M}_{k+1}}e_{\mathrm{M}_{k+1}}(x_{i+1}) \\ &\qquad + b_{\mathrm{I}_k}(i+1)a_{\mathrm{D}_k\mathrm{I}_k}e_{\mathrm{I}_k}(x_{i+1}) + b_{\mathrm{D}_{k+1}}(i)a_{\mathrm{D}_k\mathrm{D}_{k+1}}. \end{aligned}$$

◁

The forward and backward variables can then be combined to re-estimate emission and transition probability parameters as follows:

Algorithm: Baum–Welch re-estimation equations for profile HMMs

Expected emission counts from sequence x:

$$\begin{aligned} E_{\mathrm{M}_k}(a) &= \frac{1}{P(x)} \sum_{i|x_i=a} f_{\mathrm{M}_k}(i)b_{\mathrm{M}_k}(i); \\ E_{\mathrm{I}_k}(a) &= \frac{1}{P(x)} \sum_{i|x_i=a} f_{\mathrm{I}_k}(i)b_{\mathrm{I}_k}(i). \end{aligned}$$

Expected transition counts from sequence x:

$$A_{X_k M_{k+1}} = \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k M_{k+1}} e_{M_{k+1}}(x_{i+1}) b_{M_{k+1}}(i+1);$$

$$A_{X_k I_k} = \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k I_k} e_{I_k}(x_{i+1}) b_{I_k}(i+1);$$

$$A_{X_k D_{k+1}} = \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k D_{k+1}} b_{D_{k+1}}(i).$$

◁

As usual the Baum–Welch re-estimation procedure can be replaced by the Viterbi alternative described on p. 65 (see below also). Other types of estimation have also been used for estimation of profile HMMs, such as gradient descent [Baldi *et al.* 1994].

Avoiding local maxima

The Baum–Welch algorithm is guaranteed to find a local maximum on the probability 'surface' but there is no guarantee that this local optimum is anywhere near the global optimum nor a biologically reasonable solution. Much the same is true for any practical score optimising multiple alignment method (multidimensional dynamic programming finds global optima but is not practical). Part of the reason is that these models are usually quite long, and thus there are many opportunities to get stuck in a wrong solution. For instance, two variations of the same conserved motif may end up being modelled as two different motifs or a conserved region is squeezed in between two other regions and ends up as being modelled as an insert. One way to search the parameter space is simply to start again many times from different (random) initial models and keep the best scoring final one.

A more involved approach is to use some form of stochastic search algorithm that 'bumps' Baum–Welch off from local maxima. (The two approaches can be combined, and usually are.) The most common stochastic algorithm is *simulated annealing* [Kirkpatrick, Gelatt & Vecchi 1983]. We describe what simulated annealing does, and then discuss a profile HMM training algorithm inspired by simulated annealing.

Theoretical basis of simulated annealing

Some compounds only crystallise if they are slowly annealed from high temperature to low temperature. If the temperature is lowered too fast the structure ends up in a local free energy minimum and is disordered. In an optimisation problem we have some function to minimise, which we can call the 'energy' $E(x)$, where x represents all the variables in which it has to be minimised. (Maximising a function is identical to minimising the negative value of the function.) Inspired

Gibbs sampler are stochastic sampling variants of the Viterbi approximation of EM. At each iteration of Gibbs sampling, a sequence is removed from the alignment; an HMM is built of the remaining aligned sequences; and then a new alignment of the sequence to the rest is sampled probabilistically using the stochastic sampling algorithm at $T = 1$. This iteration is repeated until the model reaches a region of high probability. The Gibbs sampler is thus like running the above simulated annealing Viterbi algorithm at a constant $T = 1$, where alignments are sampled from a probability distribution unmodified by any effect of a temperature factor. For a general description of Gibbs sampling, see Chapter 11.

Adaptively modifying model architecture; model surgery

After (or even during) training a model, we can look at the alignment it produces and decide that: (a) some of the match states are redundant and should be absorbed in an insert state; or (b) it seems like one or more insert states absorb too much sequence, in which case they should be expanded (i.e. more match modules can be inserted before or after the insert state). This can happen both because the initial choice of model length was not as good as it could have been, and because of local optima encountered during training. It is advantageous to devise procedures to adaptively modify the model's architecture during training and just after training has been completed.

In Krogh *et al.* [1994] a method called *model surgery* was described. From the 'counts' estimated by the forward–backward procedure (or the Viterbi analogue) we can see how much a certain transition is used by the training sequences. The usage of a match state is the sum of counts for all letters in the state. If a certain match state is used by less than half the sequences (or some other predefined fraction) the corresponding module is deleted. Similarly if more than half (or some other predefined fraction) of the sequences use the transitions into a certain insert state, this is expanded to some number of new modules. The number of new modules is determined by the average length of the insertions. Though it is *ad hoc*, it works well.

Another approach is to re-estimate both a model architecture and model parameters using the maximum *a posteriori* (MAP) model construction algorithm given in Chapter 5. As this procedure requires an alignment, not expected counts, it cannot be applied during the usual Baum–Welch expectation maximisation procedure. It can be applied correctly during training by the Viterbi approximation to Baum–Welch, and in fact can completely replace the usual parameter re-estimation process, leading to a (locally) convergent optimisation algorithm that simultaneously optimises both the architecture and parameters of the HMM. It can also be applied periodically (much like model surgery is applied) during full Baum–Welch estimation by inserting an iteration of Viterbi alignment and MAP model construction. In this use, it is not necessarily guaranteed to improve