by different multiple alignment programs. We conclude by describing full probabilistic multiple alignment approaches based on the profile HMMs we introduced in Chapter 5 and comparing the strengths and weaknesses of profile HMM alignment to other methods. We will focus primarily on protein alignments, though most of the discussion applies to DNA alignments as well. (Alignment of RNA is complicated by long-range correlations due to base pairing and is not treated until Chapter 10.)

6.1 What a multiple alignment means

In a multiple sequence alignment, homologous residues among a set of sequences are aligned together in columns. 'Homologous' is meant in both the structural and evolutionary sense. Ideally, a column of aligned residues occupy similar three-dimensional structural positions and all diverge from a common ancestral residue. For example, in Figure 6.1, a manually generated multiple alignment of ten immunoglobulin superfamily sequences is shown. A crystal structure of one of the sequences (1tlk, telokin) is known. The telokin structure and alignments to other related sequences reveal conserved characteristics of the I-set immunoglobulin superfamily fold, including eight conserved β-strands and certain key residues in the sequences, such as two completely conserved cysteines in the b and f strands which form a disulfide bond in the core of the folded structure. The other nine sequences, from various neural cell adhesion molecules, have been manually aligned to 1tlk based on this expert structural knowledge.

Except for trivial cases of highly identical sequences, it is not possible to unambiguously identify structurally or evolutionarily homologous positions and create a single 'correct' multiple alignment. Since protein structures also evolve (though more slowly than protein sequences), we do not expect two protein structures with different sequences to be entirely superposable. Chothia & Lesk [1986] examined pairwise structural alignments in several different protein families and found that for a given pair of divergent but clearly homologous (30% identical) protein sequences, usually only about 50% of the individual residues were superposable in the two structures (Figure 6.2). The globin family, often used as a 'typical' protein family in computational work, is in fact exceptional: almost the entire structure is conserved among divergent sequences. Even the definition of 'structurally superposable' is subjective and can be expected to vary among experts.

In principle, there is always an unambiguously correct evolutionary alignment even if the structures diverge. In practice, however, an evolutionarily correct alignment can be even more difficult to infer than a structural alignment. While structural alignment has an independent point of reference (superposition of crystal or NMR structures), the evolutionary history of the residues of a sequence

```
structure:  ...aaaaa...bbbbbbbbbb.....ccccccCCC..C........ddd
1tlk        ILDMDVVEGSAARFDCKVEGY--PDPEVMWFKDDNP--VKESR----HFQ
AXO1_RAT    RDPVKTHEGWGVMLPCNPPAHY-PGLSYRWLLNEFPNFIPTDGR---HFV
AXO1_RAT    ISDTEADIGSNLRWGCAAAGK--PRPMVRWLRNGEP--LASQN----RVE
AXO1_RAT    RRLIPAARGGEISILCQPRAA--PKATILWSKGTEI--LGNST----RVT
AXO1_RAT    ----DINVGDNLTLQCHASHDPTMDLTFTWTLDDFPIDFDKPGGHYRRAS
NCA2_HUMAN  PTPQEFREGEDAVIVCDVVSS--LPPTIIWKHKGRD--VILKKDV--RFI
NCA2_HUMAN  PSQGEISVGESKFFLCQVAGDA-KDKDISWFSPNGEK-LTPNQQ---RIS
NCA2_HUMAN  IVNATANLGQSVTLVCDAEGF--PEPTMSWTKDGEQ--IEQEEDDE-KYI
NRG_DROME   RRQSLALRGKRMELFCIYGGT--PLPQTVWSKDGQR--IQWSD----RIT
NRG_DROME   PQNYEVAAGQSATFRCNEAHDDTLEIEIDWWKDGQS--IDFEAQP--RFV
consensus:  ........G..+.+.C.+.........+.W........+........++

structure:  ddd.....eeeeee.......fffffffff.......ggggggggggg.
1tlk        IDYDEEGNCSLTISEVCGDDDAKYTCKAVNSL-----GEATCTAELLVET
AXO1_RAT    SQTT----GNLYIARTNASDLGNYSCLATSHMDFSTKSVFSKFAQLNLAA
AXO1_RAT    VLA-----GDLRFSKLSLEDSGMYQCVAENKH-----GTIYASAELAVQA
AXO1_RAT    VTSD----GTLIIRNISRSDEGKYTCFAENFM-----GKANSTGILSVRD
AXO1_RAT    AKETI---GDLTILNAHVRHGGKYTCMAQTVV-----DGTSKEATVLVRG
NCA2_HUMAN  VLSN----NYLQIRGIKKTDEGTYRCEGRILARG---EINFKDIQVIVNV
NCA2_HUMAN  VVWNDDSSSTLTIYNANIDDAGIYKCVVTGEDG----SESEATVNVKIFQ
NCA2_HUMAN  FSDDSS---QLTIKKVDKNDEAEYICIAENKA-----GEQDATIHLKVFA
NRG_DROME   QGHYG---KSLVIRQTNFDDAGTYTCDVSNGVG----NAQSFSIILNVNS
NRG_DROME   KTND----NSLTIAKTMELDSGEYTCVARTRL-----DEATARANLIVQD
consensus:  ..........L.+..+...+.+.Y.C.................+.+.+..
```

Figure 6.1 *A multiple alignment of ten I-set immunoglobulin superfamily domains, adapted from Harpaz & Chothia [1994]. To the left are sequence identifiers from the* PDB *or* SWISS-PROT *databases. The eight β-strands of the telokin structure, 1tlk, are annotated at the top (a-g; C represents the c' strand). Aligned columns are annotated at the bottom if all residues are identical (letter) or highly conservative (+).*

family is not independently known from any source; it must itself be inferred from sequence alignment. Since sequence tends to diverge more rapidly than structure, parts of proteins which are structurally unalignable are typically not alignable by sequence either.

Thus, our ability to define a single 'correct' alignment will vary with the relatedness of the sequences being aligned. An alignment of very similar sequences will generally be unambiguous, but these alignments are not of great interest to us; a simple program can get the alignment right. For cases of interest (e.g. for a family of proteins sharing perhaps only 30% average pairwise sequence identity), we must keep in mind that there is no objective way to define an unambiguously correct alignment. Usually, a small subset of key residues will be identifiable which can be aligned unambiguously for all the sequences in a family almost regardless of sequence divergence [Harpaz & Chothia 1994]; core structural elements will also tend to be conserved and meaningfully alignable; but other regions may not be meaningfully alignable because of structural evolution and sequence divergence.

Assessments of multiple alignment quality must keep these considerations in mind. Asking a sequence alignment program to produce *exactly* the same align-

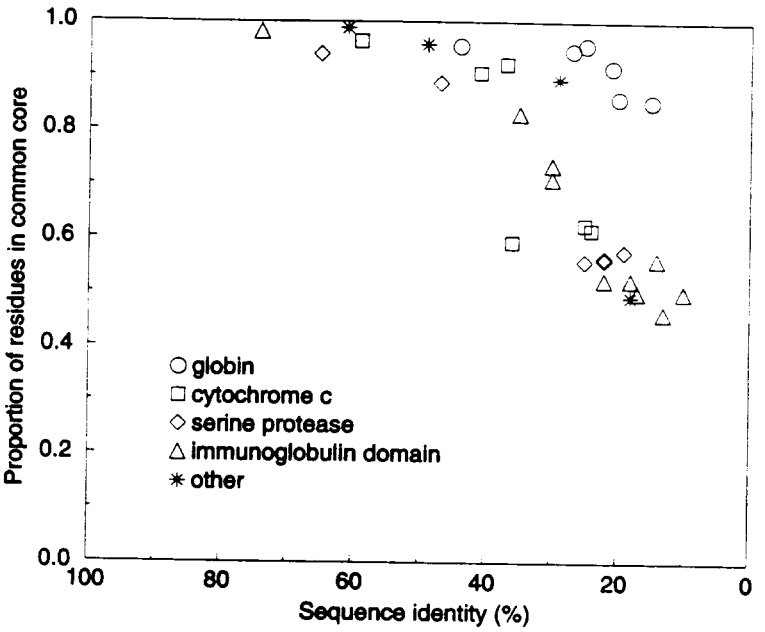

Figure 6.2 *Proportion of structurally superposable residues in pairwise alignments as a function of sequence identity; redrawn from data in Chothia & Lesk [1986]. 'Other' structural alignments include pairwise alignments of two dihydrofolate reductases, two lysozymes, plastocyanin/azurin, and papain/actinidin.*

ment as a manual structural alignment, for instance, means building in the same meaningless biases about how to 'align' structurally unalignable regions. Instead, we should focus attention on the subset of columns corresponding to key residues and core structural elements that can be aligned with confidence [McClure, Vasi & Fitch 1994].

6.2 Scoring a multiple alignment

Our scoring system should take into account at least two important features of multiple alignments: (1) the fact that some positions are more conserved than others, e.g. position-specific scoring; and (2) the fact that the sequences are not independent, but instead are related by a phylogenetic tree. An idealised way to score a multiple alignment would therefore be to specify a complete probabilistic model of molecular sequence evolution. Given the correct phylogenetic tree for the sequences, the probability of a multiple alignment is the product of the probabilities of all the evolutionary events necessary to produce that alignment via ancestral intermediate sequences times the prior probability of the root ancestral

sequence. The desired evolutionary model would be very complex. The probabilities of evolutionary change would depend on the evolutionary times along each branch of the tree, as well as position-specific structural and functional constraints imposed by natural selection, so that key residues and structural elements would be conserved. High-probability alignments would then be good structural and evolutionary alignments under this model.

Unfortunately, we do not have enough data to parameterise such a complex evolutionary model. Simplifying assumptions must be made. In this chapter, we concentrate mostly on workable approximations that partly or entirely ignore the phylogenetic tree while doing some sort of position-specific scoring of aligning structurally compatible residues. In Chapters 7 and 8 we will look at more explicit models of phylogenetic trees and molecular evolution, most of which make an approximation of a position-independent rather than position-specific evolutionary model.

Almost all alignment methods assume that the individual columns of an alignment are statistically independent. Such a scoring function can be written as

$$S(m) = G + \sum_i S(m_i) \tag{6.1}$$

where m_i is column i of the multiple alignment m, $S(m_i)$ is the score for column i, and G is a function for scoring the gaps that occur in the alignment.

We write G as an unspecified function because methods of scoring gaps in multiple alignments differ greatly. The simplest method is to treat a gap symbol as an extra residue type, which then just gives $S(m) = \sum_i S(m_i)$. However, most multiple alignment methods use affine scoring functions that pay a higher cost for opening the gap than extending it, so successive gap residues are not treated independently. For simplicity, we will focus in the next several paragraphs on definitions of $S(m_i)$ for scoring a column of aligned residues with no gaps.

Minimum entropy

We now define some notation. As above, m is a multiple alignment. Let m_i^j be the symbol in column i for sequence j. Let c_{ia} be the observed counts for residue a in column i; $c_{ia} = \sum_j \delta(m_i^j = a)$, where $\delta(m_i^j = a)$ is 1 if $m_i^j = a$ and 0 otherwise. Let m_i be the column $m_i^1, \ldots, m_i^N$ of aligned symbols in column i, and let c_i be the count vector $c_{i1}, \ldots, c_{iK}$ of observed symbols in column i for an alphabet of K different residues.

If the phylogenetic tree for the sequences has many intermediate ancestors, then the statistical dependence between sequences is complex (see Chapter 7). The scoring problem is greatly simplified if we assume that sequences have all been generated independently. If we assume that residues *within* the column are independent, as well as being independent *between* columns, then the probability

of a column m_i is

$$P(m_i) = \prod_a p_{ia}^{c_{ia}}, \tag{6.2}$$

where p_{ia} is the probability of residue a in column i. We can define a column score as the negative logarithm of this probability:

$$S(m_i) = -\sum_a c_{ia} \log p_{ia}. \tag{6.3}$$

This is an *entropy* measure directly related to the equation for Shannon entropy in information theory (Chapter 11). It is a convenient measure of the variability observed in an aligned column of residues. The more variable the column is, the higher the entropy. A completely conserved column would score 0. We could define a good alignment to be one which minimises the total entropy of the alignment (e.g. $\sum_i S(m_i)$).

As we have seen before (Chapter 5), the parameters p_{ia} can be estimated from counts c_{ia}; for instance, the maximum likelihood estimate is just

$$p_{ia} = \frac{c_{ia}}{\sum_{a'} c_{ia'}}. \tag{6.4}$$

In practice we would normally regularise this probability estimate with pseudocounts or Dirichlet priors.

This is obviously near to the HMM formulation of the problem. Profile HMMs go further and also model insertions and deletions in the alignment probabilistically. In return for giving up the evolutionary tree and assuming independence between sequences, we gain the ability to straightforwardly estimate a position-specific model of both residue probabilities in columns and insertions and deletions. Standard profiles make a similar assumption.

The assumption that the sequences are independent can be reasonable if representative sequences of a sequence family are carefully chosen. It is often the case, though, that the sample of sequences is biased and certain evolutionary subfamilies are under- or over-represented relative to others. A variety of tree-based weighting schemes have been proposed to deal with this problem to partially compensate for the defects of the sequence independence assumption (see Chapter 5).

Sum of pairs: SP scores

The standard method of scoring multiple alignments is not the HMM formulation, but is similar in that it does not use a phylogenetic tree and it assumes statistical independence for the columns. Columns are scored by a 'sum of pairs' (SP) function using a substitution scoring matrix. The SP score for a column is defined

as:

$$S(m_i) = \sum_{k<l} s(m_i^k, m_i^l), \tag{6.5}$$

where scores $s(a,b)$ come from a substitution scoring matrix such as a PAM or BLOSUM matrix. For simple linear gap costs, gaps are handled by defining $s(a,-)$ and $s(-,a)$ to be the gap cost, and $s(-,-)$ to be zero. Otherwise gaps are scored separately (e.g. for affine gap costs).

Summing all the pairwise substitution scores in the column might seem to be a natural thing to do. However, substitution scores are usually derived as log-odds scores for pairwise comparisons. The correct extension to multiple alignments would be, for instance, $\log(p_{abc}/q_a q_b q_c)$ for a three-way alignment, rather than the SP score $\log(p_{ab}/q_a q_b) + \log(p_{bc}/q_b q_c) + \log(p_{ac}/q_a q_c)$. There is no probabilistic justification of the SP score; each sequence is scored as if it descended from the $N-1$ other sequences instead of a single ancestor. Evolutionary events are over-counted, a problem which increases as the number of sequences increases. Altschul, Carroll & Lipman [1989] recognised the problem and proposed a weighting scheme designed to partially compensate for this defect in SP scores (Chapter 5).

Example: A problem with SP scores

As an intuitive, concrete example of a problem with the standard SP multiple alignment scoring system, consider an alignment of N sequences which all have leucine (L) at a certain position for some important functional reason. The score of an L aligned to L according to the BLOSUM50 substitution matrix (Figure 2.2) is 5, so the SP score of the column is $5 \times N(N-1)/2$, where $N(N-1)/2$ is the number of symbol pairs in the column. If instead there were one glycine (G) in the column and $N-1$ Ls, the score for the column would be $9 \times (N-1)$ less, because a G-L pair scores -4 instead of $+5$, and $N-1$ pairs are affected. That is, the SP score for a column with one G is worse than the score for a column of all Ls by a fraction of

$$\frac{9(N-1)}{5N(N-1)/2} = \frac{18}{5N}.$$

Notice the inverse dependence on N; the relative difference in score between the correct alignment and the incorrect alignment *decreases* with the number of sequences in the alignment. This is clearly counter-intuitive. The relative difference ought to increase with the more evidence we have for a conserved leucine. See p. 105 for another example.

□

6.3 Multidimensional dynamic programming

With some appreciation of scoring issues in mind, we turn to algorithms for constructing multiple alignments.

It is possible to generalise pairwise dynamic programming alignment (Chapter 2) to the alignment of N sequences. However, this turns out to be impractical for more than a few sequences, as we will see shortly. We assume the columns of an alignment are statistically independent, and for now we also assume that gaps are scored with a linear gap cost $\gamma(g) = gd$ for a gap of length g and some gap cost d. Thus we can calculate the overall score $S(m)$ for an alignment as a sum of the scores $S(m_i)$ for each column i:

$$S(m) = \sum_i S(m_i). \tag{6.6}$$

Multidimensional dynamic programming with affine gap costs and multiple states is possible, using methods like those in Chapter 2, but the formalism becomes tedious in many dimensions.

Define $\alpha_{i_1,i_2,\ldots,i_N}$ as the maximum score of an alignment up to the subsequences ending with $x^1_{i_1}, x^2_{i_2}, \ldots, x^N_{i_N}$. The dynamic programming algorithm is

$$\alpha_{i_1,i_2,\ldots,i_N} = \max \begin{cases} \alpha_{i_1-1,i_2-1,\ldots,i_N-1} & + \; S(x^1_{i_1}, x^2_{i_2}, \ldots, x^N_{i_N}), \\ \alpha_{i_1,i_2-1,\ldots,i_N-1} & + \; S(-, x^2_{i_2}, \ldots, x^N_{i_N}), \\ \alpha_{i_1-1,i_2,i_3-1,\ldots,i_N-1} & + \; S(x^1_{i_1}, -, \ldots, x^N_{i_N}), \\ \qquad\vdots & \\ \alpha_{i_1-1,i_2-1,\ldots,i_N} & + \; S(x^1_{i_1}, x^2_{i_2}, \ldots, -), \\ \alpha_{i_1,i_2,i_3-1,\ldots,i_N-1} & + \; S(-, -, \ldots, x^N_{i_N}), \\ \qquad\vdots & \\ \alpha_{i_1,i_2-1,\ldots,i_{N-1}-1,i_N} & + \; S(-, x^2_{i_2}, \ldots, -), \\ \qquad\vdots & \end{cases} \tag{6.7}$$

where all combinations of gaps appear except the one where all residues are replaced by gaps. There are $2^N - 1$ such combinations. Initialisation, termination, and traceback steps for the algorithm are not shown here, but also follow analogously from the pairwise dynamic programming algorithm.

It is possible to simplify the notation by introducing Δ_i which is 0 or 1 and define the 'product'

$$\Delta_i \cdot x = \begin{cases} x & \text{if } \Delta_i = 1, \\ - & \text{if } \Delta_i = 0. \end{cases} \tag{6.8}$$

Now the recursion can be written as follows [Sankoff & Cedergren 1983; Water-

man 1995]:

$$\alpha_{i_1,i_2,\ldots,i_N} = \max_{\Delta_1+\ldots+\Delta_N>0} \left\{\alpha_{i_1-\Delta_1,i_2-\Delta_2,\ldots,i_N-\Delta_N} + S(\Delta_1 \cdot x^1_{i_1}, \Delta_2 \cdot x^2_{i_2}, \ldots, \Delta_N \cdot x^N_{i_N})\right\}. \tag{6.9}$$

The algorithm requires the computation of the whole dynamic programming matrix with $L_1 L_2 \cdots L_N$ entries. To calculate each entry we need to maximise over all $2^N - 1$ combinations of gaps in a column, excluding the case where all the Δ_k are zero. Assuming the sequences are of roughly the same length $\bar{L}$, the memory complexity of the multidimensional dynamic programming algorithm is $O(\bar{L}^N)$ and the time complexity is $O(2^N \bar{L}^N)$.

Note that we did not specify the functional form of the column score $S(m_i)$. The only assumption necessary to make multidimensional dynamic programming work is that column scores are independent. In principle, $S(m_i)$ could be calculated using an evolutionary model [Sankoff 1975].

Exercise

6.1 Assume we have a number of sequences that are 50 residues long, and that a pairwise comparison of two such sequences takes one second of CPU time on our computer. An alignment of four sequences takes $(2L)^{N-2} = 10^{2N-4} = 10^4$ seconds (a few hours). If we had unlimited memory and we were willing to wait for the answer until just before the sun burns out in five billion years, how many sequences could our computer align?

MSA

A clever algorithm for reducing the volume of the multidimensional dynamic programming matrix that needs to be examined was described by Carrillo & Lipman [1988]. This algorithm was implemented in the multiple alignment program MSA [Lipman, Altschul & Kececioglu 1989]. MSA can optimally align up to five to seven protein sequences of reasonable length (200–300 residues).

Carrillo & Lipman assume an SP scoring system for both residues and gaps. We assume here that the score of a multiple alignment is the sum of the scores of all pairwise alignments defined by the multiple alignment; a somewhat broader definition of the score is possible [Altschul 1989]. Let a^{kl} denote the pairwise alignment between sequences k and l. Then the score of the complete alignment is given by

$$S(a) = \sum_{k<l} S(a^{kl}). \tag{6.10}$$

Let $\hat{a}^{kl}$ be the optimal *pairwise* alignment of k, l, which we can calculate in $O(\bar{L}^2)$ time by standard dynamic programming. Obviously $S(a^{kl}) \leq S(\hat{a}^{kl})$.

Combining this simple observation and the definition of the SP scoring system,

we can obtain a lower bound on the score of any pairwise alignment that can occur in the optimal multiple alignment. Assume for the moment that we have a lower bound $\sigma(a)$ on the score of the optimal multiple alignment, so $\sigma(a) \leq S(a)$. From the above and the SP score definition it must be true that, for the optimal multiple alignment a,

$$\sigma(a) \leq S(a^{kl}) - S(\hat{a}^{kl}) + \sum_{k'<l'} S(\hat{a}^{k'l'})$$

and thus

$$S(a^{kl}) \geq \beta^{kl}$$
$$\text{where } \beta^{kl} = \sigma(a) + S(\hat{a}^{kl}) - \sum_{k'<l'} S(\hat{a}^{k'l'}).$$

Therefore, we know we only need to consider pairwise alignments of k and l that score better than β^{kl}. This lower bound β^{kl} is easily calculated. We can obtain a good bound $\sigma(a)$ by any fast heuristic multiple alignment algorithm (for example one of the progressive alignment algorithms given below). The $N(N-1)/2$ optimum pairwise alignments $\hat{a}^{kl}$ are each calculated and scored by standard pairwise alignment. The higher these bounds are, the smaller the volume of dynamic programming matrix that must be calculated and the faster the algorithm will run. (Indeed, by default MSA heuristically picks a higher β^{kl} and so does not guarantee an optimal multiple alignment.)

Now, for each pair k,l we can find the complete set B^{kl} of coordinate pairs (i_k, i_l) such that the best alignment of x^k to x^l through (i_k, i_l) scores more than β^{kl}. This set is calculated in $O(\bar{L}^2)$ time by summing the forward and backward Viterbi scores for each cell of the complete pairwise dynamic programming table, and testing if the result is greater than β^{kl}. The costly multidimensional dynamic programming algorithm can then be restricted to evaluate only cells in the intersection of all these sets: i.e. cells $(i_1, i_2, \ldots, i_N)$ for which (i_k, i_l) is in B^{kl} for all k,l (see Figure 6.3). It is tricky to manage the intersection matrix and perform the dynamic programming calculation efficiently. Details are given in Gupta, Kececioglu & Schaffer [1995].

Altschul & Lipman [1989] extended the theory of the Carrillo–Lipman algorithm to more realistic scoring systems based on evolutionary stars and trees instead of SP scores, but we are not aware of any implementations of those ideas.

6.4 Progressive alignment methods

Probably the most commonly used approach to multiple sequence alignment is *progressive alignment*. This works by constructing a succession of pairwise alignments. Initially, two sequences are chosen and aligned by standard pairwise alignment; this alignment is fixed. Then, a third sequence is chosen and aligned

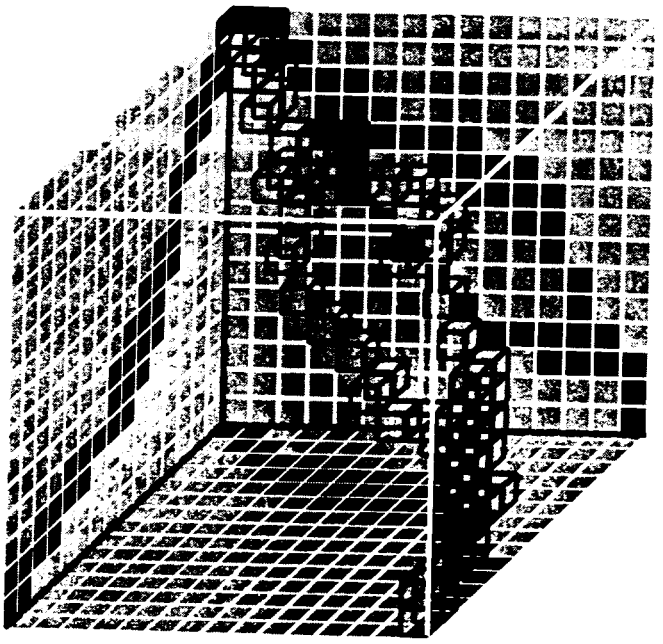

Figure 6.3 *Carrillo & Lipman's algorithm allows the search for optimal alignments to be restricted to a subset of the multidimensional programming matrix, shown here as three-dimensional. The sets B^{kl} are shown in dark grey, and the cells in the matrix to which the search can be confined are outlined in black.*

to the first alignment, and this process is iterated until all sequences have been aligned.

Progressive alignment strategies were introduced by a number of authors [Hogeweg & Hesper 1984; Waterman & Perlwitz 1984; Feng & Doolittle 1987; Taylor 1987; Barton & Sternberg 1987; Higgins & Sharp 1989]. The algorithms differ in several ways: (1) in the way that they choose the order to do the alignment; (2) in whether the progression involves only alignment of sequences to a single growing alignment or whether subfamilies are built up on a tree structure and, at certain points, alignments are aligned to alignments; and (3) in the procedure used to align and score sequences or alignments against existing alignments.

Progressive alignment is heuristic: it does not separate the process of scoring an alignment from the optimisation algorithm. It does not directly optimise any global scoring function of alignment correctness. The advantage of progressive alignment is that it is fast and efficient, and in many cases the resulting alignments are reasonable.

The most important heuristic of progressive alignment algorithms is to align the most similar pairs of sequences first. These are the most reliable alignments. Most algorithms build a 'guide tree'. This is a binary tree whose leaves represent sequences and whose interior nodes represent alignments. The root node represents a complete multiple alignment. The nodes furthest from the root represent

the most similar pairs. The methods used to construct guide trees are similar to the methods used to construct phylogenetic trees (Chapter 7), but guide trees are typically 'quick and dirty' trees unsuitable for serious phylogenetic inference.

Feng–Doolittle progressive multiple alignment

The Feng–Doolittle algorithm was one of the first progressive alignment algorithms [Feng & Doolittle 1987]. In overview, it is as follows:

Algorithm: Feng–Doolittle progressive alignment

(i) Calculate a diagonal matrix of $N(N-1)/2$ distances between all pairs of N sequences by standard pairwise alignment, converting raw alignment scores to approximate pairwise 'distances'.

(ii) Construct a guide tree from the distance matrix using the clustering algorithm by Fitch & Margoliash [1967a].

(iii) Starting from the first node added to the tree, align the child nodes (which may be two sequences, a sequence and an alignment, or two alignments). Repeat for all other nodes in the order that they were added to the tree (i.e. from most similar pairs to least similar pairs) until all sequences have been aligned. ◁

The method for converting alignment scores to distances does not need to be especially accurate, as the goal is only to create an approximate guide tree, not an evolutionary tree. Feng & Doolittle calculate the distance D as

$$D = -\log S_{\text{eff}} = -\log \frac{S_{\text{obs}} - S_{\text{rand}}}{S_{\text{max}} - S_{\text{rand}}}, \tag{6.11}$$

where S_{obs} is the observed pairwise alignment score; S_{max} is the maximum score, the average of the score of aligning either sequence to itself; and S_{rand} is the expected score for aligning two random sequences of the same length and residue composition. The last one, S_{rand}, may either be calculated by random shuffling of the two sequences, or by an approximate calculation given in Feng & Doolittle [1996]. The effective score S_{eff} can thus be viewed as a normalised percentage similarity; it is expected to roughly decay exponentially towards zero with increasing evolutionary distance, hence the $-\log$ to make the measure more approximately linear with evolutionary distance. In phylogenetic tree construction, more care must be taken in calculating distances from alignments.

The Fitch–Margoliash algorithm is one of the fast clustering algorithms that build evolutionary trees from distance matrices. Clustering algorithms are described in Chapter 7.

Sequence–sequence alignments are done with the usual pairwise dynamic programming algorithm. A sequence is added to an existing group by aligning it

pairwise to each sequence in the group in turn. The highest scoring pairwise alignment determines how the sequence will be aligned to the group. For aligning a group to a group, all sequence pairs between the two groups are tried; the best pairwise sequence alignment determines the alignment of the two groups. Thus, the scoring system is essentially the standard pairwise PAM score with an affine gap penalty. After an alignment is completed, gap symbols are replaced with a neutral X character. Feng & Doolittle call this the rule of 'once a gap, always a gap'. The rule allows pairwise sequence alignments to be used to guide the alignment of sequences to groups or groups to groups; otherwise, any given pairwise sequence alignment would not necessarily be consistent with the pre-existing alignment of a group. Since there is no cost for aligning an X with anything (including a gap symbol), the rule has a desirable side effect of encouraging gaps to occur in the same columns in subsequent pairwise alignments. The X rewriting is not needed in profile-based progressive alignment algorithms (see below).

Profile alignment

A problem with the Feng–Doolittle approach is that all alignments are determined by pairwise sequence alignments. Once an aligned group has been built up, it is advantageous to use position-specific information from the group's multiple alignment to align a new sequence to it. The degree of sequence conservation at each position should be taken into account and mismatches at highly conserved positions penalised more stringently than mismatches at variable positions. Gap penalties in positions might be reduced where lots of gaps occur in the cluster alignment, and increased where no gaps occur. This is the same argument that motivated the development of sequence profiles for database searching (Chapter 5). It also makes sense to apply profiles in progressive multiple sequence alignment.

Many progressive alignment methods use pairwise alignment of sequences to profiles [Thompson, Higgins & Gibson 1994a; Gribskov, McLachlan & Eisenberg 1987] or of profiles to profiles (see e.g. Gotoh [1993]) as a subroutine which is used many times in the process. The exact definition of the scoring function used in profile–sequence or profile–profile alignment varies. Aligned residues are usually scored by some form of a sum-of-pairs score, but the handling of gaps varies substantially between different methods.

As discussed previously, for linear gap scoring, profile alignment is simple, because the gap scores can be included in the SP score (6.5) by setting $s(-,a) = s(a,-) = -g$ and $s(-,-) = 0$. Assume we have two multiple alignments (or 'profiles'), one containing sequence 1 to n, and the other containing sequence $n+1$ to N. An alignment of these two profiles means that gaps are inserted in whole columns, so the alignment within one of the profiles is not changed. The

score (6.5) of the global alignment is then

$$\sum_i S(m_i) = \sum_i \sum_{k<l\le N} s(m_i^k, m_i^l)$$
$$= \sum_i \sum_{k<l\le n} s(m_i^k, m_i^l) + \sum_i \sum_{n<k<l\le N} s(m_i^k, m_i^l) + \sum_i \sum_{k\le n, n<l\le N} s(m_i^k, m_i^l).$$

All we did was to split up the sum into two sums only concerning the two profiles and one sum containing all the cross terms. The first two sums are unaffected by the global alignment, because adding columns of gap characters to a profile adds zero to the score ($s(-,-)=0$). Therefore the *optimal alignment* of the two profiles can be obtained by only optimising the last sum with the cross terms. This can be done exactly like a standard pairwise alignment, where columns are scored against columns by adding the pair scores. Obviously one of the profiles can consist of a single sequence only, which corresponds to aligning a single sequence to a profile.

CLUSTALW

One widely used implementation of profile-based progressive multiple alignment is the CLUSTALW program [Thompson, Higgins & Gibson 1994a], which succeeded an earlier popular program, CLUSTALV [Higgins, Bleasby & Fuchs 1992]. CLUSTALW works in much the same way as the Feng–Doolittle method except for its carefully tuned use of profile alignment methods. In overview, the CLUSTALW algorithm is as follows:

Algorithm: CLUSTALW **progressive alignment**

(i) Construct a distance matrix of all $N(N-1)/2$ pairs by pairwise dynamic programming alignment followed by approximate conversion of similarity scores to evolutionary distances using the model of Kimura [1983].

(ii) Construct a guide tree by a neighbour-joining clustering algorithm by Saitou & Nei [1987].

(iii) Progressively align at nodes in order of decreasing similarity, using sequence–sequence, sequence–profile, and profile–profile alignment. ◁

CLUSTALW is unabashedly *ad hoc* in its alignment construction and scoring stage. In addition to the usual methods of profile construction and alignment, various additional heuristics of CLUSTALW contribute to its accuracy:

- Sequences are weighted to compensate for biased representation in large subfamilies. The profile scoring function in CLUSTALW is fundamentally sum-of-pairs. As with Carrillo–Lipman, sequence weighting is important to compensate for the defects of the sum-of-pairs.

- The substitution matrix used to score an alignment is chosen on the basis of the similarity expected of the alignment; closely related sequences are aligned with 'hard' matrices (e.g. BLOSUM80), and distant sequences are aligned with 'soft' matrices (e.g. BLOSUM50).
- Position-specific gap-open profile penalties are multiplied by a modifier that is a function of the residues observed at the position. These penalties were obtained from gap frequencies observed in a large number of structurally based alignments. In general, hydrophobic residues (which are more likely to be buried) give higher gap penalties than hydrophilic or flexible residues (which are more likely to be surface-accessible).
- Gap-open penalties are also decreased if the position is spanned by a consecutive stretch of five or more hydrophilic residues.
- Both gap-open and gap-extend penalties are increased if there are no gaps in a column but gaps occur nearby in the alignment. This rule tries to force all the gaps to occur in the same places in an alignment.
- In the progressive alignment stage, if the score of an alignment is low, the guide tree may be adjusted on the fly to defer the low-scoring alignment until later in the progressive alignment phase when more profile information has been accumulated.

From the standpoint of probabilistic modelling, it is of interest to study such carefully crafted heuristics. It might be good to co-opt the heuristics into more formal probabilistic models, bringing to bear the ability of full probabilistic models to optimise large sets of free parameters.

Iterative refinement methods

One problem with progressive alignment algorithms is that the subalignments are 'frozen'. That is, once a group of sequences has been aligned, their alignment to each other cannot be changed at a later stage as more data arrive. Iterative refinement algorithms attempt to circumvent this problem [Barton & Sternberg 1987; Berger & Munson 1991; Gotoh 1993].

In iterative refinement, an initial alignment is generated, for instance as outlined above; then one sequence (or a set of sequences) is taken out and realigned to a profile of the remaining aligned sequences. If a meaningful score is being optimised, this either increases the overall score or results in the same score. Another sequence is chosen and realigned, and so on, until the alignment does not change. The procedure is guaranteed to converge to a local maximum of the score provided that all the sequences are tried and a maximum score exists, simply because the sequence space is finite.

The method of Barton & Sternberg [1987] is an example of how some of the methods mentioned so far can be combined. It works as follows:

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.