Algorithms for bioinformatics

Giacomo Fantoni
telegram: @GiacomoFantoni

Github: https://github.com/giacThePhantom/algorithms-for-bioinformatics

June 23, 2022

Contents

Chapter 1

Needleman Wunsch

1.1 Introduction

Direct comparison of two sequences based on the presence in both of the corresponding amino acids in an identical array is insufficient to establish the full genetic relationship between two proteins. Allowance for gaps multiplies the number of comparisons that can be made but introduces unnecessary and partial comparisons.

1.2 A general method for sequence comparison

The maximum match can be defined as the largest number of amino acids of one protein that can be matched with those of another protein while allowing for all possible deletions. It can be determined by representing in a matrix all possible pair combinations that can be constructed from the amino acid sequences of the protein being compared. So A_j is the jth amino acids of protein A and B_i is the ith amino acids of protein B. A_j are the columns and B_i all the rows of the matrix MAT. Then A_{ij} represent a pair combination with amino acids A_j and B_i. Every possible comparison can be represented by pathway through the matrix. A pathway is signified by a line connecting cells of the array. Complete diagonals contain no gaps. A necessary pathway begins at a cell in the first column of row. Either i or j must increase by only one, while the other may increase by one or more, leading to the next cell in a pathway. This is repeated until i, j or both reach their limiting value. Every partial or unnecessary pathway will be contained in at least one necessary pathway. The values in the matrix are computed as:

$$MAT_{ij} = \max(MAT_{i-1,j-1} + \alpha\delta(A_j, B_i), MAT_{i-i,j} + d, MAT_{i,j-1} + d)$$

Where d is the penalty factor, a number subtracted for every gap made, may be defined as a barrier for allowing the gap. And α can be a function that can represent any theory with the significance of a pair of amino acids. No gap would be allowed in the operation unless the benefit from allowing that gap would exceed the barrier. This method can be expanded for allowing the comparison of n sequences through and n-dimensional matrix. The maximum-match pathway can be obtained by beginning at the terminals of the sequences and proceeding towards the origin, first by adding to the value of each cell possessing indices $i = y - 1$ and or $j = z - 1$. The process is iterated until all cells in the matrix have been operated upon. Each cell in the outer row or column will contain the maximum number of matches that can be obtained by originating any pathway at that cell and the largest number in that row or column is equal to the maximum match. The cells of the array which contributed to the maximum match may be determined by recording the origin of the number that was added to each cell when the array was operated upon.

1.3 Evaluating the significance of the maximum match

To accomplish the estimate of if a result found differs significantly from a match between random sequences two sets of random sequences can be constructed, each one from the set of amino acid composition of each of the proteins. If the value found for the real proteins is significantly different the difference a function of of the sequences alone and not of the composition.

1.4 Cell values and weighting factors

Cells can be weighted in accordance with the maximum number of corresponding bases in codons of the represented amino acids, to make the comparison more accurate. Also the significance of the maximum match is enhanced by decreasing the weight of those pathways containing a large number of gaps through the penalty factor.

Chapter 2

Smith Watermann

2.1 Introduction

The Smith Watermann algorithm extends the one of Needleman and Wunsch to find a pair of segment, one from each of two long sequences, such that there is no other pair of segments with greater similarity. This similarity measure allows for deletion and insertion of arbitrary length.

2.2 Algorithm

Consider two molecular sequences $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_m$. Given a similarity $s(a, b)$ of elements of the sequence and W_k the weight of deletions of length k, to find pairs of segments with high degrees of similarity, a matrix H is set up such that:

$$H_{k0} = H_{0l} = 0 \qquad \forall 0 \leq k \leq n \wedge 0 \leq l \leq m$$

H_{ij} si the maximum similarity of two segments ending in a_i and b_j. H_{ij} is computed such that:

$$H_{ij} = \max(H_{i-1,j-1} + s(a_i, b_j), \max_{k \geq 1}(H_{i-k,j} - W_k), \max_{l \geq 1}(H_{i,j-l} - W_k), 0)$$

With $1 \leq i \leq n$ and $1 \leq j \leq m$. So H_{ij} is:

- $H_{i-1,j-1} + s(a_i, b_j)$ If a_i and b_j are associated.
- $H_{i-k,j} - w_k$ if a_i is at the end of a deletion of length k.
- $H_{i-k,j} - W_l$ if b_j is at the end of a deletion of length l.
- 0 is used to prevent calculated negative similarity, indicating no similarity up to a_i and b_j.

The pair of segments with maximum similarity is found first by locating the maximum elemnt of H. The other elements are determined sequentially with a traceback procedure ending with an element of H equal to 0. This procedure other than identifying the elements produces their alignment. The parameters where:

$$s(a_i, b_j) = \begin{cases} 1 & a_i = b_j \\ 0 & a_i \neq b_j \end{cases}$$

And

$$W_k = \frac{1}{3}k$$

This algorithm in particular allows for the alignment of sequences that contained both mismatches and internal deletions.

Chapter 3

PAM - a model of evolutionary change in proteins

3.1 Accepted point mutation

An accepted point mutation in a protein is a replacement of one amino acid by another accepted by natural selection. To be accepted the new amino acid usually must function in a similar way to the old one. The likelihood of amino acid X replacing Y is the same as Y replacing X is assumed the same because it depends on the product of the frequencies of occurrence and on their chemical and physical similarity. SO evolution is a vibration around given frequencies.

3.2 Mutability of amino acids

The relative mutability is the probability that each amino acid will change in a given small evolutionary intervall. To compute it the number of times that each amino acid has changed in an interval and the number of times that it has occurred in the sequences and thus has been subject to mutation. In calculating this number in for many trees, with sequences of different lengths and evolutionary distance is combined in relative mutabilities. Each relative mutability is a ration between the total number of changes on all branches of all protein trees considered and the total exposure of the amino acid to mutation, or the sum for all branches of its local frequency of occurrence multiplied by the total number of mutation per 100 links of that branch.

3.3 Mutation probability matrix for the evolutionary distance of one PAM

The individual kind of mutations and the relative mutability of the amino acids can be combined into a mutation probability matrix in which M_{ij} gives the probability that the amino acid in column j will be replaced by the amino acid in row i after a given evolutionary period. The non-diagonal elements are computed as:

$$M_{ij} = \frac{\lambda m_j A_{ij}}{\sum\limits_i A_{ij}}$$

Where:

- A_{ij} is an element of the accepted point mutation matrix.
- λ is a proportionality constant.
- m_j is the mutability of the jth amino acid.

The diagonal elements are:

$$M_{jj} = 1 - \lambda m_j$$

The sum of all elements of each column or row is 1. The probability of observing a change is proportional to the mutability of the amino acid in that place. The same proportionality constant λ holds for all columns. $100 \cdot \sum f_i M_{ij}$ gives the number of amino acids that will remain unchanged when a protein 100 links long of average composition is exposed to the evolutionary change. This depends on λ. To change the evolutionary period the matrix is multiplied my itself n times, and with $n \to \infty$ each column approaches the asymptotic amino acid composition. The percentage of amino acids that will be observed to change on the average in the interval are found by:

$$100(1 - \sum_i f_i M_{ij})$$

The term of the relatedness odds matrix are:

$$R_{ij} = \frac{M_{ij}}{f_i}$$

Or the mutation probability of a change over the probability that i will occur in the second sequence by chance. Each term of this matrix gives the probability of replacement per occurrence of i per occurrence of j. Amino acids with score > 1 replace each other more often as alternative in related sequences that in random sequences.

Chapter 4

BLOSUM

4.1 Introduction

4.1.1 Abstrac

The most used substitution matrix with scores for all possible exchanges of one amino acid is based on the Dayhoff model of evolutionary rates. This work proposes a different approach from blocks of aligned sequence segments, leading to improvement in alignments and in searches.

4.1.2 Introduction

Sequence alignment of proteins provide important insights into gene and protein function. There are different types of alignments:

- Global alignments of pairs related by common ancestry.
- Multiple alignments of members of protein families.
- Alignments made during data base searches to detect homology.

In each case a scoring scheme for estimating similarity is used. The mutation data matrices of Dayhoff are considered the default in alignment and searching programs. However the most common task is the detection of much more distant relationships, which are only inferred from substitution rates in the Dayhoff model.

4.2 Methods

4.2.1 Deriving a frequency table from a data base blocks

Local alignments can be represented as ungapped blocks with each row a different protein segment and each column an aligned residue position. Protomat can be used for obtaining a set of blocks given a group of related proteins. Considering a single block representing a conserved region of a protein family, for a new member a set of score for matches and mismatches that best favours a correct alignment with each of the other segments in the block. For each column of the block, the number of matches and mismatches of each type between the new sequence and every other are counted. This is repeated for all columns of all blocks and the summed results are stored in a table. The new sequence is then added to the group. For another new sequence the procedure is repeated. Doing so the table in the end will consist of counts of all possible amino acid pairs in a column. Counts of all possible pairs in each column of each block in the data base are summed. If a block has a width w amino acids and a depth of s sequences, contributes $ws\frac{(s-1)}{2}$ amino acids pairs to the count. This results in a frequency table listing the number of times each of the different amino acid pairs occurs among the blocks. This table is used to compute the odds ratio matrix between the observed frequencies and the expected one.

4.2.2 Computing a logarithm of odds matrix

Let the total number of amino acid pairs i, j for each entry of the frequency table be $f - ij$. Then the observed probability of occurrence for each i, j pair is:

$$q_{ij} = \frac{f_{ij}}{\sum\limits_{i=1}^{20} \sum\limits_{j=1}^{i} f_{iJ}}$$

The expected probability of occurrence for each pair is computed following the occurrence of the ith amino acid in a i, j pair:

$$p_i = q_{ii} + \sum_{j \neq i} \frac{q_{ij}}{2}$$

The expected probability of occurrence e_{ij} for each i, j pair is then $p_i p_j$ for $i = j$ and $p_i p_j + p_j p_i = 2p_i p_j$ for $i \neq j$. An odds ratio matrix is computed such that each entry is $\frac{q_{ij}}{e_{ij}}$. A lod (logarithm of odds) is then calculated in bit units as $s = \log_2\left(\frac{q_{ij}}{e_{ij}}\right)$. If the observed frequencies are as expected $s_{ij} = 0$, if less $s_{ij} < 0$< if more $s_{ij} > 0/$ Lod ratios are multiplied by a 2 scaling factor and rounded to the nearest integer value to produce the block substitution matrix BLOSUM. The relative entropy or the average mutual information per amino acid pair is computed:

$$H = \sum_{i=1}^{20} \sum -j = 1^i q_{ij} \times s_{ij}$$

And the expected score in bit units:

$$E = \sum_{i=1}^{20} \sum_{j=1}^{i} p_i \times p_j \times s_{ij}$$

4.2.3 Clustering segments within blocks

To reduce multiple contributions to amino acid pair frequencies from the most closely related members of a family, sequences are clustered within blocks and each cluster is weighted as a single sequence in counting pairs. A clustering percentage in which sequence segments identical for at least that value are clustered is used. The contribution of closely related segments to the frequency table is reduced. Varying the clustering percentage leads to a family of matrices.

4.2.4 Constructing blocks data bases

Protomat was used to build the block data base from 504 non redundant groups of proteins. Protomat uses an amino acid substitution matrix at two phases. THe motif program uses a substitution matrix when individual sequences are aligned against sequence segments containing a candidate motif. The Motomat program uses a substitution matrix when a block is extended to either side of the motif region. A unitary substitution matrix was used, next blosum was applied to the blocks and the resulting matrix was used to construct a second database. Then blousm was applied to the second data base and the resulting matrix was used to construct version of blocks data base. THe blosum program was applied to the final data base using a series of clustering percentages to obtain a family of lod substitution matrices. Similar matrices were obtained using PAM.

4.2.5 Alignments and homology searches

Global multiple alignments were done using Multalin and to provide a positive matrix each entry was increased by 8. Pearson's RDF2 program was used to evaluate local pairwise alignments. Homology searches were done using Blastp, Fasta and Ssearch. The Swiss-Prot data bank was searched. The first of the longest and most distance sequences in the group was used as a searching query, inferring distance from Protomat results. The results of each search were analysed by considering the sequences used by Protomat to construct blocks for the protein group as the true positive sequences. The number of misses is the nubmer of true positive sequences not reported for blastp. For fasta and ssearch the empirical evaluation criteria of Pearson was used: the number of misses is the number of true positive scores which ranked below the 99.5 percentile of the true negative scores.

4.3 Results

4.3.1 Comparison to Dayhoff matrices

The blosum series based on percent clustering can be compared to the Dayhoff matrices using a measure of average information per residue pair in bit units or relative entropy, which is 0 when the target distribution of pair frequencies is the same as the expected one and increases as they become more different. In the Dayhoff matrices relative entropy decreased when increasing PAM, while in blosum it increased linearly when increasing clustering percentage. Matrices with comparable relative entropy have similar expected scores.

4.3.2 Performance in multiple alignment of known structures

To test sequence alignment accuracy the results obtained to alignments seen in three dimensional structures was used. A simultaneous multiple alignment program MSA was used as a standard. Multalin, a hierarchical multiple alignment program performed worse using PAM matrices, while using blosum better. Therefore blosum matrices produced accurate global alignments of these sequences.

4.3.3 Performance in searching for homology in sequence data banks

The number of misses when searching was averaged in order to assess the overall searching performance of different matrices using blast, fasta and smith-waterman. Blosum matrices performed better than the best PAM matrix. BLOSUM improved detection of members od this family. The test was repeated with similar result of PAM for other protein families.

Chapter 5

FASTA

5.1 Introduction

5.1.1 Abstract

FASTA is a derivative of FASTP and can be used to search protein or DNA sequence data bases and compare a protein sequence to a DNA one by translating the DNA one as it is searched. FASTA includes an additional step in the calculation of the initial pairwise similarity score that allows multiple regions of similarity to be joined to increase the score of related sequences. RDF2 can be used to evaluate the significance of similarity score with a shuffling method that preserves local sequence composition. LFASTA can display all the regions of local similarity with scores greater than a threshold that can be displayed a graphic matrix plot.

5.1.2 Introduction

There is a trade-off between sensitivity and selectivity in biological sequence comparison: methods that can detect more distantly related sequences (increased sensitivity) frequently increase the similarity scores of unrelated sequences (decreased selectivity). FASTA use an improved algorithm that increases sensitivity with a small loss of selectivity and a negligible decrease in speed. LFASTA compute local similarity analyses of DNA or amino acid sequences.

5.2 Methods

5.2.1 The search algorithm

The search algorithm has four steps in determining a score for pair-wise similarity.

5.2.1.1 First step - lookup table

FASTA achieve its speed and selectivity in this step using a lookup table to locate all identities or groups of identities between two sequences. The *ktup* parameter determines how many consecutive identities are required in a match. The 10 best diagonal regions are found using a simple formula based on the number of *ktup* matches and the distance between the matches without considering shorter runs of identities, conservative replacements, insertions or deletions.

5.2.1.2 Second step - rescoring

In the second step the 10 regions are rescored using a scoring matrix that allows conservative replacements and runs of identities shorter than *ktup* to contribute to the similarity score. For each of these best diagonal regions, a subregion with minimal score is identified. This is the initial region.

5.2.1.3 Third step - ranking

The best scoring initial region is used to characterized pair-wise similarity and the initial scores are used to rank the library sequences. FASTA checks if several initial regions may be joined together. Given the location, their scores and a joining penalty, FASTA calculates an optimal alignment of initial regions as a combination of compatible regions with maximal score. The resulting score is used to rank the library sequences. Only for regions whose scores are above a threshold are used in this process.

5.2.1.4 Fourth step - alignment

The highest scoring library sequences are aligned by a modification of the Needleman-Wunsch and Smith-Waterman. The final comparison considers all possible alignments of the query and library that fall within a band centered around the highest scoring initial region.

5.2.2 Local similarity analyses

The program for detecting local similarities, LFASTA uses the same two steps for finding initial regions. Instead of saving 10, all diagonal regions with scores greater than a threshold are saved. Instead of focusing on a single region, LFASTA computes a local alignment for each initial region. It considers all initial regions and potential sequence alignments for some distance before and after the initial region. Starting at the end of the initial region, an optimization proceeds in the reverse direction until all possible alignment scores have gone to zero. The maximal local similarity score in the reverse direction is used to start a second optimization that proceeds in the forward direction. An optimal path starting from the forward maximum is displayed. The local homologies can be displayed as sequence alignments or on a two-dimensional graphic matrix sty. The maximal local similarity score in the reverse direction is used to start a second optimization that proceeds in the forward direction. An optimal path starting from the forward maximum is displayed. The local homologies can be displayed as sequence alignments or on a two-dimensional graphic matrix style plot

5.2.3 Statistical significance

To evaluate the statistical significance of an alignment RDF2 is used. RDF2 calculates three scores for each shuffled sequence: one from the best single initial region, a second from the joined initial regions and a third from the optimized diagonal. RDF2 can be used to evaluate amino acid or DNA sequences and allows to specify the scoring matrix. Moreover a global or local shuffle routine can be specified. Locally biased composition is the most common reason for high similarity scores of dubious biological significance. A Monte Carlo shuffle analysis that constructs random sequences by taking each residue in one sequence and placing it randomly along the length of the new sequence will break up these patches of biased composition. The scores may be lower and the sequences will appear to be related.

Chapter 6

BLAST

Chapter 7

How to BLAST

Chapter 8

Gapped BLAST and PSI BLAST

Chapter 9

PSI BLAST Composition-Based Statistics

Chapter 10

CLUSTAL-W

10.1 Introduction

CLUSTAL-W program implements a number of improvements to progressive multiple alignment methods, which are aimed at improving sensitivity without sacrificing speed and efficiency. Traditionally, one first aligns the most closely related sequences, gradually adding the more distant ones in order to obtain a faster alignment. This strategy works particularly well while working with sequences of different degrees of divergence. The main issues with the progressive approach are the local minimum problem and choice of alignment parameters.

10.1.1 Local minimum problem

The algorithm greedily adds sequences together, following the initial tree. There is no guarantee that the global optimal solution will be found and any mistakes made early in the alignment process cannot be corrected later. Even if the topology of the guide tree is correct,each alignment step in the multiple alignment process may have some percentage of the residues misaligned. The only way to correct this is to use an iterative or stochastic sampling procedure, which is not directly addressed in this paper.

10.1.2 The choice of alignment parameters

Traditionally, one chooses one weight matrix and two gap penalties (one for opening a new gap and one for extending an existing gap), which works quite well when sequences are closely related (as residue weight matrices give most weight to the identity). With very divergent sequences, however, the scores given to non-identical residues will become critically important; there will be more mismatches than identities.Different weight matrices will be optimal at different evolutionary distances or for different classes of proteins. Furthermore, the exact values of gap penalties are crucial for obtaining a correct solution.

In order to tackle the alignment parameter choice problem, the authors of the paper dynamically vary the gap penalties in a position- and residue-specific manner.

10.2 Alignment method

The basic multiple alignment algorithm consists of three main stages:

1. all pairs of sequences are aligned separately in order to calculate a distance matrix giving the divergence of each pair of sequences
2. a guide tree is calculated from the distance matrix;
3. the sequences are progressively aligned according to the branching order in the guide tree.

10.2.1 The distance matrix

In the original CLUSTAL programs,the pairwise distances were calculated using a fast approximate method. The scores are calculated as the number of k-tuple matches (runs of identical residues) in the best alignment between two sequences minus a fixed penalty for every gap. A more accurate score from full dynamic programming using two gap penalties (opening and extending gaps) and a full amino acid weight matrix is proposed. The scores are computed as the number of identities in the best alignment divided by the number of residues compared. Both of these scores are initially calculated as per cent identity scores and are converted to distances by dividing by 100 and subtracting from 1.0 to give number of differences per site.

10.2.2 The guide tree

The trees used to guide the final multiple alignment process are calculated from the distance matrix of step 1 using the Neighbour-Joining method. This produces unrooted trees with branch lengths proportional to estimated divergence along each branch. The root is placed by a "mid-point" method at a position where the means of the branch lengths on either side of the root are equal. These trees are also used to derive a weight for each sequence. The weights are dependent upon the distance from the root of the tree, but sequences which have a common branch with other sequences share the weight derived from the shared branch.

10.2.3 Progressive alignment

Use a series of pairwise alignments to align larger and large groups of sequences, following the guide tree from the tips of the rooted tree towards the root. Each step consists of aligning two existing alignments or sequences; gaps that are present in older alignments remain fixed. In order to calculate the score between a position from one sequence or alignmentan done from another,the average of all the pairwise weight matrix scores from the amino acids in the two sets of sequences is used,i.e.if you align 2 alignments with 2 and 4 sequences respectively,the score at each position is the average of $8(2 * 4)$comparisons. If either set of sequences contains one or more gaps in one of the positions being considered, each gap versus a residue is scored as zero. When sequences are weighted, each weight matrix value is multiplied by the weights from the two sequences.

10.3 Improvements to progressive alignment

All of the remaining modifications apply only to the final progressive alignment stage. Sequence weighting is relatively straightforward and is already widely used in profile searches, while the treatment of gap penalties is more complicated. Initial gap penalties are calculated depending on the weight matrix, the similarity of the sequences and the length of the sequences. Then,an attempt is made to derive sensible local gap opening penalties at every position in each prealigned group of sequences that will vary as new sequences are added. The use of different weight matrices as the alignment progresses is novel and largely bypasses the problem of initial choice of weight matrix. The final modification allows us to delay the addition of every divergent sequences until the end of the alignment process,when all of the more closely related sequences have already been aligned.

10.3.1 Sequence weighting

Sequence weights are calculated directly from the guide tree.The weights are normalised such that the biggest one is set to 1.0 and the rest are all less than 1.0.Groups of closely related sequences receive lowered weights because they contain much duplicated information.Highly divergent sequences without any close relatives receive high weights.These weights are used as simple multiplication factors for scoring positions from different sequences or prealigned groups of sequences.

10.3.2 Gap penalties

Gap opening penalty (GOP) and gap extension penalty (GEP) can be set by the user from a menu. The software then automatically attempts to choose appropriate gap penalties depending on following factors:

- **Dependence on the weight matrix**: the average score for two mismatched residues is used as a scaling factor for the GOP.

- **Dependence on the similarity of the sequences**: the per cent identity of the two sequences to be aligned is used to increase the GOP for closely related sequences and decrease it for more divergent sequences on a linear scale.

- **Dependence on the lengths of the sequences**: the scores for both true and false sequence alignments grow with the length of the sequences. The log of the length of the shorter sequence is used to increase the GOP with sequence length.

  $GOP \rightarrow GOP + log[min(N,M)]*$(average residue mismatch score)$*$(per cent identity scaling factor)

- **Dependence on the difference in the lengths of the sequences**: the GEP is modified depending on the difference between the lengths of the two sequences to be aligned. If one sequence is much shorter than the other,the GEP is increased to inhibit too many long gaps in the shorter sequence.

  $$GEP \rightarrow GEP * [1.0 + |log(N/M)|]$$

- **Position-specific gap penalties**: before any pair of sequences or prealigned groups of sequences are aligned, a table of gap opening penalties for every position in the two(sets of)sequences is generated. Initial gap opening penalty is manipulated in a position-specific manner, while the local gap penalty rules are applied in a hierarchical manner.

- **Lowered gap penalties at existing gaps**: if there are already gaps at a position,then the GOP is reduced in proportion to the number of sequences with a gap at this position and the GEP is lowered.

  $$GOP \rightarrow GOP * 0.3 * (\text{no. of sequences without gap/no. of sequences})$$

- **Increased gap penalties near existing gaps**: if a position does not have any gaps but is within 8 residues of an existiing gap, the GOP is increased by:

$$GOP \rightarrow GOP * 2 + [(8 - \text{distance from ga}) * 2]/8$$

- **Reduces gap penalties in hydrophilic stretches**: any run of 5 hydrophilic residues is considered to be a hydrophilic stretch. The residues that are to be considered hydrophilic may be set by the user but are conservatively set to D, E, G, K, N, Q, P, R or S by default. If, at any position, there are no gaps and any of the sequences has such a stretch,the GOP is reduced by one third.

- **Residue-specific penalties**: if there is no hydrophilic stretch and the position does not contain any gaps, then the GOP is multiplied by one of the 20 numbers according to a specific table.

10.3.3 Weight matrices

The weight matrices offered to the user are the Dayhoff PAM series and the BLOSUM series. In each case, there is a choice of matrix ranging from strict ones (useful for closely related sequences) to very "soft" ones. The distances are measured directly from the guide tree.

10.3.4 Divergent sequences

The most divergent sequences are usually the most difficult to align correctly. It is sometimes better to delay the incorporation of these sequences until all of the more easily aligned sequences are merged first, improving gap placements and matching weakly conserved positions against the rest of sequences. A choice is offered to set a cut-off, the default is 40% identity or less.

10.4 Discussion

As more divergent sequences are included,it becomes increasingly difficult to find good alignments and to evaluate them.What we find with CLUSTAL-W is that the basic block-like structure of the alignment(corresponding to the major secondary structure elements)is usually recovered,with some of the most divergent sequences misaligned in small regions. This is a very useful starting point for manual refinement, as it helps define the major blocks of similarity.

There are several areas where further improvements in sensitivity and accuracy can be made.

1. the residue weight matrices and gap settings can be made more accurate as more data accumulate and matrices for specific sequences types can be derived.

2. stochastic or iterative optimisation methods can be used to refine initial alignments

3. the average number of examples of each protein domain or family is growing steadily. New information should be used to make alignments more accurate

Chapter 11

T-COFFEE - ELISA

T-COFFEE method is broadly based on the popular progressive approach to multiple alignment but avoids the most serious pitfalls caused by the greedy nature of this algorithm. By pre-processing a data set of all pair-wise alignments between the sequences, a library is built to guide the progressive alignment. Intermediate alignments are then based not only on the sequences to be aligned next, but also on how all of the sequences align with each other. This alignment information can be derived from heterogeneous sources such as a mixture of alignment programs and/or structure superposition.

11.1 Introduction

The most commonly used heuristic methods are based on the progressive-alignment strategy, with ClustalW being the most widely used implementation. Although successful in a wide variety of cases, this method suffers from its greediness, as errors made in the first alignments cannot be rectified later as the rest of the sequences are added in. T-Coffee is an attempt to minimize that effect, and although being still a greedy progressive method, it allows for much better use of information in the early stages.

The main alternative to progressive alignment is the simultaneous alignment of all the sequences. Two such packages exist (MSA and DCA), based on the Carrilo and Lipman algorithm, but they remain an extremely CPU and memory-intensive approach. Iterative strategies do not provide any guarantees about finding optimal solutions but are reasonably robust and much less sensitive to the number of sequences than their deterministic counterparts. Alternatively, one might wish to consider local similarity, as occurs when two proteins share only a domain or motif. Lalign, a variant of SW algorithm from the FASTA package, produces sets of non-overlapping local alignments from the comparison of two sequences. For multiple sequences, the Gibbs sampler and Dialign2 are the main automatic methods, which perform well when there is a clear block of ungapped alignment shared by all of the sequences. They perform poorly, however, on general sets of test cases when compared with global methods. T-Coffee is aimed at providing a simple, flexible and, most importantly, accurate solution to the problem of how to combine global and local multiple alignments.

11.2 T-Coffee Algorithm

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) has two main features. First, it provides a simple and flexible means of generating multiple alignments, using heterogeneous data sources. The data from these sources are provided to T-Coffee via a library of pair-wise alignments. The second main feature is the optimization method, which is used to find the multiple alignment that best fits the pair-wise alignments in the input library. We use a so-called progressive strategy, which is similar to that used in ClustalW. T- Coffee is a progressive alignment with an ability to consider information from all of the sequences during each alignment step, not just those being aligned at that stage. The progressive alignment is speed and simple, and a lower tendency to make errors is observed. An overall scheme of T-Coffee algorithm is reported in figure 11.1.

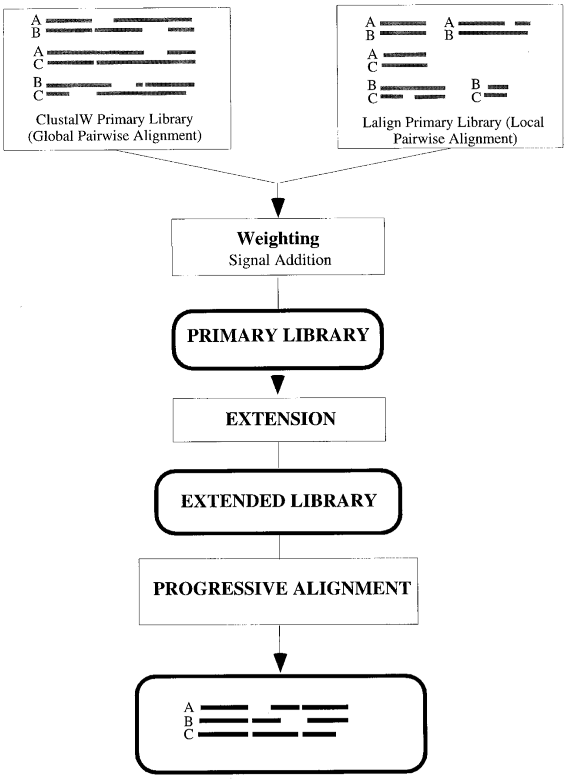

Figure 11.1

11.2.1 Generating a primary library of alignments

The primary library contains a set of pair-wise alignments between all of the sequences to be aligned. In the library, we include information on each of the $N(N-1)/2$ sequence pairs, where N is the number of sequences.

- global alignments are computed with ClustalW

- local alignments are the ten top- scoring non-intersecting local alignments, between each pair of sequences, gathered using the Lalign

In the library, each alignment is represented as a list of pair-wise residue matches (e.g. residue x of sequence A is aligned with residue y of sequence B). In effect, each of these pairs is a constraint. All of these constraints are not equally important, some may come from parts of alignments that are more likely to be correct. We take this into account when computing the multiple alignment and give priority to the most reliable residue pairs by using a weighting scheme.

11.2.2 Derivation of the primary library weights

T-Coffee assigns a weight to each pair of aligned residues in the library. An ideal primary weight will reflect the correctness of a constraint. Sequence identity is used, as it is known to be a reasonable indicator of accuracy when aligning sequences with more than 30% identity. This weighting scheme proved to be highly effective for a previous consistency-based objective function and has the advantage of great simplicity. Each constraint receives a weight equal to percent identity within the pair-wise alignment it comes from.

11.2.3 Combination of the libraries

ClustalW and Laling primary libraries are pooled in a simple process of addition. If any pair is duplicated between the two libraries, it is merged into a single entry that has a weight equal to the sum of the two weights. Otherwise, a new entry is created for the pair being considered. Pairs of residues that did not occur are not represented (by default they will be considered to have a weight of zero). This primary library can be used directly to compute a multiple sequence alignment. For each pair of aligned residues in the library, we can assign a weight that reflects the degree to which those residues align consistently with residues from all the other sequences.

11.2.4 Extending the library

Fitting a set of weighted constraints into a multiple alignment is a well-known problem, formulated as an instance of the "maximum weight trace", an NP-complete problem. The genetic algorithm is robust but may require prohibitive computation time, while the graph-theory-based algorithm has a complexity only partially characterized and may fail in some cases for reasons that are difficult to predict. The overall idea of the heuristic library extension algorithm is to combine information in such a manner that the final weight, for any pair of residues, reflects some of the information contained in the whole library. To do so, a triplet approach is used: the weight associated with a pair of residues will be the sum of all the weights gathered through the examination of all the triplets involving that pair. The more intermediate sequences supporting the alignment of that pair, the higher its weight. Extension will be carried out on each pair of residues of A and B. Once the operation is complete, sequence pair A and B will have gathered information from all the other sequences in the set. The complete set of pairs constitutes the extended library. The worst-case complexity of this computation is $O(N^3L^2)$ with L being the average sequence length. However, this will only occur when all the included pair-wise alignments are totally inconsistent. In practice the complexity is close to $(O)N^3L$.

11.2.5 Progressive alignment strategy

This alignment uses the weights in the extended library above to align the residues in the two sequences. This pair of sequences is then fixed and any gaps that have been introduced cannot be shifted later. Then the next closest two sequences are aligned or a sequence is added to the existing alignment of the first two sequences, depending which is suggested by the guide tree. The next two closest sequences or pre-aligned group of sequences are always joined.,until all the sequences have been aligned. To align two groups of pre-aligned sequences the scores from the extended library are used, as before, but the average library scores in each column of existing alignment are taken. The procedure does not require any additional parameters such as gap penalties, as the substitution values (the library weights) were computed on alignments where such penalties had already been applied. Furthermore, high scoring segments that show consistency within the data set see their score enhanced by the extension to such a point that they become insensitive to gap penalties. In practice, this means that during the progressive phase, we use a dynamic-programming algorithm with gap-opening penalties and gap- extension penalties set to zero for aligning two sequences or two groups of pre-aligned sequences.

11.3 Discussion

T-Coffee is a new progressive method for sequence alignment. It can combine signals from heterogeneous sources (e.g. sequence-alignment programs, structure alignments, threading, manual alignment, motifs and specific constraints) into a unique consensus multiple sequence alignment. The combination of local and global alignments leads to a significant increase in alignment accuracy. The main difference from traditional progressive alignment methods is that, instead of using a substitution matrix for aligning the sequences, a position-specific scoring scheme is used (the extended library). Errors are less likely to occur during early stages of the progressive alignment. As a consequence, even though the paradigm "once a gap always a gap" remains true, misplacing gaps becomes much less likely. The second important feature of T-Coffee is the combination of local and global information. The end-user benefits from the simplicity of the method and does not need to provide any extra parameter values.

A key ingredient of the method is the primary weighting scheme. The main reason why T-Coffee can tolerate small segments with high similarity is more likely to occur by chance is because short high-scoring segments are rarely consistent enough to have a strong effect on the position-specific scoring scheme after extension. Moreover, final alignments are processed using dynamic programming,making it less likely for misplaced high-scoring segments to affect the alignment. Although the protocol proposed here employs a minimal combination of local and global information, there is no theoretical limit to the number of methods that can be used. For instance, alignments from structural comparisons could be combined with sequence alignments. It is also possible to incorporate, in the library, information extracted from multiple alignments.

Chapter 12

T-COFFEE - ILARIA

T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment We describe a new method (T-Coffee) for multiple sequence alignment that provides a dramatic improvement in accuracy with a modest sacrifice in speed as compared to the most commonly used alternatives.

12.1 Introduction

The simultaneous alignment of three or more nucleotide or amino acid sequences is one of the commonest tasks in bioinformatics. Multiple alignments are an essential prerequisite to many analyses of protein families such as homology modeling or phylogenetic reconstruction, or to illustrate conserved and variable sites within a family.

12.1.1 Reasons for the development

12.1.1.1 Once a gap, always a gap

The most commonly used heuristic methods are based on the **progressive alignment strategy**, with **ClustalW** being the most widely used implementation. The idea is to take an initial, approximate, phylogenetic tree between the sequences and to gradually build up the alignment, following the order in the tree. Although successful, this method suffers from its greediness. Errors made in the first alignments cannot be rectified later as the rest of the sequences are added in. Even though the paradigm “once a gap always a gap” remains true, misplacing gaps becomes much less likely.

12.1.1.2 Global and local alignment

Some methods attempt to carry out global alignments, where one tries to align the full lengths of the sequences with each other. Alternatively, one might wish to consider local similarity, as occurs when two proteins share only a domain or motif (Smith-Waterman, Lalign). In principle, a method able to combine the best properties of global and local multiple alignments might be very powerful. This is the second motivation for T-Coffee: the design of a method that provides a simple, flexible and, most importantly, accurate solution to the problem of how to combine information of this sort.

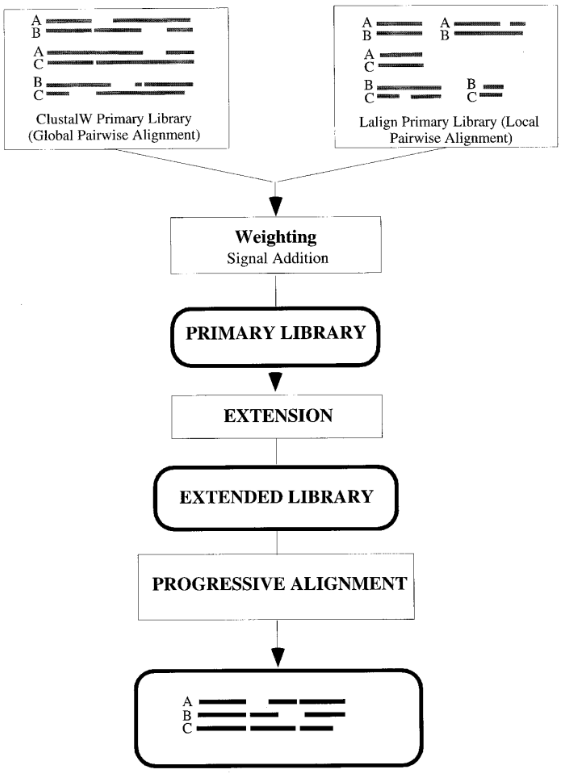

Figure 12.1: Layout of the T-Coffee strategy; the main steps required to compute a multiple sequence alignment using the T-Coffee method. Square blocks designate procedures while rounded blocks indicate data structures.

12.2 T-Coffee Algorithm

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) has two main features.
First, it provides a simple and flexible means of generating multiple alignments, using heterogeneous data sources. The data from these sources are provided to T-Coffee via a **library** of pair-wise alignments.
The main feature of T-Coffee is the **optimization** method, which is used to find the multiple alignment that best fits the pair-wise alignments in the input library. A so-called **progressive strategy** is implemented, which is similar to that used in ClustalW. The difference from CustalW is that T-Coffe makes use of the information in the library to carry out progressive alignment in a manner that allows us to consider the alignments between all the pairs while we carry out each step of the progressive multiple alignment. This gives us progressive alignment, with all its advantages of speed and simplicity, but with a far lesser tendency to make errors like the one shown in Figure 12.2(a), i.e. misalignment of the word CAT. T-Coffee is a progressive alignment with an ability to consider information from all of the sequences during each alignment step, not just those being aligned at that stage.

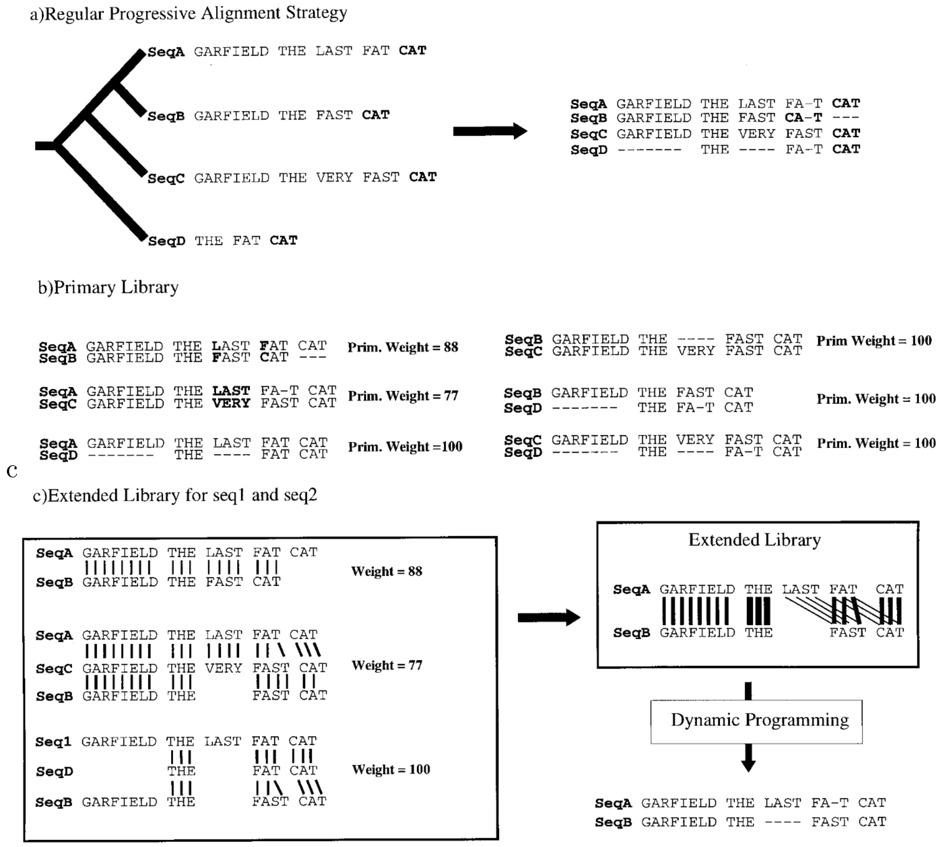
a)Regular Progressive Alignment Strategy
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
SeqC GARFIELD THE VERY FAST CAT
SeqD THE FAT CAT
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CA-T ---
SeqC GARFIELD THE VERY FAST CAT
SeqD ------- THE ---- FA-T CAT
b)Primary Library
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT ---
Prim. Weight = 88
SeqB GARFIELD THE ---- FAST CAT
SeqC GARFIELD THE VERY FAST CAT
Prim Weight = 100
SeqA GARFIELD THE LAST FA-T CAT
SeqC GARFIELD THE VERY FAST CAT
Prim. Weight = 77
SeqB GARFIELD THE FAST CAT
SeqD ------- THE FA-T CAT
Prim. Weight = 100
SeqA GARFIELD THE LAST FAT CAT
SeqD ------- THE ---- FAT CAT
Prim. Weight =100
SeqC GARFIELD THE VERY FAST CAT
SeqD ------- THE ---- FA-T CAT
Prim. Weight = 100
C
c)Extended Library for seq1 and seq2
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
Weight = 88
SeqA GARFIELD THE LAST FAT CAT
SeqC GARFIELD THE VERY FAST CAT
SeqB GARFIELD THE FAST CAT
Weight = 77
Seq1 GARFIELD THE LAST FAT CAT
SeqD THE FAT CAT
SeqB GARFIELD THE FAST CAT
Weight = 100
Extended Library
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
Dynamic Programming
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE ---- FAST CAT

Figure 12.2: The library extension. **(a) Progressive alignment**. Four sequences have been designed. The tree indicates the order in which the sequences are aligned when using a progressive method such as ClustalW. The resulting alignment is shown, with the word CAT misaligned. **(b) Primary library**. Each pair of sequences is aligned using ClustalW. In these alignments, each pair of aligned residues is associated with a weight equal to the average identity among matched residues within the complete alignment (mismatches are indicated in bold type). **(c) Library extension for a pair of sequences**. The three possible alignments of sequence A and B are shown (A and B, A and B through C, A and B through D). These alignments are combined, as explained in the text, to produce the position-specific library. This library is resolved by dynamic programming to give the correct alignment. The thickness of the lines indicates the strength of the weight.

12.2.1 Generating a primary library of alignments

The primary library contains a set of pair-wise alignments between all of the sequences to be aligned. In the library, we include information on each of the $N(N-1)/2$ sequence pairs, where N is the number of sequences. Here, we use two alignment sources for each pair of sequences, one **local** and one **global**. The global alignments (Figures 12.1 and 12.2(b)) are constructed using ClustalW on the sequences, two at a time. The local alignments (Figure 12.1) are the ten top scoring non-intersecting local alignments, between each pair of sequences, gathered using the Lalign program.

In the library, each alignment is represented as a list of pair-wise residue matches (e.g. residue x of sequence A is aligned with residue y of sequence B). In effect, each of these pairs is a constraint. All of these constraints are not equally important. Some may come from parts of alignments that are more likely to be correct. This is taken into account when computing the multiple alignment and give priority to the most reliable residue pairs. This is achieved by using a weighting scheme, which is described in subsection 12.2.2.

12.2.2 Derivation of the primary library weights

T-Coffee assigns a weight to each pair of aligned residues in the library (Figure 2(b)). The **sequence identity** weighting scheme is used, which has been prove to be effective and of great simplicity. Libraries are lists of weighted pair-wise constraints. Each constraint receives a weight equal to percent identity within the pair-wise alignment it comes from (Figure 12.2(b)). For each set of sequences, two primary libraries are computed along with their weights, one using ClustalW (global alignments; Figure 12.2(b)) and the second using Lalign (local).

12.2.3 Combination of the libraries

The aim is the efficient combination of local and global alignment information. This is achieved by pooling the global and local primary libraries in a simple process of **addition**. If any pair is duplicated between the two libraries, it is merged into a single entry that has a weight equal to the sum of the two weights. Otherwise, a new entry is created for the pair being considered (process called unofficially **"stacking" of the signal**). Pairs of residues that did not occur are not represented (weight of zero).

This primary library can be used directly to compute a multiple sequence alignment. However, we enormously increase the value of the information in the library by examining the consistency of each pair of residues with residue pairs from all of the other alignments. For each pair of aligned residues in the library, we can assign a weight that reflects the degree to which those residues align consistently with residues from all the other sequences. This process is called library extension (subsection 12.2.4).

12.2.4 Extending the library

Fitting a set of weighted constraints into a multiple alignment is a well known NP-complete problem. We circumvent the problem by using a heuristic algorithm that we call **library extension** (Figure 12.2(c)). The idea is to combine information in such a manner that the final weight, for any pair of residues, reflects some of the information contained in the whole library. To do so, a **triplet approach** is used, as summarized in Figure 12.2(c).

It is based on taking each aligned residue pair from the library and checking the alignment of the two residues with residues from the remaining sequences.

12.2.4.1 A quick example

For instance, let us consider the four sequences A, B, C and D of Figure 2. Let us call A(G) the G of GARFIELD in sequence A, B(G) the equivalent G in sequence B and W(A(G), B(G)) the weight associated with this pair of symbols in the primary library. In the direct alignment of A and B, A(G) and B(G) are matched (Figure 12.2(b) and (c)). Therefore, the initial weight for that pair of residues can be set to 88 (primary weight of the alignment of sequence A and B, which is the percent of identity of this pair). If we now look at the alignment of sequence A and sequence B through sequence C (Figure 12.2(c)), we can see that the A(G) and C(G) are aligned, as well as C(G) and A(G). We conclude that there is an alignment of A(G) with B(G) through sequence

C. We associate that alignment with a weight equal to the minimum of $W_1 = W(A(G), C(G))$ and $W_2 = W(C(G), B(G))$. Since $W_1 = 77$ and $W_2 = 100$, the resulting weight is set to 77. In the extended library, this new value is added to the previous one to give a total weight of 165 for the pair A(G), B(G). The complete extension will require an examination of all the remaining triplets. Not all of them bring information. For instance, the alignment of A and B through sequence D does not contain any information relative to A(G) or B(G), and, therefore, it has no influence on the weight associated with A(G) and B(G). In summary, the weight associated with a pair of residues will be the sum of all the weights gathered through the examination of all the triplets involving that pair.

12.2.4.2 Alignment

Weights will be zero for any residue pairs that never occur (this will be true of the majority of residue pairs). Otherwise, the weight will reflect a combination of the similarity of the pair of sequences or sequence segments that the residue pair comes from and the consistency of that residue pair with all other residue pairs in the primary library. These scores can then be used to align any two sequences from our data set using conventional dynamic programming. When one normally aligns a pair of sequences, one uses a set of scores derived from some general table of amino acid weights such as a Blosum matrix. In our case, we can replace that matrix with a set of scores that are specific to every possible pair of residues in our two sequences. This will allow an alignment to be carried out that will take account of the particular residues in the two sequences but will also be guided towards consistency with all of the other sequences in the data set.

12.2.5 Progressive alignment strategy

The *normal* progressive alignment strategy consists in creating a guide tree (a phylogenetic tree) using the neighbor-joining method. The closest two sequences on the tree are aligned first using normal dynamic programming. This pair of sequences is then fixed and any gaps that have been introduced cannot be shifted later. Then the next closest two sequences are aligned or a sequence is added to the existing alignment of the first two sequences, depending which is suggested by the guide tree.

As used here, the procedure does not require any additional parameters such as gap penalties. This stems, in part, from the fact that the substitution values (the library weights) were computed on alignments where such penalties had already been applied. Furthermore, high scoring segments that show consistency within the data set see their score enhanced by the extension to such a point that they become insensitive to gap penalties.

12.3 Biological validation

I chose not to report the comparisons with other tools and the complexity of the algorithm. If needed, exhaustive tabled can be found in the paper.

12.3.1 Application to serine/threonine kinases

A major application of any alignment algorithm will be the delineation of motifs or domains. 19 sequences from a sub-family of the serine/threonine kinases were provided. Each sequence in the alignment contains a nucleotide-binding site (NBS). In all these sequences, the NBS is followed by a second conserved motif toward the C terminus. T-Coffee was able to accurately align 18 of the 19

NBSs, ClustalW was only able to correctly align 16 of these NBSs. The second motif is more difficult because of the long indel in st11 yeast. Here as well, T-Coffee can properly align 18 of the motifs, while ClustalW get 15 correct. As a result of combining local and global alignment information, T-Coffee managed to align almost all of the motifs as in the BaliBase reference alignment. Moreover, T-Coffee was the only program that correctly aligned the second motif of kp68 human, which is an interferon-induced kinase.

Chapter 13

Protein profiles with HMMs

13.1 Introduction

An alignment of two or more sequences can be used to infer the evolutionary relationships between sets of protein (or nucleic acid) sequences and eventually predict their 3D structure. In order to obtain meaningful results, the accuracy of the alignment must be as high as possible. Consider a family of protein sequences that all have a common three-dimensional structure, for example, the family of globins. A **profile** of globins can be thought of as a statistical model for the family of globins, in that for any sequence of amino acids, it defines a probability for that sequence, such that globin sequences tend to have much higher probabilities than non-globin sequences. The aim of this paper is to build a class of structures related to profiles and propose them as statistical models of protein families, such that we are able to "learn" or "estimate" the stochastic model directly from raw unaligned sequences. Once the model is obtained, it is possible to use it to discriminate sequences in the family from sequences not in the family, or to obtain a multiple alignment of all the sequences in the family to the model.

The proposed method of multiple alignment is quite different from conventional methods, which are usually based on pairwise alignments using the standard dynamic programming scheme with gap penalties. Such alignments often depend strongly on the particular values of the parameters required by the method, in particular the gap penalties. Furthermore, the same penalties are used for both fairly conserved regions and highly variable regions; substitutions, insertions, or deletions in a region of high conservation should ideally be penalized more than in a variable region. The statistical model here proposed corresponds to multiple alignment with variable penalties, depending on the position and learned from the data itself.

The type of statistical model is a hidden Markov model (HMM, or simply “model” for short). The HMM we build identifies a set of positions that describe the (more-or-less) conserved first-order structure in the sequences. In biological terms, this corresponds to identifying the core elements of homologous molecules. Each of these positions may be viewed as corresponding to a column in a multiple alignment of the sequences, or to a position in space, Often not all positions are modeled, but only the ones with the most significantly nonrandom statistics. As in a profile, for each position in the model, the relative probabilities of the different amino acids in that position are given, as is the probability that the amino acid at that position is deleted. For any amino acid x,the negative logarithm of the probability of x at a given position in the model can be interpreted as a penalty for an alignment that puts z in this position, and the negative logarithm of the probability of deletion at that position can be similarly interpreted as a penalty for an alignment that puts a " - " (gap character) in that position. Furthermore, for each pair of adjacent positions, the model includes the probability for initiating an insertion of one or more amino acids between these positions, and the probability for extending it. Other things are also included, such as the probability that an amino acid at a particular position is deleted, given that the amino acid at the previous position was deleted. This helps model the higher probability of consecutive deletions/insertions in certain regions. By taking negative logarithms as above, these methods provide a very flexible, position-dependent form of initiation and extension gap penalties.

13.2 The protein HMM

A hidden Markov Model describes a series of observations by a "hidden" stochastic process - a Markov process. In this paper we are proposing an HMM similar to the ones used in speech to model protein families - or families of other molecular sequences like DNA and RNA. When modeling proteins, the observations are the amino acids in the primary sequence of a protein. A model for a set of proteins is one that assigns high probability to the sequences in that particular set.

The HMM contains a sequence of M states that we call match states. These correspond to positions in a protein or columns in a multiple alignment. Each of these states can generate a letter x from the 20- letter amino acid alphabet according to a distribution $P(x|m_k), k = 1...M$. For each match state m_k there is a delete state d_k that does not produce any amino acid, it is a "dummy" state, shown as circles. Finally, there are insertion states between all the match states, $M + 1$ in total, shown as diamonds. They generate amino acids in exactly the same way as the match states, but they use probability distributions $P(x|i_k)$. From each state, there are three possible transitions to other states: transitions into match or delete states always move forward in the model, whereas transitions into insertion states do not. Multiple insertions between match states can occur because of the self loop on the insert state, meaning that a transition from an insert state to itself is possible. The transition probability from state s_1 to s_2 is called $\mathcal{T}(s_2|s_1)$. The overall model is reported in figure 13.1.

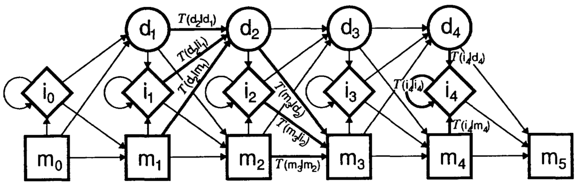

Figure 13.1

A sequence can be generated by a "random walk" through the model as follows: beginning at state m_0 (BEGIN) choose a transition randomly according to the probabilities $\mathcal{T}(m_1|m_0), \mathcal{T}(d_1|m_0), \mathcal{T}(i_0|m_0)$. If m_1 is chosen, generate the first amino acid x_1 from the probability distribution $P(xlm_1)$, and choose a transition to the next state according to probabilities $\mathcal{T}(\cdot|m_1)$where " $\cdot$ " indicates any possible next state. Continue in this manner all the way to the END state, generating a sequence of amino acids $x_1, \ldots, x_L$by following a path of states $y_0 \ldots y_{N+1}$ through the model, where $y_0 = m_0$ (the BEGIN state) and $y_N + 1 = m_{M+1}$ (the END state). Because the delete states do not produce any amino acid, N is larger than or equal to L.

If y_i is a match or insert state, we define $l(i)$ to be the index in the sequence $x_1, \ldots, x_L$ of the amino acid produced in state y_i. The probability of the event that the path $y_0 \ldots y_{N+1}$ is taken and the sequence

$$\text{Prob}(x_1, \ldots, x_L, y_0 \ldots y_{N+1}) = \mathcal{T}\,(m_{N+1} \mid y_N) \prod_{i=1}^{N} \mathcal{T}\,(y_i \mid y_{i-1})\, \mathcal{P}(x_{l(i)}|y_i) \tag{13.1}$$

where we set $\mathcal{P}(x_{l(i)}|y_i) = 1$ if y_i is a delete state. The probability of any sequence $x_1, \ldots, x_L$ of amino acids is a sum over all possible paths that could produce that sequence]which we write as follows:

$$\text{Prob}(x_1, \ldots, x_L) = \sum_{\text{paths}} \text{Prob}(x_1, \ldots, x_L, y_0 \ldots y_{N+1}) \tag{13.2}$$

In this way a probability distribution on the space of sequences is defined. The goal is to find a model that accurately describes a family of proteins by assigning large probabilities to sequences in that family.

This structure for the HMM captures the structural intuition of a protein as a sequence of positions, each with its own distribution over the amino acids, along with the possibility for either skipping a position or inserting extra amino acids between consecutive positions, and allowing for the possibility that once a position is skipped, it may be more likely that positions following that one are also skipped. This choice appears to have worked well for modeling the globins, but one can choose any structure for the states and transitions that is appropriate for the problem at hand.

13.3 Estimating the HMM

All the parameters in the model (i.e., probabilities) could in principle be chosen by hand from an existing alignment of protein sequences, or from information about the three-dimensional structure of proteins. The novel approach we take is to “learn” the parameters entirely automatically from a set of unaligned primary sequences, using an Expectation Maximization algorithm. The particular method we use to learn the model can be viewed as a computationally efficient approximation to Maximum A Posteriori (MAP) estimation of the parameters of the model.

In MAP estimation, the posterior probability is defined using Bayes rule,

$$\text{Prob}(\text{model}|\text{sequences}) = \frac{\text{Prob}(\text{sequences}|\text{model})\,\text{Prob}(\text{model})}{\text{Prob}(\text{sequences})} \tag{13.3}$$

The prior on the models contains prior beliefs on what a model should be like, and can be used to “penalize” models that are known to be bad or uninteresting. To find the best model one can optimize the posterior probability 13.3 with respect to the parameters of the model.

13.3.1 The distance from sequence to model

First, we need a way to calculate the probability Prob(sequences|model). Call the sequences $s^1 \ldots s^n$. Then, assuming independence

$$\text{Prob}(s^1 \ldots s^n \mid \text{ model }) = \prod_{\mu=1}^{n} \text{Prob}(s^\mu \mid \text{ model }). \tag{13.4}$$

The expression for Prob(s^μ|model) is given in can be calculated using the "forward algorithm". The previous equation can be approximated by the following for simplicity:

$$\operatorname{Prob}(s \mid \text{ model }) \simeq \max_{\text{paths}} \operatorname{Prob}(s, \text{path } \mid \text{ model}) \tag{13.5}$$

Instead of maximizing the probability, it is most convenient to minimize the negative logarithm of the probability, which is defined as the distance from the sequence to the model.

$$\text{dist } (s, \text{ model }) = -\min_{\text{paths}} \log \operatorname{Prob}(s, \text{ path } \mid \text{ model }) = -\min_{\text{paths}} \sum_{i=1}^{N+1} \left[\log \mathcal{T}\left(y_i \mid y_{i-1}\right) + \log \mathcal{P}\left(x_{l(i)} \mid y_i\right)\right) \tag{13.6}$$

Where:

- path: $y_0 \ldots y_{N+1}$
- sequence s: $x_1 \ldots x_L$
- penalty: $\mathcal{T}\left(y_i \mid y_{i-1}\right)$

The penalties depends on the position in the model and the distance is very similar to the standard "edit distance" from one sequence to another. The most probable path can be found using the usual backtracking technique, as explained in the following section.

13.3.2 Estimation algorithm

The Baum-Welch or forward-backward algorithm is a version of the general EM (Expectation-Maximization) method often used in statistics, which can be applied faster but using the Viterbi approximation. The Viterbi-EM adaptation of the model to the sequences is an iterative process. It proceeds as follows:

1. choose the initial model
2. find the most probable path through the model for each of the sequences. The number of these paths that pass through state y is denoted $n(y)$, the number of times the transition from y to y' occurs $n(y'|y)$, and the number of times an amino acid x was generated in state y is denoted $m(x|y)$. Obviously $n(m_k) = n(m_{k+1}|m_k) + n(d_{k+1}|m_k) + n(i_k|m_k)$ and similarly for delete and insert states
3. Reestimate the transition probabilities and amino acid probabilities in the match and insertion states from

$$\mathcal{T}\left(y' \mid y\right) = \frac{n\left(y' \mid y\right) + R\left(y' \mid y\right)}{n(y) + \sum_{y''} R\left(y'' \mid y\right)} \tag{13.7}$$

$$\mathcal{P}(x \mid y) = \frac{m(x \mid y) + R(x \mid y)}{n(y) + \sum_{i=1}^{20} R\left(a_i \mid y\right)} \tag{13.8}$$

Here $R(y'|y)$ and $R(x|y)$ represent a regularizer, which comes from the model prior 13.3 We assume that $R(y\text{''}|y) = 0$ if there is no transition from state y to state y". Often R is just set equal to one in order to avoid zero probabilities, but it can be used to bias the model in a certain direction.

4. repeat step 2 and 3 until the parameters change only insignificantly

The main difference between this method, which uses the Viterbi approximation, and the standard Baum-Welch algorithm is that in the latter all paths (not just the most probable) are considered, but weighted according to their probability.

13.3.3 Choosing the length of the model

A simple heuristic selects a good model length, and even helps in the problem of local minima. After learning, if more than a fraction γ_{del} of the optimal paths of the sequences choose d_k, the delete state at position k,that position is removed from the model. Similarly, if more than a fraction γ_{ins} msake insertions at position k (in state i_k), a number of new positions are inserted into the model after position k. This number is equal to the average number of insertions the sequences make at that position. After these changes in the model, it is retrained, and this cycle is repeated until no more changes are needed. We call this “model surgery”. Currently we choose γ_{del} and γ_{ins} to be 1/2.

13.3.4 Over-fitting and regularization

A model with too many free parameters cannot be estimated well from a relatively small data set of training sequences. If we try to estimate such a model, we run into the problem of overfitting. One way to deal with this problem is to control the effective number of free parameters in the model by regularization, which can be viewed as a way of implementing MAP estimation. In our application of regularization to HMMs, the regularizer is closely related to the model prior, and it can be used to bias the model in some specific direction. The regularizer (R)is added to the frequency counts in 13.7 and 13.8 as if we actually saw more than $n(y'|y)$ or $n(x|y)$ events. For instance, in our models we believe a prior i that sequences should spend most of their time in the match states on the main line of the HMM, so we incorporate this belief into the model prior by setting $R(m_i|y)$to a large value compared to the regularizer $R(y|m_i)$,which is added to transitions diverging from the main line. In so doing, these parameters become less adjustable, and as a result we are less prone to overfit the data. We also tend to get better models because we are encouraging the learning algorithm to first explore the more a prior i likely part of model space when trying to fit the data. In the extreme case, the regularizers $R(\cdot|\cdot)$can be such large numbers that the parameters simply become “hardwired”. We have found that the method works best when very large regularizers $R(x|y)$, proportional to the relative overall frequencies of amino acids in globin sequences, are used when y is an insert state. This has the effect of fixing the distributions in the insert states, rather than trying to estimate them from the training sequences.

13.4 Conclusions

The method utilizes the tremendous amount of information contained in many sequences from the same family. For the simple case of globins, with this method it is possible to obtain multiple alignments that mirror structural alignments using only the unaligned primary sequences as input. The method requires that many sequences be available from the family one wants to model; since the number of sequences in the protein databases is growing rapidly, this may be less of a problem in the future. There are two possible answers to the problem of small sets of training sequences.

The first is to add more prior knowledge into the training process by starting with a better initial model, and by using more involved kinds of regularization. The "soft tying" regularization method combines the idea of tying states in which the number of free parameters is reduced by having groups of states all share the same distribution on the output alphabet (the 20 amino acids in this case), and the idea of soft weight, in which the regularizer (in this case for the distribution of amino acids) is also adaptively modified during learning. Even more complex, position-specific kinds of regularization are possible. Switching from the alphabet of the primary sequences to a different representation based more on the structural or chemical properties of the amino acids in the sequence may also help.

The second answer is to give up trying to model the whole protein in cases where only few sequences are known, and instead try to model only pieces of the protein (motifs or domains) that have the same or similar structure in other proteins. In this case we can find more training sequences by adding in the corresponding pieces of these other proteins. In fact, the EM method can be used to locate the appropriate pieces by itself, given the entire sequences. Several models can also be linked together with free insertions in between, to model more than one subsequence in the molecules.

Finally, when a relatively large number of sequences are available, it is sometimes possible to get better results by dividing these sequences into clusters of similar sequences, and training a different model for each cluster. The partition into clusters can also be done automatically by the EM algorithm during training, in a procedure known as mixture modeling. To implement this, we simply start with more than one model and force the models to “compete” for the sequences, and adapt to fit only the ones they “won”.

Chapter 14

Construction of phylogenetic trees

Chapter 15

Toward defining the course of evolution: minimum change for a specific tree topology

Chapter 16

Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach