Chapter 2

Comparison of protein sequences

During the last three decades a considerable effort has been made to develop algorithms that compare sequences of macromolecules (proteins, DNA). The purpose of such algorithms is to detect evolutionary, structural and functional relations among the sequences. Successful sequence comparison would allow to infer the biological properties of new sequences from data accumulated on related genes. For example, a similarity between a translated nucleotide sequence and a known protein sequence suggests a homologous coding region in the corresponding nucleotide sequence. Significant sequence similarity among proteins may imply that the proteins share the same secondary and tertiary structure, and have close biological functions. The prediction of unknown protein structures is often based on the study of known structures of homologous proteins.

This chapter is a survey of sequence comparison, scoring schemes, and the statistics of sequence alignments which are essential for the purpose of distinguishing true relations among proteins from chance similarities. The chapter is based on books by M. Waterman [1995] and by Setubal and Meidanis [1996], as well as on various papers in this field.

2.1 Alignment of sequences

As is well known, the DNA molecule serves as a blueprint for the genetic information of every organism. Therefore, the evolution of organisms must be related to changes in the DNA. The simplest events of molecular evolution are the substitution of one base by another (point mutations) and the insertion or deletion of a base pair. Suppose that the sequence **b** is obtained from the sequence **a** by substitutions, insertions and deletions. It is customary and useful to represent the transformation by an alignment where **a** is written above **b** with the common (conserved) bases aligned appropriately. For example, say that $\mathbf{a} = ACTTGA$ and **b** is obtained by substituting the second letter from C to G, inserting an A between the second and the third letters, and by deleting the fifth base (G). The corresponding alignment will be:

$$\begin{array}{ccccccccc} \mathbf{a} & = & A & C & - & T & T & G & A \\ \mathbf{b} & = & A & G & A & T & T & - & A \end{array}$$

We usually do not actually know which sequence evolved from the other. Therefore the events are not directional and insertion of A in **b** might have been a deletion of A in **a**.

In a typical application we are given two related sequences and we wish to recover the evolutionary events that transformed one to the other. The goal of sequence alignment is to find the *correct* alignment that encodes the true series of evolutionary events which have occurred. The alignment can be assigned a score which accounts for the number of identities (a match of two identical letters), the number of substitutions (a match of two different letters), and the number of gaps (insertions/deletions). With high scores for identities, and low scores for substitutions and gaps, the basic strategy towards tracing the correct alignment seeks the alignment which scores best (see next section for details).

The algorithms described below may be applied to the comparison of protein sequences as well as to DNA sequences (coding or non-coding regions). Though the evolutionary events occur at the DNA level, the main genetic pressure is on the protein sequence. Consequently, the comparison of protein sequences has proven to be a much more effective tool [Pearson 1996]. Mutations at the DNA level do not necessarily change the encoded amino acid due to the redundancy of the genetic code. Mutations often result in conservative substitutions at the protein level, namely, replacement of an amino acid by another amino acid with similar biochemical properties. Such changes tend to have only a minor effect on the protein's functionality. Within the scope of this work, and in view of the last paragraph, this chapter focuses on the comparison of protein sequences.

2.1.1 Global similarity of protein sequences

Let $\mathbf{a} = a_1 a_2 ... a_n$ and $\mathbf{b} = b_1 b_2 ... b_m$ where $a_i, b_j \in \mathcal{A}$, the alphabet of amino acids, be two given protein sequences . A **global alignment** of these sequences is an alignment where all letters of **a** and **b** are accounted for.

Let $s(a_i, b_j)$ be the similarity of a_i, b_j and let $\alpha > 0$ be the penalty for deleting/inserting of one amino acid. The score of an alignment with N_{ij} matches of a_i and b_j and N_{gap} insertions/deletions is defined as

$$\sum_{i,j} N_{ij} \cdot s(a_i, b_j) - N_{gap} \cdot \alpha$$

The **global similarity** of sequences **a** and **b** is defined as the largest score of any alignment of sequences **a** and **b**, i.e.

$$S(\mathbf{a}, \mathbf{b}) = max_{alignments} \{ \sum_{i,j} N_{ij} \cdot s(a_i, b_j) - N_{gap} \cdot \alpha \}$$

Usually, pairwise scores are the logarithm of likelihood ratios (log-odds), as explained in section 2.4. Therefore, the definition of the alignment score as the sum of the pairwise scores may be interpreted as likelihood. The corresponding similarity score is thus essentially a maximum likelihood measure.

How to find the best alignment ? The exponentially large number of possible alignments makes it impossible to perform a direct search. For example, the number of possible alignments of two sequences of length 1000 exceeds 10^{600} [Waterman 1995]. However, a dynamic programming algorithm makes it possible to find the optimal alignment without checking all possible alignments, but only a very small portion of the search space.

Calculating the global similarity score $S(\mathbf{a}, \mathbf{b})$

Denote by $S_{i,j}$ the score of the best alignment of the substring $a_1 a_2 .. a_i$ with the substring $b_1 b_2 .. b_j$, i.e.

$$S_{i,j} = S(a_1 a_2 .. a_i \ , \ b_1 b_2 .. b_j)$$

The optimal alignment of $a_1a_2..a_i$ with the substring $b_1b_2..b_j$ can end only in one of following three ways:

$$\begin{array}{ccc} a_i & a_i & - \\ b_j & - & b_j \end{array}$$

Also, each sub-alignment must be optimal as well. Therefore, the score $S(\mathbf{a}, \mathbf{b})$ can be calculated recursively:

Initialize

$$S_{0,0} = 0 \qquad S_{i,0} = -i \cdot \alpha \quad \text{for } i = 1..n \qquad S_{0,j} = -j \cdot \alpha \quad \text{for } j = 1..m$$

Recursively define

$$S_{i,j} = max\{S_{i-1,j-1} + s(a_i, b_j),\ S_{i,j-1} - \alpha,\ S_{i-1,j} - \alpha\}$$

In practice, the scores are stored in a two-dimensional array of size $(n + 1) \cdot (m + 1)$. The initialization set the values at row zero and column zero. The computation proceeds row by row so that the value of each matrix cell is calculated from entries which were already calculated. Specifically, we need the three matrix cells, on the west, the south and the southwest of the current cell. The time complexity of this algorithm is $\Theta(n \cdot m)$.

Dynamic programming algorithms were already introduced in the late 50's [Bellman 1957]. However, the first to propose a dynamic programming algorithm for **global** comparison of macromolecules, were Needleman and Wunsch [Needleman & Wunsch 1970].

2.1.2 Penalties for gaps

In sequence evolution, an insertion or deletion of a segment (several adjacent amino acids) usually occurs as a single event. That is, the opening of the gap is the significant event. Therefore, most computational models assign a penalty for a gap of length k that is smaller than the sum of k independent gaps of length 1. If the penalty for gap of length k is $\alpha(k)$, we are thus interested in sub-additive functions and assume that $\alpha(x + y) \leq \alpha(x) + \alpha(y)$.

Denote by N_{k-gap} the number of gaps of length k in a given alignment. Then the score of this alignment is defined in this case as

$$\sum_{i,j} N_{ij} \cdot s(a_i, b_j) - \sum_k N_{k-gap} \cdot \alpha(k)$$

This discussion allows the alignment to end in a gap of arbitrary length, and therefore $S_{i,j}$ is defined as follow:

Set

$$S_{0,0} = 0 \qquad S_{i,0} = -\alpha(i) \qquad S_{0,j} = -\alpha(j)$$

then

$$S_{i,j} = max\{S_{i-1,j-1} + s(a_i, b_j),\ max_{1 \leq k \leq j}\{S_{i,j-k} - \alpha(k)\},\ max_{1 \leq l \leq i}\{S_{i-l,j} - \alpha(l)\}\}$$

In practice, to compute each matrix cell, we need now to check the cell in the diagonal, all the cells in the same row and all the cells in the same column. For arbitrary function $\alpha(k)$, the time complexity is $\sum_{i,j}(i + j + 1) = \Theta(n^2 \cdot m + n \cdot m^2 + n \cdot m)$ plus $n + m$ calculations of the function $\alpha(k)$. For $n = m$ we get $\Theta(n^3)$.

Optimization

A better time complexity can be obtained for a linear gap function $\alpha(k) = \alpha_0 + \alpha_1 \cdot (k-1)$ where α_0 is the penalty for opening a gap, and α_1 is the penalty for extending the gap.

Set $E_{i,j} = max_{1 \le k \le j}\{S_{i,j-k} - \alpha(k)\}$. This is the maximum value over all the matrix cells in the same row, where a gap is tested with respect to each one of them. Note that

$$\begin{aligned}
E_{i,j} &= max\{S_{i,j-1} - \alpha(1),\ max_{2 \le k \le j}\{S_{i,j-k} - \alpha(k)\}\} \\
&= max\{S_{i,j-1} - \alpha_0,\ max_{1 \le k \le j-1}\{S_{i,j-(k+1)} - \alpha(k+1)\}\} \\
&= max\{S_{i,j-1} - \alpha_0,\ max_{1 \le k \le j-1}\{S_{i,(j-1)-k} - \alpha(k)\} + \alpha_1\} \\
&= max\{S_{i,j-1} - \alpha_0,\ E_{i,j-1} - \alpha_1\}
\end{aligned}$$

Therefore, only two matrix cells from the same row need to be checked. These cells correspond to the two options we need to check, i.e. opening a new gap at this position (first term) or extending a previously opened gap (second term). The sub-linearity of α implies that it is never beneficial to concatenate gaps.

Similarly, for a column define $F_{i,j} = max_{1 \le l \le j}\{S_{i-l,j} - \alpha(l)\}$ and in the same way obtain

$$F_{i,j} = max\{S_{i-1,j} - \alpha_0, F_{i-1,j} - \alpha_1\}$$

Initialize

$$E_{0,0} = F_{0,0} = S_{0,0} = 0$$
$$E_{i,0} = S_{i,0} = -\alpha(i) \qquad F_{0,j} = S_{0,j} = -\alpha(j)$$

then

$$S_{i,j} = max\{S_{i-1,j-1} + s(a_i, b_j),\ E_{i,j},\ F_{i,j}\}$$

and the time complexity is $\Theta(m \cdot n)$

For arbitrary convex gap functions it is possible to obtain an $O(n \cdot m)$ algorithm [Waterman 1995]. However, such functions seems to be inappropriate for the context of comparison of macromolecules. For arbitrary concave function (and in particular, for sub-additive functions) it is possible to obtain time complexity of $O(n^2 \cdot \log(n))$ with a more complex algorithm [Waterman 1995].

Gonnet et al. [Gonnet et al. 1992] have proposed a model for gaps that is based on gaps occurring in pairwise alignments of related proteins. The model suggests an exponentially decreasing gap penalty function (see section 2.4.4). However, a linear penalty function has the advantage of better time complexity, and in most cases the results are satisfactory. Therefore the use of linear gap functions is very common.

2.1.3 Global distance alignment

In some cases it is interesting to define a distance among sequences $D(\mathbf{a}, \mathbf{b})$ - for example for the construction of evolutionary trees, or when investigating the geometry of the sequence space. The advantage of defining a distance among sequences, is the construction of a metric space on the space of sequences.

In general a metric is a function $D:\ X \times X \rightarrow \mathcal{R}$ which is:

- Nonnegative: $D(a,b) \ge 0$ for all $a, b \in X$ with equality if and only if $a = b$
- Symmetric: $D(a,b) = D(b,a)$
- Satisfies the triangle inequality: $D(a,b) \le D(a,c) + D(c,b)\ \ \forall a, b, c \in X$

In contrast to similarity, where we are interested in the best alignment and its score gives a measure of how much the strings are alike, distances approach assigns a cost to elementary edit operations (evolutionary events) and seeks a series of operations that transforms one string to another, with the minimal cost.

The definition of distance resembles the definition of similarity, except that $s(a_i, b_j)$ is replaced with $d(a_i, b_j)$ which reflect the *distance* between amino acids a_i and b_j, and $\alpha(k)$, the gap penalty now adds to the total distance (instead of decreasing the similarity). The minimum distance is obtained by minimizing the sum of matches/mismatches costs and the penalties for gaps.

$$D(\mathbf{a}, \mathbf{b}) = min_{alignments}\{\sum_{i,j} N_{ij} \cdot d(a_i, b_j) + \sum_k N_{k-gap} \cdot \alpha(k)\}$$

One restriction is that $d(\cdot, \cdot)$ is a metric on $\mathcal{A}$, the alphabet of amino acids (and so satisfies the three requirements above). Also $\alpha(k)$ has to be positive. These restrictions imply that the total global distance $D(\cdot, \cdot)$ is zero only if the two sequences are identical.

In some cases, distance and similarity measures are related by a simple formula: Let $s(a_i, b_j)$ be a similarity measure over $\mathcal{A}$ and $\alpha(k)$ the penalty for gap of length k. Let $d(a_i, b_j)$ a metric on $\mathcal{A}$ and $\hat{\alpha}(k)$ a corresponding cost for gaps of length k. If there is a constant c such that

$$d(a_i, b_j) = c - s(a_i, b_j) \quad \forall a_i, b_j \in \mathcal{A}$$

and

$$\hat{\alpha}(k) = \alpha(k) + \frac{kc}{2}$$

then each alignment A_l satisfies:

$$D(A_l) = \frac{c(n+m)}{2} - S(A_l)$$

where n (m) is the length of the sequence $\mathbf{a}$ $(\mathbf{b})$. In particular,

$$\begin{aligned} D(\mathbf{a}, \mathbf{b}) &= min_{alignments\ A_l} D(A_l) \\ &= \frac{c(n+m)}{2} - max_{alignments\ A_l} S(A_l) \\ &= \frac{c(n+m)}{2} - S(\mathbf{a}, \mathbf{b}) \end{aligned}$$

i.e. an alignment is similarity optimal if and only if it is distance optimal (the proof of this claim is based on the observation that $n + m = 2 \cdot \sum_{i,j} N_{ij} + \sum_k N_{k-gap}$).

Though the formula suggests a simple transformation from a similarity measure to a distance measure, it should be noted that the transformation from $s(a_i, b_j)$ to $d(a_i, b_j)$ does not yield a metric when applied to the common scoring schemes, since the value $s(a_i, a_i)$ varies among different amino acids (see section 2.4). Consequently, there is no c such that $d(a_i, a_i) = c - s(a_i, a_i) = 0$ for all $a_i \in \mathcal{A}$, as needed.

2.1.4 Position dependent scores

In many proteins, mutations are not equally probable along the sequence. Some regions are functionally/structurally important and consequently, the effect of mutation in these regions can be drastic. They may create a nonfunctional protein or even prevent the molecule from folding into its native structure. Such mutations are unlikely to survive, and therefore these regions tend to be more evolutionary conserved than other, less constrained regions (e.g. loops) which can significantly diverge.

Accordingly, it may be appropriate to use position-dependent scores for mismatches and gaps. The incorporation of information about structural preferences can lead to alignments that are more accurate biologically. If a protein's structure is known, the secondary structure should be taken into account. In the absence of such data, general structural criteria, such as the propensities of amino acid for occurring in secondary structures versus loops can be taken into account. For example, the probability of opening a gap in existing secondary structure can be decreased, while the probability for opening/inserting a gap in loop regions can be increased.

The dynamic programming algorithm can be adapted to account for position-specific scores as follow: Let $s_{i,j}(a,b)$ be the score for aligning a and b in the i-th position of $\mathbf{a} = a_1 a_2 ... a_n$ and j-th position of $\mathbf{b} = b_1 b_2 ... b_m$, respectively, and let α_i (γ_j) be the penalty for opening a gap at position i (j) and β_i (δ_j) for extending it. Recursion is used as before to define the similarity of $\mathbf{a}$ and $\mathbf{b}$:

$$E_{i,j} = max\{S_{i,j-1} - \gamma_j,\ E_{i,j-1} - \delta_j\}$$

$$F_{i,j} = max\{S_{i-1,j} - \alpha_i,\ F_{i-1,j} - \beta_i\}$$

and

$$S_{i,j} = max\{S_{i-1,j-1} + s_{i,j}(a_i, b_j),\ E_{i,j},\ F_{i,j}\}$$

Usually position-specific scoring matrices, or **profiles**, are not tailored to a specific sequence. Rather, they are built to utilize the information in a group of related sequences, and provide representations of protein families and domains. These representations are capable of detecting subtle similarities between distantly related proteins. Without going into detail, profiles are usually obtained by applying algorithms for multiple alignment (i.e., a combined alignment of several proteins) to align a group of related sequences. The frequency of each amino acid at each position along the multiple alignment is then calculated. These counts are normalized and transformed to probabilities, so that a probability distribution over amino acids is associated with each position. Finally, the scoring matrix is defined based on these probability distributions as well as on the similarities of pairs of amino acids (taken from a standard scoring matrix). For example, the score for aligning the amino acid a at position i of the profile is given by $s_i(a) = \sum_{b \in \mathcal{A}} prob(b \text{ at position } i) s(a,b)$ (a few minor modifications are needed to formulate the above algorithm for comparison of a sequence with a profile). For a review of algorithms for multiple alignment and profile techniques see [Waterman 1995, Setubal & Meidanis 1996, Gribskov & Veretnik 1996, Taylor 1996].

2.1.5 Local alignments

In many cases the similarity of two sequences is limited to a specific motif or domain, the detection of which may yield valuable structural and functional insights, while outside of this motif/domain the sequences may be essentially unrelated. In such cases global alignment may not be the appropriate tool. In the search for an optimal global alignment, local similarities may be masked by long unrelated regions. Consequently, the score of such an alignment can be as low as for totally unrelated sequences. Moreover, the algorithm may even misalign the common region. A minor modification of the previous algorithm solves this problem.

First we define a **local alignment** of **a** and **b** as an alignment between a substring of **a** and a substring of **b**. The **local similarity** of sequences **a** and **b** is defined as the maximal score over all possible local alignments.

Calculating the local similarity score

Define $H_{i,j}$ to be the maximum similarity of two segments ending at a_i and b_j

$$H_{i,j} = max\{0,\ S(a_x a_{x+1}...a_i,\ b_y b_{y+1}...b_j) :\ \ 1 \le x \le i,\ \ 1 \le y \le j\}$$

This quantity is calculated recursively as before. Initialize

$$H_{i,0} = H_{0,j} = 0 \quad 1 \le i \le n,\ \ 1 \le j \le m$$

then

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

and the local similarity of sequences **a** and **b** is obtained by maximizing over all possible segments

$$H(\mathbf{a}, \mathbf{b}) = max\{H_{k,l} \quad 1 \le k \le n,\ \ 1 \le l \le m\}$$

For linear gap functions (that improve computation time) the formulation is done as before, with initialization $H_{i,j} = E_{i,j} = F_{i,j} = 0$ for $i \cdot j = 0$.

In the literature, this algorithm is often called the Smith-Waterman (SW) algorithm, after those who introduced this modification [Smith & Waterman 1981].

It should be noted that whereas global similarity measures can sometime be transformed into global distance measures (as in section 2.1.3), no such transformation is known for local similarity measure. A case which seems to rule out the possibility of defining a local distance measure is depicted in Fig. 2.1. However, it is possible to define pseudo-metrics on protein sequences based on local similarity measures. For a brief discussion see chapter 4.

2.2 Other algorithms for sequence comparison

During the last two decades the sequencing techniques have greatly improved. Many large scale sequencing projects of various organisms are carried throughout the world, and as a result the

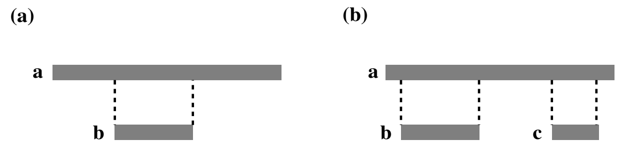

Figure 2.1: **Local similarities make it difficult to define a distance between protein sequences**. Two cases in mind are: (a) If **b** is a subsequence of **a**, the sequences are obviously related. However, a distance measure should account for those parts of the sequence **a** which are not matched with **b**. Consequently, the distance $D(\mathbf{a}, \mathbf{b})$ may be as high as for totally unrelated sequences. (b) Multi domain proteins. If **b** and **c** are unrelated sequences (i.e., $D(\mathbf{b}, \mathbf{c}) >> 1$), then assigning a low distance for $D(\mathbf{a}, \mathbf{b})$ and $D(\mathbf{a}, \mathbf{c})$ will violate the triangle inequality.

number of new sequences which are stored in the databases is rapidly increasing. In a typical application new protein sequence is compared with all sequences in the database, in search of related proteins.

The dynamic programming algorithms described above that seek optimal pairwise alignment of sequences (with insertions/deletions and substitutions) may not be suitable for this purpose. The complexity of this algorithm is quadratic (with linear gap function), and the comparison of a sequence, of average length of 350 amino acids, against a typical database (like SWISSPROT [Bairoch & Apweiler 1999], with more than 70,000 sequences), may take few CPU hours on a standard PC of nowadays (pentium-II 400 MHz).

Several algorithms have been developed to speed up the search. The two main algorithms are FASTA [Pearson & Lipman 1988] and BLAST [Altschul et al. 1990]. These are heuristic algorithms which are not guaranteed to find the optimal alignment. However, they proved to be very effective for sequence comparison, and they are significantly faster than the rigorous dynamic programming algorithm.

In the last few years, biotechnology companies such as Compugen and Paracel, have developed special purpose hardware that accelerates the dynamic programming algorithm [Compugen 1998]. This special-purpose hardware has again made the dynamic programming algorithm competitive with FASTA and BLAST, both in speed and in simplicity of use. However, meanwhile, FASTA and BLAST have become standard in this field and are being used extensively by biologists all over the world. Both algorithms are fast, effective, and do not require the purchase of additional hardware. BLAST has an additional advantage, as it may reveal similarities which are missed by the dynamic programming algorithm, for example when two similar regions are separated by a long dissimilar region.

2.2.1 BLAST (Basic Local Alignment Search Tool)

BLAST compares two sequences and seeks all pairs of similar segments, whose similarity score exceeds a certain threshold. These pairs of segments are called "high scoring segment pairs" (HSPs). A segment is always a contiguous subsequence of one of the two sequences. Segment pair is a pair of segments of the same length, one from each sequence. Hence the alignment of the segments is without gaps. The score of the match is simply the sum of matches of the amino acids (defined by a scoring matrix) along the segment pair. The segment pair with the highest score is called the "maximum segment pair" (MSP)[1].

The algorithm is an outgrowth of the statistical theory for local alignments without gaps (see section 2.3). This theory gives a framework for assessing the probability that a given similarity between two protein sequences (i.e. the MSP) could have emerged by chance. If the probability is very low, then the similarity is statistically significant and the algorithm reports the similarity along with its statistical significance.

It should be kept in mind that BLAST's similarity score is only an approximate measure for the similarity of the two sequences, since gaps are ruled out. The algorithm may indeed miss complex similarities which include gaps. However, the statistical theory of alignments without gaps provided a reliable and efficient way of distinguishing true homologies from chance similarities, thus making this algorithm an important tool for molecular biologists.

Implementing BLAST

The algorithm locates "seeds" of similarity among the query sequence and the database sequence, and then extends them. The algorithm operates in three main steps:

- Compile a list of words of length w (usually $w = 3$ or 4 for protein sequences) that score at least T with some substring of length w of the query sequence.

- Scan database sequences in search for hits with words in the list from the first step. Each hit is a seed. To allow a fast scanning of the database two approaches are used to store the list. The first is a hash table, and the second employs a deterministic finite automaton. Both methods scan each library sequence only once.

- Extend seeds: each seed is extended in both directions, without gaps, until the maximum possible score for the extension is reached. The resulting HSP is record. If the score of the extension falls below a certain threshold then the process stops. Therefore, there is a chance (usually small, depending on the threshold) for the algorithm to miss a possible good extension.

- Attempt to combine multiple MSP regions. For each consistent combination, calculate the probability of this combination using the Poisson or sum statistics [Altschul et al. 1994] and report the most significant one (lowest probability).

  In the latest version of BLAST the criterion for extending seeds has been modified, to save processing time [Altschul et al. 1997]. The new version requires the existence of two non-overlapping seeds on the same diagonal (i.e. the seeds are at the same distance apart in both sequences), and within a certain distance (typically 40) of one another, before an extension is invoked. To achieve comparable sensitivity, the threshold T is lowered, yielding more hits

[1] It is possible to find the MSP with the dynamic programming algorithm, by setting the gap penalty to ∞. However, BLAST finds it much faster because of its efficient implementation.

than previously. However, only a small fraction of these hits are extended, and the overall speed increases.

Changes in the threshold T permit a tradeoff between speed and sensitivity. A higher value of T yields greater speed, but also an increased probability of missing weak similarities. Though there are no analytic bounds on the time complexity of this algorithm, in practice, the run time is proportional to the product of the lengths of the query sequence and the database size.

Current improvements of BLAST allow gapped alignments, by using dynamic programming to extend a central seed in both directions [Altschul et al. 1997]. This is complemented by PSI-BLAST, an iterative version of BLAST, with a position-specific score matrix (see section 2.1.4) that is generated from significant alignments found in round i and used in round $i+1$. The latter may better detect weak similarities that are missed in database searches with a simple sequence query.

2.2.2 FASTA

FASTA is another heuristic that performs a fast sequence comparison. The algorithm starts by creating a hash table of all k-tuples in the query sequence (usually, k = 1 or 2 for protein sequences, where k=1 gives higher sensitivity). For each such k-tuple there is an index vector with all positions of the k-tuple in the query sequence.

Then, when scanning a library sequence, a vector indexed with offsets (see below) is initialized with zeros. Each k-tuple of the library sequence is looked up in the hash table. If the k-tuple appears in the hash-table then per each appearance of this k-tuple in the query sequence, the offset (the relative displacement of the k-tuple) is calculated (if the k-tuple appears in position i in the query sequence and position j in the library sequence then the offset is $i-j$), and the offset vector is incremented at the index corresponding to the offset value. After the library sequence has been scanned, the diagonal which corresponds to the offset with the maximal number of occurrences (highest density of identities) can serve as a seed for the dynamic programming algorithm. In practice the algorithm proceeds as follow.

At a second stage, the ten regions with the highest density of identities are rescanned. Common k-tuples which are on the same diagonal (same offset) and are sufficiently close to an existing run are added to the run (the exact parameters are set heuristically) to form a region (a gapless local alignment, or HSP in BLAST terminology). The regions are scored to account for the matches as well as the mismatches, and the best region is reported (its score is termed "initial score" or "init1"). Then, the algorithm tries to join nearby high scoring regions, even if they are not on the same diagonal (the corresponding score being termed "initn score"). Finally, a bounded dynamic programming is run in a band around the best region, to obtained the "optimized score". If the sequences are related then the optimized score is usually much higher than the initial score.

2.3 Probability and statistics of sequence alignments

In the evolution of protein sequences, not all regions mutate at the same rate. Regions which are essential for the structure and function of proteins, are more conserved. Therefore, significant sequence similarity of two proteins suggests a common evolutionary origin and possibly similar structures. In many cases it reflects close biological functions.

The algorithms that were described in the previous sections can be used to identify such similarities. However, on any two input protein sequences, even if totally unrelated, the algorithms always find some similarity. For unrelated sequences this similarity is essentially random. As the length of the sequences compared increase, this random similarity may increase as well. Therefore, in order to assess the significance of a similarity score it is important to know what score to expect simply by chance.

Naturally, we would like to identify those similarities which are genuine, and biologically meaningful. In the view of the last paragraph, the raw similarity score may not be appropriate for this purpose. However, when the sequence similarity is statistically significant we can deduce, with high confidence level, that the sequences are related[2]. The reverse implication is not always true. We encounter many examples of low sequence similarity despite structural and functional similarity.

Though statistically significant similarity is neither necessary nor sufficient for a biological relationship, it may give us a good indication of such relationship. When comparing a new sequence against the database, in search of close relatives, this is extremely important, as we are interested in reporting only significant hits, and sorting the results according to statistical significance seems reasonable.

To estimate the statistical significance of similarity scores, a statistical theory should be developed. A great effort was made in the last two decades to establish such statistical theory. Currently, there is no complete theory, though some important results were obtained. These results have very practical implications and are very useful for estimating the statistical significance of similarity scores.

The statistical significance of similarity scores for "real" sequences is defined by the probability that the same score would have been obtained for random sequences. The statistical results concern the similarity scores of random sequences, when the similarity scores are defined by ungapped alignments. However, these results have created a framework for assessing the statistical significance of various similarity scores, including gapped sequence alignments, and recently, even structural alignments [Levitt & Gerstein 1998].

2.3.1 Basic random model

In the basic random model, the sequences are random sequences of characters where the characters are drawn independently and identically (i.i.d) from a certain distribution over the alphabet $\mathcal{A}$.

[2]Two exceptions are segments with unusual amino acid composition, and similarity that is due to convergent evolution.

Each sequence is thus viewed as a sequence of i.i.d random variables drawn from the distribution $\mathcal{P}$ over the alphabet $\mathcal{A}$. In what follows, upper case letters (A_i) denote random variables and lower case letters ($a_i,\ \ a_i \in \mathcal{A}$) indicate a specific value of the random variable. Upper case bold letters denote sequences of random variables, and lower case bold letters denote sequences of amino acids.

For two random sequences $\mathbf{A}$ and $\mathbf{B}$, the scores $S(\mathbf{A},\mathbf{B})$ (the global similarity score) and $H(\mathbf{A},\mathbf{B})$ (the local similarity score) are random variables. The distributions of these scores for randomly drawn sequences differ, and the next two sections summarize the main properties known about these distributions.

2.3.2 Statistics of global alignment

Fixed alignment - global alignment without gaps

This is a very simplistic case, but the results are straight forward. Let $\mathbf{A} = A_1A_2...A_n$ and $\mathbf{B} = B_1B_2...B_n$ where A_i, b_j are i.i.d random variables as defined above. For the alignment with no gaps,

$$\begin{array}{rcllcccl} \mathbf{A} & = & A_1 & A_2 & . & . & . & A_n \\ \mathbf{B} & = & B_1 & B_2 & . & . & . & B_n \end{array}$$

the score is simply defined as $S = \sum_{i=1}^{n} s(A_i, B_i)$.

S is the sum of i.i.d random variables and therefore for large n it is distributed approximately normally with expectation $E(S) = n \cdot E(s(A,B)) = n\mu$ and $Var(S) = n \cdot Var(s(A,B)) = n\sigma^2$, where μ and σ are the mean and standard deviation of the scoring matrix $s(a,b)$.

The main characteristic of this distribution is the linear growth (or decline, depends on the mean of the scoring matrix) with the sequence length. Surprisingly, perhaps, this characteristic holds for the general case as well.

Unknown alignment

In the general case, gaps are allowed in the alignment, and the similarity score is defined as the maximum over all possible alignments,

$$S = S(\mathbf{A},\mathbf{B}) = max_{alignments}\{\sum_{i,j} N_{ij} \cdot s(a_i, b_j) - \sum_k N_{k-gap} \cdot \alpha(k)\} \tag{2.1}$$

where $\alpha(k)$ (the penalty for a gap of length k) is a general non-negative sub additive function, i.e. $\alpha(x+y) \leq \alpha(x) + \alpha(y)$.

The normal distribution limit law no longer holds because of the optimization over all possible alignments. However, it is possible to partly characterize the (limit at large n) distribution of S based on Kingman's theory and the Azuma-Hoeffding lemma [Waterman 1995]. Kingman's theory is applied to show that the global similarity score grows **linearly** with the sequences length.

Theorem 1: Let $\mathbf{A} = A_1A_2...A_n$ and $\mathbf{B} = B_1B_2...B_n$ where A_i, B_j are i.i.d random variables drawn from the distribution $\mathcal{P}$ over the alphabet $\mathcal{A}$. Define $S = S(\mathbf{A},\mathbf{B})$ as in equation 2.1, and under the same restrictions on the function $\alpha(k)$, then there is a constant $\rho \geq E(s(A,B))$ such that

$$\lim_{n\to\infty} \frac{S}{n} = \rho$$

with probability 1 and in the mean.

The last result obtained for an infinitely long sequence. However, since it holds almost surely, than the average over a large ensemble of finite sequences satisfies

$$\frac{E(S)}{n} \to \rho$$

Theorem 1 defines the expected global similarity score for random sequences. The statistical significance of a similarity score obtained for "**real**" sequences, which exceeds the expected score by a certain amount, is estimated by the probability that the similarity score of **random** sequences would exceed the expected mean by the same amount. The Azuma-Hoeffding lemma gives a bound on the probability that such a random variable exceeds its mean by a certain amount (it provides a concentration of measure result for a broad class of random variables which includes this case).

Theorem 2: Let $\mathbf{A} = A_1A_2...A_n$ and $\mathbf{B} = B_1B_2...B_n$ where A_i, B_j are i.i.d and define $S = S(\mathbf{A},\mathbf{B})$ as in equation 2.1, then

$$Prob(S - E(S) \geq \gamma \cdot n) \ \leq \ e^{-\frac{\gamma^2 n}{2c^2}}$$

where c is a constant that depends only on the scoring matrix and the penalty for a gap.

Though the linear growth of the global similarity score is an important feature, both results are theoretical and have little practical use. In spite of much effort, ρ has not yet been determined. Therefore, theorem 1 remains only a qualitative result. Moreover, the bound obtained by the Azuma-Hoeffding lemma is not very useful for a typical protein where n is of the order of 300. For example, for a typical scoring matrix such as the BLOSUM 62 matrix (see section 2.4.2), and a gap opening penalty of 12, the constant c in theorem 2 equals 30. If a global similarity score exceeds the expected mean by $3 \cdot n$ (which, for a global similarity score is usually significant), then the corresponding bound would be $Prob(S - E(S) \geq 3 \cdot n) \ \leq \ 0.223$, which is not very significant (suggesting, perhaps, that the bound is not tight). The variance of the global similarity score has not been determined either, and the best results give only an upper bound.

In practice, it is possible to empirically approximate the distribution by shuffling the sequences and comparing the shuffled sequences. By repeating this procedure many times it is possible to estimate the mean and the variance of the distribution, and a reasonable measure of statistical significance (e.g. by means of the z-score) can be obtained[3].

[3]Denote by S the global similarity score. Let μ and σ^2 be the mean and the variance of the distribution of scores. Then, the z-score associated with the score S is defined as $\frac{S-\mu}{\sigma}$. This score measures how many units of standard deviation apart the score S is from the mean of the distribution. The larger it is, the more significant is the score S.

2.3.3 Statistics of local alignment

Exact matching

As for global alignments, interesting results for local alignment are obtained already under a very simplistic model. Specifically, it is interesting to study the asymptotic behavior of the longest perfect match between two random sequences of length n, when the alignment is given and fixed. Let $\lambda = Prob(Match) = \sum_{a\in\mathcal{A}} p_a^2$ where p_a is the probability of the letter a. The problem can be rephrased in terms of the length R_n of the longest run of heads in n coin tosses, when the probability for head is λ.

According to Erdös and Renyi [Erdös & Renyi 1970] $R_n \rightarrow \log_{1/\lambda} n$. The intuition is that the probability of run of m heads is λ^m, and for $m << n$ there are approximately n possible runs (one for each possible starting point). Therefore,

$$E(\text{number of runs of length m}) \simeq n \cdot \lambda^m$$

If the longest run is unique then R_n should satisfy $1 = n \cdot \lambda^{R_n}$

$$\Rightarrow R_n = \log_{1/\lambda} n$$

The proof is now concluded, using Borel-Cantelli lemma. It is given in details in [Waterman 1995].

This result holds for exact matches, which start at the same position in both sequences. Allowing shifts makes the problem more interesting to molecular biology. The length of the longest match in this case is actually the score of the best local alignment $\mathcal{S} = H(\mathbf{A}, \mathbf{B})$ given that the score for a match is 1, it is $-\infty$ for a mismatch, and ∞ for opening a gap.

When shifts are allowed, there are n^2 possible starting positions (i, j) for the match (the number of possible starting positions of a match defines the size of the **search space**). Therefore, following the intuition of the previous claim we expect that

$$\mathcal{S} \rightarrow \log_{1/\lambda} n^2 = 2 \cdot \log_{1/\lambda} n$$

and the logarithmic characteristic is preserved (for a proof see [Waterman 1995]).

The same result is obtained when postulating that the sequences are generated by first order Markov chains, and even when the the two sequences are drawn from different distributions (this makes more sense, biologically). However, these results hold for perfect matches, while the original problem allows mismatches as well. Surprisingly, the results holds even when mismatches are allowed (see next section).

Matching with score - local alignment without gaps

So far, the score for a match was 1. This section proceeds to the case of a general scoring scheme $s(a, b)$ for $a, b \in \mathcal{A}$, with the constraints *(i)* $E(s(a, b)) < 0$ and *(ii)* $s^* = max\{s(a, b)\} > 0$. The first requirement implies that the average score of a random match will be negative (otherwise, extending a match would tend to increase the score, and this contradicts the idea of local similarity). The second condition implies that a match with a positive score is possible (otherwise a match would always consist of a single pair of residues).

The following theorem concerns local matches (as defined in section 2.1.5) with a general scoring scheme but **without gaps** (i.e., the penalty for opening a gap is ∞). It characterizes the maximal score of local alignment $\mathcal{S} = H(\mathbf{A}, \mathbf{B})$ of two random sequences, and the distribution of amino acids in the maximal scoring segments.

Theorem 3: Let $\mathbf{A} = A_1 A_2 ... A_n$ and $\mathbf{B} = B_1 B_2 ... B_n$ where A_i, B_j are i.i.d and sampled from the same distribution $\mathcal{P}$ over an alphabet $\mathcal{A}$. Assume that $s(a,b)$ satisfies $E(s(A,B)) < 0$ and $s^* = max\{s(a,b) : \ a,b \in \mathcal{A}\} > 0$. Let $\lambda > 0$ be the largest real root of the equation $E(e^{\lambda \cdot s(A,B)}) = 1$. Then

$$\lim_{n \to \infty} \frac{\mathcal{S}}{\frac{\ln n^2}{\lambda}} = 1 \tag{2.2}$$

and the proportion of letter a aligned with the letter b in the best scoring match converges to $q_{a,b} = p_a p_b e^{\lambda \cdot s(a,b)}$

The direct implication of this theorem is that for two random sequences of length n and m, the score of the best local alignment (the MSP score in BLAST jargon) is centered around $\frac{\ln(n \cdot m)}{\lambda}$. That is, the local similarity score grows **logarithmically** with the length of the sequences, and with the size of the search space $n \cdot m$. Practically, given the distribution $\mathcal{P}$ (for example the overall **background distribution** of amino acids in the database), λ is obtained by solving the equation

$$\sum_{a,b} p_a p_b e^{\lambda \cdot s(a,b)} = 1 \tag{2.3}$$

using a method such as Newton's method.

This result in itself is still not enough to obtain a measure of statistical significance for local similarity scores. This can be done only once a concentration of measure result is obtained or the distribution of similarity scores is defined. Indeed, one of the most important results in this field is the characterization of the distribution of local similarity scores without gaps. This distribution was shown to follow the extreme value distribution [Karlin & Altschul 1990, Dembo & Karlin 1991, Dembo et al. 1994b].

Formally, as the **sum** of many i.i.d random variables is distributed **normally**, then the **maximum** of many i.i.d random variables is distributed as an **extreme value distribution** [Gumbel 1958]. This distribution is characterized by two parameters: the index value u and the decay constant λ (for $u = 0$ and $\lambda = 1$, the distribution is plotted in Fig. 2.2). The distribution is not symmetric. It is positive definite and unimodal with one peak at u. Practically, the score of the best local alignment (the MSP score) is the maximum of the scores of many independent alignments, which explains the observed distribution. This is summarized in the next theorem, which concerns the distribution of local similarity scores for random sequences.

Theorem 4: Let $\mathcal{S}$ be a random variable whose value is the local similarity score for two random sequences of length n and m, as defined above. $\mathcal{S}$ is distributed as an extreme value distribution and

$$Prob(\mathcal{S} \geq x) \sim 1 - \exp(-e^{-\lambda \cdot (x-u)}) \tag{2.4}$$

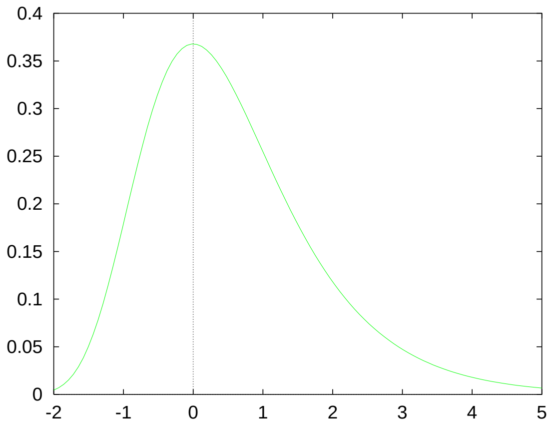

Figure 2.2: **Probability density function for the extreme value distribution with $u = 0$ and $\lambda = 1$.**

where λ is the root of equation 2.3 and $u = \frac{\ln Kmn}{\lambda}$, where K is a constant that can be estimated from first-order statistics of the sequences (i.e., the background distribution) and the scoring matrix s (K is given by a geometrically convergent series which depends only on the p_a and s_{ab} [Karlin & Altschul 1990]).

Theorem 4 holds, subject to the restriction that the amino acid composition of the two sequences that are compared are not too dissimilar [Karlin & Altschul 1990]. Assuming that both sequences are drawn from the background distribution, the amino acid composition of both should resemble the background distribution. Without this restriction theorem 4 overestimates the probability of similarity scores, and indeed, this is observed in protein sequences with unusual compositions (see also chapter 8).

For a large x we can use the approximation $1 - \exp(-e^{-x}) \sim e^{-x}$. Therefore, for a large x,

$$Prob(\mathcal{S} \geq x) \sim e^{-\lambda(x-u)} = e^{-\lambda \cdot x} e^{\lambda \cdot u} = Kmne^{-\lambda \cdot x} \tag{2.5}$$

This result helps to calculate the probability that a given MSP score could have been obtained by chance. The score will be statistically significant at the 1% level if $\mathcal{S} \geq x_0$ where x_0 is determined by the equation $Kmne^{-\lambda \cdot x_0} = 0.01$. In general, a pairwise alignment with score $\mathcal{S}$ has a **p-value** of p where $p = Kmne^{-\lambda \cdot \mathcal{S}}$. I.e., there is a probability p that this score could have happened by chance.

The probability p, that a similarity score $\mathcal{S}$ could have been obtained simply by chance from the comparison of two random sequences, should be adjusted when multiple comparisons are performed. One example of this is when a sequence is compared with each of the sequences in a database of size D. Denote by **p-match** a match between two sequences that has a p-value $\leq p$ (i.e., its score $\geq \mathcal{S}$). The probability of observing at least one p-match (i.e., at least one "success"), in a database search is distributed as a Poisson distribution (the distribution is in fact binomial, but in practice, the Poisson $\delta = Dp$ approximation is applied for the Binomial(D, p) distribution for D large and $\delta = Dp$ small). The probability of m successes is $Pr(m) \simeq \frac{\delta^m}{m!} e^{-\delta}$. Therefore, the probability P of getting by chance at least one p-match, when searching in a database containing D sequences is

$$P = Prob(m \geq 1) = 1 - Prob(m = 0) = 1 - e^{-\delta} = 1 - e^{-Dp} \tag{2.6}$$

and for $P < 0.1$ this is well approximated by Dp. Thus,

$$P \simeq 1 - e^{-Dp} \simeq Dp = KDmne^{-\lambda \cdot \mathcal{S}} \tag{2.7}$$

This discussion assumes that all sequences in the database (**library sequences**) have the same probability of sharing a similar region with the **query sequence**. However, it is more appropriate to assume that all equal-length protein segments in the database have an equal *apriori* probability that they are related to the query sequence (since similarity usually follows a domain). Therefore, if the query sequence is of length n, and the (pairwise) alignment of interest involves a library segment of length m, and the database has a total of N amino acids, then D should be replaced with N/m. Hence, equation 2.7 is corrected to obtain

$$P \simeq KNne^{-\lambda \cdot \mathcal{S}} \tag{2.8}$$

where the term Nn can be viewed as the "effective size" of the search space.

It is very common to use the **expectation value** (e-value) as a measure of statistical significance. It is well known that the parameter δ of the Poisson distribution is also the expectation value of this distribution (as well as of the original binomial distribution which is approximated by the Poisson distribution). Therefore,

$$E = E(\textsf{number of } p\textsf{-matches}) = \delta = Dp \tag{2.9}$$

and as discussed above, D should be replaced with N/m. Hence

$$E = KNne^{-\lambda \cdot \mathcal{S}} \tag{2.10}$$

This is the expected number of distinct matches (segment pairs) that would obtain a score $\geq \mathcal{S}$ by chance in a database search, with a database of size N (amino acids) and composition $\mathcal{P}$ (the background distribution of amino acids). If $E = 0.01$, then the expected number of random hits with a score $\geq \mathcal{S}$ is 0.01. In other words, we may expect a random hit once in 100 independent searches. If $E = 10$, then we should expect 10 hits with a score $\geq \mathcal{S}$ by chance, in a single database search. This means that such a hit is not significant. Note that $E \simeq P$ for $P < 0.1$ (equations 2.8 and 2.10), and consequently, in many publications in this field there is no clear distinction between E and P. However, for $P > 0.1$ they differ, where $P = 1 - exp(-E)$ (equations 2.6 and 2.9).

Finally, by setting a value for E and solving the equation above for $\mathcal{S}$, it is possible to define a threshold score, above which hits are reported. This is the score above which the number of hits that are expected to occur at random is $< E$. Therefore, we can deduce that a match with this score or above reflects true biological relationship, but we should expect up to E errors per search. The specific value of E affects both the sensitivity of a search (the number of true relationships detected) and its selectivity (the number of errors). A lower value of E would decrease the error rate. However, it would decrease the sensitivity as well. A reasonable choice for E is between 0.1 and 0.001 (see chapter 8 for more details).

In general, the same two sequences may have more than one high-scoring pair of segments. This may happen whenever insertions/deletions should be introduced to align the sequences properly. The combined assessment of scores from several ungapped alignments can be evaluated by applying Poisson statistics with the parameter $p = e^{-\lambda(x-u)}$. Given the k highest scoring HSPs, among which the lowest HSP score is x, the Poisson distribution can be used to calculate the probability that at least k segments pairs, all with a score of at least x, would appear by chance in one pairwise comparison. This approach has the disadvantage of being dependent on the lowest score among the k highest scores. Another alternative is to calculate the sum S_k of the highest k scores. The distribution of such sums has been derived [Karlin & Altschul 1993] and the probability of a given sum is calculated (numerically) by a double integral on the tail of the distribution. In either case, the HSPs should first satisfy a consistency test before the joint assessment is made.

Local alignment with gaps

Though local alignments without gaps may detect most similarities between related proteins, and give a good estimation of the similarity of the two sequences, it is clear that gaps in local alignments are crucial in order to obtain the correct alignment, and for a more accurate measure of similarity. However, no precise model has been proposed yet to explain gaps in alignments. Moreover, introducing gaps in alignments greatly complicates their mathematical tractability. Rigorous results have been obtained only for local alignments without gaps.

Recent studies suggest that the score of local gapped alignments can be characterized in the same manner as the score of local ungapped alignments: According to theorem 3 the local **ungapped** similarity score grows logarithmically with the sequence's length and the size of the search space. Arratia & Waterman [1994] have shown that for a range of substitution matrices and gap penalties, local **gapped** similarity scores have the same asymptotic characteristic. Furthermore, empirical studies [Smith et al. 1985, Waterman & Vingron 1994] strongly suggest that local gapped similarity scores are distributed according to the extreme value distribution, though some correction factors may apply [Altschul & Gish 1996].

Based on empirical observations, Pearson [1998] has derived statistical estimates for local alignment with gaps, using the extreme value distribution for scores obtained from a database search. A database search provides tens of thousands of scores from sequences which are unrelated to the query sequence, and therefore are effectively random. As discussed above, these scores are thus expected to follow the extreme-value distribution. This is true as long as the gap penalties are not too low. Otherwise the alignments shift from local to global and the extreme value distribution no longer apply.

Since the logarithmic growth in the sequence length holds in this case, scores are corrected first for the expected effect of sequence length. The correction is done by calculating the regression line $S = a + b \cdot \ln n$ for the scores obtained in a database search, after removing very high scoring sequences (probably related sequences). The process is repeated as many as five times. The regression line and the average variance of the normalized scores are used to define the **z-score**:

$$z\text{−score} = \frac{S - (a + b \cdot \ln n)}{var}$$

and the distribution of z-scores is approximated by the extreme value distribution

$$p = Prob(z\text{−score} > x) = 1 - \exp(-e^{c_1 \cdot x - c_2})$$

where c_1 and c_2 are constants, and the expectation value is defined as before by $E(z\text{−score} > x) = N \cdot p$ where N is the number of sequences in the database (the number of tests).

This empirical approach has the advantage of internal calibration of the accuracy of the estimates, and has proved to be very accurate in estimating the statistical significance of gapped similarity scores [Pearson 1998] (see also chapter 8 and [Brenner et al. 1998]).

2.4 Scoring matrices and gap penalties

Protein sequences are compared using scoring matrices that are not just +1 for a match and 0 for a mismatch. Since the late 60s, several different approaches were taken to derive reliable and effective scoring matrices. I'll briefly discuss some of this work in this section. For more details, and comparison of different scoring methods see [Feng et al. 1985, Johnson & Overington 1993].

The genetic code matrix [Fitch & Margoliash 1967] measures amino acid similarity by the number of common nucleotide bases in their codons. This quantity is maximized by considering the closest matching representatives codons. Identical amino acids share 3 bases, while non-identical share 2 or less. Despite this nice rational which returns to the very basic evolutionary events at the DNA level, most of the genetic pressure is on the protein sequence. Although there is some correlation between the codons of different amino acids and their biochemical properties[4], this correlation is not strong enough to detect weak homologies.

Some matrices are based directly on similarities between physico-chemical properties of amino acids [Grantham 1974, Miyata et al. 1979]. However, these matrices do not perform very well. The reason is that there is no single property of amino acids which can account for preserving the structure and function of a protein.

The most effective matrices are those that are based on actual frequencies of mutations that are observed in closely related proteins. Exchange occur more frequently among amino acids that share certain properties. Indeed, these matrices reflect the biochemical properties of the amino acids, which influence the probability of mutual substitution, and amino acids with similar properties have high pairwise score. Matrices which are based on sequence alignments include the family of PAM matrices [Dayhoff et al. 1978] (and their improvement by [Jones et al. 1992]), the

[4][Gonnet et al. 1992] have shown that for low divergence, the structure of the code influence the distribution of accepted point mutations.

BLOSUM matrices [Henikoff & Henikoff 1992], Gonnet matrix [Gonnet et al. 1992] and more (e.g. [McLachlan 1971]).

Several matrices were extracted from secondary structure propensities of the amino acids [Levin et al. 1986, Mohana Rao 1987]. Other matrices, which proved to be very effective for protein sequence comparison, are those that are based on structural principles [Johnson & Overington 1993, Risler et al. 1988]. These matrices reflect the statistics of pairwise substitutions that are observed at structurally equivalent positions in aligned structures of proteins from the same family. These matrices may increase the accuracy of sequence alignment and perform well in detecting homologous proteins. They might be a good choice in aligning relatively mutable regions (sequence derived matrices are based on conserved regions, and therefore cannot accurately model more mutable regions). However, in such regions gaps are much more frequent and it is necessary to include gaps in the alignments. No theoretical model has been established for gaps (see 2.4.4), and when modeled improperly, gaps can significantly reduce the performance and the alignment accuracy. Moreover, it is not clear how to select the parameters for gap penalties relative to the substitution matrix.

The two most extensively used families of scoring matrices are the PAM matrices and the BLOSUM matrices. A detailed description of these matrices is given in the next two sections.

2.4.1 The PAM family of scoring matrices

PAM matrices were proposed by Dayhoff et al in 1978 based on observations of hundreds of alignments of closely related proteins. The frequencies of substitution of each pair of amino acids were extracted from alignments of proteins of small evolutionary distance, below 1% divergence, i.e. at most one mutation per 100 amino acids, on average. These frequencies, normalized to account for the frequencies of single amino acids, resulted in the PAM-1 matrix. The PAM-1 matrix reflects an amount of evolutionary change that yields on average one mutation per 100 amino acids. Accordingly, it is suitable for comparison of proteins which have diverged by 1% or less. The PAM-1 matrix is then extrapolated to yield the family of PAM-k matrices. Each PAM-k matrix is obtained from PAM-1 by k consecutive multiplication, and is suitable for comparison of sequences which have diverged k%, or are k evolutionary units apart. For example, PAM-250 = $(\text{PAM-1})^{250}$ reflects the frequencies of mutations for proteins which have diverged 250% (250 mutations per 100 amino acids[5]). These matrices were later refined by [Jones et al. 1992] based on much larger data set. The significant differences were detected for substitutions that were hardly observed in the original data set of [Dayhoff et al. 1978].

The acronym PAM stands for Percent of Accepted Mutations (and hence the distance is in percentages) or for Point Accepted Mutations (and hence the distance in number of mutations per 100 amino acids).

[5]Though the definition of PAM-250 seems odd, it still make sense, as is subsequently explained.

Computing PAM matrices (following the scheme described in [Setubal & Meidanis 1996])

Each evolutionary distance defines a probability transition matrix M. The scores matrix S, is then obtained from M. The initial matrix - PAM-1 is derived from:

(1) A list of accepted mutations.

(2) The probability of occurrence p_a of each amino acid ($\sum_a p_a = 1$).

The list of accepted mutations is created by observing mutations that occur in alignments of closely related homologous proteins. Such pairs of proteins which have diverged through only small number of mutations, so that the correct alignment is obvious. The term "accepted" is used to emphasize that these mutations didn't destroy the protein's functionality, and the major properties of the protein were preserved. Selecting closely related proteins also minimizes the probability of mediated mutations ($a \to b \to c$), which are inappropriate for the computation of PAM-1.

The probability for each amino acid can be estimated from the relative frequencies of amino acids in a large, heterogeneous set of proteins.

The frequencies f_{ab} of mutations ($a \leftrightarrow b$) are calculated from the list of accepted mutations. The mutations are undirected events, that is, given a pair of aligned amino acids, we do not know which amino acid mutated into the other. Therefore, $f_{ab} = f_{ba}$. Let $f_a = \sum_{b \neq a} f_{ab}$, i.e. the total number of mutations in which a was involved. Let $f = \sum_a f_a$ be the total number of amino acids involved in mutations (twice the number of mutations).

In the PAM-1 matrix the element M_{ab} denotes the probability that amino acid a mutates into the amino acid b. The element M_{aa} denotes the probability that the amino acid remains unchanged during the corresponding evolutionary interval, and it is derived from the relative mutability m_a of the amino acid a (the tendency/probability of amino acid a to change). It is defined as

$$m_a = \frac{f_a}{100 \cdot f \cdot p_a} \tag{2.11}$$

Intuitively, $\frac{f_a}{f}$ gives the relative frequency of the mutations in which amino acid a is involved. The division by p_a normalize this relative frequency in the frequency of occurrence of the amino acid a, to avoid bias due to different frequencies of the amino acids. The division by 100 normalize this quantity to correspond to the specific evolutionary time interval. This way the total amount of possible change sums into 1 mutation per 100 amino acids, as will be shown soon (in terms of probability, the probability of a mutation is 0.01).

The probability that amino acid a remains unchanged is the complementary probability $M_{aa} = 1 - m_a$. The probability that a mutates into b is given by

$$M_{ab} = P(a \to b) =$$

$$P(a \to b/\text{a changed}) \cdot P(\text{a changed}) = \frac{f_{ab}}{f_a} \cdot m_a$$

These calculations are based on a simple Markov-type model of protein evolution, according to which the probability that amino acid change does not depend on the past, nor does is depend on the other amino acids in the sequence. These two assumptions are clearly oversimplified.

The matrix M has the following properties: The probability that amino acid a mutates into any amino acid b including itself (the probability that it remains unchanged) sum to 1, as required:

$$\sum_b M_{ab} = \sum_{b \neq a} \frac{f_{ab}}{f_a} \cdot m_a + M_{aa} = \frac{f_a}{f_a} m_a + M_{aa} = m_a + 1 - m_a = 1$$

Also, the probability that the sequence remains unchanged is

$$\sum_a p_a M_{aa} = \sum_a p_a (1 - m_a) = \sum_a p_a - \sum_a p_a \frac{f_a}{100 \cdot f \cdot p_a} = 1 - \frac{1}{100} = 0.99 \tag{2.12}$$

Therefore, the expected rate of mutations is 1 per 100 amino acids, and the PAM-1 matrix corresponds to one unit of evolution, as required (this explains the normalization in equation 2.11). It should be noted that one unit of evolution does not necessarily equal to one unit of time, as different proteins from various protein families have evolved at different rates.

Given the basic matrix M, it is now possible to get the transition probabilities for larger evolutionary distances. For example, the probability that a mutates to b in two PAM units of evolution is given by $\sum_c M_{ac}M_{cb}$. Therefore, the matrix that correspond to two units of evolution is given by PAM-2 $= M \cdot M$, and for k units of evolution PAM-k $= M^k$. For large k (on the order of a thousand) M^k converges to a matrix with identical rows, where each row equals the distribution of amino acids. Therefore, for large evolutionary distances, no matter what amino acid is chosen, the probability that it mutates into b is simply p_b.

The scoring matrix S is obtained from the transition probability matrix M. The entries are defined by the likelihood ratio of two events: a pair is a mutation versus a random occurrence. The probability that a changes into b is M_{ab} but there is a probability p_b that we encounter b in the second sequence, just by chance. The likelihood ratio is $\frac{M_{ab}}{p_b}$. The score S_{ab} is defined as 10 times the logarithm of this ratio. The logarithm is taken because the alignment score is the sum of pairwise scores, and this way it corresponds to the logarithm of the product of the ratios (log-likelihood). The factor 10 is used because the similarity scores are rounded to integers to speed up the calculations, and this reduces the numerical error. For distance of k evolutionary units

$$S^k_{ab} = 10 \cdot \log_{10} \frac{(M^k)_{ab}}{p_b}$$

and the matrix is symmetric $S^k_{ab} = S^k_{ba}$ (recall that the accepted mutation used for defining the scores are undirected events).

The PAM-250 matrix

The PAM-250 matrix is one of the most extensively used matrices in this field. This matrix corresponds to a divergence of 250 mutations per 100 amino acids. Naturally one may ask whether it makes sense to compare sequences which have diverged this much. Surprising as it may seem, when calculating the probability that a sequence remains unchanged after 250 PAMs (applying equation 2.12 for PAM-250), the outcome is that such sequences are expected to share about 20% of their amino acids. For reference, note that the expected percentage of identity in a random match is $100 \cdot \sum_a p_a^2$. Therefore, for a typical distribution of amino acids (in a large ensemble of protein sequences), we should expect less than 6% identities.

2.4.2 The BLOSUM family of scoring matrices

Unlike PAM matrices, which are extrapolated from a single matrix PAM-1, the BLOSUM series of matrices was constructed by direct observation of sequence alignments of related proteins, at different levels of sequence divergence. The matrices are based on “blocks” - a collection of multiple alignments of similar segments without gaps [Henikoff et al. 1999], each block representing a conserved region of a protein family. These blocks provide a list of (accepted) substitutions, and a log-odds scoring matrix can be defined:

$$s_{ab} = \log \frac{q_{ab}}{e_{ab}}$$

where q_{ab} is the observed relative frequency of the pair $a \leftrightarrow b$ given by $q_{ab} = f_{ab}/\sum_a \sum_b f_{ab}$ and e_{ab} is the expected probability of the pair, estimated from the population of all observed pairs as follows: the probability of the occurrence of amino acid a in a pair is $p_a = q_{aa} + \sum_b q_{ab}/2$, hence the expected probability of the pair $a \leftrightarrow b$ is $p_a p_b$ for $a = b$ and $p_a p_b + p_b p_a = 2p_a p_b$ for $a \neq b$.

To reduce the bias in the amino acid pair frequencies caused by multiple counts from closely related sequences, segments in a block with at least $x\%$ identity are clustered and pairs are counted

between clusters, i.e., pairs are counted only between segments less than $x\%$ identical. When counting pairs frequencies between clusters, the contributions of all segments within a cluster are averaged, so that each cluster is weighted as a single sequence. Varying the percentage of identity x within clusters results in a family of matrices BLOSUM-x, where x ranges from 30 to 100. For example, BLOSUM-62 is based on pairs that were counted only between segments less than 62% identical.

2.4.3 Information content of scoring matrix

Theorem 3 has a direct bearing on the question of how to choose the appropriate substitution matrix. It states that the frequency of a letter a aligned with the letter b in the best scoring match of two random sequences converges to

$$q_{ab} = p_a p_b \exp^{\lambda \cdot s(a,b)} \tag{2.13}$$

as the length of the compared sequences grows without bound. These frequencies are called **target frequencies**. Hence, any substitution matrix has an implicit target distribution for aligned pairs of amino acids, which can be easily calculated from the scores $s(a,b)$. According to equation 2.3 these frequencies sum to 1. The implicit target frequencies of a matrix characterize the best scoring alignments, i.e. the alignments this matrix is **optimized** to find. In other words, only if the frequencies of aligned pairs in a match resemble the target frequencies will the corresponding match have a high score. Therefore, it is claimed [Karlin & Altschul 1990, Altschul 1991] that a matrix is optimal for distinguishing true distant homologies from chance similarities, if the matrix's target frequencies correspond to the real frequencies of paired amino acids in the alignment of distantly related proteins.

Equation 2.13 can be restated as:

$$s_{ab} = \frac{1}{\lambda}\left(\ln \frac{q_{ab}}{p_a p_b}\right) \tag{2.14}$$

I.e. the score for an amino acid pair can be written as the logarithm (to some base) of the pair's target frequency divided by the product of their probability of occurrence. This denominator is the probability of the occurrence of this pair under independent selection. This ratio thus compares the probability of an event under two alternative hypotheses, and is termed likelihood or odds ratio. Therefore, each scoring matrix is implicitly a "log-odds" matrix, even if the underlying model is not based on observed substitutions. The PAM and the BLOSUM matrices are explicitly of this form.

By equation 2.3 it follows that multiplying a substitution matrix by a constant factor α is equivalent to dividing λ by α but does not alter the implicit target frequencies, nor the implicit form of log-odds matrix [6]. Such scaling merely corresponds to a different base for the logarithm in equation 2.14. If λ is chosen to be $\lambda = \ln 2$, the base for the logarithm is 2, and scores can be viewed as bits of information.

How many bits of information are needed to deduce a reliable relationship? If the expected number of MSPs with a score of at least S (equation 2.10) is set to E and the equation is solved for S, then

$$S = \log_2 \frac{K}{E} + \log_2 Nn$$

Recall that this is the score above which the number of hits that are expected to occur at random is $< E$. If a very low value is set for E, then the match is very significant and probably biologically meaningful. Therefore, this score, expressed in bits, can be viewed as the (minimal) number of bits needed to distinguish an MSP from chance (with error rate $< E$). For a typical substitution matrix K is of the order of 0.1, and for an alignment to be considered significant, E should be 0.05 or less [Altschul 1991, Altschul & Gish 1996]. Therefore the dominant term is the $\log_2 Nn$. In other words, to distinguish an MSP from chance, the number of bits needed is roughly the number of bits needed to specify where the MSP starts among Nn possible positions [Altschul 1991]. For example, for an average protein length of 350, and the SWISSPROT database with over 20,000,000 amino acids, at least 33 bits are needed.

With this interpretation, it is possible to get a rough measure of which matrix is appropriate for a search. Matrices can be evaluated by their information content, i.e., the average information per position,

$$H = \sum_{a,b} q_{ab} s_{ab} = \sum_{a,b} q_{ab} \log_2 \frac{q_{ab}}{p_a p_b}$$

which is the relative entropy of the target and background distributions. The higher the value, the better the distributions are distinguished, and the shorter is the length of an alignment *with the target distribution* that can be distinguished from chance (i.e., the **minimum significant length** is shorter).

For PAM matrices the information content decreases as the PAM distance increase. For example, PAM-120 has an information content of 0.98 bits per position. Assuming that at least 33 bits are needed to distinguish a true relationship from chance similarity (see above), the minimum significant length is 34. PAM-250 has an information content of 0.36 bits per position and the minimum significant length is 92. This is much longer than many regions of possible similarity such as domains or motifs. Therefore, short motifs can be detected by PAM matrices only if they have diverged a small PAM distance. For BLOSUM matrices the information content is higher when the index of the matrix is higher (e.g., BLOSUM-100 has an information content of 1.45 bits per position, while BLOSUM-45 has an information content of 0.38 bits per position). It is

[6] Theorem 3 and its implications for scoring matrices hold for local alignments. For global alignments, multiplying all scores by a constant has no effect on the relative scores of different alignments (as for local alignments). However, the same is true when adding a constant a to the score of each aligned pair and $a/2$ to a gap. Such transformation entails that no unique log-odds interpretation of global substitutions matrices is possible, and probably no theorem about target frequencies can be proved [Altschul et al. 1990].

important to note that higher information content does not signify a better performance in terms of detecting distant homologies. It is the target distribution of a matrix that determines if the matrix is optimal for a specific search. Therefore, a matrix like the BLOSUM-100, which reflects substitutions between closely related proteins, is not appropriate for the purpose of detecting distant homologies, despite its high information content. A commonly used matrix is the BLOSUM-62, which has an information content of 0.7 bits per position (the same as PAM-160). This matrix is appropriate for comparison of moderately diverged sequences, and it is considered to be one of the best matrices for database searches due to its overall performance, which is superior to all PAM matrices (see next section).

Choosing the scoring matrix

When comparing two sequences, the most effective matrix to use is the one which corresponds to the evolutionary distance between them (see previous section). However, we usually do not know this distance. Therefore, it is recommended to use several scoring matrices which cover a range of evolutionary distances, for example PAM-40, PAM-120 and PAM-250. In general, low PAM matrices are well suited to finding short but strong similarities, while high PAM matrices are best for finding long regions of weak similarity.

Exhaustive evaluations have been carried out to compare the performance of different scoring matrices [Henikoff & Henikoff 1993, Pearson 1995]. The evaluation procedures are performed as follows: a few hundred protein families are used as a benchmark, and the quality of a matrix is estimated by the portion of these families it is able to detect in a database search. Specifically, a query sequence is chosen from each family, and a database search is performed, each time with a different scoring matrix. All family members with a score above a certain threshold are considered to have been detected. The threshold can be chosen, for example, as the the score at 1% error rate, or the score above which the number of related sequences equals the number of unrelated sequences [Pearson 1995] (see also chapter 6). The matrix which detects the maximum number of members from the family is considered optimal for this family. The best matrix may vary among different families, therefore the quality is averaged over all families in the reference set.

These studies show that log-odds matrices derived directly from alignments of highly conserved regions of proteins (such as BLOSUM matrices or the Overington matrix, which is based on structural alignment [Johnson & Overington 1993]) outperform extrapolated log-odds matrices based on an evolutionary model, such as PAM matrices. Moreover, the accuracy of alignments based on extrapolated matrices decreases as the evolutionary distance increases. This suggests that extrapolation cannot accurately model distant relationships, and that the PAM evolutionary model is inadequate. BLOSUM matrices were shown to be more effective in detecting homologous proteins. Specifically, BLOSUM-62 and BLOSUM-50 gave superior performance in detecting weak homologies. These matrices offer good overall performance in searching the databases. The best hybrid of matrices for searching in different evolutionary ranges is either BLOSUM 45/62/100 or BLOSUM 45/100 plus the Overington matrix.

2.4.4 Gap penalties

There is no mathematical model to explain the evolution of gaps. Practical considerations (the need for a simple mathematical model, time complexity) have led to the broad use of linear gap functions. However, there is evidence that the real behavior is quite different. By observing alignments of related proteins, Gonnet et al. [Gonnet et al. 1992] have empirically shown that the probability for (opening) a gap increase linearly with the PAM distance of the two sequences. However, the probability of a gap of length k decreases as $k^{-\frac{3}{2}}$, **independent** of the PAM distance of the two sequences. This offers a further evidence to the hypothesis that gaps are not created by consecutive events of insertions/deletions. Here is a possible simplified explanation for this functional dependency: Given an accepted gap (inserted/deleted chain), it is reasonable to assume that the two ends of the extracted/inserted chain lie structurally close to each other, so that the chain's insertion/deletion does not affect much the global 3D structure of the protein, and hence its functionality. The probability that the two ends of a randomly coiled chain are placed spatially close is inversely proportional to the mean volume it occupies. This volume increases as $k^{\frac{3}{2}}$ for random chains of length k which explains the dependency of $k^{-\frac{3}{2}}$.