Pairwise Comparisons and Heuristic Searches

Lecturer: Marina Alexandersson

30 October 2002

Main questions:

1. How can we measure similarity between two sequences?
   - Pairwise alignment
   - Substitution matrices
   - Gap penalties
2. How do we find the best alignment?
   - Global and local alignment
   - Heuristic algorithms
3. When is an alignment good enough to indicate homology?
   - Significance of scores

1 Pairwise alignment

The idea is that sequence similarity might indicate a common ancestor some million years back.

The game in pairwise alignment is to place the two sequences on top of each other and insert spaces (gaps) in various numbers and places in both sequences to obtain the highest number of columns of identical residue pairs. Alignment algorithms strives to model the mutational process giving rise to the two sequences.

The basic mutational processes are

- substitutions: replace a residue with another
- insertions: add residues
- deletions: remove residues

Insertion and deletions result in gaps in the alignment.

When calculating the total score of an alignment we assume independence between residues, such that the probability of the alignment

$$\begin{aligned} x &: x_1 x_2 \ldots x_n \\ y &: y_1 y_2 \ldots y_n \end{aligned}$$

is

$$\Pr(\text{alignment}) = \Pr(x_1, y_1)\Pr(x_2, y_2) \cdot \ldots \cdot \Pr(x_n, y_n)$$

or

$$\ln(\Pr(\text{alignment})) = \ln(\Pr(x_1, y_1)) + \ln(\Pr(x_2, y_2)) + \ldots + \ln(\Pr(x_n, y_n)).$$

2 Substitution matrices

We want to separate alignments of homologous sequences from alignment of non-homologous sequences. That is we want to know whether a certain alignment infers homology or has occurred by pure chance (and thus the sequences are independent).

One way is by using a relative likelihood

$$\frac{\Pr(x,y|M)}{\Pr(x,y|R)}$$

of the model (M) for homology and the model (R) that the alignment occurred at random.

Random model: (no homology)
At each position i in each sequence the residue a_i (DNA base or amino acid) occurs with probability q_{a_i}. Both sequences and positions within each sequence are independent.

Match model: (homology)
The residues a_i and a_j at positions i and j in respective sequence occur together with probability $p_{a_i a_j}$. Positions within each sequence are still assumed to be independent, but the sequences are now assumed to be dependent (or related).

Example

Assume that we have the following alignment

A D E
B D F

This alignment has the probability

Random model: $q_A\ q_D\ q_E\ q_B\ q_D\ q_F$
Match model: $p_{AB}\ p_{DD}\ p_{EF}$

In general for sequences $x : x_1 \dots x_n$ and $y : y_1 \dots y_n$

$$\begin{aligned}\Pr(x,y|M) &= \Pr(\text{alignment if homologous}) = \prod_{i=1}^{n} p_{x_i y_i} \\ \Pr(x,y|R) &= \Pr(\text{alignment if not}) = \prod_{i=1}^{n} q_{x_i}\ q_{y_i}\end{aligned}$$

and

$$\begin{aligned}x \text{ and } y \text{ really homologous} &\iff \Pr(x,y|M) \geq \Pr(x,y|R) \\ &\iff \frac{\Pr(x,y|M)}{\Pr(x,y|R)} \geq 1 \\ &\iff S = \ln\left(\frac{\Pr(x,y|M)}{\Pr(x,y|R)}\right) \geq \ln(1) = 0.\end{aligned}$$

If S is "large enough" we reject the random model and assume x and y to be homologous.

$$S = \ln\left(\frac{\Pr(x,y|M)}{\Pr(x,y|R)}\right) = \ln \frac{p_{x_1 y_1}\ p_{x_2 y_2} \cdot \ldots \cdot p_{x_n y_n}}{q_{x_1} \cdot \ldots \cdot q_{x_n} q_{y_1} \cdot \ldots \cdot q_{y_n}} = \sum_{i=1}^{n} \ln \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} = \sum_{i=1}^{n} s(x_i, y_i)$$

where $s(x_i, y_i)$ is the score for pair (x_i, y_i).

The scores should be such that

- identities and conservative substitutions get a positive score, $p_{x_i y_i} > q_{x_i} q_{y_i}$
- neutral substitutions get score 0, $p_{x_i y_i} = q_{x_i} q_{y_i}$
- non-conservative substitutions get negative scores, $p_{x_i y_i} < q_{x_i} q_{y_i}$.

A substitution matrix for amino acids is a 20×20 matrix of the scores for each residue combination. If for all pairs (a_i, a_j) we knew the probabilities $p_{a_i a_j}$, q_{a_i} and q_{a_j}, we could calculate all 210 distinctive entries in the matrix (210 since $s(a_i, a_j) = s(a_j, a_i)$), and then use it to check if S is large enough.

But we don't! Another way is to estimate them from confirmed alignments. Problems with this:

- Hard to find a random sample. Proteins come in families, so the known proteins are highly biased.
- Different pairs of proteins have different distances to their common ancestor, and thus their scoring matrices would be different. We don't want a scoring matrix for each possible evolutionary distance, but rather just one matrix altogether.

If the common ancestor is very recent we would expect p_{ab} to be small for $a \neq b$, and $s(a, b)$ should be strongly negative. If long time has passed since the sequences diverged we expect p_{ab} to tend to $q_a q_b$, and so $s(a, b)$ should be close to zero for all a, b.

2.1 PAM matrices (Dayhoff)

PAM = point accepted mutation.

The PAM1 matrix was constructed from 71 protein families. Sequences of at least 85% identity were aligned. This requirement was because

- resulted in unambiguous alignment
- chances of *two* substitutions in one position is small.

The PAM1 matrix consists of transition probabilities $\Pr(b|a)$ from amino acid a to b in protein sequences of such an evolutionary distance that the average number of substitutions is 1% of the number of positions. That is, exposing a protein of 100 amino acids to the evolutionary change in such a time interval results in on average 1 substitution.

To extrapolate to longer evolutionary times we assume that the substitutions occur as a Markov process, and if PAM1 consists of entries $\Pr(b \mid a, t = 1)$ then a PAM2 matrix is constructed by

$$\Pr(b \mid a, t = 2) = \sum_c \Pr(c \mid a, t = 1) \Pr(b \mid c, t = 1)$$

i.e. substitution from a to b via some arbitrary residue c. A PAML is constructed by iteration

$$\Pr(b \mid a, t = L) = \sum_c \Pr(c \mid a, t = L - 1) \Pr(b \mid c, t = 1).$$

The score matrix is then obtained by

$$s(a, b \mid t) = \ln \frac{\Pr(b \mid a, t)}{q_b}.$$

One disadvantage with the PAM matrices is that while there is only a small error in PAM1 it grows with distance and is rather large in PAM250. To get around this problem the BLOSUM (BLOcks SUbstitution Matrix) matrix family was constructed.

3 Gap penalties

It is natural to assume that gap scores should be negative.

Standard gap penalties for a gap of length g:

Linear: $\gamma(g) = -gd$
i.e. gap residues independent with score $-d$

Affine: $\gamma(g) = -d - e(g-1)$
i.e. opening a gap gets score $-d$ and then every succeeding extending gap gets score $-e$.

Gap penalties correspond to a probability model of alignment in the same way as the pairing of residues and modelling of substitutions. But if we assume that the residue distribution (the frequencies by which the residues occur) is the same whether the residues appear in a gap or not, the residue factor cancels out in

$$\frac{\Pr(x, \text{gap}|M)}{\Pr(x, \text{gap}|R)}$$

and leaves a likelihood depending on gap length g only.

4 Alignment algorithms

4.1 Global alignment: Needleman-Wunsch

Again we want to align two sequences, this time not necessesarily of the same length since we're allowed to insert gaps

$$\begin{aligned} x &: x_1 x_2 \dots x_n \\ y &: y_1 y_2 \dots y_m. \end{aligned}$$

We let

$$F(i, j) = \text{score of the best alignment of subsequences } x_1 x_2 \dots x_i \text{ and } y_1 y_2 \dots y_j.$$

Arrange these scores in an $n \times m$ matrix F. F is filled by using the dynamic programming method, starting with $F(0, 0) = 0$ and then fill from top-left to bottom-right.

If we think of this procedure as being a sequence generating machine, in the first step we can either emit one residue in each sequence, or just one residue in one sequence and a gap in the other. Thus, we have the options

$$\begin{array}{ccc} x_1 & x_1 & - \\ | & | & | \\ y_1 & - & y_1 \\ F(1,1) & F(1,0) & F(0,1) \end{array}$$

Then

$$\begin{aligned} F(1,0) &= F(0,0) - d = -d \\ F(0,1) &= F(0,0) - d = -d \end{aligned}$$

$$F(1,1) = \max \begin{cases} F(0,0) + s(x_1, y_1) \\ F(1,0) - d = -2d \\ F(0,1) - d = -2d \end{cases}$$

and so on, thus in general we get

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

and the margins

$$\begin{aligned} F(i,0) &= F(i-1,0) - d = F(i-2,0) - 2d = \ldots = -id \\ F(0,j) &= -jd. \end{aligned}$$

This way $F(n,m)$ (the last cell in the matrix) gives the score of the best alignment. To not just get the score, but the actual optimal alignment as well, we keep pointers in each position in the matrix to the best previous position. That is, for position (i,j) we remember which of $(i-1, j-1)$, $(i-1, j)$ or $(i, j-1)$ that gave rise to the score $F(i,j)$. Then we can backtrack through F, starting in (n,m), to obtain the best global alignment.

4.2 Dynamic programming

Now we can summarize the dynamic programming method. It consists of three components:

1. Recurrence relation – $F(i,j)$'s relation to previous F's.
2. Tabular computation – compute all $F(i,j)$ and fill F.
3. Traceback – start in (n,m) and backtrack through F.

4.3 Local alignment: Smith-Waterman

Instead of aligning the whole sequences we might want to find subsequences that are related. For instance two proteins might share the same domain, or a region might have genes in common but the regions in between have low similarity.

Local alignment attempts to find the best alignment between subsequences x and y. There are two differences from global alignment:

1.

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

That is, $F(i,j) = 0$ if all other scores are negative. This means that a local alignment ends when the score goes from positive to zero, and a new one starts at the next non-zero entry.

2. The best local alignment doesn't necessarily start in $F(n,m)$, so the traceback is done by first finding the largest $F(i,j)$ and start there. Then we backtrack until we hit 0.

For this to work the expected score of a random match must be negative. Otherwise long random matches will get a high score just based on their length. Also we have to have $s(a,b) > 0$ for some (a,b), otherwise the algorithm will not find any alignments at all.

5 Heuristic alignment algorithms

With dynamical programming we're guaranteed to find the optimal alignment, but unfortunately these algorithms are usually computationally complex (in speed and memory usage). Heuristic searches are less sensitive, but pretty good still and much faster. The idea is that most good alignments have short stretches of ungapped very high scoring matches. Hence, good alignments without such a stretch are missed, but these are rather rare.

5.1 BLAST = Basic Local Alignment Search Tool

BLAST is one of the most widely used tool in bioinformatics, and perhaps in biology all together. BLAST takes a query sequence (DNA or protein) and searches for local alignment matches in a database. The procedure is as follows:

- Find all substrings of length w (w-length words) in the database that aligns with words in the query sequence, where the alignment has score higher than some threshold t. These words are called *hits*.
- Extend each hit to see if it is contained in an alignment segment of score higher than some other threshold S.

Usually w is about 3-5 for amino acids and ~ 12 for DNA.

Example

Let $w = 2$, $t = 8$ and use a PAM120.

Query: QLNFSAGW

w-length words: QL, LN, NF, FS, ..., GW

Possible alignments of words in query to words in database:

QL :	$\begin{matrix}\text{QL}\\\text{QL}\end{matrix} = 11,$ $\begin{matrix}\text{QL}\\\text{QM}\end{matrix} = 9,$ $\begin{matrix}\text{QL}\\\text{HL}\end{matrix} = 8$		QL, QM, HL
LN :	$\begin{matrix}\text{LN}\\\text{LN}\end{matrix} = 9$		LN
NF :	$\begin{matrix}\text{NF}\\\text{NF}\end{matrix} = 12,$ $\begin{matrix}\text{NF}\\\text{AF}\end{matrix} = 8,$ $\begin{matrix}\text{NF}\\\text{NY}\end{matrix} = 8$		NF, AF, NY

...

Database entry: NLNYTPW...

Hits: NL, LN and PW

For each hit, try to extend

$$\begin{array}{c|c|l} \text{Q} & \text{LN} & \text{FSAGW} \\ \text{N} & \text{LN} & \text{YTPW} \\ \leftarrow & & \rightarrow \end{array}$$

to see if the hit is part of segment with score ≥ 20.

BLAST reports hits with their scores and E-value.

5.2 FASTA

For nucleotide searches, FASTA may be more sensitive than BLAST. The procedure is as follows:

- A lookup table is created for all identical matching words of length *ktup* (1-2 for proteins, 4-6 for DNA) between the query sequence and the database.
- The comparison of the query sequence and the database can be viewed as a set of dotplots, with the query as the vertical sequence and the database sequences as the horizontal. Diagonals with the largest number of words are registered.
- The best regions are rescored, using a scoring matrix, trying to extend the match for as long as possible and still have a score above a given threshold.
- Ungapped regions are joined if the total score is $\geq S$.
- The highest scoring candidates are realigned using a dynamical programming algorithm, but restricting it to a band around the candidate match.

6 Significance of scores

The classical approach uses the extreme value distribution to calculate the probability that the best match from a search of N *unrelated* sequences has score $geqS$. If this probability is very small we don't believe that the sequences are unrelated, and so they are likely to be homologous.

HSP = high scoring pair.

If two random sequences of lengths n and m are aligned, the probability of finding at least one segment pair with score $\geq S$ is

$$\Pr(\text{at least one HSP with score} \geq S) \approx 1 - \exp\{-Knme^{-\lambda S}\}$$

where K, λ depends on the scoring scheme. We call this probability the P-value.

The expected number of segment pairs having score $\geq S$ in the random model is

$$E[\#\text{HSPs with score} \geq S] = Knme^{-\lambda S}.$$

We call this the E-value.

Scores are often normalized to get rid of dependence on scoring system

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

and the E-value $\approx mn/2^{S'}$.

BLAST reports the normalized score and the E-value where m is the length of the query sequence and n the length of the entire database (the sum of all sequence lengths in the database).