Sequence Alignment

Generally speaking, this course is about those things which you can learn about biological questions of interest from genomic (DNA or RNA) and protein (amino acid composition) sequence information, or the development of sequence information from other forms of data (e.g., *de novo* peptide sequencing).

The oldest problem in the area is a natural one form the point of view of computer science: finding matches in sequences. Finding exact matches of a given string of letters inside another is not a hard problem. What makes the problem more difficult for our context is evolution. For a start, let us say that we are looking for related proteins across species, or down through evolutionary time. One already sees that the essential problem here will be to create a measure of how similar proteins are from the sequence data, appropriate to any given problem at hand. This may be driven by some

biochemical understanding of evolution as it happens at the amino acid composition level, or by expert knowledge of the chemical properties of particular amino acids. These latter differ markedly, which is why proteins are able to carry out such diverse functions. For the moment let us start with the abstract sequence alignment problem.

We are given a finite alphabet $\mathcal{A} = \{a_1, \ldots, a_N\}$, and we are given two strings of letters

$\mathbf{a} = a_1 \ldots a_n$

and

$\mathbf{b} = b_1 \ldots b_m$

The letters in the string are taken from $\mathcal{A}$, and can be repeated.

We want to see how similar they are. If $m = n$, there is an obvious *alignment*:

$$\star \qquad \begin{cases} a_1 \dots a_n \\ b_1 \dots b_n \end{cases}$$

If we think of each position in the alignment as independent, we could score the similarityof this alignment by assigning a score number to each pair $s(a, b)$ where the letters $a, b \in \mathcal{A}$. The score of the alignment $\star$ could then be given as

$$S(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} s(a_i, b_i).$$

This additivity corresponds to a simple probabilistic model of the behavior of sequences. *Modeling* will enter here when we try to choose the scoring matrix $s(a, b)$, What is relevant here?

Note first however that this is an inadequate framework since over time the amino acid composition of a protein with a given function may have evolved, and so there will have been *mutations*. Some of these will have been favored

evolutionarily, and this kind of scoring for a *"mismatch"*. Finally, particular letters (amino acid constituents) might have been eliminated or new ones inserted into the protein sequence. And we may simply be trying to compare sequences of different lengths. In any case we have to consider *realigning* the original sequences, searching for the best match. So if we have

$$\mathbf{a} = a_1 \dots a_n$$

and

$$\mathbf{b} = b_1 \dots b_m$$

we can try

$$\mathbf{a}^* = (a_1, \dots, -, \dots, a_i, \dots, -, \dots, a_n) = (a_1^*, \dots, a_L^*)$$

matched with

$$\mathbf{b}^* = (b_1, \dots, -, \dots, b_i, \dots, -, \dots, b_n) = (b_1^*, \dots, b_L^*)$$

Here we interpret the dashes as insertions or deletions, or *indels*, which can be anywhere (more or less: see below), and we have filled the sequences out to the same length. We have enlarged the alphabet, or assume it to have an extra letter "-". Now we can align the sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ in the obvious fashion, and we score

$$S(\mathbf{a}^*, \mathbf{b}^*) = \sum_{i=1}^{L} s(a_i^*, b_i^*)$$

and set

$$S(\mathbf{a}, \mathbf{b}) = \max_{\text{all spacings } *} S(\mathbf{a}^*, \mathbf{b}^*).$$

Note:

1) we need $s(a, -)$ and $s(-, a)$ for all letters a.

2) don't allow a - to align against another -.

3) there are many possible alignments now – see PS1 – so we can't find $S(\mathbf{a}, \mathbf{b})$ by enumerating cases, calculating and taking the highest number in the list.

4) when a - appears in a position, call it an indel; if two letters are aligned it is a match or a mismatch.

Issues:

1) designing the scoring matrix

2) scoring indels

3) calculating the score and the optimal alignment (which yields the best score)

All of this will depend on the basic *null model* assumption: that each position in the sequence is independent of every other, and that the scoring should be the same (as above) from position to position. Neither of these is, of course, true in reality, but it will give us a reference point from which to measure. Making more sensitive, *position dependent* scoring methods will be a significant later part of the

course. For now we go with the simplified assumptions.

1) Designing Scoring Matrices

These are best described for protein sequences because they are the ones which are more directly subject to evolutionary pressure. DNA has minimal requirements and while its detailed structure and conformation can be quite informative in cases, it is relatively featureless at this level. (The overall frequencies of A, C, G, T can be quite different among species, however.)

The original scoring matrices were due to Margaret Dayhoff, the *PAM matrices*. A PAM matrix is 20 $\times$ 20, and is symmetric. PAM = **P**oint **A**cceptable **M**utation because it is built up from a notion of point mutation in a protein. That is, we assume that there is a unit of evolutionary time with a fixed chance of a given amino acid residue changing once. This

has to be established by data, which means a collection of related proteins which one knows to have changed by one, two, etc., residues. Thereafter, there is generated a family of matrices by a Markov model of evolution.

(Discrete) Markov Chains:

We have a set of states $1, \ldots, n$ (say the amino acid residue in a given sequence position), and a probability distribution $p_1, \ldots, p_n$ on these states. We also have a notion of (discrete) time, and the distribution over the states evolves with time. We have at each step a transition matrix $P_{(i,j)}$ where the entries have the interpretation

$P_{(i,j)} =$ conditional probability "start at state i,

go to state j"

The Markov condition is that the distribution at "time" n depends only on the distribution at time $n-1$ and the Markov process is called stationary if the probabilities are given by a

transition matrix $P_{(i,j)}$ which is *independent* of time. If we write the time in explicitly, we have

$$p_j(n+1) = \sum_i p_i(n) P_{(i,j)},$$

where the matrix doesn't depend on n. Under these assumptions there follows a very simple but very useful relation, the Chapman - Kolmogoroff equation. If we let let $P_{(i,j)}(n)$ denote the transition probability from state $i \to j$ in n units of time, then the CK equation states

$$P_{(i,j)}(n+m) = \sum_k P_{(i,k)}(n) \cdot P_{(k,j)}(m).$$

That is, the evolution can be calculated by steps. This is just matrix multiplication of the transition matrices. In particular, the relations imply that

$$P(n) = P(1)^n.$$

This is what Dayhoff used to establish the PAM matrices. The point is that the probabilities of seeing an amino acid residue mutation are assumed to be independent random events where the position of occurrences are independent.

It seems fair to say that the underlying chemistry should be time independent over evolutionary history. Thus the PAM 1 matrix is the transition probability matrix for one amino acid residue to mutate to another. To examine similarities between sequences which one assumes are more distant evolutionarily, one uses the CK equation to find the more distant evolutionary probabilities.

WARNING: I have simplified things here: for ease of computation, the matrices have been scaled to have integer entries, and data is corrected to guarantee the Markov property.

One critique of the system is that the data was symmetrized: this means that for a given step, a transition $W \to T$, say, was binned together with a transition $T \to W$, leading to a symmetric matrix $P(1)$. Alternatives have been tried, though they are not conclusively better at this stage (PAM matrices are still used).

The approximation implied by the PAM matrices was surprisingly good. More data was analyzed about 25 years later, leading to the BLOSSUM matrices. These were calibrated more independently on the different time scales, and prove to be slightly more accurate. The PAM matrices tended to miss multiple mutations over longer time periods.

Back to scoring: we have at a given position the probability that we have a mutation $a \to b$ calculated from PAM, say, and we compare this joint distribution to the independent distribution, as in mutual information:

$$s(a,b) = \log \frac{p(a,b)}{p(a)p(b)}.$$

Here $p(a)$ are the overall frequencies of the amino acids.

(Question: are these time independent?? That is, while it is clear that in a given position the relative frequencies of a the various amino acid residues should change, it is less clear whether

the overall distribution changes, and whether some sort of overall conservation constraint should be imposed)

2. Scoring Indels (= Gaps):

It is harder to say what one should do to model indels, or gaps (= string of indels). There are two methods standardly used, though some of the motivation is clearly computational simplicity. The first is simply *linear gap penalties*: set

$$s(a, -) = s(-, a) = -d,$$

for all a, and some d. Usually d will be ¿ 0, so that one is lowering the score by aligning two sequences with gaps. Under the additivity assumption, then, the score for a gap of length n will be nd.

However, it seems that there are at least two quantities involved here: it appears relatively unlikely to find gaps in the first place, but then, once they have opened, e.g., in DNA coding

sequence, their lengths seem to be modeled well by the linear model above. This leads to the *affien gap penalty*: the penalty for a gap of length $n > 0$ is $\{e + (n-1)d\}$. Here e is an initiaton penalty, usually quite a bit higher than d, which continues to be the penalty for extending the gap by one amino acid position. Notice that now one has to be aware of more than what position one is at when scoring an alignment, since one needs to know whether one is starting a gap or continuing it.

These scores are also interpretable in terms of a probability model of the situation. The linear gap penalty is related to a simple geometric model for the length of a gap: waiting for the arrival of the next non-gap amino acid residue. The affine gap penalty has an interpretation as a conditional geometric model: given that the gap has opened, the probability of waiting so long for it to close. See PS1.

3. Scoring Algorithms: Dynamic Programming:

As mentioned earlier, there are too many possible alignments (an exponentially large number) of two sequences of lengths m and n, especially when one considers the sequences are considered as a query sequence and a reference sequence, where the reference sequence is actually a data base of sequences, such as GenBank at the NCBI. This database is, itself, growing exponentially. Fortunately, the linearity of the strings gives a way to make the computatioin necessary very efficiently (in time O(mn)).