Lecture #4: 8 April 2004
Topics: Sequence Similarity
Scribe: Sonil Mukherjee

1 Introduction

When aligning multiple sequences, the general idea is that if the evolutionary tree is known, we just do pairwise alignment on the closest two sequences, or alignments of sequences, in the order specified by the tree. We will discuss three systems used for aligning protein sequences: ClustalW, T-COFFEE, and ProbCons.

2 ClustalW

ClustalW starts by finding the score of the pairwise alignment between each pair of sequences, using a scoring function that is appropriate for proteins. For example, since the core of a protein has less insertions and deletions, and hydrophobic regions are more likely than hydrophillic regions to be in the core, the scoring function has a lower gap penalty in hydorphillic regions that in hydrophobic regions.

Now that it has all the scores, ClustalW can construct a tree by merging pairs of sequences with a higher alignment score before merging pairs with a lower score. However, it is not always correct to do this, as the two nodes with the highest alignment score, and thus the smallest distance in the correct tree, may still be merged with other nodes in the correct tree before they are merged with each other.

2.1 Neighbor joining

Definition 1. *An* **additive distance measure** *is a distance measure in which the distance between any pair of leaves is the sum of the lengths of the edges connecting them.*

If the distances are additive, neighbor joining is guaranteed to find the correct tree. It works as follows:

Define
$D_ij = d_ij - (r_i + r_j)$
Where
$r_i = (1/(|Leafs| - 2)) * \sum_k d_ik$

Theorem 2. *The above calculations ensure that D_ij is minimal iff i and j are neighbors.*

The proof of this theorem is omitted, but it shows that ClustalW can construct the tree correctly.

2.2 Weighted progressive alignment

In weighted progressive alignment, each leaf x has weight w_x proportional to the tree length attributable to x. That is, if there is an edge of length l, and x is one of n nodes that can be reached by following l, the weight attributable to x from this edge is n/l.

2.3 Limitations

Consider the following four sequences, shown as they would be optimally aligned:

Seq A GARFIELD THE LAST FA-T CAT
Seq B GARFIELD THE ——— FAST CAT
Seq C GARFIELD THE VERY FAST CAT
Seq D ——————- THE ——— FA-T CAT

Since proteins often have different lengths, terminal gaps are preferred to internal gaps. Therefore, ClustalW will prefer to align sequences A and B like this:

Seq A GARFIELD THE LAST FAT CAT
Seq B GARFIELD THE FAST CAT ——

instead of like this:

Seq A GARFIELD THE LAST FA-T CAT
Seq B GARFIELD THE ——— FAST CAT

Now the alignment of A to B is fixed and cannot be improved later. Therefore, it is impossible for ClustalW to find the optimal alignment of the four sequences. There are two methods that attempt to overcome this limitation: iterative refinement and consistency.

2.3.1 Iterative refinement

The main idea is, given a multiple alignment, to search the space around it for a better alignment.

Algorithm(Barton-Stenberg):

1. Align most similar $x(i)$, $x(j)$
2. Align $x(k)$ most similar to $(x(i)x(j))$
3. Repeat 2 until $(x(1)$ ... $x(n))$ are aligned
4. For j=1 to N Remove $x(j)$ and realign it to $x(1)$ ... $x(j-1)x(j+1)$ ... $x(n)$

5. Repeat 4 until convergence

Each time step 4 is run, the alignment either gets better or stays the same, so the algorithm is guaranteed to converge.

For example, say we align (x,y), (z,w), and (xy,zw) to get the following multiple alignment:

x: GAAGTTA
y: GAC–TTA
z: GAACTGA
w: GTACTGA

When we remove sequence y and realign it to the rest, we will see that we gain 3 matches with this alignment:

x: GAAGTTA
y: G–ACTTA
z: GAACTGA
w: GTACTGA

However, iterative refinement does not always work this well. Consider the following multiple alignment:

x: GAAGTTA
y: GAC–TTA
v: GAC–TTA
u: GAC–TTA
z: GAACTGA
w: GTACTGA

We can see that we will get more matches if we move the gap from the fourth space to the second space for y, v, and u, but since we only realign one sequence at a time, the algorithm will never fix this. When we remove y, for example and realign it, we will keep the gap in the fourth place because u and v also have their gaps in the fourth place.

2.3.2 Consistency

The idea here is to prevent errors when performing the intial alignments instead of fixing them later. This is done by, when aligning two sequences, consulting a third sequence to resolve any conflicts. In the following example of aligning x and y, suppose a portion of x, x_i aligns well with with two different protions of y, y_j and y_k.

x: ——————-x_i————————
y: ————y_j————-y_k———

It is unclear which part to align x_i to. But if we know that x_i aligns to z_k in a third sequence z, and z_l aligns to y_j, then we can choose to align x_i to y_j, knowing that this will be best when we align xy to z.

3 T-COFFEE

This picture summarizes T-COFFEE:

T-COFFEE starts with a users library, which contains the set of n sequences to be aligned. Its first step is to create the primary library.

3.1 Primary library

The primary library is built by first builidng a ClustalW library and a Lalign(local alignment version of ClustalW). T-COFFEE calls ClustalW to construct all pairwise alignments among the n sequences, and then calls Lalign to construct a library of local alignments. Sequence identity is then used to assign a primary weight to each alignment in either library. For example, for the alignment:

Seq A GARFIELD THE LAST FAT CAT
Seq B GARFIELD THE FAST CAT ——

there are 18 letters in the shorter sequence, and only 2 of them do not match, so the sequence identity is 100*(16/18) = 88, so the primary weight for this alignment is 88.

If an alignment is duplicated across two libraries, then it is merged into one entry with the new weight = sum of two previous weights. For example, if the alignment above is in both the ClustalW and Lalign libraries with a weight of 88 in each, then it becomes one entry with weight 88+88 = 176. This way, the ClustalW and Lalign libraries are megred into one primary library.

3.2 Library extension

Lets say A and C are aligned with a score of 77, and C and B are aligned with a score of 100, as shown below:

Seq A GARFIELD THE LAST FA–T CAT
Seq C GARFIELD THE VERY FAST CAT
Seq B GARFIELD THE —— FAST CAT

Now we have an alignment of A and B through C. The weight of this alignment W(A,B) = min(W(A,B), W(B,C)) = min(77, 100) = 77. To get the final W(A,B), we add this to W(A,B) from the primary library. So for this example, the final W(A,B) = 77 + 88 = 165. This value is put in the extended library. This approach requires examining all triplets of sequences, but not all will provide information. Consider the sequence D, whose alignments to A and B provide no information:

Seq A GARFIELD THE LAST FA-T CAT
Seq D ————————- THE ——— FA-T CAT
Seq B GARFIELD THE ——— FAST CAT

3.3 Running time

Letting L be the average sequence length and N be the number of sequences, the running time can be separated into four parts:

1. O(N^2L^2) time for pairwise library computation
2. O(N^3L) time for library extension
3. O(N^3) time for computation of the neighbot joining tree
4. O(NL^2) time for progressive alignment computation

Therefore, the total running time is O(N^2L^2) + O(N^3L)

4 ProbCons

Unlike the ClustalW and T-COFFEE, ProbCons is not discrete. It is similar to T-COFFEE, but it is probabilistic. It uses a probabilistic model for alignment, probabilistic consistency, and it optimizes expected accuracy rather than the most likely alignment.

4.1 Hidden Markov Models

Hidden Markov Models (HMMs) can be used to model alignments by having three states:

1. M corresponds to match, so when the HMM is in state M, a letter is emitted from both sequences.
2. I corresponds to a gap in the second sequence, so when the HMM is in state I, a letter is emitted from the first sequence only.
3. J corresponds to a gap in the first sequence, so when the HMM is in state I, a letter is emitted from the second sequence only.

Therefore, an alignment of two sequences corresponds can only have one sequence of states in the HMM. An example is shown below:

- - A G G C
T A G - - C
J J M I I M

Probabilites are assigned to every transition arrow to determine which state to go to next, and to every emission to determine which letter(s) to emit in each state. When doing calculations, the logs of these probabilities will be used to prevent underflow.

Once an HMM and its probabilities have been constructed, the Viterbi algorithm can be run on it to find the most likely alignment. We can also find posterior probabilities that two letters align by using the Forward and Backward HMM algorithms.

4.2 ProbCons details

Since alignments of proteins have lots of small gaps and some large gaps, it makes sense to model small and large gaps separately. ProbCons does this by using an HMM with 5 states instead of 3. The new states, Ilong and Jlong, model long gaps, and I and J model short gaps. The transition probabilites to enter Ilong and Jlong are less than those to enter I and J, but the probabilities to stay in Ilong and Jlong are greater than those to stay in I or J. This way, we enter the long gap states less often but stay in them for a long time when we do.

When performing pairwise alignment on two sequences x and y, ProbCons first derives P(x_i, y_j), the probability that x_i and y_j are aligned, for all positions (i,j) in x and y. Then it maximizes expected accuracy, rather than likelihood of alignment, given by the following equation:

$E(x, y) = (1/(min(|x|, |y|)) * \sum(i, j)P(x_i\ y_i|x, y)$

Finally, it uses probabilistic consistency, which, in the previous consistency example, uses the probability of z_l being aligned to y_j and y_k to determine with one x_i should be aligned to.

The ProbCons algorithm is as follows:

1. Calculate all pairwise posterior matrices $P(xy) = (\ P(x_i, y_j|x, y)\ )$
2. Calculate all pairwise distances $d_xy = E(x, y) = (1/(min(|x|, |y|))*\sum(i, j)P(x_i\ y_i|x, y)$
3. Apply hierarchical clustering on (d_xy) to build guide tree
4. Convert each $P(xy)$ to a sparse matrix by dropping very small entries
5. Apply consistency transformation: $P(xy) = 1/N \sum_z P(xz)\ *\ P(zy)$
6. Iterate 5 for 1-3 steps to get indirect relationships x-z-w-v-y etc.

7. Apply progressive alignment along the tree, at each step maximizing expected accuracy

8. Apply 100 rounds of iterative refinement

The three features, along with iterative refinement, independently provide improvements, and test have shown that ProbCons provides better results than ClustalW and T-COFFEE.

References