COS 551: Introduction to Computational Molecular Biology
Lecture 4: September 26, 2000
Lecturer Mona Singh
Scribe: Aneel Delawalla

Multiple Sequence Alignments I

In previous lectures, we considered alignments between two sequences. Here, we consider the case where we wish to align three or more entire sequences (i.e., *global* multiple sequence alignments). Usually, *local* multiple sequence alignment methods only look for ungapped alignments, or motifs, and we will return to motif finding in a future lecture.

Definition: Given k strings, $S_1, S_2, \ldots, S_k$, a multiple sequence alignment (MSA) is obtained by inserting gaps in the strings to make them all the same length.

E.g., the following is a MSA of 4 sequences MQPILLLV, MLRLL, MKILLL, and MPPVLILV.

M	Q	P	I	L	L	L	V
M	L	R	-	L	L	-	-
M	K	-	I	L	L	L	-
M	P	P	V	L	I	L	V

No column may be all gaps.

Multiple sequence alignments are used for many reasons, including:
(1) to detect regions of variability or conservation in a family of proteins,
(2) to provide stronger evidence than pairwise similarity for structural and functional inferences,
(3) to serve as the first step in phylogenetic reconstruction, in RNA secondary structure prediction, and in building profiles (probabilistic models) for protein families or DNA signals.

For pairwise alignments, we scored each column by looking at matches, mismatches, and gaps in the two sequences (in practice, protein sequences are scored using substitution matrices). However, now we have multiple characters in each column, and it is not obvious what the best way to score a column is. There are many possibilities. The sum-of-pairs (SP) is a common scoring scheme. Here, each column in an alignment is scored by summing the scores of all pairs of symbols in that column. The score of the entire alignment is then summed over all column scores. We will assume that a match = 1, a mismatch = -1, and a gap = -2. For example, the sum-of-pairs score of the 4^{th} column of the MSA given earlier is:

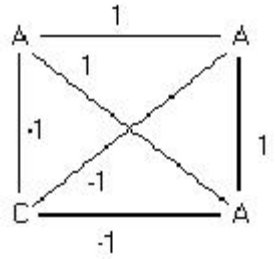
SP(I,-,I,V)
= score(I,-) + score(I, I) +score(I,V) + score(-,I) + score (-,V) + score(I,V)
= -2 + 1 + -1 + -2 + -2 + -1 = -7

Although there is never an entire column of gaps, if we look at any 2 sequences in the alignment, there may be columns where both have gaps. The value of score(-,-) is set equal to 0; this eliminates double counting (or extra penalization) for the use of gaps.

Let MSA-SP denote the optimum (i.e., highest scoring) MSA under the SP scoring system.

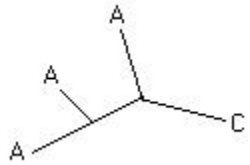
Note that it is not at all clear that sum-of-pairs is the best scoring system to use. The SP scoring system tends to overweight contributions of differences from many very similar sequences.

E.g., column A,A,A,C



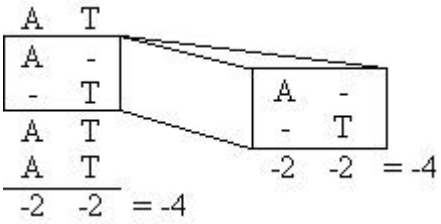
The sum-of-pairs score for this column is 3 - 3 = 0.

However, evolutionarily speaking, say the relationship between the sequences is:



Then a single C $\longrightarrow$ A mutation can explain the data, and thus SP tends to overcount mutations.

Note: An optimum MSA-SP may not give the optimal pairwise alignments. For example, if a match = 1, a mismatch = -1, and a gap = -2, then the optimum MSA-SP for the sequences AT, A, T, AT and AT is as follows, with the induced alignment between A & T also shown.



However, the optimum pairwise arrangement for the two sequences A and T is $\begin{array}{c} A \\ T \\ \hline -1 \end{array}$, which, if it were part of the MSA would yield:

$$\begin{array}{cc} A & T \\ A & - \\ T & - \\ A & T \\ A & T \\ \hline 2 & -9 \end{array} = -7$$

Determining optimum multiple sequence alignments

We now outline the algorithm for finding an optimum MSA under the SP measure. It is straighforward to see how to adapt it for other similiarity measures. However, this algorithm is usually not used in practice because it is computationally too time consuming. This approach uses dynamic programming.

Assume for simplicity that our k sequences $S_1, S_2, \ldots, S_k$ all have length n. If we were aligning 2 such sequences, we would need an array of size $[n + 1][n + 1]$ to account for all possible solutions. Similarly, for k sequences, we need an array of size $[n + 1]^k$, or a k-dimensional array.

As with the pairwise alignments, let's look at the last column in the alignment. For each sequence, there are 2 possibilities: either there is a letter or a gap in the last character position. Taken for all k sequences, there are then $2^k - 1$ possibilities (we subtract 1 because the case of every sequence having a gap in the last character position is not allowed).

Thus, we are keeping track of a k-dimensional array sim, where $\text{sim}(\vec{i}) = \text{sim}(i_1, i_2, \ldots, i_k)$
Then,
$\text{sim}(\vec{i}) = \max_{b,c} \{\text{sim}(\vec{i} - \vec{b}) + \text{SP}(\vec{c})\}$, where $\vec{b} \neq 0$, over all binary vectors $\vec{b}$ of length k

$$\vec{c}_j = \begin{cases} S_j[i_j], \text{ if } \vec{b}_j = 1 \\ \text{“-“}, \text{ if } \vec{b}_j = 0 \end{cases}$$

Vector $\vec{b}$ enumerates over all $2^k - 1$ possibilities. That is, when $\vec{b}_j = 1$, we're considering the case where the i_j^{th} character of a sequence j is used. Vector $\vec{c}$ keeps track of the actual characters used.

Therefore, we must:
(1) fill out the entire array which has $[n + 1]^k$ cells,
(2) consider $2^k - 1$ possibilities for each cell in the array, and
(3) compute the sum-of-pairs measure for each of these possibilities.

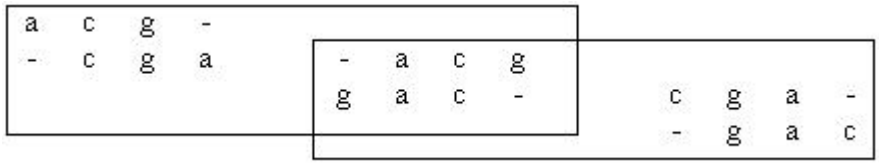
Thus, this algorithm would take roughly $(n + 1)^k(2^k - 1)(k^2)$ time for the SP measure. This is impractical even for numbers as small as $k = 10$. Thus, most packages for multiple sequence alignment do not compute the optimum MSA.

Multiple Sequence Alignment Heuristics: Progressive Alignments

Most packages use heuristics to compute multiple sequence alignments. The basic idea of many of these heuristics is to compute pairwise alignments and to merge alignments consistently. Note

that given all pairwise alignments, it is usually impossible to arrange an MSA consistent with all of them.

E.g., consider aligning the three sequences: acg, cga, gac. Their optimum pairwise alignments are as follows:



Merge using alignments with 1st sequence

-	a	c	g	-
-	-	c	g	a
g	a	c	-	-

Merge using alignments with 3rd sequence

-	-	a	c	g
-	g	a	c	-
c	g	a	-	-

Merge using alignments with 2nd sequence

a	c	G	-	-
-	c	G	a	-
-	-	G	a	c

Now the MSA is not consistent with the optimum pairwise alignments of the 2nd and 3rd sequences

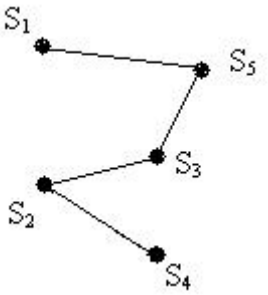
Now the MSA is not consistent with the optimum pairwise alignments of the 1st and 2nd sequences

Now the MSA is not consistent with the optimum pairwise alignments of the 1st and 3rd sequences

Note, however, that given any *k*-1 alignments relating *k* sequences, there is always an MSA consistent with all *k*-1 pairwise alignments. That is, there is an MSA such that the induced pairwise alignments in the MSA include the given *k*-1 alignments.

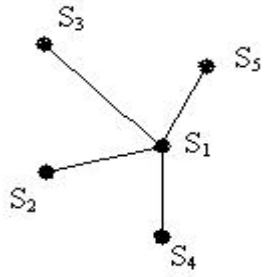
With $k = 5$, this would mean that there are 4 alignments relating all the sequences. For example, we could have pairwise alignments for the edges in the following graph.

Note that we can never have a cycle if we have *k*-1 alignments relating all *k* sequences, and thus we can maintain consistency with the *k*-1 alignments.



We can use this fact to devise multiple sequence alignment heuristics. One such heuristic for MSA is to just add the sequences to an alignment, one by one, maintaining consistency and following the “once a gap, always a gap” rule.

Star Alignment Heuristic: The twist in the star alignment heuristic is that you must (1) find the sequence that is the most similar to all the rest using pairwise alignment (see below) and (2) use it as the "center of a star" when aligning all other sequences. I.e., you will only use pairwise alignments with the center sequence when building the multiple sequence alignment.



S_1 is the center

Formally speaking, (1) is simply saying to find S_c such that $\sum_{i \neq c}$ sim (S_i, S_c) is maximized.

For example, say your sequences are:

S_1	A	T	T	G	C	C	A	T	T	
S_2	A	T	G	G	C	C	A	T	T	
S_3	A	T	C	C	A	A	T	T	T	T
S_4	A	T	C	T	T	C	T	T		
S_5	A	C	T	G	A	C	C			

Then, we first compute all pairwise alignments and scores. The following matrix gives the alignment scores for all pairs, as well as the sum of these scores for each sequence.

	S_1	S_2	S_3	S_4	S_5	
S_1	-	7	-2	0	-3	**2**
S_2	7	-	-2	0	-4	**1**
S_3	-2	-2	-	0	-7	**-11**
S_4	0	0	0	-	-3	**-3**
S_5	-3	-4	-7	-3	-	**-17**
	2	**1**	**-11**	**-3**	**-17**	

S_1 is the sequence most similar to the rest, and below are the best alignments between S_1 and the rest of the sequences.

S_1	A	T	T	G	C	C	A	T	T	
S_2	A	T	G	G	C	C	A	T	T	

S_1	A	T	T	G	C	C	A	T	T	-	-
S_3	A	T	C	-	C	A	A	T	T	T	T

S_1	A	T	T	G	C	C	A	T	T
S_4	A	T	C	T	T	C	-	T	T

S_1	A	T	T	G	C	C	A	T	T
S_5	A	C	T	G	A	C	C	-	-

Now, let's build the alignment.

Let's use the alignment of $S_1$and S_2.

S_1	A	T	T	G	C	C	A	T	T
S_2	A	T	G	G	C	C	A	T	T

S_1 and S_2 are aligned

Now, let's add S_3, using its alignment to S_1.

S_1	A	T	T	G	C	C	A	T	T	-	-
S_2	A	T	G	G	C	C	A	T	T	-	-
S_3	A	T	C	-	C	A	A	T	T	T	T

S_1, S_2, and S_3 are aligned

Then, let's add S_4, using its alignment to S_1.

S_1	A	T	T	G	C	C	A	T	T	-	-
S_2	A	T	G	G	C	C	A	T	T	-	-
S_3	A	T	C	-	C	A	A	T	T	T	T
S_4	A	T	C	T	T	C	-	T	T	-	-

S_1, S_2, S_3, and S_4 are aligned

Finally, let's add S_5, using its alignment to S_1.

S_1	A	T	T	G	C	C	A	T	T	-	-
S_2	A	T	G	G	C	C	A	T	T	-	-
S_3	A	T	C	-	C	A	A	T	T	T	T
S_4	A	T	C	T	T	C	-	T	T	-	-
S_5	A	C	T	G	A	C	C	-	-	-	-

S_1, S_2, S_3, S_4 and S_5 are aligned

For consistency, once a gap is added, it is never removed.

The running time is dominated by computing the pairwise alignments. If k sequences are length n, then:

(1) we compute $\frac{k(k-1)}{2}$ pairwise alignments

(2) each alignment takes time n^2.

Thus, the running time for computing all pairwise alignments is $O(k^2n^2)$. If l is an upperbound on alignment lengths, then we can merge alignments in k^2l time. Therefore, the total running time is $O(kn^2 + k^2l)$. Note that the Star Alignment does not optimize the SP criterion. Therefore, there is a trade-off between optimization and practicality. Actually, the star alignment method is not used much. In the next lecture, we'll discuss a similar method that is very widely used: the progressive alignment method of the ClustalW package. ClustalW uses the same basic method as the Star Alignment. However, it compares pairwise alignments and then adds individual sequences based on a given order of similarity.

Further Reading

Setubal & Meidanis. *Introduction to Computational Molecular Biology,* PWS Publishing Company, 1997. Chapter 3.