Global Multiple Sequence Alignment

Dannie Durand

In multiple sequence alignment, the goal is to align k sequences, so that residues in each column share a property of interest, typically descent from a common ancestor or a shared structural or functional role. Applications of multiple sequence alignment include identifying functionally important mutations, predicting RNA secondary structure, and constructing phylogenetic trees.

Given sequences s_1,...,s_k of lengths n_1,...,n_k, $\alpha = \{s'_1,...,s'_k\}$ is an alignment of s_1,...,s_k if and only if

- $s'_a \in (\Sigma')^*$, for $1 \leq a \leq k$
- $|s'_a| = l$, for $1 \leq a \leq k$, where $l \geq \max(n_1, \ldots, n_k)$
- s_a is the sequence obtained by removing gaps from s'_a
- No column contains all gaps

Scoring a multiple alignment

As with pairwise alignment, multiple sequence alignments (MSAs) are typically scored by assigning a score to each column and summing over the columns. The most common approach to scoring individual columns in a multiple alignment is to calculate a score for each pair of symbols in the column, and then sum over the pair scores. This is called sum-of-pairs or SP-scoring. For global multiple sequence alignment, SP-scoring can be used with either a distance metric or a similarity scoring function. The sum-of-pairs similarity score of an alignment of k sequences is

$$S_{sp}(s'_1, ..., s'_k) = \Sigma_{i=1}^{l} \Sigma_{a=1}^{k} \Sigma_{b>a}\, p(s'_a[i], s'_b[i]), \tag{1}$$

where l is the length of the alignment. As before, $p(x, y)$ is a numerical score that represents the similarity of x and y and $p(x, \_)$ is the gap score. Further, we define $p(\_, \_)$ to be zero. In pairwise alignment there is no need to assign a value to $p(\_, \_)$ because the definition of a pairwise alignment specifies that no column may contain two gaps. However, in a multiple alignment, two aligned sequences can have a gap in the same column (i.e., $s'_a[i] = s'_b[i] = \_$), as long as there exists at least one sequence in the MSA that does not have a gap in that column. As an example, let us calculate the SP-score for the alignment of three sequences shown below:

```
s1  A_TT
s2  A_T_
s3  ACAT
```

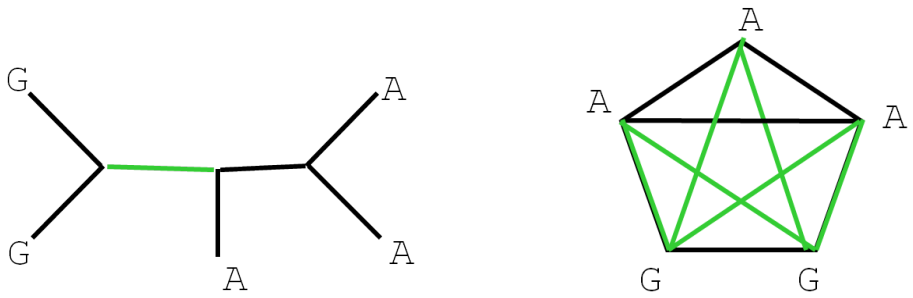
We can calculate the SP-score for each column separately:

	A_TT A_T_ ACAT
s_1, s_2	MOMg
s_1, s_3	MgmM
s_2, s_3	Mgmg

Note that the second column contains two gaps and that these are assigned a score of zero. The total SP-score is $5M + 2m + 4g$. (Is this alignment optimal? If not, how could you improve it?)

We can also use sum-of-pairs with distance scoring for global multiple alignment. This is how we would score the same alignment using unweighted edit distance:

	A_TT A_T_ ACAT
s_1, s_2	0001
s_1, s_3	0110
s_2, s_3	0111



(a) (b)

Figure 1: Two ways of scoring the column (A, A, A, G, G) in a multiple alignment. Green edges represent mismatches. **(a)** Scoring mutations on a tree. **(b)** Sum-of-pairs scoring

Sum-of-pairs scoring tends to overestimate the number of mutations required to explain the data. For example, a single mutation is required to explain the column (A, A, A, G, G), when scored on the tree in Fig. 1a. In contrast, SP-scoring assigns this column a score of six (Fig 1b), because SP-scoring is based on the implicit assumption that each pair of symbols is independent of all other pairs. Using a tree to score a multiple alignment could be more accurate than SP-scoring, but tree-scoring is rarely used for various practical reasons. Frequently, the true tree topology is

unknown. In addition, different columns in the alignment may have different histories and, in some cases, the residues in a column do not share a common ancestor. In these cases, it is not clear how a tree should be selected for scoring.

Given two sequences s_a and s_b in a multiple alignment, the pairwise alignment of s_a and s_b induced by the MSA, is the alignment obtained by deleting the other sequences in the MSA and then removing any column that contains two gaps. For example, in the multiple alignment below,

```
AC_T_G
A_GT_G
ACGTAG
```

the induced alignment of the first two sequences is

```
AC_TG
A_GTG.
```

Further, the pairwise alignment induced by the optimal multiple alignment is not necessarily the optimal pairwise alignment. In this example, the optimal pairwise alignment is

```
ACTG
AGTG.
```

Although the optimal pairwise alignment may have a better score, the induced pairwise alignment may be a biologically more realistic alignment because it reflects properties of the family as a whole.

A dynamic programming algorithm for multiple alignment

The dynamic programming algorithm used for finding the optimal global alignment of two sequences can be extended to the problem of global alignment of k sequences. First, let us consider a dynamic program to design three sequences. When a sum-of-pairs similarity score is used, the dynamic program for three sequences looks like this:

Input:

Sequences s_1, s_2, and s_3 of lengths n_1, n_2, and n_3, respectively.

Initialization:

$$\begin{aligned} a[i_1, 0, 0] &= a[i_1 - 1, 0, 0] + g \\ a[0, i_2, 0] &= a[0, i_2 - 1, 0] + g \\ a[0, 0, i_3] &= a[0, 0, i_3 - 1] + g \end{aligned}$$

Recurrence:

$$a[h,i,j] = \max \begin{cases} a[i_1-1, i_2, i_3] + 2g \\ a[i_1, i_2-1, i_3] + 2g \\ a[i_1, i_2, i_3-1] + 2g \\ a[i_1-1, i_2-1, i_3] + 2g + p(s_1[i_1], s_2[i_2]) \\ a[i_1-1, i_2, i_3-1] + 2g + p(s_1[i_1], s_3[i_3]) \\ a[i_1, i_2-1, i_3-1] + 2g + p(s_2[i_2], s_3[i_3]) \\ a[i_1-1, i_2-1, i_3-1] + p(s_1[i_1], s_2[i_2]) + p(s_1[i_1], s_3[i_3]) + p(s_2[i_2], s_3[i_3]) \end{cases}$$

Store the indices of the entry in a that minimize the right hand side of the recurrence in a traceback matrix, $\mathcal{T}$.

Trace back:

From $\mathcal{T}[n_1, n_2, n_3]$ to $\mathcal{T}[0, 0, 0]$ to obtain the optimal alignment.

Output:

The optimal alignment score, $a[n_1, n_2, n_3]$.

The optimal alignment of s_1, s_2, and s_3 with respect to similarity function, $\mathcal{S}$.

The dynamic program for multiple sequence alignment has the same structure as the algorithms for pairwise sequence alignment, but the initiation and recurrence steps are more complex. Since the alignment matrix, a, is a 3-dimensional matrix, the first row in each of the three dimensions must be initialized. The algorithm calculates the entries in a according to the recurrence proceeding diagonally from $a[0, 0, 0]$ to $a[n_1, n_2, n_3]$. As in the pairwise case, a traceback matrix, $\mathcal{T}$, is used to record the indices that gave the optimal score for each i_1, i_2, i_3 prefix. Once the entire matrix has been filled in, the optimal score is found in $a[n_1, n_2, n_3]$. Note that this is analogous to the pairwise global alignment algorithm, where $a[n, m]$ contains the optimal score.

It is straightforward, if messy, to generalize the dynamic program for three sequences to a dynamic program for k sequences. To convince yourself that you understand how this works, try writing down the algorithm for four sequences. For three sequences, the recurrence has seven entries. How many entries will there be in the recurrence when $k = 4$? How many entries will there be for arbitrary k?

The computational complexity of the dynamic programming algorithm to align k sequences is $O(n^k 2^k k^2)$. To see this, note that for k sequences, the alignment matrix has $O(n^k)$ entries. For each entry in a, the recurrence relation considers $O(2^k)$ neighboring cells. Calculating the SP-score for each of those neighbors requires $O(k^2)$ time. Thus, the time complexity of the dynamic program

for multiple sequence alignment is exponential in the number of sequences. Given 10 sequences of length at most 500, it is possible to calculate the optimal alignment using dynamic programming. For larger problem instances, it is necessary to use a heuristic.