Computational Methods in Molecular Biology
Lecture #19: MSA - Multiple Sequence Alignment

Tuvia Kochavi & Ido Tal

11/1/16

1 Background:

Until now we got sequences, then we build distance matrix and then we build phylogenetic tree:

2 MSA:

In previous lectures we learned how to deal with two sequences. We did that with pairwise alignment (global, local or overlap).

So, now we want do the same, but for higher dimensions.

For multiple alignment we build matrix that all row is a sequence, and for each column we will calculate the score of the column.
Each row represent a sequence, and each column represent position in the sequence. i.e. $M[i,j]$ represent the letter that appears in sequence $i$ in position $j$.

$$\begin{array}{c|c|c}
    & X^1 + gaps \\
    \hline
    \vdots & \vdots \\
    \vdots & \vdots \\
    \vdots & \vdots \\
    \hline
    X^n & \hline
\end{array}$$

$M = (A^i_{n-1})_{i=1}^n$

\[ \text{score}(M) = \sum_i S(\hat{A}_i) \]

How we can calculate the score of each column?

- Similarity percentage.
- Most common letter: $a = \text{mode}(\hat{A}_i)$, $S = \sum_i (\sigma(a, A^i_n))$
- Sum of the distance in pairs: $\sum_{j,k(j \neq k)} \sigma(A^m_j, A^m_k)$
- If we have phylogenetic tree, we can calculate the evolution score.

Two ways for calculate the MSA:

- In the naive algorithm, we do the same as in pairwise alignment but we will hold a multiple dimensions matrix.

In the two dimensional matrix each cell $\alpha_{i,j}$ present the best alignment that end in $x^1_i x^2_j$.

And in the multiple dimensions:

\[
\alpha_{i_1,i_2,\ldots,i_n} = \max \left\{ \begin{array}{l}
\alpha_{i_1-1,i_2-1,\ldots,i_n-1} + S(x^1_{i_1}, x^2_{i_2}, \ldots, x^n_{i_n}) \quad \text{we moved forward in all n sequences} \\
\alpha_{i_1,i_2-1,\ldots,i_n-1} + S(-, x^2_{i_2}, \ldots, x^n_{i_n}) \quad \text{we moved forward in all sequences except the first that get gap} \\
\vdots \\
\end{array} \right. 
\]

Total: $2^n - 1$ options to choose the maximum from, when $n$ is the number of sequences.
• CLUSTALω (heuristic way):
  – Take all possible sequences pairs, and run pairwise alignment for each pair.
  – Choose the best pair and find the sequence that is the closest to them. We do so by placing the sequence against the pair, and defining alignment between sequence and profile \( \sigma(P, b) \)

3 From MSA to phylogenetic tree:

If the topology \((T)\) and the branches length \((\tau)\) are given, we can estimate the tree.

3.1 Parsimony - Theory

The best tree is the tree with the minimum changes. i.e. the placement for the internal nodes that requires the minimum changes for mutation in the tree.

\[
Parsimony(Msa, Tree) = \sum_i Parsimony(\tilde{A}_i, T)
\]

\[
Parsimony(\tilde{A}_i, T) = \min_{A_{i+1} \ldots A_{2i-1}} (\text{placement for the internal nodes}) (\sum_{(j,k) \in T} 1\{A^j_i \neq A^k_i\})
\]

for finding the minimum, need to search in exponential number of placements - \(\|\Sigma\|^{(n-1)}\)

\[
\hat{T} = \text{argmin}_T (Parsimony(M, T))
\]

In dynamic programing we split the tree to subtrees - Divide and rule
Li - the leaves in the subtree of i. (for the general fans: Lij the leaves in the subtree of i when we look from j)

\[ L_j = L_k \cup L_l \]

We want to find the two sequences a and b, such that will minimize: 

\[ \text{Score}_r[a] + \text{Score}_j[b] + 1 \{a \neq b\} \]

\[ \text{Parsimony} = \text{parsimony}(L_r) + \text{parsimony}(L_j) + 1 \{r \neq j\} \]

### 3.2 Parsimony - Algorithm

1. Start from node r
2. go post_order
   - If j is leaf: \[ S_j[a] = \begin{cases} 0 & A^j = a \\ \infty & \text{else} \end{cases} \]
   - else: \[ S_j[a] = \min_{b,c} [S_k[b] + S_l[c] + 1\{a \neq b\} + 1\{a \neq c\}] = \min_b [S_k[b] + 1\{a \neq b\}] + \min_c [S_l[c] + 1\{a \neq c\}] \] - the last step is since k,l are independent forgiven a.
3. finally for r. \( j \in \text{Neighbor}(r) \)
   \[ \text{Parsimony}(A_i, T) = \min_{a,b} [S_r[a] + S_j[b] + 1\{a \neq b\}] \]

RunTime:

- Number of cells to fill: \( \|s\| = \|\Sigma\| \times O(n) \)
- Time to fill each cell: \( O(\|\Sigma\|) \)
- Width of the alignment: \( N \)
- Number of sequences: \( n \)
- Total: \( O(N \times n \times \|\Sigma\|^2) \)
Problem: we still don't know how to find good trees, because they are too many: $O(n!)$