Multiple Sequence Alignment: Multi-dimensional Dynamic Programming

Zhiping Weng
Boston University

Biological Motivation

• *Compare a new sequence with the sequences in a protein family.* Proteins can be categorized into families. A protein family is a collection of homologous proteins with similar sequence, 3-D structure, function, and/or similar evolutionary history.

• *Gain insight into evolutionary relationships.* By looking at the number of mutations that are necessary to go from an ancestor sequence to a current sequence, one can get an estimate for the time when the two sequences diverged in the evolutionary history.
Problem Statement

What are the conserved regions among a set of sequences over the same alphabet?

```
12345678   Position Index
EMQPLL     Sequence 1
DMLRLL      Sequence 2
NMKLILL     Sequence 3
DMPPVILL    Sequence 4
DMLL        Consensus sequence
```

Similarity Measurement: SP-score

Sum of Pairs (SP)-score is the similarity score among amino acids (or bases) at a particular position of a multiple sequence alignment. 

\( s(\alpha_i, \alpha_j) \)

The gap-gap alignment has 0 similarity & distance score: 

\( s(\cdot,\cdot) = 0 \)

SP-score \( (\alpha) = \sum_{i<j} s(\alpha_i, \alpha_j) \)

\( \alpha \) is the collection of amino acids at a position of alignment.

SP-score \( (P,R,\cdot,\cdot,P) = s(P,R) + s(P,\cdot) + s(P,P) + s(R,\cdot) + s(R,P) + s(\cdot,P) \)
Given a multiple sequence alignment with **Sum of Pairs (SP-score)**, we may compute the score of each position of the alignment and then add all the position scores to get the total score of the whole alignment. Or, we may compute the score for each induced pairwise alignment and add these scores.

If we have N sequences, the number of pairs is \( N \times (N-1)/2 \)
Multi-Dimensional Dynamic Programming

Similarity matrix: k-dimensional array $S$ of length $N_1 + 1$ ($N_i$ is the length of sequence $i$) in each dimension to hold the optimal similarity scores for multiple alignments of prefixes of sequences:

$S[i_1, i_2, i_3, ..., i_k]$ holds the score of the optimal alignment involving sequence1[1...$i_1$], sequence2[1...$i_1$], sequence3[1...$i_2$]..., and sequence k[1...$i_k$].

The computational complexity of such a program is exponential: $O(k^22^kN^k)$. There are $N^k$ entries in the similarity matrix need to be filled in.
All 3 pair-wise projections of the 3-D alignment

Multiple Alignment of 3 Sequences: Dynamic Programming
We have 7 possibilities ($2^3-1$) to choose the maximum similarity
The fewer entries in the similarity matrix we need to calculate, the more time we can save from dynamic programming.

Using pairwise projections for obtaining bounds on the “volume” the dynamic programming algorithm needs to explore.

This method works when:

1. Sum-of-pairs scores (SP-scores) are used
2. Sequences are similar
3. Gap penalties are not too small

The optimal alignment path is contained in a "polyhedron" close to the main diagonal. Here, a polyhedron is a solid formed by plane faces, or more complicated 2-dimensional surfaces. For better visualization, the polyhedron's shadows are displayed. While visiting a node and looking for the minimum along all the incoming edges, we can ignore those edges that are "coming from outside the polyhedron", as in the top part the inset. On its top-left side, the cube is "covered" by the polyhedron. The edges 1, 2, 3, 6 and 7 are coming from the inside, and edges 4 and 5 can be ignored.
The Carrillo-Lipman Method

$\alpha$: Any multiple alignment involving $K$ sequences $\chi_1, \chi_2, \ldots, \chi_K$

$\alpha^*$: the optimal multiple alignment involving $K$ sequences $\chi_1, \chi_2, \ldots, \chi_K$

$\alpha_{ij}$: 2-D projection of $\alpha$ related to sequences $\chi_i$ and $\chi_j$

$\alpha_{ij}^*$: 2-D projection of $\alpha^*$ related to sequences $\chi_i$ and $\chi_j$

$\beta_{ij}$: the optimal pairwise score between $\chi_i$ and $\chi_j$

score() and SP-score() denote distance scores.

$$SP\text{-score}(\alpha) = \sum_{1 \leq i < m \leq K} \text{score}(\alpha_{ln})$$

The Carrillo-Lipman Method: determine cells that belong to the polyhedron

$$SP\text{-score}(\alpha^*) = \sum_{1 \leq i < m \leq K} \text{score}(\alpha^*_{ln})$$

$$SP\text{-score}(\alpha) = \sum_{1 \leq i < m \leq K} \text{score}(\alpha_{ln})$$

$\beta_{ij} \leq \text{score}(\alpha_{ij})$

$\beta_{ij} \leq \text{score}(\alpha_{ij}^*)$

$$\sum_{1 \leq i < m \leq K} \beta_{ln} \leq SP\text{-score}(\alpha^*) \leq SP\text{-score}(\alpha)$$

For some $i, j$ projections (BUT NOT ALL),

score($\alpha_{ij}$) $\leq$ score($\alpha_{ij}^*$) can be true.
The Carrillo-Lipman Method: determine cells that belong to the polyhedron

\[ \sum_{1 \leq i < m \leq K} \text{score}(\alpha_{lm}^*) \leq \sum_{1 \leq i < m \leq K} \text{score}(\alpha_{lm}) \]

\[ \therefore \text{For any arbitrary projection } i, j: \]

\[ \text{score}(\alpha_{ij}^*) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} \text{score}(\alpha_{lm}^*) \leq \text{score}(\alpha_{ij}) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} \text{score}(\alpha_{lm}) \]

\[ \text{score}(\alpha_{ij}^*) \leq \text{score}(\alpha_{ij}) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} (\text{score}(\alpha_{lm}) - \text{score}(\alpha_{lm}^*)) \]

\[ \therefore \beta_{lm} \leq \text{score}(\alpha_{lm}^*) \]

\[ \therefore \text{score}(\alpha_{ij}^*) \leq \text{score}(\alpha_{ij}) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} (\text{score}(\alpha_{lm}) - \beta_{lm}) \]

The Carrillo-Lipman Method

\[ \text{score}(\alpha_{ij}^*) \leq \text{score}(\alpha_{ij}) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} (\text{score}(\alpha_{lm}) - \beta_{lm}) \] means:

The optimal multiple alignment path always falls within the bound:

\[ \text{score}(\alpha_{ij}) + \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} (\text{score}(\alpha_{lm}) - \beta_{lm}) = \text{SP-score}(\alpha) - \sum_{1 \leq i < m \leq K \atop l, m \neq i, j} \beta_{lm} \]

A cell is within the polyhedron only if in all projections it has a score lower than the above upper bound. Therefore we can ignore all cells outside the bound (polyhedron). \( \alpha \) can be obtained from a heuristic algorithm. The closer \( \alpha \) is from \( \alpha^* \), the tighter the bound is, and the fewer cells are within the polyhedron, and the greater enhancement the Carrillo-Lipman algorithm achieves.