

BCB410 – Course Notes

Title: Sequence Alignment (Needleman-Wunsch, Smith-Waterman)

These notes are based on lecture notes taken by Gabe Musso for CSC 2427

Topics:

1. **Needleman-Wunsch** (Global Alignment)
2. **Maximum Contiguous Subsequence Sum**
3. **Smith-Waterman** (Local Alignment)

Background: Importance of Sequence Alignment

Comparative analysis is the backbone of evolutionary biology. It was phenotypic variation which allowed Darwin to compose his theory of natural selection. That theory rests on the fact that transfer of the genetic code from parent to progeny does not exist without change. It is these changes in genetic sequence which allow for divergence of species, and thus provide a backdrop for natural selection. Just as comparative analysis was key for evolutionary biology, sequence alignment is the cornerstone of modern bioinformatics. Rapid and automated sequence analysis facilitates everything from functional classification & structural determination of proteins, to studies of genetic expression and evolution.

1. Needleman-Wunsch (Global Alignment)

Dynamic programming algorithms find the best solution by breaking the original problem into smaller sub-problems and then solving. The Needleman-Wunsch algorithm is a dynamic programming algorithm for optimal sequence alignment (Needleman and Wunsch, 1970). Basically, the concept behind the Needleman-Wunsch algorithm stems from the observation that any partial sub-path that tends at a point along the true optimal path must itself be the optimal path leading up to that point. Therefore the optimal path can be determined by incremental extension of the optimal sub-paths. In a Needleman-Wunsch alignment, the optimal path must stretch from beginning to end in both sequences (hence the term ‘global alignment’).

In order to perform a Needleman-Wunsch alignment, a matrix is created which allows us to compare the two sequences. The score $M(i, j)$ for every cell depends on the three cells corresponding to either or both sequence having 1 less letter (i.e. cells $M(i-1, j)$, $M(i, j-1)$ and $M(i-1, j-1)$). It is calculated as follows:

$$M(i, j) = \text{MAX} (M_{i-1, j-1} + S(A_i, B_j) \\ M_{i-1, j} + \text{gap} \\ M_{i, j-1} + \text{gap})$$

where gap is the gap penalty and the function S returns the score/penalty for matching the two corresponding letters. Once we have computed this score for every cell, we must do a “traceback”, that is to determine the actual set of operations that lead to the score.

Because when computing the score of a cell we took a max over three numbers, on the traceback we go to the location of the highest – going sideways or up corresponds to gaps, and going along the diagonal corresponds to a match. This algorithm performs alignments with a time complexity of $O(mn)$ and a space complexity of $O(mn)$.

Example:

Find the best alignment of these two sequences:

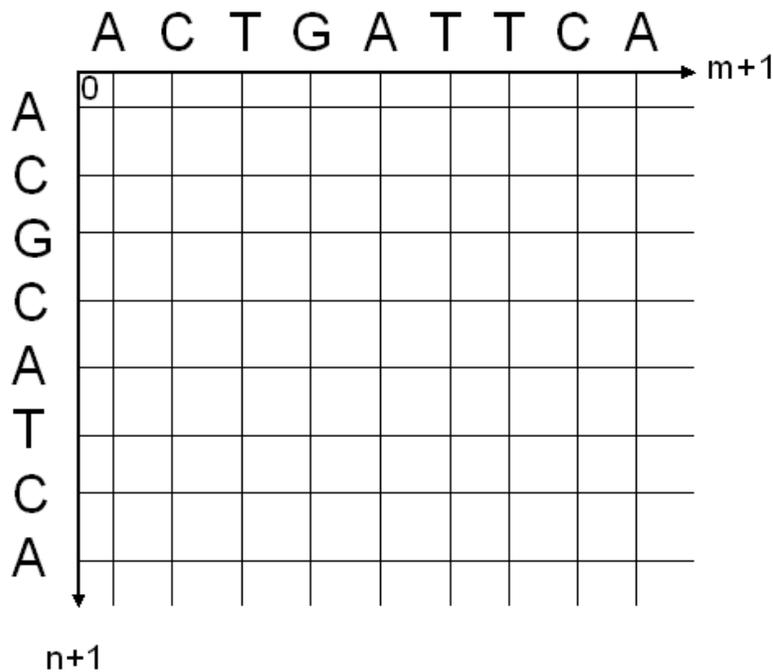
ACTGATTCA
ACGCATCA

Using -2 as a gap penalty, -3 as a mismatch penalty, and 2 as the score for a match.

Solution:

Step 1: Draw the matrix

For 2 sequences (length m and length n) what size scoring matrix is needed for their alignment? Grid dimensions must be $(m+1) \times (n+1)$. Think of each increment as a division of the sequence members:



Step 2: Assign scores

		A	C	T	G	A	T	T	C	A
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
A	-2	2	0	-2	-4	-6	-8	-10	-12	-14
C	-4	0	4	2	0	-2	-4	-6	-8	-10
G	-6	-2	2	1	4	2	0	-2	-4	-6
C	-8	-4	0	-1	2	1	-1	-3	0	-2
A	-10	-6	-2	-3	0	4	2	0	-2	2
T	-12	-8	-4	0	-2	2	6	4	2	0
C	-14	-10	-6	-2	-4	0	4	2	6	4
A	-16	-12	-8	-4	-5	-2	2	1	4	8

Step 3: Trace back

The optimal path is traced beginning from the lower right-hand corner

		A	C	T	G	A	T	T	C	A
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
A	-2	2	0	-2	-4	-6	-8	-10	-12	-14
C	-4	0	4	2	0	-2	-4	-6	-8	-10
G	-6	-2	2	1	4	2	0	-2	-4	-6
C	-8	-4	0	-1	2	1	-1	-3	0	-2
A	-10	-6	-2	-3	0	4	2	0	-2	2
T	-12	-8	-4	0	-2	2	6	4	2	0
C	-14	-10	-6	-2	-4	0	4	2	6	4
A	-16	-12	-8	-4	-5	-2	2	1	4	8

Result:

This analysis yielded the following alignment:

```
ACTG-ATTCA
||  ||  ||
AC-GCAT-CA
```

The alignment score is equal to the value in the lower right-hand corner of the matrix (8).

2. From Global to Local Similarity: Maximum Contiguous Subsequence Sum

When aligning two very large sequences, it is often useful to determine the locations of high similarity regions, even if there is no additional similarity inbetween the sequences. Now that we know how to calculate the *global* alignments, how can we find all local high-scoring hits, or *local* alignments above a given threshold for two large sequences? The answer is related to a programming “pearl”, the ‘Maximum Contiguous Subsequence Sum’ (MSS).

Problem:

Given integers A_1, A_2, \dots, A_N find (and identify the sequence corresponding to) the maximum value of:

$$\sum_{k=1}^j A_k$$

Solution:

Can be solved in time complexity of ‘n’.

```
mss(A) {
    max = 0;
    sum = 0;
    for (i=1; i ≤ n; i+1) {
        sum = sum + A[i];
        if (sum > max)
            max = sum;
        if (sum < 0)
            sum = 0;
    }
}
```

```

    }
    return max;
}

```

Analysis:

When a subsequence occurs which has a negative sum, the subsequence which will be examined next can begin after the first subsequence (the one that produced the negative sum). Basically, the entire first subsequence is regarded as not having a starting point which will generate a positive sum. For example, consider this set of numbers:

4, 6, -2, 2, -14, 9

Some sums are positive (4, 4+6, 4+6+(-2), 4+6+(-2)+2) but the sum of the first 5 terms (4+6+(-2)+2-14) is negative. Therefore it follows logically that any sequence starting between the 4 and -14 and ending with the -14 will have a negative sum.

The maximum contiguous subsequence sum searches exactly for the highest scoring local area. We now generalize this approach for sequence alignment; the only change is we do the above algorithm in two dimensions!

3. Smith-Waterman (Local Alignment)

Over a decade after the initial publication of the Needleman-Wunsch algorithm, a modification was made to allow for local alignments (Smith and Waterman, 1981). In this adaptation, the alignment path does not need to reach the edges of the search graph, but may begin and end internally. In order to accomplish this, 0 was added as a term in the score calculation described by Needleman and Wunsch.

Recall that for global alignments the value at any point is:

$$M(i, j) = \text{MAX} (M_{i-1, j-1} + S(A_i, B_j) \\ M_{i-1, j} + \text{gap} \\ M_{i, j-1} + \text{gap})$$

However for local alignments:

$$M(i, j) = \text{MAX} (M_{i-1, j-1} + S(A_i, B_j) \\ M_{i-1, j} + \text{gap} \\ M_{i, j-1} + \text{gap} \\ 0)$$

The implication of this is that there are no values below zero in a local alignment scoring matrix, and the reason for the zero is exactly the same as in the MSS problem above.

Example:

Find the best local alignment between these two sequences:

ATGCATCCCATGAC
TCTATATCCGT

Using -2 as a gap penalty, -3 as a mismatch penalty, and 2 as the score for a match.

Solution:

Traceback begins at the highest value (which is also the alignment score).

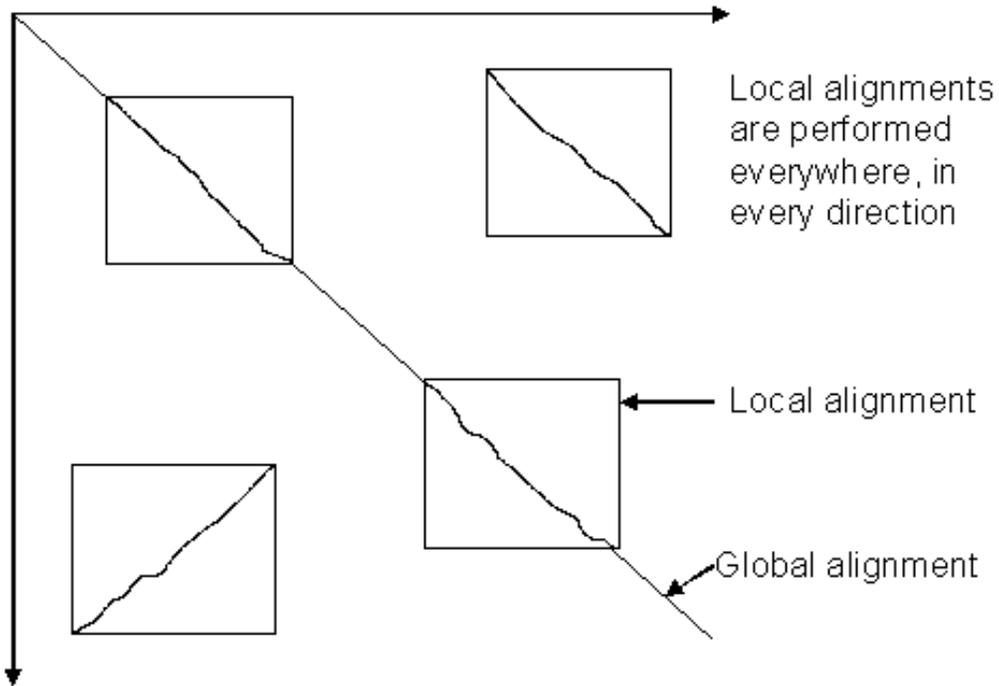
		A	T	G	C	A	T	C	C	C	A	T	G	A	C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	2	0	0	0	2	0	0	0	0	2	0	0	0
C	0	0	0	0	2	0	0	4	2	2	0	0	0	0	2
T	0	0	2	0	0	0	0	2	1	0	0	2	0	0	0
A	0	2	0	0	0	2	0	0	0	0	2	0	0	2	0
T	0	0	4	2	0	0	2	0	0	0	0	4	2	0	0
A	0	2	0	0	0	2	0	0	0	0	2	0	0	2	0
T	0	0	4	2	0	0	4	2	0	0	0	4	0	0	0
C	0	0	2	0	4	0	0	6	4	2	0	0	0	0	2
C	0	0	0	0	2	0	0	4	8	6	4	2	0	0	2
G	0	0	0	2	0	0	0	2	6	5	3	1	4	2	0
T	0	0	2	0	0	0	2	0	4	3	2	5	3	1	0

Which yields the alignment:

ATCC
| | | |
ATCC

With an alignment score of 8.

Local alignments are performed everywhere possible along two sequences.



When trying to find the best local alignments corresponding to a global alignment, a sub-matrix is created with the highest positive score for all alignments above a given threshold. Therefore, the same thing that the MSS was doing on a linear matrix, the Smith-Waterman alignment does on a rectangular matrix.