CSC2427 – Course Notes

Title: Alignment (Needleman-Wunsch, Smith-Waterman)
Date: Feb 1, 2006
Scribe: Gabe Musso

Topics:

1. **Needleman-Wunsch** (Global Alignment)
2. **Hirschberg's Algorithm**
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)

In the last class we discussed how dynamic programming can be used to solve alignment problems (recall that dynamic programming algorithms finds the best solution by breaking the original problem into smaller sub-problems and then solving). The Needleman-Wunsch algorithm is an application of a best-path strategy (dynamic programming) used to find 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'). Recall that the score at any position in a global alignment matrix is:

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

***when tracing back the alignment path, horizontal and vertical movement is a gap, and diagonal movement is a match*

In order to perform a Needleman-Wunsch alignment, a matrix is created which allows us to compare the two sequences. The score as determined through use of the above calculation is placed in the corresponding cell. 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:

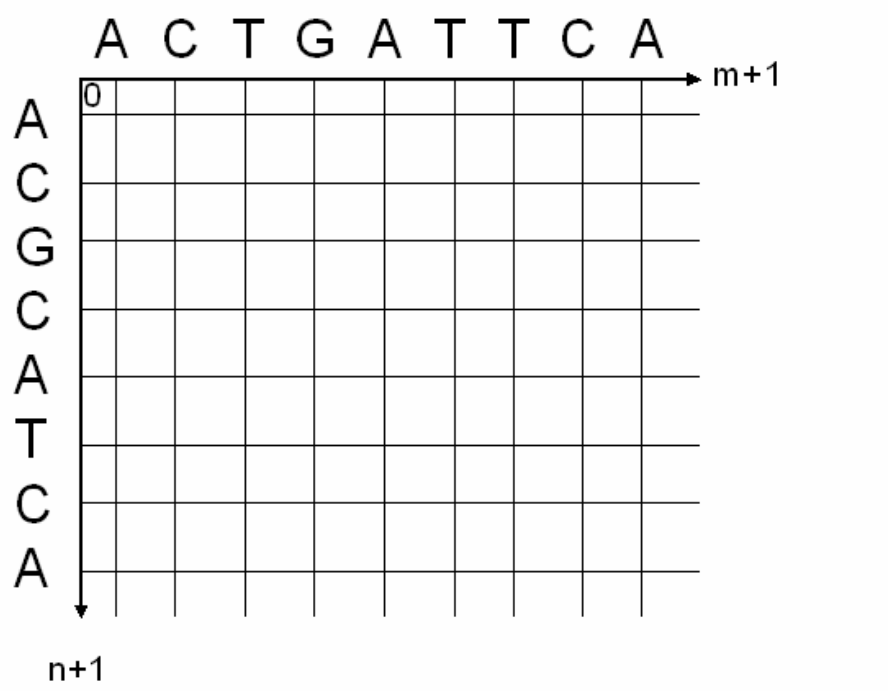
```
ACTGATTCA
ACGCATCA
```

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) × (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).

```
Score = (AA) + (CC) + (T-) + (GG) + (-C) +
        (AA) + (TT) + (T-) + (CC) + (AA)
      = 2+2-2+2-2+2+2-2+2+2
      = 8
```

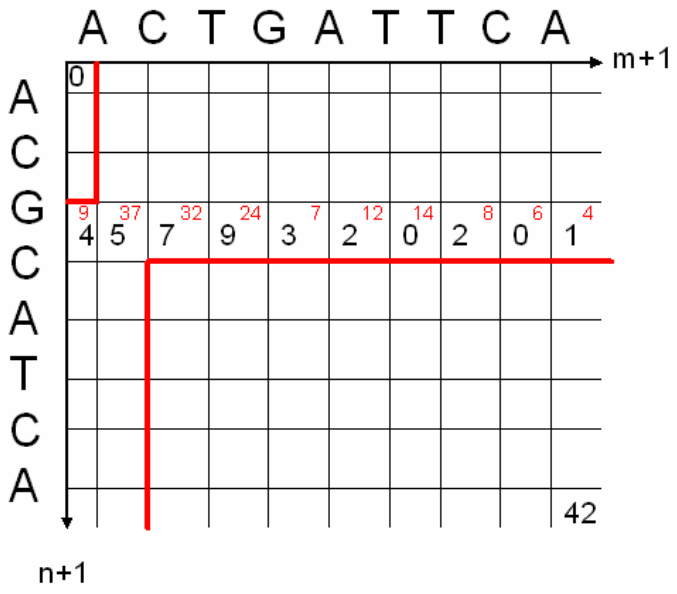
2. Hirschberg's Algorithm

While the Needleman-Wunsch algorithm works well for sequence alignment, its space complexity (O(mn)) limits the size of sequences it can align. Hirschberg's algorithm uses a divide and conquer strategy to decrease the space requirement. Specifically, for two sequences (m and n) the first string is cut (m1 and m2) and the second string is cut in a corresponding place (into n1 and n2). The alignment is then solved recursively on m1 and n1, and m2 and n2. It is important to note that the two sub-strings (i.e. n1 and n2) need not have the same length.

Example 1:

With respect to sequence alignment, the objective is to take the sum at given intervals and use the value corresponding to the alignment score as the dividing point.

In this figure the sum up to the dividing line (37 + 5) adds up to the alignment score.



Example 2:

What is the sequence of recursive calls on the following sequence alignment when using Hirschberg's algorithm:

```
ACTG
ACTT
```

The alignment is:

```
s1: ACTG
    |||
s2: ACTT
```

Solution:

```
s1[0..3] : s2[0..3]
   s1[0..1] : s2[0..1]
      s1[0..0] : s2[0..0]
      s1[1..1] : s2[1..1]
   s1[2..3] : s2[2..3]
      s1[2..2] : s2[2..2]
      s1[3..3] : s2[3..3]
```

3. Relating Local Alignments to Global Alignments

When aligning two very large sequences, it is often useful to determine the locations of high similarity regions. 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 $A_1, A_2, ..., A_N$ 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.

4. 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). Today, the Smith-Waterman alignment algorithm is the one used by the Basic Local Alignment Search Tool (BLAST) which is the most cited resource in biomedical literature. 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:

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

However for local alignments the score calculation becomes:

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

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
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

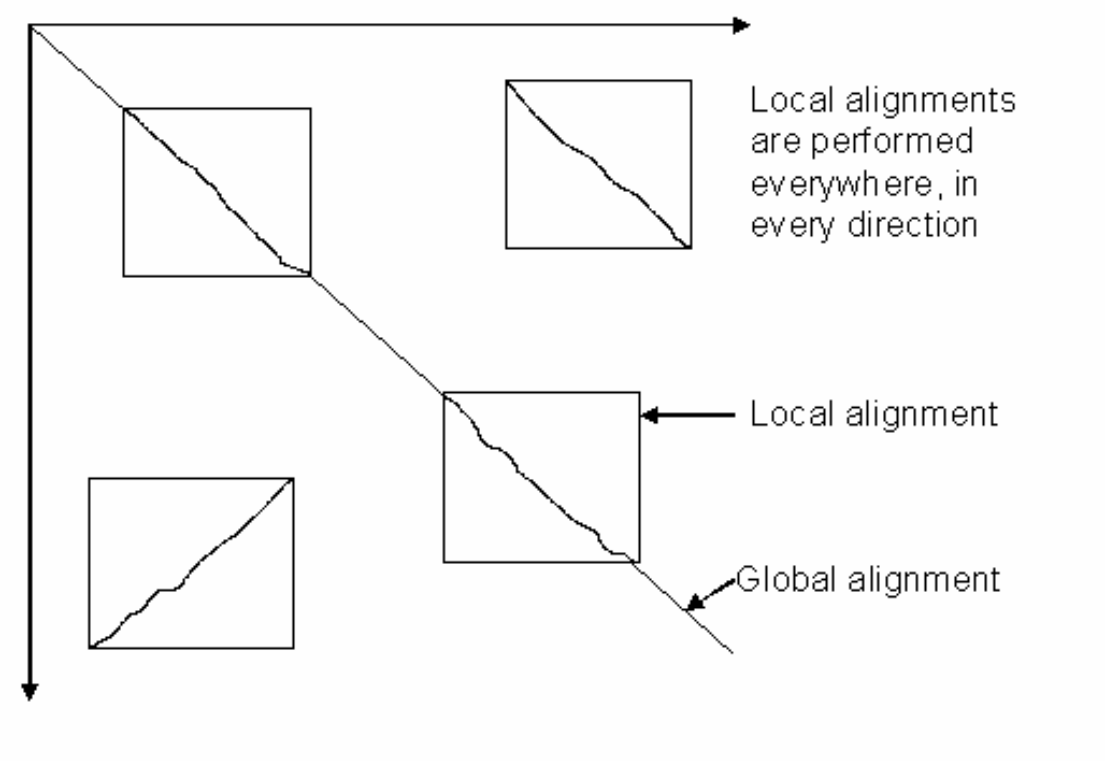
Which yields the alignment:

```
ATCC
||||
ATCC
```

With an alignment score of 8.

```
Score = (AA) + (TT) + (CC) + (CC)
      = 2 + 2 + 2 + 2
      = 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.