Introduction to Dynamic Programming

The sequence alignment problem

Wilson Leung 08/2015

Outline

- Overview of the sequence alignment problem
- Calculate the optimal global alignment
- Characteristics of dynamic programming algorithms
- Calculate the optimal local alignment

Learning objectives

- Understand the theory behind sequence alignment
- Become a better informed user of NCBI BLAST
- This presentation will not cover:
  - The BLAST algorithm
  - Parameter optimizations
  - Statistics for similarity searches (Karlin-Altschul theory)


Design goals

- Generate an alignment between two sequences
- Identify the “best” (most parsimonious) alignment
- Generate the best alignment “quickly”

Strategy #1: Visual inspection

Query: ATTACCAG

Subject: ATCACCA

Sequences must have high percent identity

Applications:
- PAM scoring matrix (align sequences with >= 85% identity)
- Align mononucleotide runs during sequence improvement

Strategy #2: Enumerate all alignments

Guaranteed to find the best alignment

Does not scale
- Combinatorial explosion
- Two 300 bp sequences have ~10^179 possible alignments (Eddy 2004)

Brute-force algorithm
- Establish baseline performance and test cases
- Identify patterns in the problem space
Apply the brute force algorithm to a single column of the alignment

<table>
<thead>
<tr>
<th>Homologous</th>
<th>Not homologous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query: A</td>
<td>Subject: A</td>
</tr>
</tbody>
</table>

- Three possible alignments for two 1 bp sequences
  - Query length ($M$) = 1; Subject length ($N$) = 1
- Only two biological interpretations:
  - A in the query is homologous to A in the subject
  - A in the query is not homologous to A in the subject

Six possible relationships between the query and subject for $M=2$, $N=2$

<table>
<thead>
<tr>
<th>2 aligned bases</th>
<th>1 aligned base</th>
<th>0 aligned bases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query: AT-AT</td>
<td>Subject: AT-AT</td>
<td>AT--AT--</td>
</tr>
<tr>
<td>AT-AT</td>
<td>AT-AT</td>
<td>AT-AT</td>
</tr>
<tr>
<td>AT--AT</td>
<td>AT--AT</td>
<td>AT--AT</td>
</tr>
<tr>
<td>AT-AT</td>
<td>AT--AT</td>
<td>AT--AT</td>
</tr>
<tr>
<td>AT--AT</td>
<td>AT--AT</td>
<td>AT--AT</td>
</tr>
<tr>
<td>AT-AT</td>
<td>AT--AT</td>
<td>AT--AT</td>
</tr>
</tbody>
</table>

Each color denotes a different evolutionary relationship

Observations from the brute force alignment strategy

- Many of the possible alignments are redundant
- Imply the same evolutionary relationship

- Large number of possible alignments
- 13 possible alignments for sequences of length 2

- Can ignore many possible alignments
  - Many are suboptimal compared to the best alignment

Strategy #3: Dot plot

- Cell position ($i, j$):
  - $i$ = Query position (x-axis)
  - $j$ = Subject position (y-axis)
- Draw a dot at $(i, j)$ if the two bases are identical
- Connect the dots to make a line (alignment)
- Level of noise depends on repeat density
  - Use longer words and higher cutoff scores to reduce noise

Assessment of the three sequence alignment strategies

- Infeasible to examine all possible alignments
  - Need to reduce the search space

- Only a small subset of alignments are “interesting”
  - Many alignments are redundant

- Connect the dots in the dot plot to create an alignment
  - Consider the cumulative levels of similarity

The optimal alignment is composed of smaller optimal alignments

Query: AT  Subject: AT

- Only the best alignment at each position could be part of the final optimal alignment
Partition the alignment problem into smaller subproblems.

Assume the query and subject sequences are the same.

Three different ways to reach cell \((i, j)\) in the alignment matrix.

Recursive definition for the optimal cumulative alignment score \(S(i, j)\):

Determine the best way to reach cell \((i, j)\) if it were part of the optimal alignment.

Use the maximum score at each cell to eliminate entire branch of suboptimal alignments.

Construct a scoring system to measure similarity between two sequences:

- Scoring system for the aligned state: \(\sigma\)
  - \(\sigma(a, b)\) = Score for aligning \(a\) in query with \(b\) in subject
  - \(\sigma(A, A)\) = Bonus for aligning \(A\) in query with \(A\) in subject
  - \(\sigma(A, T)\) = Penalty for aligning \(A\) in query with \(T\) in subject
- Penalty for adding a gap: \(\gamma\)

More sophisticated scoring systems take transitions, transversions, affine gap penalty into account:

Cumulative score $S(i, j)$ encapsulates the alignment decisions up to position $(i, j)$

- All potential optimal alignments that go through cell $(i, j)$ have the same ancestry
- Re-use the cumulative alignment score (memoization)
- Gaps are described by the cumulative score
- Do not affect the coordinates of the alignment matrix
- Do not know the optimal alignment until we complete the entire alignment matrix
- Optimal alignment has the highest cumulative score

Needleman-Wunsch algorithm (global alignment) (Query length: M; Subject length: N)

- Construct a $(M+1) \times (N+1)$ matrix
- Extra column and row = gaps at the beginning of the alignment
- Fill in the cells in the first row and first column with the cumulative gap costs
- Calculate the maximum score for subsequent cells $(i, j)$
- Keep track of the decision that leads to the maximum score ($S$)

$$S(i, j) = \max \left\{ S(i-1, j-1) + \sigma(a,b), S(i-1, j) + \gamma, S(i, j-1) + \gamma \right\}$$

Initialize the alignment matrix

(Match = +5; Mismatch = -2; Gap = -6)

Calculate the possible scores for the cell at position $(1,1)$

$$S(1,1) = \max \left\{ \sigma(T,T) + \gamma, S(0,1) + \gamma, S(1,0) + \gamma \right\}$$

Calculate the optimal score for the cell at position $(1,1)$

$$S(1,1) = 5$$

Calculate the possible scores for the cell at position $(2,1)$

$$S(2,1) = \max \left\{ S(1,0) + \gamma, S(1,1) + \gamma, S(2,0) + \gamma \right\}$$

Calculate the optimal score for the cell at position (2, 1)

\[
S(2, 1) = \max \{ \\
-6 + (-2) = -8 \\
5 + (-6) = -1 \\
-12 + (-6) = -18 \\
\} \\
S(2, 1) = -1
\]

Alignment matrix after two iterations
(Match = +5; Mismatch = -2; Gap = -6)

Calculate the optimal score for the cell at position (3, 1)

\[
S(3, 1) = \max \{ \\
-12 + (-2) = -14 \\
-1 + (-6) = -7 \\
-18 + (-6) = -24 \\
\} \\
S(3, 1) = -7
\]

Matrix after three iterations
(Match = +5; Mismatch = -2; Gap = -6)

Calculate the optimal score for the cell at position (1, 2)

\[
S(1, 2) = \max \{ \\
-6 + (+5) = -1 \\
-12 + (-6) = -18 \\
5 + (-6) = -1 \\
\} \\
S(1, 2) = -1
\]

Complete alignment matrix
(Match = +5; Mismatch = -2; Gap = -6)
Use **traceback** to recover the optimal alignment

- Start from the cell within the last row and last column that has the highest score
- **Recall the step (color)** that leads to this optimal score
- Report this step in the alignment output
- All the alignment decisions have already been made
- Repeat until we reached the beginning of the sequence

- Two options if multiple paths produce the same score
  - Report only one of the paths (pick arbitrarily)
  - Report all paths with the optimal score

---

**Calculate the optimal score for the cell at position (5, 3)**

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-2</td>
<td>-8</td>
</tr>
<tr>
<td></td>
<td>+5</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>-14</td>
</tr>
</tbody>
</table>

\[
S(5,3) = \max \{ \\
-2 + (+5) = 3 \\
2 + (-6) = -4 \\
-8 + (-6) = -14 \\
\} \\
S(5,3) = 3
\]

(Match = +5; Mismatch = -2; Gap = -6)

---

**Traceback must follow the steps that produce the optimal cumulative global alignment score**

---

The Needleman-Wunsch algorithm is an example of a **dynamic programming** algorithm

- Problem must satisfy two criteria:
  - **Optimal substructure**: Optimal solution to the complete problem is composed of optimal solutions to the subproblems
  - **Overlapping problems**: Re-use the results for the subproblems (e.g., lookup table)
- Many bioinformatics problems satisfy these criteria
  - Sequence alignment, gene prediction, RNA-folding

Smith-Waterman algorithm (local alignment)
(Query length: M; Subject length: N)

- Three changes to the Needleman-Wunsch algorithm:
  - The minimum score for a cell is zero
  - Initiate a new alignment when the cumulative score is negative
  - Begin traceback from the cell within the entire matrix that has the highest score
  - Terminate traceback when the score is zero

\[
S(i,j) = \max \begin{cases} 
S(i-1,j-1) + \sigma(a,b) \\
S(i-1,j) + \gamma \\
S(i,j-1) + \gamma \\
0 
\end{cases}
\]

Global versus local alignments

- Global alignment
  - Optimal alignment along the entire length of two sequences
  - Compare protein sequences to identify orthologs

- Local alignment
  - Optimal alignment between parts of two sequences
  - Identify conserved domains within protein sequences

- Glocal (semi-global) alignment
  - Optimal global alignment for one sequence; optimal local alignment for the other sequence
  - Map a coding exon against a genomic sequence

Techniques to improve the performance of sequence alignment

- **Time and space complexity:** $O(MN)$
- **Double the size** of the two sequences leads to a **four-fold increase** in the amount of time and space required
- **Reduce memory requirement**
- **Fill the matrix in parallel (SIMD, CUDA)**
- **Find high-scoring instead of the best alignment**

Questions?

Dynamic programming matrix:

```
0  1  2  3  4  5  6  7  8
T  0  1  1  1  1  1  1  1  1
C  1  1  2  2  2  2  2  2  2
G  2  2  2  3  3  3  3  3  3
A  3  3  3  3  4  4  4  4  4
T  4  4  4  4  4  4  4  4  4
G  5  5  5  5  5  5  5  5  5
C  6  6  6  6  6  6  6  6  6
A  7  7  7  7  7  7  7  7  7
```

Optimum alignment score: 7

T ---- T C A T A
T G C T C G T A