Algorithms for Sequence Alignment

Longshu Yang
December 10th, 2014
Center for Quantitative Biology

Content

- Introduction
  - Why do we need sequence alignment?
  - A brief history of sequence alignment

- Pairwise sequence alignment
  - Dynamic programming
  - Scoring matrix
  - Global, local, overlap and repeated alignment
  - More complex models for pairwise alignment
  - A brief introduction of BLAST

- Multiple sequence alignment (MSA)
  - Multidimensional programming v.s. progressive alignment
  - A brief introduction of Clustal W
Introduction

- Why do we need sequence alignment?
- A brief history of sequence alignment

Why do we need sequence alignment?

- Fundamental issue in bioinformatics
- For annotation (similarity)
  - Functional gene annotation
  - Conserved domain detection
- For phylogenetic purpose (distinction)
  - Construction of Phylogenetic tree
  - Detection of gene mutation
- Homology inference (similarity & distinction)
A brief history of sequence alignment

- Scoring Scheme
- Specific Algorithm
- Sequence Alignment
- Data Growth

Data Growth

- Kary Mullis develops PCR technique (1983)
### Scoring Scheme

#### PAM Matrix

![PAM Matrix Image](image)

Margaret Dayhoff (1925-1983) creates PAM matrix (1978)

#### BLOSUM Matrix

![BLOSUM Matrix Image](image)

Steven Henikoff creates BLOSUM matrix (1992)

### Specific Algorithm

#### Pairwise Alignment

- Dynamic programming
  - Needleman-Wunsch global alignment algorithm (1970)
  - Smith-Waterman local alignment algorithm (1981)

- Heuristic alignment
  - BLAST seed extension algorithm (Altschul et al., 1990)

#### Multiple Alignment

- Multidimensional dynamic programming (Sankoff and Cedergren, 1983)

- Progressive alignment
Pairwise sequence alignment

- Dynamic programming
- Scoring matrix
- Global, local, overlap and repeated alignment
- More complex models for pairwise alignment
- A brief introduction to BLAST

Scoring Matrix

Point Accepted Mutation Matrix
- Including 1572 mutations in 71 groups of proteins with similarity more than 85%
- 1 PAM = 1 mutation/100aa
- Iteration: $PAM^n = PAM1^n$

Blocks Substitution Matrix
- 504 related protein groups, 2205 blocks, 2 clusters with similarity of 62% and 80%
- Block: ungapped aligned regions

<table>
<thead>
<tr>
<th>Matrix</th>
<th>PAM 1</th>
<th>PAM 120</th>
<th>PAM 250</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOSUM 80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOSUM 62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOSUM 45</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Matrices used in BLAST

Source: NCBI
Point Accepted Mutation Matrix

Relative Mutability

Values according Dayhoff (1978) The value for Ala has been arbitrarily set at 100.

PAM1, all elements multiplied by 10000
Point Accepted Mutation Matrix

- Transfer probability matrix to scoring matrix
  For PAMn the score is:
  \[ s_{ij} = \log \frac{M_{ij}^n}{f_i} \]

A positive score \((s > 0)\) characterizes the accepted mutations.
A negative score \((s < 0)\) characterizes the unfavorable mutations.

BLOcks SUBstitution Matrix

Frequency of occurrence:
\[ q_{ij} = \frac{f_{ij}}{\sum_{i=1}^{3} \sum_{j=1}^{3} f_{ij}} \]

Expected frequency:
\[ e_{ij} = 2 p_i p_j \] 
\[ p_i = q_i + \frac{1}{2} \sum_{j \neq i} q_{ij} \]

\[ s_{ij} = 2 \log_2 \frac{q_{ij}}{e_{ij}} \]
Scoring matrix

- Penalty matrix for nucleic acid sequences:
  - Hamming matrix:
    $$d_H(a, b) = \begin{cases} 
    0, & \text{if } a = b, \\
    1, & \text{otherwise.}
    \end{cases}$$

- BLASTN matrix:
  $$\begin{pmatrix}
  \text{Similarity} & \text{Reward} & \text{Penalty} \\
  99\% (PAM1) & 1 & -3 \\
  95\% (PAM5) & 1 & -2 \\
  75\% (PAM30) & 1 & -1
  \end{pmatrix}$$


Dynamic programming

- Divide into dependent sub-problems
- Solving sub-problems just once
- Memoization: store all results for recursion
Global, local, overlap and repeated alignment

**Global alignment**
- For full length sequence comparison
- For detection of indels

Needleman-Wunsch Algorithm:
Default Initialization:

\[ F(0,0) = 0. \]

Recurrence relation:

\[ F(i,j) = \max \left\{ \begin{array}{ll} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{array} \right\} \]

Final score is \( F(m,n) \).

**Local alignment**
- For detection of common domain or subsequence
- For detection of rearrangement

Smith-Waterman Algorithm:
Default Initialization:

\[ F(0,0) = 0. \]

Recurrence relation:

\[ F(i,j) = \max \left\{ \begin{array}{ll} 0, \\ F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{array} \right\} \]

Final score is \( F_{max} \).

---

Global, local, overlap and repeated alignment

**Global alignment**

Assuming that, match score is 1, mismatch score is -3, gap penalty is -2

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2</td>
<td>-6</td>
<td>-8</td>
<td>-12</td>
<td>-10</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>G</td>
<td>-4</td>
<td>-1</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
</tr>
<tr>
<td>T</td>
<td>-6</td>
<td>-3</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
</tr>
<tr>
<td>A</td>
<td>-10</td>
<td>-7</td>
<td>-5</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>-12</td>
<td>-9</td>
<td>-6</td>
<td>-3</td>
<td>-5</td>
</tr>
</tbody>
</table>

ACGTC - -
A - GTCAG

**Local alignment**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

ACGTC . .
A. GTCAG
Global, local, overlap and repeated alignment

**Overlap alignment**

- One sequence contains the another or they overlapped

Overlap Alignment Algorithm:
Specific Initialization:
\[ F(0, 0) = F(i, 0) = F(0, j) = 0. \]

Recurrence relation:
\[ F(i, j) = \max \left\{ \begin{array}{l} F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{array} \right. \]

Final score is \( F_{\text{max}} \) on the border \((i, n) \cup (m, j)\).

**Repeated alignment**

- To find out all optimized local alignment results

Repeated alignment:
Default Initialization:
\[ F(0, 0) = 0. \]

Recurrence relation:
\[ F(i, 0) = \max \left\{ \begin{array}{l} F(i-1, 0), \\ F(i, j) - T, j = 1, \ldots, n. \end{array} \right. \]

\[ F(i, j) = \max \left\{ \begin{array}{l} F(i, 0), \\ F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d. \end{array} \right. \]

Final score is \( F_{\text{max}} \).

---

Global, local, overlap and repeated alignment

**Overlap alignment**

Assuming that, match score is 5, mismatch score is -3, gap penalty is -2, threshold is 7.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>5</td>
<td>3</td>
<td>-1</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>-3</td>
<td>3</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>-3</td>
<td>1</td>
<td>8</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>C</td>
<td>-3</td>
<td>2</td>
<td>6</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>16</td>
</tr>
</tbody>
</table>

ACGTC\text{-}\text{-} - CGTCAG

**Repeated alignment**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

ACGTC\text{-}\text{-} - CGTCAG

ACGTC\text{-} A\text{-} GTCAC A\text{-}GTAC

ACGTC\text{-}\text{-} - CGTCAG

ACGTC\text{-}\text{-} - CGTCAG
More complex models for pairwise alignment

- To control the size of gaps
  
  Affine gap penalty:

  Recurrence relation:

  \[
  M(i, j) = \max \begin{cases} 
  M(i - 1, j - 1) + s(x_i, y_j), \\
  I(i - 1, j - 1) + s(x_i, y_j); 
  \end{cases} 
  \]

  \[
  I(i, j) = \max \begin{cases} 
  M(i, j - 1) - d, \\
  M(i - 1, j) - d, \\
  I(i, j - 1) - e, \\
  I(i - 1, j) - e. 
  \end{cases} 
  \]
A brief introduction to BLAST

BMI Basic Local Alignment Search Tool
- Find high scoring local alignment between a query sequence and a target database.
- 50 times faster than pure dynamic programming.

Heuristic alignment algorithm
- ‘Neighbourhood words’ of fixed length as seeds (3 for amino acid sequence and 11 for nucleic acid sequence with 2bits per residue)
- Scanning through whole database sequences
- ‘Hit extension’ extend the possible match in both direction until achieving the maximum scoring extension


A brief introduction to BLAST

BMI Expectation value (E-value)
- The E-value (associated to a score S) is the number of distinct alignments, with a score equivalent to or better than S, that are expected to occur in a database search by chance.
- The lower the E value, the more significant the score is.

E-value:
\[ E = K \cdot n e^{-\lambda S} \]
A brief introduction to BLAST

- Functional options
  - blastn: Search a **nucleotide** database using a **nucleotide** query
  - blastp: Search **protein** database using a **protein** query
  - blastx: Search **protein** database using a **translated nucleotide** query
  - tblastn: Search **translated nucleotide** database using a **protein** query
  - tblastx: Search **translated nucleotide** database using a **translated nucleotide** query

- Procedures:
  - formatdb -i < database (fasta file)>
  - blastall
    - -i <input (fasta file)>
    - -d <database (fasta file)>
    - -e <e-value cut-off>
    - -m <the format of output, commonly choose 8>
    - -o <output file>
    - -p <type of search>
A brief introduction to BLAST

- **Result**
  - Query sequence
  - Database sequence
  - Similarity
  - No. of matches
  - No. of gaps
  - Gap extension
  - Matched region on query sequence
  - Matched region on database sequence
  - E-value
  - Score

<table>
<thead>
<tr>
<th>Score</th>
<th>1854.60</th>
<th>484</th>
<th>504</th>
<th>21</th>
<th>0.0</th>
<th>859</th>
</tr>
</thead>
<tbody>
<tr>
<td>165</td>
<td>96.90</td>
<td>489</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>484</td>
</tr>
<tr>
<td>165</td>
<td>98.78</td>
<td>483</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>483</td>
</tr>
<tr>
<td>165</td>
<td>99.00</td>
<td>512</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>511</td>
</tr>
<tr>
<td>165</td>
<td>99.79</td>
<td>504</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>504</td>
</tr>
<tr>
<td>165</td>
<td>99.78</td>
<td>502</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>502</td>
</tr>
<tr>
<td>165</td>
<td>99.40</td>
<td>526</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>525</td>
</tr>
<tr>
<td>165</td>
<td>99.81</td>
<td>532</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>532</td>
</tr>
<tr>
<td>165</td>
<td>99.81</td>
<td>529</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>529</td>
</tr>
<tr>
<td>165</td>
<td>97.84</td>
<td>515</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>515</td>
</tr>
</tbody>
</table>

Multiple sequence alignment (MSA)

- Multidimensional dynamic programming v.s. progressive alignment
- A brief introduction to Clustal W
Multidimensional dynamic programming v.s. progressive alignment

**Multidimensional DP**
- Time complexity: $O(2^N L^N)$
- Procedure:

**Progressive alignment**
- Time complexity: $O((LN)^2)$
- Procedure:
  - Pairwise alignment for all $N(N-1)/2$ pair of sequences
  - Build a guide tree using
  - Progressively align at nodes

---

A brief introduction to Clustal W

**Process of Algorithm**
- All pairs of sequences are aligned independently and calculate the alignment score as distance
- Build the guide tree by neighbor-joining method based on distances gathered in the first step
- Sequences are progressively aligned according to the branch order in the guide tree. Sequences are aligned through full dynamic programming and weighted by groups.
A brief introduction to Clustal W