BIOS 477/877

Bioinformatics and Molecular Evolution

Lecture 6

Spring 2018

Today's Topics

➢ Pairwise Alignment
  • Dotplot
  • Simple Alignments (scoring scheme)
  • Dynamic Programming Algorithm

➢ Assignment 3 (a short one!)
  Due: January 28 (Sun)

Pairwise Sequence Comparison

➢ Why do we want to compare sequences?
  • To understand their relatedness
    Protein/gene family classification
    Evolutionary relationships/history
  • To predict gene/protein functions, structures
  • To identify functional domains/motifs
    shared among a group of molecules

Hypothesis: common descent

Ancestral Sequence

Current Sequences

Hypothesis: common descent

Current Sequences

Substitution

Deletion
Alignment represents a hypothesis regarding the evolutionary history of the two sequences.

**Indel**

Insertion or Deletion?

**Pairwise Sequence Comparison**

**DNA alignment at protein level**

1. Translate DNA sequences to amino acid sequences
2. Align them at the protein level
3. Reverse translate the protein alignment to DNA alignment

- **TranslatorX** [http://translators.co.uk/](http://translators.co.uk/)
- **PAL2NAL** [http://www.bork.embl.de/pal2nal/](http://www.bork.embl.de/pal2nal/)
- **tranalign** [http://www.bioinformatics.nl/cgi-bin/emboss/tranalign](http://www.bioinformatics.nl/cgi-bin/emboss/tranalign)

**Alignment Strategy**

- **Protein alignment is easier** than DNA alignment
  - DNA has only 4 nucleotide types (they can match just by chance more easily)
  - Protein sequences evolve more slowly than DNA sequences (genetic code is redundant; nonsynonymous substitutions are less frequent than synonymous substitutions)
- If DNA sequences are from coding regions:
  - Translate them, and **align at the protein level** first
  - This ensures gaps inserted between codons (prevents insertion of frame-shifting gaps)
- Do not blindly rely on the default parameter set
  - Try various **scoring matrices, gap penalties, etc.**

**Dot Plot (Dot Matrix) visualizes sequence similarity**

**Diagonal lines** show identical sequences
Pairwise Sequence Comparison

Diagonal lines show identical sequences

Horizontal lines show gaps in Seq 2

Vertical lines show gaps in Seq 1

Identical sequences

Internal repeats

Single nucleotide repeats

Pairwise sequence comparison by Dotplot

Dotlet: http://myhits.isb-sib.ch/cgi-bin/dotlet
(Dotlet is also available: https://dotlet.vital-it.ch)


DotMatcher (A program in EMBOSS)
http://emboss.bioc.sunysb.edu/cgi-bin/emboss/dotmatcher
(see course Web for other EMBOSS links)

JDotter: Java Dot Plot Alignments
http://pgrc.ipk-gatersleben.de/jdotter/

YASS: Visualization of local pairwise alignments
http://bioinfo.lifl.fr/yass/yass.php

Dotter (now part of SeqTools)
http://www.sanger.ac.uk/science/tools/seqtools
Pairwise Sequence Comparison

➢ Protein domain detection by dotplot

Dotlet (example)
http://myhits.isb-sib.ch/cgi-bin/dotlet

Simple Alignments

➢ Same length & no gap

AATCTATA
AAGATATA

➢ Only one possible alignment

Simple Alignments

➢ Varied length & no (internal) gap

AATCTATA
AAGATA

• If we only shift the shorter sequence, how many possible alignments?

Simple Alignments

➢ Varied length & no (internal) gap

AATCTATA
AAGATA
AATCTATA
AAGATA
AATCTATA
AAGATA

➢ Three possible alignments.

• Which alignment is the best?

Simple Alignments

➢ Varied length & no (internal) gap

AATCTATA
AAGATA
AATCTATA
AAGATA
AATCTATA
AAGATA

Score

• Matches
• Mismatches

Two elements

Alignment Score = (match score) x (the number of matched pairs) + (mismatch score) x (the number of mismatched pairs)

If match score = 1, mismatch score = 0
**Simple Alignments**

- **Varied length & no (internal) gap**

  - Score

  - Alignment Score = (match score) x (the number of matched pairs) + (mismatch score) x (the number of mismatched pairs)
  
  If match score = 1, mismatch score = -2

- **Varied length & gaps considered**

  - 3 alignments possible!
  
  - If two gaps are allowed in the shorter sequence …
  
    - 28 alignments possible!

- **Pairwise Alignment Problem**

  - To find the best alignment(s) between two sequences

    - Exhaustive search method
      - Examine all possible alignments
      - Develop a method to score each alignment (match score, mismatch score, gap penalty)
      - Choose the best (optimal) alignment based on the score

    - Not feasible for long sequences (e.g., 100bp x 95bp with 5 gaps only in the shorter sequence would require testing ~ 55 million possible alignments)
**Dynamic Programming Algorithm**

What is Dynamic Programming?

- An algorithm in which an optimization problem is solved by saving the optimal scores for the solution of every sub-problem instead of recalculate them repetitively.

- A problem is broken apart into reasonably sized sub-problems.

- These partial results are used to compute the final answer.


- Problem: Align CACGA and CGA
  (Match: 1, mismatch: 0, gap penalty: -1)

If we align:

- C
  the rest to be aligned: ACGA & GA
  Score(C/C) + Score(ACGA/GA) = 1

If we align:

- C
  the rest to be aligned: ACGA & CGA
  Score(C/C) + Score(ACGA/CGA) = -1

If we align:

- C
  the rest to be aligned: CACGA & GA
  Score(C/C) + Score(CACGA/GA) = -1

Dynamic Programming Algorithm


- Problem: Align CACGA and CGA
  (Match: 1, mismatch: 0, gap penalty: -1)

Three steps for sequence alignment (Eddy, 2004)

1. Initialization
2. Matrix filling (Scoring)
3. Traceback (Alignment)

- Sequences a and b
- Simple scoring scheme:
  - \( s(a_i, b_j) = 2 \) (match)
  - \( s(a_i, b_j) = -1 \) (mismatch)
  - \( w = -2 \) (gap penalty)

Initialization step

- Fill the first row and column with multiples of the gap penalty (-2)

Dynamic Programming Algorithm
Dynamic Programming Algorithm

Matrix filling step

<table>
<thead>
<tr>
<th>a</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Start from this cell

Dynamic Programming Algorithm

Matrix filling step

<table>
<thead>
<tr>
<th>a</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
</tr>
</tbody>
</table>

3 alignment possibilities

1. Aligned

2. Gap in a

3. Gap in b

Dynamic Programming Algorithm

Matrix filling step

<table>
<thead>
<tr>
<th>a</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
</tr>
</tbody>
</table>

3 alignment possibilities

1. Aligned (match) \( S_{ij} = S_{i-1,j} + s(a_i, b_j) \)

2. Gap in a \( S_{ij} = S_{i-1,j} + w \)

3. Gap in b \( S_{ij} = S_{i,j-1} + w \)

Dynamic Programming Algorithm

Matrix filling step

<table>
<thead>
<tr>
<th>a</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
</tr>
</tbody>
</table>

3 alignment possibilities

1. Aligned (match) \( S_{ij} = S_{i-1,j} + s(a_i, b_j) \)

2. Gap in a \( S_{ij} = S_{i-1,j} + w \)

3. Gap in b \( S_{ij} = S_{i,j-1} + w \)

Score in the previous cell

Dynamic Programming Algorithm

Matrix filling step

<table>
<thead>
<tr>
<th>a</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
</tr>
</tbody>
</table>

3 alignment possibilities

1. Aligned (match) \( S_{ij} = S_{i-1,j} + 0 + s(a_i, b_j) \)

2. Gap in a \( S_{ij} = S_{i-1,j} + w + 2 = 2 \)

3. Gap in b \( S_{ij} = S_{i,j-1} + w + 2 = 2 \)

Maximum

Cell score

\( S_{ij} = S_{i-1,j} + s(a_i, b_j) \)

\( S_{ij} = S_{i-1,j} + w \)

\( S_{ij} = S_{i,j-1} + w \)

Score in the previous cell
Dynamic Programming Algorithm

Matrix filling step

- Cell score
  - \( S_{ij} = a_{ij} + S_{i-1,j} \) (Aligned match)
  - \( S_{ij} = S_{i,j-1} + w \) (Gap in \( a \))
  - \( S_{ij} = S_{i,j} + w \) (Gap in \( b \))

Maximize the score path

Matrix filling step

- Score in the previous cell
  - Maximize the score path
- Calculate the next cell score
- Keep the maximum score path

Dynamic Programming Algorithm

Matrix filling step

- Cell score
  - \( S_{ij} = a_{ij} + S_{i-1,j} \) (Aligned match)
  - \( S_{ij} = S_{i,j-1} + w \) (Gap penalty)

Maximize the score path
Dynamic Programming Algorithm

Matrix filling step

\[ a(i,j) = \begin{cases} 2 & (a = b, \text{ match}) \\ -1 & (a \neq b, \text{ mismatch}) \\ -2 & (\text{gap penalty}) \end{cases} \]

<table>
<thead>
<tr>
<th>cell</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>T</td>
<td>-4</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
<tr>
<td>C</td>
<td>-12</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
</tbody>
</table>

Dynamic Programming Algorithm

Matrix filling step

\[ a(i,j) = \begin{cases} 2 & (a = b, \text{ match}) \\ -1 & (a \neq b, \text{ mismatch}) \\ -2 & (\text{gap penalty}) \end{cases} \]

<table>
<thead>
<tr>
<th>cell</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>T</td>
<td>-4</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
<tr>
<td>C</td>
<td>-12</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
</tbody>
</table>

Traceback

\[ a(i,j) = \begin{cases} 2 & (a = b, \text{ match}) \\ -1 & (a \neq b, \text{ mismatch}) \\ -2 & (\text{gap penalty}) \end{cases} \]

<table>
<thead>
<tr>
<th>cell</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>(b)</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>T</td>
<td>-4</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
<tr>
<td>C</td>
<td>-12</td>
<td>-10</td>
<td>-5</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>-13</td>
</tr>
</tbody>
</table>

Traceback

Follow the arrows from the right-bottom corner to the left-top corner.

Alignment Score

\[ \text{Score} = (2 \times 5) + (-1 \times 1) - (2 \times 1) = 7 \]
Dynamic Programming Algorithm

Initialization step

 GAP penalty: -2

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>A</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fill the first row and column with multiples of the gap penalty (-2)

Terminal Gaps

AATCTGC

------

Gaps in the 2nd sequence

AATCTGC

-------

ATTCTC

---

Gaps in the 1st sequence

Dynamic Programming Exercise

Assignment 3

Perform a small example of pairwise alignment using the dynamic programming algorithm

Due: January 28 (Sun)

We will go over it next Tuesday

Note: There could be more than one optimum alignment path!