Part I. Sequence Analysis:

1. Global Alignment

Let’s talk about DNA more ...

Birth of Molecular Biology

\[ \text{DNA} \]

\[ \text{Phosphate} \]

\[ \text{Sugar} \]

\[ \text{Nitrogenous Base} \]

A, C, G, T

Genome: all the information contained within DNA in an organism

Discovered in 1953

Let’s talk about DNA more ...

Readings

- Readings on Lecture 2 & 3
  - This book is freely available

<table>
<thead>
<tr>
<th>Sequence Analysis (5 lectures)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- dynamic programming, de Bruijn graph</td>
</tr>
<tr>
<td>- sequence alignment, genome assembly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture 2 Global alignment [PDF]</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chap 2.1-2.4 of Durbin, Eddy, Krogh and Mitchinson &quot;Biological Sequence Analysis&quot; [link]</td>
</tr>
<tr>
<td>- What is dynamic programming?, Nature Biotechnology, 2004 [link]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lecture 3 Local alignment, Linear space algorithm [PDF]</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chap 2.6-2.8 of Durbin [link]</td>
</tr>
</tbody>
</table>
Genetics in the 20th Century

By Francis Crick in 1957

Human Genome Project

1990: Start
2000: Bill Clinton: Draft
2003: Finished
2014: now what?

Sequencing Growth

Cost of one human genome
- 2004: $30,000,000
- 2008: $100,000
- 2010: $10,000
- 2011: $4,000
- 2012-13: $1,000
- ???: $300

How much would you pay for a smart phone?

Uses of Genomes

- Medicine
  - Mendelian diseases
  - Cancer
  - Drug dosage (eg. Warfarin)
  - Disease risk
  - Diagnosis of infections
  - ...
- Ancestry
- Genealogy
- Nutrition
  -
Complete DNA Sequences

More than 1000 complete genomes have been sequenced

Evolution

Comparing Human, Chimp, and Mouse Genomes

- 95% of the chimp genome that maps to identical sequence in the human genome

The white areas indicate areas that do not map well to the other genome.

Comparing Human, Chimp, and Mouse Genomes

- 34% of the mouse genome mapped to identical sequence in the human genome
Evolution at the DNA level

SEQUENCE EDITS

DELETION

>>ACGTCAGGTTACCA<<

>>AC----CAGTCCACCA<<

Mutation

REARRANGEMENTS

Deletion

Mutation

Inversion

Translocation

Duplication

Evolutionary Rates

next generation

OK

OK

OK

X

X

Still OK?

Sequence conservation implies functional conservation

Sequence Alignment

Alignment is the key to
- Finding important regions
- Determining function
- Uncovering evolutionary events
Sequence alignment

AGGCTATCACCTGACCTCCAGGCGATGCC
TAGCTATCAGCCGCGGTGATTTGGCCGAC

AGG-CTATCAC-TGACCTCCAGGCGA-TCGCC
TAG-CTATCACA-GGCCGATGCGATTTGCCGCAG

Definition
Given two strings \( x = x_1x_2...x_M \), \( y = y_1y_2...y_N \), an alignment is an assignment of gaps to positions \( 0,..., M \) in \( x \), and \( 0,..., N \) in \( y \), so as to line up each letter in one sequence with either a letter, or a gap in the other sequence.

What is a good alignment?

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Matches</th>
<th>Mismatches</th>
<th>Gaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGGCTAGTT, AGCGAAGTTT</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>AGGCTAGTT-AGCGAAGTTT</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>AGGCTAGTT-AG-GAAGTTT</td>
<td>7</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>AGGCTAGTT-AG-CAAGTTT</td>
<td>7</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Scoring Function

- Sequence edits:
  - Mutations
  - Insertions
  - Deletions

Scoring Function:
- Match: \(+m\)
- Mismatch: \(-s\)
- Gap: \(-d\)

Score \( F = (# \text{ matches}) \times m - (# \text{ mismatches}) \times s - (#\text{ gaps}) \times d \)

How do we compute the best alignment?

Every non-decreasing path from \((0,0)\) to \((M,N)\) corresponds to an alignment of the two sequences, and vice versa. (exercise)
How do we compute the best alignment?

Too many possible alignments:

$$>> 3 \min(M, N)$$

Alignment is additive

Observation:

The score of aligning

$$x_1 \ldots x_M$$

$$y_1 \ldots y_N$$

is additive

Say that

$$x_1 \ldots x_i$$

$$x_{i+1} \ldots x_M$$

aligns to

$$y_1 \ldots y_j$$

$$y_{j+1} \ldots y_N$$

The two scores add up:

$$F(x[1:M], y[1:N]) = F(x[1:i], y[1:j]) + F(x[i+1:M], y[j+1:N])$$

Dynamic Programming

- Consider subproblems for $$i \leq M$$ and $$j \leq N$$
  - Align $$x_1 \ldots x_i$$ to $$y_1 \ldots y_j$$

- Original problem is one of the subproblems
  - Align $$x_1 \ldots x_i$$ to $$y_1 \ldots y_j$$

- Each subproblem is easily solved from smaller subproblems
  - We will show next

- Then, we can apply Dynamic Programming!!!

Let $$F(i, j) = \text{optimal score of aligning}$$

$$x_1 \ldots x_i$$

$$y_1 \ldots y_j$$

F is the DP "Matrix" or "Table"

"Memorization"

Dynamic Programming (cont’d)

Notice three possible cases:

1. $$x_i$$ aligns to $$y_j$$

$$x_1 \ldots x_{i-1} x_i$$

$$y_1 \ldots y_{j-1} y_j$$

$$F(i, j) = F(i-1, j-1) + \begin{cases} m, & \text{if } x_i = y_j \\ e, & \text{if not} \end{cases}$$

2. $$x_i$$ aligns to a gap

$$x_1 \ldots x_{i-1}$$

$$y_1 \ldots y_j \_$$

$$F(i, j) = F(i-1, j) - d$$

3. $$y_j$$ aligns to a gap

$$x_1 \ldots x_i \_$$

$$y_1 \ldots y_{j-1} y_j$$

$$F(i, j) = F(i, j-1) - d$$
Dynamic Programming (cont’d)

How do we know which case is correct?

**Inductive assumption:**
F(i, j – 1), F(i – 1, j), F(i – 1, j – 1) are optimal

Then,
\[
F(i, j) = \max \begin{cases} 
F(i – 1, j – 1) + s(x_i, y_j) \\
F(i – 1, j) – d \\
F(i, j – 1) – d 
\end{cases}
\]

where \( s(x_i, y_j) = \begin{cases} 
m, & \text{if } x_i = y_j \\
-s, & \text{if not} 
\end{cases} \)

---

**Example**

\( x = \text{ACGCTG} \) match: +2
\( y = \text{CATGT} \) mismatch, gap: -1

\[
F(i, j) = \text{optimal score of aligning } x_{1, \ldots, i} \text{ to } y_{1, \ldots, j}
\]

\[
\begin{array}{cccccc}
& j & 0 & 1 & 2 & 3 & 4 & 5 \\
i & 0 & 0 & -1 & -2 & -3 & -4 & -5 \\
1 & A & -1 & & & & & \\
2 & C & -2 & & & & & \\
3 & G & -3 & & & & & \\
4 & C & -4 & & & & & \\
5 & T & -5 & & & & & \\
6 & G & -6 & & & & & \\
\end{array}
\]

\( x = \text{ACGCTG} \) match: +2
\( y = \text{CATGT} \) mismatch, gap: -1

\[
F(i, j) = \text{optimal score of aligning } x_{1, \ldots, i} \text{ to } y_{1, \ldots, j}
\]

\[
\begin{array}{cccccc}
& j & 0 & 1 & 2 & 3 & 4 & 5 \\
i & 0 & 0 & -1 & -2 & -3 & -4 & -5 \\
1 & A & -1 & & & & & \\
2 & C & -2 & & & & & \\
3 & G & -3 & & & & & \\
4 & C & -4 & & & & & \\
5 & T & -5 & & & & & \\
6 & G & -6 & & & & & \\
\end{array}
\]
Example
\[ s(C,-) = -1 \]

\[ x = ACGCTG \quad \text{match:} \quad +2 \]
\[ y = CATGT \quad \text{mismatch, gap:} \quad -1 \]

\[ \begin{array}{cccccc}
   j & 0 & 1 & 2 & 3 & 4 & 5 \\
   i & C & A & T & G & T \\
   0 & 0 & -1 & -2 & -3 & -4 & -5 \\
   1 & A & -1 & -1 & 1 & & \\
   2 & C & -2 & 1 & & & \\
   3 & G & -3 & & & & \\
   4 & C & -4 & & & & \\
   5 & T & -5 & & & & \\
   6 & C & -6 & & & & \\
\end{array} \]

\[ x = ACGCTG \quad \text{match:} \quad +2 \]
\[ y = CATGT \quad \text{mismatch, gap:} \quad -1 \]

\[ \begin{array}{cccccc}
   j & 0 & 1 & 2 & 3 & 4 & 5 \\
   i & C & A & T & G & T \\
   0 & 0 & -1 & -2 & -3 & -4 & -5 \\
   1 & A & -1 & -1 & 1 & & \\
   2 & C & -2 & 1 & & & \\
   3 & G & -3 & & & & \\
   4 & C & -4 & & & & \\
   5 & T & -5 & & & & \\
   6 & G & -6 & & & & \\
\end{array} \]

Time = \( O(MN) \)
### Finding alignments: trace back

Arrows = (ties for) max in \( F(i,j) \); 3 LR-to-UL paths = 3 optimal alignments

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>G</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>-5</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

### The Needleman-Wunsch Matrix

- Every nondecreasing path from \((0,0)\) to \((M,N)\) corresponds to an alignment of the two sequences
- An optimal alignment is composed of optimal sub-alignments

### The Needleman-Wunsch Algorithm

1. **Initialization.**
   - a. \( F(0,0) = 0 \)
   - b. \( F(0,j) = -j \times d \)
   - c. \( F(i,0) = -i \times d \)

2. **Main Iteration.** Filling-in partial alignments
   - For each \( i = 1 \ldots M \)
   - For each \( j = 1 \ldots N \)
   
   \[
   F(i, j) = \max \begin{cases} 
   F(i-1,j-1) + s(x_i, y_j) & \text{[case 1]} \\
   F(i-1,j) - d & \text{[case 2]} \\
   F(i,j-1) - d & \text{[case 3]} 
   \end{cases}
   \]
   
   \[
   \text{Ptr}(i, j) = \begin{cases} 
   \text{DIAG}, & \text{if [case 1]} \\
   \text{LEFT}, & \text{if [case 2]} \\
   \text{UP}, & \text{if [case 3]} 
   \end{cases}
   \]

3. **Termination.** \( F(M,N) \) is the optimal score, and from \( \text{Ptr}(M,N) \) can trace back optimal alignment
Performance

- Time: \( O(NM) \)
- Space: \( O(NM) \)
- Later we will cover more efficient methods

Scoring the gaps more accurately

Current model:

- Gap of length \( n \)
- Incurs penalty \( n \cdot d \)

However, gaps usually occur in bunches

Concave gap penalty function \( \gamma(n) \)

(aka Convex \( -\gamma(n) \)):

\[ \gamma(n): \]

\[ \text{for all } n, \quad \gamma(n + 1) - \gamma(n) \leq \gamma(n) - \gamma(n - 1) \]

Convex gap dynamic programming

**Initialization:** same

**Iteration:**

\[ F(i, j) = \max \begin{cases} 
  F(i-1, j-1) + s(x_i, y_j) \\
  \max_{k=0 \ldots i-1} \{ F(k, j) - \gamma(i-k) \} \\
  \max_{k=0 \ldots j-1} \{ F(i, k) - \gamma(j-k) \} 
\end{cases} \]

**Termination:** same

**Running Time:** \( O(N^2M) \) (assume \( N > M \))

**Space:** \( O(NM) \)

Compromise: **affine gaps**

\[ \gamma(n) = d + (n - 1)e \]

\[ \begin{array}{ll}
  & - \\text{gap} \\text{open} \\text{extend} \\
  \text{gap} & | & | & | \\
  \text{open} & | & | & | \\
  \text{extend} & | & | & |
\end{array} \]

To compute optimal alignment,

At position \( i, j \), need to "remember" best score if gap is open
best score if gap is not open

\[ F(i, j): \quad \text{score of alignment } x_1 \ldots x_i \text{ to } y_1 \ldots y_j \]

\[ \text{if } x_i \text{ aligns to } y_j \]

\[ G(i, j): \quad \text{score if } x_i \text{ aligns to a gap after } y_j \]

\[ H(i, j): \quad \text{score if } y_j \text{ aligns to a gap after } x_i \]

\[ V(i, j) = \text{best score of alignment } x_1 \ldots x_i \text{ to } y_1 \ldots y_j \]
Needleman-Wunsch with affine gaps

Why do we need matrices F, G, H?

- $x_i$ aligns to $y_j$
  \[ x_1 \ldots x_{i-1} x_i \cdots \quad y_1 \ldots y_{j-1} y_j \]
  - Add $-d$
  \[ G(i+1, j) = F(i, j) - d \]
- $x_i$ aligns to a gap after $y_j$
  \[ x_1 \ldots x_{i-1} x_i \cdots \quad y_1 \ldots y_{j-1} \quad y_j \]
  - Add $-e$
  \[ G(i+1, j) = G(i, j) - e \]

Because, perhaps $G(i, j) < V(i, j)$ (it is best to align $x_i$ to $y_j$ if we were aligning only $x_1 \ldots x_i$ to $y_1 \ldots y_j$ and not the rest of $x, y$), but on the contrary $G(i, j) - e > V(i, j) - d$ ($d > e$) (i.e., had we "fixed" our decision that $x_i$ aligns to $y_j$, we could regret it at the next step when aligning $x_1 \ldots x_{i+1}$ to $y_1 \ldots y_{j+1}$).

Needleman-Wunsch with affine gaps

**Initialization:**
\[
V(i, 0) = d + (i - 1) e \\
V(0, j) = d + (j - 1) e
\]

**Iteration:**
\[
V(i, j) = \max \{ F(i, j), G(i, j), H(i, j) \}
\]
\[
F(i, j) = V(i-1, j-1) + s(x_i, y_j)
\]
\[
G(i, j) = \max \{ V(i-1, j) - d, G(i-1, j) - e \}
\]
\[
H(i, j) = \max \{ V(i, j-1) - d, H(i, j-1) - e \}
\]

**Termination:**
$V(i, j)$ has the best alignment

To generalize a bit...

... think of how you would compute optimal alignment with this gap function

... in time $O(MN)$
**Bounded Dynamic Programming**

Assume we know that $x$ and $y$ are very similar

**Assumption:** $\# \text{gaps}(x, y) < k(N)$

Then, $|i - j| < k(N)$ implies $|i - j| < k(N)$

We can align $x$ and $y$ more efficiently:

Time, Space: $O(N \times k(N)) < O(N^2)$

---

**Summary**

- Basics on genome sequences
- Why sequence alignment is important
- Dynamic programming
- Global alignment method
  - Needleman-Wunsch algorithm with various gap functions

---

**Acknowledgement**

- The lecture slides were generated based on a book by Durbin et al, and Prof. Serafim Batzoglou’s slides

---

**Bounded Dynamic Programming**

**Initialization:**

$F(i,0), F(0,j)$ undefined for $i, j > k$

**Iteration:**

For $i = 1...M$

For $j = \max(1, i - k) ... \min(N, i + k)$

$$F(i, j) = \max\begin{cases} F(i - 1, j - 1) + s(x_i, y_j) \\ F(i - 1, j) - d, \text{ if } j > i - k(N) \\ F(i - 1, j) - d, \text{ if } j < i + k(N) \end{cases}$$

Termination: same

Easy to extend to the affine gap case