Local Sequence Alignment
Smith & Waterman (1981)

Rolf Backofen & Martin Raden

Lehrstuhl für Bioinformatik
Institut für Informatik
Albert-Ludwigs-Universität Freiburg

Course Bioinformatics I
built on November 20, 2018
Local alignment - find conserved motifs

So far:

- *global* alignment, ie. how distant are \( a \) and \( b \)
  \[ \Rightarrow \text{whole } a \text{ and } b \text{ aligned and scored} \text{ by distance } w \]

\[ \arg \min_{A=(a \Diamond, b \Diamond)} w(A) \]

Now:

- *local* alignment
  \[ \Rightarrow \text{search for local motifs that are similar} \text{ by similarity } s \]

  e.g. \( a = \text{C} \boxed{\text{AACGTT}} \text{AC} \) but not necessarily identic!
  \[ b = \text{GGG} \boxed{\text{AAGGT}} \text{GG} \]

\[ \arg \max_{\hat{A}=(\hat{a} \Diamond, \hat{b} \Diamond)} s(\hat{A}) \text{ where } \hat{a}, \hat{b} \text{ are subsequences of } a, b \]

\[ \Rightarrow \text{only subalignment } \hat{A} \text{ is scored/optimized!} \]
Distance vs. Similarity

Why is distance not useful for local alignment scoring?

(I)  
\[ a = \begin{array}{c} \text{XX} \end{array} \begin{array}{c} \text{AA} \end{array} \begin{array}{c} \text{XXXX} \end{array} \]
\[ b = \begin{array}{c} \text{YYY} \end{array} \begin{array}{c} \text{AA} \end{array} \begin{array}{c} \text{YYY} \end{array} \]
\[ w(\hat{A}) = 0 \]
\[ s(\hat{A}) = 10 \]

distance \( w(x, y) = \begin{cases} 0 & \text{if } x = y \\ 5 & \text{else} \end{cases} \)

similarity \( s(x, y) = \begin{cases} 5 & \text{if } x = y \\ 0 & \text{else} \end{cases} \)

⇒ distance of (I) and (II) equal

⇒ is represented best by similarity

(II)  
\[ a = \begin{array}{c} \text{XX} \end{array} \begin{array}{c} \text{AAAA} \end{array} \begin{array}{c} \text{XX} \end{array} \]
\[ b = \begin{array}{c} \text{YY} \end{array} \begin{array}{c} \text{AAAA} \end{array} \begin{array}{c} \text{YY} \end{array} \]
\[ w(\hat{A}) = 0 \]
\[ s(\hat{A}) = 20 \]

but (II) = better/larger local motif!

Score local alignment with similarities instead of distances

⇒ \( s(x, y) \):
  - can be positive or negative (e.g. gaps typically negative)
  - no metric
  - positive means similar
  - 0 = neutral scoring

CC BY-NC-SA 4.0 - Backofen/Raden - Bioinformatics - University Freiburg - Germany
Needleman-Wunsch using similarity measurement

- **Getting local:** brute force solution = find best global alignment for all subsequence $\hat{a}, \hat{b}$ combinations (using similarity)

- **So far:**
  - minimal alignment distance wanted
  - $w(x, x) = 0 \Rightarrow$ low costs for identical symbols
  - matrix $(D_{i,j})$, where $D_{i,j}$ lowest distance of $a_1..a_i, b_1..b_j$

- **Now:**
  - maximal similarity wanted
  - $s(x, x)$ high $\Rightarrow$ high similarity for identity
  - matrix $(S_{i,j})$, where $S_{i,j}$ best similarity for prefixes $a_1..a_i$ and $b_1..b_j$

$\Rightarrow$ recursion: $S_{i,j} = \max \begin{cases} S_{i,j-1} + s(-, b_j), \\ S_{i-1,j-1} + s(a_i, b_j), \\ S_{i-1,j} + s(a_i, -) \end{cases}$

- **Problem:** have to compute $|a|^2|b|^2$ alignments $\Rightarrow O(n^6)$ time
Second try: Needleman-Wunsch of suffixes only

- **Observation:** any entry $S_{i,j}$ in prefix-based matrix $S$ allows to neglect alignment of remaining suffixes

- **Example:**
  
  \[
  a = \begin{array}{c}
  \text{CGGT}\text{AC} \\
  \text{AAGGT}\text{GG}
  \end{array}
  \]
  \[S_{4,5}\] ignores AC and GG

- **Now:**
  
  - compute $S$ for all subsequence starts (ie. suffix combinations)
  - find maximal entry among all $S$ entries in all tables

- **e.g.:**
  
  alignment of suffixes $\hat{a} = \text{GGTAC}$ and $\hat{b} = \text{GGTGG}$
  
  provides subalignment for
  
  \[
  \hat{a} = \begin{array}{c}
  \text{GGT}\text{AC} \\
  \text{GGT}\text{GG}
  \end{array}
  \]

- **Still:**
  
  have to compute $|a||b|$ suffix alignments $\Rightarrow O(n^4)$ time
Back to the drawing board . . .

- **Goal:** maximal local alignment score, ie. subsequences \( \hat{a}, \hat{b} \)

```
  \[
  \begin{array}{c}
    S_{i,j} = \max \left\{ S_{i,j-1} + s(-, b_j), S_{i-1,j-1} + s(a_i, b_j), S_{i-1,j} + s(a_i, -), S_{0,0} \right\}
  \end{array}
  \]
```

\( \Rightarrow \) if score too low, ignore prefix alignment

- **So far:** ignore right tail by \( \arg \max S_{i,j} \), which leaves \( \hat{a} \odot \hat{b} \)

\( \Rightarrow \) alignment of tails = neglectable

- **Idea:** prefix alignment with new recursion case \( S_{0,0} \) that skips \( \square \) and assigns neutral score 0 of empty sequences
Local Alignment (Smith & Waterman, 1981)

- Smith-Waterman local alignment
  ⇒ recursion still on alignments of prefixes
  ⇒ matrix $H_{i,j}$, stores best local alignment score of prefixes $a_1 \ldots a_i$ and $b_1 \ldots b_j$, where bad leading subalignments are neglected

- **Recall:** $H$ considers alignments of the form
  
  \[
  \overbrace{\hat{a}^\diamond}^{\text{\(\hat{a}\) suffix of } a_1 \ldots a_i} \quad \Rightarrow \quad 0 + s\left(\overbrace{\hat{b}^\diamond}^{\text{\(\hat{b}\) suffix of } b_1 \ldots b_j}\right) \in H_{i,j}
  \]

⇒ Search for $H_{i,j}$ with maximal value neglect respective tail alignment (\(\square\))
Smith-Waterman recursion

- **Recursion:** \( H_{i,j} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j) \\ H_{i-1,j} + s(a_i, -) \\ H_{i,j-1} + s(-, b_j) \\ 0 \end{cases} \)

  - when other entries < 0 then neglect prefixes \( a_1 \ldots a_i \) and \( b_1 \ldots b_j \triangleright H_{0,0} \)

- **Initialization:**
  - \( H_{0,0} = 0 \)
  - \( H_{0,j} = 0 \) \( \leftarrow \) ignore leading gaps \( \triangleright H_{0,0} \)
  - \( H_{i,0} = 0 \) \( \leftarrow \) (assuming gap score < 0)

- **How to do traceback:**
  - start with \( H_{i,j} \) that is maximal
  - follow directions *like Needleman-Wunsch*
  - stop when a \( H_{i,j} = 0 \) is reached
Example

- Scoring: \( s(x, y) = \begin{cases} +2 & \text{if } x = y \\ -1 & \text{else} \end{cases} \Rightarrow s(-, x) = s(x, -) = -1 \)

- Similarity can be extended to gap penalties (negative values !)

- Matrix for \( a = \text{CCC} \) and \( b = \text{ACACCTT} \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>H_{i,j}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

- Best value (= end of traceback): cell \((3, 5)\) with \( H_{3,5} = 5 \)
  \( \Rightarrow \) traceback: ↖←↖↖

- Associated alignment: \( \text{C–CC} \) \( \Rightarrow \) value: \( 5 = 2 - 1 + 2 + 2 \)
Summary

- Local alignment requires similarity-based scoring and maximization.

- Optimal local alignment by neglecting tail alignments
  \[ \hat{a} \quad \hat{b} \]
  - Neglect trailing part via max screen among all entries
  - Neglect bad leading alignment by additional recursion case

- Smith/Waterman: DP approach for local pairwise alignment
  \[ \Rightarrow \text{still prefix-based recursion scheme} \]
  \[ \Rightarrow \text{computes optimal alignment score in } O(n^2) \text{ time and space} \]
  \[ \Rightarrow \text{traceback finds respective alignment(s) in } O(n) \text{ time each} \]