Bioinformatics
(Matching and Alignment)

Marc Hellmuth

May 13, 2015
Aim:
Compare strings to score/evaluate the (dis)similarity between them.

Sequence alignment arises in many fields:
- Molecular biology
- Inexact text matching (e.g. spell checkers; web page search)
- Speech recognition

Biology:
In biomolecular sequences (DNA, RNA, Proteins) high sequence similarity implies significant functional or structural similarity.

Important:
similar function $\nRightarrow$ similar structure $\nRightarrow$ similar sequences
Sketch: Pattern Matching

**Problem:** Given text $T$ and pattern $P$

**Aim:** Find all occurrences of $P$ in $T$.

**Brute-Force:**
For $i = 1$ to $|T| - |P| + 1$
/* cmp $T_i \ldots T_{i+|P|-1}$ with $P_1 \ldots P_{|P|}$ */
   For $j=1$ to $|P|$
      If $P_j \neq T_{i+j-1}$ then GoTo *mismatch*
   EndFor
write $i$
*mismatch*
EndFor
**Sketch: Improved Pattern Matching**

**Idea:** For string $S$ generate Datastructure $Z_i(S)$

$Z_i(S)$ for $i = 2, \ldots, |S|$ is length of longest substring of $S$, that starts on position $i$ and is a prefix of $S$.

If $Z_i(S) > 0$ the interval $[i, i + Z_i(S) - 1]$ is called $Z$-box

Assume we could efficiently compute the $Z_i(S)$ - Does this help us?
Sketch: Improved Pattern Matching

Given text $T$ and pattern $P$.
Let $\$ be a character neither included in $T$ nor in $P$.
Let $S = P\$T$
Compute $Z_i = Z_i(S)$ for $i = 2, \ldots, |S|$.
(This can be done in $O(|S|)$ time → "Datenstrukturen und effiziente Algorithmen")

For $i = 1$ to $|T| - |P| + 1$
    If $Z_{i+|P|-1} = |P|$ then write $i$
EndFor
(This can be done in $O(|T|)$ time.

⇒ "approximate" pattern matching ⇒ alignment
Edit Distance

Edit Operations:

• Insertion of character
• Deletion of character
• Replacement of one character by some other one

Edit Distance = Min. Nr. of Edit Operations to transform string $u$ to string $v$ (equivalent transform string $v$ to string $u$)

$$
\begin{array}{ccccccc}
D & M & M & R & M & M & I \\
writer & - & (M = \text{Match}) \\
- & riders
\end{array}
$$

Edit Script = string over alphabet \{I, D, R, M\} that describes transformation from $u$ to $v$.

Edit Distance Problem: For two strings compute edit distance and optimal edit script.
Example

$u = \text{TGCATAT}$  $v = \text{ATCCGAT}$

$u = \text{TGCATAT} \xrightarrow{\text{del. last } T} \text{TGCATA} \xrightarrow{\text{del. last } A} \text{TGCAT} \xrightarrow{\text{add } A \ 1.\text{pos}} \text{ATGCA}$

$\text{ATGCAT} \xrightarrow{\text{repl. } G \text{ by } C \ 3.\text{pos}} \text{ATCCAT} \xrightarrow{\text{insert } G \ 5.\text{pos}} \text{ATCCGAT} = v$

Edit Distance $\leq 5$

$u = \text{TGCATAT} \xrightarrow{\text{ins. } A \ 1.\text{pos}} \text{ATGCATAT} \xrightarrow{\text{del. } T \ 6.\text{pos}} \text{ATGCGTAT} \xrightarrow{\text{repl. } G \text{ by } C \ 3.\text{pos}} \text{ATCCGAT} = v$

Edit Distance $\leq 4$ (OPTIMAL?)
(global pairwise) Alignment

Alternative way to edit script: Alignment

For two strings $u = u_1 \ldots u_m$ and $v = v_1 \ldots v_n$ an alignment $\mathcal{A}$ is a matrix with two rows and entries $\mathcal{A}[i,j]$ that are characters from Alphabet $\Sigma$ (e.g. $\Sigma = \{A, C, G, T\}$) or a gap “-” s.t.

- 1st row $= u$ after deleting all gaps
- 2st row $= v$ after deleting all gaps
- in no column are two gaps

```
writer   -   T   G   C   A   T   A   T
- reader  A   T   C   C   G   -   A   T
```

Cost-Function $\delta : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \rightarrow \mathbb{R}_{\geq 0}$

Unit-Cost-Function

- $\delta(a,b) = 1$ if $a \neq b$
- $\delta(a,b) = 0$ if $a = b$

Alignment Costs $w(\mathcal{A}) = \sum_{i=1} \delta(a_i, b_i)$
**(global pairwise) Alignment**

Alternative way to edit script: Alignment

For two strings \( u = u_1 \ldots u_m \) and \( v = v_1 \ldots v_n \) an alignment \( \mathcal{A} \) is a matrix with two rows and entries \( \mathcal{A}[i,j] \) that are characters from Alphabet \( \Sigma \) (e.g. \( \Sigma = \{A, C, G, T\} \)) or a gap “-” s.t.

- 1st row = \( u \) after deleting all gaps
- 2st row = \( v \) after deleting all gaps
- in no column are two gaps

\[
\begin{align*}
\text{writer} & \quad - \quad T \quad G \quad C \quad A \quad T \\
- \quad \text{riders} & \quad A \quad T \quad C \quad C \quad G \quad - \quad A \quad T
\end{align*}
\]

**Cost-Function**

\[ \delta : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \to \mathbb{R}_{\geq 0} \]

**Unit-Cost-Function**

\[ \delta(a, b) = 1 \text{ if } a \neq b \]
\[ \delta(a, b) = 0 \text{ if } a = b \]

**Alignment Costs**

\[ w(\mathcal{A}) = \sum_{i=1}^{\infty} \delta(a_i, b_i) \]
Lemma

*Edit Distance of two strings* $u, v$ *equals the min. alignments costs* $w(\mathcal{A})$ *between* $u$ *and* $v$ *with unit-cost function.*

How to compute Edit Distance? **Dynamic Programming!**
Recurrence Function $D$ (Needleman-Wunsch Algorithm)

Given the strings $u = u_1 \ldots u_m$ and $v = v_1 \ldots v_n$

Assume $D[i, j]$ are the costs for an optimal alignment of substrings $u_1 \ldots u_i$ and $v_1 \ldots v_j$, $1 \leq i \leq m$, $1 \leq j \leq n$

$i = 0$: alignment empty string $\varepsilon$ and $v_1 \ldots v_j$

$j = 0$: alignment $u_1 \ldots u_i$ and empty string $\varepsilon$

Init: $D[i, 0] = i; D[0, j] = j, i, j \geq 0$

Compute

$$D[i, j] = \min \left\{ \begin{array}{c} D[i - 1, j] + \delta(u_i, -) \\ D[i - 1, j - 1] + \delta(u_i, v_j) \\ D[i, j - 1] + \delta(-, v_j) \end{array} \right\}$$

($\delta = \text{unit-cost-function}$)

Lemma

$D[m, n] = \text{cost of optimal alignment between } u \text{ and } v$. 
Backtracing

Given the strings $u = u_1 \ldots u_m$ and $v = v_1 \ldots v_n$

$Tracematrix$ is an $m \times n$ matrix with $T[i,j] \subseteq \{\leftarrow, \nearrow, \uparrow\}$.

Init: $T[0,0] = \emptyset, T[i,0] = \uparrow, T[0,j] = \leftarrow$ for $1 \leq i \leq m, 1 \leq j \leq n$

$\uparrow \in T[i,j]$ if $D[i-1,j] + \delta(u_i,-)$

Set: $\nearrow \in T[i,j]$ if $D[i-1,j-1] + \delta(u_i,v_j)$

$\leftarrow \in T[i,j]$ if $D[i,j-1] + \delta(-,v_j)$

Runtime: $O(mn)$
Alignment with variable Gap-Costs

ACCGTCTGCT  ACCGTCTGCT  \( w(\mathcal{A}) = 5 \)
A--C--C--G--T  ACCGT------

This contradicts “biological intuition”:
Insertion of gap of length \( k \) is “evolutionary simpler to realize” then insertion of \( k \) gaps of length 1.

*gap penalty function* \( g : \mathbb{N} \rightarrow \mathbb{R} \)
\( g(k) \) is penalty for inserting a gap of length \( k \).
we need:
\[
g(k + l) \leq g(k) + g(l),
\]
as otherwise it might be better to insert 2 gaps of length \( k \) and \( l \) then one gap of length \( k + l \).
Alignment with variable Gap-Costs
(Smith-Waterman-Alg.)

Init: $D[0,0] = 0; D[0,k] = D[k,0] = g(k), k \geq 1;$

$$D[i,j] = \min \left\{ \begin{array}{l}
D[i-1,j-1] + \delta(u_i, v_j) \\
\min_{1 \leq k \leq i} D[i-k,j] + g(k) \\
\min_{1 \leq k \leq j} D[i,j-k] + g(k)
\end{array} \right. $$

*Tracematrix* is an $m \times n$ matrix with $T[i,j] \subseteq \{\leftarrow_k, \kappa, \uparrow_k, k \in \mathbb{N}\}$
Example Multiple Alignment

First 90 positions of a protein multiple sequence alignment of instances of the acidic ribosomal protein P0 (L10E) from several organisms. (wikipedia)
Distance VS Scoring Function

Note: Instead of using a distance matrix $D$ we can use a
Similarity/Scoring Matrix $S$ and maximize.

Init: $S[i, 0] = -i; S[0, j] = -j;$ for $i, j \geq 0$;
Compute

$$S[i, j] = \max \begin{cases} 
S[i - 1, j] + \delta(u_i, -) \\
S[i - 1, j - 1] + \delta(u_i, v_j) \\
S[i, j - 1] + \delta(-, v_j)
\end{cases}$$

with e.g.

$$\delta(a, b) = \begin{cases} 
1 & \text{if } a = b \\
-1 & \text{if } a \neq b \text{ and } a, b \neq - \\
-3 & \text{else (gap-costs)}
\end{cases}$$
Local vs Global Alignment

Needleman-Wunsch computes a global optimal Alignment
NW reasonable if sequences have almost same length
If sequences have quite different length, then the sequences are “shredded”:
R--------LCPMNLCGCSQ-----------------------KY
RCGEQGSNMECPNNLC--CSQYGYCGMGGDYGCGKGCQNGACWTSKR

Reason: gaps are penalized equally on each position
Reasonable: less penalization of gaps at end and beginning

Local Alignment: find best alignment of two substrings of two sequences (Smith-Waterman-Algorithm)
Smith-Waterman-Algorithm

Need scoring function that penalizes insertion/deletions with a negative value

Compute

\[
S[i, j] = \max \begin{cases} 
0, \\
S[i - 1, j] + \delta(u_i, -), \\
S[i - 1, j - 1] + \delta(u_i, v_j), \\
S[i, j - 1] + \delta(-, v_j)
\end{cases}
\]

with init: \( S[i, 0] = S[0, j] = 0 \); for \( i, j \geq 0 \);

First row states: we can start on each point a new alignment, if the score of the alignment computed so-far has a negative weight.

here maximize score. either minimize distance or maximize score
BLAST = Basic Local Alignment Search Tool

- quick heuristic alignment algorithm
- divides query sequences into short strings and initially only looks for (exact) matches of those strings in database strings. This is afterwards extended to get the entire alignment.
- much faster, but no optimality guarantee

Databases e.g. for nucleotide sequences (Genbank of NCBI, EMBL, ...) or protein databases (SwissProt, RefSeq, Pfam, ...).

BLAST homepage: blast.ncbi.nlm.nih.gov/
Tutorial: digitalworldbiology.com/BLAST
# BLAST “Types”

<table>
<thead>
<tr>
<th>type</th>
<th>query</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastn</td>
<td>nucleotide</td>
<td>nucleotide</td>
</tr>
<tr>
<td>blastp</td>
<td>protein</td>
<td>protein</td>
</tr>
<tr>
<td>blastx</td>
<td>nucleotide (transl)</td>
<td>protein</td>
</tr>
<tr>
<td>tblastn</td>
<td>protein</td>
<td>nucleotide (transl)</td>
</tr>
<tr>
<td>tblastx</td>
<td>nucleotide (transl)</td>
<td>nucleotide (transl)</td>
</tr>
</tbody>
</table>