CS 466
Introduction to Bioinformatics
Lecture 3

Mohammed El-Kebir
September 2, 2020
Outline

1. Edit distance recap
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.8 and 6.9
• Lecture notes
**Weighted Edit Distance – Practice Problem**

- Compute weighted edit distance between \( \mathbf{v} = \text{AGT} \) and \( \mathbf{w} = \text{ATCT} \).

\[
\begin{array}{cccccc}
\text{V} & \text{W} & 0 & 1 & 2 & 3 & 4 \\
\hline
\text{A} & 0 & 1 & 2 & 3 & 4 \\
\text{G} & 1 & 1 & 2 & 3 & 4 \\
\text{T} & 2 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
d[i, j] = \begin{cases} 
0, & \text{if } i = 0 \text{ and } j = 0, \\
\d[i-1, j] + 1, & \text{if } i > 0, \\
\d[i, j-1] + 1, & \text{if } j > 0, \\
\d[i-1, j-1] + 2, & \text{if } i > 0, j > 0 \text{ and } v_i \neq w_j, \\
\d[i-1, j-1], & \text{if } i > 0, j > 0 \text{ and } v_i = w_j.
\end{cases}
\]
Weighted Edit Distance – Practice Problem

• Compute weighted edit distance between \( v = \text{AGT} \) and \( w = \text{ATCT} \).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
d[i, j] = \min \begin{cases} 
0, & \text{if } i = 0 \text{ and } j = 0, \\
\min(d[i-1, j] + 1, \min(d[i, j-1] + 1, \min(d[i-1, j-1] + 2, d[i-1, j-1]))) & \text{if } i > 0 \text{ and } j > 0 \text{ and } v_i \neq w_j, \\
\min(d[i-1, j] + 1, d[i, j-1]) & \text{if } i > 0 \text{ and } j > 0 \text{ and } v_i = w_j. 
\end{cases}
\]
Edit Distance – Additional Insights

• An alignment corresponds to a series of elementary operations

Example
T−ACAT−
TGAT−AT

TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGA$\text{T}$AT $\xrightarrow{\text{del}}$ TGAT$\text{T}$ $\xrightarrow{\text{subst}}$ TGATA $\xrightarrow{\text{ins}}$ TGAT$\text{A}$T

Examples from http://profs.scienze.univr.it/~liptak/ACB/files/StringDistance_6up.pdf
Edit Distance – Additional Insights

• An alignment corresponds to a series of elementary operations

Example
T−ACAT−
TGAT−AT

TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT $\xrightarrow{\text{del}}$ TGATT $\xrightarrow{\text{subst}}$ TGATA $\xrightarrow{\text{ins}}$ TGATAT

• But not every series of elementary operations corresponds to an alignment! Why?

• TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT

• TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT

• TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGA$\xrightarrow{\text{subst}}$ TGATAT

Examples from http://profs.scienze.univr.it/~liptak/ACB/files/StringDistance_6up.pdf
A **distance function** (metric) on a set $X$ is a function $d : X \times X \to \mathbb{R}$ s.t. for all $x, y, z \in X$:

1. $d(x, y) \geq 0$ [non-negativity]
2. $d(x, y) = 0$ if and only if $x = y$ [identity of indiscernibles]
3. $d(x, y) = d(y, x)$ [symmetry]
4. $d(x, y) \leq d(x, z) + d(z, y)$ [triangle inequality]

**Question:** Is edit distance a distance function?
Edit Distance is a Distance Function

**Edit distance** $d(v, w)$ is the minimum number of **elementary operations** to transform $v \in \Sigma^*$ into $w \in \Sigma^*$.

**Claim**: edit distance is a distance function.

**Proof**: Let $u, v, w \in \Sigma^*$.

1. $d(v, w) \geq 0$  [non-negativity]
   
   Edit distance is defined by an alignment. This in turn uniquely determines a series of elementary operations, each with cost either 0 (match) or 1 (otherwise). Thus, $d(v, w) \geq 0$. 
Edit Distance is a Distance Function

**Edit distance** \(d(v, w)\) is the minimum number of **elementary operations** to transform \(v \in \Sigma^*\) into \(w \in \Sigma^*\).

**Claim:** edit distance is a distance function.

**Proof:** Let \(u, v, w \in \Sigma^*\).

1. \(d(v, w) = 0\) if and only if \(v = w\)  
   [identity of indiscernibles]

   
   (=>) By the premise, \(d(v, w) = 0\). By definition, the optimal alignment can only consist of operations with cost 0. That is, the alignment consist of only matches. Thus, \(v = w\).

   (<=) By the premise, \(v = w\). Thus, there exists an alignment where every pair of columns is a match. This means that \(|v| = |w|\) and each letter \(v_i\) equals \(w_i\) (where \(i \in [|v|]\)). Moreover, only the match operations has cost 0, the other operations have cost 1. Hence, this is the optimal alignment with cost \(d(v, w) = 0\).
**Claim:** edit distance is a distance function.

**Proof:** Let \( u, v, w \in \Sigma^* \).

**iii.** \( d(v, w) = d(w, v) \) [symmetry]

Let \( A = [a_{i,j}] \) be the optimal alignment corresponding to \( d(v, w) \), i.e. \( A \) is an \( 2 \times k \) matrix where \( k \in \{ \max(|v|, |w|), ..., |v| + |w| \} \). Define the function \( f(A) = B \) such that \( B \) is obtained by interchanging the two rows of \( A \). Since the cost of any insertion, deletion and mismatch is 1, we have that alignment \( B \) has cost \( d(v, w) \). The existence of an alignment from \( w \) to \( v \) with cost less than \( d(v, w) \), yields a contradiction as it implies that \( A \) is not an optimal alignment from \( v \) to \( w \). Hence, \( d(w, v) = d(v, w) \).
**Edit Distance is a Distance Function**

**Edit distance** $d(v, w)$ is the minimum number of **elementary operations** to transform $v \in \Sigma^*$ into $w \in \Sigma^*$.

**Claim:** edit distance is a distance function.

**Proof:** Let $u, v, w \in \Sigma^*$.

**iv.** $d(v, w) \leq d(v, u) + d(u, w)$  \[\text{[triangle inequality]}\]

Assume for a contradiction that $d(v, w) > d(v, u) + d(u, w)$. Let $S$ be the sequence of elementary operations for transforming $v$ into $u$. Let $S'$ be the sequence of elementary operations for transforming $u$ into $w$. Note that $d(v, u) = |S|$ and $d(u, w) = |S'|$. Concatenate $S$ and $S'$ and remove redundant operations, yielding sequence $S''$. By definition, $|S''| \leq |S| + |S'|$. We can obtain an alignment of $v$ and $w$ from $S''$ with cost $|S''| \leq d(v, u) + d(u, w)$. This yields a contradiction with $d(v, w) > d(v, u) + d(u, w)$ being the cost of the optimal alignment of $v$ and $w$. 
Outline

1. Edit distance recap
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.8 and 6.9
Biological Sequence Alignment

• Weighted edit distance: find alignment with minimum distance
  • Shortest path in weighted edit graph

• Sequence alignment: find alignment with maximum similarity
  • Longest path in weighted edit graph
  • Score function: \( \delta : (\Sigma \cup \{-\})^2 \rightarrow \mathbb{R} \)

**Question:** What is an example of \( \delta \)?
Scoring Matrices

**Transitions**: interchanges among purines (two rings) or pyrimidines (one ring)
- A ↔ G
- C ↔ T

**Transversions**: interchanges between purines (two rings) and pyrimidines (one ring)
- A ↔ C, A ↔ T
- G ↔ C, G ↔ T

Transitions more likely than transversions!
Scoring Matrices

**Transitions:** interchanges among purines (two rings) or pyrimidines (one ring)
- A $\leftrightarrow$ G
- C $\leftrightarrow$ T

**Transversions:** interchanges between purines (two rings) and pyrimidines (one ring)
- A $\leftrightarrow$ C, A $\leftrightarrow$ T
- G $\leftrightarrow$ C, G $\leftrightarrow$ T

Transitions more likely than transversions!
Global Alignment – Needleman-Wunsch Algorithm

**Global Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find alignment with maximum score.

- An alignment is a source-to-sink path in the edit graph
- An alignment $A = [a_{i,j}]$ is a $2 \times k$ matrix s.t. (i) $k = \{\max(m, n), \ldots, m + n\}$, (ii) $a_{i,j} \in \Sigma \cup \{-\}$ and (iii) there is no $j \in [k]$ where $a_{1,j} = a_{2,j} = -$.

$$s[i, j] = \max\begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i - 1, j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i, j - 1] + \delta(-, w_j), & \text{if } j > 0, \\ s[i - 1, j - 1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$
Demonstration

- \( \text{http://alfehrest.org/sub/nwa/index.html} \)

- \( \mathbf{v} = \text{ATGTTAT} \) and \( \mathbf{w} = \text{ATCGTAC} \).

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>T</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>G</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-( \infty )</td>
</tr>
</tbody>
</table>
Outline

1. Edit distance recap
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.7 and 6.9
• Lecture notes
Next Generation Sequencing (NGS) Technology

Cost per Genome

Log Scale

- $100M
- $10M
- $1M
- $100K
- $10K
- $1K


Moore’s Law

NGS

NIH National Human Genome Research Institute

genome.gov/sequencingcosts

November, 2017
NGS Characterized by Short Reads

Allow for inexact matches due to:
- Sequencing errors
- Polymorphisms/mutations in reference genome

**Genome**
Millions - billions nucleotides

**Next-generation DNA sequencing**

10-100’s million short reads

Short read: 100 nucleotides

... CATTCAGTAG ...
... AGCCATTAG ...
... GGTAGTTAG ...
... GGTAACTAG ...
... TATAATTAG ...
... CGTACCTAG ...
NGS Characterized by Short Reads

Genome
Millions -billions nucleotides

Next-generation DNA sequencing

10-100’s million short reads
Short read: 100 nucleotides

Allow for inexact matches due to:
• Sequencing errors
• Polymorphisms/mutations in reference genome

Human reference genome is 3,300,000,000 nucleotides, while a short read is 100 nucleotides. Global sequence alignment will not work!

**Question**: How to account for discrepancy between lengths of reference and short read?
For short read alignment, we want to align complete short read $\mathbf{v} \in \Sigma^m$ to substring of reference genome $\mathbf{w} \in \Sigma^n$. Note that $m \ll n$.

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function $\delta$, find a alignment of $\mathbf{v}$ and a substring of $\mathbf{w}$ with maximum global alignment score $s^*$ among all global alignments of $\mathbf{v}$ and all substrings of $\mathbf{w}$.
Fitting Alignment – Naive Approach

**Fitting Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find an alignment of $v$ and a substring of $w$ with maximum global alignment score $s^*$ among all global alignments of $v$ and all substrings of $w$.

- Consider all contiguous non-empty substrings of $w$, defined by $1 \leq i \leq j \leq n$.
- How many?
Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function $\delta$, find an alignment of $\mathbf{v}$ and a substring of $\mathbf{w}$ with maximum global alignment score $s^*$ among all global alignments of $\mathbf{v}$ and all substrings of $\mathbf{w}$

- Consider all contiguous non-empty substrings of $\mathbf{w}$, defined by $1 \leq i \leq j \leq n$
- How many? Answer: $n + \binom{n}{2}$
- What are their total lengths?
- What is the running time?
Fitting Alignment – Dynamic Programming

**Fitting Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find an alignment of $v$ and a substring of $w$ with maximum global alignment score $s^*$ among all global alignments of $v$ and all substrings of $w$

$$s[i, j] = \max \begin{cases} 0, & \text{if } i = 0, \\ s[i - 1, j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i, j - 1] + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\ s[i - 1, j - 1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \\ \end{cases}$$

$$s^* = \max \{s[m, 0], \ldots, s[m, n]\}$$
## Fitting Alignment – Dynamic Programming

**Fitting Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find an alignment of $v$ and a substring of $w$ with maximum global alignment score $s^*$ among all global alignments of $v$ and all substrings of $w$.

<table>
<thead>
<tr>
<th>$v$ \ $w$</th>
<th>0</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\begin{align*}
s[i, j] &= \max \begin{cases} 0, & \text{Start anywhere on first row if } i = 0, \\ s[i - 1, j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i, j - 1] + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\ s[i - 1, j - 1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \\ \end{cases} \\

s^* &= \max \{s[m, 0], \ldots, s[m, n]\} \quad \text{End anywhere on last row}
\end{align*}$
**Fitting Alignment problem:** Given strings \( \mathbf{v} \in \Sigma^m \) and \( \mathbf{w} \in \Sigma^n \) and scoring function \( \delta \), find an alignment of \( \mathbf{v} \) and a substring of \( \mathbf{w} \) with maximum global alignment score \( s^* \) among all global alignments of \( \mathbf{v} \) and all substrings of \( \mathbf{w} \).

\[
s[i, j] = \begin{cases} 
0, & \text{if } i = 0, \\
\max\{s[i - 1, j] + \delta(v_i, -), s[i, j - 1] + \delta(-, w_j), s[i - 1, j - 1] + \delta(v_i, w_j)\}, & \text{if } i > 0 \text{ and } j > 0.
\end{cases}
\]

\[
s^* = \max\{s[m, 0], \ldots, s[m, n]\}
\]

**Question:** Let match score be 1, mismatch/indel score be -1. What is \( s^* \)?

**Question:** Same scores. What is optimal global alignment and score?
Fitting Alignment – Dynamic Programming

• Online: https://valiec.github.io/AlignmentVisualizer/index.html

\[
s_{i, j} = \max \begin{cases} 
0, & \text{if } i = 0, \\
0, & \text{if } j = 0, \\
s_{i-1, j} + \delta(v_i, -), & \text{if } i > 0, \\
s_{i, j-1} + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\
s_{i-1, j-1} + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0.
\end{cases}
\]

\[s^* = \max\{s[m, 0], \ldots, s[m, n]\}\]

**Question:** Let match score be 1, mismatch/indel score be -1. What is $s^*$?

**Question:** Same scores. What is optimal global alignment and score?
Outline

1. Edit distance
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.8 and 6.9
• Lecture notes
Proteins are composed of functional units called domains. Such domains may occur in different proteins even across species.

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function $\delta$, find a substring of $\mathbf{v}$ and a substring of $\mathbf{w}$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $\mathbf{v}$ and $\mathbf{w}$.
Global Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find alignment of $v$ and $w$ with maximum score.

Fitting Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find an alignment of $v$ and a substring of $w$ with maximum global alignment score $s^*$ among all global alignments of $v$ and all substrings of $w$.

Local Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find a substring of $v$ and a substring of $w$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $v$ and $w$. 
Local Alignment – Naive Algorithm

Local Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find a substring of $v$ and a substring of $w$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $v$ and $w$

Brute force:
1. Generate all pairs $(v', w')$ of substrings of $v$ and $w$
2. For each pair $(v', w')$, solve global alignment problem.

Question: There are $\binom{m}{2}\binom{n}{2}$ pairs of substrings. But they have different lengths. What is the running time?
Key Idea

**Global alignment:**
- Start at \((0,0)\) and end at \((m,n)\)

**Local alignment:**
- Start and end anywhere

---

Figure 6.16 (a) Global and (b) local alignments of two hypothetical genes that each have a conserved domain. The local alignment has a much worse score according to the global scoring scheme, but it correctly locates the conserved domain.
Local Alignment Recurrence

**Local Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find a substring of $v$ and a substring of $w$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $v$ and $w$

$$s[i, j] = \max \begin{cases} 
0, & \text{if } i = 0 \text{ and } j = 0, \\
 s[i - 1, j] + \delta(v_i, -), & \text{if } i > 0, \\
 s[i, j - 1] + \delta(-, w_j), & \text{if } j > 0, \\
 s[i - 1, j - 1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0.
\end{cases}$$

$$s^* = \max_{i, j} s[i, j]$$

**Figure 6.16** (a) Global and (b) local alignments of two hypothetical genes that each have a conserved domain. The local alignment has a much worse score according to the global scoring scheme, but it correctly locates the conserved domain.
Local Alignment Recurrence

**Local Alignment problem:** Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find a substring of $v$ and a substring of $w$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $v$ and $w$

\[
s[i,j] = \begin{cases} 
0, & \text{Start anywhere if } i = 0 \text{ and } j = 0, \\
\max(s[i-1,j] + \delta(v_i,-), s[i,j-1] + \delta(-,w_j), s[i-1,j-1] + \delta(v_i,w_j)), & \text{if } i > 0 \text{ and } j > 0.
\end{cases}
\]

\[s^* = \max_{i,j} s[i,j] \quad \text{End anywhere}\]

**Running time:** $O(mn)$

---

Figure 6.16 (a) Global and (b) local alignments of two hypothetical genes that each have a conserved domain. The local alignment has a much worse score according to the global scoring scheme, but it correctly locates the conserved domain.
Local Alignment – Dynamic Programming

- Online: https://valiec.github.io/AlignmentVisualizer/index.html

**Question:** Let match score be 2, mismatch score be -2 and indel be -4. What is $s^*$?
Global, Fitting and Local Alignment

**Global Alignment problem:** Given strings \( v \in \Sigma^m \) and \( w \in \Sigma^n \) and scoring function \( \delta \), find alignment of \( v \) and \( w \) with maximum score.

**Fitting Alignment problem:** Given strings \( v \in \Sigma^m \) and \( w \in \Sigma^n \) and scoring function \( \delta \), find an alignment of \( v \) and a substring of \( w \) with maximum global alignment score \( s^* \) among all global alignments of \( v \) and all substrings of \( w \).

**Local Alignment problem:** Given strings \( v \in \Sigma^m \) and \( w \in \Sigma^n \) and scoring function \( \delta \), find a substring of \( v \) and a substring of \( w \) whose alignment has maximum global alignment score \( s^* \) among all global alignments of all substrings of \( v \) and \( w \).
Outline

1. Edit distance
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.8 and 6.9
• Lecture notes
Scoring Gaps

Let $\mathbf{v} = \text{AAC}$ and $\mathbf{w} = \text{ACAGGC}$

**Match** $\delta(c, c) = 1$;
**Mismatch** $\delta(c, d) = -1$ (where $c \neq d$); **Indel** $\delta(c, -) = \delta(-, c) = -2$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>-</th>
<th>-</th>
<th>A</th>
<th>C</th>
<th></th>
<th>A</th>
<th>-</th>
<th>A</th>
<th>-</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>v</strong></td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td></td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td><strong>w</strong></td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td></td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

Both alignments have 3 matches and 2 indels.
Score: $(3 \times 1) + (2 \times -2) = -1$
Let \( \mathbf{v} = \text{AAC} \) and \( \mathbf{w} = \text{ACAGGC} \).

Match \( \delta(c, c) = 1 \);
Mismatch \( \delta(c, d) = -1 \) (where \( c \neq d \)); Indel \( \delta(c, -) = \delta(-, c) = -2 \).

Both alignments have 3 matches and 2 indels.

**Score:** \( (3 \times 1) + (2 \times -2) = -1 \)

**Question:** Which alignment is better?
Scoring Gaps – Affine Gap Penalties

**Desired:** Lower penalty for consecutive gaps than interspersed gaps.

**Why:** Consecutive gaps are more likely due to slippage errors in DNA replication (2-3 nucleotides), codons for protein sequences, etc.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>–</th>
<th>–</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>–</th>
<th>A</th>
<th>–</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>
**Desired:** Lower penalty for consecutive gaps than interspersed gaps.

**Why:** Consecutive gaps are more likely due to slippage errors in DNA replication (2-3 nucleotides), codons for protein sequences, etc.

**Affine gap penalty:** Two penalties: (i) gap open penalty $\rho \geq 0$ and (ii) gap extension penalty $\sigma \geq 0$. Stretch of $k$ consecutive gaps has score $-(\rho + \sigma k)$. 

---

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>-</th>
<th>-</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>-</th>
<th>A</th>
<th>-</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>
### Scoring Gaps – Affine Gap Penalties

**Desired:** Lower penalty for consecutive gaps than interspersed gaps.

**Why:** Consecutive gaps are more likely due to slippage errors in DNA replication (2-3 nucleotides), codons for protein sequences, etc.

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>–</th>
<th>–</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>–</th>
<th>A</th>
<th>–</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

**Affine gap penalty:** Two penalties: (i) gap open penalty \( \rho \geq 0 \) and (ii) gap extension penalty \( \sigma \geq 0 \). Stretch of \( k \) consecutive gaps has score \(-(\rho + \sigma k)\).

Let \( \rho = 10 \) and \( \sigma = 1 \). Left: \((3 \times 1) - (10 + 1 \times 2) = -9\).

Right: \((3 \times 1) - (10 + 1 \times 1) - (10 + 1 \times 1) = -19\).
Affine Gap Penalty Alignment – Naive Approach

**Affine gap penalty:** Two penalties: (i) gap open penalty $\rho \geq 0$ and (ii) gap extension penalty $\sigma \geq 0$. Stretch of $k$ consecutive gaps has score $-(\rho + \sigma k)$.

**Idea:** Insert horizontal (deletion) and vertical (insertion) edges spanning $k > 1$ gaps with score $-(\rho + \sigma k)$. 
**Affine gap penalty:** Two penalties: (i) gap open penalty $\rho \geq 0$ and (ii) gap extension penalty $\sigma \geq 0$. Stretch of $k$ consecutive gaps has score $-(\rho + \sigma k)$.

**Idea:** Insert horizontal (deletion) and vertical (insertion) edges spanning $k > 1$ gaps with score $-(\rho + \sigma k)$.

**Question:** What’s the recurrence?

**Question:** What’s the running time?
Affine Gap Penalty Alignment

**Idea:** Three separate recurrences:

(i) Gap in first sequence $s^\rightarrow[i, j]

(ii) Match/mismatch $s^\downarrow[i, j]

(iii) Gap in second sequence $s^\uparrow[i, j]$
Affine Gap Penalty Alignment

**Idea:** Three separate recurrences:

(i) Gap in first sequence \( s^{-}[i, j] \)

(ii) Match/mismatch \( s^{\downarrow}[i, j] \)

(iii) Gap in second sequence \( s^{\uparrow}[i, j] \)

\[
\begin{align*}
s^{-}[i, j] &= \max \left\{ s^{-}[i, j-1] - \sigma, \quad \text{if } j > 1, \\
& \quad s^{\downarrow}[i, j-1] - (\sigma + \rho), \quad \text{if } j > 0, \\
& \quad 0, \quad \text{if } i = 0 \text{ and } j = 0, \\
& \quad s^{-}[i, j], \quad \text{if } j > 0, \\
& \quad s^{\downarrow}[i, j], \quad \text{if } i > 0, \\
& \quad s^{\downarrow}[i-1, j-1] + \delta(v_i, w_j), \quad \text{if } i > 0 \text{ and } j > 0,
\end{align*}
\]

\[
\begin{align*}
s^{\downarrow}[i, j] &= \max \left\{ s^{\downarrow}[i-1, j] - \sigma, \quad \text{if } i > 1, \\
& \quad s^{\downarrow}[i-1, j] - (\sigma + \rho), \quad \text{if } i > 0.
\end{align*}
\]
Affine Gap Penalty Alignment

**Idea:** Three separate recurrences:

(i) Gap in first sequence $s\rightarrow[i, j]$

(ii) Match/mismatch $s\nwarrow[i, j]$

(iii) Gap in second sequence $s\uparrow[i, j]$

$$s\rightarrow[i, j] = \max \begin{cases} s\rightarrow[i, j-1] - \sigma, & \text{if } j > 1, \\ s\nwarrow[i, j-1] - (\sigma + \rho), & \text{if } j > 0, \end{cases}$$

$$s\nwarrow[i, j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s\rightarrow[i, j], & \text{if } j > 0, \\ s\uparrow[i, j], & \text{if } i > 0, \\ s\nwarrow[i-1, j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0, \end{cases}$$

$$s\uparrow[i, j] = \max \begin{cases} s\uparrow[i-1, j] - \sigma, & \text{if } i > 1, \\ s\nwarrow[i-1, j] - (\sigma + \rho), & \text{if } i > 0. \end{cases}$$

**Running time:** $O(mn)$
Affine Gap Penalty Alignment – Example

Let $\rho = 10$ and $\sigma = 1$. Match = 1. Mismatch = -1

$v = \text{AAC} \quad w = \text{ACAAC}$

$$s_{\rightarrow}[i, j] = \max \begin{cases} s_{\rightarrow}[i, j - 1] - \sigma, & \text{if } j > 1, \\ s_{\leftarrow}[i, j - 1] - (\sigma + \rho), & \text{if } j > 0, \\ 0, & \text{if } i = 0 \text{ and } j = 0, \\ s_{\rightarrow}[i, j], & \text{if } j > 0, \\ s_{\leftarrow}[i, j], & \text{if } i > 0, \\ s_{\leftarrow}[i - 1, j - 1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0, \end{cases}$$

$$s_{\leftarrow}[i, j] = \max \begin{cases} s_{\rightarrow}[i - 1, j] - \sigma, & \text{if } i > 1, \\ s_{\rightarrow}[i - 1, j] - (\sigma + \rho), & \text{if } i > 0. \end{cases}$$
Gapped Alignment – Additional Insights

• Naive approach supports arbitrary gap penalties given two sequences \( \mathbf{v} \in \Sigma^m \) and \( \mathbf{w} \in \Sigma^n \). This results in an \( O(mn(m + n)) \) algorithm.

• Alignment with convex gap penalties given two sequences \( \mathbf{v} \in \Sigma^m \) and \( \mathbf{w} \in \Sigma^n \) can be computed in \( O(mn \log m) \) time.

Take Home Messages

1. Edit distance
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment

Reading:
• Jones and Pevzner. Chapters 6.6, 6.8 and 6.9
• Lecture notes