CS 466
Introduction to Bioinformatics
Lecture 4

Mohammed El-Kebir

September 6, 2019

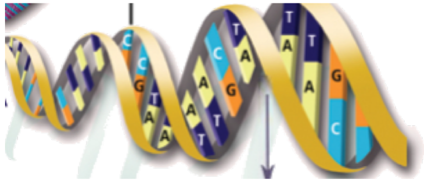
Outline

1. Fitting alignment
2. Local alignment
3. Gapped alignment
4. BLOSUM scoring matrix

Reading:

- Jones and Pevzner. Chapters 6.6-6.9
- Lecture notes

NGS Characterized by Short Reads



Genome
Millions -billions nucleotides

Next-generation DNA sequencing

… CATTCAGTAG …

… AGCCATTAG …

… GGTAGTTAG …

… GGTAAACTAG …

… TATAATTAG …

… CGTACCTAG …

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

Allow for inexact matches due to:

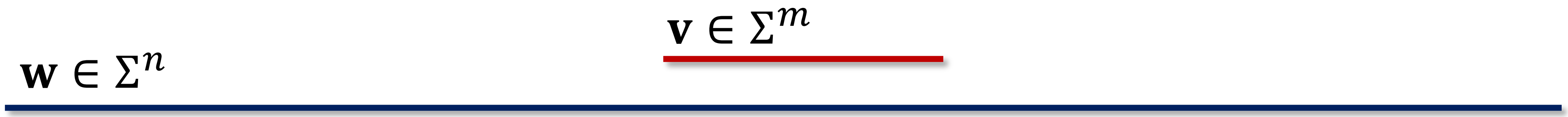
- Sequencing errors
- Polymorphisms/mutations in reference genome

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?

Fitting Alignment

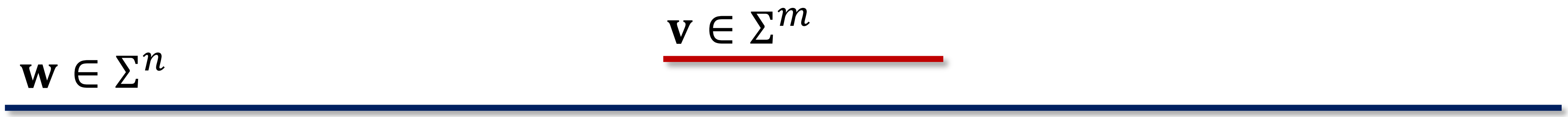
For short read alignment, we want to align complete short read $\mathbf{v} \in \Sigma^m$ to substring of reference genome $\mathbf{w} \in \Sigma^n$. Note that $m \ll n$.



Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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

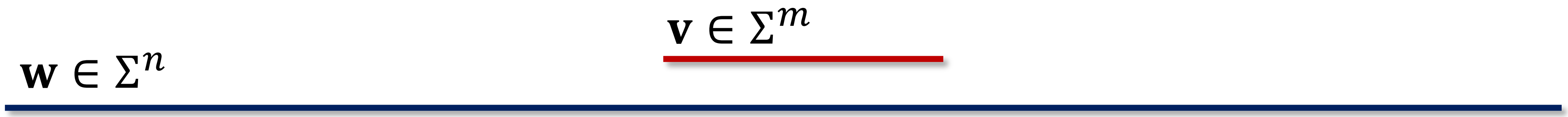
Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find an alignment of $\mathbf{v}$ and a substring of $\mathbf{w}$ with maximum global alignment score s^* among *all* global alignments of $\mathbf{v}$ and *all* substrings of $\mathbf{w}$



- Consider all contiguous non-empty substrings of $\mathbf{w}$, defined by $1 \leq i \leq j \leq n$
- How many?

Fitting Alignment – Naive Approach

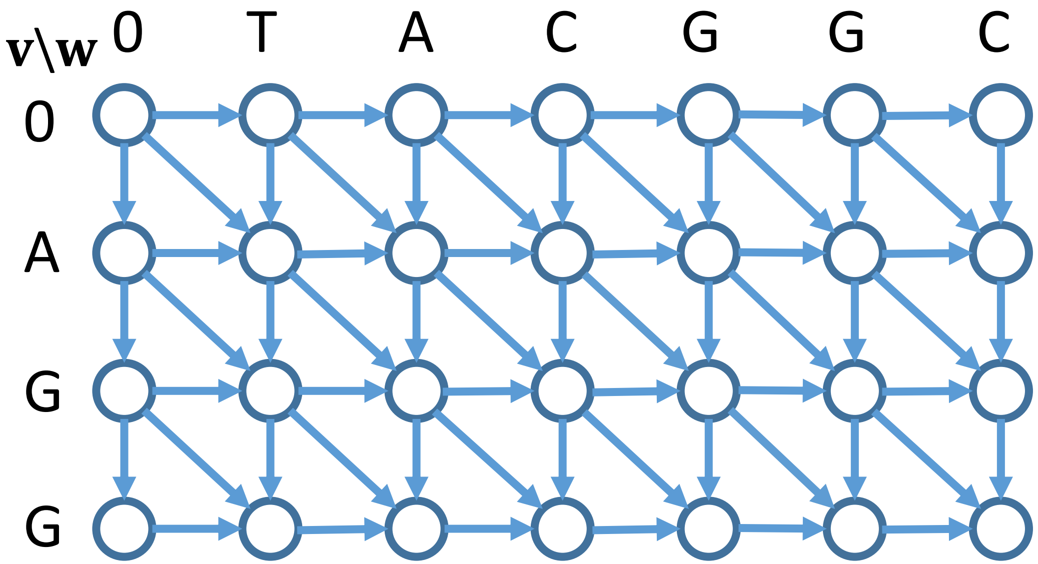
Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find an alignment of $\mathbf{v}$ and a substring of $\mathbf{w}$ with maximum global alignment score s^* among *all* global alignments of $\mathbf{v}$ and *all* substrings of $\mathbf{w}$



- 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 $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

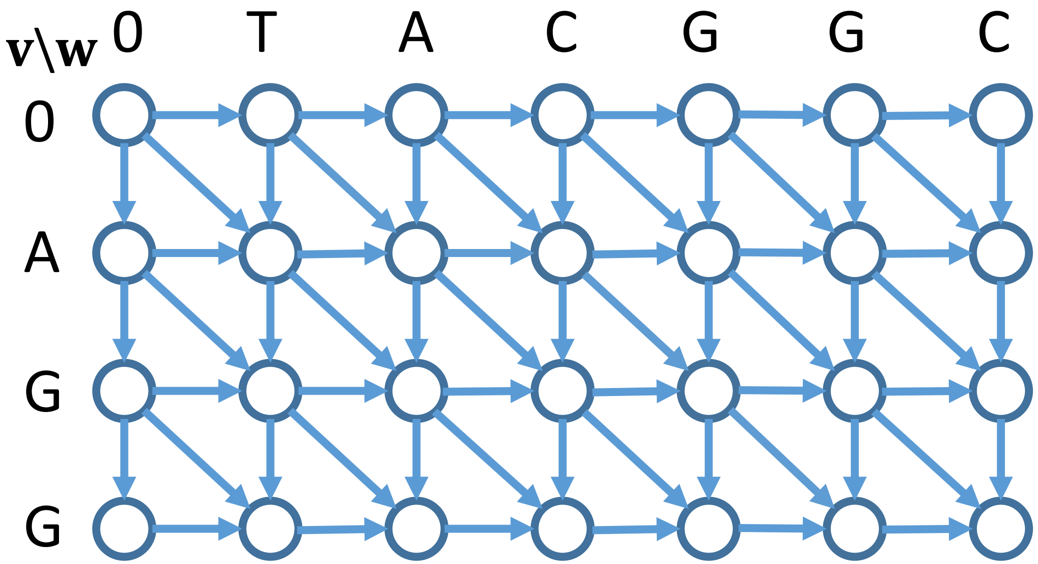


$$s[i,j] = \max \begin{cases} 0, & \text{if } i = 0, \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i,j-1] + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

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

Fitting Alignment – Dynamic Programming

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

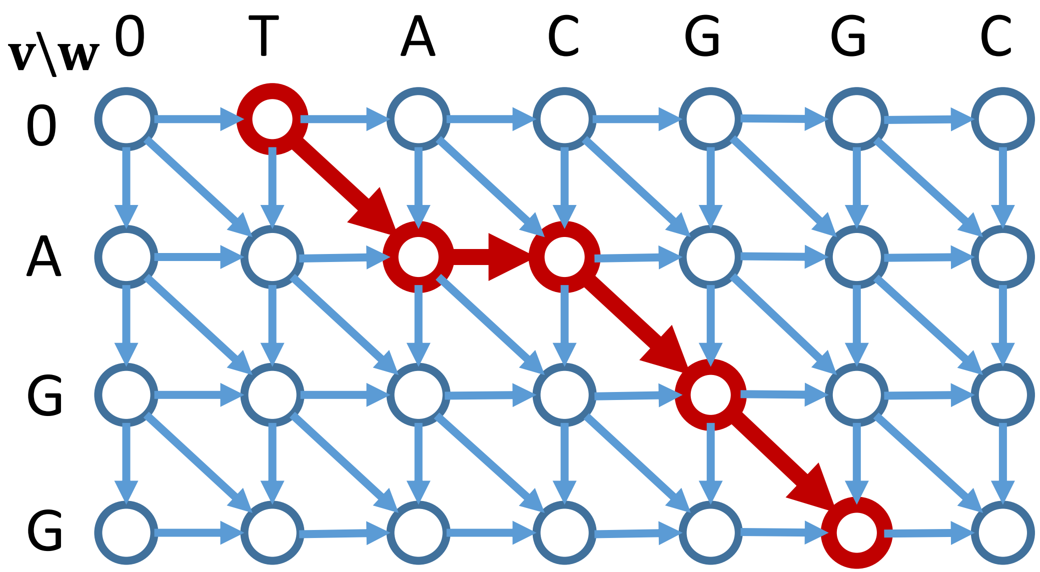


$$s[i,j] = \max \begin{cases} \boxed{0, \ \textbf{Start anywhere on first row} \ \text{if } i = 0,} & \\ s[i-1,j] + \delta(v_i, -), & \text{if } i > 0, \\ s[i,j-1] + \delta(-, w_j), & \text{if } i > 0 \text{ and } j > 0, \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

$$s^* = \boxed{\max\{s[m,0], \ldots, s[m,n]\} \ \ \textbf{End anywhere on last row}}$$

Fitting Alignment – Dynamic Programming

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$



v	–	A	–	G	G	–
w	T	A	C	G	G	C

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

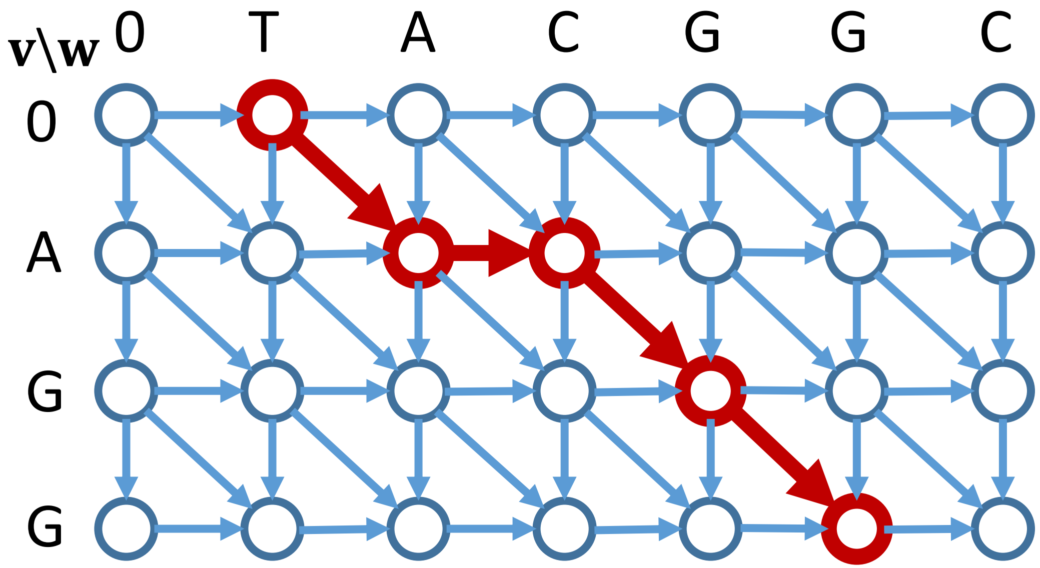
$$s^* = \max\{s[m,0], \ldots, s[m,n]\} \quad \textbf{End anywhere on last row}$$

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



V	-	A	-	G	G	-
W	T	A	C	G	G	C

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

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

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

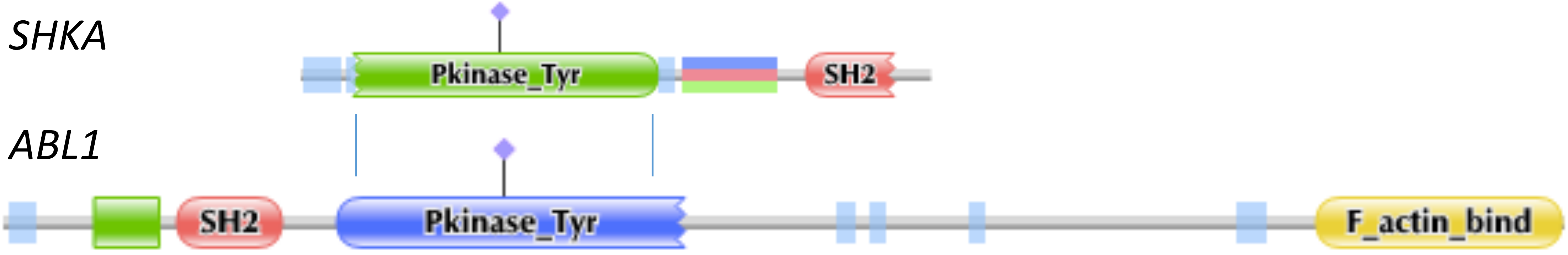
1. Fitting alignment
2. Local alignment
3. Gapped alignment
4. BLOSUM scoring matrix

Reading:

- Jones and Pevzner. Chapters 6.6-6.9
- Lecture notes

Local Alignment – Biological Motivation

Proteins are composed of functional units called domains. Such domains may occur in different proteins even across species.



From Pfam database (http://pfam.sanger.ac.uk/)

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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, Fitting and Local Alignment

Global Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment of $\mathbf{v}$ and $\mathbf{w}$ with maximum score.

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

Local Alignment – Naive Algorithm

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

Brute force:

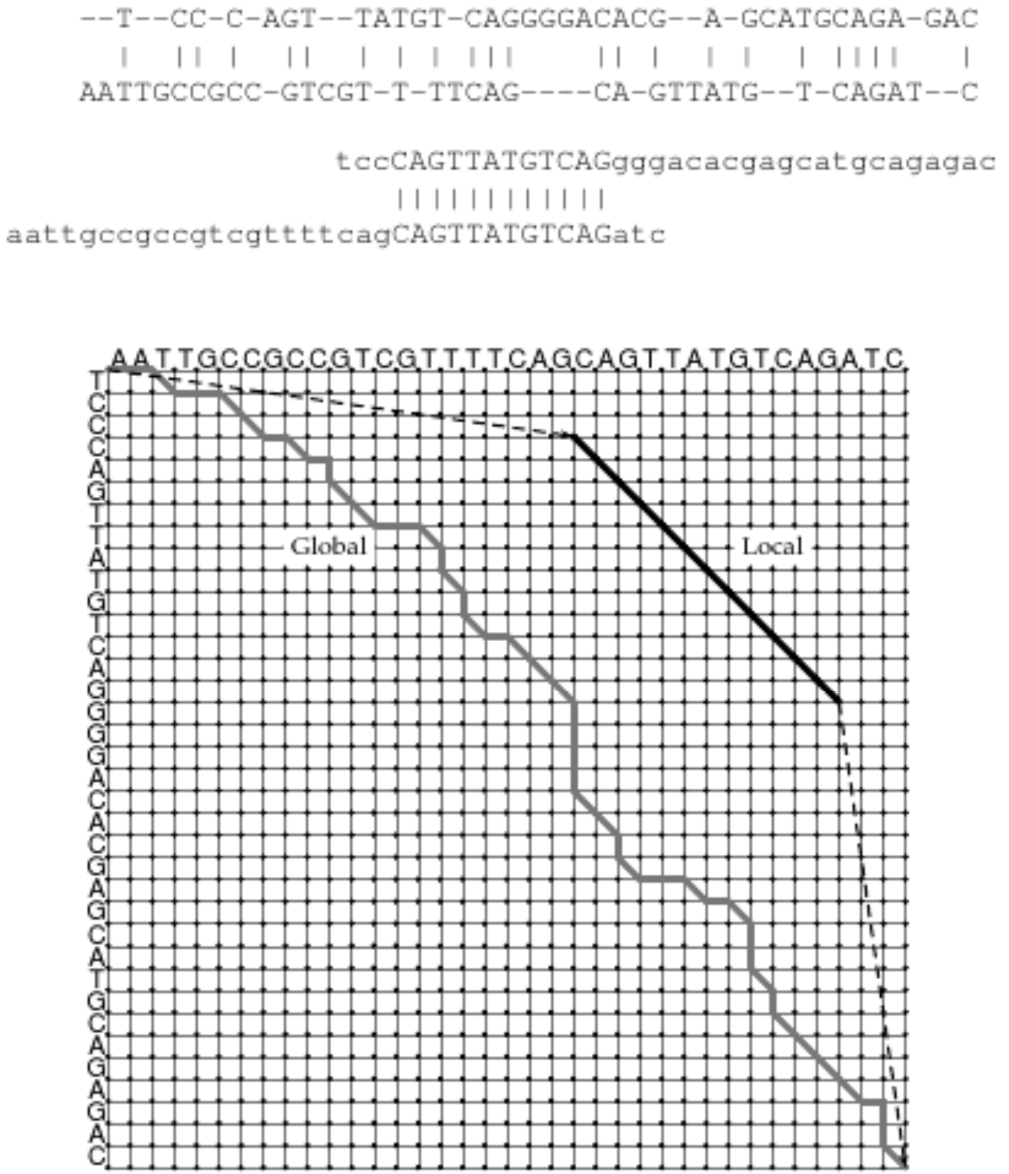
1. Generate all pairs $(\mathbf{v}', \mathbf{w}')$ of substrings of $\mathbf{v}$ and $\mathbf{w}$
2. For each pair $(\mathbf{v}', \mathbf{w}')$, solve global alignment problem.

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



```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || |  | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C

                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

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 $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

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

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

```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || |  | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C
```

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

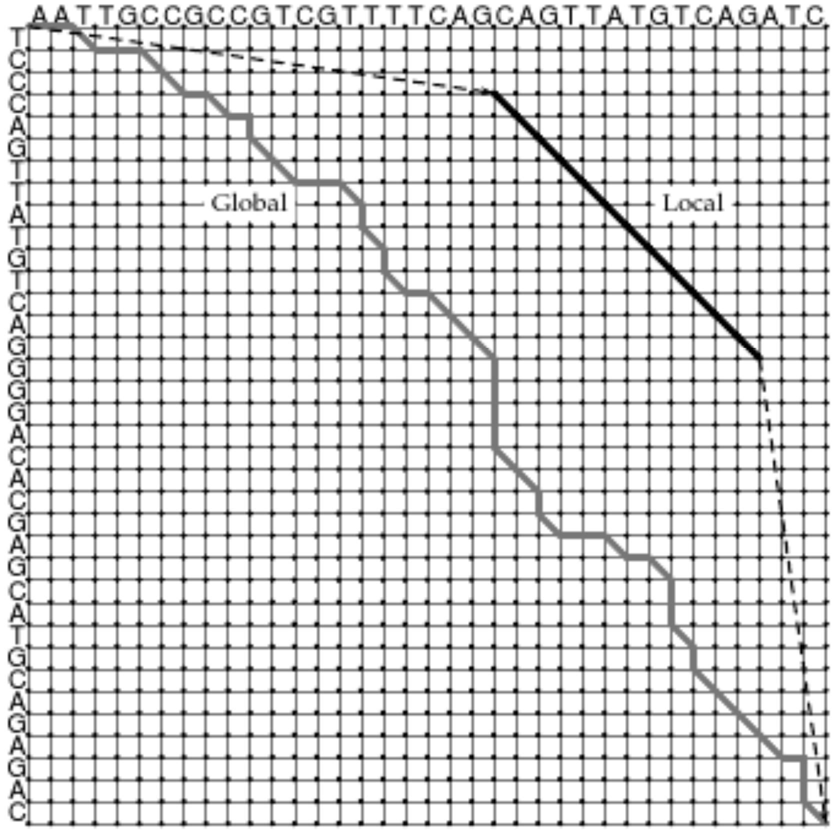


Figure 6.16 (a) Global and (b) local alignments of two hypothetical genes that each have a conserved domain. The local alignment has a much worse score according to the global scoring scheme, but it correctly locates the conserved domain.

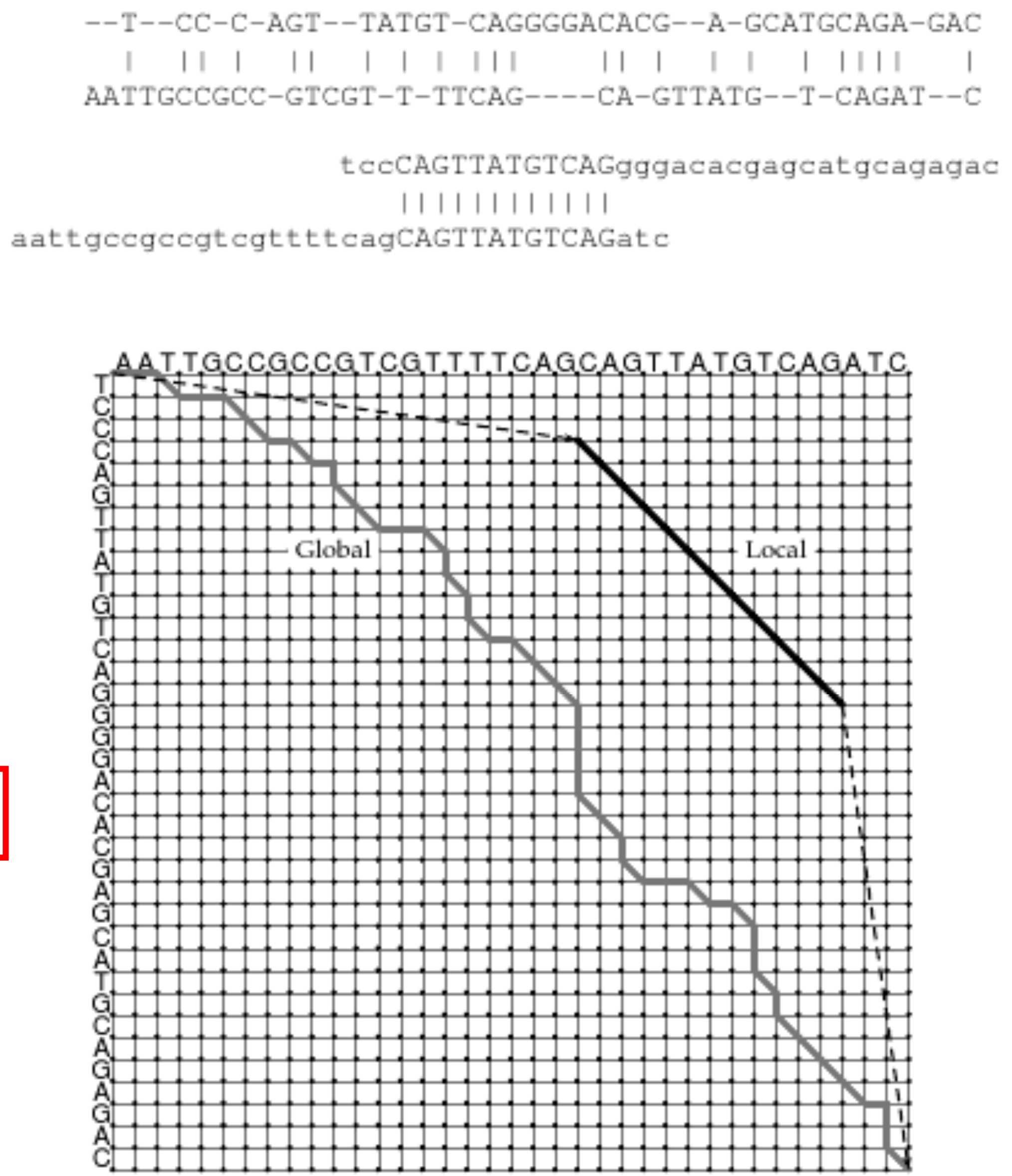
Local Alignment Recurrence

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

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

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

Running time: $O(mn)$



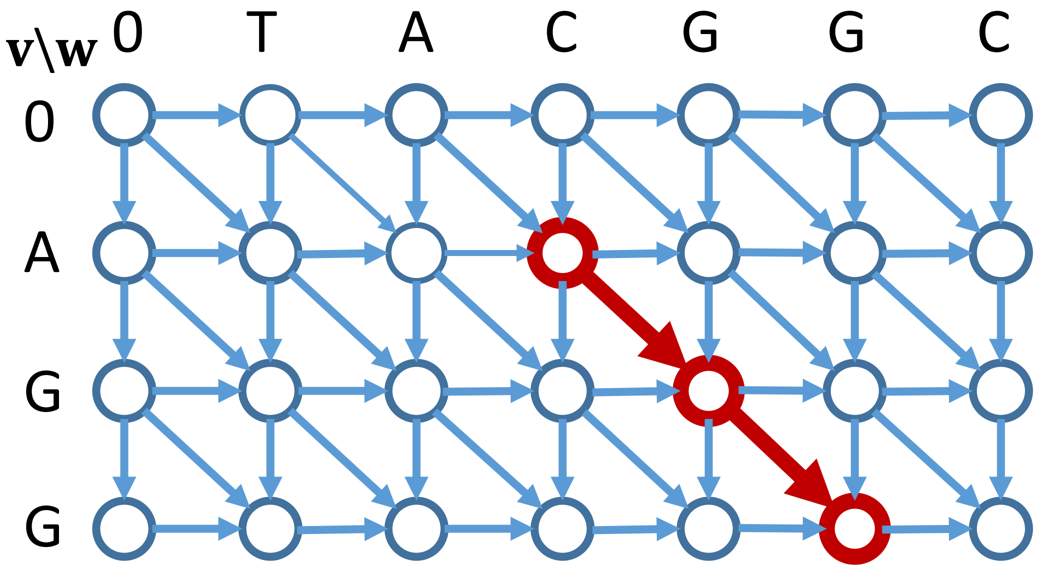
```
--T--CC-C-AGT--TATGT-CAGGGGACACG--A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || |  | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT--C
```

```
                  tccCAGTTATGTCAGggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

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



v	G	G
w	G	G

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

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

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 $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, find alignment of $\mathbf{v}$ and $\mathbf{w}$ with maximum score.

Fitting Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

Local Alignment problem: Given strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$ and scoring function δ, 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 $\mathbf{w}$

Outline

1. Fitting alignment
2. Local alignment
3. Gapped alignment
4. BLOSUM scoring matrix

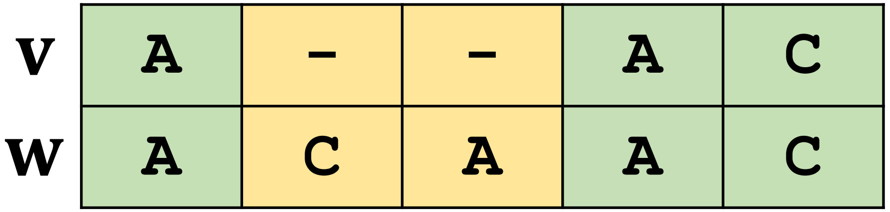
Reading:

- Jones and Pevzner. Chapters 6.6-6.9
- Lecture notes

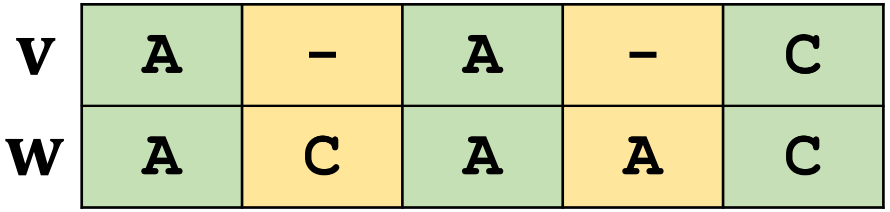
Scoring Gaps

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 $\delta(c, -) = \delta(-, c) = -2$



v	A	-	-	A	C
w	A	C	A	A	C



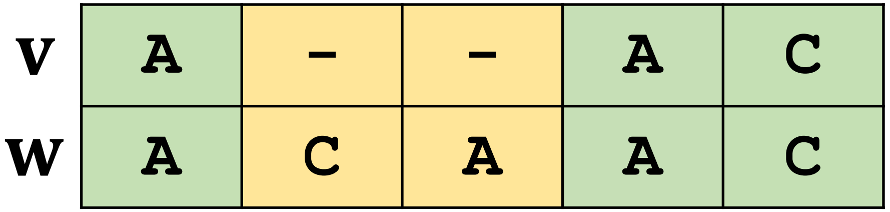
v	A	-	A	-	C
w	A	C	A	A	C

Both alignments have 3 matches and 2 indels.
Score: $(3 * 1) + (2 * -2) = -1$

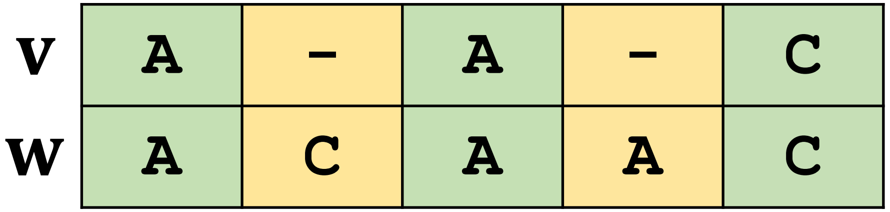
Scoring Gaps

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

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



v	A	-	-	A	C
w	A	C	A	A	C



v	A	-	A	-	C
w	A	C	A	A	C

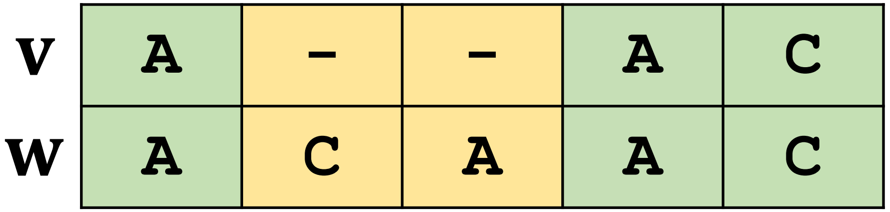
Both alignments have 3 matches and 2 indels.
Score: $(3 * 1) + (2 * -2) = -1$

Question: Which alignment is better?

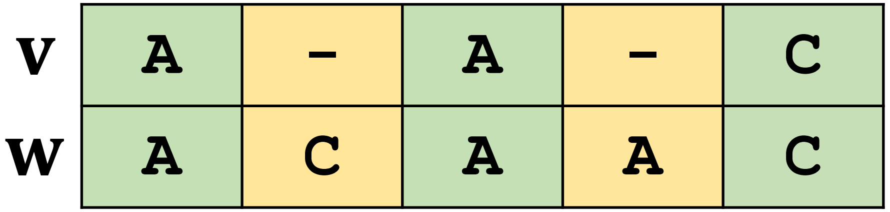
Scoring Gaps – Affine Gap Penalties

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.



v	`A`	`-`	`-`	`A`	`C`
w	`A`	`C`	`A`	`A`	`C`

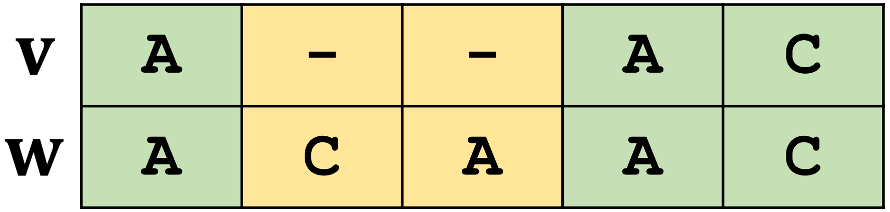


v	`A`	`-`	`A`	`-`	`C`
w	`A`	`C`	`A`	`A`	`C`

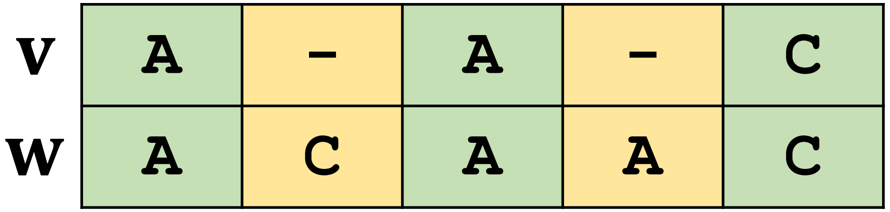
Scoring Gaps – Affine Gap Penalties

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.



v	`A`	`-`	`-`	`A`	`C`
w	`A`	`C`	`A`	`A`	`C`



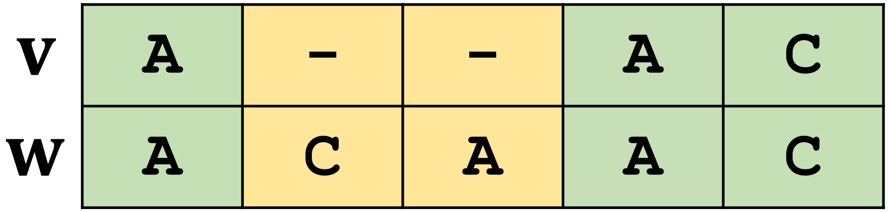
v	`A`	`-`	`A`	`-`	`C`
w	`A`	`C`	`A`	`A`	`C`

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 $-(\rho + \sigma k)$.

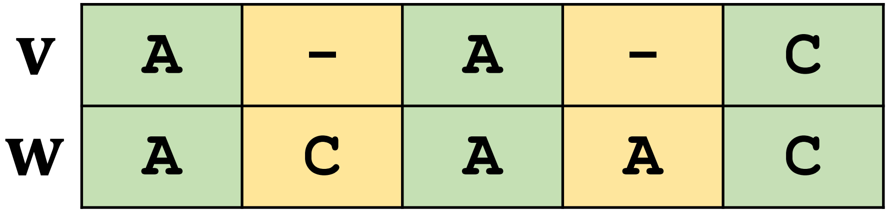
Scoring Gaps – Affine Gap Penalties

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.



v	A	-	-	A	C
w	A	C	A	A	C



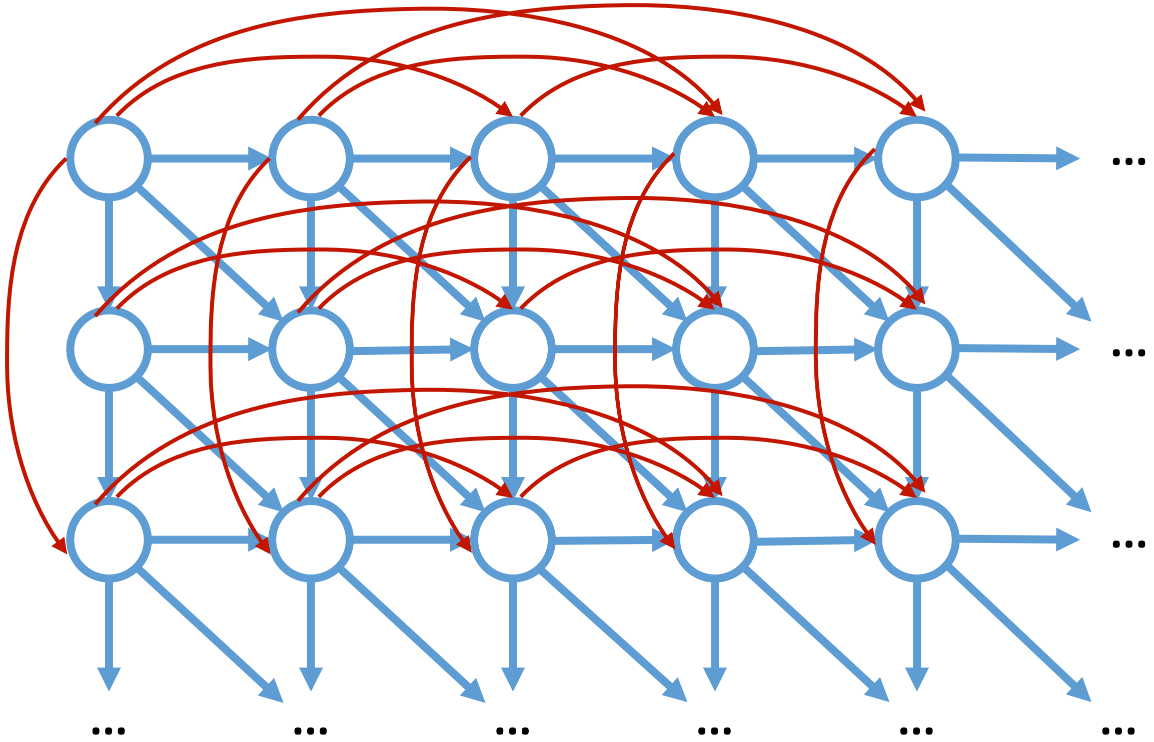
v	A	-	A	-	C
w	A	C	A	A	C

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: $(3 * 1) - (10 + 1 * 2) = -9$.
Right: $(3 * 1) - (10 + 1 * 1) - (10 + 1 * 1) = -19$.

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 $-(\rho + \sigma k)$.



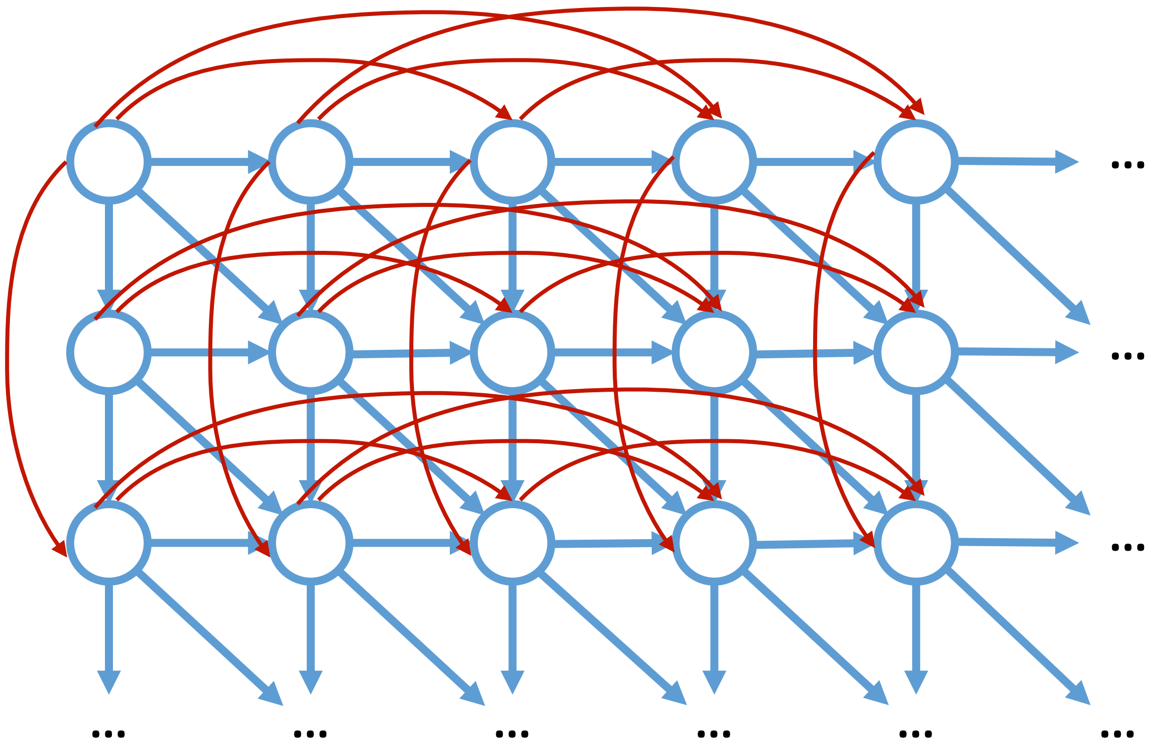
Idea: Insert horizontal (deletion) and vertical (insertion) edges spanning $k > 1$ gaps with score $-(\rho + \sigma k)$.

→ new edges

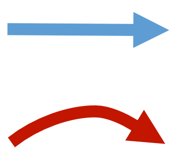
↷ old edges

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 $-(\rho + \sigma k)$.



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

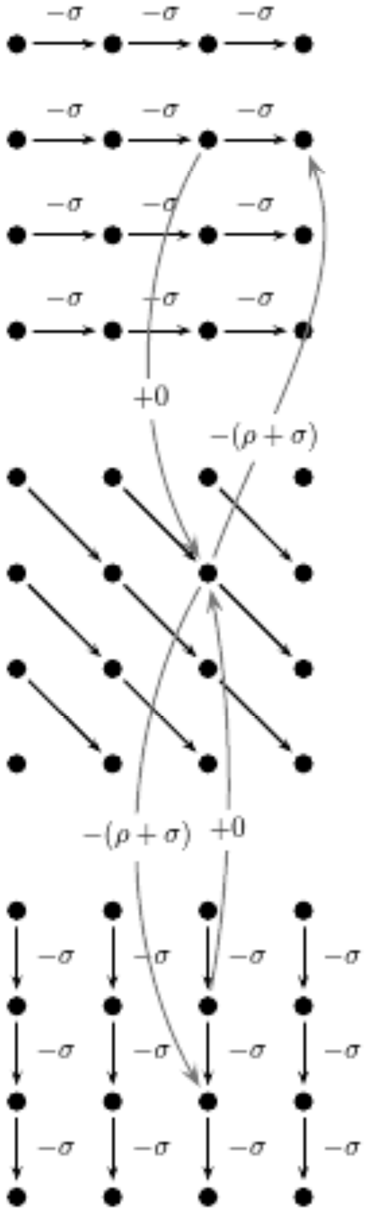


Figure 6.18 A three-level edit graph for alignment with affine gap penalties. Every vertex (i,j) in the middle level has one outgoing edge to the upper level, one outgoing edge to the lower level, and one incoming edge each from the upper and lower levels.

Affine Gap Penalty Alignment

Idea: Three separate recurrences:

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

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

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

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

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

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

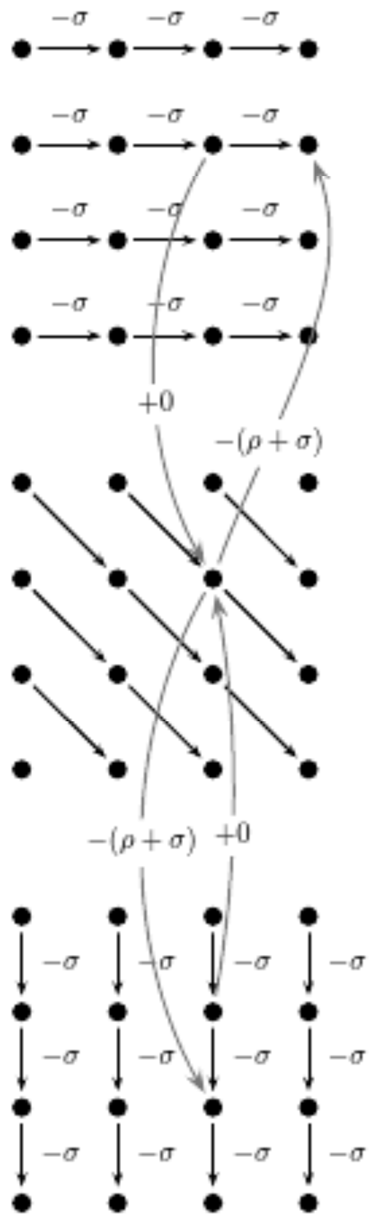


Figure 6.18 A three-level edit graph for alignment with affine gap penalties. Every vertex (i, j) in the middle level has one outgoing edge to the upper level, one outgoing edge to the lower level, and one incoming edge each from the upper and lower levels.

Affine Gap Penalty Alignment

Idea: Three separate recurrences:
(i) Gap in first sequence $s^{\rightarrow}[i,j]$
(ii) Match/mismatch $s^{\searrow}[i,j]$
(iii) Gap in second sequence $s^{\downarrow}[i,j]$

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

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

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

Running time: $O(mn)$

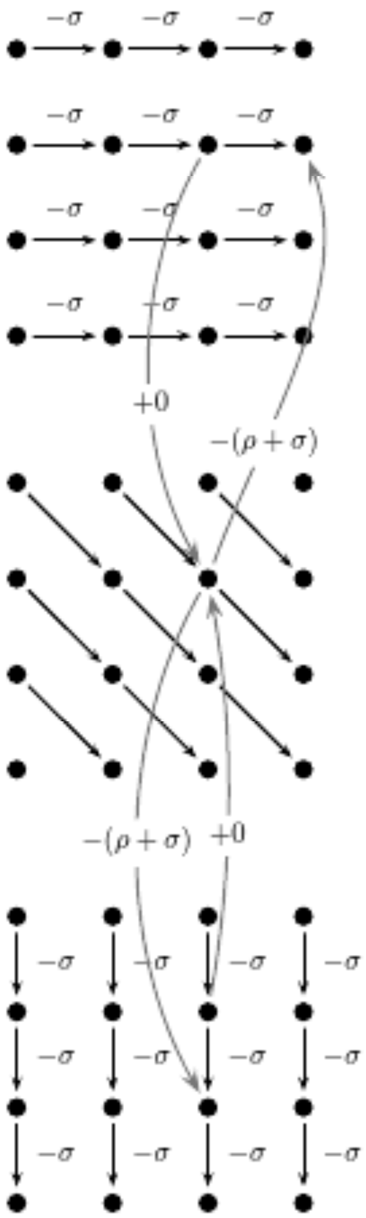


Figure 6.18 A three-level edit graph for alignment with affine gap penalties. Every vertex (i, j) in the middle level has one outgoing edge to the upper level, one outgoing edge to the lower level, and one incoming edge each from the upper and lower levels.

Affine Gap Penalty Alignment – Example

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

$\mathbf{v} = \text{AAC}$

$\mathbf{w} = \text{ACAAC}$

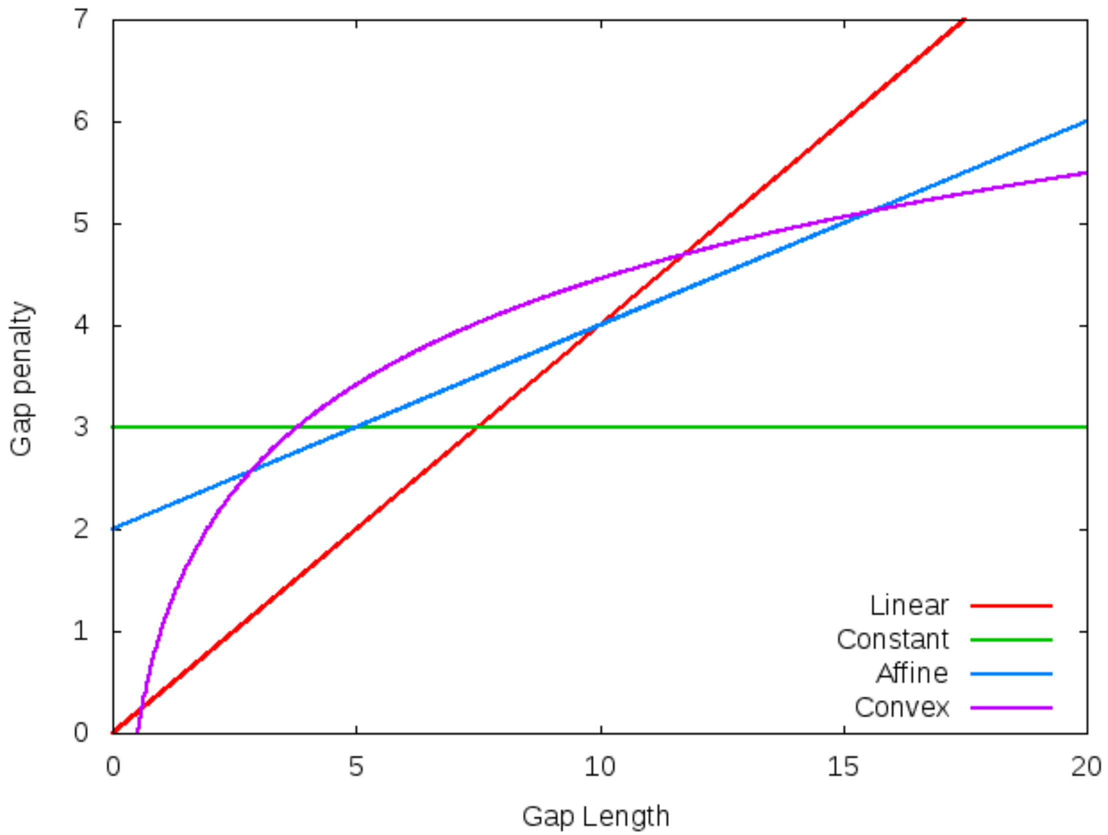
$$s^{\rightarrow}[i,j] = \max \begin{cases} s^{\rightarrow}[i,j-1] - \sigma, & \text{if } j > 1, \\ s^{\searrow}[i,j-1] - (\sigma + \rho), & \text{if } j > 0, \end{cases}$$

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

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

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.
  See: Dan Gusfield. 1997. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, NY, USA.



Outline

1. Fitting alignment
2. Local alignment
3. Gapped alignment
4. BLOSUM scoring matrix

Reading:

- Jones and Pevzner. Chapters 6.6-6.9
- Lecture notes

Substitution Matrices

- Given a pair $(\mathbf{v}, \mathbf{w})$ of aligned sequences, we want to assign a score that measure the **relative likelihood** that the sequences are **related** as opposed to being **unrelated**
- We need two models:
  - Random model R: each letter $a \in \Sigma$ occurs independently with probability q_a
  - Match model M: aligned pair $(a, b) \in \Sigma \times \Sigma$ occur with joint probability $p_{a,b}$

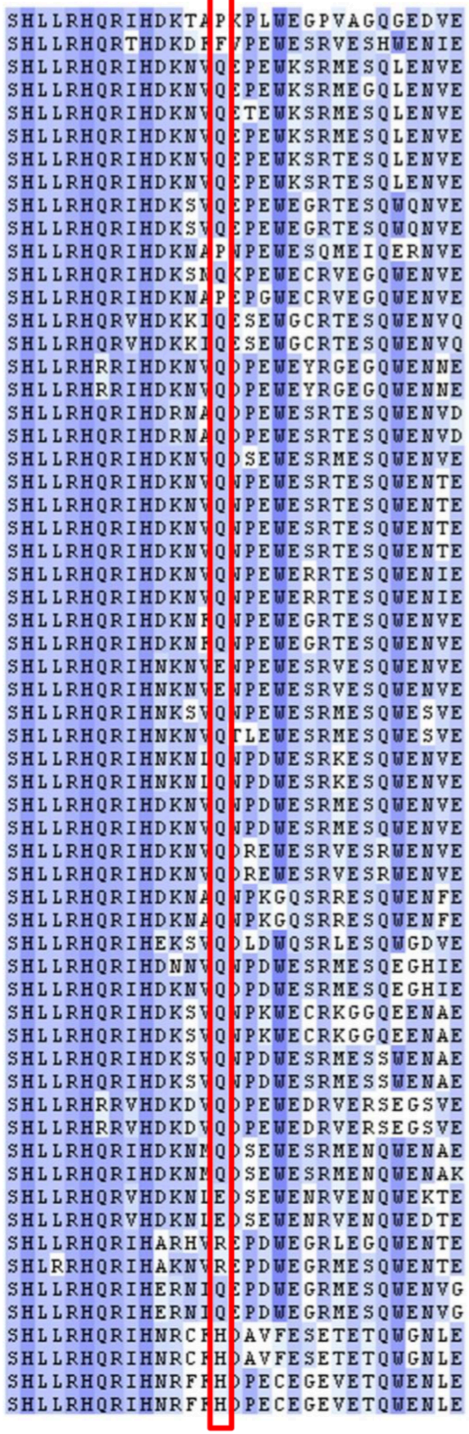
$$\Pr(\mathbf{v}, \mathbf{w} | R) = \prod_i q_{v_i} \cdot \prod_i q_{w_i}$$

$$\Pr(\mathbf{v}, \mathbf{w} | M) = \prod_i p_{v_i, w_i}$$

$$\log \frac{\Pr(\mathbf{v}, \mathbf{w} | M)}{\Pr(\mathbf{v}, \mathbf{w} | R)} = \sum_i s(v_i, w_i) \text{ where } s(a, b) = \log \frac{p_{a,b}}{q_a q_b}$$

BLOSUM (Blocks Substitution Matrices)

- Henikoff and Henikoff, 1992
- Computed using **ungapped** alignments of protein segments (blocks) from BLOCKS database
- Thousands of such blocks go into computing a single BLOSUM matrix
- Example of a one such block (right):
  - 31 positions (columns)
  - 61 sequences (rows)
- Given threshold L, block is pruned down to largest set C of sequences that have at least $L\%$ sequence identity to another sequence in C
  - How to compute C?

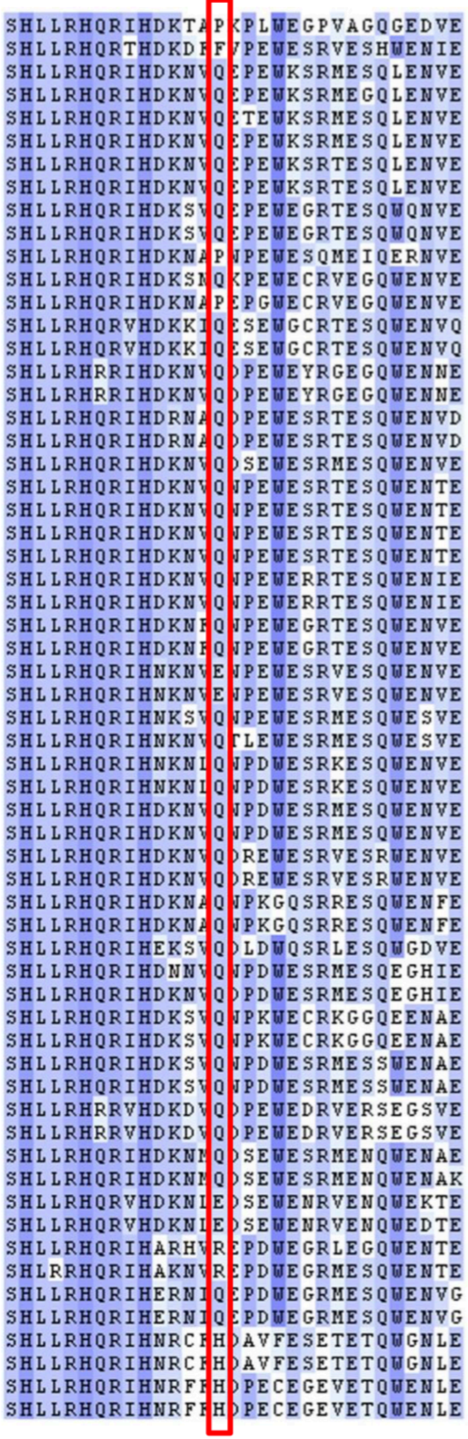


```
SHLLRHQRIHDKTAPKPLWEGPVAGQGEDVE
SHLLRHQRTHDKDFFVPEWESRVESHWENIE
SHLLRHQRIHDKNVQEPEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRMEGQLENVE
SHLLRHQRIHDKNVQETEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRTESQLENVE
SHLLRHQRIHDKNVQEPEWKSRTESQLENVE
SHLLRHQRIHDKSVQEPEWEGRTESQWQNVE
SHLLRHQRIHDKSVQEPEWEGRTESQWQNVE
SHLLRHQRIHDKNAPNPEWESQMEIQERNVE
SHLLRHQRIHDKSNQKPEWECRVEGQWENVE
SHLLRHQRIHDKNAPEPGWECRVEGQWENVE
SHLLRHQRVHDKKIQESEWGCRTESQWENVQ
SHLLRHQRVHDKKIQESEWGCRTESQWENVQ
SHLLRHRRIHDKNVQDPEWEYRGEGQWENNE
SHLLRHRRIHDKNVQDPEWEYRGEGQWENNE
SHLLRHQRIHDRNAQDPEWESRTESQWENVD
SHLLRHQRIHDRNAQDPEWESRTESQWENVD
SHLLRHQRIHDKNVQDSEWESRMESQWENVE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWERRTESQWENIE
SHLLRHQRIHDKNVQNPEWERRTESQWENIE
SHLLRHQRIHDKNFQNPEWEGRTESQWENVE
SHLLRHQRIHDKNFQNPEWEGRTESQWENVE
SHLLRHQRIHNKNVENPEWESRVESQWENVE
SHLLRHQRIHNKNVENPEWESRVESQWENVE
SHLLRHQRIHNKSVQNPEWESRMESQWESVE
SHLLRHQRIHNKNVQTLEWESRMESQWESVE
SHLLRHQRIHNKNLQNPDWESRKESQWENVE
SHLLRHQRIHNKNLQNPDWESRKESQWENVE
SHLLRHQRIHDKNVQNPDWESRMESQWENVE
SHLLRHQRIHDKNVQNPDWESRMESQWENVE
SHLLRHQRIHDKNVQDREWESRVESRWENVE
SHLLRHQRIHDKNVQDREWESRVESRWENVE
SHLLRHQRIHDKNAQNPKGQSRRESQWENFE
SHLLRHQRIHDKNAQNPKGQSRRESQWENFE
SHLLRHQRIHEKSVQDLDWQSRLESQWGDVE
SHLLRHQRIHDNNVQNPDWESRMESQEGHIE
SHLLRHQRIHDKNVQDPDWESRMESQEGHIE
SHLLRHQRIHDKSVQNPKWECRKGGQEENAE
SHLLRHQRIHDKSVQNPKWECRKGGQEENAE
SHLLRHQRIHDKSVQNPDWESRMESSWENAE
SHLLRHQRIHDKSVQNPDWESRMESSWENAE
SHLLRHRRVHDKDVQDPEWEDRVERSEGSVE
SHLLRHRRVHDKDVQDPEWEDRVERSEGSVE
SHLLRHQRIHDKNMQDSEWESRMENQWENAE
SHLLRHQRIHDKNMQDSEWESRMENQWENAK
SHLLRHQRVHDKNLEDSEWENRVENQWEKTE
SHLLRHQRVHDKNLEDSEWENRVENQWEDTE
SHLLRHQRIHARHVREPDWEGRLEGQWENTE
SHLRRHQRIHAKNVREPDWEGRMESQWENTE
SHLLRHQRIHERNIQEPDWEGRMESQWENVG
SHLLRHQRIHERNIQEPDWEGRMESQWENVG
SHLLRHQRIHNRCKHDAVFESETETQWGNLE
SHLLRHQRIHNRCKHDAVFESETETQWGNLE
SHLLRHQRIHNRFKHDPECEGEVETQWENLE
SHLLRHQRIHNRFKHDPECEGEVETQWENLE
```

BLOSUM (Blocks Substitution Matrices)

$$\log\frac{\Pr(\mathbf{v},\mathbf{w}|\,M)}{\Pr(\mathbf{v},\mathbf{w}|\,R)} = \sum_i s(v_i,w_i) \text{ where } s(a,b) = \frac{1}{\lambda}\log\frac{p_{a,b}}{q_a q_b}$$

- Null model frequencies $q_a q_b$ of letters a and b:
  - Count the number of occurrences of a (b) in all blocks
  - Divide by sum of lengths of each block (sequences * positions)
- Match model frequency $p_{a,b}$:
  - Count the number of pairs (a,b) in all columns of all blocks
  - Divide by the total number of pairs of columns:
    - $\sum_C n(C)\binom{m(C)}{2}$
    - $m(C)$ is the number of sequences in block C
    - $n(C)$ is the number of positions in block C



```
SHLLRHQRIHDKTAPKPLWEGPVAGQGEDVE
SHLLRHQRTHDKDFFVPEWESRVESHWENIE
SHLLRHQRIHDKNVQEPEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRMEGQLENVE
SHLLRHQRIHDKNVQETEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRMESQLENVE
SHLLRHQRIHDKNVQEPEWKSRTESQLENVE
SHLLRHQRIHDKNVQEPEWKSRTESQLENVE
SHLLRHQRIHDKSVQEPEWEGRTESQWQNVE
SHLLRHQRIHDKSVQEPEWEGRTESQWQNVE
SHLLRHQRIHDKNAPNPEWESQMEIQERNVE
SHLLRHQRIHDKSNQKPEWECRVEGQWENVE
SHLLRHQRIHDKNAPEPGWECRVEGQWENVE
SHLLRHQRVHDKKIQESEWGCRTESQWENVQ
SHLLRHQRVHDKKIQESEWGCRTESQWENVQ
SHLLRHRRIHDKNVQDPEWEYRGEGQWENNE
SHLLRHRRIHDKNVQDPEWEYRGEGQWENNE
SHLLRHQRIHDRNAQDPEWESRTESQWENVD
SHLLRHQRIHDRNAQDPEWESRTESQWENVD
SHLLRHQRIHDKNVQDSEWESRMESQWENVE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWESRTESQWENTE
SHLLRHQRIHDKNVQNPEWERRTESQWENIE
SHLLRHQRIHDKNVQNPEWERRTESQWENIE
SHLLRHQRIHDKNFQNPEWEGRTESQWENVE
SHLLRHQRIHDKNFQNPEWEGRTESQWENVE
SHLLRHQRIHNKNVENPEWESRVESQWENVE
SHLLRHQRIHNKNVENPEWESRVESQWENVE
SHLLRHQRIHNKSVQNPEWESRMESQWESVE
SHLLRHQRIHNKNVQTLEWESRMESQWESVE
SHLLRHQRIHNKNLQNPDWESRKESQWENVE
SHLLRHQRIHNKNLQNPDWESRKESQWENVE
SHLLRHQRIHDKNVQNPDWESRMESQWENVE
SHLLRHQRIHDKNVQNPDWESRMESQWENVE
SHLLRHQRIHDKNVQDREWESRVESRWENVE
SHLLRHQRIHDKNVQDREWESRVESRWENVE
SHLLRHQRIHDKNAQNPKGQSRRESQWENFE
SHLLRHQRIHDKNAQNPKGQSRRESQWENFE
SHLLRHQRIHEKSVQDLDWQSRLESQWGDVE
SHLLRHQRIHDNNVQNPDWESRMESQEGHIE
SHLLRHQRIHDKNVQDPDWESRMESQEGHIE
SHLLRHQRIHDKSVQNPKWECRKGGQEENAE
SHLLRHQRIHDKSVQNPKWECRKGGQEENAE
SHLLRHQRIHDKSVQNPDWESRMESSWENAE
SHLLRHQRIHDKSVQNPDWESRMESSWENAE
SHLLRHRRVHDKDVQDPEWEDRVERSEGSVE
SHLLRHRRVHDKDVQDPEWEDRVERSEGSVE
SHLLRHQRIHDKNMQDSEWESRMENQWENAE
SHLLRHQRIHDKNMQDSEWESRMENQWENAK
SHLLRHQRVHDKNLEDSEWENRVENQWEKTE
SHLLRHQRVHDKNLEDSEWENRVENQWEDTE
SHLLRHQRIHARHVREPDWEGRLEGQWENTE
SHLRRHQRIHAKNVREPDWEGRMESQWENTE
SHLLRHQRIHERNIQEPDWEGRMESQWENVG
SHLLRHQRIHERNIQEPDWEGRMESQWENVG
SHLLRHQRIHNRCRHDAVFESETETQWGNLE
SHLLRHQRIHNRCRHDAVFESETETQWGNLE
SHLLRHQRIHNRFRHDPECEGEVETQWENLE
SHLLRHQRIHNRFRHDPECEGEVETQWENLE
```

BLOSUM (Blocks Substitution Matrices)

$$\log \frac{\Pr(\mathbf{v},\mathbf{w}|\ M)}{\Pr(\mathbf{v},\mathbf{w}|\ R)} = \sum_i s(v_i, w_i) \text{ where } s(a,b) = \frac{1}{\lambda}\log\frac{p_{a,b}}{q_a q_b}$$

- Null model frequencies $q_a q_b$ of letters a and b:
  - Count the number of occurrences of a (b) in all blocks
  - Divide by sum of lengths of each block (sequences * positions)
- Match model frequency $p_{a,b}$:
  - Count the number of pairs (a, b) in all columns of all blocks
  - Divide by the total number of pairs of columns:
    - $\sum_C n(C)\binom{m(C)}{2}$
    - $m(C)$ is the number of sequences in block C
    - $n(C)$ is the number of positions in block C

Example: $(\lambda = 0.5)$

```
A A T
S A L
T A L
T A V
A A L
```

$$q_A = \frac{7}{15}$$

$$q_T = \frac{3}{15}$$

$$p_{A,T} = \frac{4}{30}$$

$$s(A,T) = 2 \cdot \log \frac{\frac{4}{30}}{\frac{7}{15}\cdot\frac{3}{15}} \approx 0.3$$

BLOSUM62

$$\log\frac{\Pr(\mathbf{v},\mathbf{w}\mid M)}{\Pr(\mathbf{v},\mathbf{w}\mid R)} = \sum_i s(v_i,w_i) \text{ where } s(a,b) = \frac{1}{\lambda}\log\frac{p_{a,b}}{q_a q_b}$$

Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2									
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4
	Ala	**Arg**	**Asn**	**Asp**	**Cys**	**Gln**	**Glu**	**Gly**	**His**	**Ile**	**Leu**	**Lys**	**Met**	**Phe**	**Pro**	**Ser**	**Thr**	**Trp**	**Tyr**	**Val**

This explains some details in BLOSUM62 that may seem counterintuitive at first glance. For instance, tryptophan (W/W) pairs score +11, while leucine (L/L) pairs only score +4; why shouldn't all identitites get the same score? The rarer the amino acid is, the more surprising it would be to see two of them align together by chance. In the homologous alignment data that BLOSUM62 was trained on, leucine/leucine (L/L) pairs were in fact more common than tryptophan/tryptophan (W/W) pairs (p_{LL} = 0.0371, p_{WW} = 0.0065), but tryptophan is a much rarer amino acid (f_L = 0.099, f_W = 0.013). Run those numbers (with BLOSUM62's original λ = 0.347) and you get +3.8 for L/L and +10.5 for W/W, which were rounded to +4 and +11.

https://doi.org/10.1038/nbt0804-1035

Take Home Messages

1. Edit distance
2. Global alignment
3. Fitting alignment
4. Local alignment
5. Gapped alignment
6. BLOSUM substitution matrix

Edit distance is shortest path in DAG

Global alignment is longest path in DAG

Small tweaks enable different extensions

Reading:

- Jones and Pevzner. Chapters 6.6-6.9
- Lecture notes