Distance metrics for Sequential Data

- Dot matrix
- Sequence Alignment (Dynamic programming)
- Dynamic Time Wrapping (DTW)
- q-Gram distance

What is sequential data ?

$$X = \{x_1, x_2, \ldots, x_N\} \quad x_i \in \Omega$$

- **Sequence of measurement (s)** of a quantity (or more) **over time**.
- There is **no need equally spaced time points**
- **Two types** of sequential data
  - **Continuous** time series (***Ω real values***)
  - **Categorical (discrete)** sequences (***Ω alphabet***)

Continuous time series

- **Stock market prices** (for a single stock, or for multiple stocks).
- **Heart rate** of a patient over time.
- **Position** of one or multiple people/cars/airplanes over time (trajectories).
- **Speech:** represented as a sequence of audio measurements at discrete time steps.
- A **musical melody**: represented as a sequence of pairs (note, duration).
- ….

Categorical (discrete) sequences

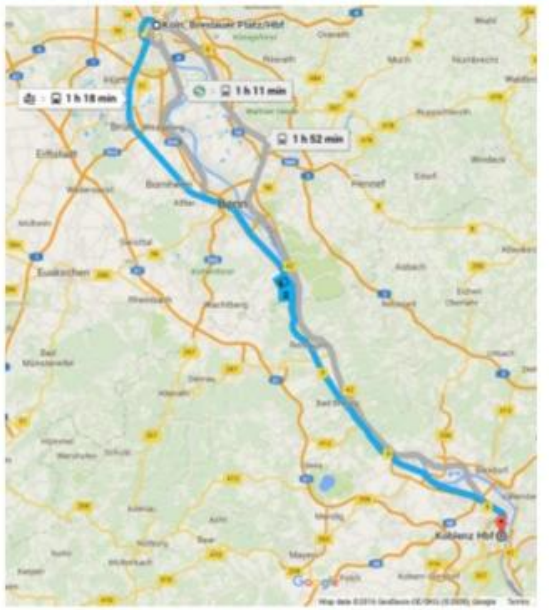
- Biological sequences (DNA / Protein)
- Web navigation
- Song listening sequences
- Sunny/Rainy weather sequences
- Mobility in a network of cities / regions
- ...

Examples

Stock price (Bitcoin)

Daily degrees in a region

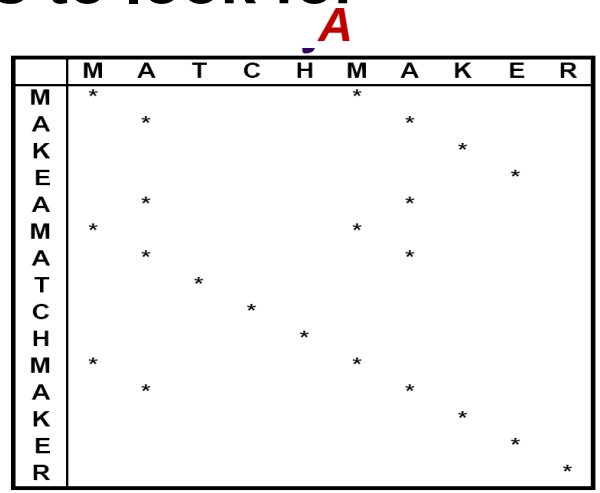
Human mobility



Dot matrix *(Gibbs and McIntyre 1970)*

- **Method for comparing two sequences to look for possible alignment**

- **Algorithm for a dot matrix:**



1. One sequence (A) is listed across the top of the matrix and the other (B) on the left side

2. Starting from the first character in B, one moves across first row and placing a dot in many column where the character in A is the same

3. The process is continued in rows until all possible comparisons between A and B are made

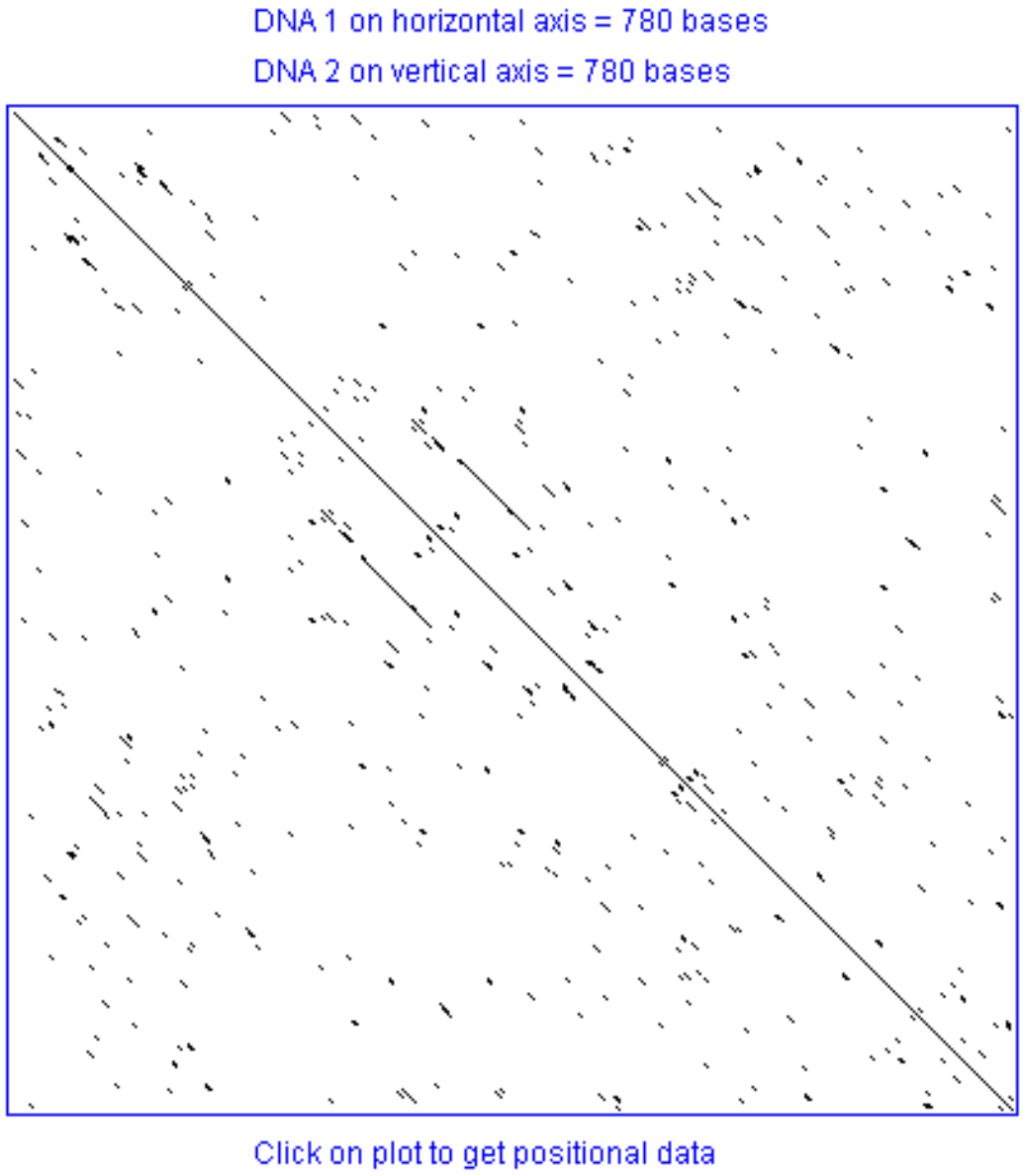
4. Any region of similarity is revealed by a diagonal row of dots

5. Isolated dots not on diagonal represent random matches

Dot matrix

- Improve visualization of identical regions among sequences by using sliding windows, instead of writing down a dot for every character, that is common in both sequences
- We compare a number of positions (window size), and we write down a dot whenever there is minimum number of identical characters

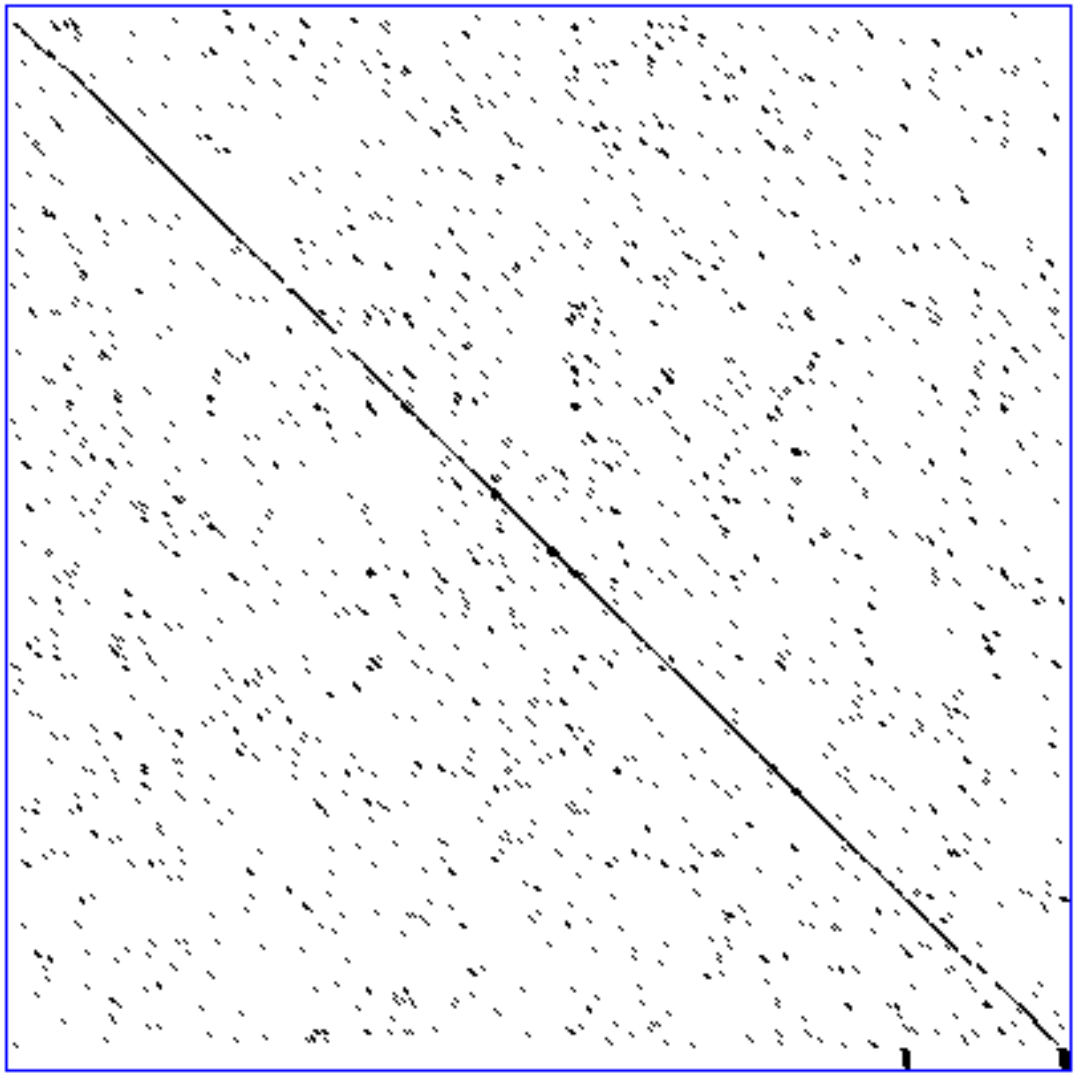
two identical sequences



two similar sequences sequences

DNA 1 on horizontal axis = 1348 bases

DNA 2 on vertical axis = 1365 bases

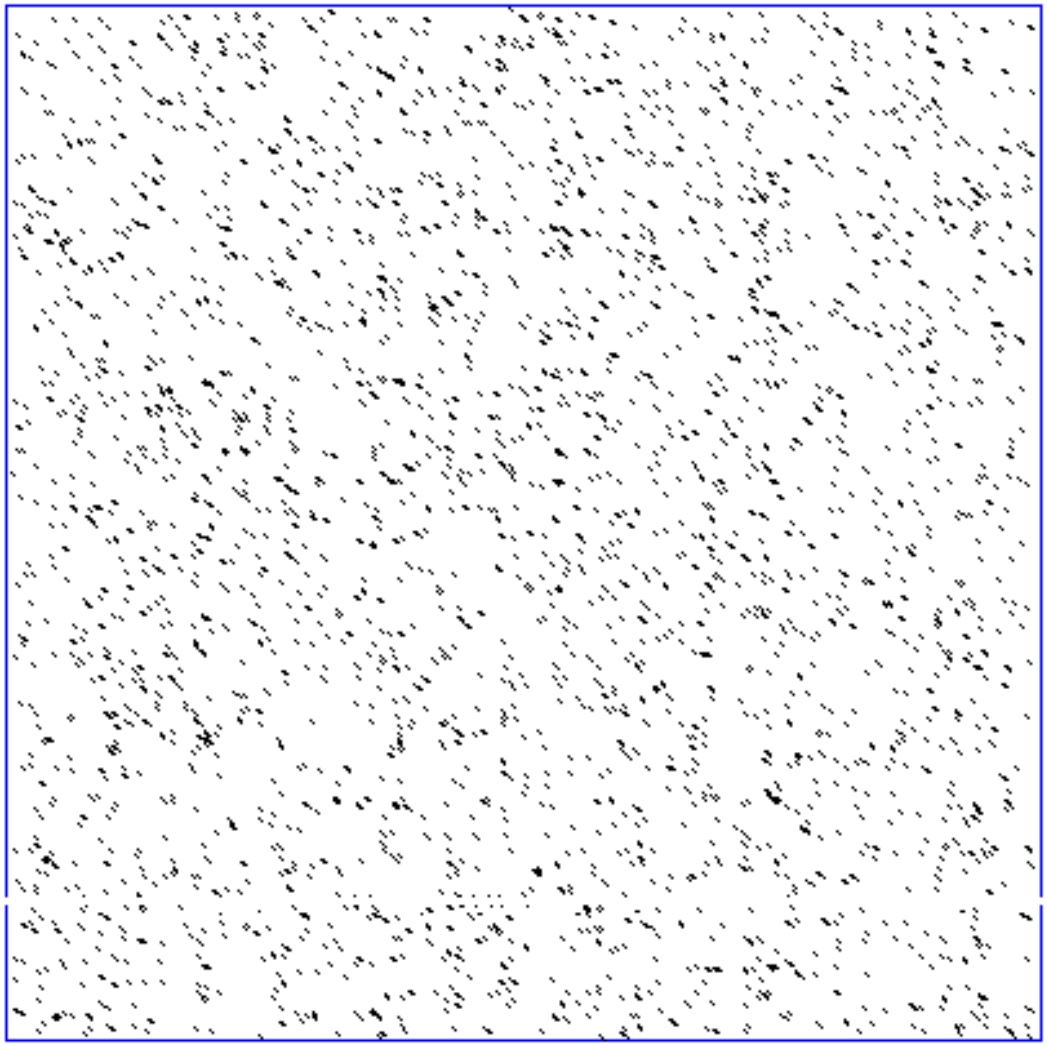


Click on plot to get positional data

two very different sequences

DNA 1 on horizontal axis = 1348 bases

DNA 2 on vertical axis = 2322 bases



Click on plot to get positional data

Sequence Alignment

- Sequence alignment is a way of arranging the sequences to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships.

```
 CGATGCAGACGTCA
        ||||||||
CGATGCAAGACGTCA
```

```
CGATGCAGACGTCA
||||||||
CGATGCAAGACGTCA
```

- The procedure of comparing two (pair-wise alignment) sequences is to search for a series of individual characters or patterns that are in the same order in the sequences.

Pairwise sequence alignment

Idea:

Display one sequence above another with **spaces** (or **gaps**) **inserted** in both to reveal similarity

```
A: C A T - T C A - C
   |   |     | |   |
B: C - T C G C A G C
```

Alignment Scoring

X= CTGTCG-CTGCACG

Y= -TGC-CG-TG----

Reward for matches: α
Mismatch penalty: β
Space penalty: γ

Score: $S(X,Y) = \alpha w - \beta x - \gamma y$

w = #matches x = #mismatches y = #spaces

Alignment Scoring

Reward for matches: 10
Mismatch penalty: 2
Space penalty: 5

C	T	G	T	C	G	–	C	T	G	C
–	T	G	C	–	C	G	–	T	G	–
-5	10	10	-2	-5	-2	-5	-5	10	10	-5

Total similarity score = 11

Types of Alignment algorithms

- **Global** alignment

```
CTGTCG-CTGCACG
-TGC-CG-TG----
```

- **Local** alignment

```
CTGTCGCTGCACG--
-------TGC-CGTG
```

- **Semi-Global** alignment

```
CTGTCGCTGCACG
-TGCCG-TG----
```

- Requirements:
  - *Scoring matrices*
  - *Gap penalty function*

Optimum Alignment problem

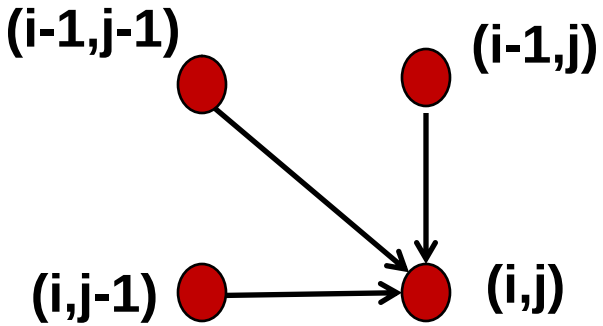
- The score of an alignment is a **measure of its quality**

- **Optimum alignment problem:** Given a pair of sequences *X* and *Y*, find an alignment (global or local) with **maximum score**

- The **similarity** between *X* and *Y*, denoted *sim*(*X*,*Y*), is the maximum score of an alignment of *X* and *Y.*

Global Alignment (Needelman-Wunsch, 1970)

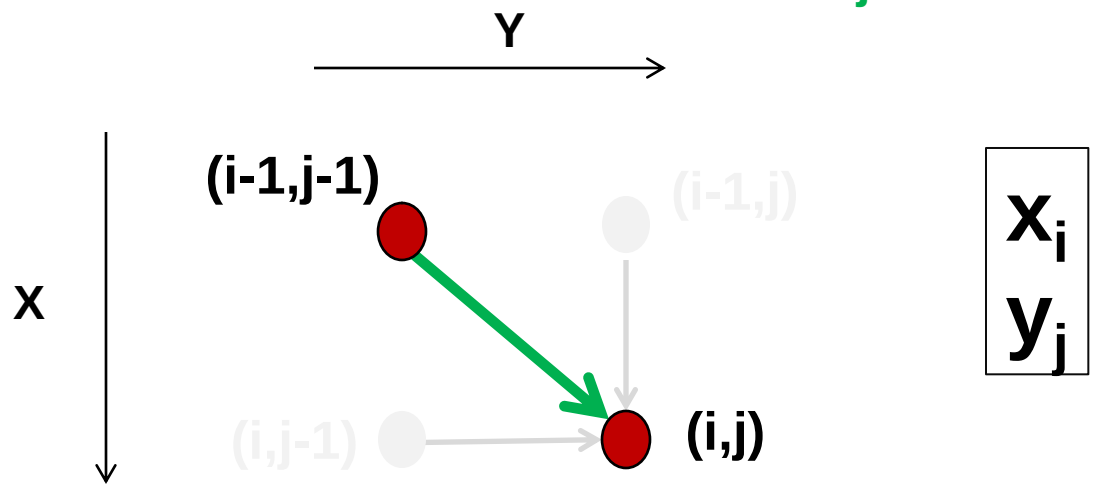
- 2 sequences: $X=(x_1, x_2, \ldots, x_n)$, $Y=(y_1, y_2, \ldots, y_m)$
- Suppose a similarity function S(a, b) between two members of discrete alphabet Ω
- **Idea:** Every edge in the directed graph has weight **equal to the type of alignment**.



Global Alignment (Needelman-Wunsch, 1970)

- 3 types of alignment

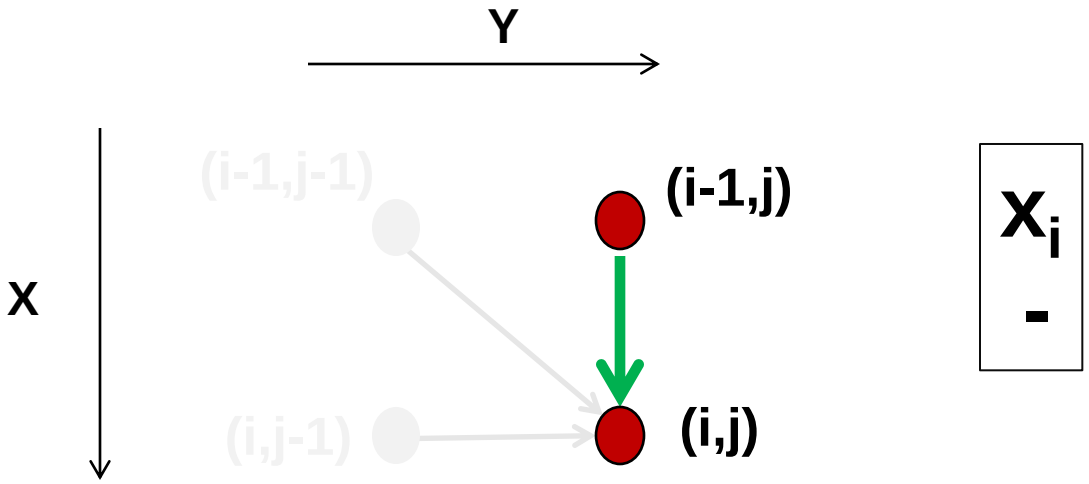
- **Case 1:** Characters from **both sequences** are aligned each other with cost $S(x_i, y_j)$



Global Alignment (Needelman-Wunsch, 1970)

- 3 types of alignment

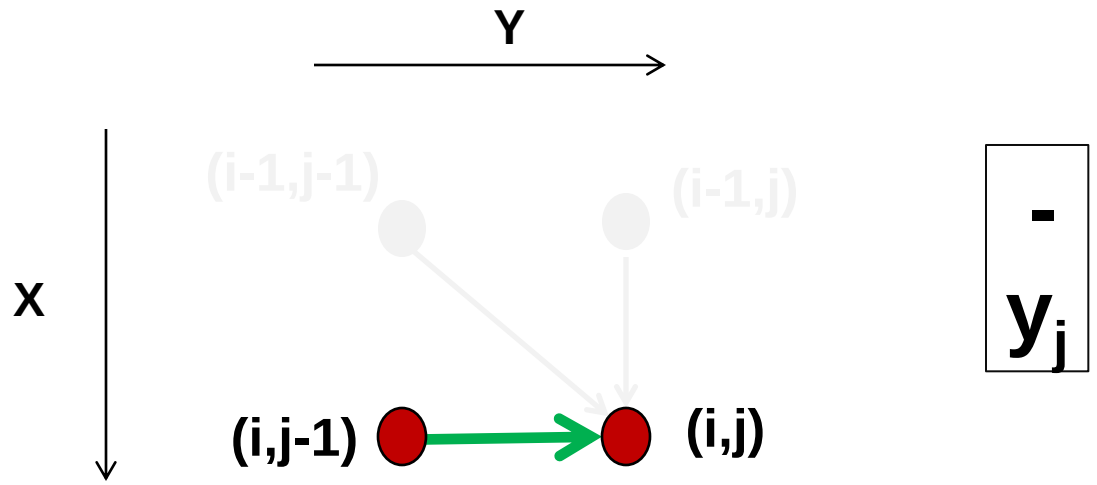
- **Case 2:** Character from (column) sequence **x is aligned with a gap from y** with cost $S(x_i, -) = -d$



Global Alignment (Needelman-Wunsch, 1970)

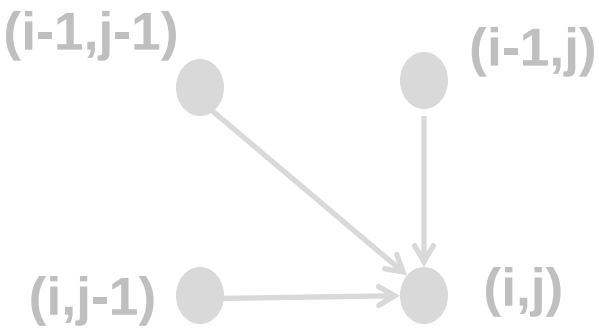
- 3 types of alignment

- **Case 3:** Character from (row) sequence **y is aligned with a gap from x** with cost $S(-, y_j) = -d$



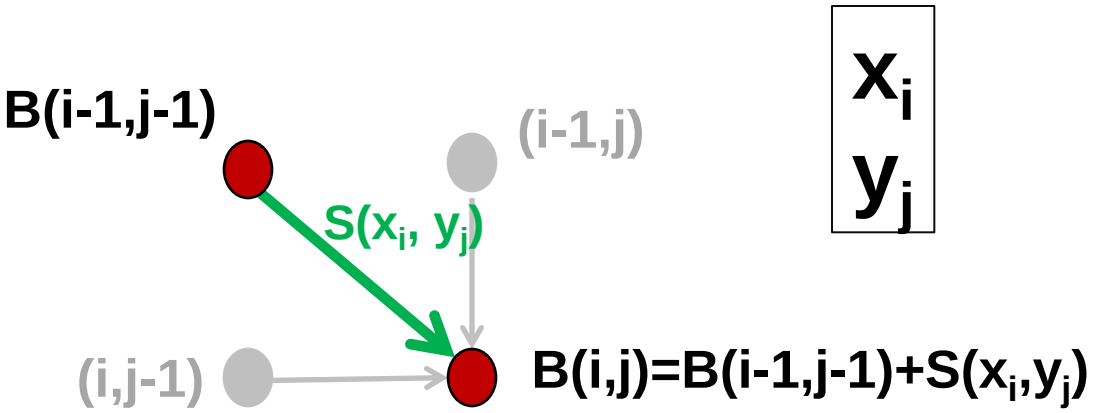
Global Alignment (Needelman-Wunsch, 1970)

- Problem: **Optimal path** with the maximum total similarity cost
- Insert **Cost function** $B(i,j)$, $0 \leq i \leq n$, $0 \leq j \leq m$
- **Initial** $B(0,0) = 0$



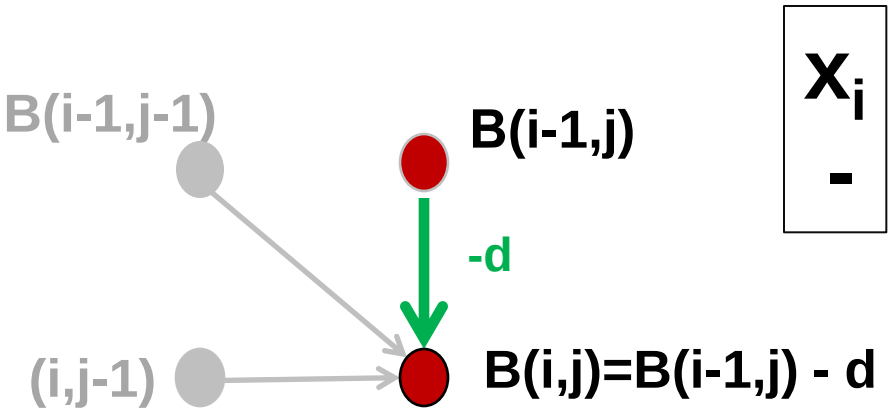
Global Alignment (Needelman-Wunsch, 1970)

- Problem: **Optimal path** with the maximum total similarity cost
- Insert **Cost function** *B(i,j)* , $0 \le i \le n$, $0 \le j \le m$
- **Initial** $B(0,0) = 0$
- *Case 1: alignment*



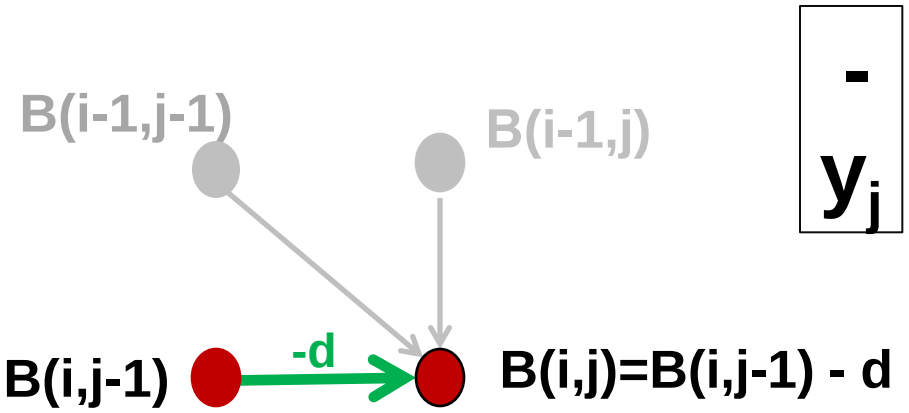
Global Alignment (Needelman-Wunsch, 1970)

- Problem: **Optimal path** with the maximum total similarity cost
- Insert **Cost function** $B(i,j)$, $0 \le i \le n$, $0 \le j \le m$
- **Initial** $B(0,0) = 0$
- *Case 2*



Global Alignment (Needelman-Wunsch, 1970)

- Problem: **Optimal path** with the maximum total similarity cost
- Insert **Cost function** $B(i,j)$, $0 \le i \le n$, $0 \le j \le m$
- **Initial** $B(0,0) = 0$
- *Case 3*



Global Alignment (Needelman-Wunsch, 1970)

- Problem: **Optimal path** with the maximum total similarity cost
- Insert **Cost function** $B(i,j)$, $0 \le i \le n$, $0 \le j \le m$
- **Initial** $B(0,0) = 0$
- General step:

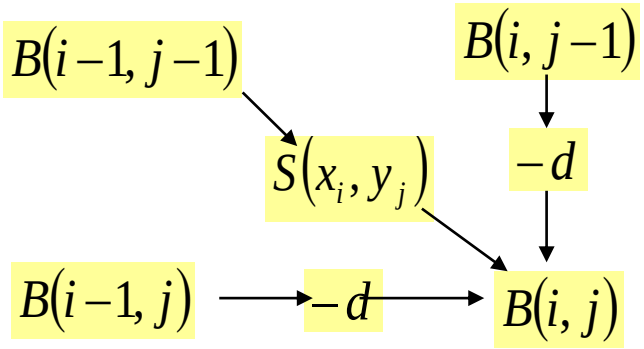
$$B(i,j) = \max\{B(i-1,j-1) + S(x_i, y_j), B(i-1,j) - d, B(i,j-1) - d\}$$

- Store which edge(s) was(were) selected
- At the end perform a **back-tracking** starting from B(n,m) to obtain the optimal path.

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

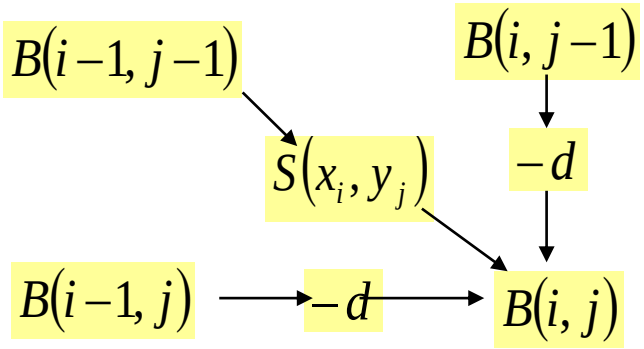
Find the optimal alignment of X=AAG and Y=AGC.
Use a gap penalty of d=-5.



		A	G	C
A				
A				
G				

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

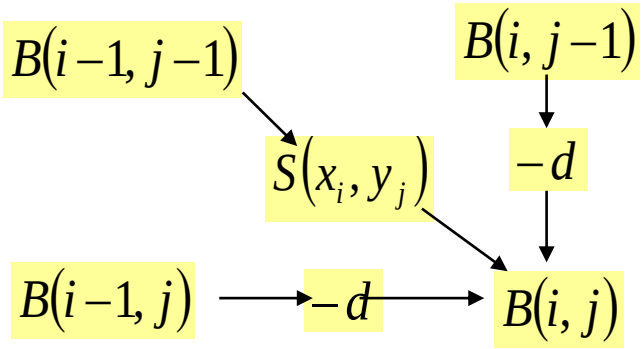


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0			
A				
A				
G				

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

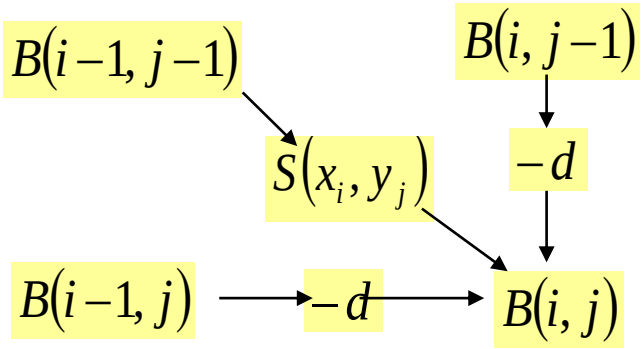


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5			
A	-10			
G	-15			

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

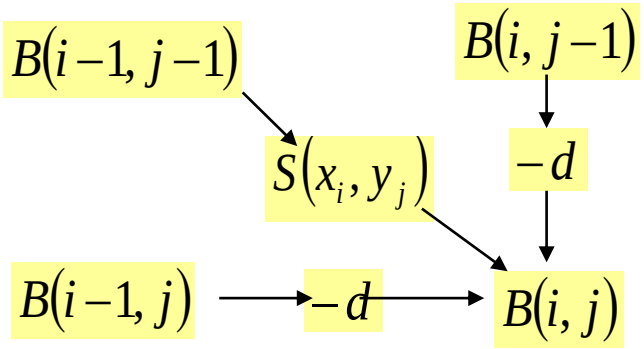


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2		
A	-10			
G	-15			

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

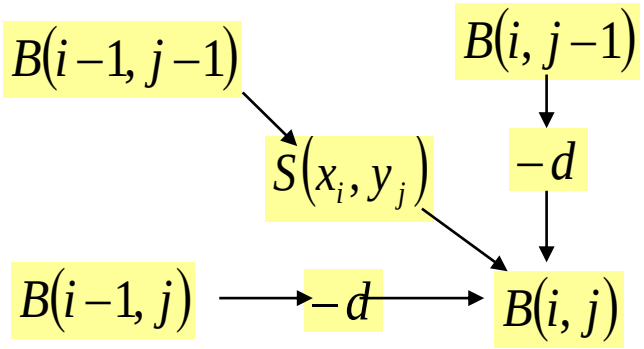


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3	
A	-10			
G	-15			

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

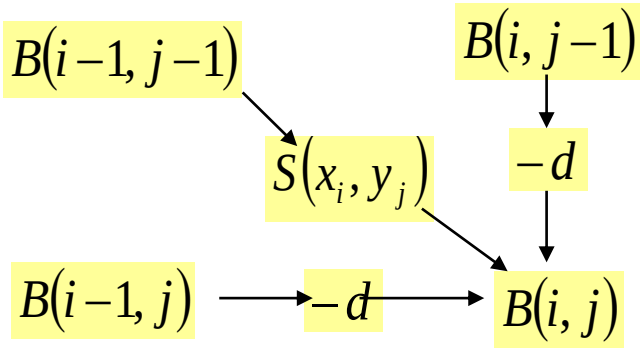


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10			
G	-15			

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

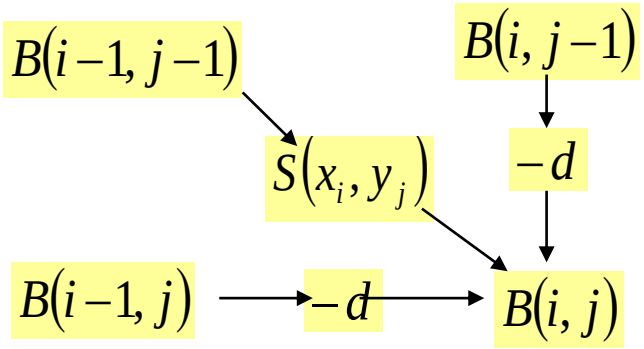


Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3		
G	-15			

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2



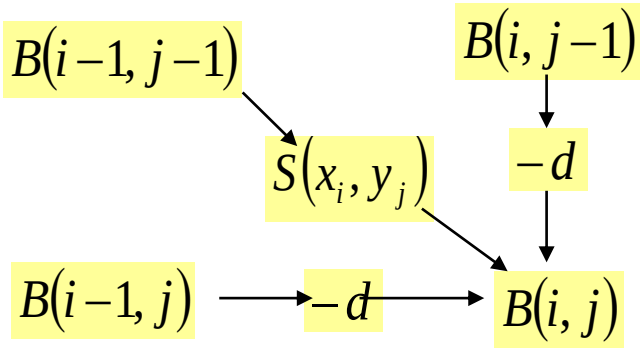
Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3		
G	-15	-8		

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

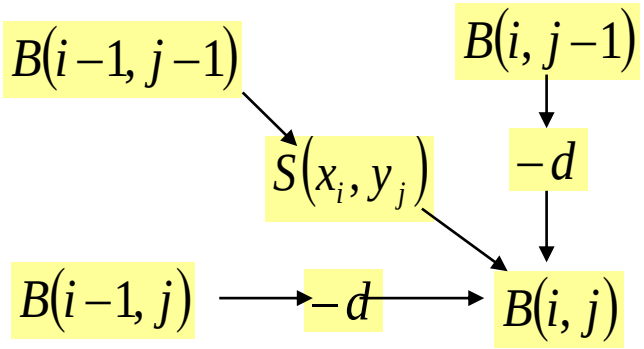


		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3	
G	-15	-8		

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

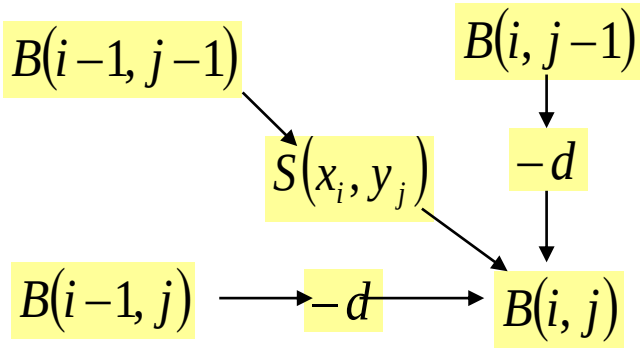


		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8		

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.

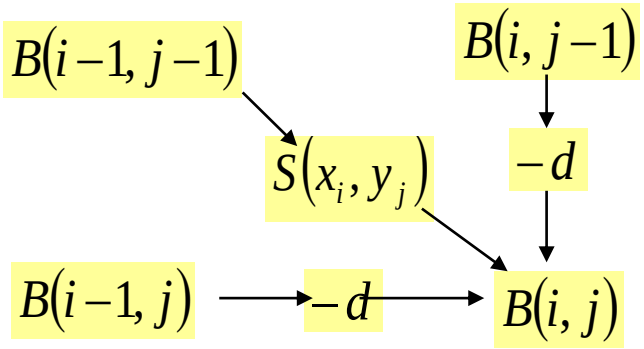


		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8	-1	

A simple example

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Find the optimal alignment of AAG and AGC.
Use a gap penalty of d=-5.



		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8	-1 →	-6

A simple example

Find the **optimal path**

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8	-1 →	-6

A simple example

Back-tracking

Each arrow introduces one character at the end of each aligned sequence.

- A **horizontal** move puts a gap in the **left sequence**.
- A **vertical** move puts a gap in the **top sequence**.
- A **diagonal** move uses one character from each sequence.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8	-1 →	-6

A simple example

Back-tracking

Each arrow introduces one character at the end of each aligned sequence.

- A **horizontal** move puts a gap in the **left sequence**.
- A **vertical** move puts a gap in the **top sequence**.
- A **diagonal** move uses one character from each sequence.

		A	G	C
	0 →	-5 →	-10 →	-15
A	-5	2 →	-3 →	-8
A	-10	-3	-3 →	-8
G	-15	-8	-1 →	-6

2 possible alignments with the same cost (-6)

X= AAG-
Y= A-GC

or

AAG-
-AGC

Example 2

X=ACCA **Y=AATCCGA**

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
A								
C								
C								
A								

Example 2

X=ACCA Y=AATCCGA

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	1	-1	-3	-5	-7	-9	-11
C	-4	-1	0	-2	-2	-4	-6	-8
C	-6	-3	-2	-1	-1	-1	-3	-5
A	-8	-5	-2	-3	-2	-2	-2	-2

Example 2

X=ACCA Y=AATCCGA

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	1	-1	-3	-5	-7	-9	-11
C	-4	-1	0	-2	-2	-4	-6	-8
C	-6	-3	-2	-1	-1	-1	-3	-5
A	-8	-5	-2	-3	-2	-2	-2	-2

Optimal path

Example 2

X=ACCA Y=AATCCGA

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	1	-1	-3	-5	-7	-9	-11
C	-4	-1	0	-2	-2	-4	-6	-8
C	-6	-3	-2	-1	-1	-1	-3	-5
A	-8	-5	-2	-3	-2	-2	-2	-2

X=-A-CC-A

X=AATCCGA

Similarity score = - 2

Local Alignment (Smith-Waterman, 1981)

- 2 sequences: $X=(x_1, x_2, \ldots, x_n)$, $Y=(y_1, y_2, \ldots, y_m)$
- Find **two common subsequences** with maximum **longest length** and **optimum cost** of alignment
- If $X_k^i = (x_k, x_{k+1}, \ldots, x_i) \quad 1 \le k \le i \le n$
  $Y_l^j = (y_l, y_{l+1}, \ldots, y_j) \quad 1 \le l \le j \le m$
  two subsequences
- Problem: $Find \quad \max_{\substack{1 \le k \le i \le n \\ 1 \le l \le j \le m}} \left\{ B\left(X_k^i, Y_l^j\right) \right\}$

Local Alignment (Smith-Waterman, 1981)

- Problem: $Find \quad \max_{\substack{1 \le k \le i \le n \\ 1 \le l \le j \le m}} \left\{ B\left(X_k^i, Y_l^j \right) \right\}$
- $\frac{n(n-1)}{2} = \binom{n}{2}$ possible subsequences X_k^i

  $\frac{m(m-1)}{2} = \binom{m}{2}$ possible subsequences Y_l^j

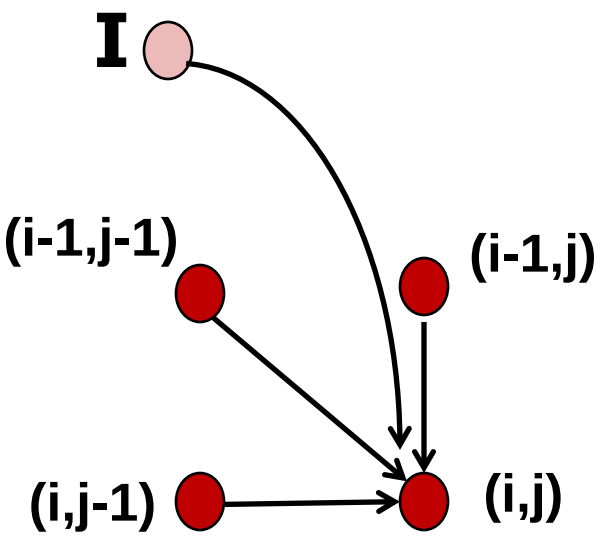
- Totally there are $\binom{n}{2} \times \binom{m}{2}$ possible pairs of subsequences that must be examined

Local Alignment (Smith-Waterman, 1981)

- **Dynamic programming** to solve this problem
- Introduce a **score function L(i,j)** and consider a (supposed) **starting point I**, where **L(I)=0**.
- **L(i,j) denotes** the maximum total score all paths **starting from I** and **ending at (i,j).**
- Then there are **four (4) cases**:

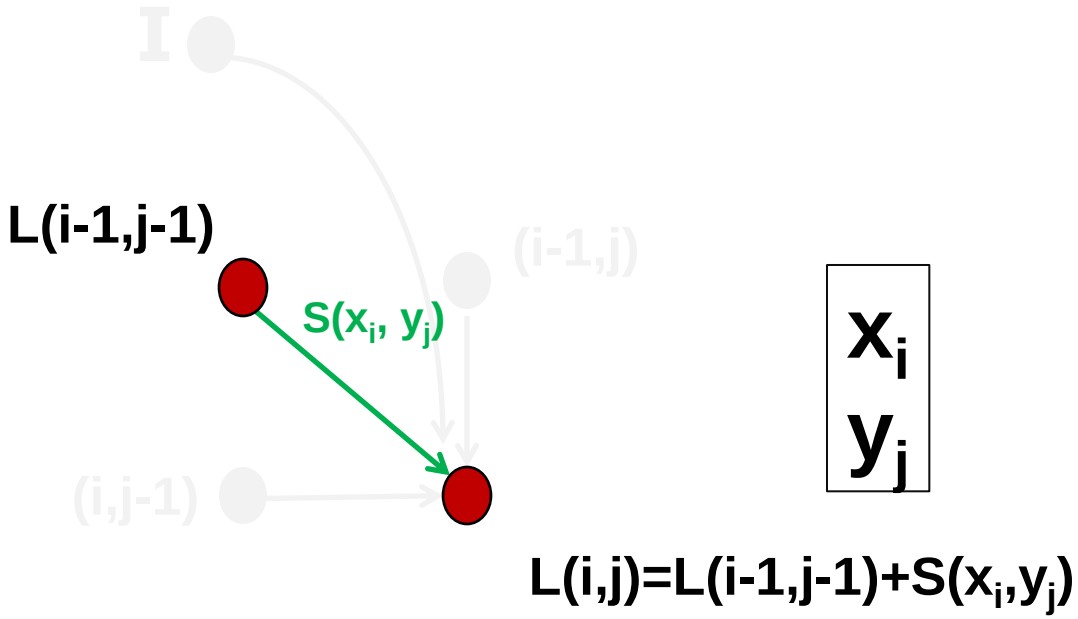
Local Alignment (Smith-Waterman, 1981)

- There are four (4) cases:



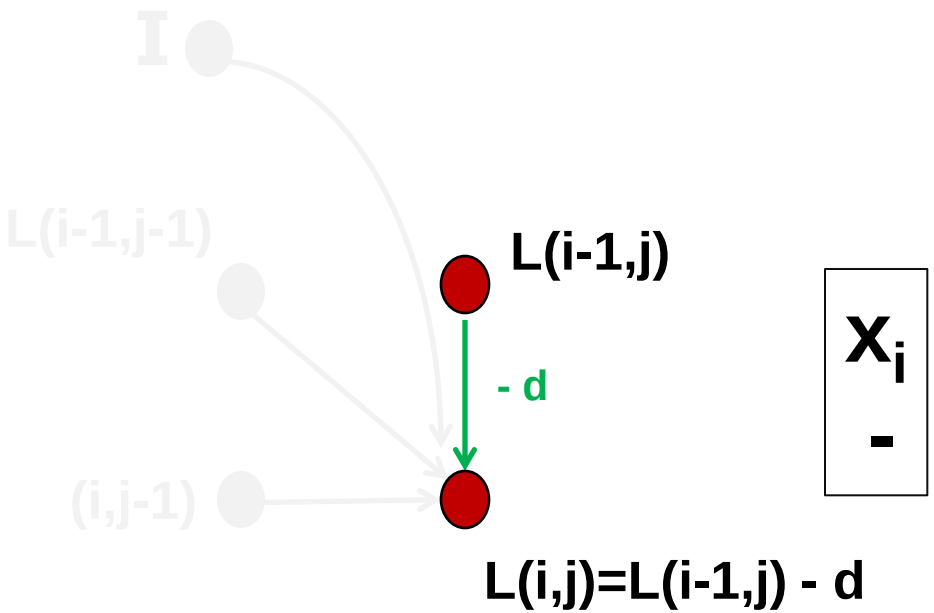
Local Alignment (Smith-Waterman, 1981)

- Then there are four (4) cases:



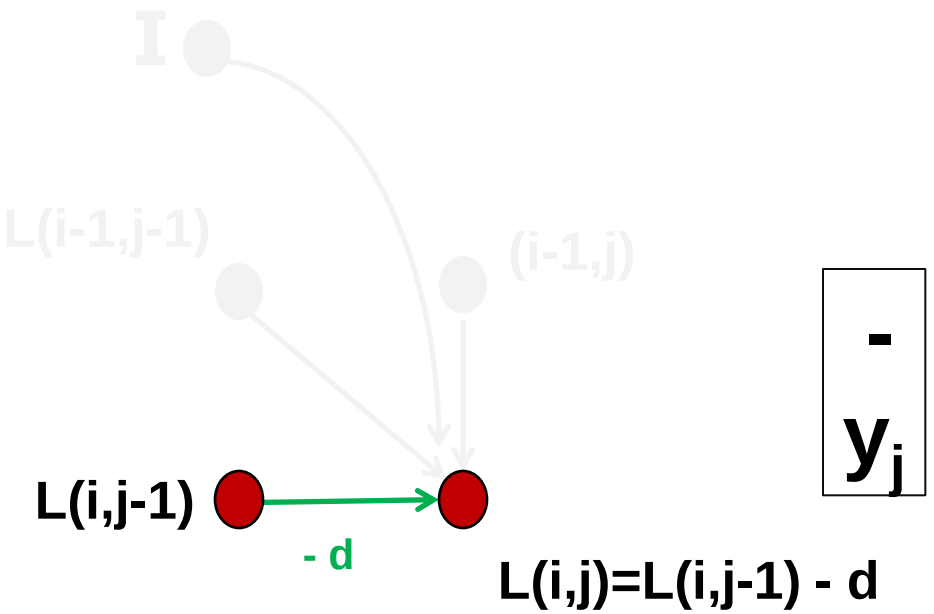
Local Alignment (Smith-Waterman, 1981)

- Then there are four (4) cases:



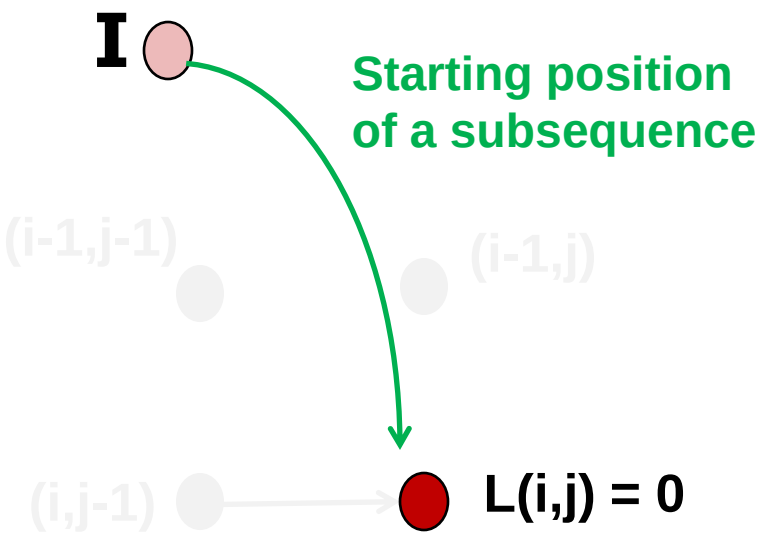
Local Alignment (Smith-Waterman, 1981)

- Then there are four (4) cases:



Local Alignment (Smith-Waterman, 1981)

- There are four (4) cases:



Local Alignment (Smith-Waterman, 1981)

- Problem: **Longest Common Subsequence**
- **Cost function** $L(i,j)$, $0 \leq i \leq n$, $0 \leq j \leq m$

Local Alignment (Smith-Waterman, 1981)

- Problem: **Longest Common Subsequence**
- **Cost function** $L(i,j)$, $0 \le i \le n$, $0 \le j \le m$
- **Initial** $L(0,0) = 0$
- General step:

$$L(i, j) = \\ = \max\{L(i-1, j-1) + S(x_i, y_j), L(i-1, j) - d, L(i, j-1) - d, 0\}$$

Local Alignment (Smith-Waterman, 1981)

- Problem: **Longest Common Subsequence**
- **Cost function** $L(i,j)$, $0 \le i \le n$, $0 \le j \le m$
- **Initial** $L(0,0) = 0$
- General step:

$$L(i, j) = \\ = \max\{L(i-1, j-1) + S(x_i, y_j), L(i-1, j) - d, L(i, j-1) - d, 0\}$$

- At the end find the maximum L(i,j) and then perform a **back-tracking** until found **zero (0)**. This will set the optimum local alignment.

Example 1

X=ACCA Y=AATCCGA

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}, \quad d=2$

		A	A	T	C	C	G	A
A								
C								
C								
A								

$$L(i,j)=\max\begin{Bmatrix}L(i-1,j-1)+S(x_i,y_j)\\ L(i-1,j)-d\\ L(i,j-1)-d\\ 0\end{Bmatrix}$$

Example 1

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
	0	0	0	0	0	0	0	0
A	0	1	1	0	0	0	0	1
C	0	0	0	0	1	1	0	0
C	0	0	0	0	1	2	0	0
A	0	1	1	0	0	0	0	1

$$L(i,j)=\max\begin{Bmatrix}L(i-1,j-1)+S(x_i,y_j)\\ L(i-1,j)-d\\ L(i,j-1)-d\\ 0\end{Bmatrix}$$

Example 1

Scoring scheme: $S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}$, $d=2$

		A	A	T	C	C	G	A
	0	0	0	0	0	0	0	0
A	0	1	1	0	0	0	0	1
C	0	0	0	0	1	1	0	0
C	0	0	0	0	1	2	0	0
A	0	1	1	0	0	0	0	1

Local alignment (subsequence length 2)

X=CC

Y=CC

Example 2

		T	T	C	T	A	T	T	G
A									
G									
C									
T									
A									
A									
C									

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases},\quad d=2$$

$$L(i,j)=\max\begin{cases}L(i-1,j-1)+S(x_i,y_j)\\ L(i-1,j)-d\\ L(i,j-1)-d\\ 0\end{cases}$$

Example 2

		T	T	C	T	A	T	T	G
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	0	0	1
C	0	0	0	1	0	0	0	0	0
T	0	1	1	0	2	0	1	1	0
A	0	0	0	0	0	3	1	0	0
A	0	0	0	0	0	1	2	0	0
C	0	0	0	1	0	1	0	1	0

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases},\quad d=2$$

$$L(i,j)=\max\begin{cases}L(i-1,j-1)+S(x_i,y_j)\\ L(i-1,j)-d\\ L(i,j-1)-d\\ 0\end{cases}$$

Example 2

		T	T	C	T	A	T	T	G
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	0	0	1
C	0	0	0	1	0	0	0	0	0
T	0	1	1	0	2	0	1	1	0
A	0	0	0	0	0	3	1	0	0
A	0	0	0	0	0	1	2	0	0
C	0	0	0	1	0	1	0	1	0

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases},\quad d=2$$

Example 2

		T	T	C	T	A	T	T	G
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	0	0	1
C	0	0	0	1	0	0	0	0	0
T	0	1	1	0	2	0	1	1	0
A	0	0	0	0	0	3	1	0	0
A	0	0	0	0	0	1	2	0	0
C	0	0	0	1	0	1	0	1	0

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases},\quad d=2$$

Local alignment (subsequence length 3)

X=CTA

Y=CTA

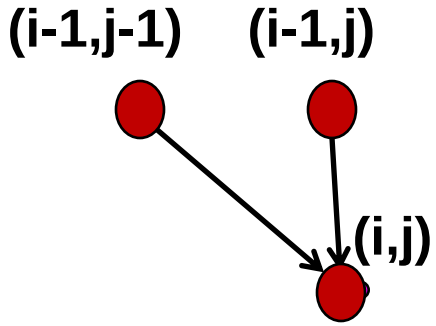
Semi Global Alignment

- 2 sequences: $X=(x_1, x_2, \ldots, x_n)$, $Y=(y_1, y_2, \ldots, y_m)$
- In case where one sequence (e.g. x) has much **larger length** from the other (e.g. y), i.e n>>m.

Semi Global Alignment

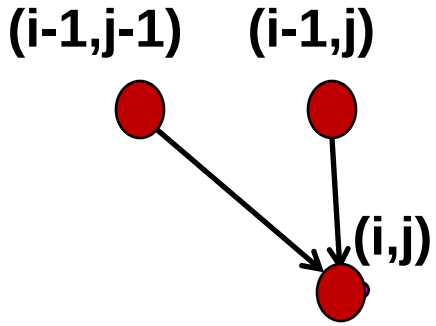
- 2 sequences: X=(x_1, x_2, …, x_n), Y=(y_1, y_2, …, y_m)
- In case where one sequence (e.g. x) has much **larger length** from the other (e.g. y), i.e n>>m.
- Then, keep the **longer gapless** and allow gaps only to the smallest.
- **Then:**

$$B(i,j)=\max\begin{cases} B(i-1,j-1)+S(x_i,y_j) \\ B(i-1,j)-d \end{cases}$$



Semi Global Alignment

- 2 sequences: X=(x_1, x_2, …, x_n), Y=(y_1, y_2, …, y_m)
- In case where one sequence (e.g. x) has much **larger length** from the other (e.g. y), i.e n>>m.
- Then, keep the **longer gapless** and allow gaps only to the smallest.
- **Then:**



$$B(i, j) = \max \begin{cases} B(i-1, j-1) + S(x_i, y_j) \\ B(i-1, j) - d \end{cases}$$

- At the end find the **max B(i,j) value in the last column** and traceback until 1st column.

		T	A	C
A				
G				
A				
T				
A				
T				
C				
C				

Example

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}, \quad d=2$$

Example

		T	A	C
	0	-2	-4	-6
A	0	-3	-1	-5
G	0	-1	-3	-2
A	0	-1	0	-1
T	0	1	0	-1
A	0	0	2	-1
T	0	1	0	1
C	0	-1	0	1
C	0	-1	-2	1

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}, \quad d=2$$

		T	A	C
	0	-2	-4	-6
A	0	-3	-1	-5
G	0	-1	-3	-2
A	0	-1	0	-1
T	0	1	0	-1
A	0	0	2	-1
T	0	1	0	1
C	0	-1	0	1
C	0	-1	-2	1

Example

Scoring scheme:

$$S(a,b)=\begin{cases}1 & a=b\\ -1 & a\neq b\end{cases}, \quad d=2$$

3 alignments (score=1)

```
X=AGATATCC
Y=------TAC
```

```
X=AGATATCC
Y=---TA-C-
```

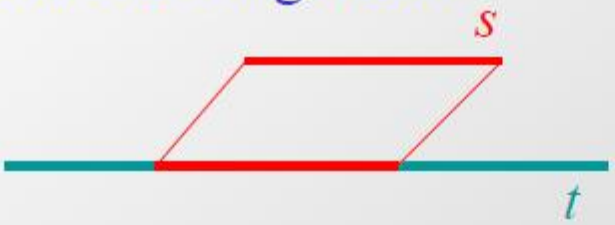
```
X=AGATATCC
Y=---TAC--
```

DP Algorithm Variations

Global Alignment

	-	A	G	T
-	0	-2	-4	-6
A	-2	1	-1	-1
A	-4	-1	-1	-2
G	-6	-1	0	0
C	-8	-3	0	-1

Semi-Global Alignment



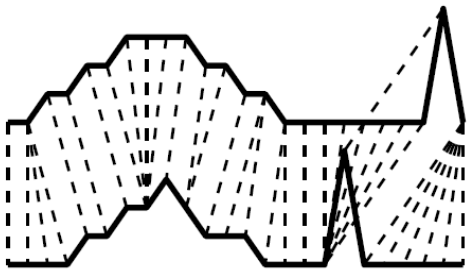
	-	A	G	T
-	0	-2	-4	-6
A	0	1	-1	-1
A	0	1	0	-2
G	0	-1	2	1
C	0	-1	0	0

Local Alignment

	-	A	G	T
-	0	0	0	0
A	0	1	0	0
A	0	1	0	0
G	0	0	2	0
A	0	1	0	1

Dynamic Time Warping (DTW) distance

(Berndt & Clifford, 1994)



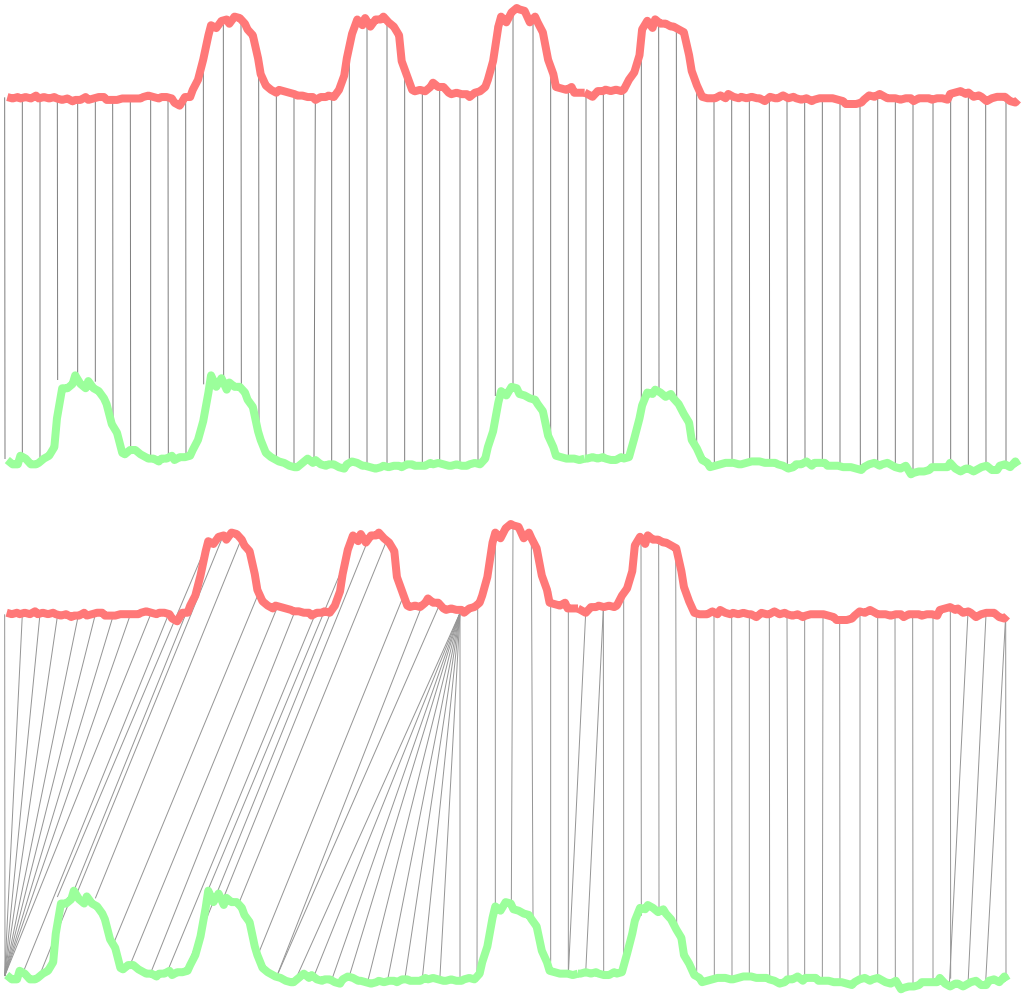
- Compare two time-series

- Use a distance function between two values, e.g. $d(a,b)=(a-b)^2$ or d(a,b)=|a-b|
- Following dynamic programming, we use a function **D(i,j)** that **stores the minimum distance** between the substrings $(x_1, \ldots, x_i)$ and $(y_1, \ldots, y_j)$. Then:

$$D(i,j) = \\ = d(x_i, y_j) + \min\{D(i-1, j-1), D(i, j-1), D(i-1, j)\}$$

- Not satisfying triangle inequality
- Slow to compute - O(nm) time

Euclidean vs. DTW distance



- **Euclidean distance**:
  - Matches rigidly along the time axis.
- **DTW distance**:
  - Allows stretching and shrinking along the time axis.

q - Gram distance

- 2 sequences: $X=(x_1, x_2, \ldots, x_n)$, $Y=(y_1, y_2, \ldots, y_m)$
- **q-gram:** all the possible subsequences (w_q) of length q.
- There are $|\Omega|^q$ such possible subsequences.
- Then, **q-gram distance**:

$$D(X,Y)=\sum_{w_q}\left|n_X(w_q)-n_Y(w_q)\right|$$

where $n_X(w_q)$ **frequency** of w_q on sequence X

q - Gram distance

Example: **X = a a b a Y = a b a a Ω = {a,b}**

- **q=2** : w_q = {aa, ab, ba, bb}. 2-gram distance

$$D(X,Y)=\sum_{w_q}\left|n_X(w_q)-n_Y(w_q)\right|=|1-1|+|1-1|+|1-1|+|0-0|=0$$

- **q=3 :** w_q = {aaa,aab,aba,abb,baa,bab,bba,bbb}. 3-gram distance

$$D(X,Y)=\sum_{w_q}\left|n_X(w_q)-n_Y(w_q)\right|=0+1+0+0+1+0+0+0=2$$