Algorithms for computing and evaluating sequence alignments

Distance between aligned sequences

Alignments with gaps

```
A:   ALASVLIRLIT--YP
B:   ASAVHL---ITRLYP
```

Substitution

Correspondence between residue *i* and residue *j* is scored with value : *s(i,j)* [Substitution matrix] ,

Deletion and Insertion

gap score is always negative and depends only on its length

LINEAR $\sigma(n) = -nd$ (each gapped position is equivalent)

AFFINE $\sigma(n) = -d-(n-1)e$ (*d*: opening, *e*: extension with *d*>*e*)

Alignment score

$$Score(A, B, aligment) = \sum_{nonGapPositions\_k} s(A^k, B^k) + \sum_{gap\_j} \sigma_j(gap\_length)$$

Sequence alignment algorithm

Given two sequences, how to find the maximal scoring alignment ?

Naïf solution: try all the possible alignments and choose the best scoring

The score of any alignment can be computed with

$$Score(A, B, aligment) = \sum_{nonGapPositions\_k} s(A^k, B^k) + \sum_{gap\_j} \sigma_j(gap\_length)$$

How many possible alignments between to sequences?

Write ALL the possible ungapped alignments between the two sequences

A: tca
B: ga

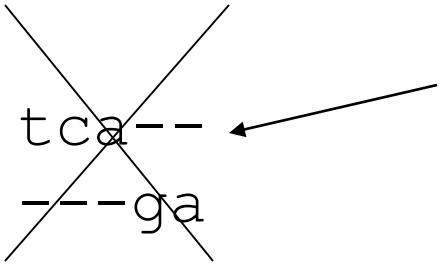
Score the alignments using the following matrix and the linear gap penalty (d=2)

	A	**C**	**T**	**G**
A	2	-1	-1	0
C		2	0	-1
T			2	-1
G				2

How many possible alignments between two sequences?

Ungapped

Identical to the first



```
--tca   -tca   tca   tca   tca-   tca--
ga---   ga--   ga-   -ga   --ga   ---ga
```

-10 ***-7*** ***-4*** ***-1*** ***-6***

Given two sequences with lengths *m* and *n*, the number of shifts is *m* +*n*

How many possible alignments between to sequences?

Write ALL the possible gapped alignments between the two sequences

A: tca
B: ga

Score the alignments using the following matrix and the linear gap penalty (d=2)

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

How many possible alignments between two sequences?

Gapped

--tca	-tca	-tca	-tca	t-ca
ga---	ga--	g-a-	g--a	ga--
gatca	***gtaca***	***gtcaa***	***gtcaa***	***tgaca***
22111	***21211***	***21121***	***21112***	***12211***

tca	tca	tc-a	tca	tca-
ga-	g-a	-ga-	-ga	--ga
tgcaa	***tgcaa***	***tcgaa***	***tcgaa***	***tcaga***
12121	***12112***	***11221***	***11212***	***11122***

The number of possible alignments is equal to the possible ways to intercalate two sequences, preserving the order
Given two sequences with lengths m and n, the number of possible alignments is $(m+n)!/n!m!$

If n=m=80 there are $9 \cdot 10^{42}$ possible alignments !!!!!!!

Solution: adopt dynamic programming strategies

Needleman-Wunsch: GLOBAL ALIGNMENTS
Smith-Waterman: LOCAL ALIGNMENTS

Basic idea of dynamic programming

The exhaustive computation of the alignment scores for all the possible alignments leads to compute the same things many times

Consider two sequences to be optimally aligned
ALSKLASPALSAKDLDSPALS ; ALSKIADSLAPIKDLSPASLT

Consider the two alignments:

ALSKLASPALSAKDLDSPALS
ALSKIADSLAPIKDLSPASLT

The two alignments are equal for most of the length

ALSKLASPALSAKDLDSPAL-S
ALSKIADSLAPIKDLSPASLT-

Basic idea of dynamic programming

The complete computation of the alignment scores for all the possible alignments leads to compute the same things many times

Consider the two alignments:

ALSKLASPALSAKDLDSPALS
ALSKIADSLAPIKDLSPASLT

The two alignments are equal for most of the length

ALSKLASPALSAKDLDSPAL-S
ALSKIADSLAPIKDLSPASLT-

$$Score(A, B, aligment) = \sum_{nonGapPositions\_k} s(A^k, B^k) + \sum_{gap\_j} \sigma_j(gap\_length)$$

Scores are additive through the alignment: with naif method the score for the first part of the alignment is computed twice **BETTER TO STORE AND REUSE IT**

Basic idea of dynamic programming

The alignment can be built step by step:

Given an alignment of length n between two subtrings of the sequences, three alignments of length n+1 can be originated

Consider the two sequences
ALSKLASPALSAKDLDSPALS ; ALSKIADSLAPIKDLSPASLT

Consider an alignment between the two substrings lacking the final characters

```
ALSKLASPALSAKDLDSPA-L   Score = Sc; length = 21
ALSKIADSLAPIKDL-SPASL
```

It can give origin to three alignments of length = 22

Basic idea of dynamic programming

The alignment can be built step by step:

It can give origin to three alignments of length = 22

1) Add a character in both sequences:

```
ALSKLASPALSAKDLDSPA-LS          Score = Sc+Match(S,T)
ALSKIADSLAPIKDL-SPASLT
```

2) Add a character in the first sequence (and a gap in the second):

```
ALSKLASPALSAKDLDSPA-LS          Score = Sc+Gap
ALSKIADSLAPIKDL-SPASL-
```

3) Add a character in the second sequence (and a gap in the first):

```
ALSKLASPALSAKDLDSPA-L-          Score = Sc+Gap
ALSKIADSLAPIKDL-SPASLT
```

Scores for the three alignments are computed starting from the score of the alignment of length 21

Basic idea of dynamic programming

It is not necessary to explicitely evaluate and store ALL the possible alignments

Consider the two sequences
ALSKLASPALSAKDLDSPALS ; ALSKIADSLAPIKDLSPASLT

Consider two of the possible alignments between the substrings lacking the last characters

```
ALSKLASPALSAKDLDSPAL          Score = Sc1
ALSKIADSLAPIKDLSPASL

ALSKLASPALSAKDLDSPA-L         Score = Sc2
ALSKIADSLAPIKDLS-PASL
```

In general the two scores are different: let say that Sc1>Sc2
Then we discard the second aligment for the following steps

Basic idea of dynamic programming

It is not necessary to explicitely evaluate and store ALL the possible alignments

```
ALSKLASPALSAKDLDSPAL          ALSKLASPALSAKDLDSPA-L
ALSKIADSLAPIKDLSPASL          ALSKIADSLAPIKDLS-PASL
Score = Sc1             >     Score = Sc2
```

Adding a character in both sequences in the next step:

```
ALSKLASPALSAKDLDSPALS         ALSKLASPALSAKDLDSPA-LS
ALSKIADSLAPIKDLSPASLT         ALSKIADSLAPIKDLS-PASLT
Score = Sc1 + Match (S,T)  >  Score = Sc2 + Match (S,T)
```

Due to the additivity of the scores, **alignment 2 will give always origin to non optimal alignment in the next steps** (also for the two other possibilities of increasing the alignment)

Basic idea of dynamic programming

It is not necessary to explicitely evaluate and store ALL the possible alignments

We need to store only the BEST score for the alignment of any two substrings of the original sequence

{ ALSKLASPA
  ALSKIAD

Let's indicate with curly bracket the best score alignment between the substrings

Basic idea of dynamic programming

Build the alignment step by step, storing the optimal alignment between substrings

Given the two sequences
ALSKLASPALSAKDLDSPALS, ALSKIADSLAPIKDLSPASLT

the best alignment between the substrings

ALSKLASPA
ALSKIAD

is for sure deriving from one of the following possibilities:

ALSKLASP + A ALSKLASP + A ALSKLASPA + –
ALSKIA   D ALSKIAD   – ALSKIA   D

and it is the highest scoring one

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Initialization $F(0,0) = 0$ No character is aligned

From this, three aligments can be originated

$0A^1$	$0A^1$	0 -
$0B^1$	0 -	$0B^1$
Score =	Score =	Score =

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Initialization $F(0,0) = 0$ No character is aligned

From this, three aligments can be originated

$0A^1$	$0A^1$	0 -
$0B^1$	0 -	$0B^1$
Score=match(A^1B^1)	Score = gap	Score = gap

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Iteration *F(1,0)=?*

it is the score of the aligment in which the first letter of A is aligned with no letters of B.

Only one possibility:

$0A^1$

0 -

Score = gap → F(1,0) =gap

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Iteration *F(0,1)=?*

it is the score of the aligment in which the first letter of B is aligned with no letters of A.

Only one possibility:

0 -

$0B^1$

Score = gap → F(0,1) =gap

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Iteration *F(1,1)=?*

it is the score of the aligment in which the first letter of A is aligned with the first letter of B.

One possibility:

$0A^1$
$0B^1$

Score=match(A^1B^1)

is it the only one?

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)x(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3.....A^i$ and $0B^1B^2B^3.....B^j$.

Iteration *F(1,1)=?*

it is the score of the aligment in which the first letter of A is aligned with the first letter of B. Three possibilities:

$0A^1$	$0A^1$ -	0 - A^1
$0B^1$	0 - B^1	0 B^1 -
Score=match(A^1B^1)	Score = F(1,0)+gap	Score = F(0,1)+gap

F(1,1)= Maximum of the three possibilities

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3\ldots\ldots A^i$ and $0B^1B^2B^3\ldots\ldots B^j$.

Initialization $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(A^i,B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

$$\begin{cases} \texttt{ALSKLASP} & + & \texttt{A} \\ \texttt{ALSKIA} & & \texttt{D} \end{cases} \qquad \begin{cases} \texttt{ALSKLASP} & + & \texttt{A} \\ \texttt{ALSKIAD} & & - \end{cases} \qquad \begin{cases} \texttt{ALSKLASPA} & + & - \\ \texttt{ALSKIA} & & \texttt{D} \end{cases}$$

$F(i-1,j-1)+M(A,D)$ $F(i-1,j)+Gap$ $F(i,j-1)+Gap$

Needleman and Wunsch algorithm

GLOBAL alignment, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)(b+1) matrix *F(i,j)* storing the score of the best alignment between the substrings

$0A^1A^2A^3\ldots\ldots A^i$ and $0B^1B^2B^3\ldots\ldots B^j$.

Initialization $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(A^i,B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Keep trace of the option that maximises F(i,j): match, gap1, or gap2

Termination *Best alignment score = F(a,b)*

Back-trace the optimal path

Needleman and Wunsch algorithm

Align the sequences

ACTGG and ACCA

Initialization

		0	A	C	T	G	G
	0	0					
	A						
	C						
	C						
	A						

Substitution matrix,
Linear gap penalty

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

$d = 2$

Needleman and Wunsch algorithm

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

$d = 2$

Iteration 1

	0	**A**	**C**	**T**	**G**	**G**
0	0 ←	-2				
	↑					
A	-2					
C						
C						
A						

F(1,0)= -d

F(0,1)= -d

Needleman and Wunsch algorithm

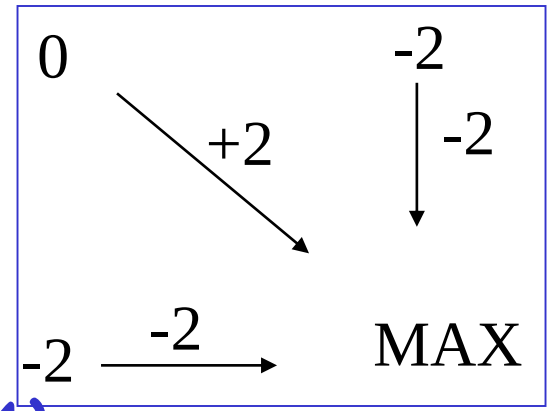
	A	C	T	G
A	2	-1	-1	0
C		2	0	-1
T			2	-1
G				2

$d = 2$

Iteration 1

	0	**A**	**C**	**T**	**G**	**G**
0	0 ←	-2				
A	-2	2				
C						
C						
A						

$$F(1,1) = MAX \begin{cases} F(1,0) - d \\ F(0,1) - d \\ F(0,0) + S(A,A) \end{cases}$$

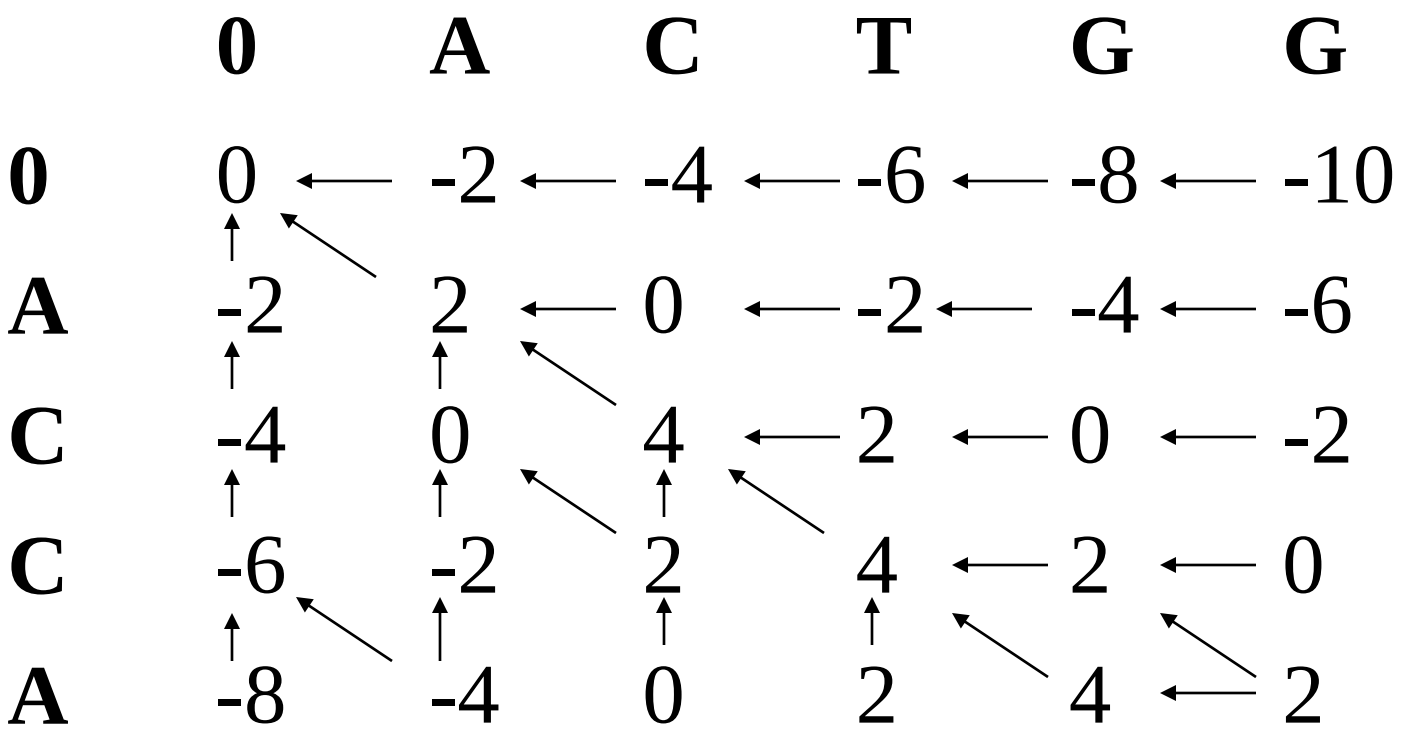


Needleman and Wunsch algorithm

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

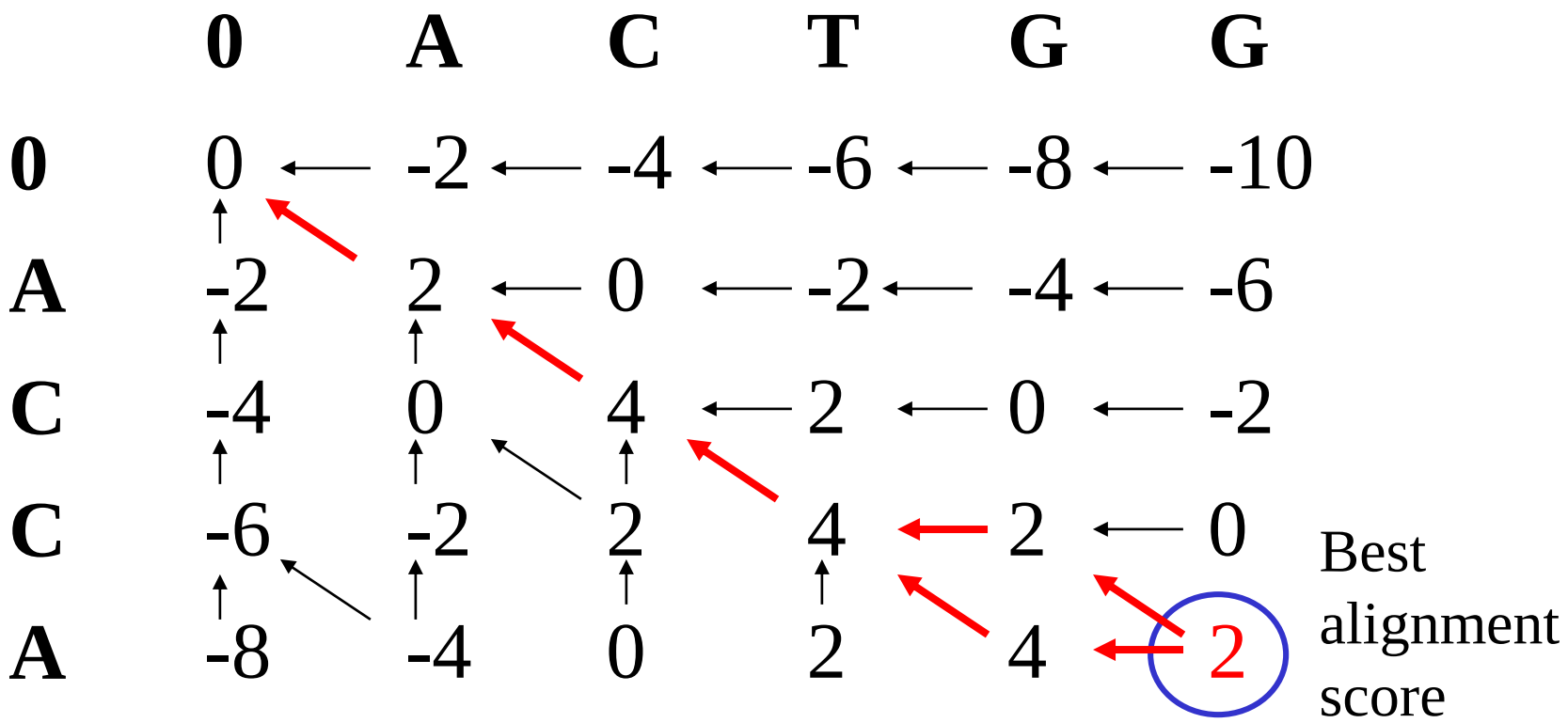
$d = 2$

Iterations



Needleman and Wunsch algorithm

Termination



← Gap in sequence 2 ↑ Gap in sequence 1 ↖ Match

```
0 A C T G G        0 A C T G G
0 A C C A -        0 A C C - A
```

Needleman and Wunsch algorithm

Best alignments: verify the alignment score

```
0 A C T G G
0 A C C A -
```

Score= 2+2+0+0-2=2

```
0 A C T G G
0 A C C - A
```

Score= 2+2+0-2+0=2

	A	**C**	**T**	**G**
A	2	-1	-1	0
C		2	0	-1
T			2	-1
G				2

d = 2

Computational complexity

Number of elementary operations needed to compute the optimal score with an algorithm

Naïf algorithm

Given two sequences of length *n* the number of possible alignments is $(2n)!/(n!)^2$

Each alignment requires between *n* to 2*n* operations (depending on the alignment length).

Since $n! \approx n^n (2\pi n)^{1/2} e^{-n}$ *(Stirling's formula)*

Complexity $\approx n^* [(2n)^{2n} (2\pi 2n)^{1/2} e^{-2n})/(n^n (2\pi n)^{1/2} e^{-n})^2 \approx$

$\approx O(2^{2n} n^{1/2})$

Estimated universe age
4.32×10^{17}sec

$n = 100 \rightarrow 2^{2n} \approx 1.6 \times 10^{60}$

$n = 1000 \rightarrow 2^{2n} \approx 1.1 \times 10^{602}$

Computational complexity

Number of elementary operations needed to compute the optimal score with an algorithm

Needleman-Wunsch algorithm

$(n+1)^2$ matrix values must be computed.

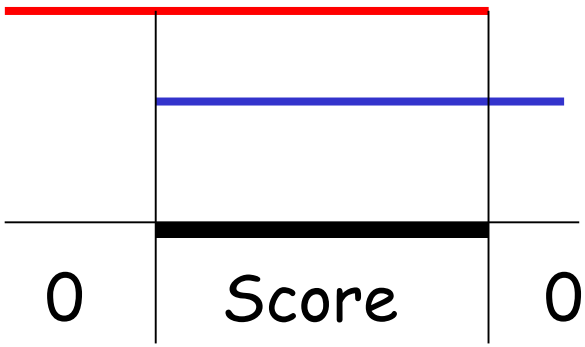
Each requires 4 operations (3 sums and one choice)

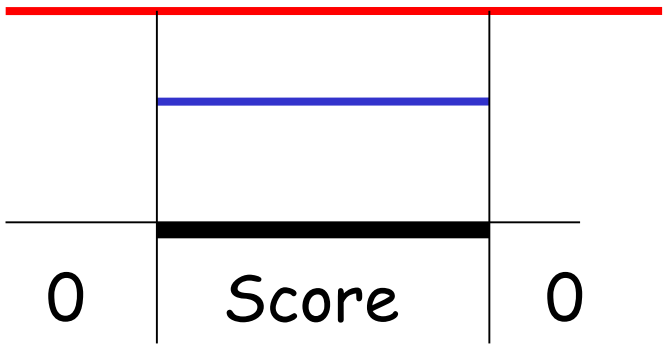
Complexity $\approx O(n^2)$

Dynamic programming makes the alignment feasible

Variations on the Needleman and Wunsch algorithm

Suppose you want to align two sequences WITHOUT penalizing gaps at the begin or at the end of the alignment





How could you mofify the NW algorithm?

Needleman and Wunsch algorithm without end gaps

GLOBAL alignment, linear gap penalty

Initialization $F(0,0) = 0,\ F(i,0) = 0,\ F(0,j) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(A^i, B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Keep trace of the option that maximises F(i,j): match, gap1, or gap2

Termination *Best alignment score = Max* [F(a,j);F(i,b)]

Backtrace the optimal path

Needleman and Wunsch algorithm without end gaps

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

$d = 2$

Initialization

	0	**A**	**C**	**T**	**G**	**G**
0	0	0	0	0	0	0
A	0					
C	0					
C	0					
A	0					

Needleman and Wunsch algorithm without end gaps

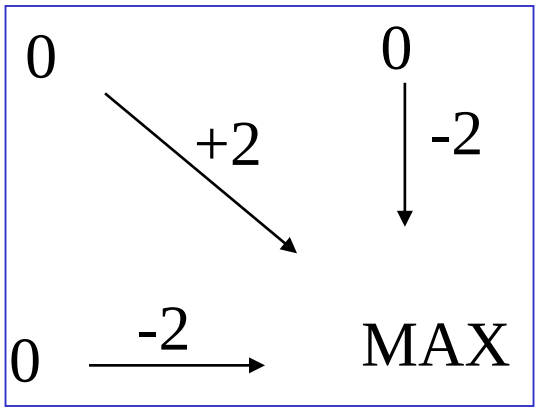
	A	C	T	G
A	2	-1	-1	0
C		2	0	-1
T			2	-1
G				2

$d = 2$

Iteration 1

	0	**A**	**C**	**T**	**G**	**G**
0	0	0	0	0	0	0
A	0	2				
C	0					
C	0					
A	0					

$$F(1,1) = MAX \begin{cases} F(1,0) - d \\ F(0,1) - d \\ F(0,0) + S(A,A) \end{cases}$$



Needleman and Wunsch algorithm without end gaps

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

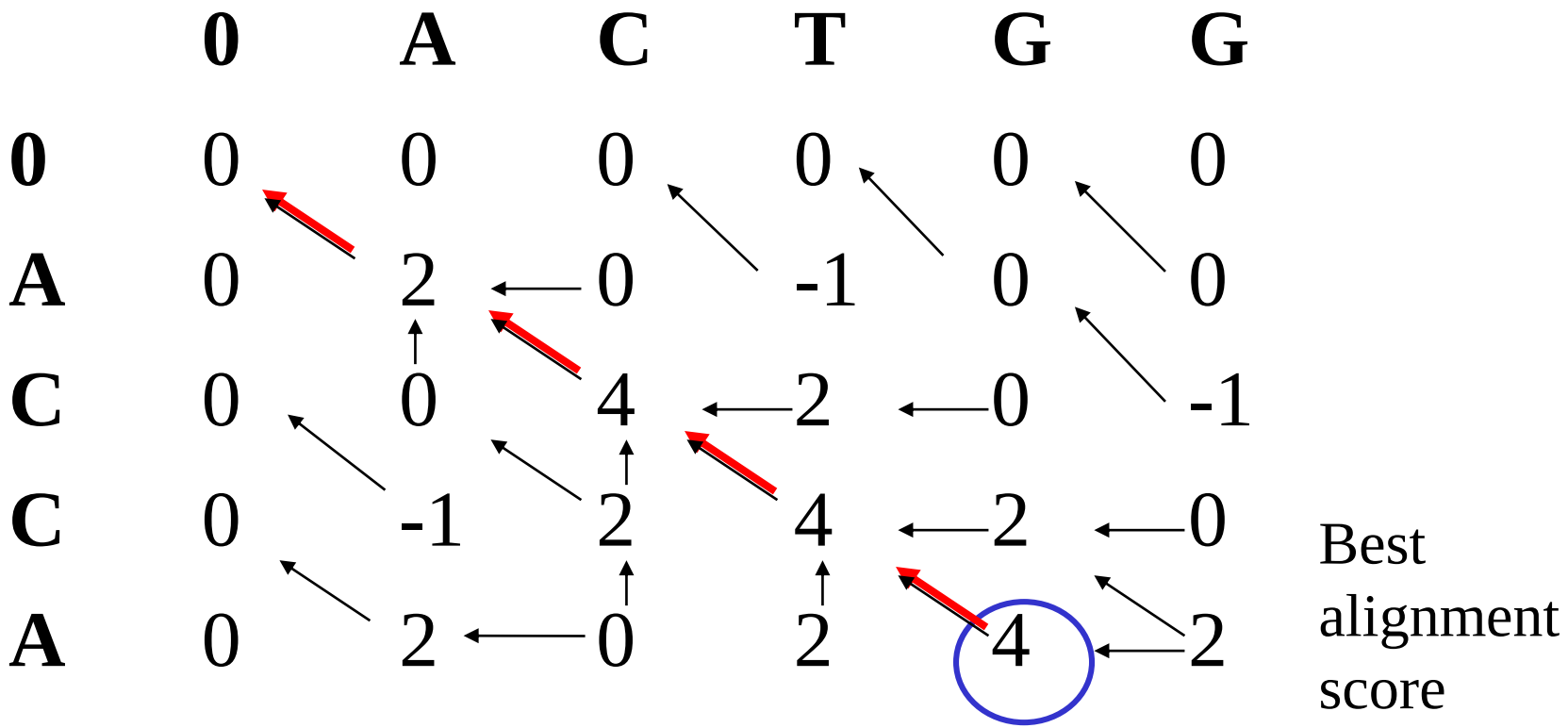
$d = 2$

Iterations

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

Needleman and Wunsch algorithm without end gaps

Termination



← Gap in sequence 2 ↑ Gap in sequence 1 ↖ Match

```
0 A C T G G
0 A C C A -
```

Local alignments

Since now GLOBAL alignment (involving the entire sequences) have been considered.

Sometimes it is useful to detect the portions of the sequences that can be better superimposed (functional or structural motifs, common exons….)

We adopt exactly the same strategy, with just a modification:

Negative scores are not accepted: Better to start a new alignment than to have negative score alignment

Smith and Waterman algorithm

Local alignment of sequences, linear gap penalty

Given two sequences *A* and *B*, *with lenghts a* and *b*, we introduce the (a+1)(b+1) matrix *F(i,j)* storing the score of the best LOCAL alignment between the substrings $0A^1A^2A^3\ldots\ldots A^i$ and $0B^1B^2B^3\ldots\ldots B^j$.

Initialization $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(A^i,B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \\ 0 \end{cases}$$

Termination The maximum value of matrix *F(i,j)* corresponds to the best local alignment

Smith and Waterman algorithm

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

$d = 2$

	0	**A**	**C**	**T**	**C**	**T**
0	0	0	0	0	0	0
A	0	2	0	0	0	0
T	0	0	2	0	0	2
T	0	0	0	4	2	2
A	0	2	0	2	3	1
A	0	2	1	0	1	2

Smith and Waterman algorithm

	0	**A**	**C**	**T**	**C**	**T**
0	0	0	0	0	0	0
A	0	2	0	0	0	0
T	0	0	2	0	0	2
T	0	0	0	4	2	2
A	0	2	0	2	3	1
A	0	2	1	0	1	2

← Gap in sequenza 2 ↑ Gap in sequenza 1 ↖ Match

```
0 A C T
0 A T T
```

The Smith-Waterman algorithm

Termination:

- If we want the best local alignment…

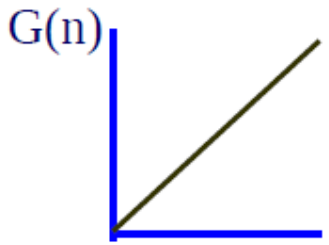
$$F_{OPT} = \max_{i,j} F(i, j)$$

- If we want all local alignments scoring > t

For all i, j find F(i, j) > t, and trace back

Scoring the gaps more accurately

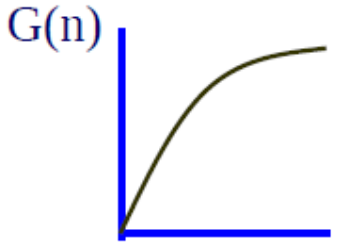
Current model:



Gap of length n
incurs penalty n×d

However, gaps usually occur in bunches

Concave gap penalty function:



G(n):
for all n, $G(n + 1) - G(n) \leq G(n) - G(n - 1)$

General gap dynamic programming

Initialization: same

Iteration:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ \max_{k=0\ldots i-1} F(k,j) - \gamma(i-k) \\ \max_{k=0\ldots j-1} F(i,k) - \gamma(j-k) \end{cases}$$

Termination: same

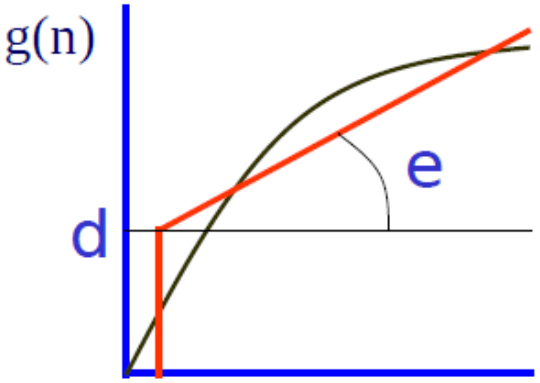
Running Time: $O(N^2M)$ (assume $N>M$)

Space: $O(NM)$

Compromise: **affine gaps**

$g(n) = d + (n - 1) \times e$

gap open | gap extend



To compute optimal alignment,

At position i,j, need to "remember" best score if gap is open
best score if gap is not open

F(i, j): score of alignment $x_1 \ldots x_i$ to $y_1 \ldots y_j$
if x_i aligns to y_j

G(i, j): score **if** x_i, or y_j, aligns to a gap

Needleman-Wunsch with affine gap (Gotoh algorithm)

Global alignment of sequences, affine gap penalty (h+g:opening, g:extension)

Given two sequences X and Y, *with lengths* x and y, we introduce **three** (x+1)(y+1) matrices

M(i,j): score of the best alignment between the substrings $0X^1X^2X^3....X^i$ and $0Y^1Y^2Y^3....Y^j$ **with X^i aligned to Y^j**

I_x(i,j): score of the best alignment between the substrings $0X^1X^2X^3....X^i$ and $0Y^1Y^2Y^3....Y^j$ **with X^i aligned to a gap**

I_y(i,j): score of the best alignment between the substrings $0X^1X^2X^3....X^i$ and $0Y^1Y^2Y^3....Y^j$ **with Y^j aligned to a gap.**

Needleman-Wunsch with affine gap (Gotoh algorithm)

Global alignment of sequences, affine gap penalty (h+g:opening, g:extension)

Given two sequences *X* and *Y*, *with lengths x* and *y*, we introduce **three** (x+1)(y+1) matrices

INITIALIZATION

$M(0,0) = 0$ $\quad M(0,j) = -\infty$ $\quad M(i,0) = -\infty$

$I_x(0,j) = -\infty$ $\quad I_x(i,0) = h + i \times g$

$I_y(i,0) = -\infty$ $\quad I_y(0,j) = h + j \times g$

Needleman-Wunsch with affine gap (Gotoh algorithm)

Global alignment of sequences, affine gap penalty (h+g:opening, g:extension)

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + s(x_i, y_j) & \text{match } x_i \text{ with } y_j \\ I_x(i-1,j-1) + s(x_i, y_j) & \text{insertion in } x \\ I_y(i-1,j-1) + s(x_i, y_j) & \text{insertion in } y \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j) + h + g & \text{open gap in } x \\ I_x(i-1,j) + g & \text{extend gap in } x \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1) + h + g & \text{open gap in } y \\ I_y(i,j-1) + g & \text{extend gap in } y \end{cases}$$

Needleman-Wunsch with affine gap (Gotoh algorithm)

Global alignment of sequences, affine gap penalty (h+g:opening, g:extension)

Given two sequences *X* and *Y*, *with lengths* *x* and *y*, we introduce **three** (x+1)(y+1) matrices

TERMINATION

- Look at the largest among the rightmost-bottom value:

$$M(x,y),\ I_x(x,y),\ I_y(x,y)$$

- Backtrace until reaching any of $M(0,0),\ I_x(0,0),\ I_y(0,0)$

- note that pointers may traverse all three matrices

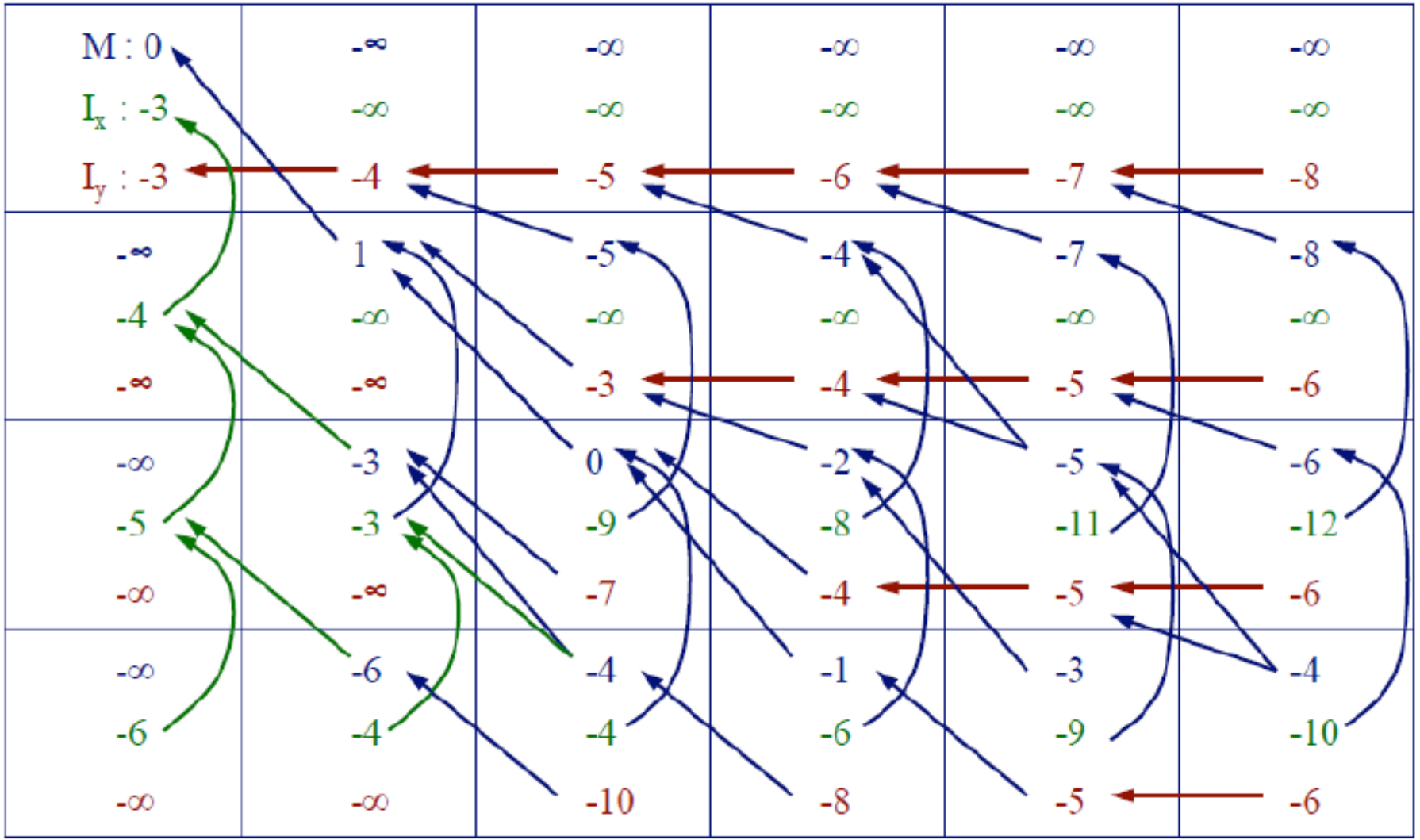
Needleman-Wunsch with affine gap (Gotoh algorithm)

$s(x_i, y_i) =$

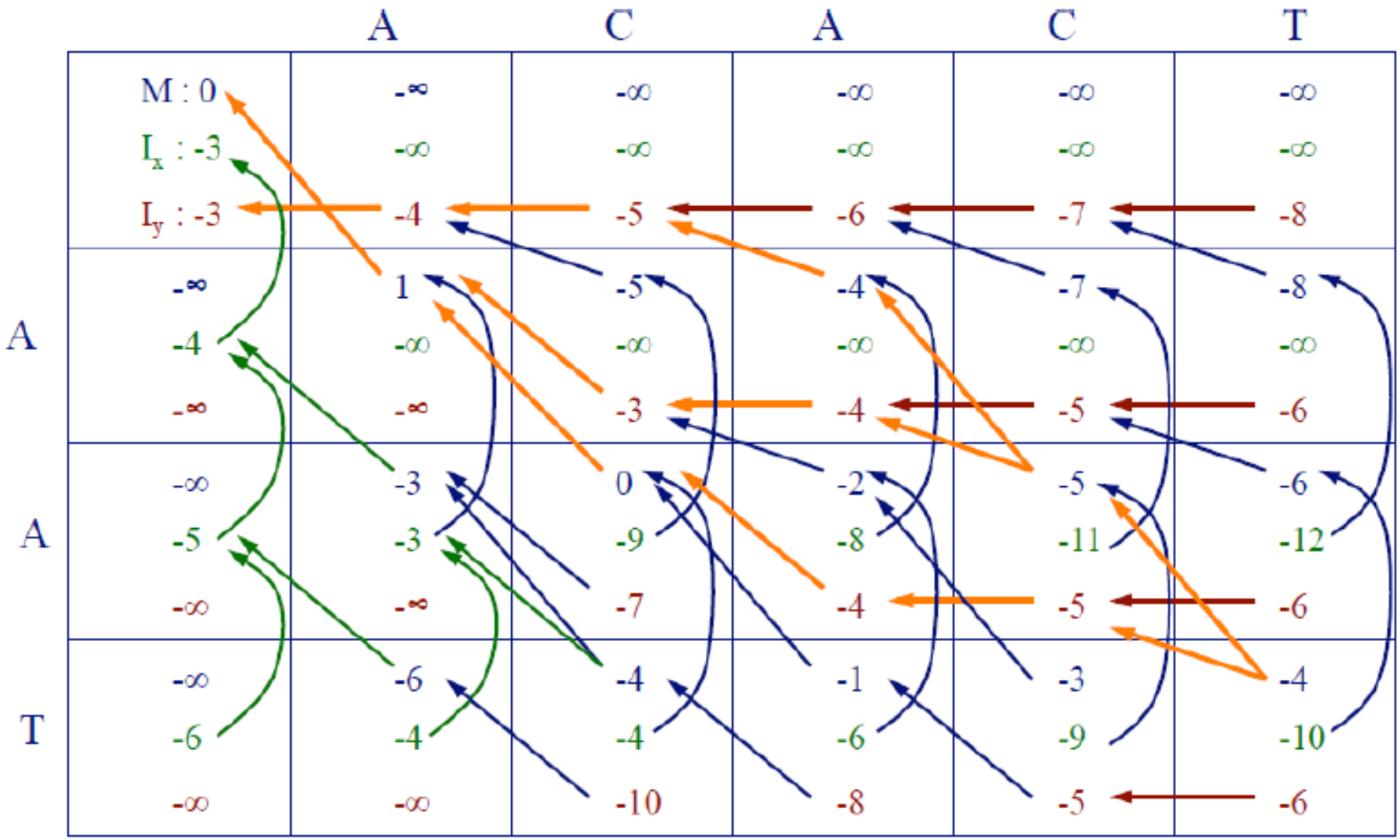
+1 when $x_i = y_i$

-1 when $x_i \neq y_i$

h = -3, g = -1



Needleman-Wunsch with affine gap (Gotoh algorithm)



three optimal alignments:

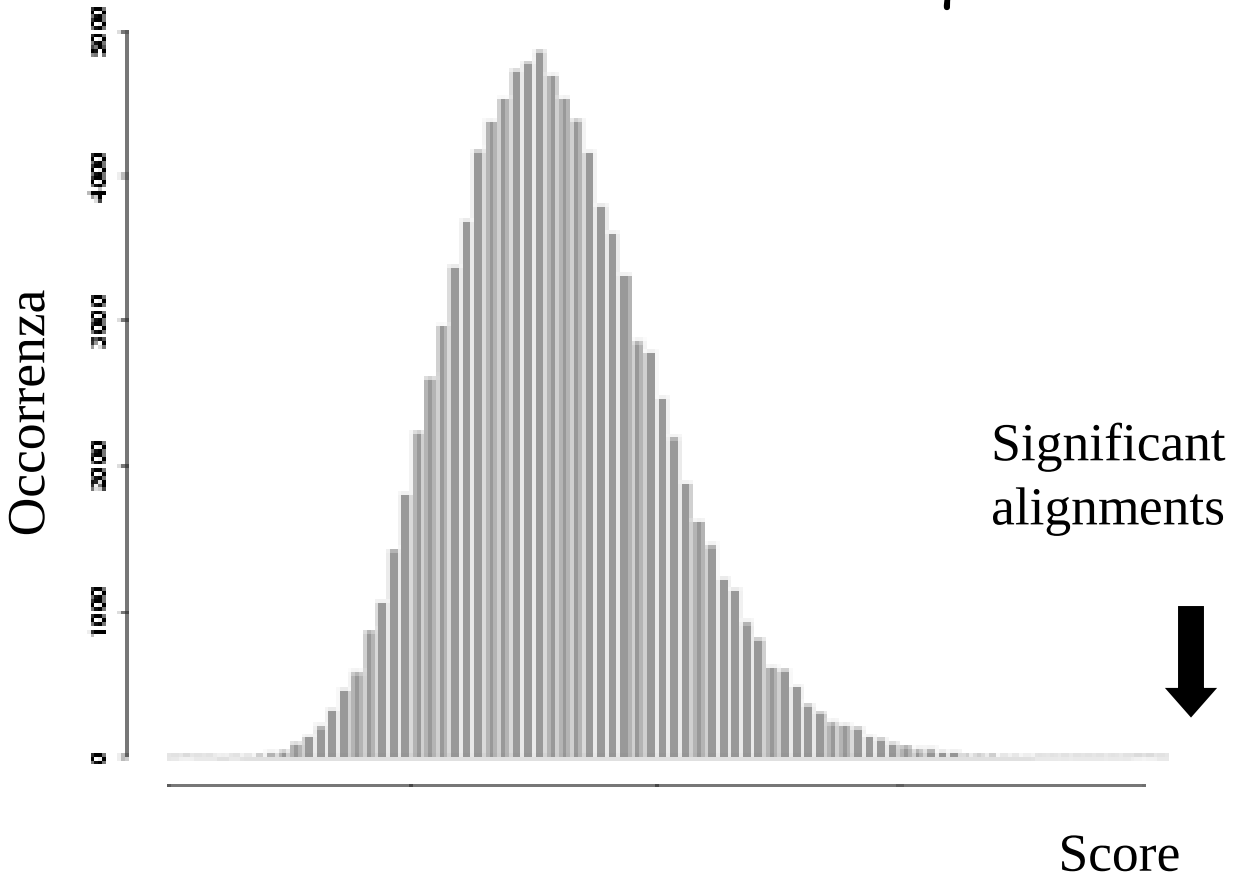
```
ACACT   ACACT   ACACT
AA--T   A--AT   --AAT
```

Significance of an alignment

Given an alignment with score S, is it significant?

How are the score of random alignments distributed?

100,000 alignments of unrelated and shuffled sequences:



Z-score

$Z=(S-\langle S\rangle)/\sigma_s$

S= Alignment score
$\langle S\rangle$= average of the scores on a random set of alignments
σ_s= Standard deviation of the scores on a random set of alignments

Significance of the alignment

$Z<3$	not significant
$3<Z<10$	probably significant
$Z>10$	significant

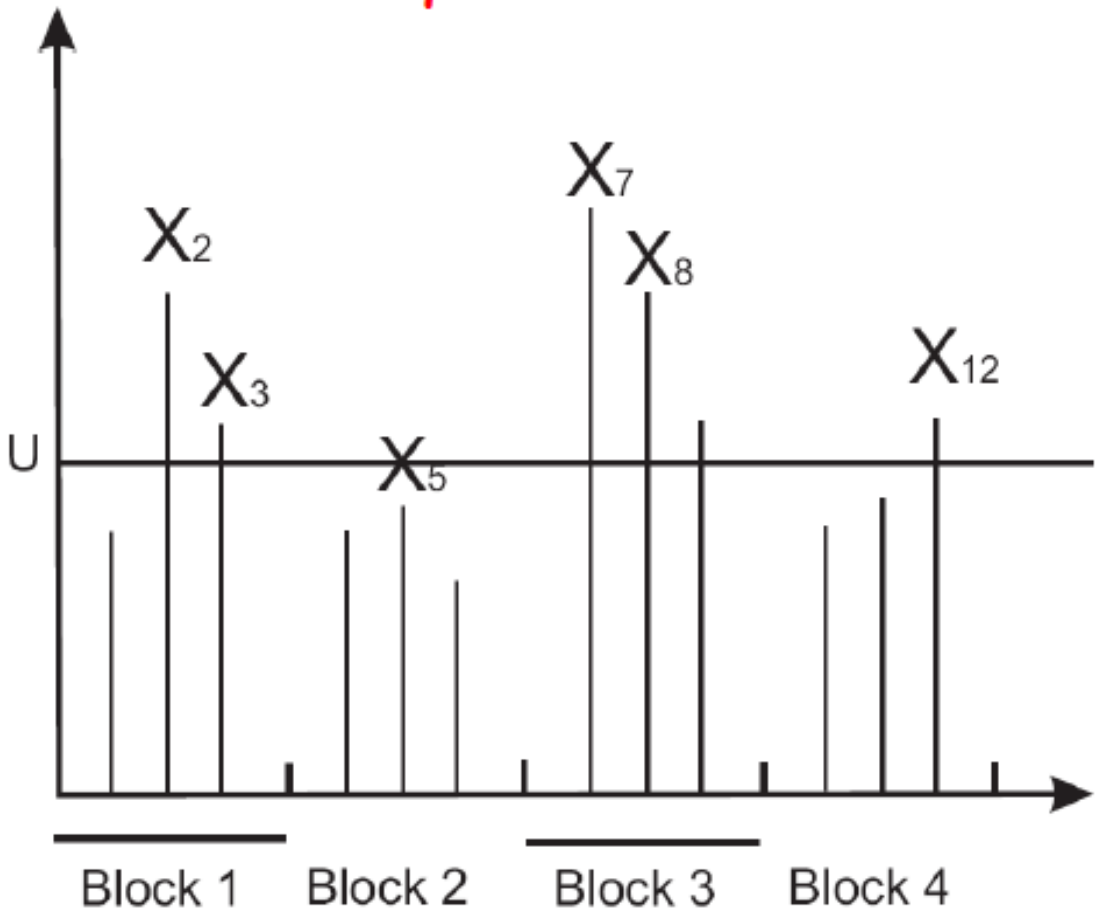
Is the Z-score reliable?

The Z-score of this alignment is 7.5 over 54 residues
Sequence identity is as high as 25.9%.
The sequences have a completely different structure

Citrate synthase (2cts) vs transthyritin (2paba)

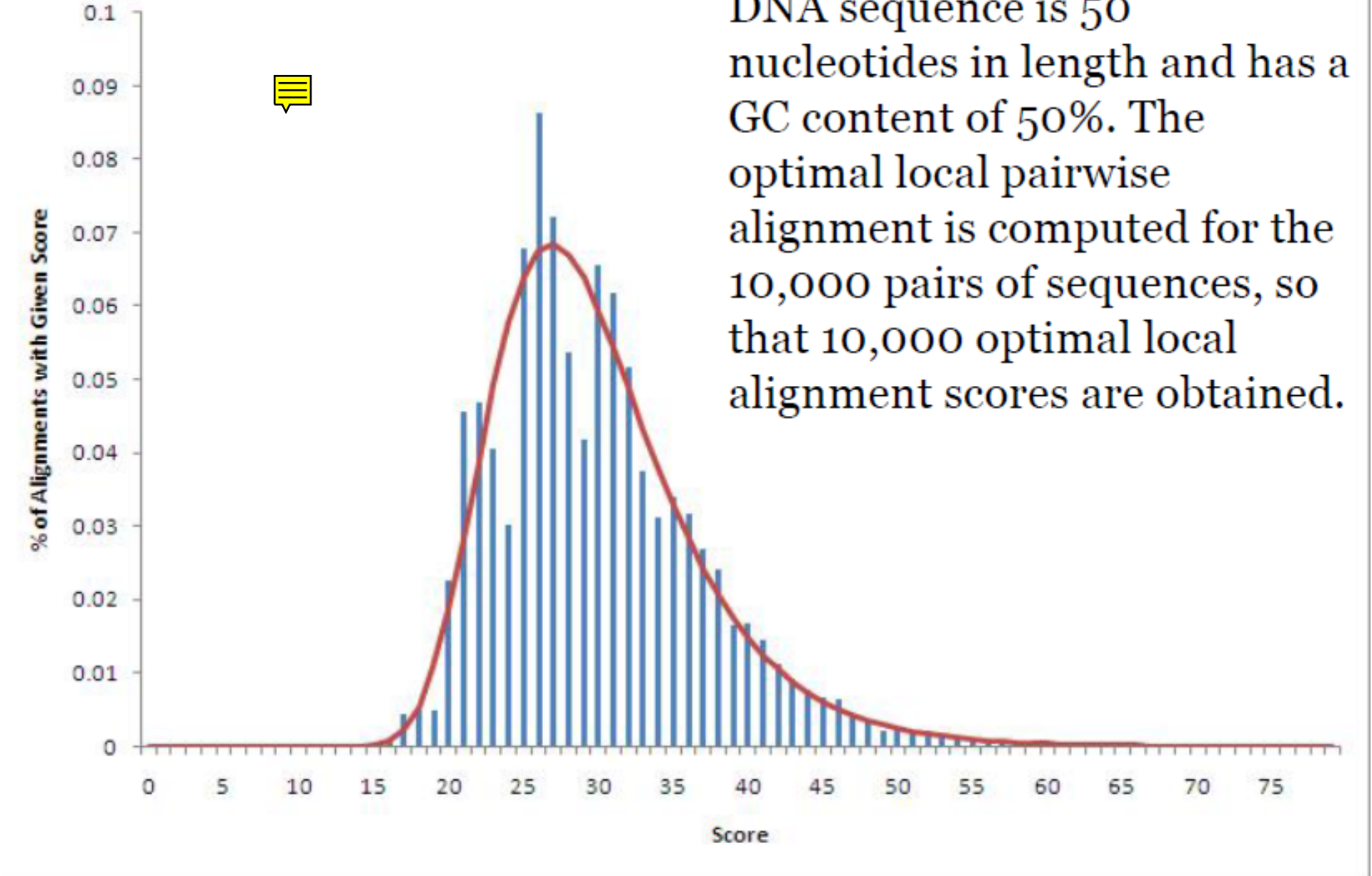
Why isn't Z-score reliable ?

We are measuring the BEST scores of the aligments between random sequences: NOT NORMAL



Extreme value theory states that, in the limit, extreme values (tails of distributions) are distributed by extreme value distributions, regardless of the parent distribution.

10,000 pairs of random DNA sequences are generated. Each DNA sequence is 50 nucleotides in length and has a GC content of 50%. The optimal local pairwise alignment is computed for the 10,000 pairs of sequences, so that 10,000 optimal local alignment scores are obtained.

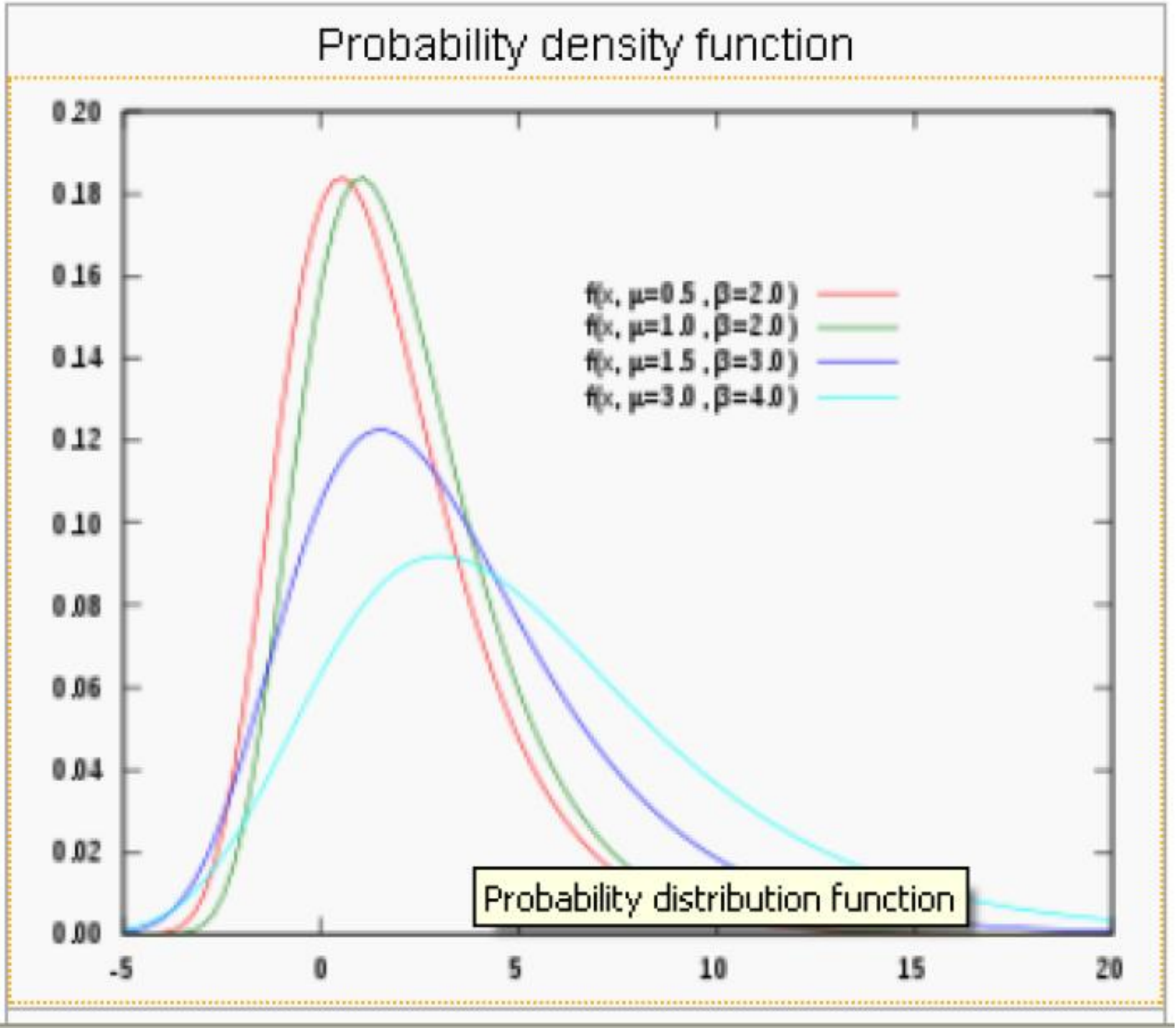


Extreme value distribution

- the **Gumbel distribution** is used to model the distribution of the maximum of a number of samples of various distributions.
- For example we would use it to represent the distribution of the maximum level of a river in a particular year if we had the list of maximum values for the past ten years. It is useful in predicting the chance that an extreme earthquake, flood or other natural disaster will occur.

Extreme value distribution

support:	$x \in (-\infty; +\infty)$
pdf:	$\frac{z\, e^{-z}}{\beta}$ where $z = e^{-\frac{x-\mu}{\beta}}$
cdf:	$\exp(-e^{-(x-\mu)/\beta})$
mean:	$\mu + \beta\,\gamma$
median:	$\mu - \beta \ln(\ln(2))$
mode:	μ
variance:	$\frac{\pi^2}{6}\beta^2$


Probability density function
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-5
0
5
10
15
20
f(x, μ=0.5 , β=2.0)
f(x, μ=1.0 , β=2.0)
f(x, μ=1.5 , β=3.0)
f(x, μ=3.0 , β=4.0)
Probability distribution function

E-value

Expected number of random alignments obtaining a score greater or equal to a given score (s)

P-value

Probability for random alignments to obtain a score greater or equal to a given score (s)

P-value

Given the E-value (expected number of alignments with score ≥ S), which statistics do describe the probability of having a number ***a*** of random alignments with score ≥ S?

Poisson: $P(a) = e^{-E} \frac{E^a}{a!}$

Which is the probability of finding at least one random alignment with score ≥ S?

$$Pvalue = P(a \geq 1) = 1 - P(0) = 1 - \exp(-E)$$

P-value = 1 - exp [-(E-value)]

E-Value	P-value
1.00E-10	1.000E-10
1.00E-05	1.000E-05
1.00E-04	1.000E-04
1.00E-03	9.995E-04
1.00E-02	9.950E-03
1.00E-01	9.516E-02
1.00E+00	6.321E-01
2.00E+00	8.647E-01
3.00E+00	9.502E-01
5.00E+00	9.933E-01
1.00E+01	1.000E+00

E-value

Alignment significance

The significance of the E-value depends on the length of the considered database. Considering Swiss Prot,

$E > 10^{-1}$	non significant
$10^{-1} > E > 10^{-3}$	probably not significant
$10^{-3} > E > 10^{-8}$	probably significant
$E < 10^{-8}$	significant

Similarity search in Data Bases

Given a sequence, search for similar sequences in huge data sets

In principle, the alignments between the query sequence and ALL the sequences in the data sets could be tried

Too many sequences!

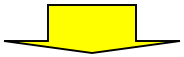
Heuristic algorithms can be used. They do not assure to find the optimal alignment

FASTA
BLAST

FASTA

The query sequence is chopped in words of k-tup characters. Usually k-tup = 2 for proteins, 6 for DNA

```
ADKLPTLPLRLDPTNMVFGHLRI
```



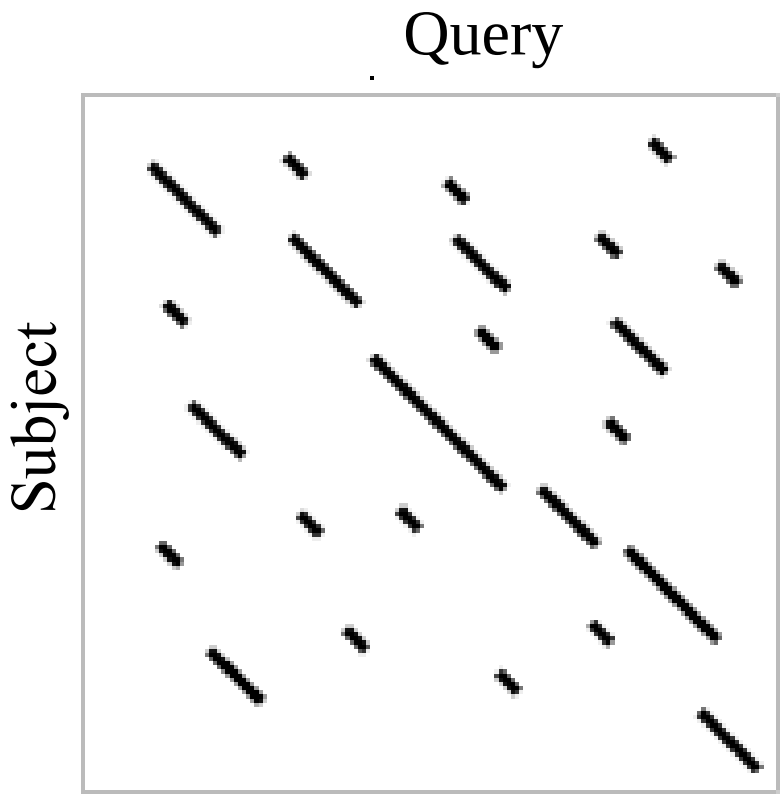
Words (indexed by position):

```
AD, DK, KL, LP, PT, TL, LP, PR, RL, …,…,
1   2   3   4   5   6   7   8   9   ….
```

The list of indexed words is compiled for each sequence in the data set (subject)

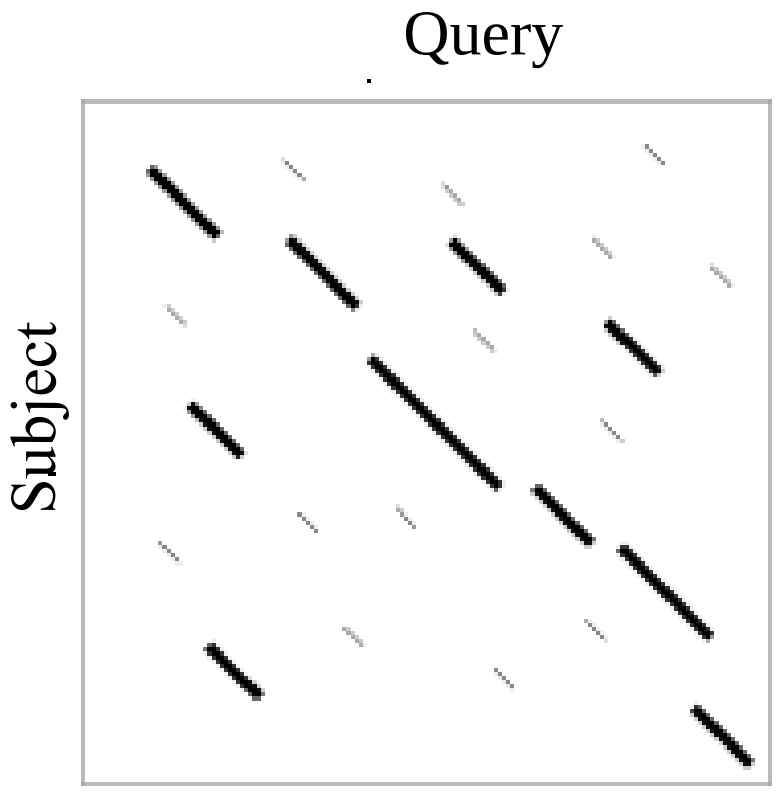
The search of the correspondence between the words is very fast.
The difference between the indexes of the matches in the query and the subject sequences determines the distance from the main diagonal

FASTA



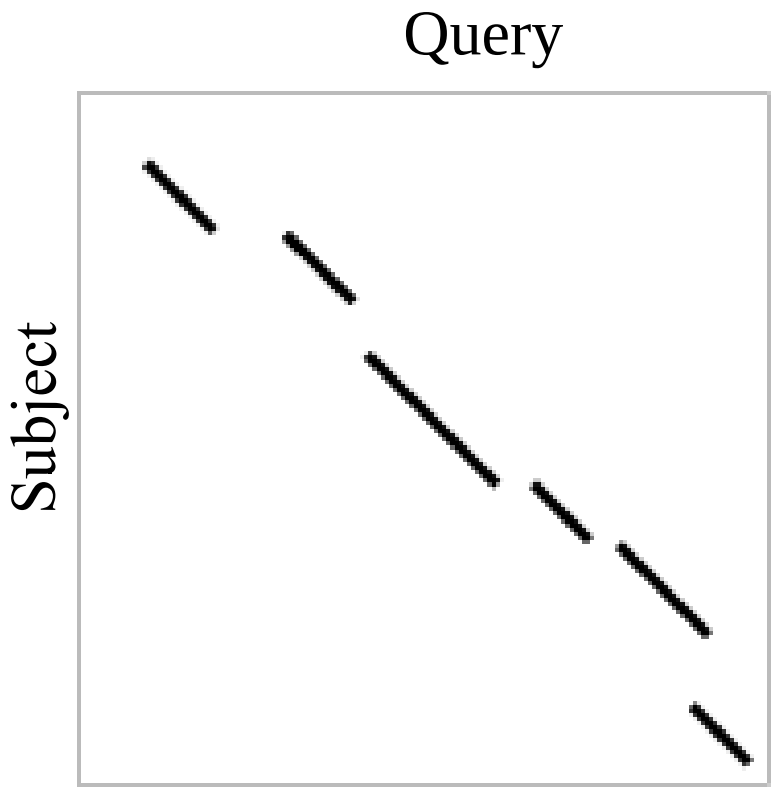
Many matches along the same diagonal correspond to longer identical segments along the sequences

FASTA



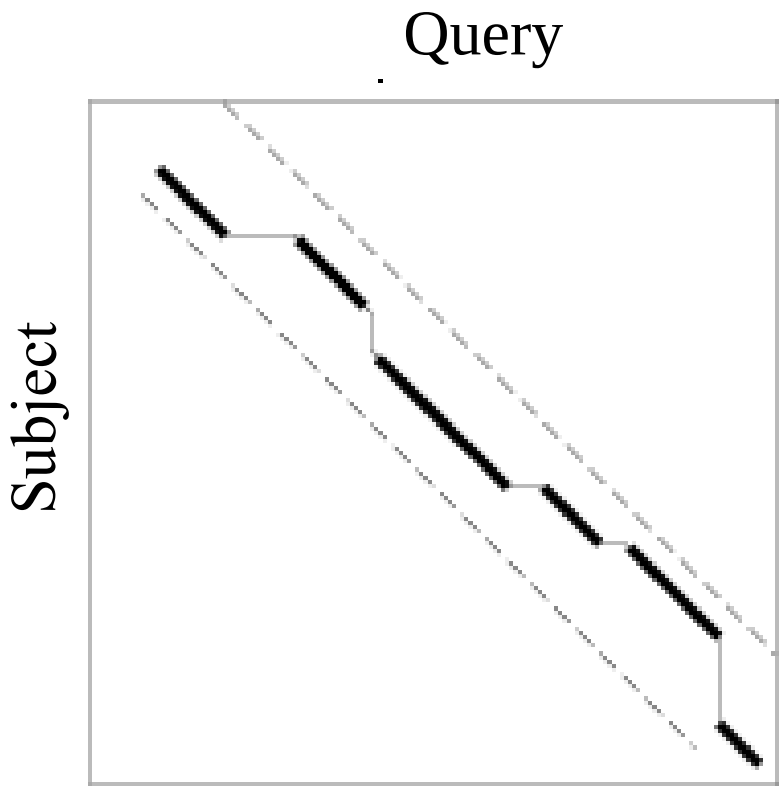
The alignment of the longest matched diagonals are evaluated with a score matrix (PAM or BLOSUM)

FASTA



Most similar regions on close diagonals are isolated

FASTA



An exact Smith-Waterman alignment is computed on a narrow band around the diagonal endowed with the highest similarity (a 32-residue band is usually adopted)

Sequence similarity with FASTA

Step 1 Identify regions shared by the two sequences with the highest density of identities (*ktup=1*) or pairs of identities (*ktup=2*).

Step 2 Rescan the ten regions with the highest density of identities using the BLOSUM50 matrix. Trim the ends of the region to include only those residues contributing to the highest score. Each region is a partial alignment without gaps.

Step 3 If there are several initial regions with scores greater than the CUTOFF value, check to see whether the trimmed initial regions can be joined to form an approximate alignment with gaps. Calculate a similarity score that is the sum of the joined initial regions minus a penalty (usually 20) for each gap (*initn*). The score of the single best initial region found in Step 2 is also reported (*init1*).

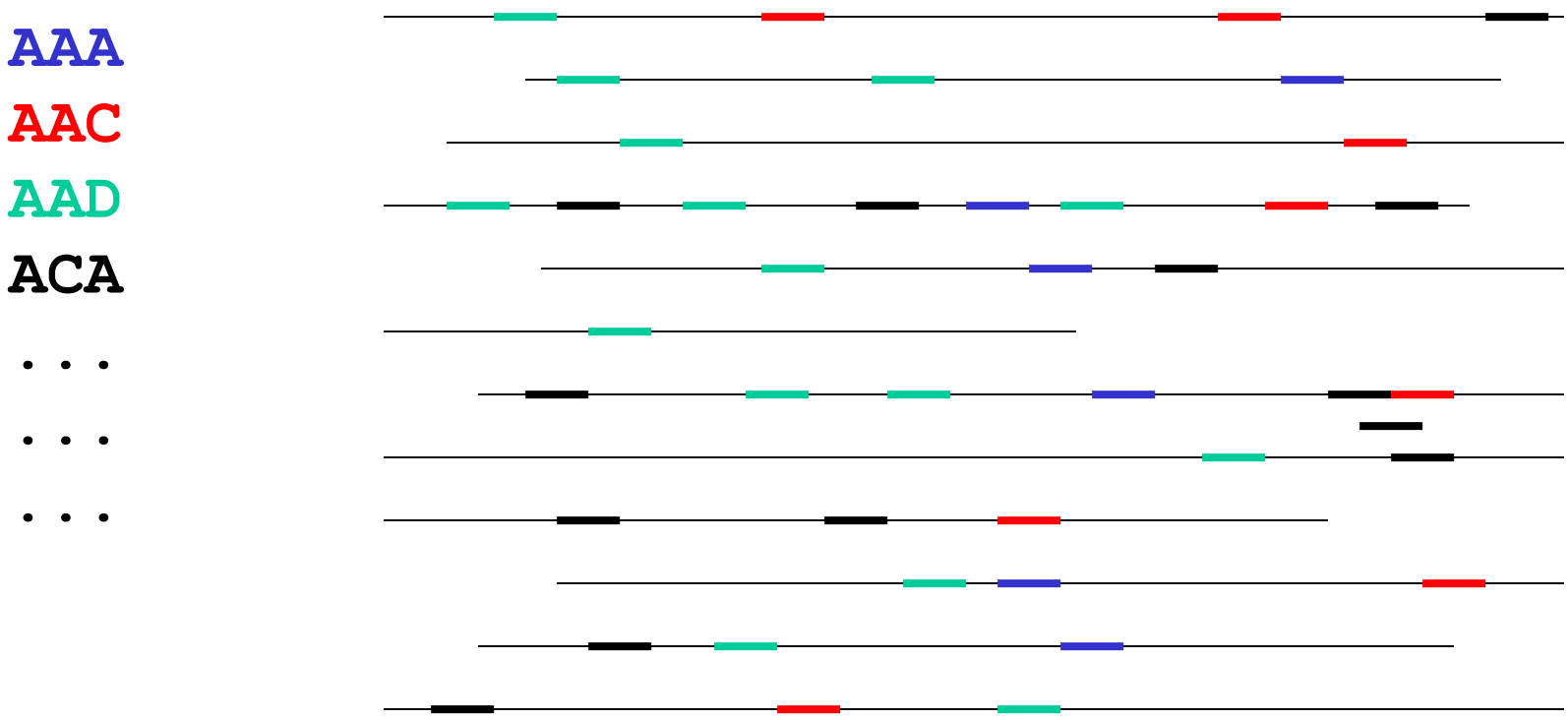
Step 4 For sequences with scores greater than a threshold, construct an optimal local alignment of the query sequence and the library sequence, considering only those residues that lie in a band centered on the best initial region found in Step 2. For protein searches with *ktup=2* a 16 residue band is used by default. A 32 residue band is used with *ktup=1*. This is the optimized (*opt*) score.

Step 5 After all (or the first 10–20,000) scores have been calculated, normalize the raw similarity scores by regressing the similarity score against ln(library-sequence length) and calculating the average variance in similarity scores. *Z-values* (normalized scores with mean 0 and variance 1) are calculated, and the calculation is repeated with library sequences with *z-values* greater than 5.0 and less than -5.0 removed. These z-values are used to rank the library sequences.

Step 6 The Smith-Waterman algorithm (without limitation on gap size) is used to display alignments.

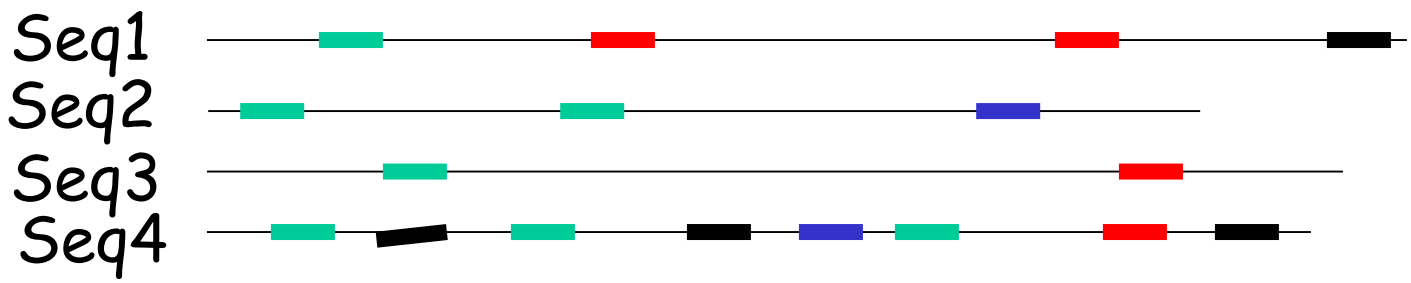
BLAST

The sequence **data set** is indexed as follow:
for each possible residue triplet the occurrence and the position along the sequences are stored in a dictionary (FORMATDB)



BLAST

The sequence **data set** is indexed as follow:
for each possible residue triplet the occurrence and the position along the sequences are stored in a dictionary (FORMATDB)



AAA→Seq2:*pos*; Seq4:*pos*

AAC→Seq1:*pos1*;Seq1:*pos2*;Seq3:*pos*;Seq4:*pos*

AAD→Seq1:*pos*;Seq2:*pos1*;Seq2:*pos2*;Seq3:*pos*;..

ACA→Seq1:*pos*;Seq4:*pos1*;Seq4:*pos2*;Seq4:*pos3*

. . .

. . .

. . .

BLAST

The **query** sequence is chopped in words, W-residue long (usually W=3 for proteins)

LSHLPTLPLRLDPTNMVFGHLRI

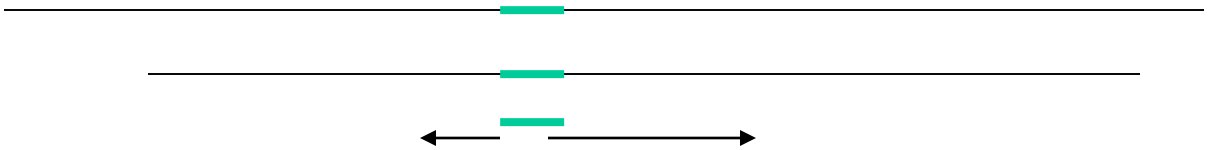
LSH, SHL, HLP, LPT, PTL, TLR, …,…,

For each word, all the similar proteins are generated, using the BLOSUM62 matrix and setting a similarity threshold (usually T = 11--13)

LSH	16
ISH	14
MSH	14
VSH	13
LAH	13
LTH	13
LNH	13

BLAST

Each word included in the list of the similar words is compared with the in the sequences in the data set by means of the indexes
All the matches are retrieved



The match is extended:
Until the score (with BLOSUM62) is higher than a threshold S (BLAST1.0)

When two non-overlapping hits are found within distance A on the same diagonal (BLAST2.0)

With gaps (gapped-BLAST, the current default)

Sequence similarity with BLAST (Basic Local Alignment Search Tool)

Step 1 For each three amino acids in the query sequence, identify all of the substitutions of each word that have a similarity score greater than a threshold score $T = 11$. In practice, word-matches with scores $\geq T$ are seen on average 150 times per library sequence.

Step 2 Build a discrete finite automaton (DFA) to recognize the list of identical and substituted three letter words.

Step 3 Use the DFA to identify all of the matching words in sequences in the database. If a match is found, attempt to extend the match both forwards and backwards using the BLOSUM62 matrix to produce a score that is higher than a threshold score. Save all of the high scoring regions shared by the query sequence and each library sequence. The best of these scores is reported as the best single MSP (maximal segment pair) score. These high scoring regions do not contain gaps.

Step 4 Attempt to combine multiple MSP regions. For each "consistent" combination, calculate the probability of obtaining that may consistent matches using either "poisson" or "sum" statistics.(Karlin & Altschul, 1993) Report the lowest probability score based on statistics used.

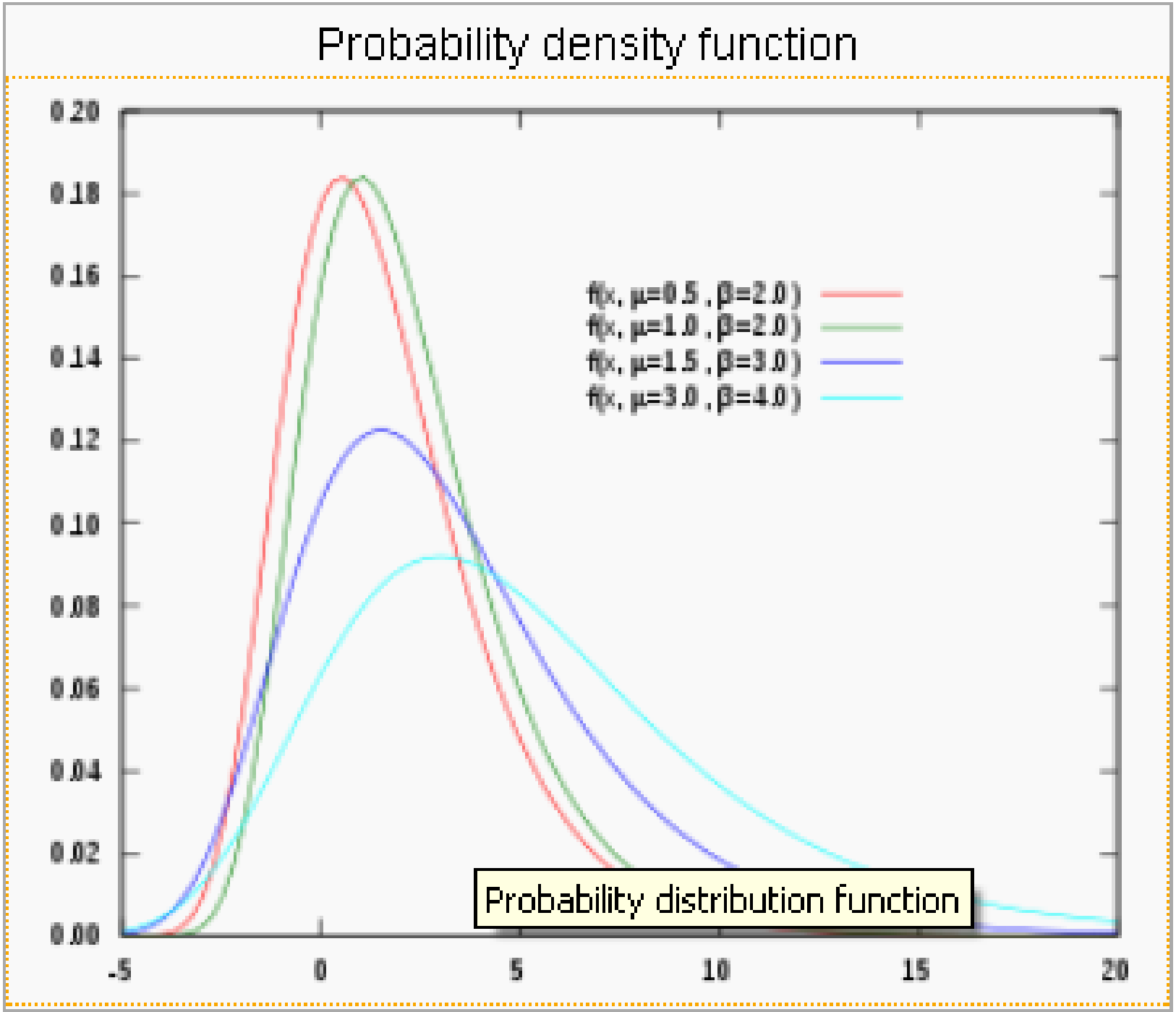
Step 5 Report all of the significant alignments. Frequently, a query and library sequence will contain several MSPs because of the requirement that they do not contain gaps.

P-value for BLAST
Extreme value distribution

- the **Gumbel distribution** is used to model the distribution of the maximum (or the minimum) of a number of samples of various distributions.
- For example we would use it to represent the distribution of the maximum level of a river in a particular year if we had the list of maximum values for the past ten years. It is useful in predicting the chance that an extreme earthquake, flood or other natural disaster will occur.

Extreme value distribution

support:	$x \in (-\infty; +\infty)$
pdf:	$\frac{z\, e^{-z}}{\beta}$ where $z = e^{-\frac{x-\mu}{\beta}}$
cdf:	$\exp(-e^{-(x-\mu)/\beta})$
mean:	$\mu + \beta\,\gamma$
median:	$\mu - \beta \ln(\ln(2))$
mode:	μ
variance:	$\frac{\pi^2}{6}\beta^2$


Probability density function
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-5
0
5
10
15
20
f(x, μ=0.5 , β=2.0)
f(x, μ=1.0 , β=2.0)
f(x, μ=1.5 , β=3.0)
f(x, μ=3.0 , β=4.0)
Probability distribution function

P-value for BLAST

- In accordance with the Gumbel Extreme Value Distribution, the probability p of observing a score S equal to or greater than x is given by the equation

$$p(S \geq x) = 1 - \exp\left(-e^{-\lambda(x-\mu)}\right)$$

$$\mu = {[\log(Km'n')]}/{\lambda}$$

m, n: lengths of the sequences
K, λ: "scaling" constants

P(a ≥ 1) = 1 – P(0) = 1 – exp (-E)

E-value

Expected number of random alignments obtaining a score greater or equal to a given score (s)

$$p(S \geq x) = 1 - \exp\left(-e^{-\lambda(x-\mu)}\right) \qquad \mu = {[\log(Km'n')]}/{\lambda}$$

P= 1 - exp (-E)

Then

$E=Kmn\, e^{-\lambda s}$

m, n: lengths of the sequences
K, λ: "scaling" constants

The number of high scoring random alignment increases when the sequence lengths increase and decreases in an exponential way when the score increases.

PSIBLAST-
Alignment of all the retrieved sequences

ATTENTION: It is not a multiple sequence alignment

Sequence profile

1	Y	K	D	Y	H	S	-	D	K	K	K	G	E	L	-	-
2	Y	R	D	Y	Q	T	-	D	Q	K	K	G	D	L	-	-
3	Y	R	D	Y	Q	S	-	D	H	K	K	G	E	L	-	-
4	Y	R	D	Y	V	S	-	D	H	K	K	G	E	L	-	-
5	Y	R	D	Y	Q	F	-	D	Q	K	K	G	S	L	-	-
6	Y	K	D	Y	N	T	-	H	Q	K	K	N	E	S	-	-
7	Y	R	D	Y	Q	T	-	D	H	K	K	A	D	L	-	-
8	G	Y	G	F	G	-	-	L	I	K	N	T	E	T	T	K
9	T	K	G	Y	G	F	G	L	I	K	N	T	E	T	T	K
10	T	K	G	Y	G	F	G	L	I	K	N	T	E	T	T	K

Position →

A	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	70	0	0	0	0	60	0	0	0	0	20	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0
F	0	0	0	10	0	33	0	0	0	0	0	0	0	0	0	0
G	10	0	30	0	30	0	100	0	0	0	0	50	0	0	0	0
H	0	0	0	0	10	0	0	10	30	0	0	0	0	0	0	0
K	0	40	0	0	0	0	0	0	10	100	70	0	0	0	0	100
I	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0
N	0	0	0	0	10	0	0	0	0	0	30	10	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	40	0	0	0	30	0	0	0	0	0	0	0
R	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	33	0	0	0	0	0	0	10	10	0	0
T	20	0	0	0	0	33	0	0	0	0	0	30	0	30	100	0
V	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
W	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	70	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0

Usefulness of the sequence profiles

Sequence profiles describes the basic features of all the sequence used in the alignment

Most conserved regions and most frequent mutations for each position are highlighted

Sequence-to-profile alignment

The alignment score are weighted position by position using the profile. The same mutations in different positions are scored with different values

How can we exploit the information contained in profiles? Sequence-to-profile alignment

Given the position *i* along a sequence profile, it is represented by a 20-valued vector $P_i = P_i(A)\ P_i(C)\ \ldots\ldots\ P_i\ (Y)$

A	0
C	0.85
D	0
E	0
F	0.05
G	0
H	0
I	0
K	0
L	0.02
M	0
N	0.08
P	0
Q	0
R	0
S	0
T	0
V	0
W	0
Y	0

Given the residue in position *j* along the sequence to align: S_j
The score for aligning S_j to the vector P_i is:

$$Score(i, j) = \sum_{k=1}^{20} P_i(r_k) \cdot M(r_k, s_j)$$

where M is a matrix score (BLOSUM or PAM)
The score can be used in dynamic programming procedures (Needleman-Wunsch, Smith-Waterman)

Sequence-to-profile alignment

P_i =

A	0
C	0.85
D	0
E	0
F	0.05
G	0
H	0
I	0
K	0
L	0.02
M	0
N	0.08
P	0
Q	0
R	0
S	0
T	0
V	0
W	0
Y	0

S_j = "C"

Alignment score between Pi and Si is

$$Score(i,j) = \sum_{k=1}^{20} P_i(r_k) \cdot M(r_k, s_j)$$

=0.85*M(C,C)+0.05*M(C,F)+
+0.02*M(C,L)+0.08*M(C,N) =

=0.85*9+0.05*(-2)+
+0.02*(-1)+0.08*(-3)=

=7.29

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Needleman-Wunsch algorithm

Global alignment, linear gap approximation

Given a profile P and a sequence B, with lengths *a* and *b*, define the matrix $F(i,j)$ storing the score of the best alignment between the subsequences: $P^1P^2P^3.....P^i$ e $B^1B^2B^3.....B^j$.

Initialisation $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(P^i,B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Conclusion $F(a,b)$ the best alignment score

Here the substitution between the profile position i and the residue j of the B sequence

Smith and Waterman algorithm

Local alignment, linear gap approximation

Given a profile P and a sequence B, with lengths *a* and *b*, define the matrix $F(i,j)$ storing the score of the best local alignment between the subsequences: $P^1P^2P^3.....P^i$ e $B^1B^2B^3.....B^j$.

Initialisation $F(0,0) = 0$

Iteration

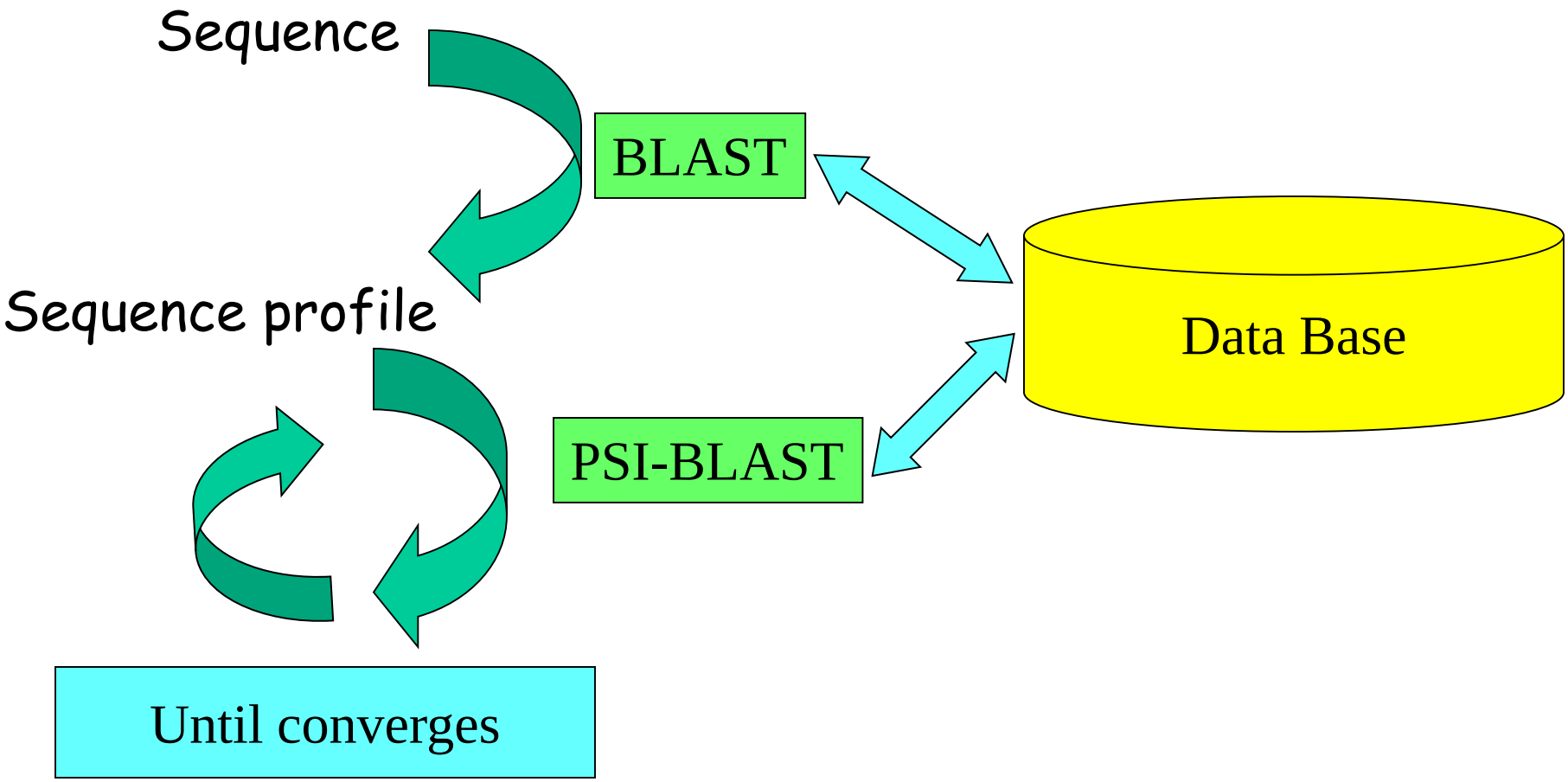
$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(P^i,B^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \\ 0 \end{cases}$$

Conclusion The maximum $F(i,j)$ gives the score of the best local alignment

Here the substitution between the profile position i and the residue j of the B sequence

PSI-BLAST

http://www.ncbi.nlm.nih.gov/BLAST/



The design of PSI-BLAST

(1) PSI-BLAST takes as an input a single protein sequence and compares it to a protein database, using the gapped BLAST program
(2) The program constructs a multiple alignment, and then a profile, from any significant local alignments found. The original query sequence serves as a template for the multiple alignment and profile, whose lengths are identical to that of the query. Different numbers of sequences can be aligned in different template positions
(3) The profile is compared to the protein database, again seeking local alignments. After a few minor modifications, the BLAST algorithm can be used for this directly.
(4) PSI-BLAST estimates the statistical significance of the local alignments found. Because profile substitution scores are constructed to a fixed scale, and gap scores remain independent of position, the statistical theory and parameters for gapped BLAST alignments remain applicable to profile alignments.
(5) Finally, PSI-BLAST iterates, by returning to step (2), an arbitrary number of times or until convergence.

What about comparing simultaneously a group of sequences

Multiple Sequence Alignment

What is a multiple alignment?

- a representation of a set of sequences, where equivalent residues (e.g. functional, structural) are aligned in rows or more usually columns

Example: part of an alignment of SH2 domains from 14 sequences

* conserved identical residues
: conserved similar residues

```
                  *     *     :       * **:*                          *  :                :  :  ::
lnk_rat   ----------YPWFHGPISRVRAAQLVQLQGPDAHGVFLVRQSESRR-GEYVLTFNLQ-----GRAKHLRLVLTERGQCRVQH--LHFPSVVDML
crk1_mouse----------SAWYMGPVTRQEAQTRLQGQR---HGMFLVRDSSTCP-GDYVLSVSEN-----SRVSHYIINSLPNRRFKIGD--QEFDHLPALL
nck_human ----------WYYGKVTRHQAEMALNERGH--EGDFLIRDSESSP-NDFSVSLKAQ-----GKNKHFKVQLK-ETVYCIGQ--RKFSTMEELV
ht16_hydat----------WYHGKITREVAVQVLLRKGGR-DGFFLIRDCGNAP-EDYVLSMMFR-----SQILHFQINCLGDNKFSIDNG-PIFQGLDMLI
pip5_human----------KPWYYDSLSRGEAEDMLMRIPR--DGAFLIRKREGS--DSYAITFRAR-----GKVKHCRINRDG-RHFVLGTS-AYFESLVELV
fer_human ----------WYHGAIPRIEAQELLKK-----QGDFLVRESHGKP-GEYVLSVYSD-----GQRRHFIIQYV-DNMYRFEG--TGFSNIPQLI
1ab2      ----------EEWFHGVLPREEVVRLLNN-----DGDFLVRETIRNEESQIVLSVCW------NGHKHFIVQTTGEGNFRFEG--PPFASIQELI
1mil      ----------HSWYHGPVSRNAAEYLLSSGI---NGSFLVRESESSP-GQRSISLRYE-----GRVYHYRINTASDGKLYVSSE-SRFNTLAELV
1blj      ----------EPWFHGKLSRREAEALLQL-----NGDFLVRESTTTP-GQYVLTGLQS-----GQPKHLLLVDP-EGVVRTKD--HRFESVSHLI
1shd      GSVAPVETLEVEKWFFRTISRKDAERQLLAPMNK-AGSFLIRESESNK-GAFSLSVKDITTQ-GEVVKHYKIRSLDNGGYYISPR-ITFPTLQALV
1lkkA     -------SIQAEEWYFGKITRRESERLLLNAENP-RGTFLVRESEA-----YCLSVSDFDNAKGLNVKHYKIRKLDSGGFYITSR-TQFNSLQQLV
1csy      --------LEPEPWFFKNLSRKDAERQLLAPGNT-HGSFLIRESESTA-GSFSLSVRDFDQNQGEVVKHYKIRNLDNGGFYISPR-ITFPGLHELV
1bfi      -------SHEKMPWFHGKISREESEQIVLIGSKT-NGKFLIRARDNN--GSYALCLLHE-----GKVLHYRIDKDKTGKLSIPEG-KKFDTLWQLV
1gri      -------HHDEKTWNVGSSNRNKAENLLRGKR---DGTFLVRESSKQ--GCYACSVVVD-----GEVKHCVINKTATG-YGFAEPYNLYSSLKELV
1lck      -------EMKPHPWFFGKIPRAKAEEMLSKQRH--DGAFLIRESESAP-GDFSLSVKFG-----NDVQHFKVLRDGAGKYFLWV--VKFNSLNELV
```

Multiple sequence alignments

How to score an alignment of many sequences?

Given M sequences A_i , we can define a score for the multiple sequence alignment as the sum of the scores of all the induced pair alignments

$$S=\sum_{i<j} S(A_i,A_j)$$

$$S\left\{\begin{array}{l}\texttt{1>ASPTLPLSLA}\\ \texttt{2>SS-TLPA--A}\\ \texttt{3>SSPTLPA--A}\end{array}\right\} =$$

$$=S\left\{\begin{array}{l}\texttt{1>ASPTLPLSLA}\\ \texttt{2>SS-TLPA--A}\end{array}\right\} +S\left\{\begin{array}{l}\texttt{1>ASPTLPLSLA}\\ \texttt{3>SSPTLPA--A}\end{array}\right\} +S\left\{\begin{array}{l}\texttt{2>SS-TLPA--A}\\ \texttt{3>SSPTLPA--A}\end{array}\right\}$$

Multiple sequence alignments

The algorithmic problem is to find the alignment with the maximum score

Exact algorithms

Algorithms based of multi-dimensional dynamic programming have been implemented. However they are too slow when many sequences have to be compared.

Progressive alignments

Iterative algorithms

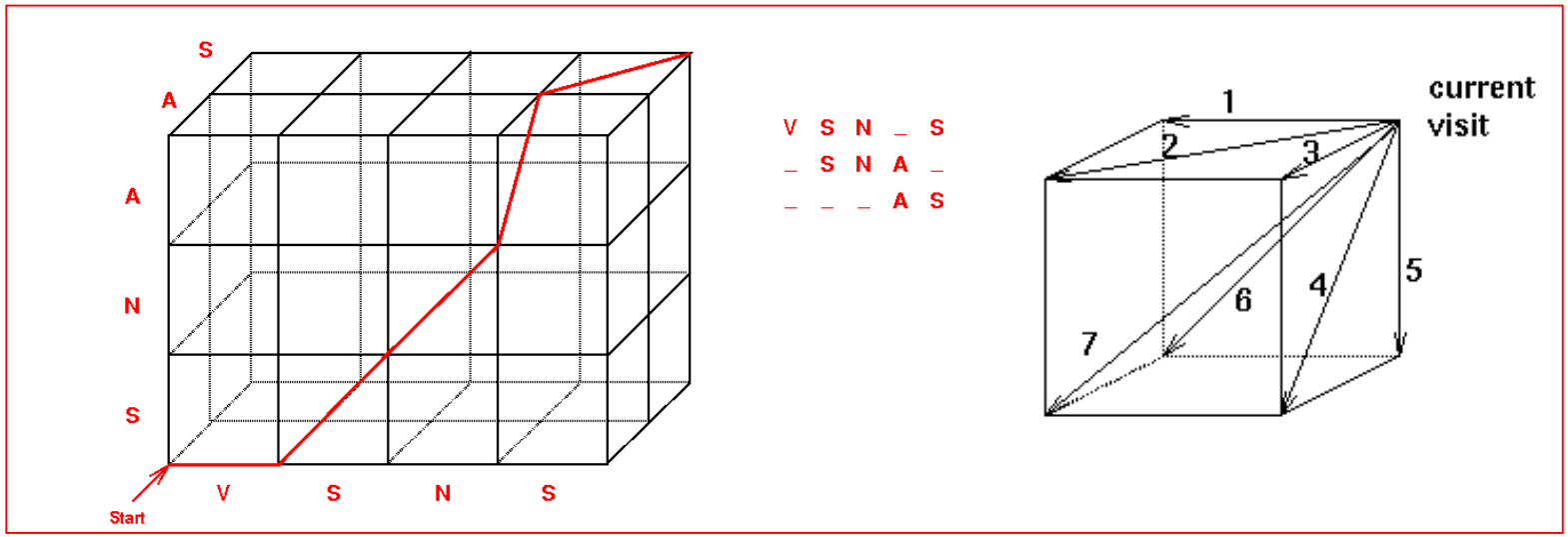
Consistency-based algorithms

Optimal multiple alignment

- Optimal multiple alignment
  MSA (Lipman et al. 1989, Gupta et al. 1995)

Extension of dynamic programming for 2 sequences => N dimensions

Example : alignment of 3 sequences



Problem : calculation time and memory requirements

Time proportional to N^k for k sequences of length N => limited to less than 10 sequences

Progressive multiple alignment

Idea :

Progressively align pairs of sequences (or groups of sequences)

Problem :

Start with which sequences ? How to decide order of alignment ?

↳ *first align the most closely related sequences*

How to measure the similarity of the sequences ?

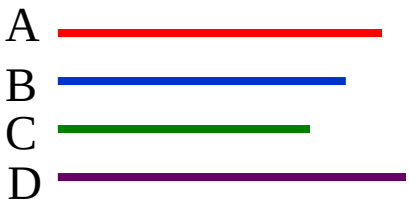
↳ *align all the sequences pairwise*

↳ *calculate the similarity between each pair from the alignment*

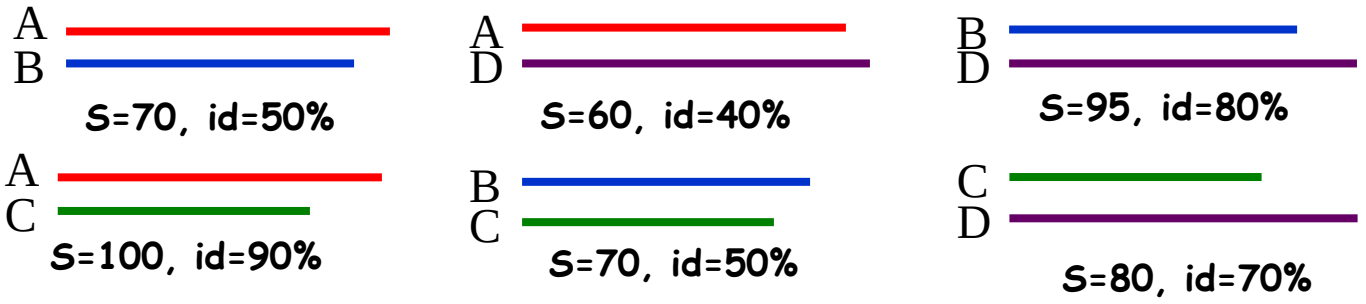
ClustalW (Thompson et al. NAR. 1994)

ClustalX (Thompson et al. NAR. 1997

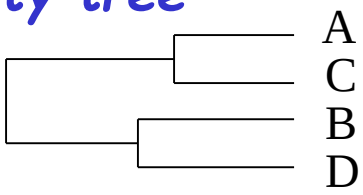
Progressive multiple alignments



Step1: Pairwise sequence alignment: exact, all-against-all



Step2: Build a similarity tree



Progressive multiple alignments

Step 3: Exact alignment of the most similar sequences, following the tree

Step 4: Build the profile from the sub alignments)

Step 5: Perfom profile-to profile alignment following the similarity tree, until comprising all the sequences

Profile-to-profile score

The position *i* along the first sequence profile, it is represented by a 20-valued vector $P^1_i = P^1_i(A)\ P^1_i(C)\ \ldots\ P^1_i(Y)$

The position *j* along the second sequence profile, it is represented by a 20-valued vector $P^2_j = P^2_j(A)\ P^2_j(C)\ \ldots\ P^2_j(Y)$

The score for aligning the two positions is

$$Score(i, j) = \sum_{m=1}^{20} \sum_{k=1}^{20} P^1{}_i(r_m) P^2{}_j(r_k) \cdot M(r_m, r_k)$$

where M is a matrix score (BLOSUM or PAM)
The score can be used in dynamic programming procedures (Needleman-Wunsch, Smith-Waterman)

Needleman-Wunsch algorithm

Global alignment, linear gap approximation

Given two profiles P and R, with lengths *a* and *b*, define the matrix $F(i,j)$ storing the score of the best alignment between the subsequences: $P^1P^2P^3.....P^i$ e $R^1R^2R^3.....R^j$.

Initialisation $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max} \begin{cases} F(i-1,j-1) + s(P^i,R^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

Here the substitution between the profile positions i and j of profiles P and R

Conclusion $F(a,b)$ the best alignment score

Smith and Waterman algorithm

Local alignment, linear gap approximation

Given two profiles P and R, with lengths *a* and *b*, define the matrix $F(i,j)$ storing the score of the best local alignment between the subsequences: $P^1P^2P^3.....P^i$ e $R^1R^2R^3.....R^j$.

Initialisation $F(0,0) = 0$

Iteration

$$F(i,j) = \text{Max}\begin{cases} F(i-1,j-1) + s(P^i,R^j) \\ F(i-1,j) - d \\ F(i,j-1) - d \\ 0 \end{cases}$$

Here the substitution between the profile positions i and j of the two profiles

Conclusion The maximum $F(i,j)$ gives the score of the best local alignment

Problems with Progressive algorithms

Dependence of the ultimate MSA on the initial pairwise sequence alignment with the highest score

Errors in initial alignments are propagated

Gaps can proliferate, if not careful

Gaps can be amino-acid specific, so that you penalize introduction of gaps into segments that are less likely to have gaps (e.g. hydrophobic core)

"once a gap, always a gap"

Where gaps are added is a critical question

Gaps are often added to the first two (closest) sequences

To change the initial gap choices later on, would be to give more weight to distantly related sequences

To maintain the initial gap choices is to trust that those gaps are most believable