

---

# EE381V: Genomic Signal Processing



## Lecture #5

---

### Last Lecture: Alignment Problems

There are four basic alignment problems:

**1. Global alignment**

- Basic global similarity. Essential when designing certain biosensors.

**2. Local alignment**

- Biologically often far more meaningful than global similarity – especially when long stretches of non-coding DNA are compared, since only small regions within those strings may be related. Likewise for proteins.

**3. Ends-free (overlap) alignment**

- Useful in the shotgun sequence assembly procedure. Here, a large set of partially overlapping substrings that come from many copies of one original but unknown DNA sequences. Suffix/prefix detection essential.

**4. Alignment with a gap penalty function**

- Useful when, e.g., comparing mRNA against the genome to reveal exons (since this alignment allows special treatment of long gaps)

2

## Global Alignment: Needleman-Wunsch Algorithm

- Let  $F(i,j)$  denote the score of the best alignment of  $s[1\dots i]$  and  $t[1\dots j]$
- Recursively compute  $F(i,j)$  and fill the matrix

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + c(s_i, t_j) \\ F(i-1,j) + c(s_i, -) \\ F(i,j-1) + c(-, t_j) \end{cases}$$

$F(i-1,j-1)$	$F(i-1,j)$
$F(i,j-1)$	$F(i,j)$

match=+1

mismatch=-1

gap=-2.

		A G C			
	t	0	1	2	3
s					
	0	0	-2	-4	-6
A	1	-2	1	-1	-3
A	2	-4	-1	0	-2
A	3	-6	-3	-2	-1
C	4	-8	-5	-4	-1

- The best alignment score of **AAAC** and **AGC** is, therefore, **-1**

3

## Complexity and Storage Requirements

- The amount of computations required:
  - 3 summations and a max-operation per matrix entry
  - total of  $m \times n$  entries need to be computed
- Therefore, the complexity is  $O(mn)$
- So is the storage requirement
  - need to store matrix entries to be able to backtrack
- If only the score is needed, can do with less storage
  - compute  $F$ , say, column-by-column -- need to store only the last two columns
  - however, cannot backtrack and find the optimal alignment itself
- There are techniques which trade complexity vs. storage
  - there are also efficient heuristic techniques -- more on these later

4

## Today

---

- Local pairwise alignment
- Ends-free (overlap) alignment
- Affine gap score function
- Choice of the scoring rules
  - a probabilistic model
- Pairwise alignment in practice
- Multiple sequence alignment

5

## Local Alignment

---

- Next problem: optimal local alignment
  - look for similar substrings of  $s$  and  $t$
  - formally: given  $s[1\dots m]$ ,  $t[1\dots n]$ , find  $i, j, k, l$  such that the score of aligning  $s[i\dots j]$  and  $t[k\dots l]$  is maximal
- The solution: Smith-Waterman algorithm
  - the idea: the same procedure as before but allow fresh starts
  - in other words: allow a new alignment instead of extending an old one
- More specifically, the scoring matrix is now:

$$F(i, j) = \max \begin{cases} 0 \\ F(i-1, j-1) + c(s_i, t_j) \\ F(i-1, j) + c(s_i, -) \\ F(i, j-1) + c(-, t_j) \end{cases}$$

$$F(0, 0) = 0, \quad F(i, 0) = \max(0, F(i-1, 0) + c(s[i], -)), \quad F(0, i) = \max(0, F(0, i-1) + c(-, t[i]))$$

6

## Local Alignment

---

- The scoring matrix:

$$F(i, j) = \max \begin{cases} 0 \\ F(i-1, j-1) + c(s_i, t_j) \\ F(i-1, j) + c(s_i, -) \\ F(i, j-1) + c(-, t_j) \end{cases}$$

- Taking the option 0 above corresponds to starting a new alignment
  - the reasoning: if the best alignment at some point has a negative value, it is better to start anew than to try to extend
  - fine print: expected score for a random match must be negative
    - this affects the choice of match/mismatch/indel scoring scheme
- Unlike the global alignment, we do not necessarily take the value in the bottom right corner as the best score
  - instead, start with the highest value in F and backtrack until hitting 0

7

## Smith-Waterman: Example

---

- Let us use the same scoring as before: match=+1, mismatch=-1, gap=-2.
- The sequences: **TAATA** and **TACTAA**

t \ s		T	A	C	T	A	A
0	0	0	0	0	0	0	0
T	1	0	1	0	0	1	0
A	2	0	0	2	0	0	2
A	3	0	0	1	1	0	1
T	4	0	0	0	0	2	0
A	5	0	0	1	0	0	3

8

## Smith-Waterman: Example

- To find the best local alignment, look for the largest entry in the matrix
- in this example, the solution is not unique

		t							
		T	A	C	T	A	A		
s	0	0	0	0	0	0	0	0	
	T	1	0	1	0	0	1	0	0
	A	2	0	0	2	0	0	2	1
	A	3	0	0	1	1	0	1	3
	T	4	0	0	0	0	2	0	1
	A	5	0	0	1	0	0	3	1

TACTAA  
TAATA

TACTAA  
TAATA

9

## Roadmap



- Local pairwise alignment
- Ends-free (overlap) alignment
- Affine gap score function
- Choice of the scoring rules
  - a probabilistic model
- Pairwise alignment in practice
- Multiple sequence alignment

10

## Overlap Alignment

---

Consider the following problem:

- Find the most significant **overlap** between two sequences.
- Possible overlap relations:
  - a. 
  - b. 

Different from **local** alignment in that we require alignment between the **endpoints** of the two sequences.

Formally: given  $s[1..n]$ ,  $t[1..m]$  find  $i, j$  such that

$$d = \max\{D(s[1..i], t[j..m]), D(s[i..n], t[1..j]), D(s[1..n], t[i..j]), D(s[i..j], t[1..m])\}$$

is maximal.



Solution: Same as Global alignment except no penalizing overhanging ends.

11

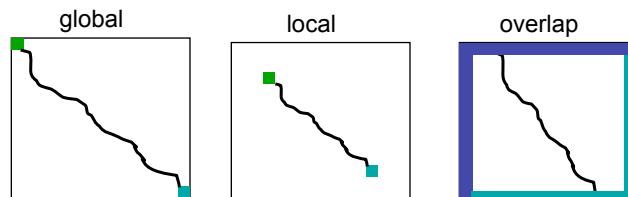
## Overlap Alignment

---

- Initialization:  $F(0,0) = 0$ ,  $F(i,0) = 0$ ,  $F(0,j) = 0$ , for all  $i, j$ .
- Recursion: as in the global alignment

$$F(i, j) = \max \left\{ \begin{array}{l} F(i-1, j-1) + c(s[i], t[j]) \\ F(i-1, j) + c(s[i], -) \\ F(i, j-1) + c(-, t[j]) \end{array} \right\}$$

- Score: the maximum value at the bottom or the rightmost frame
  - the best alignment starts from the maximum score cell and ends when it hits the top or the leftmost frame



12

## Illustration of Overlap Alignment

	H	E	A	G	A	W	G	H	E	E
<i>s</i> = PAWHEAE	0	0	0	0	0	0	0	0	0	0
<i>t</i> = HEAGAWGHEE	0	0	0	0	0	0	0	0	0	0
<b>P</b>	0									
<b>A</b>	0									
<b>W</b>	0									
<b>H</b>	0									
<b>E</b>	0									
<b>A</b>	0									
<b>E</b>	0									

Scoring scheme :  
 Match: +4  
 Mismatch: -1  
 Indel: -5



Still not addressing the issue of score choice... assume these are given

## Illustration of Overlap Alignment

	H	E	A	G	A	W	G	H	E	E
<i>s</i> = PAWHEAE	0	0	0	0	0	0	0	0	0	0
<i>t</i> = HEAGAWGHEE	0	0	0	0	0	0	0	0	0	0
<b>P</b>	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
<b>A</b>	0	-1								
<b>W</b>	0	-1								
<b>H</b>	0	4								
<b>E</b>	0	-1								
<b>A</b>	0	-1								
<b>E</b>	0	-1								

Scoring scheme :  
 Match: +4  
 Mismatch: -1  
 Indel: -5

## Illustration of Overlap Alignment

$s =$ PAWHEAE		H	E	A	G	A	W	G	H	E	E
$t =$ HEAGAWGHEE		0	0	0	0	0	0	0	0	0	0
<b>P</b>		0	-1	-1	-1	-1	-1	-1	-1	-1	-1
<b>A</b>		0	-1	-2	3	-2	3	-2	-2	-2	-2
<b>W</b>		0	-1	-2	-2	2	-2	7	2	-3	-3
<b>H</b>		0	4	-1	-3	-3	1	2	6	6	1
<b>E</b>		0	-1	8	3	-2	-3	0	1	5	10
<b>A</b>		0	-1	3	12	7	2	-2	-1	0	5
<b>E</b>		0	-1	3	7	11	6	1	-3	-2	4

15

## Illustration of Overlap Alignment

The best overlap is:

```
PAWHEAE-----
---HEAGAWGHEE
```

A different scoring scheme could yield a different result. For instance,

<p>Scoring scheme :</p> <p>Match: +4</p> <p>Mismatch: -1</p> <p>Indel: -5</p>		<p>Scoring scheme :</p> <p>Match: +4</p> <p>Mismatch: -1</p> <p>Indel: -2</p>
---	--	---

<pre style="font-family: monospace;">PAWHEAE----- ---HEAGAWGHEE</pre>		<pre style="font-family: monospace;">---PAW-HEAE HEAGAWGHEE-</pre>
---	--	--

16



## Roadmap

---

- Local pairwise alignment
- Ends-free (overlap) alignment
- Affine gap score function
- Choice of the scoring rules
  - a probabilistic model
- Pairwise alignment in practice
- Multiple sequence alignment

17

## Alignment With Affine Gap Scores

---

- **Observation:** insertions and deletions often occur in blocks longer than a single nucleotide
  - stated in probabilistic terms:

$$P(\text{gap of length } m) > P(\text{gap of length } 1)^m$$

- The scoring we considered so far assumes constant penalty per gap unit
  - need to modify constant-penalty scheme to account for the above phenomenon
- The most general solution: introduce a general (nonlinear) cost function  $\gamma(g)$ 
  - for instance, the global alignment recursion rule would be:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + c(s_i, t_j) \\ F(i-1, j) + \gamma(i-k), & k = 0, \dots, i-1, \\ F(i, j-1) + \gamma(j-k), & k = 0, \dots, j-1. \end{cases}$$

- Downside: the look into the past is costly and increases the complexity from  $O(n^2)$  to  $O(n^3)$  (assuming both sequences are of the length  $n$ )

18

## Alignment With Affine Gap Scores

- Fortunately, affine gap function enables complexity reduction back to  $O(n^2)$ 
  - we will, however, need more storage space –  $F(i,j)$  will not be enough
- Penalty score for a gap of length  $g$  :

$$\gamma(g) = -d - (g-1)e$$

- $d$  - penalty for introduction of a gap
- $e$  - penalty for elongating the gap by one unit

Typically  $d > e$

- A problem: when aligning the  $i$ -th letter in sequence to a gap, we do not know how much to penalize

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + c(s[i], t[j]) \\ F(i-1, j) + c(s[i], -) \\ F(i, j-1) + c(-, t[j]) \end{cases} \quad \left. \begin{array}{l} \leftarrow \\ \leftarrow \end{array} \right\} d \text{ or } e ?$$

- Need to distinguish between opening and extending a gap

19

## Alignment With Affine Gap Scores

- An example: consider three separate situations,

<b>IGAx<sub>i</sub></b>	<b>AIGAx<sub>i</sub></b>	<b>GAx<sub>i</sub>--</b>
<b>LGVy<sub>j</sub></b>	<b>GVy<sub>j</sub>--</b>	<b>SLGVy<sub>j</sub></b>

- We will have to compute three matrices simultaneously
  - $M(i,j)$  - the score obtained by aligning  $\mathbf{x}_i$  to  $\mathbf{y}_j$
  - $I_x(i,j)$  - the score obtained by aligning  $\mathbf{x}_i$  to a gap
  - $I_y(i,j)$  - the score obtained by aligning  $\mathbf{y}_j$  to a gap

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + c(x_i, y_j) \\ I_x(i-1, j-1) + c(x_i, y_j) \\ I_y(i-1, j-1) + c(x_i, y_j) \end{cases}$$

$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d \\ I_x(i-1, j) - e \end{cases}$$

$$I_y(i, j) = \max \begin{cases} M(i, j-1) - d \\ I_y(i, j-1) - e \end{cases}$$

20

## Alignment With Affine Gap Scores

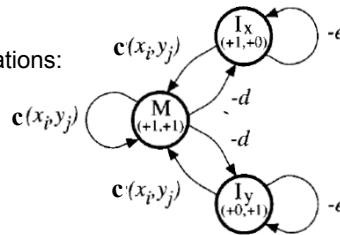
- **Initialization:** depending on the problem (global, local, overlap)
- **Recursion:** uses already known values -  $M(i', j')$ ,  $I_x(i', j')$ ,  $I_y(i', j')$

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + c(x_i, y_j) & (i \leq i, j \leq j, i+j < i+j) \\ I_x(i-1, j-1) + c(x_i, y_j) \\ I_y(i-1, j-1) + c(x_i, y_j) \end{cases}$$

$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d \\ I_x(i-1, j) - e \end{cases} \quad I_y(i, j) = \max \begin{cases} M(i, j-1) - d \\ I_y(i, j-1) - e \end{cases}$$

We assume that a deletion will not be followed directly by an insertion.  
This can be obtained by using  $score(mismatch) > -2e$

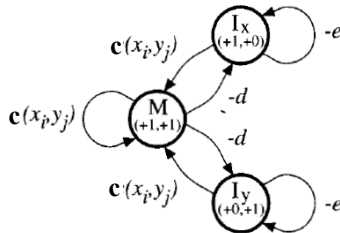
A diagram of the relations:



21

## Alignment With Affine Gap Scores

- A diagram:



- Two of the equations can be collapsed:

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + c(x_i, y_j) \\ I(i-1, j-1) + c(x_i, y_j) \end{cases}$$

$$I(i, j) = \max \begin{cases} M(i, j-1) - d \\ I(i, j-1) - e \\ M(i-1, j) - d \\ I(i-1, j) - e \end{cases}$$

22

## Roadmap

---

- Local pairwise alignment
- Ends-free (overlap) alignment
- Affine gap score function
- Choice of the scoring rules
  - a probabilistic model
- Pairwise alignment in practice
- Multiple sequence alignment

23

## How to choose scoring rules / interpret scores

---

- Recall our definition of the scoring operation:
$$c : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathcal{R}$$
  - $c(x,y)$ : the score of replacing  $x$  by  $y$
  - $c(x,-)$ : the score of deleting  $x$
  - $c(-,x)$ : the score of inserting  $x$
- How do we choose a meaningful score? How to interpret a score?
- Use a simple probabilistic model which describes similarity of sequences
  - similarity is probabilistic in nature: mutations, selection, etc. are random
- Essential question: how likely is that two sequences are similar
  - incorporate any side information: for instance, we may know something about the mutations on a specific area of a chromosome
  - distinguish between “similarity for a reason” and “random similarity”

24

## A Probabilistic Model

- Given a pair of aligned sequences, we would like to assign a score to the alignment
  - the score should be a measure of the relative likelihood that the sequences are related as opposed to being unrelated
  - assign probability to each of the two cases (related/unrelated), and find the ratio of the two probabilities
- For simplicity, assume no indels for the time being
  - only substitutions
- Also, assume that each position in a sequence is independent
  - can be made more general
- Let us consider two models:
  - **R**: the random model (sequences are unrelated)
  - **M**: the match model (sequences are related)

25

## A Probabilistic Model

- **R**: the random model (sequences are unrelated)
  - assume that each position in the two sequences is sampled independently from a distribution  $q(\cdot)$  over the alphabet
  - moreover, assume that we know the distribution  $q(\cdot)$  is known
  - then:
$$P(s, t|R) = \prod_i q(s_i)q(t_i)$$
- **M**: the match model (sequences are related)
  - assume that aligned pairs occur with joint probability  $p(s_i, t_i)$
  - we may think about the pair having evolved/derived from a common ancestor
  - then the probability for the whole alignment is

$$P(s, t|M) = \prod_i p(s_i, t_i)$$

26

## Test for Alignment

- The ratio of these two likelihoods is the odds ratio:

$$Q = \frac{P(s, t|M)}{P(s, t|R)} = \frac{\prod_i p(s_i, t_i)}{\prod_i q(s_i)q(t_i)} = \prod_i \frac{p(s_i, t_i)}{q(s_i)q(t_i)}$$

- $Q > 1$  implies that s and t are more likely to be related than unrelated
- $Q < 1$  implies that s and t are more likely to be unrelated than related
- The log-odds ratio:

$$S = \log Q = \sum_i \log \frac{p(s_i, t_i)}{q(s_i)q(t_i)}$$

- $S > 0$  implies that s and t are more likely to be related than unrelated
- $S < 0$  implies that s and t are more likely to be unrelated than related

27

## Probabilistic Interpretation of Scores

- The log-odds ratio:

$$S = \log Q = \sum_i \log \frac{p(s_i, t_i)}{q(s_i)q(t_i)}$$

- $S > 0$  implies that s and t are more likely to be related than unrelated
- $S < 0$  implies that s and t are more likely to be unrelated than related
- Define the scoring function as

$$c(s_i, t_i) = \log \frac{p(s_i, t_i)}{q(s_i)q(t_i)}$$

- With this definition,  $S = \sum_i c(s_i, t_i)$
- So, the score of an alignment is the log-likelihood ratio between the two models
  - scores often presented in a matrix, e.g., BLOSUM50 for amino acids

28

## Estimating Probabilities

- How do we come up with probabilities that describe the models R & M?
  - basic counting/estimation
- Assume we have a long sequence  $s$ , and want to estimate  $q(s_i)$ 
  - simply count the frequency of the letters from the alphabet
  - for instance, if a letter “a” occurs  $N_a$  times in the sequence of length  $n$ , then

$$q(a) = \frac{N_a}{n}$$

- Similarly, if we want to estimate  $p(s_i, t_i)$ , use an aligned pair of sequences  $s$  and  $t$  (“training” sequence alignment), and estimate the probability of pairs  $(a,b)$  as

$$p(a, b) = \frac{N_{a,b}}{n}$$

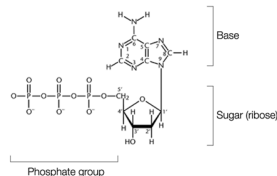
where  $N_{a,b}$  is the number times  $a$  is aligned with  $b$  in  $(s,t)$ .

- the choice of “training” sequence alignment depends upon application

29

## Estimating Probabilities

- The choice of “training” sequence alignment depends upon application
  - need pairs of sequences which are surely “related”
- In DNA/RNA, mutation is possible due to enzyme mistakes
  - can quantify it experimentally
  - note: it is known that some errors are more likely than others – e.g., the substitutions A/G or C/T (so-called transitions) are more likely than the other substitutions (so called transversions)



- In proteins, there are families that are known to be related
  - have a common ancestor

30

## Roadmap

---

- Local pairwise alignment
- Ends-free (overlap) alignment
- Affine gap score function
- Choice of the scoring rules
  - a probabilistic model
- Pairwise alignment in practice
- Multiple sequence alignment

31

## Alignment in Practice

---

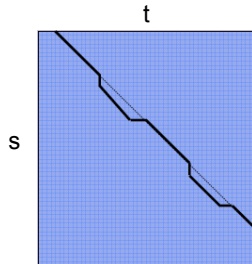
- Often we are asked to compare a newly determined sequence (mRNA, protein) with another one in a database
- This can be rather time consuming
- Example: compare a 1000-letters-long protein sequence with the protein database that has  $10^9$  entries
  - $10^{11}$  matrix elements need to be formed
  - at 10 Mflops, it would take 3 hours
  - 1000 sequences would require 4 months of running time
- What to do?
  - heuristics: computationally very efficient but may miss the true solution
  - broadly used in practice (FASTA, BLAST)
  - main observation: optimal alignments often contain long strings with no gaps in them – so, let's look for long matches with no gaps and attempt to extend them

32



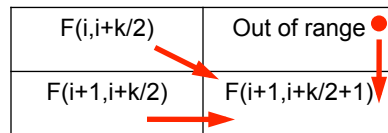
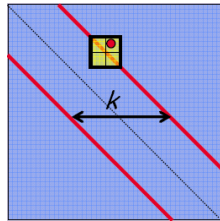
## Alignment in Practice

- Suppose that we have two strings of approximately the same length ( $m$  and  $n$ )



- If the optimal alignment of  $s$  and  $t$  has few gaps, then the path of an alignment will be close to the diagonal
- So, search inside a diagonal band of the matrix

- If the width of the band is  $k$ , the dynamic programming takes  $O(kn)$

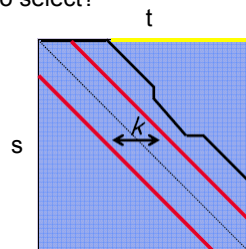


33

## Alignment in Practice

A few details to address:

- Where is the banded diagonal?
  - need not be the main diagonal
- Which subsequences to select?



- Heuristically find potential diagonals and evaluate them
- Heuristic sequence searching packages: FASTA, BLAST
  - FASTA: allows gaps in the alignment; BLAST: primarily ungapped

34

## FASTA

---

- Uses multistep approach to find local high scoring
  - start from exact short matches and then attempt to extend them

Formalizing the algorithm:

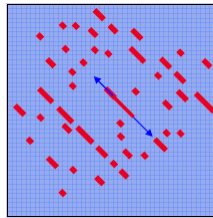
- Input: strings  $s$  and  $t$ , and a parameter “ktup”
  - The parameter “ktup” is the length of short exact matches that we will look for. It should be long enough to efficiently initiate the alignment, but not too long so that the matches are rare
- Output: A highly scored local alignment obtained via the following steps
  1. Find pairs of matching substrings (seeds)
$$s[i\dots i+ktup] = t[j\dots j+ktup]$$
  2. Extend to ungapped diagonals
  3. Extend to gapped matches using banded diagonal

35

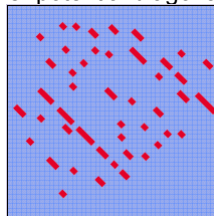
## FASTA

---

- Uses multistep approach to find local high scoring
  - start from exact short matches and then attempt to extend the



- The seeds are typically short (4-6 for DNA, 2 for proteins)
  - otherwise, we find fewer potential diagonals

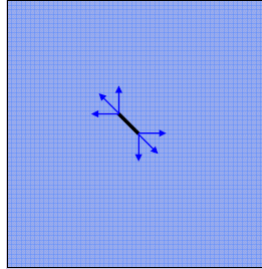


36

## BLAST

---

- BLAST: Basic Local Alignment Search Tool
- Similar idea to FASTA
  - search for short seeds (typically, length 11 for DNA, 3 for proteins)
  - attempt to extend them as far as possible in a greedy manner



- The search stops when the score passes below some lower bound, typically determined in a statistical manner