COSC 348: Computing for Bioinformatics

Lecture 5:
Sequence Alignment – Global Alignment

Lubica Benuskova, Ph.D.

http://www.cs.otago.ac.nz/cosc348/

Sequence Alignment

- **Sequence alignment** is a way of arranging two or more sequences of characters to *identify regions of similarity*
  - b/c similarities may be a consequence of functional or evolutionary relationships between these sequences.
- Another definition: Procedure for comparing two or more sequences by searching for a series of individual characters that are *in the same order* in those sequences
  - **Pair-wise alignment**: compare two sequences
  - **Multiple sequence alignment**: compare > 2 sequences

Similarity versus identity

- In the process of evolution, from one generation to the next, and from one species to the next, the amino acid sequences of an organism's proteins are gradually altered through the action of DNA mutations. For example, the sequence:
  - **ALEIRYLRD**
- could mutate into the sequence: **ALEINYLRD**
- in one generation and possibly into **AQEINYQRD**
- over a longer period of evolutionary time.
  - Note: a hydrophobic amino acid is more likely to stay hydrophobic than not, since replacing it with a hydrophilic residue could affect the folding and/or activity of the protein.

Sequence alignment: example

- Task: align **abcdef** with somehow similar **abdgf**
- Write second sequence below the first one

```
abcdef
abdgf
```

- Move sequences to give maximum match between them.
- Show characters that match using vertical bar.

Sequence alignment: example

```
abcdef
||
abdgf
```

- In order to *maximise the alignment*, we insert gap between **b** and **d** in lower sequence to allow **d** and **f** to align

```
abcdef
||  |  |
ab-dgf
```

- Note **e** and **g** don't match

Quantitative global alignments

- We are looking for an alignment, which
  - maximizes the number of base-to-base matches;
  - if necessary to achieve this goal, inserts gaps in either sequence (a gap means a base-to-nothing match);
  - the order of bases in each sequence must remain preserved and
  - gap-to-gap matches are not allowed.

- We need some scheme to evaluate the goodness of alignment

Scoring scheme

- The scoring scheme consists of character *substitution scores* (i.e. score for each possible character replacement) plus penalties for gaps.
- The *alignment score* is the sum of substitution scores and gap penalties. The alignment score reflects **goodness of alignment**.
- From this slide on, we use the ideas and examples from the lecture of Dr. Vladimir Likić given at the 7th Melbourne Bioinformatics Course.

Scoring scheme: example

- For DNA we can construct the following *substitution matrix*: '+1' as a reward for match, and '-1' as the penalty for mismatch, and ignore gaps:

	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

- Using this scoring scheme, let us evaluate the following alignments (penalty for a gap is = 0):

```
   ATGGCG  query sequence
1. ATG-AG  The score: +1+1+1+0-1+1 = 3 ☑
2. A-TGAG  The score: +1+0-1+1-1+1 = 1 ☒
```


Real scoring schemes

- For DNA (pyrimidines and purines are mutually OK):

	C	T	A	G
C	+2	+1	-1	-1
T	+1	+2	-1	-1
A	-1	-1	+2	+1
G	-1	-1	+1	+2

- Protein substitution matrices are significantly more complex than DNA scoring matrices.
  - PAM, i.e. "Point Accepted Mutation" family (PAM250, PAM120, etc)
  - BLOSUM, i.e. "BLOcks SUbstitution Matrix" family (BLOSUM62, BLOSUM50, etc.)

Scoring schemes: PAM and BLOSUM

- Substitution scores of both PAM and BLOSUM matrices are derived from the analysis of known alignments of evolutionary related proteins.
- Point Accepted Mutation (PAM), is the point mutation per 100 amino acids:
  - PAM1 means a 1 point mutation/100 amino acids.
  - Different PAM matrices are derived from the multiplication of the PAM1 matrix; as multiple substitutions can occur at the same site
- The BLOSUM matrices are newer and considered better.
  - E.g., BLOSUM62 is the matrix calculated by using the observed substitutions between proteins which have at most 62% sequence identity, etc.

BLOSUM62

	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

Gap penalties

- *Constant gap penalty.* Constant gap penalty means that any gap, whatever size it is, receives the constant negative penalty, $-g$.
  - The total number of gaps matters not their length.
  - *Minimizes the number of gaps*.
- *Linear gap penalty.* Linear gap penalty depends linearly on the size of a gap. Parameter, $-g$, is the penalty per unit length of a gap.
  - The overall penalty for one large gap is the same as for many small gaps.

Gap penalties – continuation

- *Affine gap penalty*. In biological sequences, it is more likely that a one big gap of length 10 occurs in a sequence, than 10 small gaps of length 1.
  - Therefore, affine gap penalties *favour longer gaps* over single gaps of the same total length.
  - They use a gap opening penalty, $o < 0$, and a gap extension penalty, $e < 0$, such that $|e| < |o|$, to encourage gap extension rather than gap introduction.
  - A gap of length L is then given a penalty $g = o + (L-1)e$.

Exhaustive alignment: brute force

- Having the scoring scheme we can proceed to generate and evaluate alignments:
- Brute force: Generate the list of all possible alignments between two sequences, score them and select the alignment with the best score.
- The number of possible global alignments between two sequences of length L is $2^{2L} / (\pi L)^{1/2}$. For two sequences of 250 bases this is $\sim 10^{149}$.
- Practically useless…

Needleman-Wunsch algorithm

- We have two 2D matrices: the *score matrix* and the *traceback matrix*.
- The Needleman-Wunsch algorithm consists of 3 steps:
  - Initialisation of the score and the traceback matrices
  - Calculation of scores and filling in the score and traceback matrices
  - Inferring the alignment from the traceback matrix
- In the example we align 2 sequences of amino acids SEND and AND with the BLOSUM62 substitution matrix and the constant gap penalty **g = –10.**

Step 1: initialisation

- During the initialisation, the 1st row and 1st columns of the score and traceback matrices, **Score** and **Tr**, respectively, are initialised in the following way:

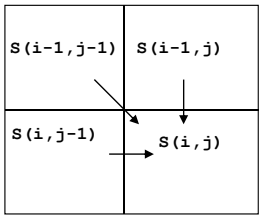
Score		S	E	N	D
	0	-10	-20	-30	-40
A	-10	?			
N	-20				
D	-30				

Tr		S	E	N	D
	done	left	left	left	left
A	up	?			
N	up				
D	up				

The next step is to determine S(2, 2) and T(2, 2)

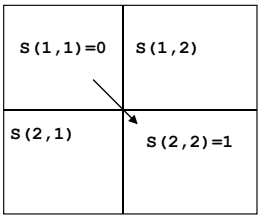
Step 2: calculation of scores

- The next step is to find the score value for an element S(2, 2).
- Rule: Value S(i, j) will become the **maximum** of:
  - diag = S(i – 1, j – 1) + r(i, j) // r is char replacement score
  - up = S(i – 1, j) + g
  - left = S(i, j – 1) + g



Step 2: calculation of scores

- The substitution score for letters i and j according to the BLOSUM62 is denoted as r(i, j) and g = –10 is the gap penalty.
- Thus for S(2, 2) we have:
  - diag = S(1, 1) + r(A, S) = 0 + 1 = 1 // this is maximum score
  - up = S(1, 2) + g = –20
  - left = S(2, 1) + g = –20



Step 2: filling in the score matrix

- We calculate the matrix S elements iteratively. Resulting matrix looks like this:

Score		S	E	N	D
	0	-10	-20	-30	-40
A	-10	1	-9	-19	-29
N	-20	-9	-1	-3	-13
D	-30	-19	-11	2	3

Step 2: filling in the traceback matrix

- The traceback items are indices of maximal scores. Both matrices look like this:

Score		S	E	N	D
	0	-10	-20	-30	-40
A	-10	1	-9	-19	-29
N	-20	-9	-1	-3	-13
D	-30	-19	-11	2	3

Tr		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Step 3: deducing the best alignment

- Traceback is the process of deduction of the best alignment from the traceback matrix.
- The traceback always begins with the last cell, i.e. the bottom rightmost cell in **Tr**.
- Sequences are aligned *backwards*, i.e. from right to left.

Resulting alignment:

1. D
   D
2. ND
   ND
3. END
   -ND
4. SEND
   A-ND

Step 3: deducing the best alignment

- There are 3 possible moves along the *traceback path*:
  - Diag: the letters from two sequences are aligned
  - Left: gap is introduced into the left sequence
  - Up: a gap is introduced into the top sequence

Tr		S	E	N	D
	done	left	left	left	left
A	up	diag	left	left	left
N	up	diag	diag	diag	left
D	up	up	diag	diag	diag

Resulting alignment:

1. D
   D
2. ND
   ND
3. END
   -ND
4. SEND
   A-ND

Evaluation of the alignment

- Let us evaluate, i.e.score, all possible alignments :

```
SEND
-AND   score = +1
A-ND   score = +3 ← the best
AN-D   score = -3
AND-   score = -8
```

- Thus, the global alignment found by the NW algorithm is indeed the best one as we have confirmed by evaluating all possible alignments in this small example, where we can afford an exhaustive search.

Conclusions

- The NW alignment is over the entire length of two sequences:
  - the traceback starts from the lower right corner of the traceback matrix, and completes in the upper left cell of this matrix.
- The Needleman-Wunsch algorithm works in the same way regardless of the length or complexity of sequences and ***guarantees*** to find the best alignment.
- The Needleman-Wunsch algorithm is appropriate for finding the best alignment of two sequences which are
  - (i) of similar length;
  - (ii) similar across their entire lengths.