1 Introduction

1.1 How are we going to solve a biological problem?

- Definition - defining the problem.
  What is the biological question?
  Sometimes the hardest step is to take a natural event and make it computable.
  For example, the first biological problem we’ll encounter in this course is sequence alignment - we should define a measure of imagination.

- Algorithm - how to solve it efficiently?
  In the example of sequence alignment, if we only wish to define imagination as percents of identity, the problem is quite easy, but if we take into account some amino acids which are quite similar to each other, we’ll have to use a more complex score function

- Learning
- Evaluation

1.2 Course Topics

- sequence analysis
- evolution
- structure
- network
- expression
2 Sequence Analysis

The biggest contribution of computational biology to biology is probably sequence alignment.

2.1 Defining the problem

- Input: two biological sequences (with a given alphabet)
- Objective: find the resemblance between the sequences

![Example DNA sequences](image1.png)

Figure 1: An example of two DNA sequences we would like to align

Are these sequences alike? We would like to align the given sequences with one another so that the resembling parts are one against the other (for example, the GCGC sequence which is marked bold in both sequences). One option to align the given sequences s and t is:

![Aligned sequences](image2.png)

Figure 2: Notice that "-" is a special letter which describes an indel, which stands for insertion/deletion, meaning a base was inserted or deleted in one of the sequences in comparison to the other.

A good alignment is one which has many similar letters one against the other.

We have three types of matches:
- perfect match (PM)
- mismatch (MM)
- indel (I/D)

This alignment is of course not the only one possible for these two sequences. The number of all possible alignments is exponential in the lengths of the sequences, since in each position there could be either a letter against a letter or two cases of a letter against a gap ("-" against "-" is not allowed). Thus we wouldn’t be interested in scanning all possible alignments.

Obviously, there are also poor alignments, for example:

![Poor alignment](image3.png)

Figure 3: Example of poor alignment, each sequence is aligned against gaps only

What is worse? An indel or a mismatch?
First, we should answer the question: what is "good"?
Clearly we want to get perfect matches.
We should give scoring alignment, for example:

\[
\begin{array}{l}
PM & +1 \\
MM & -1 \\
I/D & -2 \\
\end{array}
\]

Figure 4: In this case we would prefer MM over I/D, unless there is an indel which by using it will provide more perfect matches then otherwise.

We can complicate the problem, for example, by giving a MM of \(i\) against \(C\) a lower score, compared to \(A\) against \(T\).

Scoring alignment (for comparing DNA sequences) can be described as a function \(S\) from the alphabet \{A,C,G,T,-\} to the real numbers \(R\)

\[S : \{A,C,G,T,-\}^2 \rightarrow R\]

Figure 5. For amino acids the table is more complicated.

The score alignment is an additive function - the final score is the sum of all scores.
Is this function indeed a biological additive?
We assumed that there is no connection to the context. This is a simplifying assumption.
Consider methylation of Cytosine in the DNA. The Cytosine is methylated only if it is adjacent to a Guanine, in CG-islands. When Cytosine is methylated, there are greater chances that in the process of replication it will be copied as Thymine instead of as Cysteine. These areas which undergo methylation has more tendency to disappear in replication, so we might consider mismatch of \(T\) against \(C\) which appear after \(G\) in a different way then when the \(C\) was originally after another nucleotide.
Score alignment function based on pairs will be more complex both in the aspect of algorithm and learning. Therefore, we'll convert each biological problem into a simplified mathematical problem.
The question is what is the price we pay for this simplicity?
For example, the function used in Blast is additive except for indels.
There are cases in which Blast is missing, for example, Cystein is quite rare amino acid that has a structural function (disulfide bonds), therefore two Cysteins which are in a match has quite a high score, when sometimes all the other amino acids are not aligned well. Biological, this alignment might be worthless. This is one of Blast’s weaknesses.
2.2 Modeling

Giving two sequences s and t length n and m respectively, how will we find an alignment which gets the highest score? We’ll use dynamic programming.

Let’s define

\( s[1 \ldots i+1] \) the prefix length \( i+1 \) of \( s \).
\( t[1 \ldots j+1] \) the prefix length \( j+1 \) of \( t \).

To get the best alignment for these two sequences, we should get the best alignment for each sub-sequence.

![Alignment Diagram](image)

Figure 6: The prefix length \( i+1 \) of \( s \), \( x \) in the last position, and the prefix length \( j+1 \) of \( t \), \( y \) in the last position.

We have three options to align those two prefixes:

1. Align \( x \) against \( y \)
2. Align \( x \) against “-” (a gap in \( t \))
3. Align “-” against \( y \) (a gap in \( s \))

Let’s observe the score for each option:

Assume \( d() \) returns the best alignment for two sequences.

1. \( d(s[1 \ldots i], t[1 \ldots j]) + s(x, y) \)
2. \( d(s[1 \ldots i], t[1 \ldots j+1]) + s(x, -) \)
3. \( d(s[1 \ldots i+1], t[1 \ldots j]) + s(-, y) \)

All of the three options are easier to calculate than the original sequences, since at least one of the sequences is shorter.

The base case: the length of one of the sequences is 0; in that case we’ll align the second sequences against indels.

In general:

\[
d(s[1 \ldots i+1], t[1 \ldots j+1]) = \max \left\{ \begin{array}{l}
d(s[1 \ldots i], t[1 \ldots j]) + s(x, y) \\
d(s[1 \ldots i], t[1 \ldots j+1]) + s(x, -) \\
d(s[1 \ldots i+1], t[1 \ldots j]) + s(-, y) \\
\end{array} \right. \\
\]

\[
d(\varepsilon, t[1 \ldots j]) = \sum_{l=1}^{j} s(-, t[l]) \\
\]

\[
d(s[1 \ldots j], \varepsilon) = \sum_{l=1}^{i} s(s[l], -) \\
\]

Our database (matrix \( n \times m \)) will contain prefix’ lengths (0 - the base case).
Let’s look on the alignment of AGC against AAAC:

![Alignment Diagram](image)

Figure 7. In order to fill the value of the cell marked with o we should first fill the values of the three cells marked *

Let’s use the scoring alignment function we saw earlier:

<table>
<thead>
<tr>
<th></th>
<th>PM</th>
<th>MM</th>
<th>I/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+1</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The matrix will look like that:

![Scoring Matrix](image)

Figure 8. The scoring Matrix for the alignment of AGC against AAAC

The score for the alignment of the sequences will be in the cell[n,m] (in this case the score is -1).

In order to find the actual alignment we should remember in each step how we reached the score (for example, we reached -1 in cell[4,3] from the cell[3,2], but to cell[3,2] we had two options to reach for from cell[2,1] or from cell[2,2]).

This alignment is called a global alignment - one whole sequence against a second sequence.