SCIE2100 Practical 3, Week 6

Pairwise sequence alignment

Aim

In this practical you will write Python code for aligning two sequences using the dynamic programming algorithm described in lectures (and section 5.2 of the textbook), and then investigate the effects of different substitution matrices and gap penalties on alignments. You will also use the custom substitution matrix you constructed in the previous practical to see how well it works for sequences that it is tailor-made for, in comparison to general substitution matrices.

Files

The files you will need for this practical are: this guide, an updated *sequence.py* and three substitution matrix files: *blosum62.matrix*, *pam250.matrix* and *dna.matrix*. Also provided are *p450.matrix* and *rhodopsin.matrix*, generated as described in Practical 2.

The three matrix files contain substitution matrices that you will work with later in the practical. The Sequence class in *sequence.py* has had two incomplete functions added to it – `alignGlobal` and `alignLocal`.

The Alignment class

Previously you have read alignments from files, but this practical will ask you to construct one yourself from a pair of sequences, to return an `Alignment` object. Leaving the alignment algorithm aside for the moment, let's see how that might work. Note that an `Alignment` is constructed from a list (or set, or any other collection) of `Sequence` objects.

```
>>> from sequence import *
>>> s1 = Sequence('THISLINE', Protein_Alphabet)
>>> s2 = Sequence('ISALIGNED', Protein_Alphabet)
>>> aln = Alignment([s1, s2])
RuntimeError: Alignment invalid: different lengths
```

Ouch! Clearly, an alignment needs to line up perfectly, using the symbol '-'.

```
>>> s1 = Sequence('THISLINE-', Protein_Alphabet)
RuntimeError: Invalid symbol: - in sequence
```

A `Sequence` needs to be flagged as 'gappy' (i.e. to accept that additional gap symbol).

```
>>> s1 = Sequence('THISLINE-', Protein_Alphabet, gappy = True)
>>> aln = Alignment([s1, s2])
>>> print aln
 THISLINE-
 ISALIGNED
```

Global Pair-wise Sequence Alignment

Both pair-wise alignment functions take two `Sequence` objects (not aligned, not gappy), a substitution matrix (a `SubstMatrix` object, which you worked with in the previous practical) and a gap penalty as arguments. It aligns the two sequences using the Needleman-Wunsch algorithm (dynamic programming global alignment) and returns an `Alignment` object. Make sure you are familiar with each of these classes.

1. The function is missing some of the code for the 'traceback' section of the algorithm (see the comments for details). Using the algorithm description given in lectures (or section 5.2 of the textbook) as a guide, complete the code.

   Hint: Remember that the traceback algorithm starts at the bottom-right corner of the score matrix (the end of the two sequences), and follows the scores *back* in the order they were generated, adding symbols from the sequences and gaps where appropriate, until it reaches the top-left corner (the start of the sequences). Try to understand the code that is already there, see what steps are missing, and then fill those in.

   Submit (Q1) code with your additional comments. Don't submit the entire functions, just the traceback sections (i.e., the part you completed).

Local Pair-wise Sequence Alignment

The Smith-Waterman algorithm, which is very similar to the algorithm you've just finished implementing, performs a local alignment instead of a global one. The textbook outlines the main differences in the local alignment algorithm:

- There are no negative scores in the score matrix, and
- The traceback starts at the largest score, and stops when it hits a zero score.

2. The `alignLocal` function is missing the same code as the global alignment function. Once you have completed the global alignment function, fill in the same section of this function.

   Submit (Q2) code with your additional comments. Only submit the part you completed.

3. Can you relate the two differences in the global and local algorithms described above to the differences in the code in the two functions? Briefly explain the differences in the code, and how they implement the differences in the algorithms.

   Submit (Q3) your explanation to this conceptual difference with reference to the provided implementations.

Test Your Functions

Included at the bottom of *sequence.py* is some example code which runs the same examples as given in the algorithm description in the textbook (aligning "*THISLINE*" and "*ISALIGNED*" with the BLOSUM-62 substitution matrix; *blosum62.matrix*), so you can check that your functions are working correctly.

Global alignment with gap penalty -8 and -4 respectively should give you:

```
THISLINE-
ISALIGNED

THIS-LI-NE-
--ISALIGNED
```

For local alignment using the same gap penalties, you should see the following:

```
SLI-NE
ALIGNE

IS-LI-NE
ISALIGNE
```

4. Do the global and local alignments differ significantly above? If so, why? **Hint: Think about the difference in the algorithms you just implemented in the previous section** (We will revisit this question later.)

Aligning DNA Sequences

We will now align some DNA sequences to further test your functions, and to investigate the effects of choosing different gap penalties and substitution matrices on the alignment.

5. Use the `getSequence` function, just as you have done in previous practicals, to retrieve the DNA sequences identified by the following ID numbers: *HQ659871.1* and *JX416721.1*. (Beware that if you don't have external Internet access getSequence will fail, e.g. if you need to authenticate to get past UQ's block for guest access.)

```
>>> seqA = getSequence('HQ659871.1', 'genbank')
>>> seqB = getSequence('JX416721.1', 'genbank')
```

   The former is a region containing a gene from the Influenza A virus A/Brisbane/2015/2009 (H1N1) originating from a patient in Dalby in Queensland. The latter is a less well understood, but related sequence, also containing the same gene (nucleocapsid).

6. Align these two sequences, both globally and locally. For now, use a gap penalty of -4.

   Note: As these are DNA sequences and not amino acid sequences, you will need to use a DNA substitution matrix. The *dna.matrix* file contains such a matrix. Load the DNA substitution matrix. Also have a look at it using a text editor.

```
>>> sdna = readSubstMatrix('dna.matrix', DNA_Alphabet)
```

```
>>> aln = alignGlobal(seqA, seqB, sdna, -4)
```

It may take a minute or two to compute the alignment. Save your alignment to an HTML file so that you can look at it (using your web browser). We will ask a few questions below about it.

```
>>> aln.writeHTML('h1n1.html')
```

The alignment algorithms we've seen so far work by (in theory) evaluating each possible alignment of a pair of symbols in the sequences, and figuring out whether matching those symbols or placing a gap in a sequence gives a better score. These scores depend on the gap penalty and substitution matrix used by the alignment.

For example, if the gap penalty is very negative, more so than any score in the substitution matrix, then it will never be good score-wise to place gaps. As a result, the sequences will be lined up with no gaps, regardless of how bad the result is. Conversely, a too-small gap penalty may lead to an alignment where there is little or no identity overlap between the sequences.

Good alignments are achieved by choosing a good combination of gap penalty and substitution matrix, which may be different for different sequences and situations, and can be affected by biological knowledge of the sequences.

7. You know that the two DNA sequences above contain protein-coding regions. Play with the values in the DNA substitution matrix, and with the gap penalty, and see the effects the two have on the generated alignments. Use the sequences you downloaded in the previous exercise. Try to find a combination of matrix values and gap penalty to obtain the best alignment you can.

   Hint: Edit scores using your text editor (e.g. TextWrangler).

   Submit (Q7) the answers to the following questions:
   a. Explain why it took so long for the alignment to be computed. In your answer, consider how many cells would the S matrix contain when the scores are worked out.
   b. If you leave the DNA substitution matrix untouched, what is a biologically sensible gap penalty? Explore different settings to determine your answer, and define a criterion for evaluating it.
   c. Where does the protein start (in each gene)? Hint: proteins start with Met, encoded by the 'ATG' codon. Do the start codons align? Are there alternative start codons? What influence would alternative start codons have on the final protein? After having answered these questions, you may want to re-evaluate which alignment that is biologically sensible. (For those who did prac 0, you can refer back to the code you developed for translation. Others can use *translate.py*.)

Custom Substitution Matrices

In the previous practical you generated a substitution matrix from a trusted alignment of many sequences from a specific family of proteins. We provide *p450.matrix* and *rhodopsin.matrix*, built using the methods from the previous practical. This matrix should therefore be especially good at aligning sequences from that family.

8. You need to download, as you did above, two sequences from the protein family from which you created your custom substitution matrix.

   If your matrix is for the P450 protein family, download: `P08684` and `Q16802`

   If your matrix is for the Rhodopsin family, download: `P07550` and `P18089`

   Save the pair you downloaded to a FASTA file, so that you only have to download them once.

These pairs of sequences are part of the families you used to generate the matrices in the previous practical, but these sequences were not included in those alignments. The sequences are not very similar to each other, so a custom matrix should give a better alignment than a more general matrix.

9. Load your own, custom substitution matrix, just like you did above with the supplied matrices.

   Using your global pair-wise alignment function, align the two sequences in the file you saved in the previous exercise, using your custom matrix.

   View your alignment.

Note that you may have to play with the gap penalty to get a good alignment.

10. Now repeat exercise nine, but this time use the supplied BLOSUM-62 substitution matrix. Remember to play with the gap penalty to get a good alignment.

    View your alignment.

    Submit (Q10) your answers to these questions:
    a. Provide the picture of the alignments generated using the custom and BLOSUM-62 matrices. Do the two alignments differ greatly?
    b. Which one do you think is the better alignment?
    c. Explain why you think this either does or doesn't meet your expectations?
    d. How big a role do you think the gap penalty selection played?

    Hint: Think about the inherent differences between the custom substitution matrices and the BLOSUM-62 matrices.

Using specialised matrices for certain tasks is all well and good when they are being used on the right sequences. However, using a matrix that has been constructed from an alignment of a small subset of proteins for aligning sequence outside that subset is not likely to work very well. The BLOSUM and PAM matrices are calculated from huge data corpuses, and therefore (hopefully) avoid biases of special properties associated with certain proteins or protein families. The matrix you constructed in the

previous practical is unlikely to perform well for general sequence alignment (you can try this out to see for yourself, if you wish).

Assessment

Answer the following questions in a PDF or Word document and submit it to Blackboard by the due date.

Provide your responses to the items marked with **Submit** above.