Sequence Alignment

Project on DNA and protein sequence local and global alignment.
This is my take on the Needleman-Wunsch algorithm which is a global alignment technique using dynamic programming. The algorithm is extended for realizing local alignment following the Smith and Waterman approach.
The optimization algorithm essentially divides a large problem (the full sequence) into a series of smaller problems. It then uses the solutions to the smaller problems to find an optimal solution to the larger problem. The algorithm assigns a score to every possible alignment and outputs the alignment for the optimal score.

Install

A minimum python version 3 is required to run the program. Install all required python modules as follows:

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

Usage

1. open a terminal
2. direct yourself into the folder that must contain alignment.py, 2 fasta-formatted files, BLOSUM62.csv and requirements.txt (cd /my/project/directory)
3. To run the script use:

`python alignment.py [args]`

Type `python alignment.py -h` to obtain the output below for possible arguments:

```
usage: alignment.py [-h] [-s SUBSTITUTION_MATRIX] [-g GAP_SCORE] [-e EXTENSION_SCORE] [-p] [-l] fasta1 fasta2

positional arguments:
  fasta1                The </path/to/> name of the first fasta file of which the first sequence will be alig
  fasta2                The </path/to/> name of the first fasta file of which the first sequence will be alig

optional arguments:
  -h, --help            show this help message and exit
  -s SUBSTITUTION_MATRIX, --substitution_matrix SUBSTITUTION_MATRIX
                        csv file <path/to/> name containing the substitution matrix. Note that both a half co
                        is Blosum62 matrix. For nucleotide sequences find default matrix in README under Argu
  -g GAP_SCORE, --gap_score GAP_SCORE
                        Score for a gap. Default score is -10
  -e EXTENSION_SCORE, --extension_score EXTENSION_SCORE
                        Score for a gap extension, for a gap directly following after another gap. Default sc
  -p, --protein_alignment
                        2 protein sequences will be aligned. Take into account option -s
  -l, --local_alignment
                        local alignment technique using Smith and Waterman approach
  -sm, --show_matrices  flag to plot trace-back and score matrix
```

Arguments explained

`fasta1` and `fasta2`
example: `</path/to/>myseq.fasta </path/to/>myotherseq.fasta`

The names (with relative paths) of 2 files in fasta format. The first sequence contained in each file will be aligned. For global alignment using the default substitution matrix the sequence can contain A, a, C, c, G, g, T, t, U and u. In general, only the nucleotides contained in the substitution matrix can be used in the sequences.

`-s <string>` example: `-s </path/to/>my_matrix.csv`

The substitution matrix defines the score for matches and mismatches that is used throughout the alignment. With this argument you can input your proper substitution matrix. The input needs to be the relative path to your csv file containing the matrix. The csv file must be structured the same way as the default matrix shown below. All letters have to be capital and the matrix can be half filled like the one below or completely filled like the one in `BLOSUM62.csv`.
The default matrix that is used without the -s option is:

```
,A,C,G,T
A,2,,,
C,-1,2,,
G,1,-1,2,
T,-1,1,-1,2
```

Here a match costs 2 and a mismatch -1 except for GA and TC for which the cost is 1.
In case of option `-l` where 2 protein sequences are being aligned the default matrix is the Blosum62 matrix (see `BLOSUM62.csv`).

`-g <int>` example: `-g -6`

The score for a gap opening. Penalizes a insertion mutation or deletion mutation that follows a match or mismatch. In general it occurs when an extra nucleotide is added to or deleted from the DNA strand during replication. By default the penalty is -10.

`-e <int>` example: `-e -0.5`

The score for a gap extension. This is the case when a deletion/ insertion directly follows after another deletion or insertion. By default the penalty is -1. It should be less penalized than a gap opening.

`-p`

Flag indicating that 2 protein sequences are aligned. This does not influence the way of the functioning of the algorithm but the substitution matrix that is used (see argument -s)

`-l`

Flag indicating the local alignment technique. Which means that only parts of the sequences will be aligned if it results in a better score. In this case the waterman and smith approach is applied.
If this flag is not set the whole sequences will be aligned.

`-sm`

The score and trace-back matrix will be plotted in 2 separate windows simultaneously.

Examples for running the program

`python3 ./alignment.py seq1.fasta seq2.fasta`

Finds the optimal solution for seq1 and seq2 using the default scores -10 and -1 as well as the default substitution matrix.

`python3 ./alignment.py seq1.fasta seq2.fasta -g -15 -e -2`

Finds the optimal solution for seq1 and seq2 using a gap opening penalty of -15 and a gap extension penalty of -2 with the default substitution matrix.

`python3 ./alignment.py seq1.fasta seq2.fasta -s BLOSUM62.csv`

In this case the BLOSUM62.csv is specified.

`python3 ./alignment.py seq1.fasta seq2.fasta -p -sm`

It is indicated that we want to align 2 protein sequences. It is basically the same as using `-s BLOSUM62.csv` like in the example above. In addition, the flag to display score and trace-back matrices is set.

`python3 ./alignment.py seq1.fasta seq2.fasta -l`

Performing a local alignment using only default parameters.

`python3 ./alignment.py seq1.fasta seq2.fasta -g -50 -e -10 -l -p -s PAM250.csv -sm`

Example uses all arguments. It uses the local alignment technique for protein alignment with a different substitution matrix namely PAM250. As well as a gap opening score of -50 and a gap extension penalty of -10. The matrices will be displayed.

Results Output

```
The Alignment of Seq1 and Seq2
______________________________________________________

        SEND
        : ||
        A*ND
______________________________________________________

Length: 4
Number of matches: 2
Number of mismatches: 1
Number of gaps: 1
______________________________________________________

Total Score: 3
```

This is the result produced by:
`python3 ./alignment.py seq1.fasta seq2.fasta -p -sm`
The input sequences are:
`>Seq1 Send` and
`>Seq2 And`

In the first line both sequence identifiers are represented. The alignment of the two sequences is shown below that. In the case of a gap, a `*` is added to the respective sequence. There is no symbol between the two sequences. This is different with a match which can be recognized by the character `|`. A mismatch is represented by a `:`.

In the following section of the output a small statistic of the alignment can be seen. How long is the alignment, how many gaps, matches and mismatches are there in total? Finally, the optimal score is displayed.

Furthermore, both score and trace-back matrices will be plotted as follows:

How the program works

What does the *main* do

1. getting the users input

With the help of pythons argparse package we create an argument parser with all the above mentioned arguments. The user input is then stored in the variable `args`.

2. transforming the input

Firstly, identifiers and nucleotide sequences are extracted from fasta files with the function `fasta2nuc(filename)`. The `get_substitution_matrix(csv)` function reads the substitution matrix from a csv formatted file and saves it as a dictionary. The nucleotide sequences are transferred to upper cases. A `U` corresponds to a `T` if there is no specific score for it in the substitution matrix.

3. aligning the sequences

The function `align(...)` is called which does the alignment. Moreover, the score and trace-back matrices are stored in 2 variables.

4. reconstructing the optimal alignment

The optimal score is determined. In the case of a global alignment, this score is located in the bottom rightmost cell of the score matrix. For a local alignment, it corresponds to the maximum score of the matrix. If there are multiple occurrences of the maximum all will be taken into account.

From this position on the `traceback(...)` function reconstructs the path up to the beginning of the alignment with the help of the indicated directions in the trace-back matrix. This path is then stored in the variable `align_types`. The start position of the alignment is as well returned by the function.

5. printing the results

Finally, `show_results(...)` prints the results as explained in the results output section. In case of local alignment and if the optimal score occurs more than once, `show_results(...)` will be called for each optimal alignment. In case the flags for printing the matrices are set `show_matrix(...)` prints both matrices as 2 different matplotlib figures in 2 distinct windows, also shown in the results output section.

How to *align* 2 sequences

1. initialization

The sequences to align, the gap and gap extension score, the substitution matrix as well as the information if this is supposed to be a local or global alignment are provided in order to successfully align 2 sequences using the function `align(...)`.

Firstly, the score and trace-back matrices are initialized using `init_score_matrix(...)` and `init_trcback_matrix(...)`. The initialization is done such that the first row and column of the score matrix only contain gap or gap extension scores. Basically, this allows the alignment to start with gaps in one of the 2 sequences. If the gap score was -10 and the gap extension penalty -1 the initialized matrix would look like this:

		s	e	q	1
	0	-10	-11	-12	…
s	-10	0	0	0	…

		s	**e**	**q**	**1**
e	-11	0	0	0	…
q	-12	0	0	0	…
2	…	…	…	…	…

The trace-back matrix has the purpose of making it possible to determine where there are most likely gaps (insertions or deletions of nucleotides), mismatches (a nucleotide substitution) and matches (identical nucleotides). Since in the case of the first row, where sequence 1 is aligned with nothing and the first column, where sequence 2 is aligned with nothing there are only gaps it is only possible to move to the left or up. After initialization the trace-back matrix will be as follows:

		s	**e**	**q**	**1**
	done	left	left	left	…
s	up	0	0	0	…
e	up	0	0	0	…
q	up	0	0	0	…
2	…	…	…	…	…

If 2 sequences shall be locally aligned, it is possible that a sequence is cut off in the end or beginning in case this results in a better score. This means that both matrices need to be initialized such that when tracing back, the alignment could terminate at any position of the matrix. In order to allow this the trace-back matrix is initialized with only the string 'done' and the score matrix containing only zeros.

2. filling the matrices

The goal here is to fill both matrices such that at any position the score matrix contains the optimal score for aligning both sequences until this position as well as being able to track and trace the alignment starting at any position of the trace-back matrix.

In order to achieve this the algorithm loops over both matrices and fills them cell by cell with the help of a recurrence function. This function considers the previously calculated scores and looks for the optimal score for the current cell.

2.1 *recurrence*

In order to keep the function clearly arranged and readable the recurrence function is written as a inner function of `align(...)`. Keeping it inside the function allows the recurrence-function to not need any parameters, since the score values, the matrices and the current position (i, j) are known in `align(...)`.

First of all, the score for a substitution or match of the nucleotide at position i of sequence 1 and the nucleotide at j of sequence 2 is determined using the given substitution matrix. This score is than added to the score in cell (i-1, j-1), because it can be assumed that this cell already contains the optimal score for its position.

The scores for a gap on sequence 1 or 2 are calculated by adding the gap to the score of the upper cell or the cell on the left hand side. If the previously aligned position is a gap the gap extension score is used. Finally, the highest of these 3 scores is chosen. To put it in a nutshell, the recurrence-function computes the following:

```
             | D(i-1, j-1) + s(xi,xj)
D(i, j) = max| D(i-1, j) + g
             | D(i, j-1) + g
             | 0   (only for local alignment)
```

```
where D is the score matrix,
s is the score for xi and xj and
g is the score for gap extensions or openings
```

In case of a local alignment, ending the alignment at the current position is considered as a possible optimal solution, which means the score 0.

After having found the highest score the value for the trace-back matrix is being determined. If the highest score belongs to a match or substitution this would be the string 'diag', because both nucleotides can be aligned to each other. If it is a gap on seq1 the value is 'up' and for a gap on seq2 it is 'left'. In case of a local alignment if the highest score is 0 this means that the alignment ends at (i, j). The value would be 'done'. Both, optimal score and direction are returned by the recurrence-function.

3. returning the matrices

For more details on other functions please refer to the docstrings in the code.