Project 2

Alejandro Roca Arroyo and Emil Maag

September 18, 2018

Introduction

In this project we implemented and experimented with pairwise sequence comparison methods to compute optimal global alignments of two sequences, and optimal global aligment score, where the object was to minimize a cost. We wrote a program that can preform a pairwise global aligment using either a linear gap cost or a affine gap cost.
We choose to implement the algorithms using dynamic programming which means that the implemenation of the algorithm should be runinng in quadratic time and space. We investigated if this indeed was the case.

Methods

We defined classes for the data in the exercise with the methods we needed, in order to access them easily.

The following example shows how to use the program from the command line:

global_alignment.py seq1.fasta seq2.fasta score_matrix -alignment_type b [a] [-o]

*The brackets denotes optional arguments.

The followng list describes the arguments in more detail:

Arguments:

- seq1.fasta: fasta file containing sequence 1.
- seq2.fasta: fasta file containing sequence 2.
- score_matrix: file containing the score matrix used for the alignment. In a "Phylip-like" format.
- -alignment_type: type of alignment to be performed:

  for linear gap cost use -l or -linear

  for affine gap cost use -a or -affine
- b, a: parameters for gap cost function

  b -> constant gap cost or slope when performing linear/affine gap constant (extension penalty)

  a -> instersect for affine gap cost (opening gap penalty)
- -o: output alignment. if missing then outputs optimal score

The sequence fasta files (seq1.fasta and seq2.fasta) as well as the score_matrix file have to be located in the same folder as the program for excution.

Tests

Test cases for global alignment usign linear and affine gap cost

Case 1

```
Using linear gap cost g(k)=5*k, the expected score of an optimal alignment is -> 22
Obtained result:
22.0

And the optimal alignment should be one of these:

acgt-gtcaacgt-
acgtcgt-agc-ta

acgt-gtcaacgt
acgtcgt-agcta

Obtained alignment:
>seq1
ACGT-GTCAACGT
>seq2
ACGTCGT-AGCTA

Using affine gap cost g(k)=5+5*k, the expected score of an optimal alignment is -> 24
Obtained result:
24

and an optimal alignment is:

acgtgtcaacgt
acgtcgtagcta

Obtained alignment:
>seq1
ACGTGTCAACGT
>seq2
ACGTCGTAGCTA
```

Case 2

```
Using linear gap cost g(k)=5*k, the expected score of an optimal alignment is -> 14
Obtained result:
14.0

And the optimal alignment should be:

aataat
aa-gg-

Obtained alignment:
>seq1
AATAAT
```

```
>seq2
AA-GG-

Using affine gap cost g(k)=5+5*k, the expected score of an optimal alignment is -> 22
Obtained result:
22.0

And the optimal alignment should be one of these:

aataat
aagg--

aataat
aa--gg

aataat
a--agg

Obtained alignment:
>seq1
AATAAT
>seq2
A--AGG
```

Case 3

```
Using linear gap cost g(k)=5*k, the expected score of an optimal alignment is -> 20
Obtained result:
20.0

And the optimal alignment should be:

tccagaga
tc--gat-

tccagaga
t-c-gat-

tccagaga
tc--ga-t

tccagaga
t-c-ga-t


Obtained alignment:
>seq1
TCCAGAGA
>seq2
T-C-GA-T

Using affine gap cost g(k)=5+5*k, the expected score of an optimal alignment is -> 29
Obtained result:
```

```
29.0

And the optimal alignment should be one of these:

tccagaga
tc---gat

Obtained alignment:
>seq1
TCCAGAGA
>seq2
TC---GAT
```

Evaluation cases for global alignment usign linear and affine gap cost.

All questions are answered by showing the program used in the command line and the answers the program gave.

For question 3 and 4 a bash program were written to produce the desired matrices.

Question 1

Compute the score of an optimal alignment and an optimal alignment of seq1 and seq2 above using the programs global_linear using the above score matrix M and gap cost g(k)=5*k.

Answers:

```
./global_alignment.py sequences/seq1.fasta sequences/seq2.fasta score_matrix -l 5
```

```
226.0
```

```
./global_alignment.py sequences/seq1.fasta sequences/seq2.fasta score_matrix -l 5 -o
```

```
>seq1
TATGGA-GAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TCGCGAGATACT-CACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAA
>seq2
-ATGGATGTCAATCCGA-CTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCCATAAG-CACCACATTCCCTTATACTGGAGATCCT-CCA-TACAGCCATGG
```

optimal alignment:

seq1

TATGGA-GAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TC

GCGAGATACT-CACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAAAG

seq2

A-TGGATGTCAATCCGA-CTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCC

ATAAG-CACCACATTCCCTTATACTGGAGATCCT-CCA-TACAGCCATGGAA

Question 2

Compute the score of an optimal alignment and an optimal alignment of seq1 and seq2 above using the program global_affine using the above score matrix M and gap cost g(k)=5+5*k.

Answers:

```
./global_alignment.py sequences/seq1.fasta sequences/seq2.fasta score_matrix -a 5 5
```

```
266.0
```

optimal alignment score: 269.0

```
./global_alignment.py sequences/seq1.fasta sequences/seq2.fasta score_matrix -a 5 5 -o
```

```
>seq1
TATGGAGAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TCGCGAGATACTCACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAAAG
>seq2
-ATGGATGTCAATCCGACTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCCATAAGCACCACATTCCCTTATACTGGAGATCCTCCA--TACAGCCATGGAA
```

optimal alignment:

seq1

TATGGAGAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TCG

CGAGATACTCACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAAAG

seq2

ATGGATGTCAATCCGA-CTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCCA

TAAGCACCACATTCCCTTATACTGGAGATCCT-CCA-TACAGCCATGGAA

Question 3

Compute the optimal score of an optimal alignment for each pair of the 5 sequences above using global_linear with the score matrix M and gap cost g(k)=5*k. The result is a 5x5 table where entry (i,j) the optimal score of an alignment of seqi and seqj.

Answer:

```
./evaluate_global_alignment.sh 0 5
```

```
0 226.0 206.0 202.0 209.0
231.0 0 242.0 223.0 220.0
206.0 239.0 0 219.0 205.0
202.0 223.0 219.0 0 210.0
214.0 220.0 205.0 210.0 0
```

0 231.0 206.0 202.0 214.0

226.0 0 239.0 223.0 220.0

206.0 242.0 0 219.0 205.0

202.0 223.0 219.0 0 210.0

209.0 220.0 205.0 210.0 0

Question 4

Compute the optimal score of an optimal alignment for each pair of the 5 sequences above using global_affine with the score matrix M and gap cost g(k)=5+5*k. The result is a 5x5 table where entry (i,j) the optimal score of an alignment of seqi and seqj.

Answer:

```
./evaluate_global_alignment.sh 5 5
```

```
0 266.0 242.0 243.0 256.0
269.0 0 284.0 259.0 254.0
242.0 283.0 0 269.0 243.0
243.0 259.0 270.0 0 247.0
261.0 254.0 243.0 247.0 0
```

0 269.0 242.0 243.0 261.0

266.0 0 283.0 259.0 254.0

242.0 284.0 0 270.0 243.0

243.0 259.0 269.0 0 247.0

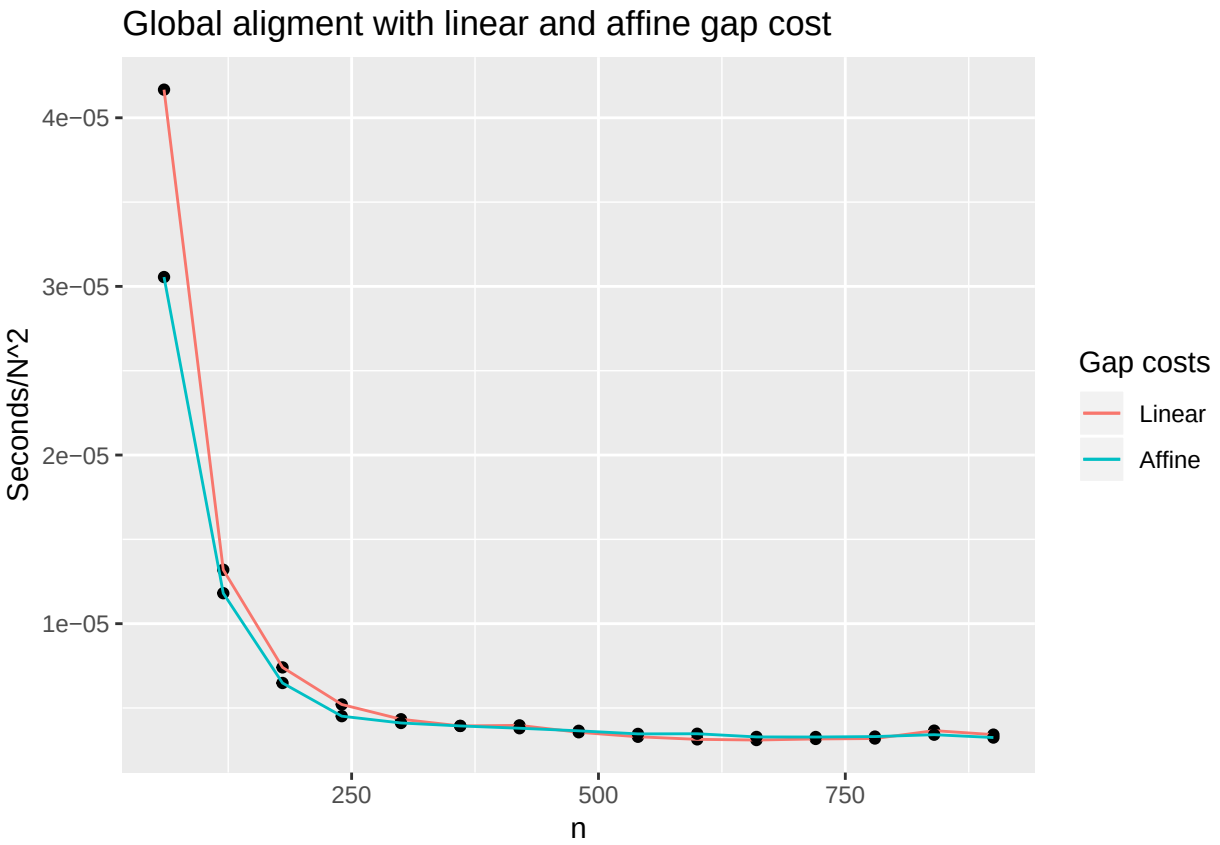
256.0 254.0 243.0 247.0 0

Experiments

As mentioned the algorithm should be running in quadratic time and space. This is firstly investigated for the algorithm doing pairwise global aligment with a linear gap cost (global_linear) and secondly for the algorithm doing pairwise global aligment with a affine gap cost (global_affine).

A bash script was written that measure the time consumption for the algorithm. This bash script takes a number that defines how many times two predefined sequences(sequence 1 and sequence 2) are duplicated. These sequences are passed to to a command line that runs the main algortihm with the two sequences, and measures the time consumption of the algorithm. The bash script were run iterating through the values 1 to 10, measuring the time consumption of the algorithm for increasing n. (1 to 10 duplications of the original sequences)

```
./measureTime_global_alignment.sh 0 5 15 2> Output/times_linearCost.txt
./measureTime_global_alignment.sh 5 5 15 2> Output/times_affineCost.txt
```



As seen from the plot the algorithm does run in quadratic time as expected. Global alignment with linear gap cost seems pretty semilar to global aligment with affine gapcost. As the affine gap cost uses more matrices it makes sense that this version of the algorithm seems to take a little more time then the linear gap cost version when n becomes larger. Much larger numbers of n might elucidate the difference better.