Project #2 - Genetic Algorithm Sequence Alignment

Biological Computation
Spring 2017
Due: March 30th

The goal of this experiment is to apply one or more of our Python sequence alignment algorithms (global, local, etc.) to data generated with our genetic algorithm (GA) to determine what information, if any, we can derive about the evolutionary process.

Ideally you will work in pairs with one team member generating the GA data and the other team member analyzing it. However, you should work together to design and plan the details of the experiment. Both team members should work on both parts of the code (the GA code and the sequence alignment code). One member should generate the GA data and the other member should analyze it. You only need to turn in one project write-up for both team members.

Because the GA we created in class uses a single population and there’s no speciation in the traditional sense the analysis will be fairly limited. For example, estimating relative evolutionary age based on alignment scores (but see below) - did a given individual/genome come from near the beginning or the end of the GA run? In this case one team member would run the GA and collect all of the generated genomes, reorder them (or a sub-set, such as one genome from each iteration), and then the other team member would use alignment scores to try to reconstruct the order of the genomes. If you have a more complex GA, for example with islands/sub-populations, more complex analysis is possible.

Because the GA uses fixed length “chromosomes” it makes sense to use the global alignment algorithm, but you may try (and turn in data for) the other algorithms.

## Encoding and Fitness

The base encoding in the genetic algorithm should either be nucleic acids (ATCG) or amino acids. From that point there are several options:

1. If your GA uses nucleic acids it should encode them, by codons, into amino acids so that the alignment algorithms we wrote apply. A variation of the decode.py code from the course schedule will do this.
2. Fitness can be based directly on the encoded strings. For example:
	1. The fitness could be the sum of points where A = 0 points, R = 1 points, N = 2 points, D = 3, …. Or the points per letter could come from another source, e.g. Scrabble. In this case the genomes could be expect to evolve to be nearly all copies of one amino acid – not very interesting, but sufficient for the analysis.
	2. The fitness could be based on the string itself, e.g. the closer the evolved string is to a palindrome the higher its fitness, the more times the word “cat” is repeated the higher the fitness, the more English words in the string the more points, etc.
	3. The fitness could be based on a very simple biochemical model. For example, strings with hydrophilic amino acids on the ends and hydrophobic amino acids in the middle get higher fitness, the more strongly hydrophobic and hydrophilic the amino acids the higher the fitness.
	4. The fitness could also include non-coding regions, regions that don’t contribute to fitness. In this case part of the analysis could be to try to identify these regions.
3. Alternatively, you could map the amino acids to digits (as discussed in class) and create a fitness function based on the values of the digits. A simple example would be to evolve the highest sum.

## Analysis

Lots of options are available, here are a few:

1. As discussed above, use one of the alignment algorithms to try to determine the relationship between (a subset of) the evolved genomess, relationship generally just meaning relative age – more similar genomes are presumably closer in age.
2. Do multiple runs with different mutation rates and/or selection pressure. By comparing the alignment scores between individuals from different generations try to identify which individuals are from which population. (E.g. individuals from successive generations in a high mutation rate population may have lower similarity scores.)
3. Have non-coding regions in the genome and try to identify them based on how rapidly different sections of the genome evolve (using, for example, local alignment scores).
4. If the fitness function is strongly multi-modal, with a few well defined fitness peaks, alignment scores for individuals in the final population could be used to try to determine if the population is “on” one or more than one of the peaks.

You will need to pick a scoring matrix, or make up your own based on what you know about the GA, mutation, fitness function, etc. for the alignments.

## Write-up

1-3 pages describing the GA used, the data collected, and the data analysis process. This can be directly taken from Sub-Project 2a (see below). Graphs and or tables showing the results of the analysis, including the “truth”, which should be known from running the GA. A conclusion addressing whether the experiment “worked” and what could be done to improve or extend it.

Sub-Project #2a - Genetic Algorithm and Sequence Alignment

Biological Computation
Spring 2017
Due: March 23rd

At this point the data must be generated for analysis. Turn in a description of the genetic algorithm and of the data. The genetic algorithm description should include the encoding, fitness function, population size, mutation rate, etc. The data description should include a description of the data itself (how many sequences are there, how long are they, etc.), the type of analysis that will be applied to it (global alignment, local alignment, etc.), and the questions being addressed (relative “age” of the sequences, regions under the strongest selection, etc.).