Introduction to Machine Learning

Evolutionary Algorithms Exercise

These exercises should be solved using Python notebooks (Jupyter) due to the ability to generate a report integrated with the code. It is assumed you are proficient with programming. All answers must be justified and the results discussed and compared to the appropriate baselines.
Each exercise is scored 1 point. Max score of the assignment is 4 points. Optional exercises help achieving the max score by complementing the errors or mistakes in the mandatory exercises.

Deadline*: November 1st, 2021*

Imagine a simplification of the mastermind game where we need to figure out a specific pattern. In our simplification we only have two colors and we can vary the pattern size. The pattern will be a bit sequence of a specific size.

1. Build the simulation environment:

    a) Create a function that produces a random bit pattern of a specific size (the bit pattern may be coded as a String of 0s and 1s)
    b) Create a function that will generate random patterns and measure how many attempts (and how much time) does it take to generate the "correct" bit pattern. Make two graphics on the evolution of attempts / time vs the number of bits in the pattern (2, 4, 8, 12, 16, ...). Record also the average run-times per pattern-size. Compare the difference in attempts and runtimes for each pattern-size. Each "point" in the graph should be a box-plot based on the results of 30 trials. Use a fixed set of seeds to be able to reproduce the experiments. Stop when run-times on your machine surpass 1h to gather the results for a given number of bits even if you have not reached 16 bits.
    c) Create an evaluation function that measures the proximity to a pattern, resulting in a single number that should be zero if the pattern is an exact match and growing as the difference between the attempted pattern and the "correct" pattern increase.
    d) Create a function inverse to the previous one that measures the "fitness" of the guessed pattern, this function should have a maximum value when the guessed pattern matches exactly the "correct" pattern and decrease as distance between patterns increases.

2. Create a function to mutate (flip one bit) a given pattern. Use this function in a cycle where the change is accepted only if it generates a better solution (i.e. a solution with higher *fitness*). Stop when you cannot find a better solution after 1000 mutations. Does it always converge to the best solution? If not, can you understand if there is one mutation of the last sequence that could result in a better pattern?

3. Generate a random set of patterns (a population, for example, 100 patterns) and evaluate each of the patterns. Select the best 30% and, based on the best 30%, generate (by mutation) a new set of 100, repeat the processes until the best evaluation stagnates. Compare fairly the search methods tested so far, including their run-times.

4. Generate part of the population (30%) by copying parts of the pattern from two different "parent" patterns (crossover), also chosen from the 30% best in the previous generation. Compare the results with the previous ones.

Optional

5. Explain the changes that would be necessary in the evaluation function, mutation and crossover to deal with a similar problem where the size of the target bit pattern would be unknown.

6. Explain the changes that would be necessary in the evaluation function, mutation and crossover to deal with a similar problem where the pattern would be of decimal numbers and not binary.

7. Explain the changes that would be necessary in the evaluation function, to apply this procedure to the robot-maze problem in the previous set of exercises.