INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Introduction to Machine Learning — 2025/2026
Evolutionary Algorithms

> These exercises must be solved using Python notebooks due to the ability to generate an integrated report with the code. It is assumed that the students are proficient in programming. All answers must be justified and the results discussed and compared with the appropriate baselines.
> Max score of the assignment is 2 points. Extra exercises (if existing) help achieving the max score by complementing errors or mistakes.
> **Deadline:** end of class in week November, 11th – 12th, 2025

Imagine a simplification of the *mastermind* game (https://en.wikipedia.org/wiki/Mastermind_(board_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 for each game. The pattern will be a bit sequence of a specific size. The type of feedback given will be detailed later in the exercise.

Exercise 1

1. Build the simulation environment. Develop the following functionalities:
    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 graphs on the evolution of attempts / time *vs* the number of bits in the pattern (2, 4, 8, 12, 16, …). 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 one hour to gather the results for a given number of bits even if you have not reached 16 bits. Compare the difference in attempts and run-times for each pattern size. In the next

questions use the biggest pattern that computes in a reasonable time.

c) Create a function 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 decreases as the distance between patterns increases.

d) Create a function to mutate (flip one bit) a given pattern. Use this function in a loop where the change is accepted only if it generates a better solution (i.e., a solution with higher fitness). Stop after 1000 mutations or when the pattern generated by mutation is equal to the solution. Does it always converge to the correct solution?

2. 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 them generate by mutation a new set of 100: 30% of the population corresponds to the 30% best and the remaining 70% are the ones generated by mutation. Repeat the processes until the best evaluation stagnates. Compare fairly the search methods tested so far, including their run-times. Allways use 30 tests for each result. Boxplots with whiskers are a good option to compare the results distribution for the several tests.

3. Design a crossover procedure for this type of pattern. Proceed in the same way as in the previous item.

Exercise 2

Change the process (the evaluation function, mutation, and crossover) to deal with a similar problem where **the size of the target bit pattern is unknown (but always between 2 and 32)**.

Exercise 3

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 digits and not binary**.

Recommendation: if you try to implement your ideas in order to better understand the necessary changes, include the code in the submission.