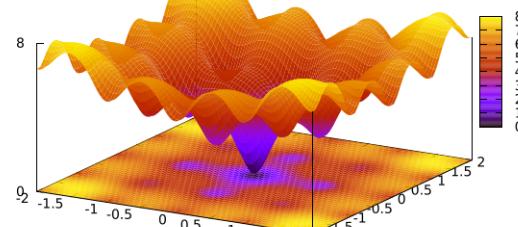
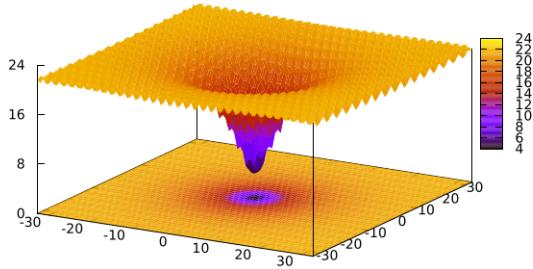


Task sheet 3 – Evolutionary Algorithms

Deadline : 19.11.2025

Objectives:

- ▶ implement a complete evolutionary algorithm and a simple ANN
- ▶ investigate effects concerning selection, mutation rate, population size etc.



1 Classical optimization with evolutionary algorithms

In evolutionary robotics the fitness of a robot controller is indirectly defined via the generated behavior and the fitness function. In standard optimization problems, however, the fitness is (almost) directly defined via the objective function. In the following, we implement an evolutionary algorithm to solve the ‘Ackley Problem’ in three dimensions. The task is to minimize this objective function :

$$f(x, y, z) = -20 \exp \left(-0.2 \sqrt{\frac{1}{3}(x^2 + y^2 + z^2)} \right) - \exp \left(\frac{1}{3}(\cos(2\pi x) + \cos(2\pi y) + \cos(2\pi z)) \right) + 20 + \exp(1),$$

on the interval $x, y, z \in [-32.768, 32.768]$. Two plots of this function for 2-d are shown above. In order to translate this problem into a maximization problem we define fitness as $1/(f(x, y, z) + 1)$.

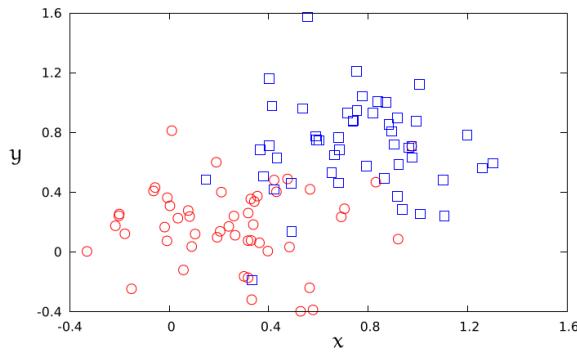
- a) Implement a complete evolutionary algorithm of your choice to solve this problem. Choose a genetic representation, a population size (> 1 , no hill climbers please), a selection method, a replacement strategy (generational replacement? elitism?), a mutation rate, and decide on whether you want to use recombination. Based on the above plots of the function you might be tempted to optimize your population initialization technique — please don’t do it here (no cheating) and initialize on the full interval $x, y, z \in [-32.768, 32.768]$.
- b) Debug your code. Check whether your selection works and make sure you are working on the right objective function.
- c) Log best fitness and average fitness for each generation and plot it. Think about what you are seeing there.
- d) Tweak your parameters. What works well? What parameters fail?

Optional:

- compare to a hill climber,
- check your algorithm's performance for higher dimensions of the Ackley Problem.

2 Optimal classification with evolutionary algorithms

Statistical classification is a classical problem of machine learning. It is an example for supervised learning (a set of already classified examples is provided) and hence in difference to typical situations in evolutionary robotics (classifying sensor input with appropriate actuator outputs is exactly what we do not want to do by hand). In the following, the task is to evolve a classifier for a simple problem with two-dimensional features. For this purpose, we evolve a small ANN without a hidden layer.



a) Download the training set (file ‘data’, see plot above). It has four columns: number of example, class it belongs to (either 0 or 1), feature x , and feature y . Write some code that imports the data.

b) Extend your code from task 1: implement a simple ANN that has two inputs, one output neuron, and a bias neuron (as on ANN slide 20) but no hidden layer. This ANN is fully described by just three weights. As activation function we use

$$\phi(x) = \frac{2}{1 + \exp(-2x)} - 1.$$

We say that the ANN classifies an input as ‘class 0’ if it outputs $\phi < 0$ and as ‘class 1’ if it outputs $\phi > 0$. To calculate the fitness, you have to successively evaluate an ANN’s output for all entries of the data set and account for how many have been classified correctly.

c) Evolve an ANN that solves the problem satisfyingly. Output the three weights of the best evolved ANN and plot the dividing line that is implemented by the ANN:

$$y = \frac{w_0}{w_2} - \frac{w_1}{w_2}x.$$

Optional: Download the training set file ‘data2’ and try to evolve an ANN that satisfyingly classifies it. However, you will need to add a hidden layer to your ANN. Furthermore, bias neurons seem beneficial, too.

For task 1 please turn in:

- ▶ please zip your submission in a single file named:
`evoRobo_sheet3_YOURLASTNAME1_YOURLASTNAME2.zip`
- ▶ a document with the full names of all group members, a list of the tasks and subtasks you have completed (with written answers if applicable) and/or a list of tasks/subtasks you have not completed
- ▶ accepted file formats for plots: png and jpg
- ▶ all your code (either in C/ C++ or python)
- ▶ a plot of best fitness and population average fitness over generations
- ▶ test at least one parameter of the evolutionary algorithm and provide evidence of what's a good choice for that parameter
- ▶ optional but very preferable: include a video where you go through and explain your code

For task 2 please turn in:

- ▶ all your code (either in C/ C++ or python)
- ▶ a plot of best fitness and population average fitness over generations
- ▶ the best three weights that you found
- ▶ a plot of the data and the separating line as defined by your weights
- ▶ optional but very preferable: include a video where you go through and explain your code