Evolutionary Robotics - Assignment 3

Amritanshu Amrit, Bhavesh Gandhi

November 17, 2025

1 Task 1: Classical optimization with evolutionary algorithms

1.1 Implementation

We implemented an evolutionary algorithm to solve the Ackley problem in three dimensions. The implementation can be found in the file `ackley-optimisation.py`.

1.2 Fitness Plot

The following plot (Figure 1) shows the best and average fitness over 1000 generations.

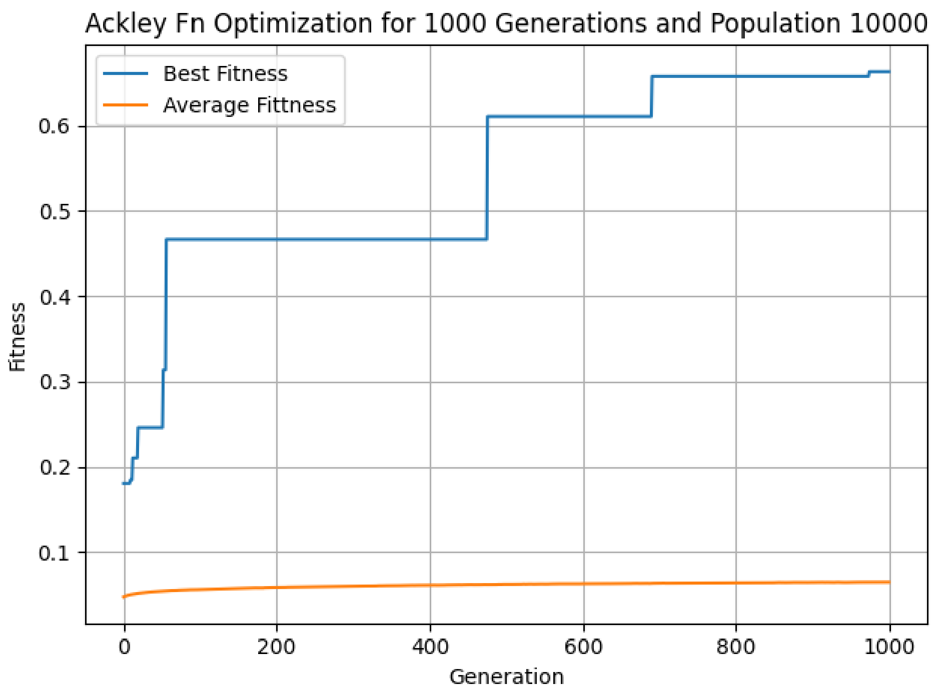

Figure 1: Best and average fitness for the Ackley function optimization.

- *Best Fitness:* The blue line shows that the algorithm is working as intended. It progressively finds better solutions, causing the fitness to jump up in steps. The fact that it plateaus around a fitness of 0.65 indicates the algorithm has likely converged to a good local optimum, but not the absolute best solution (the global optimum).
- *Average Fitness:* The orange line shows the average fitness of the entire population. It rises slightly and then flattens out at a low value. This, combined with the high best fitness, suggests that a small group of elite individuals is driving the progress, while the majority of the population is not converging towards the optimal solution.

1.3 Parameter Tuning

We tested various parameters for the evolutionary algorithm. The final parameters used are:

- Population size: 1000
- Number of generations: 10000
- Elitism: 10%
- Crossover: Single point
- Mutation rate: 20%

These parameters were chosen because they provided a good balance between exploration and exploitation, allowing the algorithm to consistently find a good solution. A lower population size or a very low mutation rate often resulted in the algorithm getting stuck. A higher mutation rate introduced too much randomness, preventing the algorithm from converging to the optimal solution.

Figure 2 shows an example with population 11 run over 1000 generations. It shows that the function plateaus prematurely and at a much lower fitness of 0.225

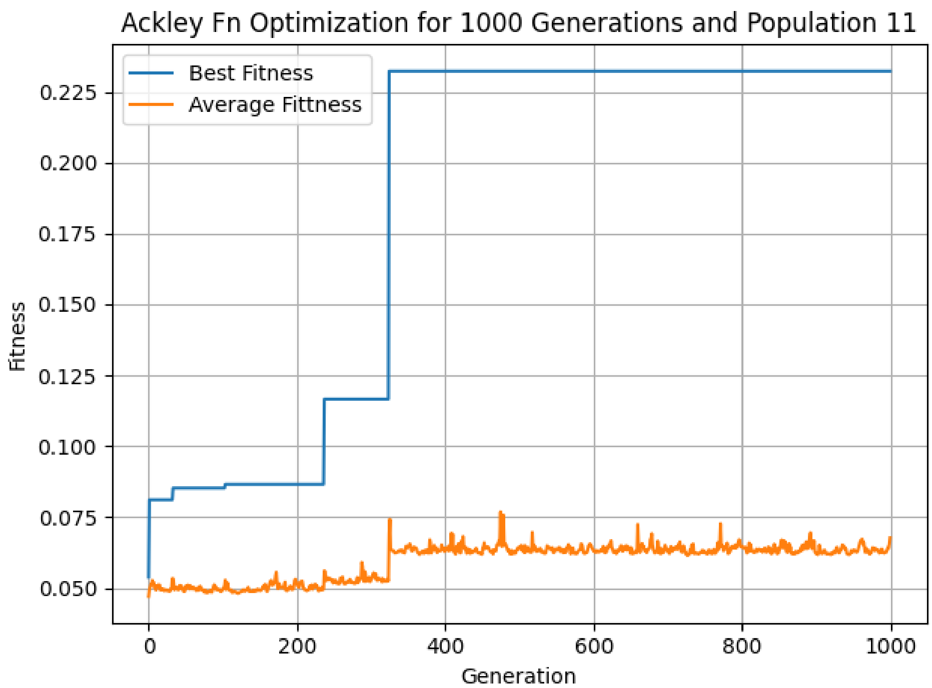

Figure 2: Best and average fitness for the Ackley function incomplete optimization.

2 Task 2: Optimal classification with evolutionary algorithms

2.1 Implementation

We implemented an evolutionary algorithm to evolve a simple ANN for a binary classification task. The implementation can be found in the file `ex2/ex2_ann_classifier.py`. The fitness evolution and results can also be seen in the text files `ex2/ex2_ann_classifier.txt`.

2.2 Fitness Plot

The following plot shows the best and average fitness over 100 generations.

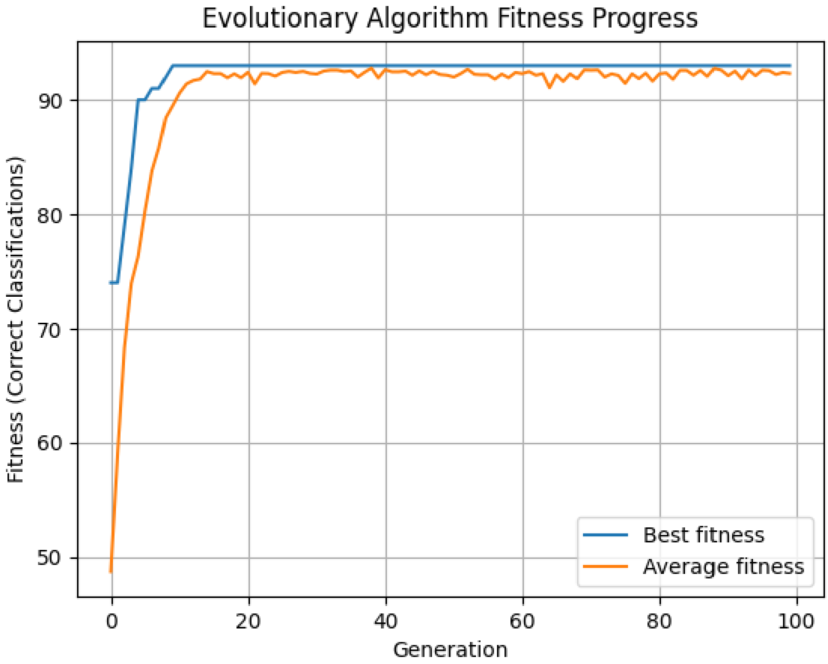

Figure 3: Best and average fitness for the ANN classifier.

2.3 Best Weights and Classifier Plot

The best weights found by the algorithm are:

- w0 (bias): -2.8835
- w1 (x-weight): 2.8466
- w2 (y-weight): 2.7764

The following plot shows the data points and the decision boundary of the evolved classifier.

2.4 Optional Task: Classification with a more complex dataset

2.4.1 Implementation

For the optional task, we used a more complex dataset that is not linearly separable. To handle this, we implemented a multi-layer ANN with one hidden layer containing two neurons. The evolutionary algorithm was tasked with finding the 9 weights for this network. The implementation can be found in the file `ex2/ex2_op_ann_classifier.py`. The evolution and final results can be seen the the file `ex2/ex2_op_ann_classifier.txt`.

2.4.2 Fitness Plot

The Figure 5 shows the best and average fitness over 300 generations.

2.4.3 Best Weights and Classifier Plot

The best weights found by the algorithm are:

- h1_w0: 2.3397, h1_w1: -1.9873, h1_w2: -1.7921
- h2_w0: -1.6181, h2_w1: 1.4844, h2_w2: 2.8663
- o_w0: -2.1063, o_w1: 4.2518, o_w2: 3.26

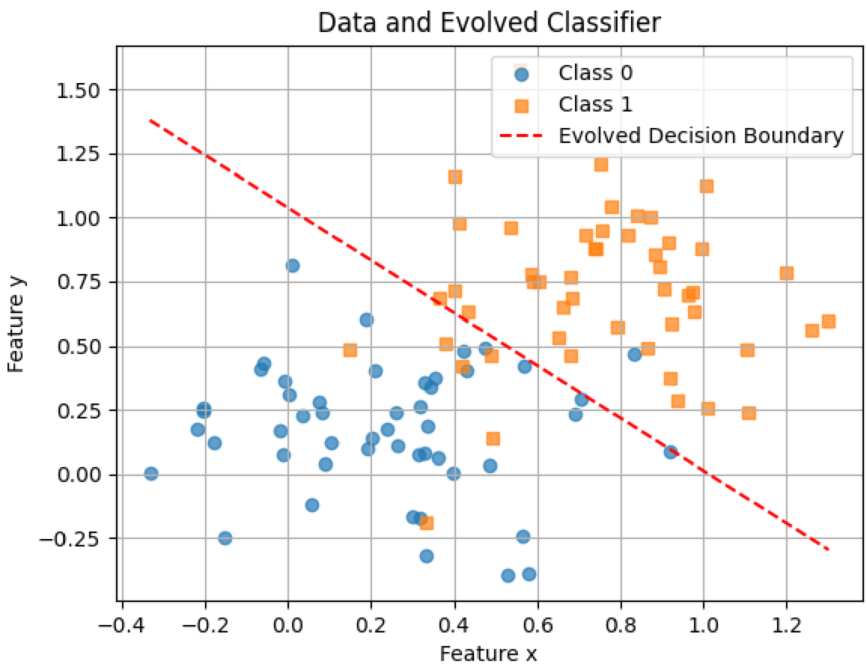

Figure 4: Data and evolved classifier.

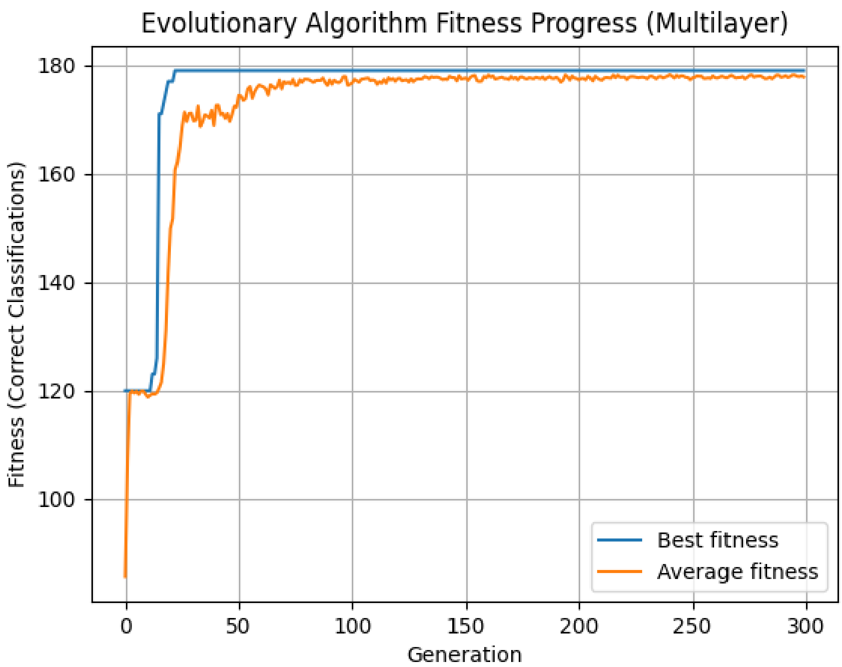

Figure 5: Best and average fitness for the multilayer ANN classifier.

The Figure 6 shows the data points and the decision boundary of the evolved classifier.

The algorithm was able to find a set of weights that correctly classifies 179 out of 180 data points. The non-linear decision boundary created by the multi-layer ANN is able to separate the two classes with high accuracy.

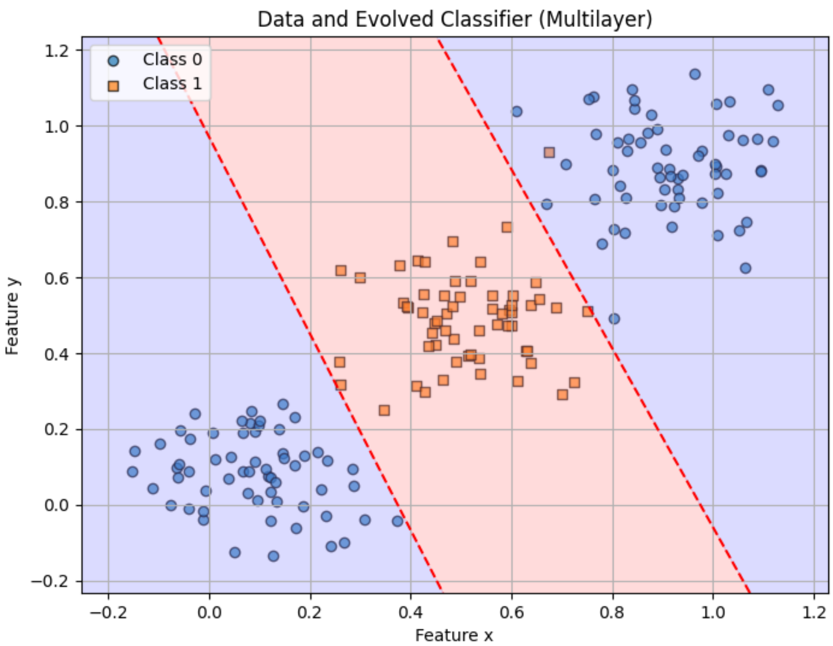

Figure 6: Data and evolved multilayer classifier.