An exercise in ANNs and Genetic Algorithms

Teo Bergkvist

May 2024

1 Introduction

This project is an implementation of an artificial neural network (ANN), trained using a genetic algorithm to play the classic game: Pong. The ANN is structured to process the game's state and output optimal paddle movements to compete effectively against an opponent. The genetic algorithm, inspired by natural selection processes, optimizes the neural network's parameters through iterative cycles of selection, crossover, and mutation. The project is a proposal of an exercise for the course `EXTG15` at Lund University.

2 Objective

The primary objective of this project is to demonstrate the practical application of evolutionary algorithms in training artificial neural networks for competitive gameplay. By simulating a series of Pong games and progressively refining the ANN's performance, the project highlights the potential of combining genetic algorithms with neural network training for solving complex decision-making tasks.

3 Simplicity

The project has been designed to require minimal setup before the code can be utilized. The only non-standard python libraries used are `numpy` and `pygame` which are both very popular, common and easy to install. The code and instructions of usage can be found at `https://github.com/tbergkvist/ANN-GA-pong`.

4 Example of computer exercise

It is of great importance that the students not only run the code, but also understand it. The best way to learn something, is to get hands-on and work with it. Thus, the exercise involves some modification of the code.

Following is a proposal on exercise tasks:

4.1 Exercise tasks

1

Set up the code using the simple instructions at `https://github.com/tbergkvist/ANN-GA-pong`.

2

Go through the python notebook `main.ipynb`, and make sure you understand what the code does. It can be necessary to look in the files `ann.py`, `ga.py` and `simulation.py` as well.

Question: Does the genetic algorithm do a good job of training the ANNs?
Question: How many generations is needed for good results?
Tip: Use pre-trained files if you have a slow computer, ask the teacher.

3

Change the number of neurons in each layer of the networks.

Note: Large amounts of neurons will take very long time.
Question: How do you think this affects the performance of the ANN?

4

Change the number of generations and the number of players.

Question: How do you think this affects the performance of the genetic algorithm?

5

Change the mutation rate.

Question: How do you think this affects the performance of the genetic algorithm?

6

Look at the multiplication function, how does it work? Try to implement another multiplication function, using one of the crossover functions named in the lectures.

7

Come up with 3 improvements that could be done to this implementation of training ANNs.

Side quest: If you come up with an improvement and write high quality code, you can push the code to `https://github.com/tbergkvist/ANN-GA-pong` and create a pull-request.

8

Come up with 3 other uses of this technology.

5 Improvements to the excercise

There are some improvements that could be done to make this exercise better.

5.1 Efficiency

Since the main objective was to write code that can be easily used and run, the neural networks are implemented using `numpy` arrays. This is quite a slow approach. If, for example `pytorch` was used instead, it could probably be more efficient. There are other improvements that can be done to increase efficiency as well, such as parallelization of the games.

5.2 Multiplication function

This function could be improved. Maybe some other crossover functions could be added to aid the student's understanding the theory.

5.3 Parameter values

The values of the parameters used for the ANN and genetic algorithm were chosen arbitrarily and tested only to some extent. For better and faster results these should be tuned.