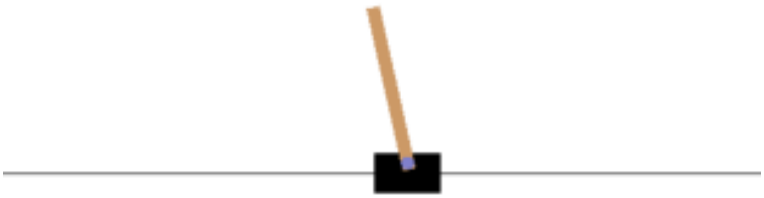
University of Sussex



Computer Science w/ Artificial Intelligence - Acquired Intelligence Adaptive Behaviour

Evolving agents

November 21, 2022

Report

234591

Contents

Abstract

Genetic Algorithms and Artificial Neural Networks can be combined to create a neural network that has its weights and biases trained by a genetic algorithm. We trained a neural network to solve Open AI's CartPole task which is the challenge of ballancing a pendulum on a cart on a frictionless track. We were successful with our endevour and discovered that the max fitness does not always stay at max once it has been reached unlike many other genetic algorithms. In this report I will explain the process of making a evolutionary artificial neural network to solve the CartPole-v1 task.

1 Introduction

Biology has formed the basis for many of the learning algorithms we use to train artificially intelligent machines. Genetic Algorithms and Artificial Neural Networks work in completely different ways, they can both be use for optimisation and learning each having their own strengths and weaknesses. Although they have generally kept two separate paths,recently Genetic Algorithms and Neural Networks can be combined to create a evolutionary artificial neural networks. Neuroevolution uses evolutionary algorithms to generate artificial neural networks , parameters, topology and rules. We decided to use neuroevolution to train a neural network to solve the CartPole-v1 task to see whether neuroevolution is worth it or if backpropagation makes neural networks work better

2 Methods

2.1 The Cart Pole Task

CartPole-v1 is one of OpenAI's enviroments that open source. In the cart pole task, A pole is attached by an un-actuated joint to a cart, which moves along a friction-less track. The pendulum starts upright and the goal is to prevent it from falling over. A reward of +1 is given for each time-step that the pole remains upright. The task is terminated when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center of the track.

2.2 Neuroevolution Genetic Algorithms

Many have attempted to use random forests, logistic regression and linear regression to solve the task with varying degrees of success but we chose to use neuroevolution for three main reasons;

1. Neuroevolution is not intensive computationally as there are no linear algebra calculations to be done.the only machine learning calculations necessary are forward passes through the neural networks.[1]
2. Neuroevolution is very adaptable so one can think of many different ways to manipulate the flexible nature of genetic algorithms.
3. Neuroevolution was found to be less likely to get stuck in local minima.[2]

In our Neuroevolutionary genetic algorithm, the individuals in our population have genotypes made up of weights and biases that are applied to the neural network used to train the agent. Our genotypes are represented in as a one-dimensional floating point array. The indexes for our weights are in the range of 0 to the number of inputs * the number of outputs in the neural network. We created an agent that could take in the weights and biases from the genotype and uses this to formulate a neural network from which the agent will use observations to decide on its action.

1. let a = Sum of all the weights
2. let b = Sum of all the inputs
3. let c = the bias

Action = $(a+b)*c$
This implies that the environment accepts a single number as an action, a discrete number (1 or 0). Here 0 corresponds to moving the cart to the left and 1 corresponds to moving the cart right.
The observations are within the range of -3.4 to 3.4 so therefore, the four observations will be a continuous *(float32)*.
These four observations correspond to:

- Position of cart
- Velocity of cart
- Angle of Pole
- Rotation Rate Of Pole

To use tournament selection in our Genetic Algorithm the steps we followed were:

1. Pick two genotypes
2. Differentiate the winner from the loser
3. Get the initial fitness of the loser
4. Crossover the loser with the winner
5. Mutate the loser
6. Get the final fitness after mutation and crossing over
7. Decide whether the final loser's genes are better than before
8. Change the genes accordingly

3 Results

3.1 Predictions

We believed that the max fitness will reach 500 (the maximum fitness) very quickly and stay there. We also thought that this would be consistent between the different iterations of the runs. Furthermore, We estimated that the average fitness would slowly creep up and then maintain a high fitness rate till the end

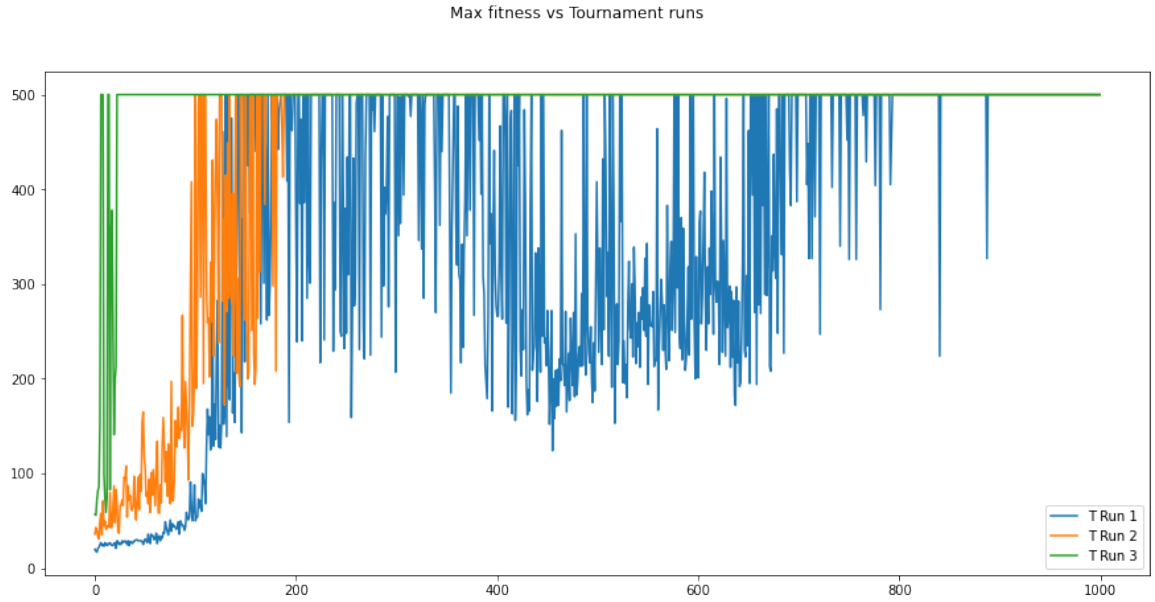


Figure 1: Performance comparison of population size from 5 to 25

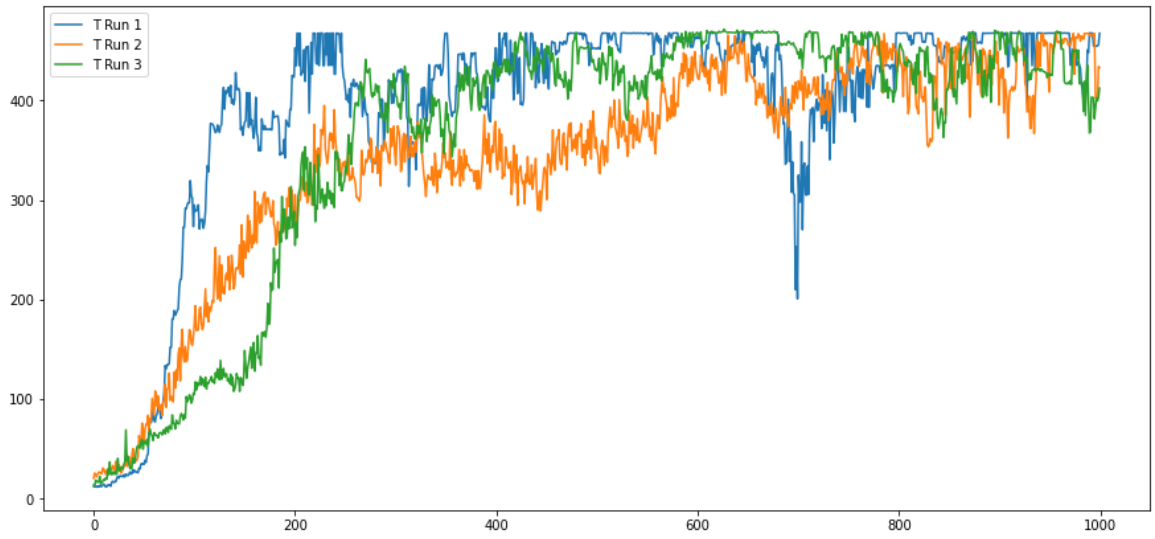


Figure 2: Performance comparison of population size from 5 to 25

3.2 Discussion

3.2.1 Max Fitness vs Tournament runs

The Max fitness rose to 500 very quickly, within the first 10 iterations, and consistently went up and down. Interestingly, Run three was consistently at 500 past the initial stage. Although this could be a very lucky

tournament run, it is more likely that this is down to an error regarding training crossing over from the previous tournaments.

3.2.2 Mean Fitness vs Tournament runs

The mean fitness steadily climbed to max fitness and then consistently stayed within 100-200 fitness of this level. It is interesting to note that on tournament run 1, the average fitness dropped drastically at the 500 tournament run mark which might imply that some mutations caused a serious drop in average fitness. From looking at tournament run 1 in [**Figure 1**], the constant up and down spikes of the max fitness along with crossing over of other individuals in the population could lead to the dramatic drop in mean fitness.

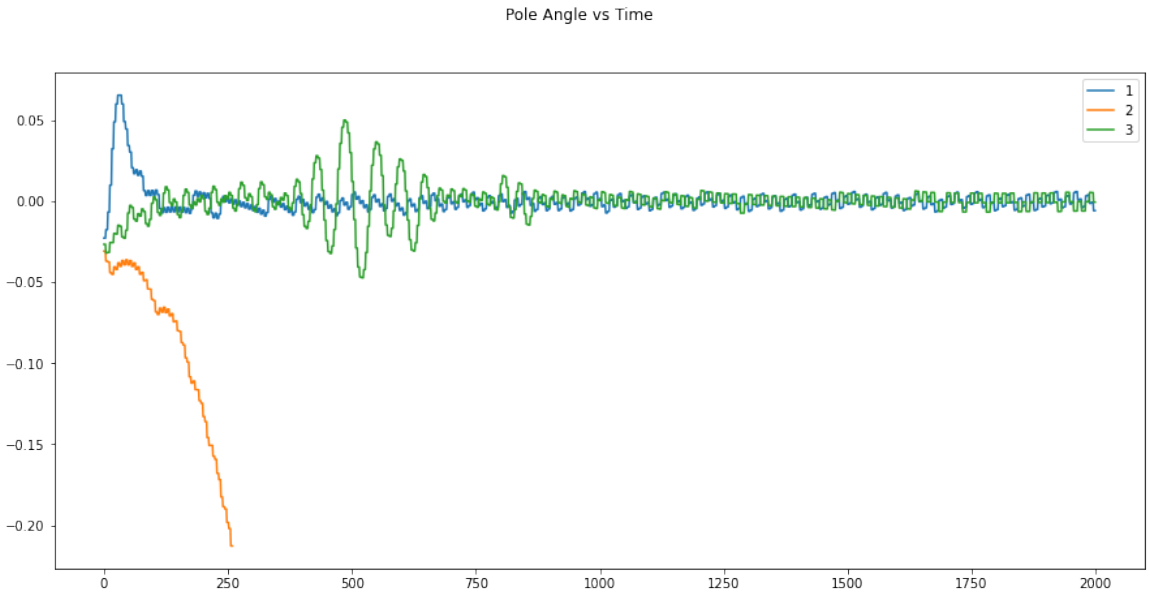


Figure 3: Performance comparison of population size from 5 to 25

3.2.3 Pole Angle vs time

The Pole Angle maintained within the range of 0.05 from 0 which actively showing how the agent is attempting to keep the pole around the value of 0 - when the value strays too far, it is quickly recalled to zero.This graph resembles the graphs of homeostasis with the feedback loops occurring once the temperature can gone past a specific limit[3]. In addition to this, in run 1 and 2 the pole-angle is maintained the entire time. This not only implies the success of the algorithm but also demonstrates the behaviour of the cart pole.In spite of this, there seems to be an anomaly in run two. Where the other runs maintain a steady value of +-0.05 around , run two seems to drop at 250 showing that that run failed and the cart pole fell.

4 Conclusions

It was interesting to see how quickly the max and mean fitness rose to the maximum possible fitness.I believe this is a testament to how good the evolutionary neural network(ENNs) works. Despite this, It still is still too early to tell whether ENNs are more efficient than regular NNs. To be able to conclude this we would have to conduct more experiments on regular NNs solving the cartpole task. Regardless in conclusion, We have discovered that a ENN is highly effective when solving the cart pole task.

References

[1] Edgar Galvan and Peter Mooney. “Neuroevolution in Deep Neural Networks: Current Trends and future challenges”. In: *IEEE Transactions on Artificial Intelligence* 2.6 (2021), pp. 476–493. DOI: 10.1109/tai.2021.3067574.

[2] Matthew Hutson. “Artificial intelligence can ‘evolve’ to solve problems”. In: *Science* (2018). DOI: 10.1126/science.aas9715.

[3] Lumen Learning amp; OpenStax. *Anatomy and physiology I*. URL: https://courses.lumenlearning.com/cuny-csi-ap-1/chapter/homeostasis-and-feedback-loops/.