AIAB CW2 Report

By 246743

Abstract

Cart Pole is a realistic physics model developed by Gym[4] that needs to be trained to make the data fit the rules of the corresponding game to complete the game. In this experiment, I used the Genetic Algorithm (GA) to continuously filter and refine the populations so that the model has an almost 100% probability that any one population will evolve to have individuals with 500 fitness after multiple generations. With respect to the genetic algorithm, I use elitism, crossover and mutation to improve the data at each epoch, with a high crossover probability and low mutation probability. In addition, several videos of the training results is attached and the corresponding data will be plotted and analysed in graphical form.

Introduction

1. Model

The rules of the Cart-Pole game are simple[6]: give the cart below a 0 or 1 command to move it to the left or right. The goal is to keep the stick on the cart tilted no more than 15 degrees to the left or right, and the cart cannot move more than 2.4 units to the left or right. At each time unit the reward value is also returned depending on specific calculation method, the higher the total reward, the better the result.

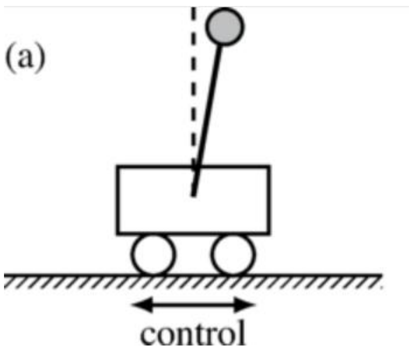


Fig 1

2. Genetic Algorithm (GA)

Genetic algorithms are based on Darwin's theory of evolution[3], which simulates natural selection, the survival of the fittest, and the evolution of optimal solutions to problems through amounts of generations of inheritance, mutation, crossover and replication. Genetic algorithms may seem magical, but the idea is relatively simple to implement.

2.1. Individuals and Population

An individual corresponds to an individual gene, whereas a population is a group of genes. When mapped from biology to mathematics, a gene should be represented by a list of numbers, e.g. [0, 1, 2]. A population with three genotypes would mathematically look like this: [[1, 2, 3], [1, 2, 1], [2, 1, 3]]. In a genetic algorithm, each of these numbers can be referred to as a gene.

2.2. Fitness function

In nature, it is the rules of competition that determine the winners and losers, and in a mathematical model there should be a set of rules set by humans to distinguish between good and bad genes, which is the original purpose of the fitness function. The fitness value will also be stored as an attribute of each individual and can be called upon when needed.

2.3. Elitism

The core point of the genetic algorithm is to apply a meritocracy in each generation. By meritocracy, we mean that the batch with the greatest fitness in the current generation is retained. They should not go through crossover and mutation, but be transferred directly to the next generation.

2.4. Crossover

Two genotypes are selected and each gene from the high quality genotype overwrite onto the poor quality genotype with a certain probability. For example, if the fitness of [3, 3, 3] is higher than that of [2, 2, 2], then for each gene in [2, 2, 2], there is a chance that it will be transformed into a gene at the corresponding position in [3,3,3]. This operation ensures a steady increase in the average fitness of the genes, so performing the crossover operation is key to effectively reducing the training time.

2.5. Mutation

To further increase the fitness value of the genotype, mutation is performed by mutating each gene in the genotype into a random gene (random number) with a certain probability. It is a way to break out of Local Maxima, i.e. it allows the population to be calculated closer to Global Maxima after a certain number of generations.

Methods

1. Agent

The Agent Class needs to pass in input_size and output_size to confirm the number of genes in each individual. The calculation formula is linear:

$$\text{num\_genes} = \text{num\_input} \times \text{num\_output} + \text{num\_output}$$

The Agent determines the movement of the cart. Whenever the pointer points to a new genotype, Agent reassigning the weight and bias according to the genes, and then calculating the forward formula. If forward > 0, the cart moves to the right, otherwise the cart moves to the left.

$$\text{forward} = \text{observation} \times \text{Weight} + \text{bias}$$

2. Flowchart

The flowchart of the one training is shown in Fig 2. First, Elitism is used to classify the population into Elite and Loser, crossover and mutation is performed on the Loser, then the Elite and the changed Loser are synthesised into a new population[2]. The fitness value of each individual in this population is calculated, after that determine whether any individual has reached the target (fitness value >= 500). If so, it will be output directly. Otherwise, the loop is repeated until epoch >= 1000 and the final result is output.

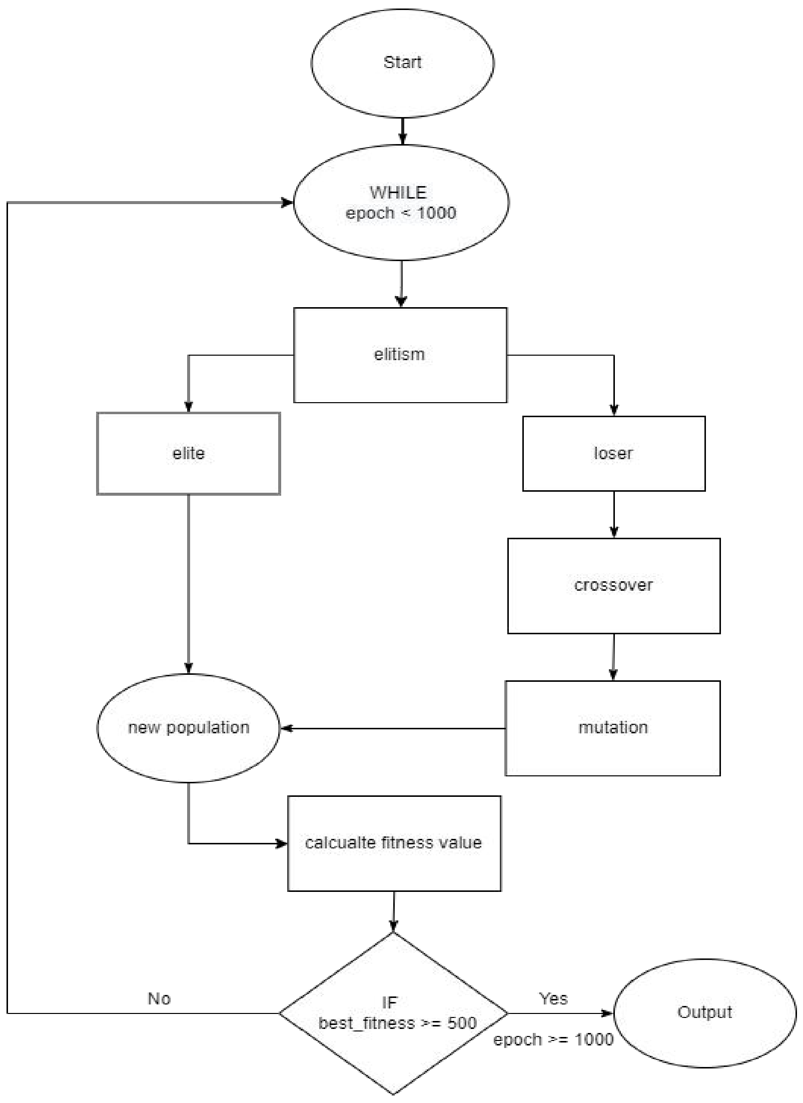


Fig 2. Process of one training (up to 1000 epoch)

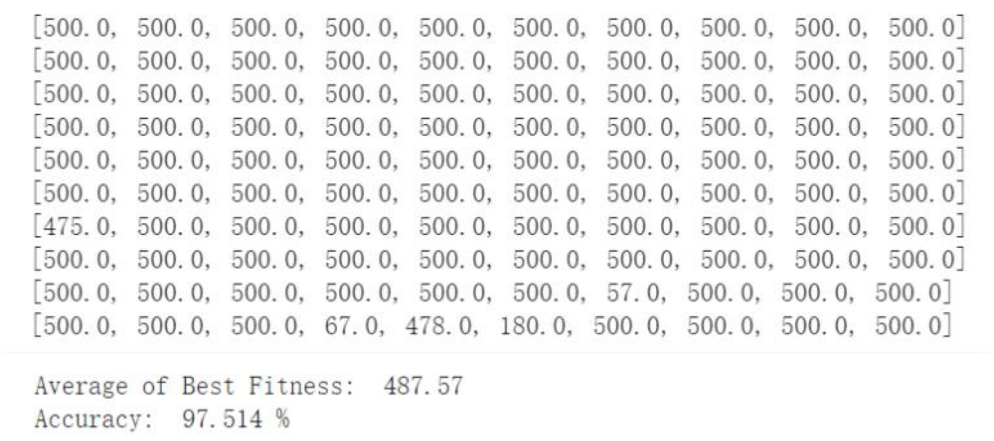
Results

Video link: [Best Result Visualize](#) (only render episode showed)

Video link: [Live Demo Training](#) (use **Cartpole_GA.py** as Google Colab cannot visualize result)

The training model is able to maintain 96% ~ 98% accuracy.

An example of result after 100 times training is shown below:



```
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[475.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 500.0, 500.0, 500.0, 57.0, 500.0, 500.0, 500.0]
[500.0, 500.0, 500.0, 67.0, 478.0, 180.0, 500.0, 500.0, 500.0, 500.0]

Average of Best Fitness:  487.57
Accuracy:  97.514 %
```

Fig 3. One hundred best fitness, their average and accuracy

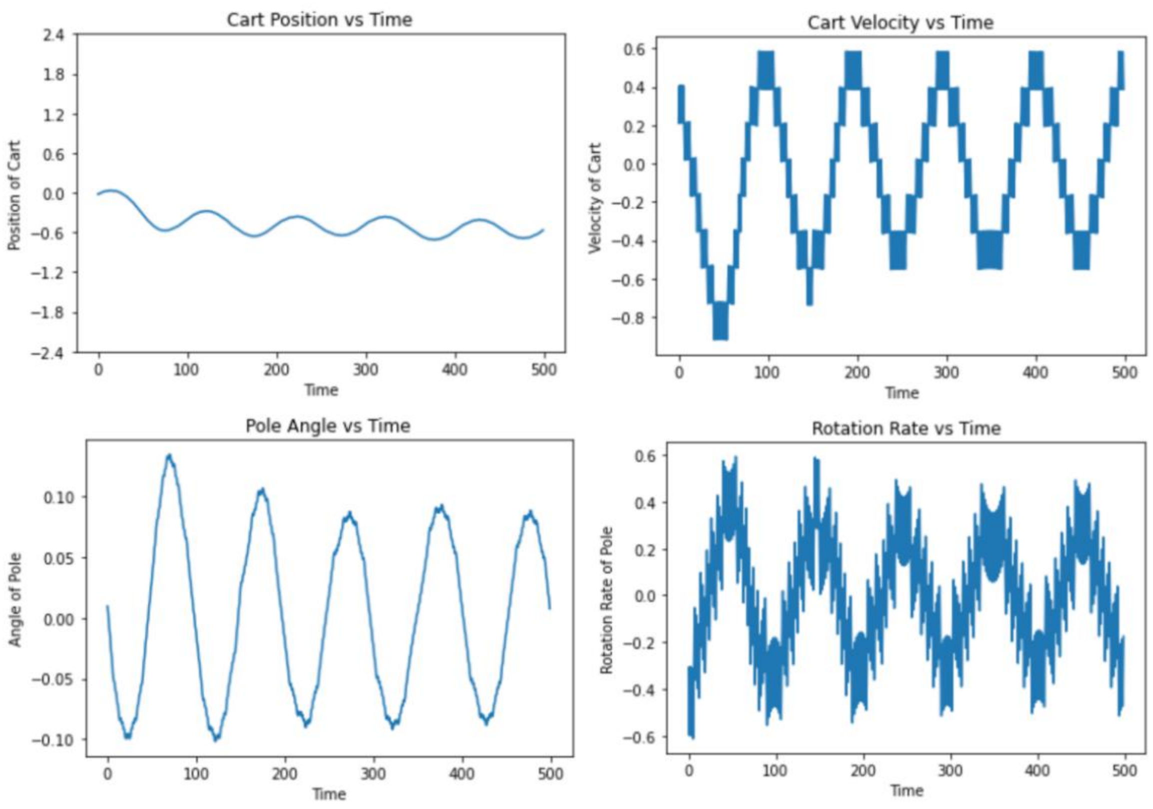


Fig 4. Plots of a random training result

Discussion

1. Evolving of the controller

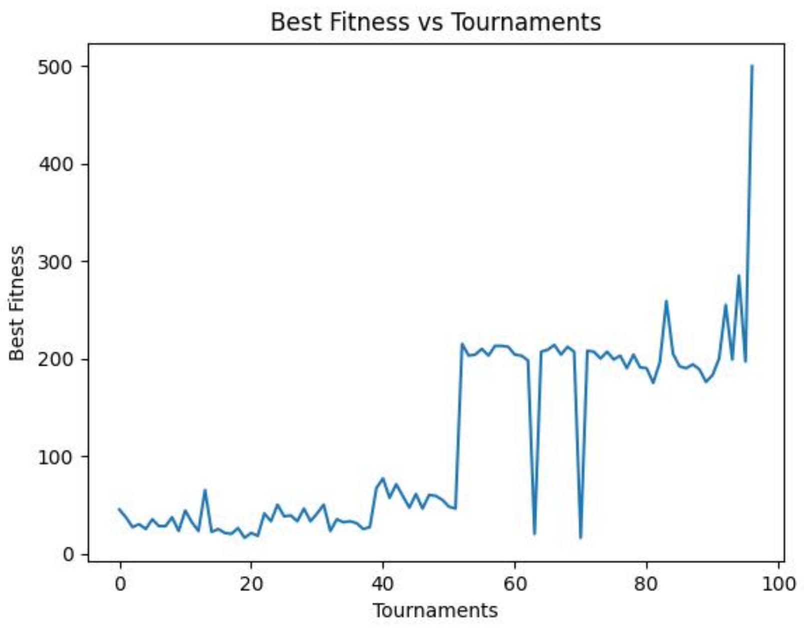


Fig 5. Best Fitness vs Tournaments

As can be seen from Fig 5, the training process of GA is a step wise ascent. Whenever a stage is reached (e.g. 0 to 50, 50 to 90), it stays at that stage and waits for a new evolution. Occasional cliff drops are possible, e.g. x=62, x=70, but the probability is extremely small. And presumably, this probability is related to the Elite Retention Ratio, since elites are inherited directly without changes, so theoretically, the lower the Elite Retention Ratio, the less likely a cliff drop will occur.

2. Behaviour of the best controller at different stages

Video link: Different Stages Behavior

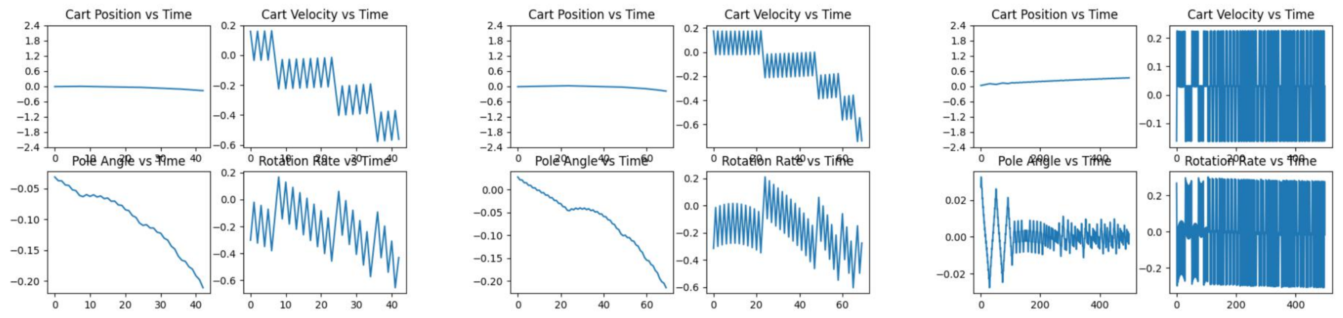


Fig 6. Graphs plotted in Video

The Best Controller is visualised at three different stages (start, halfway, end of optimisation). Fig 6 shows the graph plotted in the video "Different Stages Behavior".

As can be seen from the horizontal coordinates of the image, the fitness values of the controllers are indeed increasing. At the beginning the cartpole only lasts for 40 time units (fitness ≈ 40). In the halfway, it maintains about 70 units of time until it finally succeeds in obtaining a fitness score of 500.

Comparing the first two graphs (start and halfway) it is clear that the trajectory of the cartpole is very similar, they both tilt to the left (with a negative pole angle) and lose focus and lose the game. However, at the end of the optimisation, the cartpole's trajectory is understandably very different from the previous two. Because Mutation is constantly providing random parameters, this keeps the controller fresh and may lead to amazing progress.

3. About the Elitism

Elitism is a function that combines new genotypes with old best genotypes. However, unlike what is stated in the lection (Fig 7), I did not take a sampling filter to assign new genotypes[5], but more explicitly moved losers with lower fitness to the next step for crossover and mutation (which also operate on the basis of elites). After all this, the elites from the old population were combined with altered losers to form a new population, and this proved to be equally feasible.

4. Mutation and the Local Maxima[1]

In this experiment I have made one change to the Mutation function. Prior to this change, the mutation function was only used to vary one of the five genes with a certain probability, as traditionally known. However, the results were not satisfactory, and the final population had a high probability of being stuck on local maxima. Even though fitness=500 could sometimes be reached, this was due to the fact that the randomly initialised generation 0 had a very high fitness value (e.g. 475), and in most cases fitness would stagnate after reaching between 100 and 300.

After combing through the algorithm a bit, I let each gene in the genotypes have a probability of getting a change. It turned out that this did effectively solve the local maxima problem. Also the increase of populations number could effects on local maxima, but it is not decisive, rather the effect follows significantly after the mutation function boost.

Conclusion

Overall, the experiment achieved satisfactory results, and the accuracy can always be guaranteed to be above 95% (usually in the range of 96% to 98%). The behaviour of the best controller at different stages was analysed reasonably well in relation to the graphs. Some of the algorithms were also based on Lecture's examples, with some modifications to achieve better results in practice.

There is still space to improve the algorithm, for example, the probability of mutation and crossover can be determined by the fitness value of each individual. Individuals with higher fitness values have a higher probability of experiencing change, which is in line with nature's principle that only the stronger have a greater chance of surviving.

Reference

1. Carlson, E. A. (2011). *Mutation : the history of an idea from Darwin to genomics*. Cold Spring Harbor, N.Y: Cold Spring Harbor Laboratory Press. https://www.cshlpress.com/default.tpl?cart=1652315331374228912&fromlink=T&linkaction=full&linksortby=oop_title&--eqSKUdatarq=911
2. Corus, D., & Oliveto, P. S. (2018). *Standard Steady State Genetic Algorithms Can Hillclimb Faster Than Mutation-Only Evolutionary Algorithms.* IEEE Transactions on Evolutionary Computation, 22(5), 720–732. https://doi.org/10.1109/TEVC.2017.2745715
3. Man, K. F. (Kim F. ., Tang, K. S., & Kwong, S. (1999). *Genetic algorithms : concepts and designs*. London, [England: Springer. https://ebookcentral.proquest.com/lib/suss/detail.action?docID=3073392
4. OpenAI gym (2022) *CartPole-v1* https://gym.openai.com/envs/CartPole-v1/
5. Shao, X., Li, X., Gao, L., & Zhang, C. (2009). *Integration of process planning and scheduling—A modified genetic algorithm-based approach*. Computers & Operations Research, 36(6), 2082–2096. https://doi.org/10.1016/j.cor.2008.07.006
6. Wang, Z. T., Ashida, Y., & Ueda, M. (2020). *Deep Reinforcement Learning Control of Quantum Cartpoles*. Physical Review Letters, 125(10), 1–100401. https://doi.org/10.1103/PhysRevLett.125.100401