Evolutionary Algorithms - Computational Intelligence

40272321 - Connor Ness

ABSTRACT

Report detailing the development process of evolutionary algorithms that evolve the weights contributing to a neural network. Primarily focusing on selection and crossover methodologies, why such methods were selected and their parameters. Discussion of experiments and analysis undertaken to further investigate the performance of the algorithms to assist in identification of trends and any observations made from these parameter alterations. Concluding with an analysis of the average outputs coming from the selected algorithm and parameters and a discussion of how to further improve these algorithms.

ACM Reference format:
40272321 - Connor Ness. 2018. Evolutionary Algorithms - Computational Intelligence. In *Proceedings of Coursework, Edinburgh Napier University, April 2018 (SET10107)*, 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Evolutionary algorithms are functions that serve the same purpose as biological evolution. A population set of individuals is created where each individual has a "fitness" rating which correlates to that individuals ability to overcome an obstacle. Algorithms are used to select the "best" individuals of a population by fitness and create children from the selection. Children receive gene components from their parents but also have a mutation factor to permit new genetic options and variety within the population for further evaluation.

2 APPROACH

The methods used utilise evolutionary algorithms to land rockets on a pad while using fuel as a resource. The algorithms evolve weights of the neural network employed to control the rockets.

The aim of this work is to design an evolutionary algorithm to reduce the fitness rating of simulated runs within confines of a training set to then apply to a testing set of data.

2.1 Selection

Various methods of selection where chosen as potential implementations for algorithm choice. Of these researched methodologies, Tournament and Roulette type selection were chosen. Selection pressure is the pressure within a population to favour certain traits/genes over others, this pressure will allow these desired features to spread throughout a population as children receive the relevant genes [1].

SET10107, Edinburgh Napier University

DOI: 10.1145/nnnnnnn.nnnnnnn

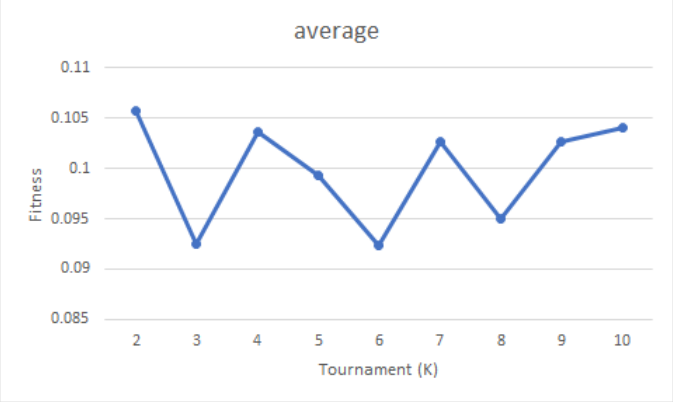

Figure 1: Establishing which k provides a better fitness

2.1.1 Tournament. Tournament selection methodology was chosen due to simplicity of implementation which provided a solid foundation for parameter modification to observe results. This method involves selecting a variable number of individuals within a population ***k***. The best individual within ***k*** is then selected to become the ***parent***. This process can be repeated several times taking ***k*** as a population to select a new ***k*** from. Tournament can have the selection pressure modified by modifying the number of individuals encapsulated within ***k*** - a high number of individuals translates to an increased selection pressure, a low number of individuals translates to a decreased selection pressure. A higher selection pressure can assist in a faster convergence on improved solutions which is of benefit due to the evaluation limit in place.

For this project, several varying capacities of ***k*** were tested to observe which size provided a better fitness rating.

In order to determine an efficient tournament size, an average of ten runs was taken for each size of k from 2 to 10, the results were as follows [Figure 1.].

Due to the variations with no seeming trend, it is believed tournament size has influence over the fitness; but its influence is either small, or would work better in junction with more parameter changes. For this reason the tournament sizes which resulted in the highest and lowest rating will be tested with changes (2 and 6 respectively) until a more obvious trend is observed.

2.1.2 Roulette. The roulette methodology was chosen due to the randomisation selection method ensuring all parents have a chance of being selected and unlike tournament - the variable parameter changing is not chosen by the author providing a differing perspective on selectivity. This method involves calculating ***s***, the sum of all chromosome fitnesses in the population; generating ***r***, a random positive number between zero and ***s***; then finally looping through the population, adding each fitness for each loop until this total is greater than ***r*** - this individual in the loop is selected.

The roulette method was tested by observing ten runs [Figure 2].

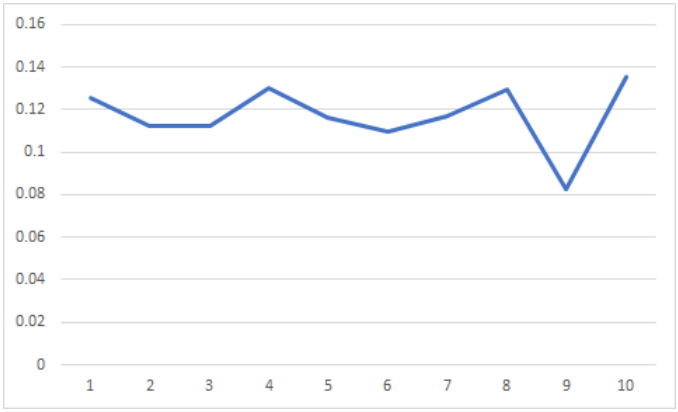

Figure 2: Observing differences on roulette with base parameters

2.2 Crossover

Crossover is the combination process of at least two parent chromosome resulting in a child chromosome. One point (1pt) crossover was utilised in this project. In one point, a random "cut point" is made in each parent chromosome. The length of genes after this cut are then swapped resulting in a combination gene which is attributed to a child [Figure 3.]. One point crossover allows each parent an equal opportunity at influencing a childs genes. Due to time constraints, n-point crossover was not implemented successfully, however in theory this crossover functions similarly but cuts and swaps the chromosome ***n*** times resulting in more variance in the child chromosomes.

3 PARAMETERS

For genetic algorithms, a main three parameters are present. Mutation rate - the probability that a gene within a chromosome will mutate, crossover rate - the frequency in which crossovers are applied to chromosomes, and population size - the number of individuals within the group [2]. Mutation rate was set to varying different values to determine which would yield the best outcome, the roulette selection method was used for this purpose due to having more consistent outputs than tournament.

MutateRate	MutateChange	Fitness
0.5	0.2	0.1807
0.3	0.2	0.255
0.1	0.2	0.0.1961
0.05	0.2	0.2521
0.01	0.2	0.174

From this 0.01 was established as the best mutation rate for fitness outcome and was kept, the outcomes were noted to be different from expected. As a high mutation rate can negatively effect a fitness of a population due to too many mutations occurring at once, it was expected that the mutation rate would notably decrease with each test. A lower mutation rate limits

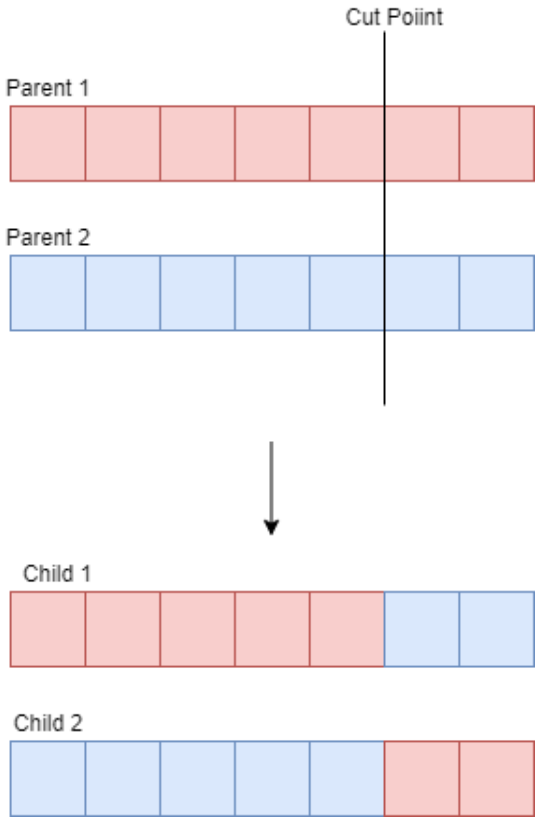

Figure 3: One point crossover

MutateRate	MutateChange	Fitness
0.01	0.5	0.2702
0.01	0.3	0.1671
0.01	0.15	0.3095
0.01	0.1	0.2998
0.01	0.05	0.0805 0.211

As lower mutation change rates were tested, results became more inaccurate; for this reason a higher change rate was used at 0.3.

Finally the maximum and minimum gene weight values were edited, these were kept to be equal when altering.

minGene	maxGene	Fitness
-5	+5	0.2211
-4	+4	0.2918
-2	+2	0.1824
-1	+1	0.1995

The weighting of -2/+2 allowing for a greater range of gene values to be explored while not increasing "waste time" as runs explore more values than necessary for improvement factors.

Unless stated otherwise, for experimentation the values used are a 0.01 Mutation rate, a 0.3 Mutation change rating, and -2/+2 for gene weightings.

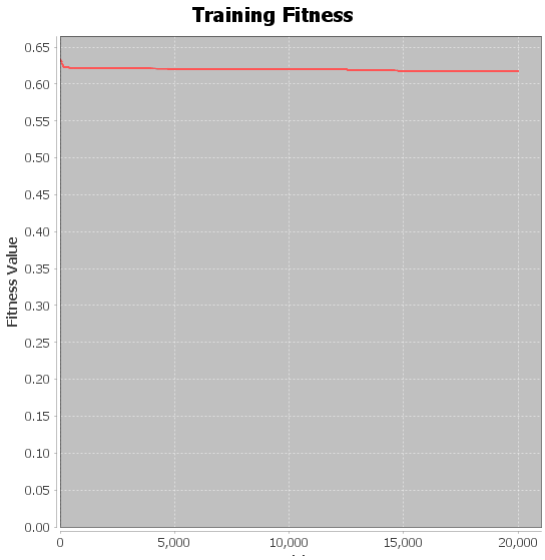

Figure 4: Unable to overcome a problem

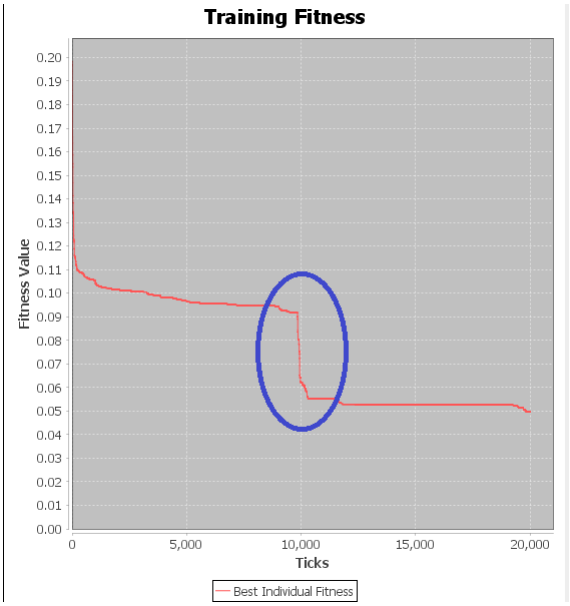

Figure 5: Exponential decay followed by a sudden increase in improvement rate

4 EXPERIMENTS AND ANALYSIS

In initial testing, a GUI was utilised to allow observations on fitness improvements to be made in real time, this revealed that occasionally a run would experience a dramatic slow in improvement as a halting point is reached. Improvement generally resumed the rate prior to the slow down after a time, but this incurs a loss in overall improvement due to the limited number of runs. In the shown run, the algorithm was unable to overcome a difficulty to any meaningful capacity - while rare, results like these occurred [Figure 4.].

Alternatively, the results can experience a sudden massive improvement rate which affects the outcome in the opposite way [Figure 5.].

4.1 Population size

Base population size was changed to several different values and tested with both selection methodologies implemented.

Roulette Population	Fitness
100	0.2252
200	0.2448
300	0.1722
400	0.1959
500	0.1282

Tournament Population	Fitness
100	0.2307
200	0.2274
300	0.2244
400	0.2114
500	0.2325

The hypothesis before testing was that population size would consistently improve the fitness, this was true for the roulette selection but not for the tournament selection. Therefor to further investigate the effect of population size on the selection process, using the roulette method only; evaluation count will be limited in order to observe the effects on a simulation with fewer possibilities of improvement. To provide a robust dataset to evaluate, population sizes 100 and 500 will both be tested.

4.2 Evaluation Count

Evaluations	Fitness
15000	0.2267
10000	0.2379
5000	0.2591
1000	0.2998

Population Size: 100

Evaluations	Fitness
15000	0.1915
10000	0.2776
5000	0.3381
1000	0.4068

Population Size: 500

As expected within population size 500, the less evaluations able to take place - the worse the fitness rating. However unexpectedly within smaller population sizes - the evaluation rating, while still effected in the same way, experiences a smaller variance in fitness rating when under less evaluations.

5 CONCLUSIONS

Set Mutation Rate/Change	Best/Avg. Fitness Activation
Training 0.01/0.3	0.0963/0.1195 Binary Step
Test 0.01/0.3	0.1006/0.125 Binary Step

Roulette Selection

Set Mutation Rate/Change	Best/Avg. Fitness Activation
Training 0.01/0.3	0.0906/0.1075 Binary Step
Test 0.01/0.3	0.0991/0.1025 Binary Step

Tournament Selection

The project was successful in reducing fitness rating from the base solution, implementing several methods and altering values to further reduce the values where appropriate.

The experimental alterations made assisted in displaying the effects of the variables upon an implemented solution which allowed observations to be reliably made upon the data.

The roulette method permitted a greater reduction in fitness rating than the tournament method, and the crossover implementation functioned as intended.

There were several instances where the values of data on duplicated simulations were of larger discrepancy than expected, for this reasons averages were taken for observations to be made. This was far more prevalent in the tournament method than the roulette method.

6 FUTURE WORK

To further improve the fitness reduction, several other algorithms could be employed in both selection and crossover. For selection I would improve the tournament selection. I believe the tournament implementation to be less reliable than the roulette and returns results with far too much variation on duplicate runs. I would also implement a 2 or N-step crossover to further enhance the genetic differences in children from parents.

A potential further development would be mutation rate depending on fitness, the fittest in the population could only have minor mutations whereas the lower fitness ratings have higher mutations in order to maintain a consistent fitness while allowing for a greater random chance of extremely positive mutations.

7 REFERENCES

[1] Downing, K. (2015). Evolutionary Algorithms. In Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems (p. Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems). MIT Press.

[2] Jacobson, L., Kanber, B. (2015). Genetic algorithms in Java basics (Expert's voice in Java). New York, NY: Apress.

[3] Introduction to Evolutionary Computing. (2004). Kybernetes, 33(5/6), 1064-1065.

[4]