Parameter Tuning of Evolutionary Algorithm for Rocket Landing
40401270

ABSTRACT

For algorithms that needs to interact with an high quantity of data, or high number of possible solutions, traditional approaches to problem solving may be not be approached, because of the computational power needed to find the best solution.

Evolutionary Algorithms (EAS), are defined as general purposes optimization algorithms that can be applied to any existing problem, by associating to the algorithm, a fitness function, to find an optimal solution. Lower is the fitness score associated to training and testing data, once have run our evolutionary algorithm, and better the EAs will perform. However, find an optimal solution, is necessary to find the best set of parameters. This project focuses on the parameters tuning of a Multi-layer Perceptron, in order to find the best weights for our artificial neural network, applied to the automation of a Rock landing system.

ACM Reference format: 40401270. 2022. Parameter Tuning of Evolutionary Algorithm for Rocket Landing‰ In Proceedings of Coursework, Edinburgh Napier University, April 2022 (SET10107), 5 pages. DOI: 10.1145/nnnnnnn.nnnnnnn

1. INTRODUCTION

The aim of this report is to describe and analyze the outcome of the Computational Intelligence coursework, focused on the implementation of an evolutionary algorithm capable to calculate optimal weights of a MLP (Multi-Layer Perceptron) artificial neural network adopted in the automation of a rocket landing system.

The nature of the evolutionary algorithm implemented is a steady state genetic algorithm, a simplified version of a generational evolutionary algorithms, where two parents are carefully selected across the population, obtaining two offspring (or variants / mutations / children), that will be inserted inside the population. [7]

2. APPROACH

An evolutionary algorithm is a population-based optimization algorithm, that based on a given a subset of possible solutions (the population of solutions), fetch the best set of solutions.

Generally, an evolutionary algorithm approach the following stages:

1. Initialize the population: extract two parents from the population set.
2. Apply selection for reproduction of the two parents (which generates two offspring).
3. Mutate the offspring
4. Substitute other members of population with the off-spring, by evaluating the offSpring
5. Replace subsets of population (worst genes), or Discard them, by reducing so the genes under consideration.
6. Repeat from stage 1 to 5 until a well-defined condition is met.

The maximum number of function evaluation that our evolutionary algorithm can run is 20000, this value, represents the conditions to meet, before stop the execution of the EA and evaluate fitness score in both training and testing data.

As the goal of this project is to find the best fitness value, is important to know a first approach could be implement and evaluate the performances offered by different evolutionary operators: crossover, selection and replacement.

Crossover: allows to reassemble the information obtained from two members of the populations defined as parents. Variants of crossover operator, implemented in this coursework are:

- **One and two points crossove**r, where N subsections of the chromosomes are selected and swapped between them, generating so two new children;
- **Uniform Crossover**, where each member of the population is considered at the same level, and the choice behind the genes swap is purely random.

Selection is another evolutionary operator, where main optimal variants are:

- **Tournament selection:** is a relatively simple selection algorithms with Space Complexity o(N), where N, is the size of current population dataset shortlisted. As it is relatively simple to implement, something it can introduce noise with the fitness analysis method. The logic behind this selection method is the following: by having in input a Population P, the tournament selection operator randomly select N children (solutions), as part of the current tournament, and just the fittest (or best) children (set of solutions), are shortlisted and allowed to be considered in the following section, with the next generation of children.
- **Best selection:** the fitness score of each member of the population is analyzed, and just the two best genes are selected as parents.

The third but not less important operator is the replacement one:

- **Tournament replacement:** a subset of the population N is choice, and the single chromosome with worst fitness score is replaced.
- **Replacement worst:** selection operator that at each run, select the children with the worst fitness score, and replace it with a new one (without compare the fitness score of the current worst one, with the fitness score of the new children).

Between the list of parameters used, there are parameters related to the mutation. Mutation allows to generates a diversification of currently

3. Experimentation & Analysis

Empirical research was carried out in order to find the best set of parameters that could provide a loss fitness score value. Every implemented evolution operation has been tested, running all the tests for a maximum of 20,000 generations. To have a better estimation of the fitness score in both training and testing data, all the given results are the average of 10 runs. It means that every evolution operation has been run 10 times, and the fitness score is the average of the 10 sets of results produced.

3.1 Initialization

The random and augmented initiations operators have been tested, by achieving the following performances:
The produced results, have been obtained by using the default parameters values, provided by the coursework draft:

- Mutation: 0.04 and mutation change: 0.1
- Replacement: tournament with given size:
- Hidden Nodes: 5
- Minimum and maximum gene: -3 +3
- Population size: 40
- Crossover: 2 points

Operator	Training set fitness	Test dataset fitness
Random	0.0942	0.18566

The goal is to obtain an optimal fitness score, which theatrical, has value 0.00;

Even if it possible to achieve values very close to the absolute zero, there is the risks that we may

have found a list of parameters that may create an overfitted multi-layer neural network[6]. Overfitting is the scenario where the implemented machine learning model, neural network, or in general the statistical model implemented, completely fits to the training set, providing corrects outputs just to the input already used in the training dataset, and wrong output, for any other data given in input. We can see indeed, that the nature of values adopted before were giving a great fitness score on training dataset, but low performances on testing set, sign that the implemented neural network, is overfitted.

3.2 Tournament selection

As we can notice from the achieved results, the fitness values obtained by using the given parameters was not satisfying. Empirical research has been executed, obtaining more optimal parameters:

- Number of hidden: 12
- Population size: 50
- Mutation rate: 0.45
- Mutation changes 0.95
- Cross-over: 2-points
- Number of hidden: 12
- Minimum and Maximum gene: -1 and 1

Below, we can see the report of the experimentation executed, to find a best tournament size:

Tournament size	Training dataset fitness	Testing dataset fitness
5	0.0143	0.0227
6	0.0124	0.0300
8	0.0123	0.0240
9	0.0111	0.0201
10	0.0131	0.0267
20	0.0100	0.0227
30	0.0117	0.0259
40	0.0140	0.4482
50	0.0170	0.0542
60	0.0139	0.0485
70	0.0176	0.06716

As we can see from the experimentation results reported in the table above, Tournament size 9 is the best performing so far, and is the tournament size that will be adopted for the next experimentations.

3.3 Cross Over Operator

The table below shows the results obtained by the three different crossover operator implemented.

Cross-Over Operator	Training dataset fitness	Testing dataset fitness
UNIFORM	0.03242	0.06242
2-point	0.0132	0.0197
1-point	0.01377	0.0400

We can notice that 2points and 1 point crossover operations are the best performing ones. For its performances with the training set and low fitness score in the testing data, 2-point cross over operator has been chosen as default operator for the next experimentations.

3.4 Activation Function

The table below, shows the results produced by the different activation functions available, where description of the individual activation functions, fall-out from the goals of this report, and so, just the related results will be reported and compared:

Activation Function	Training dataset fitness	Testing dataset fitness
RELU	0.0170	0.0378
TANH	0.0234	0.0608
SELU	0.0135	0.0199

From the experimentations executed, it has been noticed that SELU activation function, is actually the best performing in terms of fitness function, but RELU function provided optimal results anyway, but in an extremely fast training time, compared to any other activation function used.

3.5 Minimum and Maximum gene

Min and max gene represents the range of values of the weights assigned to the Neural network, when created. Below are reported the average of fitness score on both training and testing dataset, realized by 10 runs, in correlation to the different minimum and maximum gene value assigned.

Minimum gene	Maximum gene	Training dataset fitness	Testing dataset fitness
-5	+5	0.0851	0.1265
-4	4	0.0117	0.0116
-3	3	0.0123	0.0354
-2	2	0.0131	0.0305
-1	1	0.0167	0.05116

.

4. CONCLUSION AND FUTURE WORKS

The previous sections provided the background knowledge and implementation details that allowed us to find an optimal fitness value, among the parameters considered. For the final testing, with what are so far the best parameters found during past experimentations made, the algorithm has been run ten times, and an average of the obtained results has been noted, producing the following results:

Attempt Number	Training dataset fitness	Testing dataset fitness
1	0.0075	0.0939
2	0.0171	0.0315
3	0.0128	0.0208
4	0.0138	0.0151
5	0.0108	0.0145
6	0.0067	0.0069
7	0.1093	0.0081
8	0.0105	0.0207
9	0.0085	0.0954
10	0.0125	0.0118

The average values among the 10 tests run are:

Training dataset fitness	Testing dataset fitness
0.0209	0.3187

Of course, these results may change in according to the parameters values adopted. The values of the parameters adopted to obtain the above results are:

- **Selection:** Tournament (having size equal to 9).
- **Population size**: 50
- **NumHidden**: 12
- **Number of hidden nodes**: 12
- **Minimum gene**: -1
- **Maximum gene**: 1
- **Mutation rate**: 0.45
- **Colling rate:** 0.0011
- **Activation function:** SELU
- **Probability of mutation rate:** 0.95 (95%)

It has been shown and proofed that number of hidden layers and populations size are extremely sensible and may leads to overfitted neural network; It has been fundamental to adopt a

minimum of mutation probability, to further diversity the family of solutions to analyze.

The weights of the neural network implemented are now optimal and provides acceptable results in both training and testing set. For this reason, we can confirm the successful completement of the project, and affirm that the neural network is now able to make the rocket land successfully.

5. REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[2] Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. University of Maryland at College Park.

[3] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of Physics: Conference Series* (Vol. 1168, No. 2, p. 022022). IOP Publishing.

[4] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM, New York, NY, 1-10. DOI= http://doi.acm.org/10.1145/964696.964697.

[5] Yu, Y. T. and Lau, M. F. 2006. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *J. Syst. Softw.* 79, 5 (May. 2006), 577-590. DOI= http://dx.doi.org/10.1016/j.jss.2005.05.030.

[6] Series. ACM, New York, NY, 19-33. DOI= http://doi.acm.org/10.1145/90417.90738

[7] Gacôgne, L. (2002). Steady-state evolutionary algorithm with an operators family. *EISCI Kosice*, 173-182.