Challenge 01

CS5048 Evolutionary Computation

Mendez Ruiz, Mauricio
A00812794

De Luna Pámanes, Alejandra
A01281001

Sept 25, 2020

1 Introduction

In this challenge we implement a Genetic Algorithm (GA) to find the weights for a neural network with the aim of approximating an unknown function described by the partial data shown in Figure 1. We first present the topology of the neural network to be trained, then we present the GA that will be used and, at the the end, we present the trained neural network and some general conclusions.

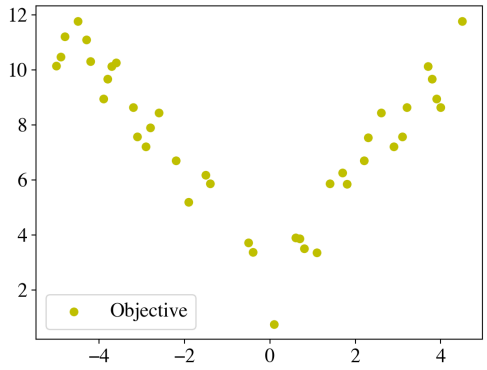


Figure 1: Objective function data points.

2 The neural network

The neural network used in this challenge was provided by Dr. Bayliss. The neural network's topology shown in Figure 2 contains two layers of neurons, in which the first layer consists of 5 neurons and is activated using a log-sigmoid activation function. The second layer consists of 1 neuron and is activated with a linear activation function. The input consist of a single value (x_0), which is connected to the first layer. To evaluate our parameters, the result of using them in the neural network is obtained and compare the given value with the real one by using the sum of squared error function.

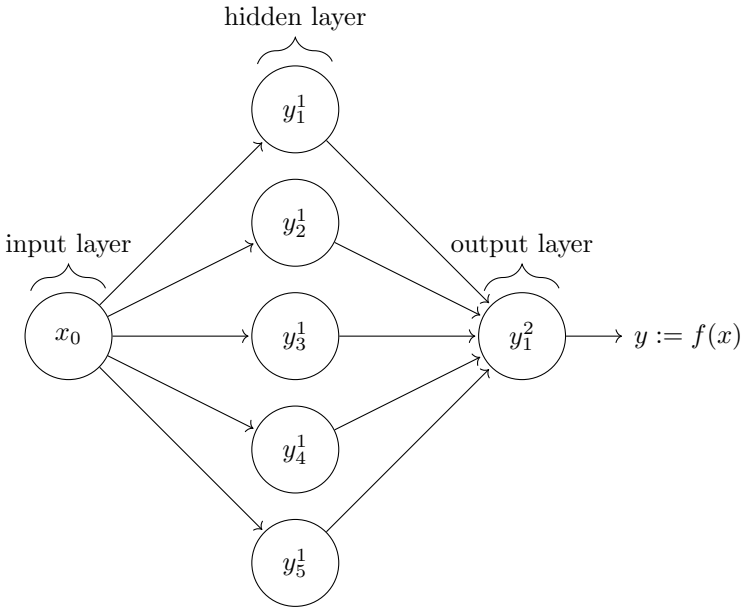


Figure 2: Network topology.

3 The Genetic Algorithm

The GA used in this challenge was also provided by Dr. Bayliss. The GA implemented consists of the evolution of a population containing μ individuals. Each generation μ offspring are produced, so each generation $\mu/2$ pair of parents are selected through a tournament selection of size 3 and undergo crossover with a probability of r_c and mutation with a probability of r_m to produce the μ offspring. Afterward, the offspring population replaces the parent population. The process repeats for t generations and, at the end, the best individual found throughout the evolution process is returned.

The representation of an individual contains the real-valued parameters W^1, b^1, W^2, b^2, since the hidden layer has 5 neurons and the output layer has only 1 neuron. Thus, the individual's genotype has 5 genes representing W_1 weights, 5 genes representing the respective constants b_1, 5 genes representing the weights W_2 and 1 gene for the last respective constant b_2.

W_1^1	W_2^1	W_3^1	W_4^1	W_5^1	b_1^1	b_2^1	b_3^1	b_4^1	b_5^1	W_1^2	W_2^2	W_3^2	W_4^2	W_5^2	b_5^2

Figure 3: Chromosome representation.

4 Results

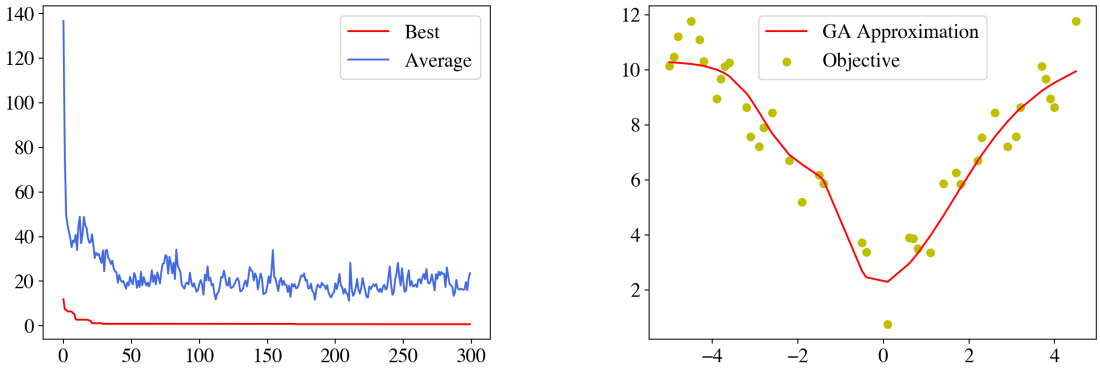
Several trials were done with different parameter values of the GA. For example, we varied the population size, the number of generations, the mutation rate and the crossover rate. The best result of 0.7069 was obtained with a crossover rate of 75%, a mutation rate of 15%, 100 individuals

and 300 generations. The best evaluation found and the corresponding weights are shown in Table 1.

Evaluation	W_1	b_1	W_2	b_2
0.7069	2.56859937	7.22249868	-0.4055281	1.00642419
	0.5340256	-4.07974936	0.55502693	
	3.92051277	3.27150822	-0.53074187	
	1.11815195	-1.83392654	0.87045808	
	-0.34146594	-5.91474613	1.47252551	

Table 1: Evaluation and weights found.

An interesting observation is that the GA found a good solution rapidly in the first 30 generations and only found small improvements afterwards. Furthermore, thereafter the average evaluation of the population did not improve and varied in a not so small range. Figure 4a shows the behavior of the GA throughout the evolution process and Figure 4 shows the objective function data points and the final approximation, where it is visible that the GA succeeded while not over-training the neural network.



(a) Genetic Algorithm run.

(b) Resulting Neural Network approximation.

Figure 4

5 Conclusions

In this challenge, we were able to utilize a GA to approximate the weights of a neural network to approximate the given objective function. The GA successfully found an optimal set of values to model the data. Figure 4b shows that the best solution found found goes through the data points with while minimizing the relative distance. Thus, in this challenge we experimented by changing the GA parameters, though more experimentation could me made by making modifications to the neural network's topology. For instance, increasing the neurons in each layer or adding extra layers to observe if the the trained neural network better models the objective function.