EXPERIMENTING GENETIC ALGORITHM WITH REAL VALUE DATA AND BINARY DATA

ALI HASSAN (S1900077)

1. Introduction

There are many algorithms that can successfully classify given set of inputs into outputs. For this assignment, 3 data sets were provided. It was advised to use any form of evolutionary algorithm to classify and evolve the algorithm and analyze its performance.
I have decided to use genetic algorithm with rule base technique to calculate the fitness of the individuals in the population for the 1st and 2nd data set. However, for the 3rd dataset, a neural network is used classify the inputs and Genetic algorithm is used to find the best parameters to be used on the neural network as its quite different from other two sets of data.

2. Background Research

Genetic algorithms are search algorithms that have been designed to understand and imitate natural system mechanisms: Natural selection and genetics. (Holland, 1975). Genetic Algorithm belongs to category called evolutionary algorithms as it mimics biological process of national evolution and selection of the fittest.

While researching, it was found that genetic algorithms provide solution from the search space but it does not provide an explanation of how it reached to that solution. Therefore, data mining can be used to find the hidden information that is in solution generated from the genetic algorithm.

By definition, data mining is a process of identifying multiple models, summaries, and derived values from a given set of data (Mehmed, 2003).

Data mining has developed into a discipline of information technology and has made significant strides in recent years. The reason for the improvement is because new technology or resources that have been developed which are capable of managing the huge volumes of data generated everyday by consumers.

Genetic Algorithms evolve from a population's initial state. The initial population can be picked randomly or created explicitly using a neural network, allowing for the best outcomes to be chosen and the likelihood of reaching a local minimum or early convergence to be decreased. (Koonce and Tsai, 2000).

Following the creation of the first population, the genetic algorithm calculates the initial population's fitness. The IF-THEN rule may be used to determine the fitness of the produced individual. If the individual's gene or rule is present in the training data, fitness must be increased. Thus, the fitness score will be calculated as fitness divided by the number of regulations.

After calculating fitness, two individuals from the initial population will be chosen. Selection shall

be carried out in accordance with the selection process. There are several selection strategies available, but for this task, only three will be evaluated. Roulette wheel selection, tournament selection, and rank-based selection are the three options.

After that, two parents were chosen. They go through a process called crossover, in which a portion of the parent's DNA are transferred and new offspring are created. Based on the mutation rate, this offspring will be subjected to mutation, and new offspring will be added to the present population. This cycle will continue until the initial population has been replaced. And the cycle will repeat itself till the termination requirements are met.

Once the termination conditions have been met, the best solution will be generated and mined in order to uncover hidden information and retrieve relevant data.

3. Experimentation

Python was chosen as the programming language for the experiments since it made logic easier to apply than other languages.

Python has a number of libraries that can deal with large amounts of data. Pandas is a package with a variety of capabilities ranging from charts to matrix computing.

3.1 Data set 1

The first data set was the smallest of them all. It contained 32 lines, with an input value and an output value separated by a space on each line. After eliminating and merging the input and output, each line included six '1's and '0's, resulting in 32 rules.

The data set was divided into two parts: training and testing. Only half of the data was utilized to train the genetic algorithm.

To produce the initial population, a 192-bit bit string with an initial population of 10 was randomly generated. This option is arbitrary and may be altered to examine the effect of altering the initial population size.

After generating the initial population, the fitness of each individual is determined. Fitness-related function is

$$F = \frac{m}{N} \, x \, 100$$

where m stands for the number of matches and N stands for the total number of rules in the train data set. The result is multiplied by 100 to obtain a percentage value rather than a ratio value.

The fitness is assigned to each individual of the population. For data set 1, roulette wheel selection was used to determine the impact on the performance.

Roulette selection is a conventional selection method. The proportionate reproductive operator selects a string from the mating pool with a probability proportional to fitness. This method involves a linear search across a roulette wheel with slots weighted in proportion to the individual's fitness scores. The goal value is a random fraction of the population's finesses. Population is increased till the objective is attained. Fit individuals are not guaranteed to be picked, but have a better chance. (Wadhwa, 2016). Other selection method that was used is tournament selection. tournament selection entails holding multiple "tournaments" among a few randomly picked individuals (or "chromosomes"). for crossover, the winner of each competition (the one with the best fitness) is chosen.

From the mating pool, two parents are chosen and the cross over operator is used. In the first data set, a single point cross over was assessed. A random cross over point is produced in single point cross over, and two separate parents are divided at the cross over point. To create new offspring, the cut points of two parents are connected. The offspring with the greatest fitness score are given the opportunity to be mutated and then added to the existing population.

The bit flip method was utilized for the mutation operator. Every bit in data set one is flipped between a random value of ‘0' and ‘1'. If it ever happened, this would enhance the chances of escaping out of local minimum. Only if and only if specific requirements are met will the mutation occur. A number between 0 and 1 is set for mutation rate before the process starts. After that, a random number is created for each generation, and if this value is larger than the mutation rate, mutation occurs for every individual produced from cross over.

Every generation follows the same pattern until the termination requirements are met. After 1000 generations, the algorithm is terminated and examined. To illustrate how the algorithm worked, a graph containing the best, worst, and average fitness is drawn.

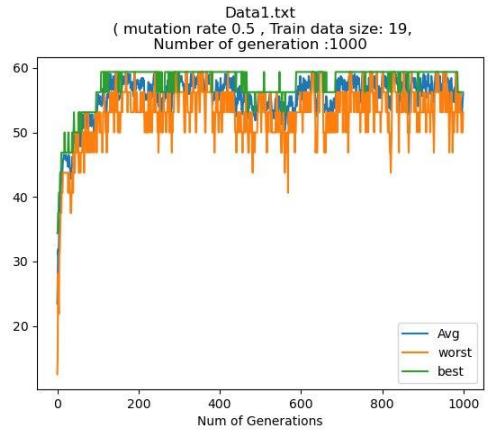

Figure 1 illustrates how the algorithm performs on data set 1 with a mutation rate of 0.5. Out of 32 rules on 19 rules are were used in training. Single point cross over was used. The algorithm was able to achieve a score of 60 percent during first 200 generations. Generations were unable to go past score 60 percent which could be due to local optima that it was unable to escape.

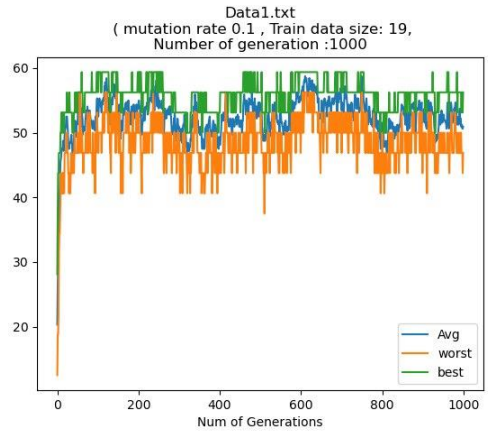

Figure 2 demonstrates how the algorithm fared over 1000 generations with a mutation rate of 0.1 and a training data size of 16. We can observe that the fitness score has not improved. The maximum fitness scores the individual were able to achieve was 60. This could be due to a smaller number of training data being used. A change in the mutation rate or selection method does not make any significant improvement in the genetic algorithm.

3.2 Data set 2

The second data set is distinct from the first. It included more rules and raised the length of each line to seven. Six bits controlled the outcome of dataset 2. There were 64 lines in total. The same techniques were used with data set 2.

In this data set subset of the data is taken for the training. Same rule base approach was used in this data set as well. First random population size of n, was created. The next step was to calculate their fitness. Two parents from the initial population were then selected based on the roulette wheel selection and the performance was compare with the tournament selection method.

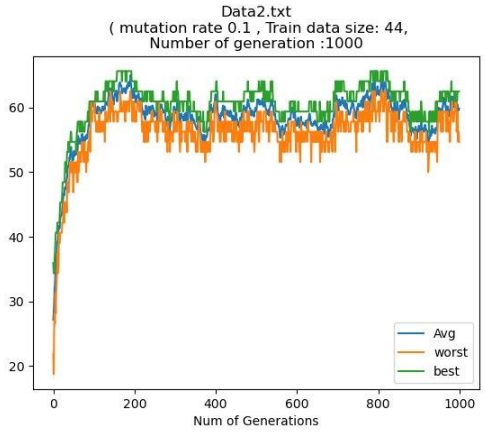

Figure 3 Performance of genetic algorithm when roulette wheel selection was used on the dataset2

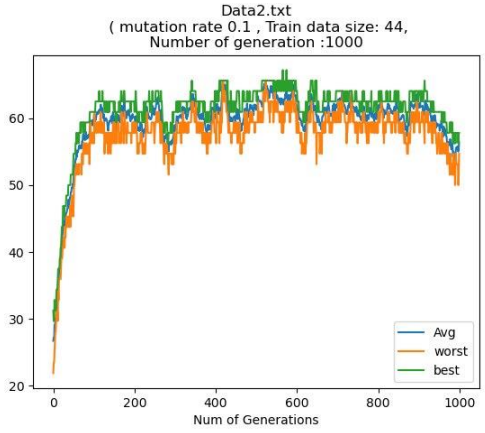

Figure 4 Shows the performance of the genetic algorithm when the tournament selection method was used

As seen in Figures 4 and 5, there is no discernible change in performance. The fitness score remained nearly constant in both approaches. It is clear that when a greater mutation rate was utilized, the fitness score increased significantly quicker in both techniques. And individuals get trapped in local optima between the score of 60 and 70, unable to leave.

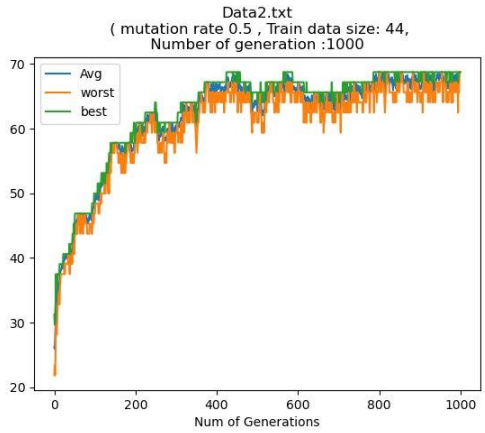

Figure 5 Shows performance of the genetic algorithm when mutation rate was increased with mutation rate of 0.5 with tournament selection method

As we can see from the figure 6 the performance has increased significantly during the experiment. And it was able to leave the local optimum and achieve better fitness score over 1000 generations. This is due the exploratory mechanism when the mutation rate was increase enabling the algorithm to explore and converge.

3.3 Data Set 3

Data set contained real valued data so its quite different from previous two data sets. It contained 6 inputs of floating-point number and one input same as previous data sets. Neural network with two layers was used to train the modal and it achieved good performance. Neural network was experimented with tuning the learning rate to see how it affects the performance. The activation function used in this neural network is sigmoid function. The

Dataset was split into train and test sets before proceeding to train the network. Mean squared error was a loss function.

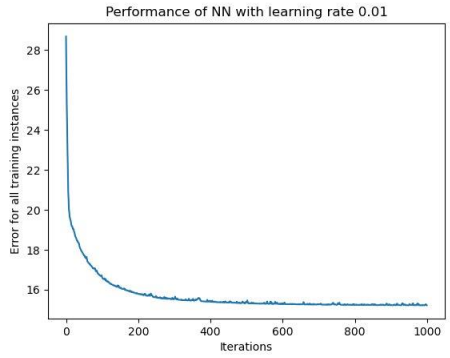

Figure 6 Shows how the neural network performed with learning rate of 0.01

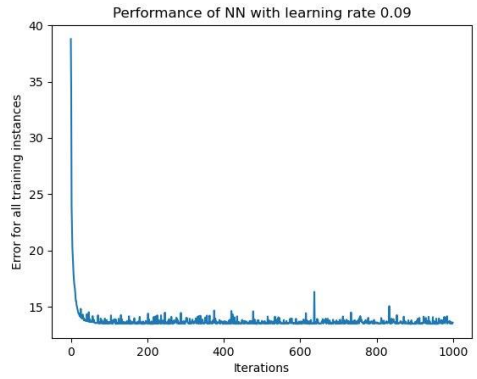

Figure 7 Shows how the neural network performed with the learning rate of 0.09

Since the neural network is very simple as its performance only depends on few layers and the learning rate. As seen on figure 7 and figure 6, we can deduce that lower the learning rate the better the performance of the neural network. I have also figured out that performance can also be increased if a different activation function is used. Also increasing the number of hidden layers would increase the performance.

4. Conclusion

By experimenting with the given datasets, I've learned a great deal of information about the area of data mining. Additionally, the process through which evolutionary algorithms generate random sets of data in order to create a collection of data that it has never seen before is extremely remarkable. Additionally, I gained a great deal of knowledge about neural networks while working on data set 3, which comprised real-valued data. For these studies, I selected Python because it is simple to use and because it is often used in AI and machine learning frameworks.

References

Mehmed, K. (2003) *Data Mining: Concepts, Models, Methods, and Algorithms | IEEE eBooks | IEEE Xplore*. Available at: https://ieeexplore.ieee.org/book/5265979 (Accessed: 23 July 2021).

Holland, J. H., Holland, P. of P. and of E. E. and C. S. J. H. and Holland, S. L. in H. R. M. (1992) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press.

Koonce, D. A. and Tsai, S.-C. (2000) 'Using data mining to find patterns in genetic algorithm solutions to a job shop schedule', *Computers & Industrial Engineering*, 38(3), pp. 361–374. doi: 10.1016/S0360-8352(00)00050-4.

Wadhwa, A. (2016) *Analysis of Selection Techniques in Genetic Algorithm*.

GITHUB: https://github.com/ahorigin/ASSIGNMENT_BIO.git