CLASSIFICATION WITH THE USE OF GENETIC ALGORITHMS

Mariyam Yasmeen
S1800367, BSc (Hons) Computer Science, UFCFY3-15-3, Villa College

CONTENTS

FIGURES

CLASSIFICATION WITH THE USE OF GENETIC ALGORITHMS

Mariyam Yasmeen
S1800367, BSc (Hons) Computer Science, UFCFY3-15-3, Villa College

***Abstract*—This paper presents two implementations of genetic algorithms to solve classification problems for three datasets. Additionally, the parameters are tested in order to identify the ones which result in the highest accuracy.**

1. INTRODUCTION

Data mining is a useful process that helps to solve a variety of real world problems. This report highlights the genetic algorithm methods applied on three datasets in order to determine the logic and parameters that achieve the highest level of accuracy. Dataset one has 32 binary strings made of 5 bits, dataset two has 64 binary strings made of 6 bits and dataset 3 has 2000 floating point value sets of 6 numbers where the value is $0 < x > 1$ and has 6 decimal places. All the datasets have an expected output value of 1 or 0 and the genetic algorithm which is implemented will try to determine this value. The accuracy is calculated on the genetic algorithm's ability to get the expected result.

Terminologies and methods which are common for all three sets will be discussed followed by the logic-gate based method implemented for dataset one and two as well as the neural network method implemented for dataset three.

2. RESEARCH

Data mining helps to identify patterns in fields such as research, marketing and healthcare. The first step to implement the algorithm was determining which method would potentially yield the best results for the provided datasets.

While K-nearest neighbor is an effective method used in data mining to determine patterns and clean up large sets of data to obtain useful information, it does not suit the assignment. Additional evolutionary methods would be needed since KNN is only able to predict a likely result based on 'nearest neighbor'.

It was also important to look at the data itself to determine a suitable method. Dataset one and two are similar in how they both contain binary strings with the differences being the length of the string and the number of values. It is likely that the same method could be used for both.

However, since dataset three has floating values as well as sets of six values where the genetic algorithm needs to determine the expected output, a good solution would be an artificial neural network.

2.1. EVOLUTIONARY COMPUTING

Evolution is the process by which an organism changes based upon its dynamic environment and fitness is a measure by which the organism is able to anticipate these changes. It is an important biological process that allows individuals with better adaptability to have a greater chance of survival and reproduce to create fitter offspring.

In the same way, evolutionary computation allows us to create a population of parents, evaluate their fitness levels and create offspring using a variety of methods where the goal is to create fitter offspring. The concept was introduced in the early 1970s by John Holland where he manipulated binary strings by selecting parents, and performing crossover and mutation (Negnevitsky, 2005).

2.2. GENETIC ALGORITHMS

While there are several ways in which genetic algorithms can be implemented, they all follow some common steps. These steps were implemented for all three datasets. The concepts that apply uniquely to the datasets will be discussed separately.

2.2.1. CHROMOSOME

In the same way that natural evolution depends on the chromosomes of the parent, genetic algorithms have to be provided with data in such a way that it can be manipulated. This is why binary strings are a popular representation of chromosomes in genetic algorithms. The parent chromosome has to be of a fixed length, which is true for all three datasets, and the crossover and mutation rates have to be defined.

2.2.2. FITNESS FUNCTION

The fitness function is the measure by which we determine how fit a parent is. For example, in nature an insect could have better chances of survival if it had better camouflage. In this case, the fitness function would be how well the insect blends into its environment and successful evolution would occur if the insects that stood out became extinct, while the insects that camouflage better could pass their genes forward.

For this assignment, the fitness function is how well the algorithm is able to predict the expected outcome. When a parent is able to determine whether the output is a 1 or 0 accurately, then it has a higher fitness and gets to pass on its genes to the next generation. This could potentially help to increase the fitness levels over multiple generations (iterations).

2.2.3. POPULATION & MATING

A randomly generated population of a set size is required as the 'parents' who would produce the offspring. The fitness function of the parents needs to be determined so that those with lower fitness have a lower chance of mating compared to those with a higher fitness. This helps to ensure that the best genes are passed onto offspring when the selected parent chromosomes mate. The population size has an effect on the algorithm (Roeva, Fidanova and Paprzycki, 2013).

2.2.4. CROSSOVER

Crossover is a genetic operation that allows a pair of chromosomes to mate. For example dataset one has the following binary strings.

Parent A	1	1	0	0	1
Parent B	0	0	1	1	0

Figure 1: Parent chromosomes in dataset one.

A randomly generated single-point crossover implementation allows us to take a part of one parent and a part of the other which results in the new child.

Point	1	2	3	4	5
Parent A	1	1	0	0	1
Parent B	0	0	1	1	0
Child	1	1	0	1	0

Figure 2: Crossover at point 3 resulting in a new child.

If Parent A and Parent B have a high fitness level, then there's a chance that the new child who is different from both parents has an overall higher fitness level.

2.2.5. MUTATION

Mutation is a genetic operation that allows us to potentially increase the fitness level when the chromosomes stagnate at a local optimum. For example, the following child is the result of mating two different chromosomes taken from Dataset one.

Parent A	1	1	0	0	1
Parent B	1	1	1	0	1
Child	1	1	0	0	1

Figure 3: Offspring that results in the same genes as a parent.

However, the child has the same gene sequence as Parent A and therefore this does not help to produce children that change in some form to

potentially achieve better results. Mutation allows us to flip a single gene so that we do not end up at a local optima with no further improvements.

The mutating gene is selected at random and the probability of mutations is usually very low, such as 0.01. For example, mutating the child created in Figure 3 will potentially allow a different parent to enter the population.

Child	**1**	1	0	0	1
Mutated Child	**0**	1	0	0	1

Figure 4: Mutation performed on child

If the mutation has led to a chromosome with a low fitness it will simply get eliminated through the iterations.

2.2.6. GENERATIONS

When offspring are created using crossover and mutation, they become the new parent chromosomes for the next generation. For example, Dataset one is run with a population size of 16 in order to create offspring. The algorithm will continue to create offspring until 16 new parent chromosomes replace the first generation. This will continue until the termination criteria is satisfied. In this case, the termination criteria is either reaching a 100% accuracy rate or the number of iterations defined. The goal of each iteration or run is to end up with chromosomes that are fitter than what we started with.

2.2.7. SELECTION TECHNIQUE

A selection process allows generating a population with fitter parents and has an effect on convergence (Lipowski and Lipowska, 2012).

While there are several selection techniques, the one implemented for the three datasets is roulette wheel selection. This allows parents with higher fitness levels to be given a higher chance to be chosen to mate.

3. DATASET ONE & DATASET TWO

The binary strings for Dataset one and two were put through logic gates in order to determine whether the expected result of 1 or 0 could be determined. The following 8 rules were implemented and led to a highly efficient genetic algorithm that was able to reach 100% accuracy below 3000 iterations.

Rule No.	**Logic Applied**
Rule 1	A AND B
Rule 2	A OR B
Rule 3	A XOR B
Rule 4	NOT A AND B
Rule 5	NOT A OR NOT B
Rule 6	NOT A XOR B
Rule 7	NOT A OR B
Rule 8	A OR NOT B

Figure 5: Logic gates used for dataset one and two

The accuracy is calculated by dividing the number of correct answers with the total where a score of 1 is 100% accuracy

3.1. PARAMETERS

Ensuring that the algorithm is provided with the optimal parameters helps to improve the accuracy of the algorithm as well as the computation time. Therefore, we will test the algorithm for Dataset one and two with a few parameters to identify the optimal values.

The results can vary each time the algorithm is run due to how the generations are affected by randomised values, such as the point of crossover, the genes that mutate, and especially the population that gets selected. Appendix A will include the full sized figures evaluating Dataset one and two.

3.1.1. EFFECTS OF THE MUTATION RATE

Altering the mutation rate can change the accuracy of the algorithm. For example, the following chart shows the best score obtained for Dataset one at 3000 iterations, population of 16, using three mutation rates.

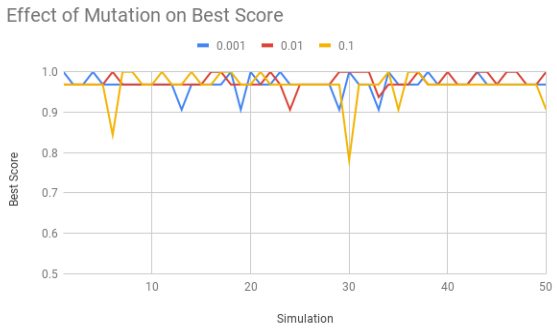

Figure 6: Mutation rates tested for Dataset one.

While the algorithm generally results in an accuracy score of 97% for all three mutation rates, the highest mutation rate of 0.1 leads to the simulations with the lowest accuracy and the mid value of 0.01 has resulted in the most instances where an accuracy rate of 100% is achieved for these 50 simulations.

3.1.2. EFFECTS OF THE POPULATION SIZE

The following graph shows the best score obtained for Dataset 2 when the population size was set to 8, 16 and 32 for 3000 iterations.

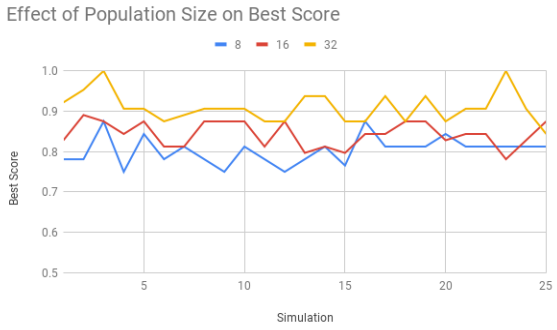

Figure 7: Effect of population size on best score.

Both Dataset one and two perform better when the population size is 50% of the total population. Dataset two is also able to achieve an accuracy of 100% when the population is set to 32 without having to increase the iterations.

3.1.3. EFFECTS OF NUMBER OF ITERATIONS

As long as a local optima is not met, the longer a genetic algorithm can run, the higher its chances for improving fitness over generations. However, computational power or time may limit the number of iterations.

The following graph shows the result of changing the number of iterations for what we have established as optimal parameters for Dataset 2; which is a population size of 32 and mutation rate of 0.01.

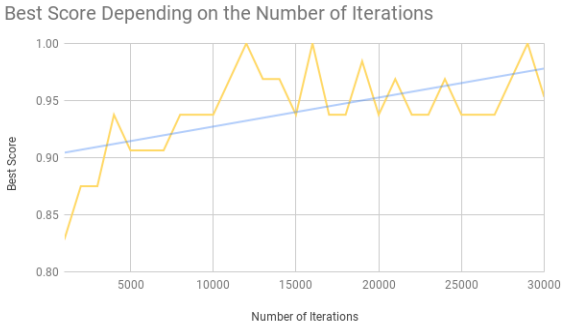

Figure 8: Effect of number of iterations on best score.

The ability for the algorithm to get a better score increases as the number of iterations is raised from 1000 to 10,000. Beyond 10,000, the algorithm is always able to get a best score accuracy rate above 90%.

3.2. RESULT WITH THE HIGHEST BEST SCORE

3.2.1. DATASET ONE

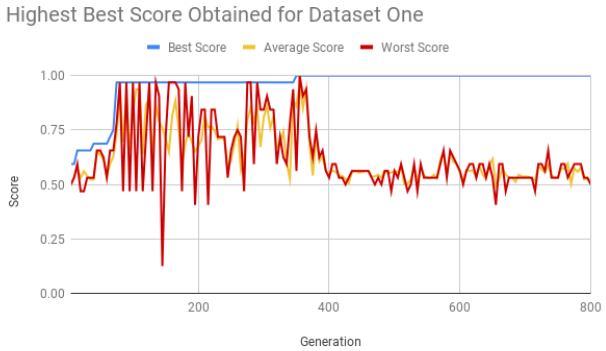

Figure 9: Highest best score obtained for Dataset one.

Dataset one is able to achieve an accuracy of 100% within 350 iterations of the simulation presented in the chart.

3.2.2. DATASET TWO

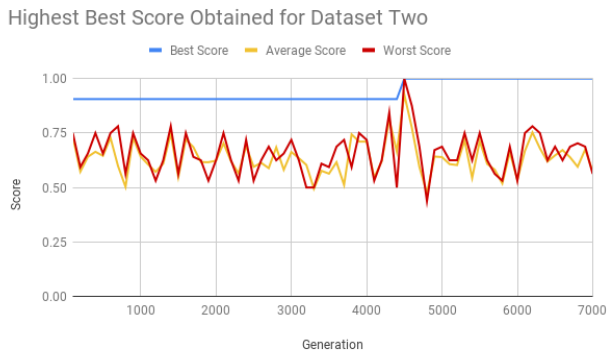

Figure 10: Highest best score obtained for Dataset two.

Dataset two is able to achieve an accuracy of 100% within 4500 iterations of the simulation presented in the chart.

4. DATASET THREE

Since Dataset three is made of floating point value sets of 6 numbers, a feed-forward neural network was implemented. The values would be assigned to 6 input layer nodes, multiplied with randomly generated weights and passed to the hidden layer nodes as the sum of the calculation and the bias. This sum value is passed to the activation function to determine whether the node gets activated (Baheti). Once the calculation reaches the output layer, the higher value of the two will determine whether the result is a 1 or 0.

4.1. SIMPLIFIED NEURAL NETWORK EXAMPLE

As a simplified example, the following diagram will show 3 nodes with values from Dataset three.

The values in the input layer nodes will get multiplied with their respective weights and passed to the hidden layer. The hidden layer node value here would be the sum of the input node values multiplied by their weights plus the bias.

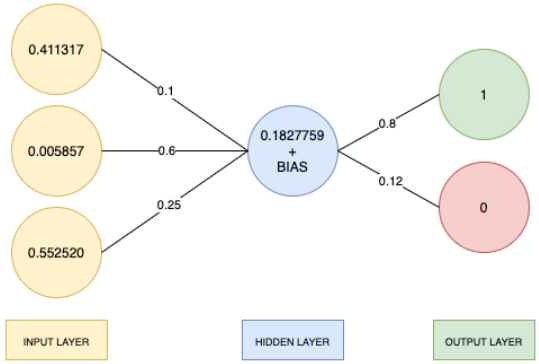

Figure 11: Simplified neural network example

This process repeats from the input layer through the hidden layers until it reaches the output layer. At this point if the value passed to 1 is greater than the value passed to 0, then the output will be 1 or else the output will be 0. The accuracy is calculated by the number of correct answers the neural network got divided by the total number of answers.

4.2. PARAMETERS

The main parameters which have an effect on the accuracy of the neural network include the number of hidden layers, nodes per layer, and mutation rate and magnitude. There are other factors, such as the number of values in the test and train sets, iterations and population size. However, in order to limit the parameter changes, all the tests are carried out on a population of 100 with 400 values in the test set and 1600 values in the training set, upto 5,000 iterations. Appendix B will include the full sized figures evaluating Dataset three.

4.2.1. HIDDEN LAYERS AND NODES

The neural network seems to perform best when it has a single hidden layer. While two hidden layers still results in the neural network getting an accuracy above 58%, a single layer results in the highest accuracy obtained by the neural network which is 65%.

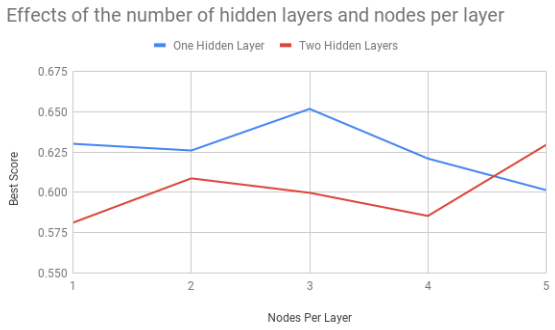

Figure 12: Accuracy based on hidden layers and nodes per layer

4.2.2. MUTATION RATE AND MAGNITUDE

The mutation rate changes one of the 6 values in the set to a random number less than 1 and greater than 0. The mutation magnitude determines the rate of this change.

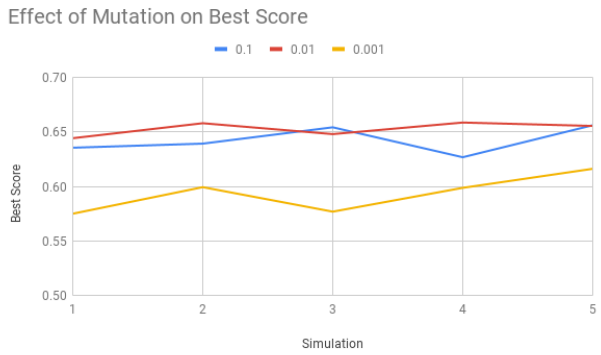

Figure 13: Mutation rates tested for Dataset three.

A mutation and magnitude rate of 0.01 performs the best while 0.001 performs the worst.

4.2.3. ACTIVATION FUNCTION

Different activation functions were also tested to see if there is a difference. The functions tested were Tahn, ReLU and Leaky ReLU (Sharma, 2017).

Tahn and ReLU resulted in similar levels of accuracy and Leaky ReLU resulted in a lower accuracy compared to the other two methods.

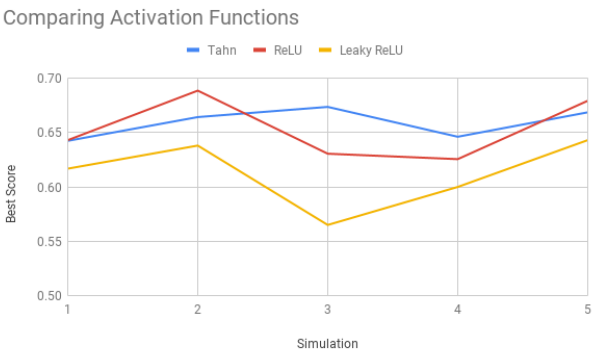

Figure 14: Comparison of Activation Functions

4.3. RESULT WITH THE HIGHEST BEST SCORE

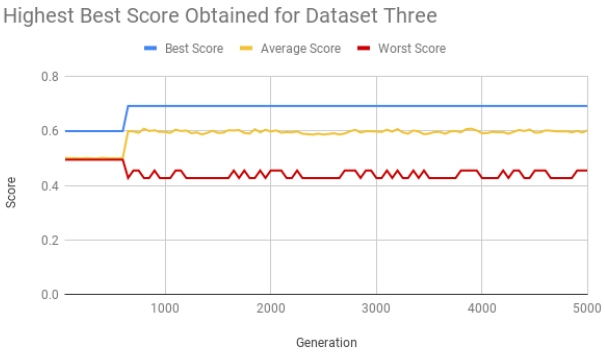

Figure 15: Highest best score obtained for Dataset three.

The neural network is able to very quickly achieve above 60% accuracy with 69% as the highest recorded accuracy. However, methods such as backpropagation could potentially help the neural network to achieve higher accuracy.

5. CONCLUSION

Genetic algorithms with logic-gates are a highly accurate method for binary classification problems presented in Dataset one and two and a neural network shows significant improvement in accuracy when used to classify Dataset three which has floating point values.

The high accuracy obtained for Dataset one and two could be because of the limited data and further testing could help determine its efficiency with larger datasets. Further improvements could also be made to the neural network to achieve higher levels of accuracy.

6. REFERENCES

1. Baheti, P.B. (no date) *12 Types of Neural Networks Activation Functions: How to Choose?* Available from: https://www.v7labs.com/blog/neural-networks-activation-functions [Accessed 01 August 2021].
2. Lipowski, A.L. and Lipowska, D.L. (2012) Roulette-wheel Selection Via Stochastic Acceptance. *Physica A: Statistical Mechanics and Its Applications* [online]. 391 (6), pp. 2193-2196. [Accessed 05 August 2021].
3. Negnevitsky, M.N. (2005) *Artificial Intelligence a Guide to Intelligent Systems*. 2nd ed. Edinburgh Gate, Harlow, Essex Cm20 2je, England: Pearson Education Limited.
4. Roeva, O.R., Fidanova, S.F. and Paprzycki, M.P. (2013) Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling. *Federated Conference on Computer Science and Information Systems*. IEEE [online]. Available from: https://ieeexplore.ieee.org/document/6644027 [Accessed 5 August 2021].
5. Sharma, S.S. (2017) *Activation Functions in Neural Networks*. Available from: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [Accessed 30 July 2021].

7. BIBLIOGRAPHY

1. Durán-rosal, A.M.D., Fernández, J.C.F., Casanova-mateo, C.C., Sanz-justo, J.S., Salcedo-sanz, S.S. and Hervás-martínez, C.H. (2018) Efficient Fog Prediction with Multi-objective Evolutionary Neural Networks. *Applied Soft Computing* [online]. 70, pp. 347-358. [Accessed 25 July 2021].
2. Jiaei, J.H. (2011) *Data mining: concepts and techniques* [online]. 3rd Edition. Massachusetts, USA: Morgan Kaufmann Publishers. [Accessed 12 July 2021].
3. Karimi, H.K. and Yousefi, F.Y. (2012) Application of Artificial Neural Network–genetic Algorithm (Ann–ga) to Correlation of Density in Nanofluids. *Fluid Phase Equilibria* [online]. 336, pp. 79-83. [Accessed 13 July 2021].
4. Mallawaarachchi, V.M. (2017) *Introduction to Genetic Algorithms — Including Example Code*. Available from: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3 [Accessed 10 July 2021].
5. Simoneau, M.J.S and Price, J.P. (1998) *Neural Networks Provide Solutions to Real-World Problems: Powerful new algorithms to explore, classify, and identify patterns in data*. Available from: https://www.mathworks.com/company/newsletters/articles/neural-networks-provide-solutions-to-real-world-problems-powerful-new-algorithms-to-explore-classify-and-identify-patterns-in-data.html [Accessed 10 July 2021].

APPENDIX A - FIGURES FOR DATASET ONE AND TWO

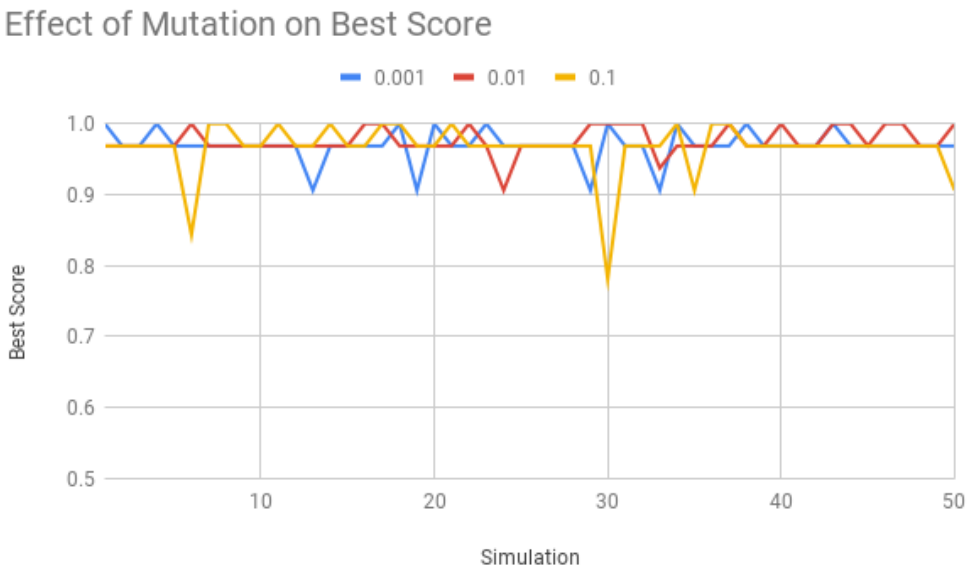

Figure 6: Effect of mutation rate on best score.

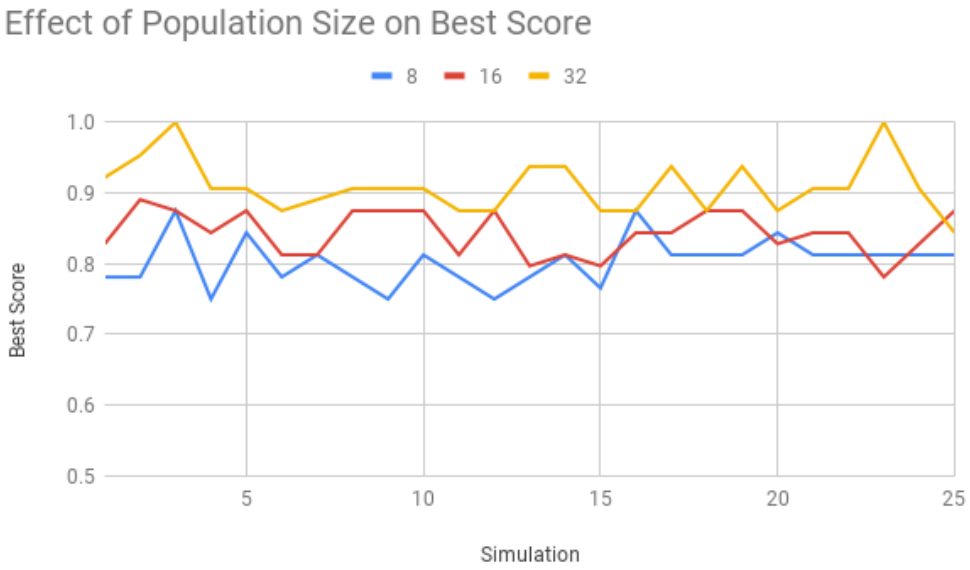

Figure 7: Effect of population size on best score.

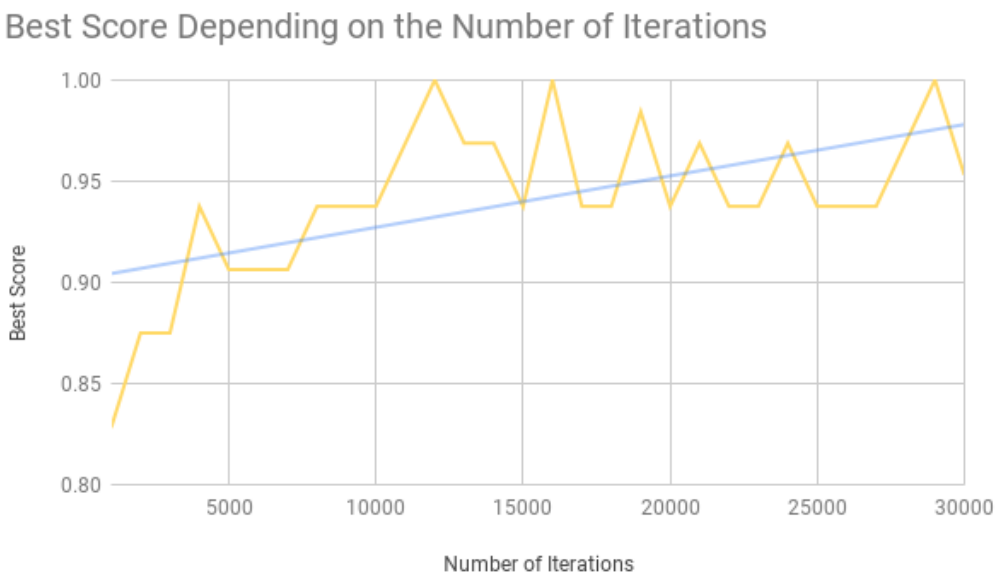

Figure 8: Effect of number of iterations on best score.

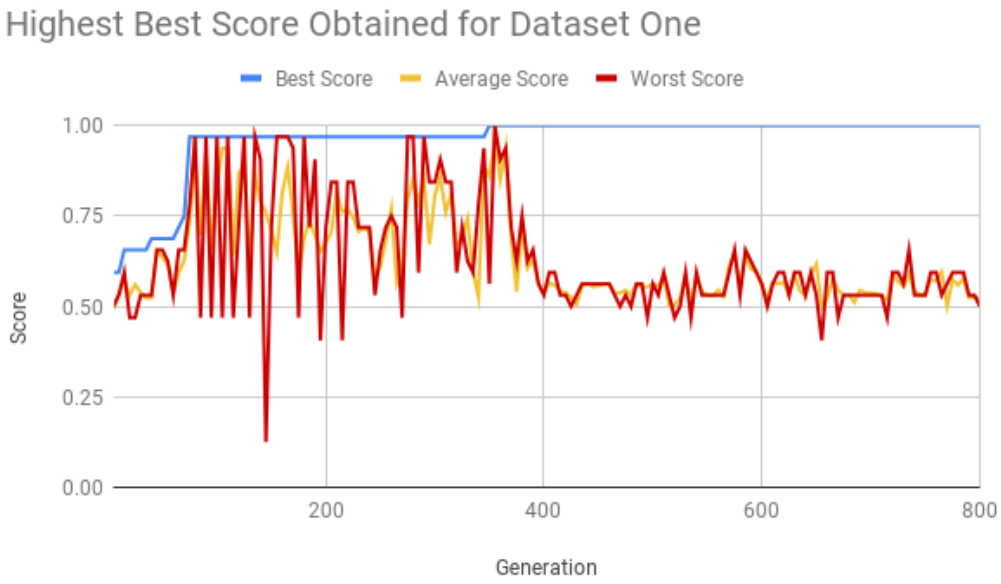

Figure 9: Best result for dataset one

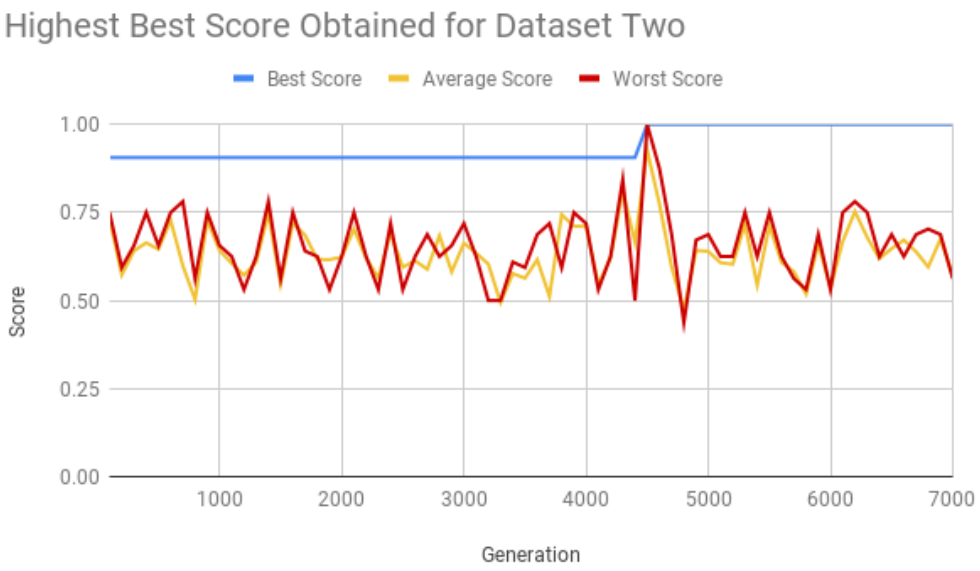

Figure 10: Best result for dataset two

APPENDIX B - FIGURES FOR DATASET THREE

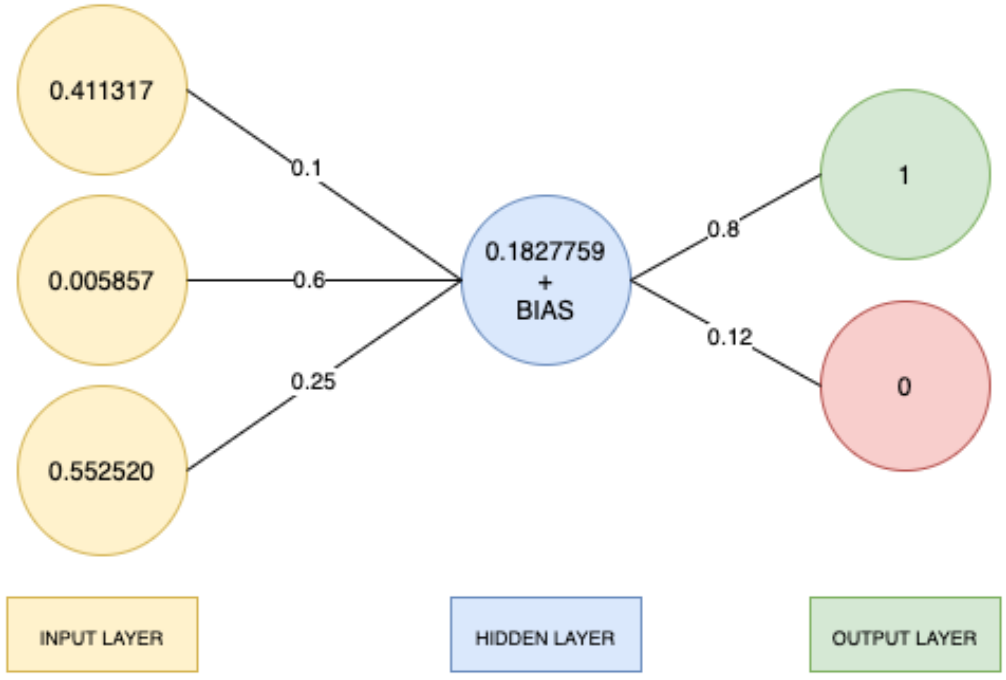

Figure 11: Simplified neural network example

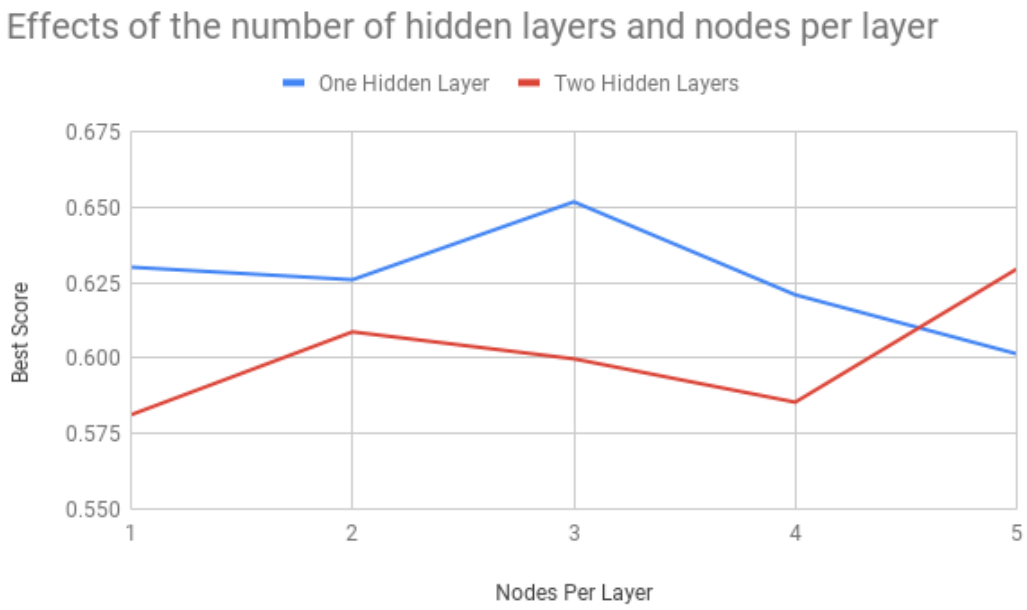

Figure 12: Accuracy based on hidden layers and nodes per layer

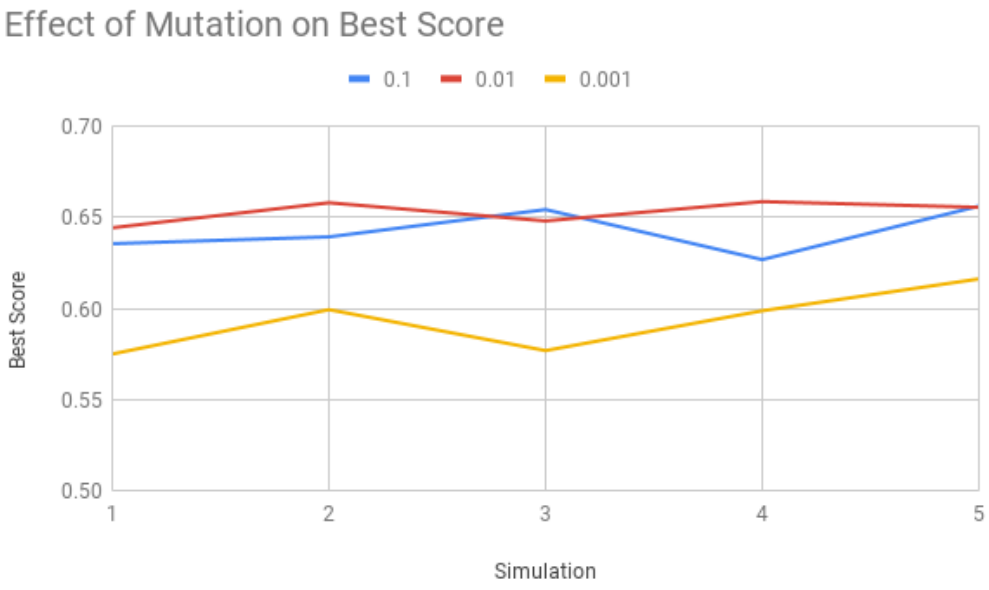

Figure 13: Mutation rates tested for Dataset three.

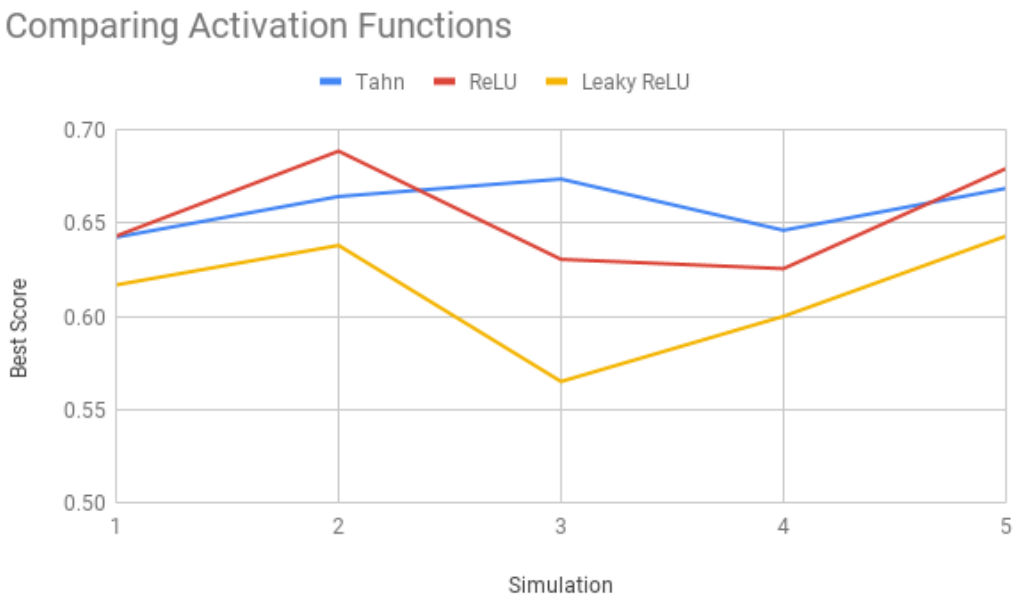

Figure 14: Comparison of Activation Functions

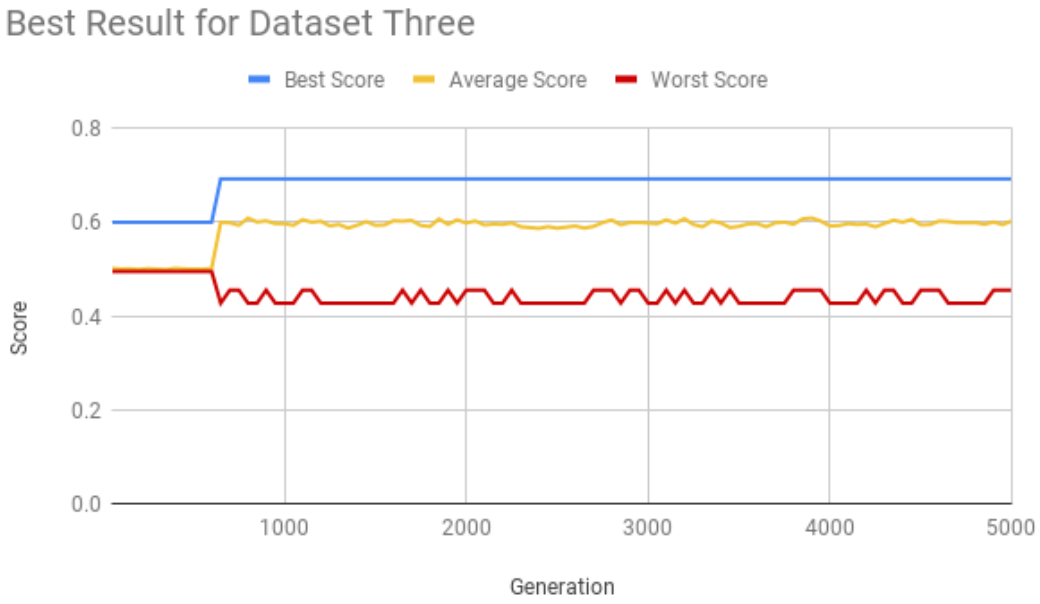

Figure 15: Highest best score obtained for Dataset three.