BSC COMPUTER SCIENCE

BIOCOMPUTATION

STUDENT NO: S1801001
HASSAN ANIM ALI

INTRODUCTION

The purpose of this assignment is to study and analyze three given datasets and evolve a system that can correctly classify the output based on the given inputs using appropriate modern artificial intelligence techniques.

Data set 1 and 2 are binary data sets where all inputs are binary strings with binary output. Data set 3 is a real-valued data set.

I have decided to experiment using Artificial Neural Networks (ANN) and Genetic Algorithms (GA) for this research. GA will be used to classify all data sets, while ANN will only be used to classify data set 3. There is not enough data in data set 1 and 2 to properly train an ANN.

RESEARCH

Data mining is the process of obtaining knowledge from raw data. There may be some data in raw data that is not useful, or not correctly formatted for the requirements. From all the available raw data, it is important that appropriate target data is selected for the process of data mining. After selecting the target data, it can be preprocessed to detect outliers. This is important as some algorithms may not work as expected if there is some data that differs from the entire data distribution. By preprocessing data we can also look out for important missing data. There are algorithms that can be used to estimate what the missing data can be. Preprocessed data can then be normalized and transformed. For example, the real-valued data set 3 in this assignment can be transformed into a binary dataset by applying some rules. Once data is normalized and transformed, classification algorithms such as GA or ANN can be used to identify patterns from the data. Once patterns have been identified by the classification algorithms, it is important to interpret these patterns to extract knowledge from the patterns.

Classification is used to create models, which can assign classes or categories to data items. Models trained using classification algorithms can be used to predict which class a new observation belongs to based on the given input. Data classification includes two steps. Which are:

1. Building the classification model using an algorithm

2. Evaluate the accuracy of the model using test data

A rule-based classification method has been used for this research. In this method, there are IF-THEN rules, where the 'IF' part is a condition that is a conjunction of attributes and the 'THEN' part is the class label. Therefore, the objective is to create a model that can correctly label the data as a particular class, for given conjunction of attributes.

There are many evolutionary computing algorithms available that can be used for classification. They include Ant Colony Optimization, Particle Swarm Optimization, ANN, GA, etc. In this report, I will mainly focus on GA and ANN as these are the two algorithms that will be used to solve the given problem.

GA is an algorithm that turns one population of candidate encodings and their corresponding solutions into another using a number of stochastic operators. Selection exploits information in the current population, concentrating interest on high-fitness solutions. The selection process is biased in favor of the encodings corresponding to better solutions. Crossover and mutation perturb these solutions in an attempt to uncover even better solutions. Mutation does this by introducing new gene values into the population, while crossover allows the recombination of fragments of existing solutions to create new ones (Anthony Brabazon, Michael O'Neill, and Sean McGarraghy, 2015).

Figure 1.0 is a flowchart representing the process of a GA. First, in the initialization process, parameters are defined and an initial random population of chromosomes is generated. The fitness of these chromosomes is calculated based on a fitness function, and highly fit individuals are selected for mating. In the crossover process, genetic information from two parents is combined to create new offsprings. A mutation is applied to new offspring based on a defined mutation rate. The process of selection, crossover, and mutation is continued until termination criteria have been reached.

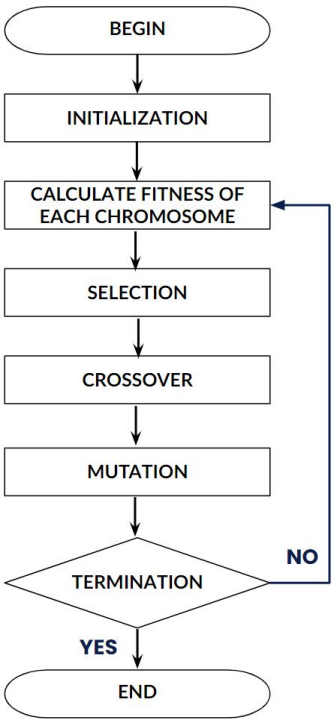

Figure 1.0 Genetic Algorithm Flowchart

Artificial Neural Networks attempts to simulate the process of how the human brain analyzes information. An ANN usually has multiple layers and all the layers are fully connected. Neurons in these layers can receive input and send signals to neurons in other layers based on the input given to them. The edges that connect neurons have weights that are adjusted for the neural network to learn. Each time the network makes a classification, the error in the classification is calculated and this error is then backpropagated to adjust the weights that contributed to the erroneous classification. Adjustment of these weights over multiple epochs results in the neurons in the network learning useful properties of the input. Once the neural network is trained using a training data set, a validation data set is used to test the accuracy of the model. With neural networks, overfitting is possible where the network is too closely modeled to the training data, and the model does not generalize well to other situations. Using a large data set is one of the things that can be done to prevent overfitting. The decision not to use ANN to solve data set 1 and 2 was made because the data sets were too small. A well trained ANN should generalize to new data, and therefore it should be able to correctly classify data it has not seen before.

EXPERIMENTATION

The first two datasets used in this experiment consists of binary data, while dataset 3 consists of real data in the form of floating-point values with an output.

The fitness of the rules generated by the GA will be calculated based on how much data in the dataset can be correctly classified by the generated rules. If an individual can correctly classify 5 rows of data then the fitness will be 5. Wildcards have been introduced to the GA to generalize the rules so that more data can be classified using a single rule. A wildcard is represented as a 2 in the bitstring. 2 can be considered as either a 1 or a 0.

DATA SET 1

In the first dataset, there are 32 rows of data. Each row is a 6-bit bitstring where the first 5 bits are the condition and the last bit is the output. In GA implementation for dataset 1 each rule is represented as a data structure consisting of a 5 element array for conditions, and one integer value for output. Each individual is represented as an array of rules with a fitness value. The fitness function of the GA evaluates the rules by going through the bit string bit by bit. If all the bits in the bitstring of a rule matches with all the bits in a row of data, then the fitness of the individual is increased. An individual's final fitness value is determined only after going through all the rules in the rule array of the individual.

In the GA implementation first, the data from the data file is loaded into the program. Rules of individuals will be compared against this data to calculate their fitness. After loading the data, an initial population of individuals is randomly generated and the fitness of each individual is calculated. Then we loop through all the generations. In each generation, individuals are selected based on either roulette wheel selection or tournament selection. After parents have been selected, a crossover operator is used on the parents. In this implementation only single-point crossover is used. After crossover, if the mutation is enabled, then the mutation is performed based on the mutation rate. Once this step is complete we will

have a new population. The current population is replaced by the newly generated population. Then the fitness of the individuals in the new population is calculated. This process is continued for the defined number of generations.

There are four main parameters that can be adjusted in this GA implementation. They are the number of rules of an individual, the number of individuals in a population, the number of generations, and the mutation rate. Mutation can be disabled by setting the mutation rate to 0.

For each configuration of parameters, ten runs of GA were considered and the average of the ten runs is presented in this document.

For the initial run, I have chosen 10 rules, 10 individuals, 100 generations, and no mutation. These parameters were tested with both roulette wheel selection and tournament selection. For roulette wheel selection average mean fitness was 8.1, while the average mean fitness of tournament selection was 8.5. Therefore, the roulette wheel selection solution can classify 25% of the data while the tournament selection solution can classify 26.3% of the data. Figure 2.0 shows the performance of a run with these parameters using tournament selection. The performance of this run is slightly better than the average as the mean fitness is 10 in figure 2.0.

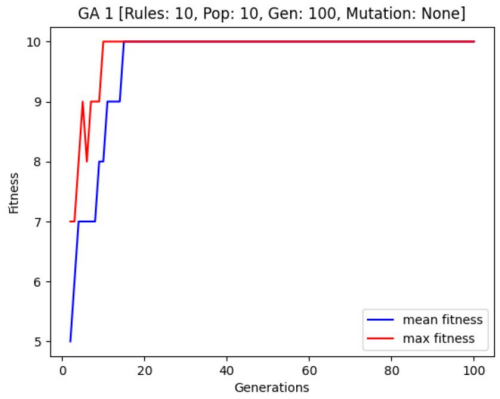

Figure 2.0

For the next run, I adjusted the number of individuals in the population to 50. All other parameters have been left the same. Changing the number of individuals in the population increased the average mean fitness to 10. In the previous run, the fitness of individuals varied between different runs. In this experiment, in 9 out of 10 runs the mean fitness was 10. This implies that changing population size has made the performance of GA more consistent. In this run the performance of both roulette wheel selection and tournament selection were identical.

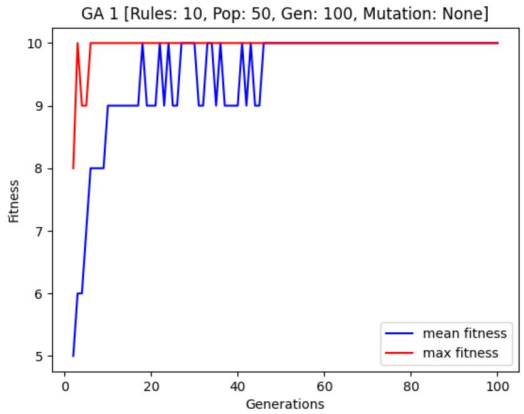

Figure 2.1

For this data set I will not change the number of generations because, from the previous experiments, we can see the algorithm converges around the 50th generation. Moreover, the number of rules will also not be changed as 10 rules should be enough for such a small data set.

From figures 2.0 and 2.1 we can observe that the population is stuck in a local optimum. When all members of the population converge to the same genotype, the population will cease to generate diversity. In order to remedy this issue, we can introduce mutation. The mutation will introduce genetic diversity to the population based on the mutation rate.

For this run, the mutation rate has been changed to 0.02 per gene. All other parameters are the same as the previous run.

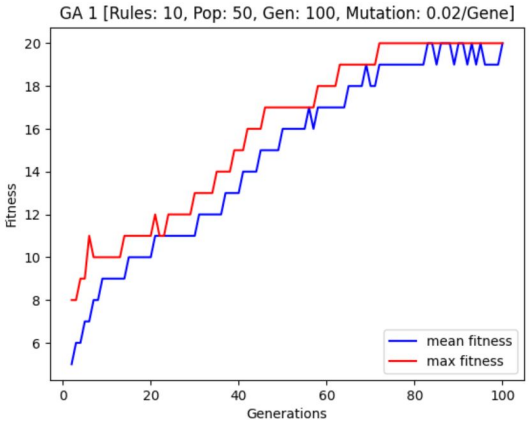

Figure 2.2

As we can see from figure 2.2, introducing the mutation operator has doubled the performance of the GA. The average mean fitness has increased from 10 to 19.6. Therefore 60% of the dataset can be classified using these parameters. I used both roulette wheel selection and tournament selection for this experiment as well. The average mean fitness of roulette wheel selection was 19.2, while the average mean fitness of tournament selection was 19.6. Therefore both of these selection methods are suitable for classifying this data set.

#	CONDITION	OUTPUT
1	11000	1
2	22100	1
3	22220	0
4	01011	0
5	22202	0
6	11112	0
7	02211	1
8	01101	0
9	10021	0
10	12211	1

The above table is one of the solutions found using these parameters. This solution has a fitness of 21 and therefore can classify 65% of the data set. 21 was the highest fitness I was able to achieve using GA.

DATA SET 2

Data set 2 is very similar to the first data set. In data set 2, we have 7-bits in the bit string including the output. So there is one additional bit in the condition or the variables of the rule. In data set 2, there are 64 rows of data. Therefore, there is twice as much data in this data set compared to the first data set. The GA code used for data set 1 will be reused here. The data loading function from the previous code has been changed to load data set 2. Apart from this some global variables and filenames have been changed. All other aspects of the code are the same.

From the previous data set, we have observed that mutation and population size has a positive impact on the performance of GA. Therefore, for data set 2, we will start with mutation enabled. For the first experiment, I have chosen 10 rules, 50 individuals, 100 generations, and a mutation rate of 0.02 per gene.

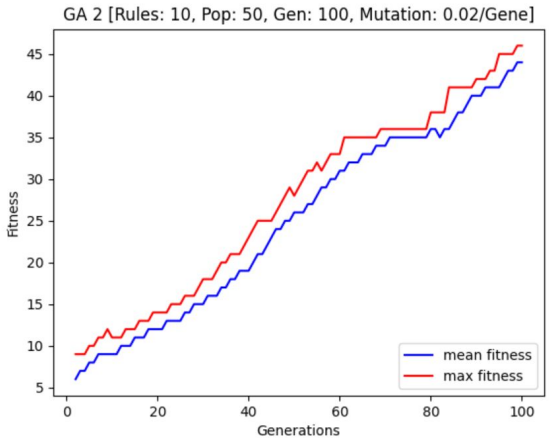

Figure 3.0

As shown in figure 3.0 we were able to achieve a mean fitness of 43, which can correctly classify 66% of the data set. This was achieved using tournament selection. I ran the same experiment using roulette wheel selection as well. Mean fitness dropped to 41 when roulette selection was used. As with the experiments done in data set 1 tournament selection's performance was slightly better. Therefore, we can conclude tournament selection is better than roulette wheel selection. Moving forward all other experiments will use tournament selection. From figure 3.0 we can observe that the fitness of individuals is continuously improving and this trend has not stopped by the 100th generation. Therefore, it is a good idea to increase the number of generations.

For the second experiment I will use the same parameters from the previous experiment, but with the number of generations increased to 200.

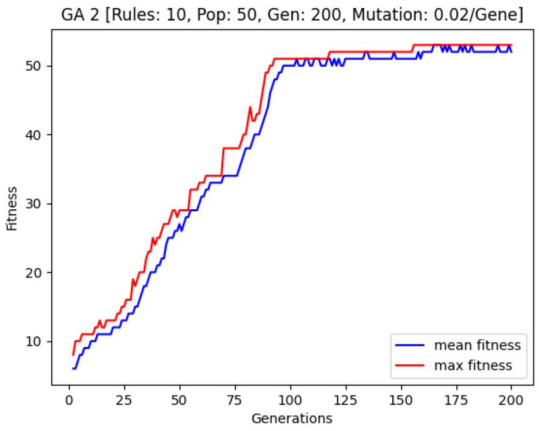

Figure 3.1

Increasing the number of generations resulted in an average mean fitness of 53.4 which can classify 82.9% of the data set. Increasing the generation further won't have a significant effect as we can see from figure 3.1 that the fitness improvement trend has stopped. Therefore, further increasing the number of generations will cost more time without significant improvement in results.

The following table is one of the solutions found using these parameters. This solution has a fitness of 53 and therefore can classify 82% of the data set.

#	CONDITION	OUTPUT
1	120022	0
2	110222	0
3	102122	1
4	222200	0
5	210110	1
6	111221	1
7	212010	1
8	012001	0
9	222222	1
10	200112	1

It is also very likely we can further improve the performance of this GA by increasing the number of rules. For this experiment, I will use the same parameters from the previous experiment, but with the number of rules increased to 15.

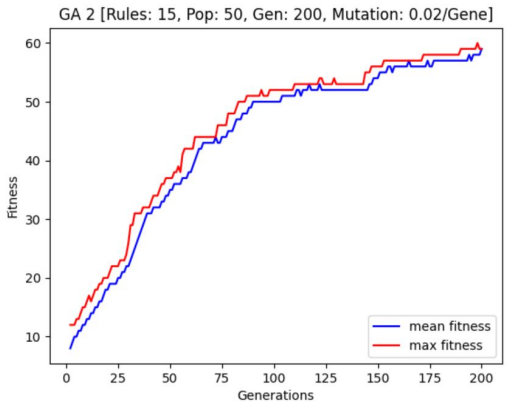

Figure 3.2

As shown in figure 3.2 we were able to achieve an average mean fitness of 58. This can classify 90% of the data set 2. The performance of this GA can now be considered satisfactory.

DATA SET 3

Data set 3 is a real-valued data set containing 2000 rows. Instead of binary values, it contains floating-point values. Each floating-point value is between 0 and 1. Similar to data set 2 there are 6 variables in the condition of the rule. However, as these are floating-point values, they need to be encoded for the dataset to work with existing code. This encoding is achieved by rounding off the variables in the condition. The data loading function from the previous code has been changed to load the new data set and some global variables and filenames have been changed. All other aspects of the code are the same.

As 2000 rows of data are available, it will be possible to train an Artificial Neural Network (ANN) with this data set. ANN requires a large dataset to be trained effectively. There was not enough data in the previous two data sets to effectively train ANN. Therefore, I will be using this technique only in this data set. Data will not be encoded for ANN training and testing. Data will be used as it is. Encoding will only be done for GA experimentation.

I will start the experimentation process with GA. I will use the same parameters last used in data set 2. With these parameters, I was able to achieve an average mean fitness of 1751.3 which can classify 87% of the data set 3. Figure 4.0 shows the results of an experiment using these parameters.

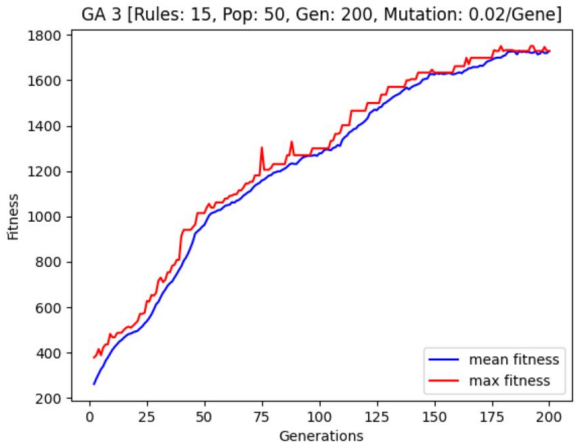

Figure 4.0

The following table is one of the solutions found using these parameters. This solution has a fitness of 2000. Therefore, this solution can correctly classify the full data set.

#	CONDITION	OUTPUT
1	112201	0
2	022021	0
3	222111	1
4	001221	1
5	120202	0
6	012222	1
7	220101	1
8	011020	2
9	110121	0
10	121020	1
11	122011	1
12	102201	0
13	220212	0
14	121120	1
15	022222	0

It is possible to further increase the average mean fitness of the GA. In order to accomplish this, I increased the number of rules to 20 and the number of generations to 250. As shown in figure 4.1 this has improved the performance. The average mean fitness was increased to 1872.8 which can classify 93% of the data set. As with the previous experiment I was able to generate an individual with fitness of 2000. Therefore, we can consider that this problem can be solved by GA.

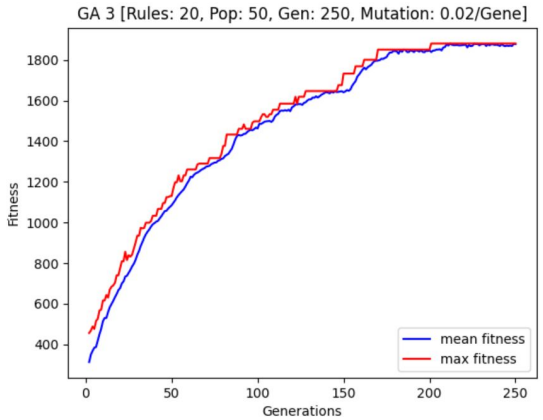

Figure 4.1

As this data set is larger compared to other data set, I will attempt to classify this data set using ANN. For this research, I will use an open-source machine learning library called TensorFlow for creating ANN models. Two-thirds of the data set will be used to train the model, while the remaining data will be used to validate the model. First I will run an experiment with one hidden layer. This layer will contain 10 neurons. I will use three different activation functions to measure their performance. They are Sigmoid, Hyperbolic Tangent (Tanh), and rectified linear unit (ReLU). The output layer of the model will have one neuron and this neuron will use the sigmoid function. The sigmoid function will give a value between 0 and 1 which can be inferred as how confident the model is of the example being in the class. I will be using the 'adam' optimizer function as 'adam' is very widely used and works well in most cases. I will be using the binary cross-entropy loss function as it is intended for use with binary classification where the target values are in the set {0, 1}. This loss function is very suitable in this case as our output is either 0 or 1. I will train the model for 1000 epochs.

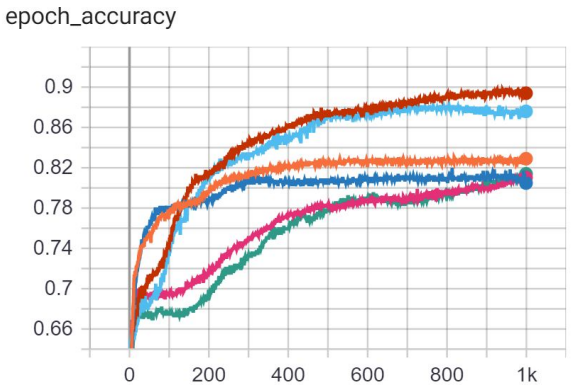

Figure 4.2

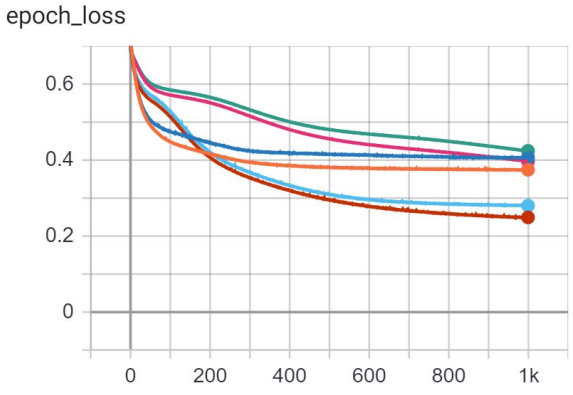

Figure 4.3

Legend for Figure 4.2 and 4.3

ReLU - Train
ReLU - Validation
Tanh - Train
Tanh - Validation
Sigmoid - Train
Sigmoid - Validation

From figures 4.2 and 4.3 we can see that sigmoid function is worst both in terms of accuracy and loss. Tanh has the best performance while ReLU's performance is in between Sigmoid and Tanh. With Tanh function validation loss was 0.28 while validation accuracy is 0.87. It is important to note that loss is more important than accuracy as the neural net attempts to minimize loss rather than attempt to maximize accuracy.

For the next experiment, I will add an extra hidden layer with 10 neurons. This layer will also make use of the Tanh function.

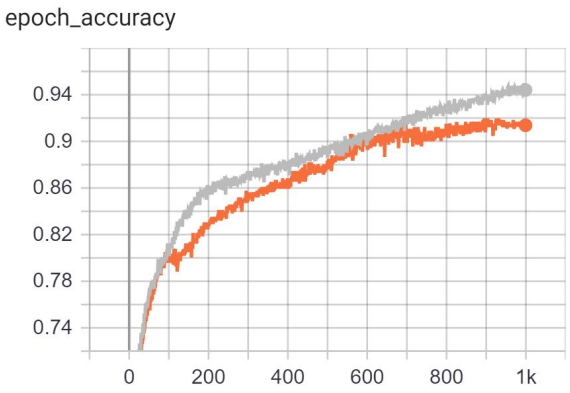

Figure 4.4

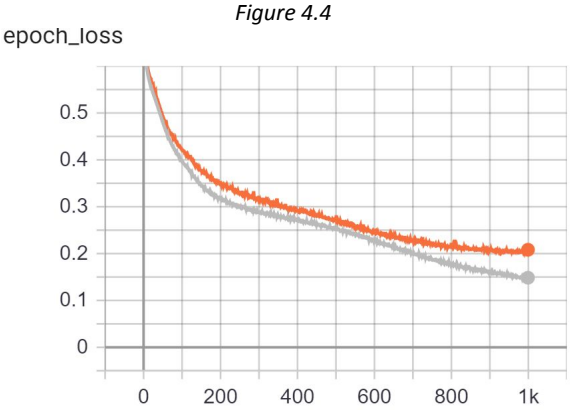

Figure 4.5

Legend for Figure 4.4 and 4.5

Validation
Train

Train accuracy in this experiment was 0.94, while validation accuracy was 0.91. Moreover, train loss was 0.14 while validation accuracy was 0.2. From these values, we know that the performance of the model is very good. However, it is important to check if the model has been overfitted. We can check this by comparing the training performance and validation performance. If there is a great difference between training and validation performance it implies that the model has been overfitted. However, figures 4.4 and 4.5 show that the performance of both training and validation is very close. In the accuracy graph, the difference is 0.03 in the last epoch. In the loss graph, the difference is 0.06 in the last epoch. From the results of this experiment, we can conclude that this model is well generalized and can correctly classify over 90% of data set 3.

CONCLUSION

We attempted to classify 3 data sets in this research paper. For data set 1, the best solution was only able to classify 60% of the data set, while for both data set 2 and 3 the solutions produced were able to correctly classify over 90% of the data set. This implies that the data set 1 was too small and lacked diversity for it to be classified by GA. Alternatively, we could have increased the number of rules in individuals. But this would result in the rules being less generalized. We are not interested in 32 rules that can classify 32 rows of data. We are interested in a few rules that can classify a lot of data.

From this research, we have observed that GA can perform very well when the parameters of the GA are finely tuned. Therefore, we can conclude that GA is very suitable for finding optimized solutions for problems with a large search space. Furthermore, the performance of ANN for data set 3 was also very good. We were able to create a well-generalized model that can classify over 90% of the data in data set 3. It is important to note that coding the ANN was much easier than coding the GA due to the use of the machine learning library TensorFlow.

For future research, both GA and ANN can be combined to produce better results. Furthermore, other evolutionary algorithms such as Ant Colony Optimization or Particle Swarm Optimization can also be used for experimentation.

REFERENCES

1. Appendix. Source code https://github.com/Anim-Ali/Biocomputation.git

2. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. (1996). The KDD process of extracting useful knowledge from volumes of data. Communications of the ACM

3. Anthony Brabazon, Michael O'Neill, and Sean McGarraghy. (2015). Natural Computing Algorithms. Springer Publishing Company, Incorporated

4. Vijini Mallawaarachchi. (2017). Introduction to Genetic Algorithms https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3#:~:text=A%20genetic%20algorithm%20is%20a,offspring%20of%20the%20next%20generation.

5. Jason Brownlee. (2019). How to Choose Loss Functions When Training Deep Learning Neural Networks https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

6. Stacey Ronaghan. (2018). Deep Learning: Which Loss and Activation Functions should I use? https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

7. Tutorialspoint. (2021). Data Mining - Classification & Prediction https://www.tutorialspoint.com/data_mining/dm_classification_prediction.html

8. Ben Coppin. (2004). Artificial intelligence illuminated. Jones and Bartlett Publishers

9. Michael Negnevitsky. (2011). Artificial Intelligence. Pearson Education UK