Comparison of Backpropagation and Genetic algorithm with Bimodal Distribution Removal on Mark Prediction Dataset

Utkarsh Pandey

Research
School of Computer Science, Australian National University

{u6018954@anu.edu.au}

Abstract. Artificial neural networks ('neural networks') is a network that can be trained and tested to provide predictions about various domains such as stocks, weather, political elections and even tertiary marks. The data quality is an integral part when it comes to the performance of the neural network. Commonly, noisy training sets mar the effectiveness of neural network's ability to predict accurately and thus generalization of the neural network also deteriorates. The Bimodal Distribution Removal (BDR) method was developed to provide better method to remove outliers from the training sets. Now to further optimize the neural network we shall combine Genetic Algorithm with Neural Network to leverage the benefits. Thus, now this research report will seek to implement a Genetic Algorithm Neural Network while (GANN), using BDR to see how the model can be optimized to provide better computational cost and ultimately better mark prediction accuracy. The aim of this report is to investigate and assess GANN to improve the predictions and evaluate its advantages and disadvantages. Further, we will compare the genetic algorithm to the standard back-propagation results to conclude which one is more beneficial.

Keywords: Neural Networks, bimodal distribution removal, mark prediction, back propagation and genetic algorithm neural network.

1 Introduction

For the past decade or two, Machine Learning algorithms have taken inspiration from biological and evolutionary sciences. Similar to neural network's inception, evolutionary computation has been inspired by Charles Darwin' theory of natural evolution to enable algorithms to learn and adapt to perpetually evolving environments [1]

Simple path finding algorithms can be really complicated. Hence, a combination of neural network with Genetic Algorithms (GA) can enhance the predictions of the model. The aim of this research paper is to investigate, analyse, evaluate the advantages and disadvantage of utilizing neural network with GA to predict marks using the mark dataset.

This experiment will use the same neural network and results from the previous research with BDR [2] implementation to assess the improvement of the model using GA. Thus, the network topology, preprocessing of data and normalization implementation will remain the same and other aspects of the experiment will be subject to change.

As this research paper will be predicting final marks using assessment results, it is natural to assume that there is ought to be perpetual change in the way assessment scores influence the finals mark. Leveraging the GA advantages of selection, mutation and crossover will remarkably assist the model in reaching the most optimal models of chromosome to attain the best accuracy. At this stage, this may seem a heap lot of biological and algorithmic jargon, but they concepts will be clarified later in the research paper before results are shared.

This problem of predicting final marks based on a collection of assessment is analogous to using GA to predict stock markets indexes using GA. Academics and professionals from various industries dedicate numerous hours into data analysis to understand the trends and predict the occurrence of an event(s) happening.

Though traditional methods of analyzing data can be beneficial, it is highly time consuming and expensive. Neural networks possess the ability to learn from data and examples and predict the results effectively if the neural network is trained with the right measures to minimize the incorrect output.

The effectiveness of training the neural network to predict accurately is highly dependent on the quality of the data. Commonly, in the real world, data can be prevalent with outliers which are different from the majority of the data, hence can affect the learning of the neural network.

The focus of this research is around a mark sheet dataset which involves numerical and non-numerical data. This research will investigate how various measures such as cross validation, BDR [2] and GA can be applied to improve the performance and the accuracy of the neural network model. The measure of effectiveness of implementing GA will be relative to the current model which achieves an accuracy of 3% to 5% with outliers. The mark prediction dataset has 153 rows of data which is further preprocessed.

This research paper will firstly discuss the method it implemented and the controls it has in place to assess the effectiveness of the neural network with and without GA implementation. The second section will delineate the neural network topology, preprocessing data methodology, normalization of data, cross validation and the experimental set up. Secondly, the focus will be to discuss GA in general and then in context of the problem. Thirdly, the paper will discuss the results and finally conclude by discussing the findings and the future research endeavors.

2 Method

Initially, I designed the network topology, which I endeavored to keep consistent throughout the journey of enhancing the neural network via cross validation, BDR and GA. The approach I took in implementing the neural network involved preprocessing the data, developing an appropriate neural network model for the mark prediction dataset, normalization of data, encoding of the output from the network/training value, training the parameters, testing, implementing GA. The section will discuss the aforementioned implementation, respectively.

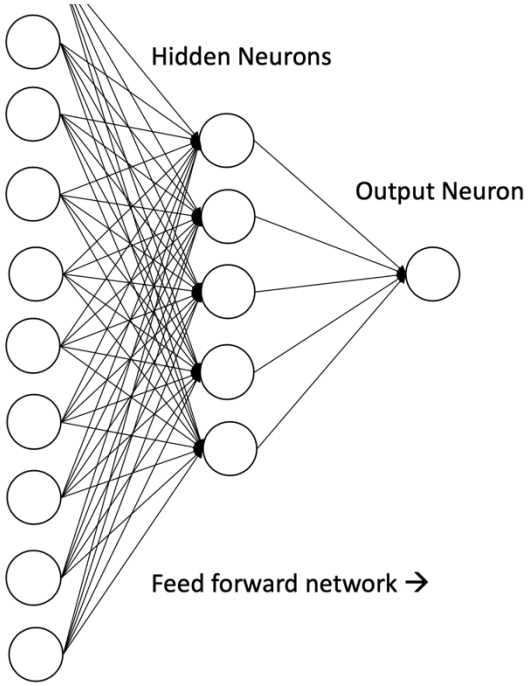
2.1 Neural Network Topology

The topology of the neural network has three layers, which consists of 10 input neurons, 5 hidden neurons and 1 final output neuron. The model was a feed forward network which utilized back propagation. No bidirectional pathways used at any point.

Note, only the RELU activation function was used throughout model. While, various optimizers were tested throughout the experiment.

The Adam optimizer was used in the final solution as it yielded the best results. Pytorch's back propagation properties were also utilized on the neural network in this experiment.

Figure 1: Topology of the neural network



2.2 Preprocessing Data

The original data consisted of 16 columns of numerical and non-numerical data. Firstly, the data was visually assessed and separated into categories of numerical columns and non-numerical columns.

To minimize the complexity of the neural network, only numerical columns data were used in this experiment. We processed the data and dropped columns 'Regno', 'Crse/Prog','S', 'ES', 'Tutgroup' from the original dataset.

Moreover, after assessing just the numerical columns, we noticed values that would potentially affect the neural network model.

For instance, we noticed '.' as values for some students, thus an assumption was made that the '.' would be considered as 0 for that assessment. All the '.' values were converted into 0 for the numerical columns. Further, it was discovered that some students had a zero value for all their assessments including their final score hence we removed all the students from the dataset as well.

After all the preprocessing of the data, the resulting dataset consisted of 11 columns and 144 rows of assessment data. At this stage, features and labels in the dataset were clearly visible. The columns 'lab2', 'tutass', 'lab4', 'h1', 'h2', 'lab7', 'p1', 'f1', 'mid', 'lab10' were designated as features and column 'final' as the label for the neural network. Therefore, 10 columns were identified as features and 1 column was identified as label or target (see Figure 1).

2.3 Normalization

The next stage was to normalize the data according to the weights stipulated in the mark prediction dataset. It was also important to note that assessments only compromised 40% of the total score and the final was 60% of the total score. Hence, the aim was to use assessments of students to predict their final score [3].

The varying weights of the various assessments would remarkably influence the prediction of the neural network as for instance, if it were to learn trends of students achieving better score in higher weighting assessments and ultimately achieve higher final score relatively. It would learn to predict considering the weights of the assessment innately as the normalization of the columns would confine data to a range. All preprocessing and normalization of data was implemented because research has shown that it minimizes their interval difference and can potentially reduce training time [4].

2.4 Experiment Set Up

The PyTorch and sklearn library were used in this experiment to train the neural network. Initially, the dataset was split into 80% of train data and 20% of test data. The K-fold method was used for cross validation. Where, dataset is split into K folds and each fold consists of a test set. Our mark prediction data was split into 3 as we only had 145 rows of data. However, the number of splits was not used as a control and was subject to change if the model did not perform well.

As the aim of the research report was to establish a GA, it is important to follow a principle structure of a GANN system. Initially, the aim was to choose the appropriate size of DNA and population. Noting that the population size had to reflect the search space. Then fitness, selection, crossover and mutation functions were established. The next aim was to encode the best offspring to function in the neural network and make final mark predictions. Thereby, comparing the results of backpropagation and GA on the accuracy of the model [5].

3 Genetic Algorithm

Genetic Algorithms (GA) are search heuristics that evolve according to the situation to improve the prediction of the model. The GA are analogous to processes of natural selection where a certain individual of a population is only able to survive and reproduce if it is fit enough for the situation, as stipulated in Charles Darwin theory of natural evolution. In terms of computational algorithms, after establishing a chromosome representation, there are five major' phases that are considered; initial population, fitness function, selection, crossover and mutation [1, 6].

The natural selection process commences by choosing the fittest individuals from a population. The group of the fittest individuals produce offspring that inherit the characteristics of the parent. The basic logic is that by mating between two fittest individuals will enable the next generation to be fitter as it would have inherited the best of both parents and this trend will continue for the future generations [1].

Organisms of the nature possess characteristics that significantly influence the likelihood of survival and reproduction. The characteristics of organisms are stored in a long chain of information known as a chromosome. These chromosomes consist of long chains of genes, where the genes are inherited from the parent as they establish the anatomy and physiology of an organism. Every individual in a population is ought to have unique chains of genes. In these following sections we will discuss the phases initial population, fitness function, selection, crossover and mutation in more detail [1].

3.1 Initial Population

The initial population is composed of individuals and these individuals may be the candidate solution to the optimization problem one is aiming to solve. These individuals are represented in form of a chromosome, where the chromosome is collection of characteristics of the individual known as genes or allele. The chromosome can be encoded as binary values for classical representation and the genes can be encoded as each bit (variable) of the binary values. The genes are the smallest unit of information and are optimized to produce better accuracy rate as the generations progress. Note representing chromosome as a binary value is a classical representation and other data type may be used to implement GA [1, 6].

One of the prerequisites before implementing GA is to set up an initial population. The classical method to set up an initial population is to generate random values and assign them to each of the genes of all chromosomes. The search spaces need to be reached by the initial population and there needs to be a uniform representation of the entire search space. With regards to computational complexity, the size of initial populations directly influences the diversity of the populations and as generation progress the exploration also improve; hence it can lead to high time complexity. Whereas, small population do not cover the search space and require more generations when compared to large populations [1, 6].

3.2 Fitness function

In Charles Darwin theory, the required fitness of an individual is relative and subjective to the situation. Likewise, in GA the fitness of a certain individual in a population is relative to the desired output of the algorithm. Hence, in GA one has to define a fitness function that acts a threshold to assess whether a certain individual is fit enough to survive and reproduce the next generations and so forth [1].

Meaning, the individuals with the best characteristics have the higher likely hood of surviving and reproducing. In GA, commonly a mathematical function is used to evaluate the quality of the solution by assessing the individual's chromosome. The fitness function is used as an absolute measure of fitness [1, 6].

Moreover, lets investigate the different types of optimization problems that are utilised. The first is unconstrained optimization problem is where the fitness function used to achieve a single objective function. Further, the constrained optimization problem is where fitness function is used to achieve two objectives [1].

The multi-objective optimization problem uses the weighted sum of all sub-objectives in its fitness function. Lastly, the Dynamic optimization problems is where the fitness function is time dependent. Establishing a relevant fitness function is contingent to implementing the following phases of the algorithm; selection, crossover and mutation [1].

3.3 Selection

After the fitness function has assigned fitness scores to all individuals, a judgement is required to select the next generation of individuals according to their scores. These individuals are able survive and reproduce the next generation which are reasonably better than their parents. The selection phase seeks to conduct the 'survival of the fittest' concept and choose a collection of individuals that had better solutions [1, 6].

The selection phase is applied at the end of each generation to select next best candidates. It is important to note that the next generation of candidates are only the offspring, which were produced from the best candidates from the previous generation[1].

During reproduction, crossover of the candidate is conducted, and this will be discussed in the following sub-section. The selection operators make sure the best offspring is produced. There are numerous methods of selection such as selective pressure, random selection, proportional selection, tournament selection, rank-based selection, Boltzmann selection, elitism and hall of fame etc. We will be utilising the random selection in our experiment [1, 6].

3.4 Crossover

During the reproduction of the new offspring, crossover between the pair of parents occurs, where there can one-point crossover, two-point cross overs or uniform crossovers etc. The crossover points are generally selected at random points of the chromosome. Once the crossover between the parent gene is made, the offspring is created and added to the new population of individuals. For instance, consider the following binary representation of the two parents 101101000 and 011000011. If we utilize one-point cross over these two parents to produce two offspring, then their binary representation will be 101000011 and 011101000. The following will depict this reproduction [1, 6].

Parents: 101 | 101000
011| 000011

Children: 101000011 and 011101000

3.5 Mutation

Mutation phase is where some genes are randomly changed, however in most classical cases with low random probability. This enables the population to increase its diversity and prevent premature convergence. Mutation is commonly used with low random probability, so it does not destroy the good genes. Note, mutation is not a critical phase of the GA [1, 6].

In some cases, the probability of mutation changes according to the fitness score of the candidate, the worse the fitness score the more mutation is applied to those candidates and vice versa for candidates with larger fitness scores. An approach that leverages the advantages of mutation is that the mutation probability is initially increased in the early phases of exploration to cover the search space and it gradually falls as candidate produce effective solutions.

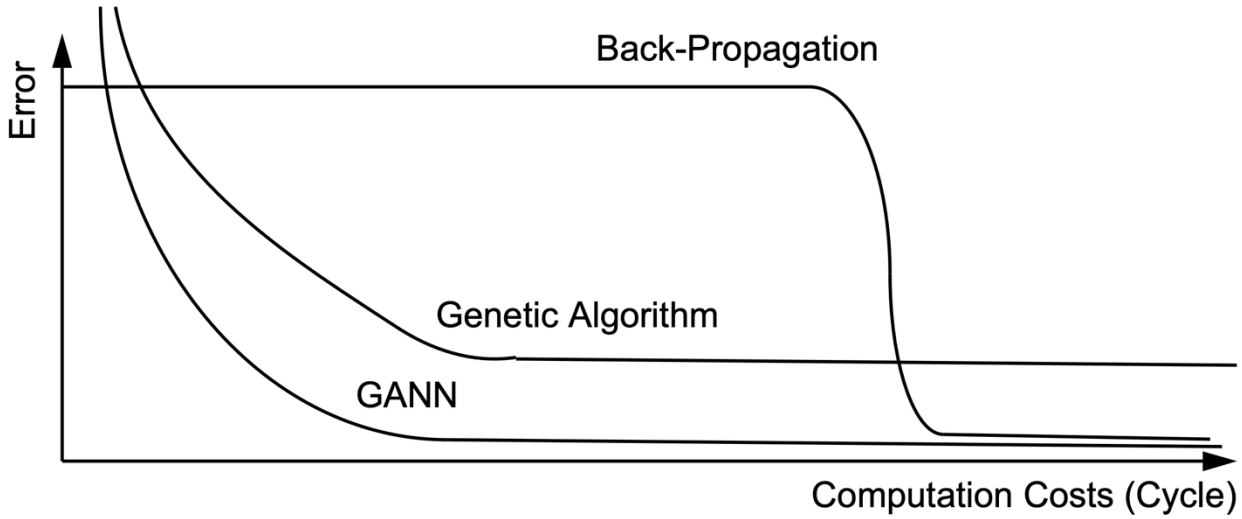
Lastly, it's also important to discuss when the algorithm terminates, the algorithm terminates until the specific stopping condition is satisfied. The most classical method to terminate is to set a limitation on the number of generations a GA can compute. This is the approach I have implemented in my code base. While, it's also contingent to terminate the algorithm once the population has converged. Convergence means that the new generations are only improving in performance by small amounts and the best possible solution has been attained [1, 6].

4 Results and Discussion

Before beginning to investigate the results of our experiment, lets discuss the findings other researchers have made with regards to the implementation of Genetic Algorithm on their neural network. Computational costs can be a major factor in implementing neural networks. Research has found that GA implementation can remarkably reduce the computational cost [5]. This Figure below is a good depiction of this claim (see Figure 2). This graph compares three different learning algorithms; classical backpropagation, GA and Genetic Algorithm Neural Network (GANN).

One is easily able to comprehend the significant advantages of theses implementations against the computation costs. The aim of the research is to implement the GA and see how we can leverage this algorithm to deliver better prediction for the mark prediction dataset. Kitano's thesis paper also discusses that large network is not appropriate for GANN systems as they struggle [5]. This claim has also been supported by other researchers who have implemented GANN [7, 8].

Figure 2: Search speed of different implementations [5]



Before implementing GA, I decided to capture the loss of the back-propagation neural network from previous research where train test split and cross validation methods were implemented to achieve accurate predictions. The Figure 1 depicts the finding of the neural network. This graph shows the loss comparison of k folds before BDR [2] was applied to the neural network.

As we observe that there is a steep fall in the loss for the first fold, however this is not the case for the next 2 folds. Because as we did more training over the folds, the starting point for the loss is relatively lower as the neural network has learnt to predict score better relatively. In Figure 3, as we ran the neural network multiple times, we can observe that after the first fold the loss became more asymptotic towards approximately 0.005.

Figure 2: One run over every 50 epochs

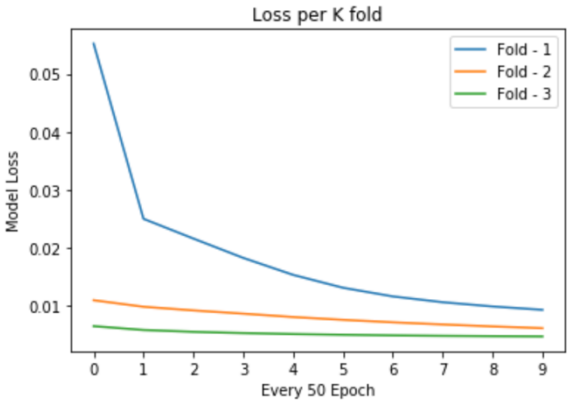
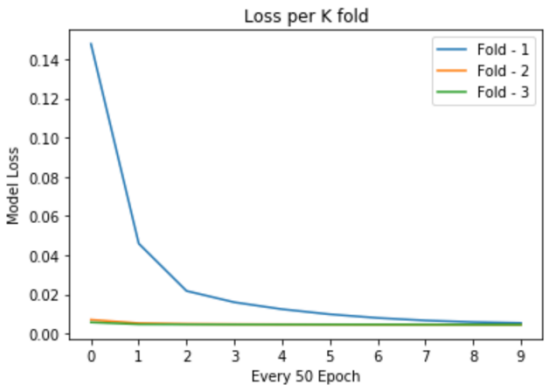


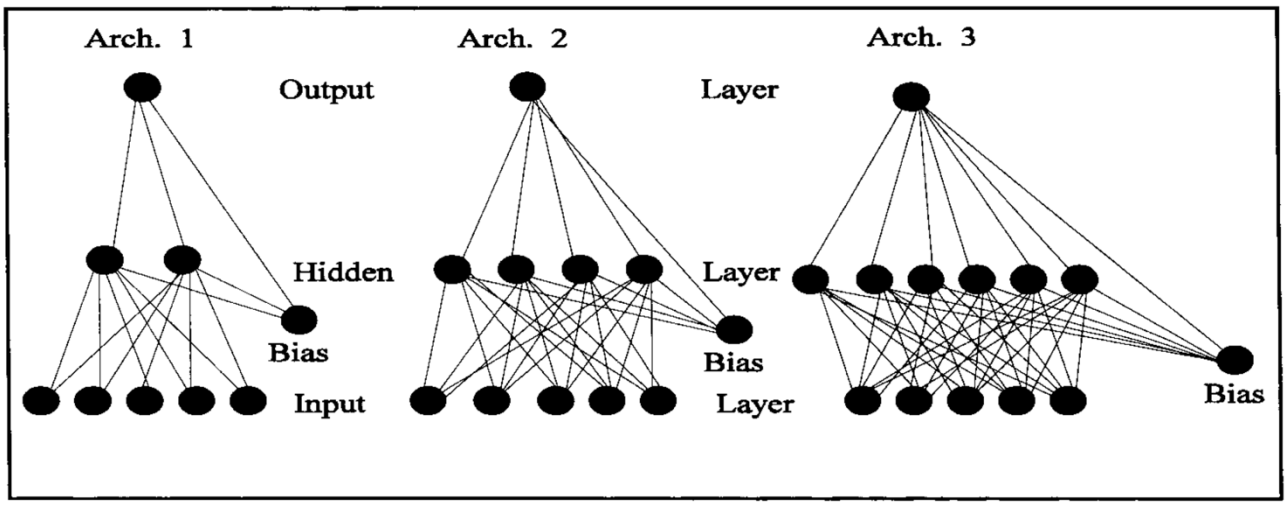
Figure 3: Multiple runs over every 50 epochs.



There have been multiple researchers in recent times that have compared back-propagation with a genetic algorithm for the neural network [9,10,11,12]. Let us discover some methods that researchers have implemented to compare the classical back-propagation approach and the genetic algorithm approach to optimize the neural network.

A research paper conducted comparisons between the two using three different neural network architectures where the neural networks only differed in the number of neurons in the hidden layer (see Figure 4) [9].

Figure 4: The three neural network architectures [9]



These algorithms were compared using the Root Mean Squared Error using in-sample and out-sample test sets. It is evident in Figure 5 that regardless of the number of varying hidden neurons the GA outperformed the back-propagation algorithm with in-sample set and test sets.

Note, in Figure 5 SBP is an abbreviation for Simple Back Propagation where GA is an abbreviation for genetic algorithm. The average RMSE difference between the GA and back-propagation is 0.1784, which is large margin in terms of neural networks.

Figure 5: Average RMSE comparison of backpropagation and GANN [9]

Average RMSE comparisons

Hidden node	6			4			2			
Data sets	SBP	EDBD	GA	SBP	EBDB	GA	SBP	EDBD	GA	LR
In-sample	0.079	0.124	0.041	0.054	0.125	0.043	0.259	0.423	0.145	0.573
Test set 1	0.135	0.183	0.050	0.096	0.174	0.055	0.348	0.478	0.164	0.639
Test set 2	0.166	0.218	0.050	0.147	0.243	0.066	1.108	0.439	0.172	0.640

This research further investigates the mean CPU time to assess if they are other benefits of GA when compared to standard back propagation algorithm. In terms of average CPU time, the GA produced better results across all neural network architectures (see Figure 6). This was always supported in Kitano's thesis paper [5].

Figure 6: RMSE Standard deviation and mean CPU time comparison of SBP and GA [9]

RMSE Standard Deviations and mean CPU time

Hidden nodes	6			4			2		
Algorithm	SBP	EDBD	GA	SBP	EBDB	GA	SBP	EDBD	GA
Std. Dev.	0.21	0.012	0.007	0.024	0.023	0.010	0.077	0.089	0.013
Mean CPU time	611.2	623.8	329.3	570.3	583.2	245.1	497.9	532.1	134.4

Moreover, other researchers have also discovered similar results when comparing back-propagation algorithm to GA [9,10,11]. Sexton and Gupta conducted a research specifically to compare these algorithms and realized that GA outperformed backpropagation on in-sample and test set with different neural network architectures (see Figure 7) [12].

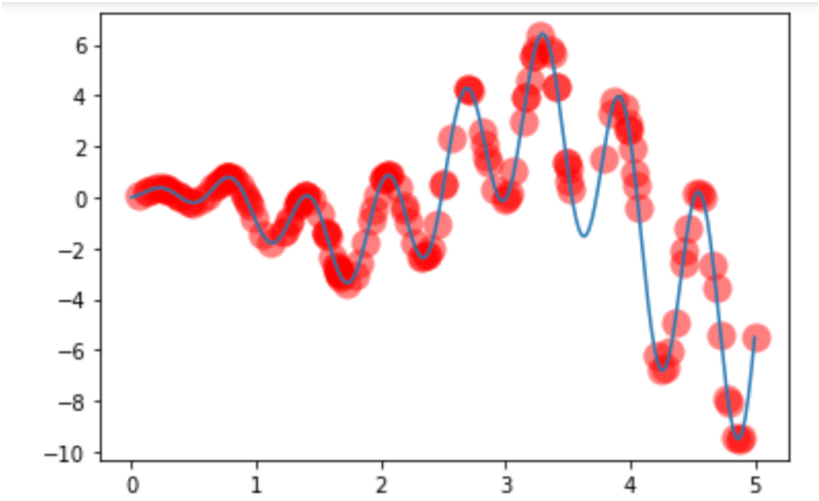
Figure 7: Average RMSE comparison of backpropagation and GANN [12]

In-sample RMSE comparisons

Hidden nodes problem	6		4		2	
	BP	GA	BP	GA	BP	GA
1	0.0174	0.0000	0.0189	0.0000	0.0171	0.0000
2	0.7152	0.0000	0.0281	0.0000	0.7140	0.0000
3	0.0409	0.0000	0.0462	0.0000	0.0381	0.0000
4	0.0033	0.0000	0.0035	0.0000	0.0350	0.0000
5	0.0789	0.0414	0.0538	0.0429	0.2593	0.1451

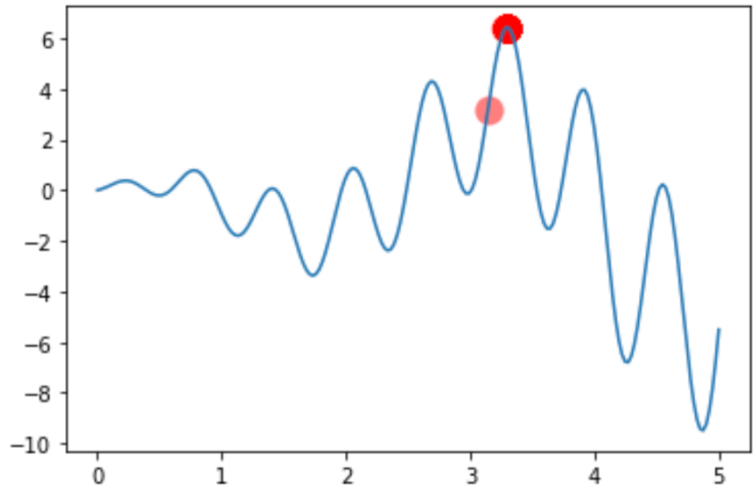
I did not manage to implement the GA to compare back-propagation to it. However, I was able to utilize a researcher's code base to find the most fitted DNA for my population and DNA size [13]. The following two figures delineate the gradual evolution and survival of the best solutions paths (see Figure 8 & 9) A limitation I encountered in this experiment was that I was not able to encode into my problem of final mark predictions.

Figure 8: The First-Generation DNAs




```
Most fitted DNA:  [1 0 1 0 0 1 1 1 1 1]
```

Figure 9: The Last-Generation DNA




```
Most fitted DNA:  [1 0 1 0 1 0 0 0 1 1]
```

5 Conclusion and Future work

It was difficult to adjudicate whether standard backpropagation was better or worse than GA from my experiment, however, many other researchers have found that GA produces better results based on RMSE and CPU time. I plan to investigate more into the implementation of the GA and compare it with back-propagation algorithms with varying neural network architectures in the future. I would also be interested in investing time in implementing hybrid of back propagation and genetic algorithm neural network and discover if that produces better results when compared to back-propagation and GA.

References

1. Engelbrecht, A., 2008. Computational Intelligence. 2nd ed. Hoboken: John Wiley & Sons, Ltd.
2. Slade P., Gedeon T.D. (1993) Bimodal distribution removal. In: Mira J., Cabestany J., Prieto A. (eds) New Trends in Neural Computation. IWANN 1993. Lecture Notes in Computer Science, vol 686. Springer, Berlin, Heidelberg
3. Choi, E.C.Y. & Gedeon, T.D. 1995, "Comparison of extracted rules from multiple networks", IEEE, , pp. 1812.
4. Sola, J., Sevilla, J.: Importance of input data normalization for the application of neural networks to complex

industrial problems. IEEE Transactions on Nuclear Science. Vol.44, pp.1464-1468 (1997)
5. Koehn, P, 1994, 'Combining Genetic Algorithmsand Neural Networks: The Encoding Problem, PhD thesis, The University of Tennessee, Knoxville.
6. Meyer-Baese,A & Schmid, V 2014, Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition),Academic Press, Oxford.
7. Whitley, D., 1991. Genetic reinforcement learning with multilayer neural networks. In Proc. of the 4th Int. Conf. on Genetic Algorithm (pp. 562-569).
8. White, D.W., 1993. GANNET: a genetic algorithm for searching topology and weight spaces in neural network design. The first step in finding a neural network solution.
9. Gupta, J.N. and Sexton, R.S., 1999. Comparing backpropagation with a genetic algorithm for neural network training. Omega, 27(6), pp.679-684.
10. Sexton, R.S., Dorsey, R.E. and Johnson, J.D., 1998. Toward global optimization of neural networks: a comparison of the genetic algorithm and backpropagation. Decision Support Systems, 22(2), pp.171-185.
11. Che, Z.G., Chiang, T.A. and Che, Z.H., 2011. Feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm. International journal of innovative computing, information and control, 7(10), pp.5839-5850.
12. Sexton, R.S. and Gupta, J.N., 2000. Comparative evaluation of genetic algorithm and backpropagation for training neural networks. Information Sciences, 129(1-4), pp.45-59.
13. MorvanZhou 2017, Genetic Algorithm Basic.py, GitHub, viewed 20 May 2020, <https://github.com/MorvanZhou/Evolutionary-Algorithm/blob/master/tutorial-contents/Genetic%20Algorithm/Genetic%20Algorithm%20Basic.py>.