Implementing Genetic algorithm with the bimodal distribution on Comp 1111 dataset

Prateek Arora, u674244@anu.edu.au

Research School of Computer Science, Australian National University

Abstract: A good dataset and fine-tuned hyperparameters are essential as it helps the neural network to learn the relationship between the inputs and outputs easily. So, for refining the hyperparameters which will reproduce great results, genetic algorithm is used. Dataset provided for this project is comp 1111 dataset and it is noisy. So, it is important to remove the outliers from the dataset as it will refine the dataset for the usage. In this project, we are implementing the Bimodal distribution removal algorithm to remove the outliers. Encoding and feature selection is an efficient way for the algorithm to understand the dataset. This report explains that how the genetic algorithm produces great hyperparameters for the model. And how that hyperparameters make us conclude that the BDR algorithm is not an efficient algorithm for this particular dataset.

Keywords: Outlier Detection, BDR (Bimodal Distribution Removal) algorithm, Genetic algorithm, Roulette wheel selection, Encoding, Feature Selection, COMP 1111 dataset, RMSE and variance

1 Introduction

Feedforward neural network is used to make the prediction from the given dataset. It helps the algorithm to learn the relationship between inputs and outputs to draw some inference and make really good predictions. Backpropagation applied in the neural network helps the algorithm to update the weights in the neural network which makes the model efficient. But outliers in the dataset makes the algorithm to overfit mostly. So, Bimodal distribution is used to remove the outliers from the dataset to improve the accuracy as well as decrease the error rate. But, as we know that the dataset is noisy, so using the any hyperparameters could lead to overfitting or could lead inefficient results. So, evolutionary algorithm or genetic algorithm is used to get the retrieve the optimum hyperparameters which provides us with the low RMSE value and improves the model.

This report explains how the genetic algorithm works and how it helps in extracting the refine hyperparameters for the model. And, it also helps in explanation that how the BDR algorithm helps in removing the outliers using those hyperparameters but it worsens the results. The given dataset contains the marks of the students who are enrolled in the COMP 1111 course at UNSW. Regression neural network model is used to predict the final exam marks based on the marks they got in the mid-semester, assignments as well as lab tests. But before the dataset is used for prediction by the neural network, we should do feature selection as well as pre-processing for the good performance.

2 Feature Selection

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in [4]. It is important to consider feature selection as a part of the model selection process because if you don't, you may inadvertently introduce bias into your models which can result in overfitting [6]. The reason for not selecting the Registration number is, because it is unique for every student and we cannot extract any meaningful information from it. The feature named tutgroup and Crse/Prog is chosen because it could be that the program in which the student has enrolled is difficult for them or it could be that the tutor in the tutgroup is either good or bad at teaching which can indirectly affect marks as well as learning of the students. The marks of the students are also added in the training and testing set because the marks are the main key components on which prediction will be made in the neural network. In addition to it, some of the data points where all the values where empty including the tutgroup was removed from the dataset initially.

3 Encoding

The objective of encoding in the dataset is to extract the quality data and useful information from the data so that it can be easily interpreted by the algorithm [1]. Given dataset contains some incomplete rows which are replaced with the 0's using the encoding technique. This makes the dataset complete and almost ready for the neural network. There are some

columns named ‘Tutgroup’ as well as Crse/Prog which has categorical labels. So, for them to be understood by the algorithm, we assign numeric labels using the Label Encoder.

The dataset is converted to numeric as most of the data given is in the form of string. Normalization is the next step towards making the dataset easy for the neural network to understand. In this dataset, normalization is needed because the dataset has values ranging from 0 to 100. So, interpretation of the dataset is difficult for the algorithm. So, to ease it, we normalize each column with its maximum value that could have been achieved. Now, the normalized data is ready to be used for the neural network prediction.

4 Implementation

4.1 Model

The network topology formed in this neural network is that there are 13 inputs with 2 hidden layers, with 15 nodes in the first hidden layer and 10 hidden neurons in the second hidden layer and the output node for that neural network is 1. The reason for choosing these number of hidden layers is because the number of nodes below that makes the model to underfit and neurons more than that makes the model to overfit. MSE error is used as the loss function as it performs best in the regression model. Adam is used as an optimizer in the neural network. The reason for choosing this optimizer is because it requires very less memory as well as it is well suited for the dataset having too many parameters or features [5]. The learning rate of the Adam optimizer is kept to 1e-4 because if the learning rate is increased, the model tends to overfit. After all, the optimizer Adam converges quickly as well as doesn’t generalize much which results in overfitting. The learning rate is not further decreased as the learning rate lower than that doesn’t give good results and at 1e-2 learning rate, the results are optimum. Number of epochs is kept to 500 so that the model can be trained properly.
In this model, dataset is only divided into test and train data. The reason for doing splitting the dataset in that is because there are very few entries in the dataset. So, if we divide the dataset into train, validation and test set, then it is plausible that the training dataset could be less which means model wouldn’t learn much in this case as that data would be small and will not provide good results. By splitting the data only into train and test, it provides more training dataset. RMSE (root mean squared error) is one of the best loss functions for the regression neural network model. It is basically a standard deviation of the errors. The reason for using the RMSE as the loss function is because it is not only one of the most used loss functions, but it also provides more stable and closed solutions.

4.2 BDR Algorithm

BDR algorithm is an algorithm which helps in removal of the outliers from the dataset. It works much better than the least trimmed squared error since BDR produces lower bias and variance in the training data as compared to the least trimmed error [2]. Removing the outliers from the dataset makes the algorithm more robust towards predicting and learning the relationship between the inputs and output. In this algorithm, we are more concerned about the variance of the losses. We have to make sure that the initial normalized variance should be less than 0.1. After that, we can start training the model on the training dataset. Mean error and standard deviation should be calculated for each epoch. From that we can use the formula given below that helps us know to the threshold error

$$\text{Threshold} \geq \mu_{ss} + \alpha\delta_{ss}$$

[2]

Here, μ_{ss} is the mean of the error of the epochs, δ_{ss} is the standard deviation of the error of the epochs and α is the variable that lies between the range of 0 and 1. Over here, the value of α is set as 0.5 in the algorithm.
In the loop, when the error of any data point is greater than the threshold at any epoch, then that particular data point is removed from the dataset as it is considered as an outlier in the dataset. So, this procedure repeats until we get the variance less than 0.01. Now at this point, the training is stopped and then the prediction is made on the testing dataset that we have.

4.3 Genetic Algorithm

Genetic algorithm is a type of evolutionary algorithm which is inspired by Charles Darwin. This algorithm reflects the process of natural selection where the fittest individuals are selected for the reproduction in order to produce the offspring for the next generation [7]. So, over here, our fittest individuals would be ones which have less RMSE value but also making sure that the model is not overfitting. Hyperparameters that are to be tuned in this model is learning rate

of the optimizer, number of epochs, number of first hidden layer and number of second hidden layer. These parameters are fine-tuned because these hyperparameters helps the model to reach the optimum level.

So, after the encoding is done, we now initialize the population for which named “random_initial_population” is made. In this we try to extract some random numbers for the population. So, we take 10 samples as the population size provided to the model is 10. In that function, value is assigned to each hyperparameter in each sample population.

After that, the population went through the fitness function because we want to get the fitness score which will help us in predicting the fit individuals and the help of which, we can extract the parents. So, we now form a fitness function which provides us with the fitness score as the test loss and the training loss. Training loss is also taken in consideration so as to see whether the model overfits or not on those particular parameters. After the step of getting fitness score, we then make a function in which helps in extracting the best fit as the parent so that we can get a good offspring. For extracting the parents, we use the algorithm called “Roulette wheel selection”.

After the parents have been selected using the roulette wheel selection, crossover is performed on them. This paper adopts uniform crossover, as it is probably the most wildly used crossover operator, highly efficient in not only identifying, inheriting and protecting common genes, but also re-combining non-common genes [9], [10]. Simply speaking, in uniform crossover, each gene of an offspring chromosome inherits the associated gene from its two parent chromosomes with a 50% chance. Thanks to the proposed permutation representation, the *i*th genes of all chromosomes share the same set of possible states for link *i*, and therefore, uniform crossover will cause no feasibility problems. Uniform crossover can be viewed as an ultimate multi-point crossover, so, it is much more powerful than one-point crossover. Regarding how to choose two parent chromosomes, any chromosome in an old generation may be chosen as the first parent chromosome at a fixed probability of *p*c, and then a different chromosome may be chosen as the second parent chromosome at a probability proportional to its fitness. In this way, every chromosome stands the same chance to become the first parent, while a fitter chromosome stands a better chance to cross over with most other chromosomes [11].

After getting the child, it goes through mutation phase. In this, bit flipping is done. Random bits are flipped for the child that we got. It is done in controlled way so that it doesn’t change the things drastically. So, if suppose, the learning rate is 0.003 and If I change the first bit, then it is most likely that the learning rate would be greater than one which is not valid. So, to avoid these problems or error, it is done in proper fashion. Only last three bits can be changed randomly for the learning rate, first and second hidden layer.

After the mutation, the child again goes through the fitness function to extract the RMSE value. This is done 15 times for this model as we want to produce 15 generations and select best for the model. Some of the generations are removed from the race because the training error was low than the testing error which means that the model is overfitting. So, to avoid model overfitting, we use the generations which don’t overfit the model. And, then we get the refine hyperparameters which will be best for the model.

4.4 Roulette Wheel selection

So, for this algorithm, we should initially have the fitness scores for different population samples. After that, we find the sum of all the fitness score which is basically RMSE value. Then, we divide each one’s fitness value with the sum that we got which will produces the probability for each chromosomes or samples. Below is given a picture which explains the sample chromosomes taken as well as its probabilities in the pie chart. So, after we have got the RMSE value’s probability, then we turn the wheel keeping the pointer stationary in the random fashion and where the wheel stops and matches the pointer, we select that sample or the chromosome as the parent for the next generation. This algorithm is done twice so as to select two parents which will help in generating new chromosome.

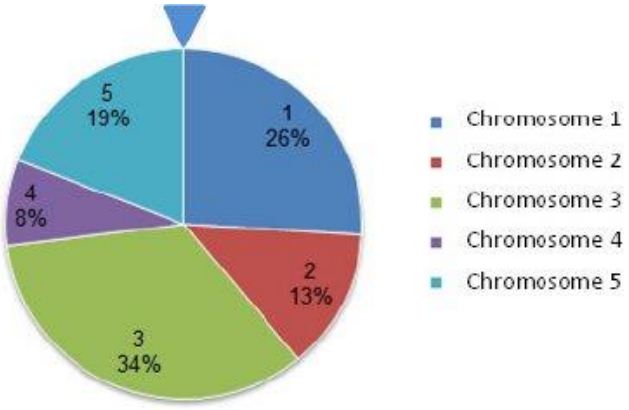

Fig 1 (This picture explains the probability assigned to each chromosomes or sample)

This algorithm could have been problem if improvisation wasn't made for this algorithm for this model. The reason for saying is that this wheel that we got is most likely to stop at the point the area of cross-sectional is more which means that the probability is more. But, if we think that, lower RMSE value means that the model will be really good. So, the fitness value of the best sample would be low as we are using RMSE value. So, occurrence of the best sample would be less which basically fails the purpose of this algorithm. So, to overcome this problem, the probabilities are subtracted from 1. By this, the probability of occurrence of best model is more. Then, it is again normalized so that the sum becomes equal to 1. In this way, problem is solved and the model works perfectly for the RMSE values.

5 Results and Analysis

So, when the Genetic algorithm is finished running, we get the results out of which we have to select the best hyperparameter which gives the least RMSE value. The following table represents the results that we got. It is quite noticeable that the offspring to be generated is 15, but, over here, we jus have 9 off springs displayed in the table. The reason is because the rest of the 6 results were not good as they were overfitting the model. So, they were discarded for this reason.

Learning rate	First Hidden layer	Second Hidden layer	Epochs	Training loss	Testing loss
0.0078125	14	11	251	0.00448	0.00439
0.0126953	14	11	203	0.00456	0.00451
0.0126953	14	11	239	0.00448	0.00437
0.0136718	13	8	239	0.00450	0.00437
0.0078125	13	11	203	0.00448	0.00439
0.0136718	13	8	203	0.00448	0.00432
0.0126953	14	8	233	0.00449	0.00439
0.0126953	13	11	233	0.00457	0.00450
0.0126953	14	8	234	0.00452	0.00438

After getting the results, we have to select the best hyperparameters for the model. So, the hyperparameter that would be chosen is ((0.0136718, 13, 8, 203) , 0.00448 , 0.00432) because it produces the least RMSE error.

We can see that test RMSE loss has been decreased drastically from 0.03078 to 0.00430 which can be derived from the previous assignment. So, we can deduce that the genetic algorithm has helped the model in getting the best hyperparameters which will give the least RMSE value and makes the model optimum. Moreover, the convergence rate is really high in the optimum hyperparameters.

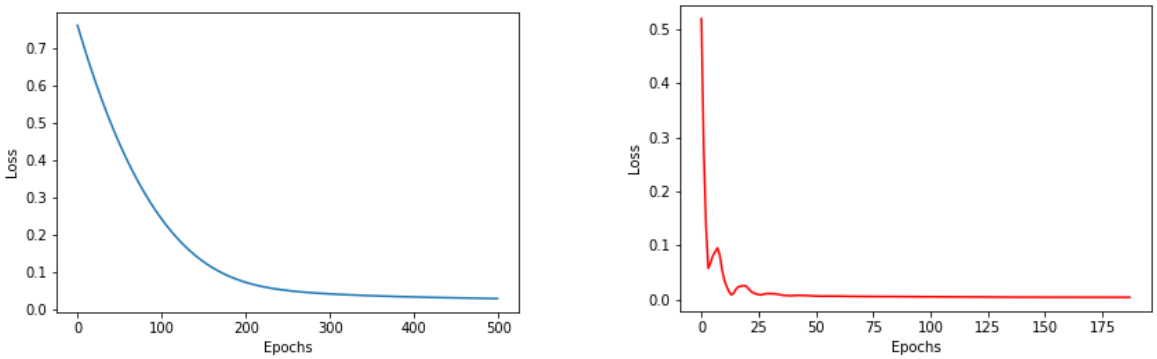

Fig 5.1

(The graph with the blue line represents the loss graph when the model was made using the random hyperparameter selected in assignment1 whereas on the other hand, i.e., the graph which represents red line the graph which is made using the fine hyperparameters which we extracted from the genetic algorithm)

From this graph with blue line, we can clearly see that the model converges at epoch 250, but when the best hyperparameters are used for the model, the model or the loss converges at 25 epochs.
For more details, on how the graph looks like when it converges for the model with optimum hyperparameters, it shown below:

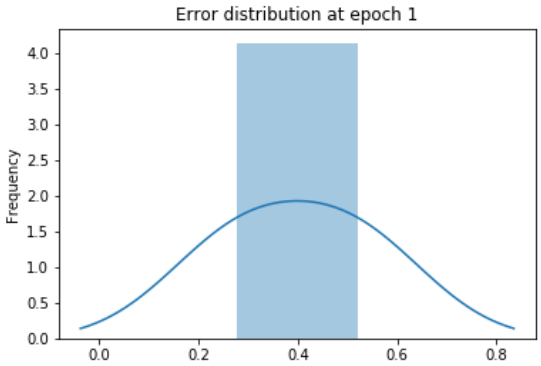

Fig 5.2 (error graph at 1st epoch)

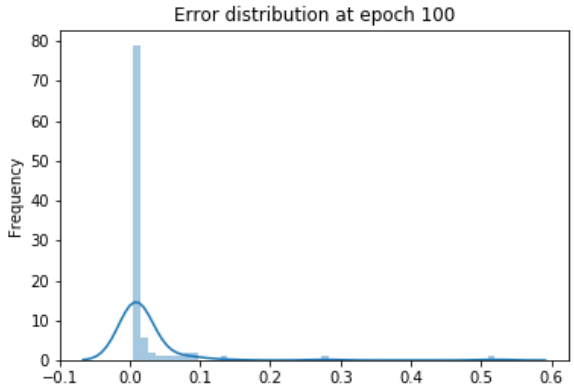

Fig 5.3 (error graph at 100th epoch)

So, as we now have the tuned hyperparameters, then we can try to run the BDR algorithm using these hyperparameters and see that whether these hyperparameters make the results worse or not.
So, when the algorithm runs using these hyperparameters, we observe that at epoch zero, there are many points where error is greater than the threshold. So, that points are considered as outliers. In the below graph explains the error distribution at epoch 0.

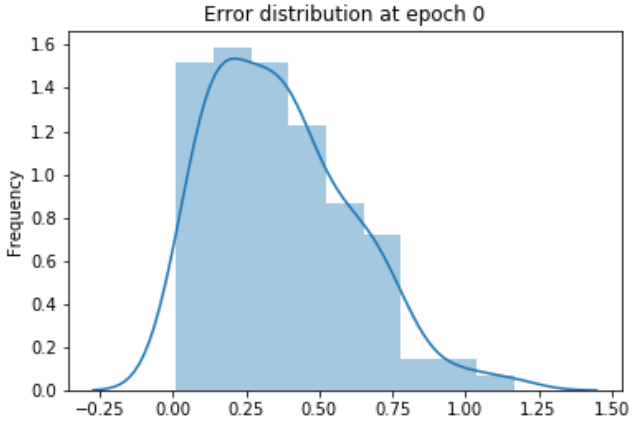

Fig 5.4

(error distribution at epoch 0 when fine hyperparameters obtained from EA is used)

Training error at 0 is 0.11022637039422989
Epoch Number 0
Mean error 0.2719892
Standard deviation error 0.19038974
Threshold error 0.3671840652823448
Number of data points removed from the dataset at 0 epoch is 36

At epoch zero, 36 data points are removed the dataset. So, now only 90 data points are left for the training set and the rest for the testing set. So, after the BDR is applied using the hyperparameters, the loss or the RMSE value is increased to 0.011 which is quite large as compared to the error that we got from only applying EA or genetic algorithm to the model which was 0.00432. This means that the BDR algorithm is not efficient for this dataset. Initially, in the first assignment, it was mentioned that when the BDR algorithm was applied, it didn't affect the model much, but now the model has turned out to be bad. The reason it turned inefficient algorithm is because initially, in the first assignement the hyperparameters chosen for building the model where chosen from hit and trial. But afterwards, we chose the optimum hyperparameters for the model using the genetic algorithm. So, in a way, GA helped in concluding whether the BDR is good algorithm or not.

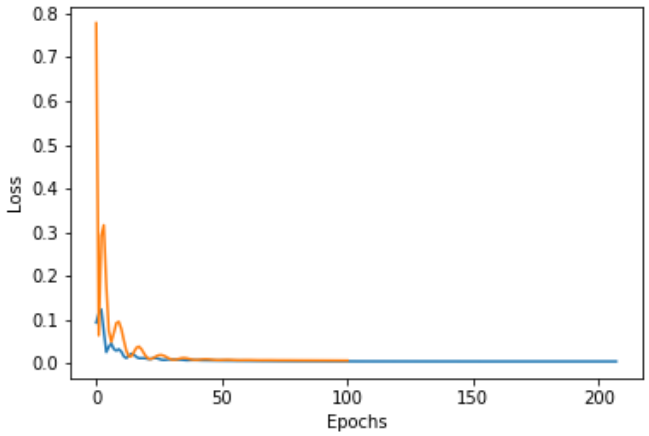

Fig 5.5

(Graph between the error of the feedforward neural network with fine hyperparameters extracted by EA and the error when the BDR algorithm is applied to the training dataset with the same fine hyperparameters)

It is plausible that there are some points in the testing set which are outliers and if we try to remove those outliers also, we might get the good result instead of getting bad results. Initially BDR algorithm is applied on the whole dataset and then the data points which act as an outlier in the data are removed from the whole dataset. When the whole dataset is passed to the algorithm, the following result is obtained.

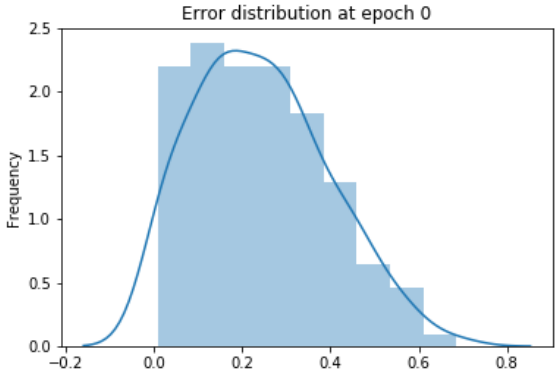

Fig 5.6
(Error distribution at epoch 0, when the BDR is applied to the whole dataset)

Training error at 0 is 0.231109619140625
Epoch Number 0
Mean error 0.38333935
Standard deviation error 0.29010442
Threshold error 0.49938111305236815
Number of data points removed from the dataset at 0 epoch is 50

Now 98 data points are left in the dataset and that dataset is splitted between training data and testing data and is passed through the feed forward neural network and then following results are obtained:

Epoch	Training error
0^{th}	0.29303
50^{th}	0.01126
100^{th}	0.00709
150^{th}	0.00572
200^{th}	0.00462

Test loss: 0.006380

So, over here, we can infer that the BDR algorithm over here also produces inefficient or bad results when BDR algorithm is applied before the train and test split of the data. We can teel this because the error that we got is 0.006 which is quite high than that of the loss (0.0048) that we got when we used normal feed forward neural network.

3. Comparison

From the technique paper diagram and description of the dataset [2] as well as from the paper given with the dataset [3], we can infer that the dataset used for all three papers is same. We cannot compare much of the results as the technique paper as well as the dataset paper have solved the problem using the classification. The neural network regression model is chosen for this dataset because our priority is to predict the marks of the students not the grades as grades can be easily deduced from the system of grade marking provided in the technique paper. In the technique paper, it says that the BDR algorithm is good for the dataset used for the technique paper but BDR algorithm is not efficient for this dataset. In this paper, we are also building evolutionary algorithm or genetic algorithm to get the fine hyperparameters. This is not used in the technique paper or the paper given with the dataset. So, there cannot be much comparison.

Conclusion and Future Work

Choosing the hyperparameter for the model is one of the crucial things. So, genetic algorithm was used to get the optimum hyperparameters which will basically improve the accuracy of the model. When the genetic algorithm was applied, then we got the optimum results. The reason for saying this is because the RMSE value was decreased from 0.03078 to 0.00430 which is a great change. So, this means that the genetic algorithm is great for this dataset. Similar hyperparameters were used for the performing the BDR as well as running the feedforward neural network.

Initially, in the back propagation neural network in the graphs, it showed that there are no 2 peaks in the error graph. So, it is evident that there is no graph in which we can see the 2 peaks and bimodal means that there are two modes in the graph. So, bimodal algorithm shouldn't be applied theoretically. In the first assignment, two peaks were seen because the hyperparameters chosen were extracted from hit and trial. So, that's why it didn't give efficient results. But when, we got the refined hyper-parameter, from the genetic algorithm, it helped us to conclude that the BDR algorithm is not good for this dataset.

We can further extend this project by increasing size of the dataset as the dataset given to us is very small. Moreover, as the dataset given is noisy and BDR doesn't work for this dataset, we should find another algorithm for outlier removal. In addition to it, cross validation can be applied to the model which could further refine the optimum hyperparameters even more.

4. References

1. Dhairya Kumar, Introduction to Data Pre-processing in Machine Learning, Dec 9, 2019
2. P.Slade, T.D.Gedeon , Bimodal Distribution Removal, June 01, 2005
3. Edwin Che Yiu Choi and T.D.Gedeon, Compare the extracted Rules from Multiple Networks, August 06, 2002
4. Raheel Shaikh, Feature selecting techniques in Machine Learning with Python, Oct 28, 2018
5. Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimisation, Dec 22, 2014
6. Jason Brownlee, An introduction to feature selection, Oct 06, 2014

7. Vijini Mallawaarachchi, Introduction to Genetic Algroithms- Including sample code, July 08, 2017
8. Alicia Y.C. Tang, Hybridizing Genetic Algorithm and Record-to-Record Travel Algorithm for Solving Uncapacitated Examination Timetabling Problem, June 2013.
9. G. Sywerda, “Uniform crossover in genetic algorithms”, Proceedings of the 3rd International Conference on Genetic Algorithms, USA, 2-9.
10. E. Falkenauer, “The worth of uniform crossover”, Proceedings of the 1999 Congress on Evolutionary Computation, **1**, 782, CEC99, USA.
11. Xiao Bing Hu, Mark leeson, Evor Hines, Dynamic network coding problem: An Evolutionary approach, October 2009.