Does Bimodal Removal Distribution outlier detection improve training?

Evolutionary Algorithms for hyperparameter selection and comparing BDR vs non-BDR

Shikhar Mishra

Research School of Computer Science, Australian National University

u6203537@anu.edu.au

Abstract. Outlier detection in noisy datasets has been a hot topic of research for decades. The aim is to increase the generalisation during training by reducing the bias in the data. Such approaches generally perform well on artificially generated data, but not as well for real-world data, as it becomes hard to classify between outliers and 'rare' but valid data points. Bimodal Distribution Removal (BDR) is an outlier removal technique that claims to perform well on real world data. This paper evaluates the BDR technique and compares its results with those obtained without outlier removal. The paper uses a back-propagation trained feed-forward neural network in the context of predicting final examination scores of students using grades obtained in smaller assessments during the semester. The dataset used has been thoroughly studied in literature previously, and found to have a significant proportion of outliers. The paper will show that BDR is an effective method for general outlier removal in the training set, as demonstrated with the increase in accuracy of prediction on the test dataset. It was found that the choice of hyperparameters play an important role in the final results obtained. The research is extended by using an evolutionary algorithm for hyperparameter selection. It was found that with these hyperparameters, the accuracy of the neural networks improves by roughly 30%.

Keywords: bimodal distribution removal, BDR, neural networks, outlier detection, mark prediction, evolutionary

1 Introduction

Feed-forward neural networks can be thought of as non-parametric 'universal function approximators' [1], and their approximation properties have been thoroughly proved [2]. In fact, it has been shown that certain mathematical conditions holding, a network with arbitrary depth and neurons can approximate any continuous function [2], which makes them especially valuable for use in regression and classification type tasks.

The non-parametric nature of neural networks helps to explain a lot of their limitations such as the bias and variance dilemma which is well known in the field of machine learning and statistics [1]. Many ways have been proposed to improve the balance between bias and variance, such as pruning, cross-validation, penalisers in loss function, and outlier removal [1]. The paper looks at outlier removal and its role reducing noise in the data, and hence improving variance. The specific outlier removal technique in focus was proposed by Slade and Gedeon in their seminal paper titled Bimodal Distribution Removal (BDR) [3]. Later, we extend the approach using a genetic algorithm for selecting the 'best' hyperparameters of the neural network.

1.1 Dataset

The dataset used in the paper belongs to class marks in an undergraduate course COMP1111 at the University of New South Wales (UNSW). It records the scores students received in all assessments in the course. These assessments included mid-terms, lab exercises and assignments that contribute a total of 40% of their final aggregate grade. This final grade is heavily weighted on their final-exam result (60%), however, in the analysis, the final-exam scores are not considered. The goal is to predict final mark based on the partial marks the student obtains. This dataset was obtained by Gedeon and Bowden, and has been used in numerous studies [3, 4, 5].

Regno	Crse/Prog	S	ES	Tutgroup	lab2	tutass	lab4	h1	h2	lab7	p1	f1	mid	lab10	final
168826	3971	2	FS	T1-yh	2	3	2.5	19.5	.	2.5	11	.	33	2.5	71
168883	3971	2	F	T9-ko	3	5	2	20	17	.	8	.	27	2.4	67
168907	3400	1	F	T6-no	3	3	3	10	.	2	.	.	9.5	2.4	30
169379	3970/1061	2	F	T3-ku	2	3	2	20	19.5	2	15.5	.	21	.	62
169717	3970/1061	2	F	T10-yh	2	3.5	1.5	19	15.5	2	17.5	.	13	2.5	58
170092	3971	2	FS	T1-yh	3	3.5	2.5	20	16	3	16	11	22	2.5	68

Table 1. Snapshot of the raw data

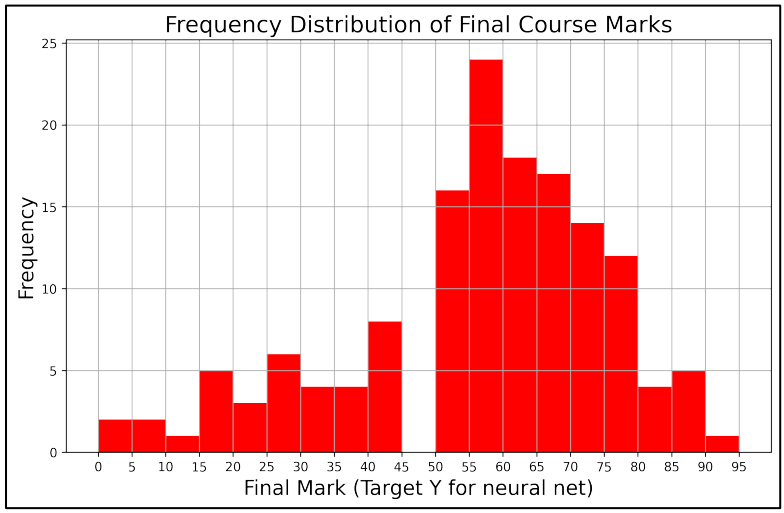

Fig. 1. Histogram of the distribution of final marks, target variable of the neural networks. It peaks in the 55-60 range, with majority between 50 to 80 marks.

2 Methodology and Model Design

2.1 Feed-forward neural networks

This study uses multi-layer feed-forward network which are trained using error backpropagation. The connections between the neurons are made without any multi-layer, backward or lateral type links; only forward connections that go from one layer to the next are made, and similarly, every unit is connected to the previous layer. A simple weighted link is utilised. The data is split into a training set and validation set at random to prevent introducing extra bias; the training set is trained using the input patterns with corresponding output, and the testing is done using a validation set of patterns which were previously unseen by the network. This property of the validation set helps to analyse the generalisation the neural net achieves. It can also indicate prevalence of overfitting if the training error is higher than the testing error.

2.2 Prior Literature on the Data

Choi and Gedeon describe two reasons why there is noise in the chosen dataset; they provide two scenarios in which a student's marks prior to the final grade may not reflect their ability [6].

Firstly, at one extreme is the student who gets great marks during the semester by copying assignments but doesn't understand the course content. This student then performs poorly in the final exam. The other case is of the student who exerts minimal enthusiasm to learn and effort into the course content during the semester and obtains very low marks. Subsequently, this student wishing to get a good grade studies very hard for the final exam and obtains a very good mark. As the final grade is more heavily weighted on the final exam (60%), such a student gets a higher grade than other students in the cohort who exhibited sluggish performance during the semester.

The authors found that these two cases are relatively rare but significant, roughly 10% of the observations. They can be classified as the noise in the data. The aim of the noise removal is to remove these data points.

2.3 Bimodal Distribution Removal (BDR)

There exist many methods that to 'clean' noisy training sets such that generalisation of the network is improved, such as the Absolute Criterion Method, Least Median Squares and Least Trimmed Squares [3]. But they all suffer from a range of drawbacks such as slower convergence time to increasing chances of overfitting. Further this improved performance is only evident in artificially generated noisy data, and not in real-world data where there is a thin line to distinguish between noisy and valid/rare data points [3].

Slade and Gedeon propose the Bimodal Distribution Removal method (BDR) for outlier removal which gives similar level of performance while overcoming the weaknesses of the previous methods. BDR addresses the key weaknesses of other outlier removal methods, and it has several advantages such as improving generalisation, employing a data-driven approach to remove number of patterns rather than hard-coding; giving the network time to learn the data by slowly removing patterns as well as defining a natural halting criterion which significantly reduces training time as well as chances of over-fitting.

The paper also notes that "if the training examples are 'clean' and large in number, normal backpropagation will best approximate the regression E[y|x]". This suggests that the performance of a regression model may be boosted by employing Bimodal Distribution Removal during the pre-processing stage.

Gedeon and Bowden also show that a reduction in the complexity of a training set can improve learning [5]. Their paper shows that a simple heuristic method of reduction of the size of a training set gives a trained network with better performance on the validation test set.

Further, research of Yu [7] concluded that bimodal distribution removal does not benefit network training with a really small dataset for a binary classification problem. However, the study does show that BDR is effective as a general outlier removal tool. This suggests that BDR can still be used to 'clean' a dataset.

The BDR algorithm as described in [3] is implemented. Key steps of the algorithm are summarized in the next section.

2.4 How BDR works?

BDR is based on the empirical observation that frequency distribution of the errors on the training set are roughly normal with large variance early during training. Over time, as 'better' trained weights are obtained, the errors for most of the training data are reduced, but some patterns persist with high training error, which leads to a rough bi-modal error distribution [3]. For visualization of this observation, please refer to figures in Section 3.2. In this bimodal distribution, the peaks corresponding to low errors (towards the left side) are from those data that the network has learnt well, and those with high errors are those corresponding to patterns which are outliers. As the neural network learns **E**(**y**/x), the high errors are outliers because they are not quite what the network is expecting given x. In the middle, there are some patterns that being learnt at a slow rate by the network.

To enable removal of these high data points, an algorithm using the variance of the error distribution is used. It is imperative that data is not removed too fast, as ideally, we want the 'slow coaches' in the middle to be learnt. The key steps are summarized below.

1. Start training normally.
2. Once the network is "trained enough", the variance of the errors becomes low (paper suggest $v_{ts} = 10\%$). BDR technique can be started now.
3. Find the mean of the errors, denoted by $\bar{\delta}_{ts}$. This mean is higher than the low error peak data, but lower than the outliers. Take all errors greater than $\bar{\delta}_{ts}$ into a subset, ss.
4. Find the mean and standard deviation of the errors in this subset, denoted by $\bar{\delta}_{ss}$ and σ_{ss}.
5. The dominant outliers are skewed towards the right in this subset. So we remove points permanently from the training dataset with and error $\geq \bar{\delta}_{ss} + \alpha\sigma_{ss}$ where α is a constant that determines how many standard deviations away does the error have to be in order to be removed.
6. Technically, BDR can continue until there are no patterns in the training set. Hence, a natural stopping condition is used for training. Once v_{ts} is below a constant, like 1% or 0.5%, training is halted.

For more in-depth details on the technique, refer to the Slade and Gedeon technique paper.

2.5 Data Cleaning, Pre-processing, and Importing

The given data is in a semi-structured in a text file. Firstly, the headers were organised in the correct order. Then, comments on the data were removed. This text file was then opened in Excel with delimiter as the space character. Some final cleanup operations were performed and then the file was exported in csv file format.

This csv file was read with all 153 records and stored in a pandas data frame. Some records contained no information on the grades. These were removed from the data, to leave 146 records. Missing columns were imputed with 0 values. The ten feature columns correspond to “lab2”, “tutass”, “lab4’, “h1”, “h2”, “lab7”, “p1”, “f1”, “mid”, and “lab10" marks. These marks were then normalised to between 0 and 1. The MinMaxScaler function from the sklearn preprocessing package was used. The target that is the final marks were converted into percentage.

2.6 Network Topology

The remaining 146 patterns were split randomly into a validation set of 50 patterns and the remaining 96 in the training set. The neural network created has 10 input neurons corresponding to each of the normalised assessment marks. There are two hidden layers with 5 neurons in the default neural net (this is later changed after we find the ‘optimal’ hyperparameters in using a GA). The ReLU activation function is used. Since we are doing a regression problem, our final output layer returns a single output. The Adam optimiser was used with the MSE loss function. The MSE is used because we are doing regression, not classification. The Adam optimiser [8] is used during training, as it provided significant improvements over the Stochastic Gradient Descent optimizer (used in preliminary stages of the work) in the results. Adam is also considered by the neural network academia to be a better optimizer than SGD.

It is interesting to note that due to the small size of the dataset, it had a tendency to overfit very early, so to improve training, a slower learning rate was used. This problem of overfitting on this particular dataset was mentioned in the literature previously in both [5, 3].

2.7 Genetic Algorithms

Genetic Algorithms (GA) are computer-based problem-solving systems based on principles of evolutionary theory, and are one of the more popular forms of the bigger class of Evolutionary Algorithms [9]. They can be thought of as a stochastic search in the global space to find the optimal solution. There are many kinds of EA, but they all follow a similar 7 step algorithm, consisting of selection, recombination and mutation.

The basis of the theory is that all variables in the optimisation can be represented as a bit-string, with each ‘chromosome’ being a possible solution to the problem. Each of these solutions have a corresponding ‘fitness’ value, which is a score assigned to them of how fit the individual is to reproduce. We begin by initialising the population, with the aim to have a diverse gene pool. New generations of offspring are made by selecting from the of gene pool of the older generations, with preference given to the ‘better’ solutions. A genetic crossover is performed to allow for a ‘mixture’ of the solutions and is based on a probability value (usually large). A small value of mutation is done on the resulting string, which adds some random search behaviour in the algorithm. This procedure is repeated until a fixed number of generations is reached. The big idea is that the strongest solutions/individuals survive the optimisation process, and there is optimal balance between exploitation and exploration of the search space. For further details on GA, please see textbook [10] and papers [11, 9].

2.8 Hyperparameter optimisation using GA

A GA was applied for optimising the hyperparameters of the neural networks. The hyperparameters were chosen as the number of hidden neurons in the hidden layers, the total number of epochs of training and the learning rate used with the Adam optimizer. The settings for the GA, and the reasoning are discussed in the next section.

3 Results

3.1 Exploratory Data Analysis and success of Simple Linear Regression

Exploratory data analysis revealed that the assessment marks are appropriate to be used to predict final exam mark. A simple linear regression model using ordinary-least squares and all predictors was fit to the data. The *statsmodel* package in python was used. The adjusted R^2 of the model was 85.1%, implying that most of the variance in the data can be explained and captured using all of the predictors. Summary output of the regression can be seen in Appendix.

3.2 Evidence of Bimodal and Normal Distribution at Stages

It can be seen that during early stages of training, the error distribution is normal and later becomes bimodal. This verifies the empirical claims from the theory [3]. These results are taken from previous work in Assignment 1.

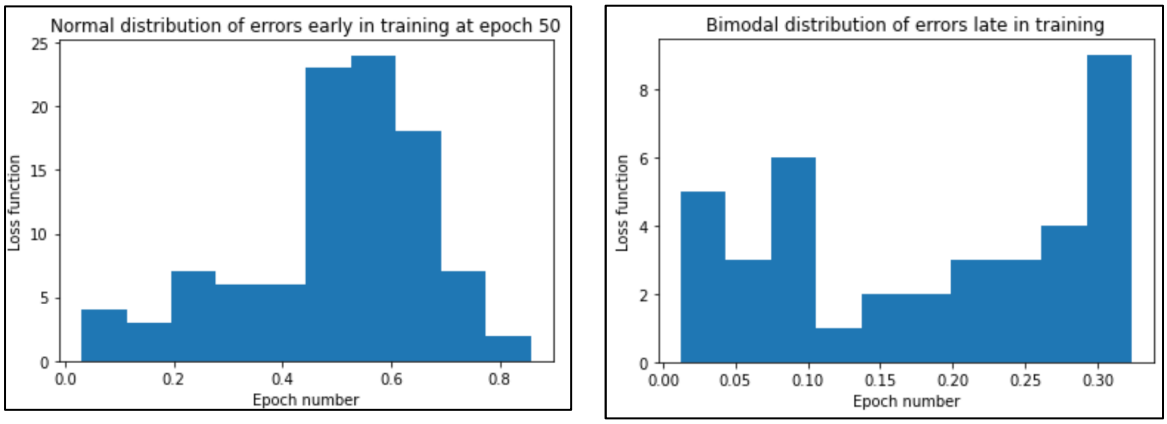

Fig. 2. Verification of appearance of Bimodal Distribution

3.3 Normal Backpropagation Results

A python class based on the network topology specified in Section 2 was written. To measure how 'accurate' our model is doing, a prediction within 5 marks of the actual grade was classified as 'correct', and if not, 'incorrect'. This accuracy was found for the test set of patterns (which the network has never seen before) and plotted as a function of epoch. We observe the typical diminishing returns of the accuracy on the test set, as exhibited in the convergence of the plot. To account for fluctuations based on randomness, the result shown is the average of 5 runs of the neural network.

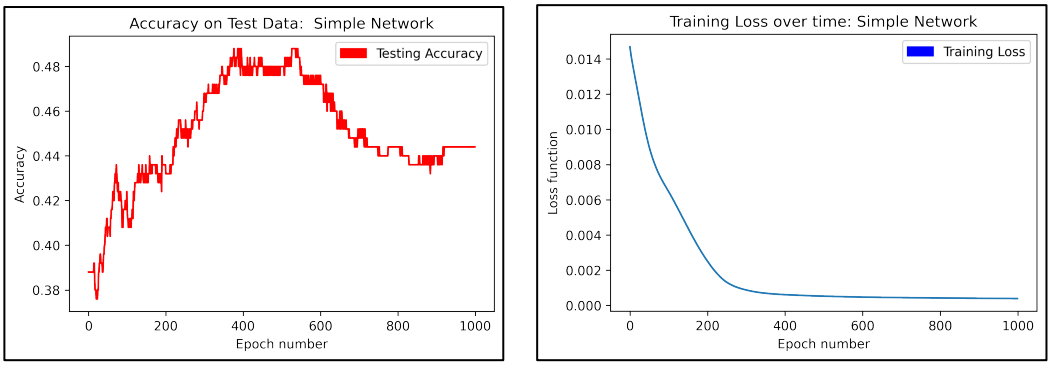

Fig. 3. Training of simple neural network

We obtain similar results as those in paper [3]. Because we have introduced noise in the training by not including the final marks, we get overfitting after about 500 epochs, and test accuracy reduces. At the elbow point, the network achieves a maximum of 48% 'accuracy', that is within 5 marks of correct prediction.

3.4 BDR

It is found that by applying BDR, the training loss reduces much faster than when no BDR was used, as can be seen by comparing the rate of change of the slope. Further details can be found in the training code provided in Jupyter notebook format. Data points were removed every 100 epochs. The threshold was reached after 1600 epochs, and a total of 20 points were removed by the BDR algorithm, which falls within the range of acceptable number of outliers. The accuracy on the test set reaches about 52%, which is slightly better than the simple neural network. Some output is also shown.

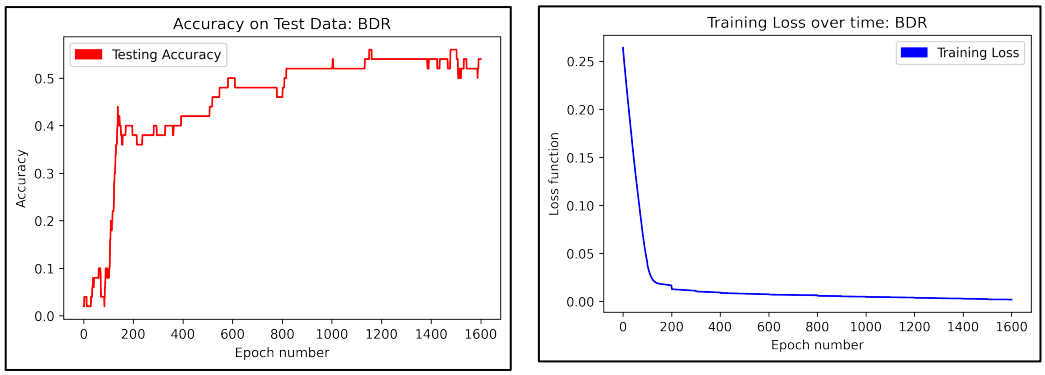

Fig. 4. Training of neural network with BDR

```
Training Done!
Threshold reached at Epoch: 1600
Remaining training patterns: 76 points
BDR removed total 20 points from the training dataset
```

3.5 Initial Comparisons

It can be seen that while loss reduces at a much faster rate with BDR than without, the difference in accuracy is only about 4-5%, so not that much. To see if we can achieve better results, we will find the 'optimal' hyperparameters, and then repeat analysis above.

3.6 Hyperparameter optimisation using GA

The initial results from the previous sections indicate to the delicate nature of the dataset, and its tendency to overfit early in training, as well as dependence on the initialisation of hyperparameters. To alleviate such concerns, a genetic algorithm was used to determine the optimum setting for some of the hyperparameters of the neural network. I observed that by using the Adam algorithm and a higher learning rate, the network start overfitting very early in the training, after about 200-300 epochs. I implemented an evolutionary algorithm in order to find the 'best' learning rate, between 0.0005 and 0.005 (typical default learning rate for Adam is ~0.001 [8]). The next parameter to optimise is the total number of epochs run. Since we start overfitting, it should be possible to save training time by only training until test set accuracy is increasing. The third parameter we can optimise is the size of the hidden layers. Previously, we used two hidden layers of 5 neurons each. It would be interesting to see what happens if this is changed to between 0-10, and whether we achieve any efficiencies by changing this. The initialisation is done with 10 samples, and the genetic algorithm was used for 15 generations. The fittest individual of each generation was stored in a kind of hall-of-fame format. The final choice of hyperparameter was the best out of all the hall of famers. Standard implementation of the binary encoding, random mutation, and roulette wheel cross-over was used, based on the algorithm in the lecture slides [10]. The accuracy on the test set was used as the fitness function.

3.7 Optimal Parameters with simple backpropagation

Using the parameters obtained from the GA (epochs = 350, learning rate = 0.004, hidden neuron = 8), I re-ran the tests in Section 3.3 for the simple neural network. This time, we achieve better test accuracy which converges to about 62%.

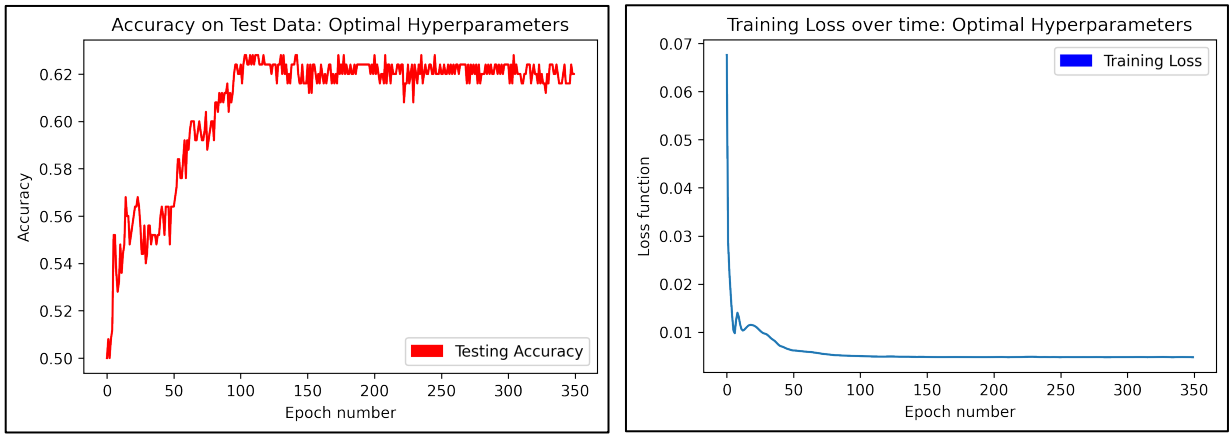

Fig. 5. Training of simple neural network using GA hyperparameters

3.8 Optimal Parameters with BDR

With the optimal learning rate and the hidden neurons setting, the neural network with BDR converges to about 70% testing accuracy, which is a good improvement over both its previous run in Section 3.4 as well as with respect to the results from without BDR in Section 3.7. Model starts to overfit after 250 epochs.

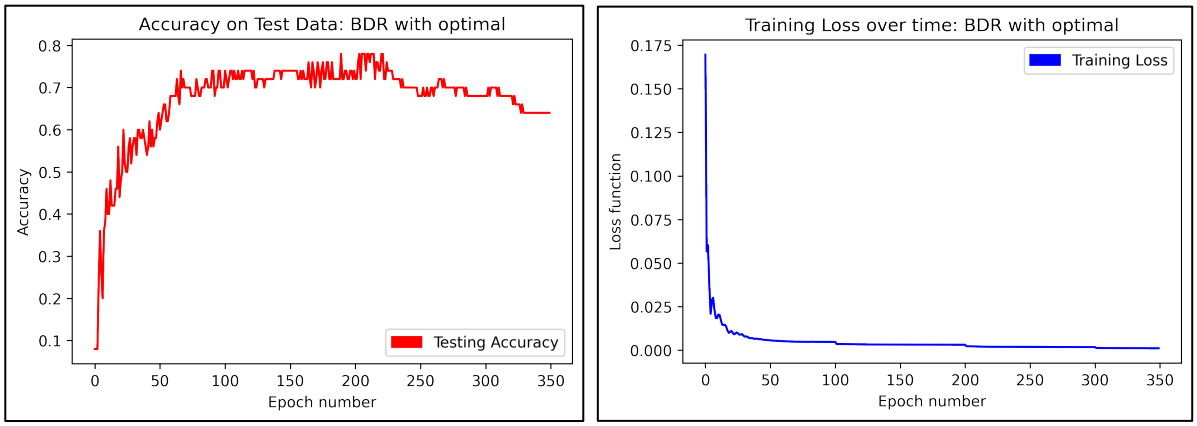

Fig. 6. Training of neural network with BDR using GA hyperparameters

4 Discussion, Conclusion and Future Work

It was found that BDR shows small performance improvements over simple neural network training, in terms of accuracy on the test set. It also enables convergence to a better performance level faster. This happens because the 'outliers' in the training set are removed, thus allowing for 'better' weight updates, in fewer epochs. Another benefit of the BDR technique is that it has a natural stopping criterion, which helps to prevent overfitting by cutting off training. Overall, BDR is no worse than simple training, with some additional benefits.

After using the Genetic Algorithm to find the optimal hyperparameters, it was found that the accuracy of the network improved by about 30-35%, because 62/48 ≅ 1.3, and 70/52 ≅ 1.35. This shows that extending the approach with genetic algorithms is useful to determine best hyperparameters for a neural network, and helps to increase accuracy.

All of the results presented were based on marks prediction dataset. There are some drawbacks of this dataset that need to be considered in the context of the conclusion it presents. Primarily, the data is very small, only about ~150 records. There are also a large amount of missing values, which have been assumed to be zero. It is possible that it is not the case, in which case we are adding additional bias into the dataset.

Further, due to the small amount of data, it was very easy to start overfitting the models, as evidenced in the results section. The ratio of training to test samples was roughly 2:1. We would most definitely observe different results if this ratio was changed. Also, the splitting was done randomly. It is possible that the proportion of 'true' outliers was different in both samples, and gave us an inaccurate representation of the full set. Both extreme cases are possible where we have all or no outliers in the training or test set.

It is also possible that there is a weaker relationship between marks in smaller assessments and the total marks. Perhaps the final exam mark, which we do not have and is weighted more heavily, is much more correlated with the total marks, so essentially all of the training data is noise. Or it could be the complete opposite, and that when we are removing training points, we are basically ruining our training by throwing away perfectly good data!

Also, while in theory BDR removes outliers, in practice, some proportion of those removed points may be perfectly valid data on the borderline of the removal threshold. These points could have been learned by the network eventually.

As a trained statistician, I find it hard to remove data unless there is a compelling reason. And while the BDR theory suggests that it is a good way to remove outliers, it should be considered whether a throwing away a large proportion of the dataset makes sense to the problem at hand. Outliers and influential points are interesting for statistical analysis. They can help tell whether there was some error during data collection or typing.

Also, as noted in Section 3.1, the results from a simple linear regression model gave and adjusted R^2 of 85.1% (see Appendix 6.1 for full output). While the aim for this paper was to illustrate GA and BDR theory in practice, it should be considered whether there is any need to apply a neural network technique for this particular dataset at all. I think an approach continuing along a more statistical path would be appropriate for this data.

The original research by Gedeon also compares BDR with other outlier detection methods like Least Trimmed Squares (LTS), Heuristic Reduction and the Absolute Criterion Method. Further analysis should compare those methods. Also more focus can be given to the statistical analysis of the neural nets, and its relationship with regression could be explored. More variants of the networks can be tested.

We could experiment with changing the ratio of the testing to training data, or even better would be to, if there were adequate computing power and time, use leave-one-out cross-validation or similar method. Results from that methodology would be least affected by randomness in the training/test splitting and could provide for useful verification of results.

Overall, we evaluated the BDR technique for outlier removal and compared the training stage and results with those obtained without outlier removal. We later extended this by using a genetic algorithm to find the best hyperparameters and repeated the previous analysis. It was found that GA improves the testing accuracy. BDR is a reasonable method for outlier removal.

5 References

[1] C. M. Bishop, Pattern recognition and machine learning, M. Jordan, J. Kleinberg and B. Scholkopf, Eds., New York: Springer, 2006.

[2] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359-366, 1989.

[3] P. Slade and T. D. Gedeon, "Bimodal distribution removal," in *IWANN International Conference on Neural Networks*, 1993.

[4] T. D. Gedeon and H. Turner, "Explaining student grades predicted by a neural network," in *Proceedings International Join Conference on Neural Networks*, 1993.

[5] T. D. Gedeon and T. G. Bowden, "Heuristic Pattern Reduction," in *International Conference on Neural Networks, vol.2*, 1992.

[6] E. C. Y. Choi and T. D. Gedeon, "Comparison of extracted rules from multiple networks," in *Proceedings of ICNN'95, International Conference on Neural Networks, vol.4*, 1995.

[7] Y. Yu, "The Implementation of Bimodal Distribution Removal on Neural Network and Experiment on Anger Dataset," in *ANU Annual Bio-Inspired Computing Student Conference vol.2*, 2019.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[9] V. Miranda, D. Srinivasan and L. M. Proenca, "Evolutionary computation in power systems," *International Journal of Electrical Power & Energy Systems,* vol. 20, p. 89–98, 1998.

[10] T. D. Gedeon, "COMP4660 ANU Course Notes," 2019.

[11] T. D. Gedeon, P. M. Wong and D. Harris, "Balancing bias and variance: Network topology and pattern set reduction techniques," in *Lecture Notes in Computer Science, vol .930*, BERLIN 33, 1995.

[12] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Parallel Distributed Processing," MIT Press, 1986, pp. 399-421.

[13] I. J. Ramirez-Rosado and J. L. Bernal-Agustin, "Genetic algorithms applied to the design of large power distribution systems," *IEEE Transactions on Power Systems,* vol. 13, p. 696–703, 1998.

[14] M. Gen and L. Lin, "Genetic algorithms," *Wiley Encyclopedia of Computer Science and Engineering,* p. 1–15, 2007.

[15] J. A. Joines and M. W. White, "Improved generalization using robust cost functions," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 1992.

6 Appendix

6.1 Simple Linear Regressions Output in Python

OLS Regression Results

Dep. Variable:	final	R-squared:	0.862
Model:	OLS	Adj. R-squared:	0.851
Method:	Least Squares	F-statistic:	84.04
Date:	Sat, 02 May 2020	Prob (F-statistic):	5.92e-53
Time:	17:11:30	Log-Likelihood:	-495.72
No. Observations:	146	AIC:	1013.
Df Residuals:	135	BIC:	1046.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	10.0037	2.253	4.441	0.000	5.549	14.459
lab2	0.0237	0.694	0.034	0.973	-1.348	1.395
tutass	1.2973	0.505	2.568	0.011	0.298	2.296
lab4	1.9155	0.896	2.138	0.034	0.144	3.687
h1	0.2000	0.133	1.510	0.134	-0.062	0.462
h2	0.3880	0.116	3.339	0.001	0.158	0.618
lab7	4.0070	0.843	4.753	0.000	2.340	5.674
p1	0.5833	0.133	4.386	0.000	0.320	0.846
f1	-0.1807	0.137	-1.316	0.190	-0.452	0.091
mid	0.9233	0.078	11.907	0.000	0.770	1.077
lab10	0.6027	0.884	0.682	0.496	-1.145	2.351