Refining Neural Network Training Data for decreased loss: A technical report

Alex Ilowski, Research School of Computer Science, Australian National University (2018)
Contact details: u5350297@anu.edu.au

Abstract. When creating a Neural Network (NN) to solve a classification problem, where the data comes from a real-world data set. We can often find that the data can be unreliable. It can contain two much information on some classes. Whilst not providing enough information on others and in small datasets significant outliers can pose an issue to data quality. These factors all contribute to the difficulty of training and testing effective NNs. In this report I will present a comparison of a basic NN of my own design and a NN implemented by Jamie Twycross[1]. I will then compare with a new NN with the addition of some of the techniques for improving NN data outlined in the paper by R.A. Bustos and T.D. Gedeon[2]. Then demonstrate the benefit of the extra techniques used to improve the NN. As well as an implementation of the technique to a more complex Recurrent Neural Network (RNN).

1 Introduction

When creating a Neural Network (NN) for a classification problem. There are many applicable measures of performance. I will be focusing on the accuracy of the NN, that is how many objects it predicts correctly compared to the total number of objects in the supplied dataset. Along with the loss of the NN, that is the difference between the predicted output and the actual output. These variables have been chosen as each is an important measure demonstrating the quality of a model and allows direct comparison between different NNs to be made. Which can then be used to quantify whether the differences between those NNs are improving or decreasing overall quality.

Having a suitable representation of the patterns available in the data set that the NN takes as input. Will have many implications for both the accuracy of the NN and the loss incurred when the NN makes an incorrect decision.

The data set used in this report comes from the UCI data repository[3] and will henceforth be referred to as the Voting dataset. The Voting dataset set includes the votes for each of the U.S. House of Representatives Congressmen on 16 key votes. The dataset contains 435 instances and 16 attributes. The class in this data set is binary as each instance is either republican or democrat. Similarly, each attribute is primarily binary recording either a yes or no. In some cases, the vote was unknown and is encoded as a third option.

The motivation to pick this data set was in part due to the simple, binary nature of the data. Thus, making the implementation of additional techniques easier to perform and directly measure their impact on NN performance. As well as a personal interest in the psychological nature of the data. That is the ability for a program to accurately determine what sort of person (politically) someone might be based on the decisions (in this case voting patterns) that they have made.

In this report I will present the results of three simple NNs being used to solve a classification problem using the Voting data set. This was a simple classification problem, attempting to train a NN such that it could correctly classify a republican or a democrat based on the way they voted. The first of which being a simple NN of my own design with three layers. Which works by predicting the class Y based on the input value X which contains the values for each attribute. Which will now be

referred to as NN1. The second being the results of a simple naïve-Bayesian NN implemented in a paper by Jamie Twycross[1]. Referred to as NB1. This will provide a base for my own NN to be compared to. Then I will present another NN based on the first NN that I produced. With the addition of a new method for refining the input data set based on techniques demonstrated in the paper by R.A. Bustos and T.D. Gedeon[2]. Referred to as NN2. I have presented the results and experience as below using statistical techniques to quantify differences between the different NN's outputs. Finally, I will attempt to apply the same technique to a more complex deep neural network specifically a recurrent neural network (RNN). The RNN differs from the previous NNs as it has feedback connections from output to input giving it the ability to store information in memory[4]. It will be tested using a different dataset called "textData" that can be found in the appendix of this paper and was supplied by the ANU.

2 Method

2.1 Refining the Dataset

Upon downloading the dataset from the UCI repository (house-votes-84.csv) I first went about performing some pre-processing on the data. I moved the class column that held the republican/democrat value for each instance from the 1st column in the data set to the last column of the dataset. This was a change I performed to make it easier on myself and might not be necessary in future work. The next step was to remove the header rows from the dataset. This information was no longer relevant as I knew that the last column was the class column and I would extract the information on democratic/republican favoured votes later. As all the binary values were encoded using strings these values needed to be re-encoded as numeric values to fit my NNs. I replaced the 'y' and 'n' values with 0 and 1 respectively. The '?' value that encoded an unknown vote was encoded to 0.5 which when taken with yes being 0 and no being 1 reveals no information about the possible class. Finally, I replaced the strings "republican" and "democrat" in the class column with 0 and 1 respectively.

2.2 Testing and Training the NN1

The dataset was then split into two sets a training set that contained 90% of the original data and a testing set that contained the remaining 10% of the data. The dataset is then fed into a NN that consisted of three layers. An input layer with 16 neurons, a hidden layer with 30 neurons and an output layer with a single neuron representing the binary class 0 or 1 (republican or democrat). The NN trained over 2500 epochs which was the maximum number before it started losing accuracy. These attributes were chosen based on initial testing of the NN with various attribute values and these provided good results. This NN was trained using Stochastic Gradient Descent as an optimiser and had a learning rate of 0.01. This simple NN was compared to NB1 and the results are listed below. Both NNs were used to run a 10-fold cross validation. Where the training and test data are split into 10 sets, 9 are used for training 1 is used for testing. Then the 1 set that is used for testing is rotated until each set has been used for testing once.

2.3 Testing and Training the NN2

NN2 is the same as NN1, detailed above. Except it implements additional methods for refining training data. In order to increase the quality of the results I wanted to refine the input dataset. To do this the NN takes a sum of each of the two classes. It then creates an object to record the type of each vote. That is whether the vote was dominated by republican or democratic yes voters. Then by

using this list create a mean for each of republicans who voted in favour of votes dominated by democrats and a mean for democrats who voted in favour of votes dominated by republicans. These means provide a baseline for each group that allows us to begin removing outliers. The variance can then be taken and finally the standard deviation can be calculated. Using the standard deviation remove the outliers from the dataset. I chose to remove those voters who sat outside of a 95% confidence interval in the positive direction. This is because a democrat who votes more republican than the majority of other members of their class can confuse the NN, but a democrat who votes less republican does not lose information about their class and vice versa.

2.4 Testing and Training RNN1 and RNN2

RNN1 and RNN2 are functionally the same except that RNN2 implements some additional functions to process the input data based on the same principles as the technique used for the simpler NNs. Both RNNs are run over 5000 iterations. This is the point that the decreases in loss become minimal. Each RNN has 100 hidden neurons, a sequence length of 50 and a learning rate of (1e -1). The data is read in to each NN as a list data structure. In the case of RNN1 it starts processing immediately. As stated RNN2 implements additionally pre-processing. The NN takes a copy of the data and removes all alphabet characters and the single whitespace character. It then takes the number of occurrences of the remaining chars, produces a mean, calculates variance and standard deviation and then collates a list of outlier characters. In this case the outliers on the lower end of the spectrum are removed as the removal of these characters does not remove as much information as removing outliers on the other end of the spectrum. This list is then used to remove these characters from the original dataset and then processing begins in the same manner as RNN1.

3 Results

3.1 Results for NN1

As detailed previously I ran a 10-fold cross validation on each NN.

10-fold cross validation of NN1:

Testing Accuracy: 94.23 %	Loss: 0.2096
Testing Accuracy: 91.89 %	Loss: 0.2112
Testing Accuracy: 97.37 %	Loss: 0.2141
Testing Accuracy: 87.18 %	Loss: 0.2073
Testing Accuracy: 92.31 %	Loss: 0.2117
Testing Accuracy: 95.00 %	Loss: 0.2098
Testing Accuracy: 97.22 %	Loss: 0.2102
Testing Accuracy: 92.68 %	Loss: 0.2086
Testing Accuracy: 96.43 %	Loss: 0.2204
Testing Accuracy: 97.14 %	Loss: 0.2022

Figure 3.1 Table contains testing accuracy and loss for each test for first NN

Based on the results from the testing we can calculate the mean, standard deviation and 95% confidence interval for both the Testing Accuracy and the Loss.

	Mean	Standard Deviation	95% Confidence interval
Accuracy	94.145%	3.230%	92.143% - 96.147%
Loss	0.211	0.005	0.208 - 0.213

3.2 Results for the NB1

This NN was also tested with 10-fold cross validation

	Mean	Standard Deviation	95% Confidence interval
Accuracy	90.1%	4.9%	85.7% - 94.5%

The predictive accuracy was 90.1% with standard deviation of 4.9% and a 95% confidence interval of 8.8%. No data was given for the loss of this NN.

3.3 Results for NN2 with additional techniques implemented

10-fold cross validation of NN2:

Testing Accuracy: 91.67 %	Loss: 0.1234
Testing Accuracy: 90.70 %	Loss: 0.1053
Testing Accuracy: 86.27 %	Loss: 0.1052
Testing Accuracy: 92.00 %	Loss: 0.1125
Testing Accuracy: 95.45 %	Loss: 0.1056
Testing Accuracy: 84.62 %	Loss: 0.1148
Testing Accuracy: 88.89 %	Loss: 0.1080
Testing Accuracy: 85.71 %	Loss: 0.1081
Testing Accuracy: 82.22 %	Loss: 0.1098
Testing Accuracy: 92.00 %	Loss: 0.1131

Figure 3.2 Table contains testing accuracy and loss for each test for second NN

Based on the results from the testing we can calculate the mean, standard deviation and 95% confidence interval for both the Testing Accuracy and the Loss.

	Mean	Standard Deviation	95% Confidence interval
Accuracy	88.953%	4.124%	86.397% - 91.509%
Loss	0.11058	0.006	0.107 - 0.114

3.4 Results for RN1

Loss
2.091
1.912
2.156
1.794
1.883
1.976
1.970
1.974
1.806
1.899

Figure 3.4 Table contains loss recorded for each test of RNN1

Based on the results from the testing we can calculate the mean, standard deviation and 95% confidence interval for both the Testing Accuracy and the Loss.

	Mean	Standard Deviation	95% Confidence interval

Loss	1.946	0.114	1.875 - 2.017

3.5 Results for RN2

Loss
1.855
1.804
2.017
1.881
1.928
1.954
1.868
1.674
1.861
1.938

Figure 3.3 Table contains loss recorded for each test of RNN2

Based on the results from the testing we can calculate the mean, standard deviation and 95% confidence interval for both the Testing Accuracy and the Loss.

	Mean	Standard Deviation	95% Confidence interval
Loss	1.878	0.093	1.820 - 1.927

4 Discussion

4.1 Comparison of NN1 and NB1

Using the same Voting dataset, initially we see that NN1 performs better than NB1. NN1 appears to have a higher mean predictive accuracy, lower standard deviation and as a result a smaller 95% confidence interval than NB1. However, these metrics can be difficult to determine if either model is truly better as accuracy can vary greatly from test to test and given that the mean of NN1 is within one standard deviation of NB1's mean this indicates that the results are too similar to say that one is significantly better than the other. In spite of the similarities between the two. NN1 serves its purpose well. As a base line that is comparable to other NN implementations on the same data set. That can then be then be used to compare to and understand the value of other techniques in relation to NN performance.

4.2 Comparison of NN1 and NN2

Each of these NNs used the same initial Voting dataset. However as outlined in the method section NN2 has methods to remove outliers from the data in an attempt to refine the data. Which in turn creates a better NN for the classification problem. Based on the results above we can see that the removal of outliers has caused the accuracy to decrease. Although accuracy is not always the best indicator of performance due to accuracy varying, sometimes greatly, within runs on a single NN. Additionally, we may be willing to trade off accuracy in favour of significant increases in other areas for certain fields of NN problems. In saying this we can easily see that the Mean predictive accuracy of NN2 has fallen compared to NN1. However, we can see that the Mean of the loss for the NN2 has improved significantly reducing by almost half of that of NN1. It seems to be that the improvements in loss brought about by the new technique is outweighing the slight decrease in accuracy that has

come with it. Suggesting that this technique has reduced the predictive accuracy of the NN on the testing set (decreased accuracy) but that the NN is making less mistakes during training.

4.3 Comparison of RNN1 and RNN2

Each of the RNNs used the same initial dataset. The difference in function between the two is the pre-processing functions implemented by RNN2 on said data set. These functions were based on the principles for data processing that saw a successful decrease in loss when applied to NN2 using the voting dataset. The concept is still to remove outliers from the data that lead to increased loss in the data. However, unlike the success seen for NN2 compared to NN1 the difference in results between RNNs is no where near as significant.

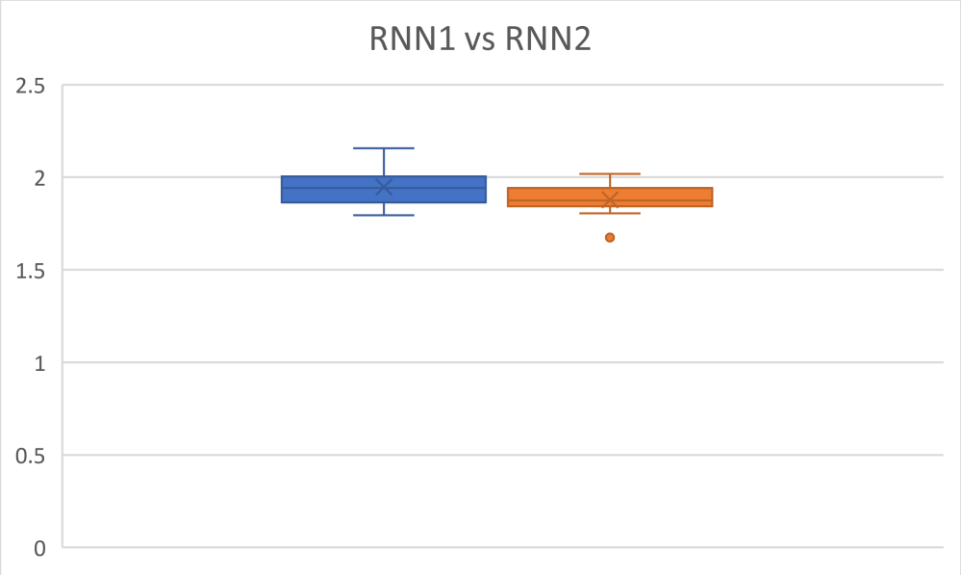

Figure 4.1 Comparison of RNN1 and RNN2

As seen in figure 4.1 we can see that RNN2 has a tighter range than RNN1 and it indicates a slightly decreased loss in comparison. However, we can see that the majority of the data points of RNN1 fits within one standard deviation of the mean of RNN2 so it is difficult to say whether there was a real effect on the loss brought about by the pre-processing functions or whether the improvement is an anomaly of the runs recorded for these results. Even proceeding under the assumption that the additional pre-processing is what resulted in the decreased loss we can see that the end result was ultimately very small.

5 Conclusion

5.1 Future Work

In order to keep the results consistent and comparable I chose to keep the internal values for each NN consistent. i.e Number of epochs, numbers of neurons, numbers of layers, learning rates and optimisers. After implementing and testing the second NN I discovered that there were other effects that the techniques were having on the results other than the improved loss and decreased accuracy described above.

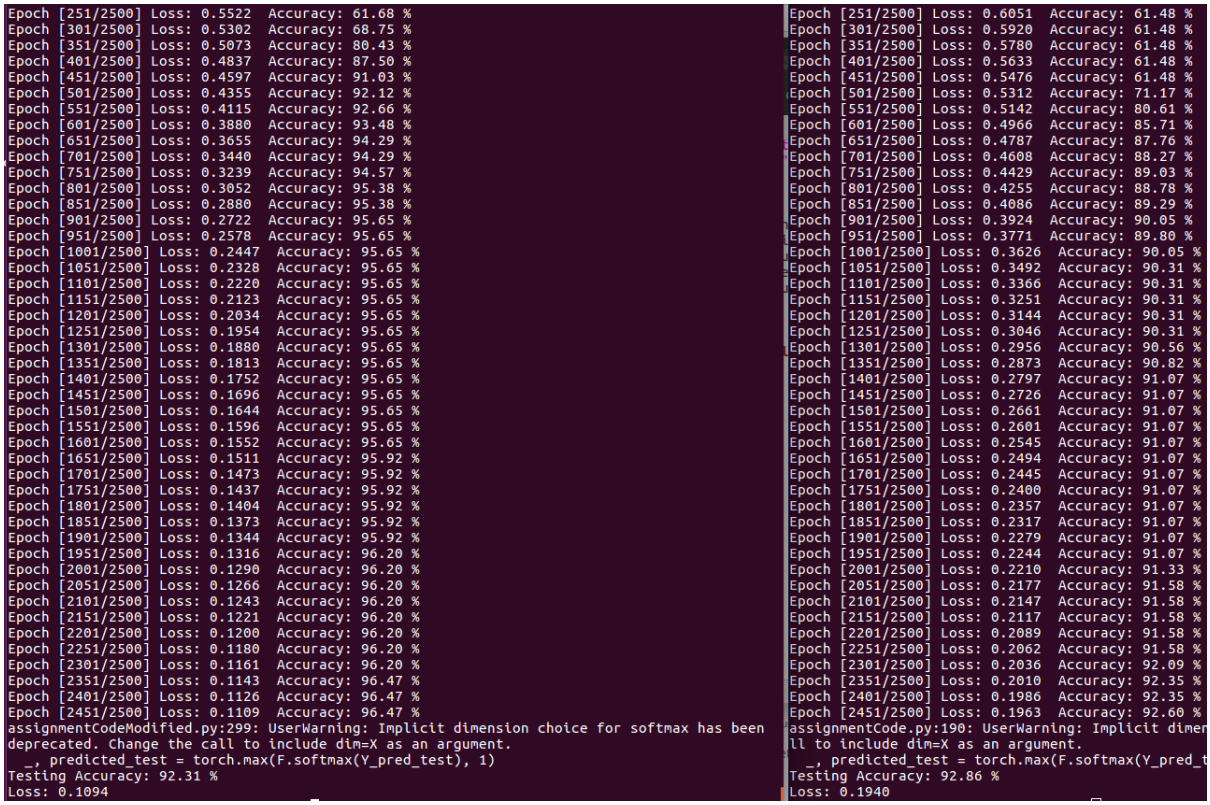

Figure 4.1 Side by side comparison of first and second NNs (second NN on the left, first NN on the right)

As can be seen in figure 4.1 the number of epochs before accuracy improvement is minimal is substantially lower for NN2 and the final testing accuracy has actually decreased from the training accuracy. Which is not the case for NN1. This is supported by the lowered loss indicating less mistakes being made during training leading to NN2 being trained faster than NN1.

So, in future work I would want to look at modifying the number of epochs and other internal NN attributes to see if in combination with the newly implemented techniques the NN could be improved further for the Voting classification problem. I also think that the methods used on the RNNs could be further improved by conducting more research on what outliers can be removed with resulting information losses being as small as possible. This could then tested over an increased series of runs to improve confidence that the methods are improving the output or confidently rule them out as useful in the field of RNNs.

5.2 Closing Statements

I have shown that for a classification problem using the Voting dataset we can produce a simple NN to solve this problem. That this NN can perform as well as other simple NNs produced by others in the field for the same problem and dataset. I have shown that the result of this NN can then be improved upon in terms of Loss if not Accuracy. By refining the input dataset to have a reduced number of outlier classes that can cause a NN to lose performance.

I have shown that using a similar methods to try and improve RNNs has proved mostly ineffective and at best has produced the most marginal positive effect.

References:

1. J. Twycross, An Immune System Approach to Document Classification, COGS, University of Sussex, U.K. Hewlett-Packard Research Labs, Bristol, U.K. August 30, 2002
2. R.A. Bustos, T.D. Gedeon, DECRYPTING NEURAL NETWORK DATA: A GIS CASE STUDY, School of Computer Science Engineering University of New South Wales, Sydney NSW 2052, Australia, 1995
3. UCI Machine Learning Repository,
http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
4. L.R. Medsker, L.C. Jain RECURRENT NEURAL NETWORKS, American University Washington, D.C. and University of South Australia, 2001

Appendix:
Dataset “textData” is in the attached zip files submitted to easychair