The Effects of a Sparse-Autoencoder of the Detection of Spam Emails

Robert Whittaker

Research School of Computer Science
The Australian National University, Acton, ACT, 2601, Australia

Abstract. Detecting spam emails is an important component of bringing revenue to email providers and is useful for the owner of an email address. While there are numerous spam email detection systems, none reliably detect spam at a 100% accuracy. This paper shows that we can reliably get an accuracy of 93.72%. using normalization, a sparse auto-encoder and dropout. It explains the processes needed to get this accuracy and compares the results found to Dimitrakakis and Bengio. It also makes recommendations for future methods that can be used to increase the accuracy of a spam detection mechanism.

Keywords: Neural Network, Normalize, Spambase, Deep-Learning, Sparse-Autoencoder

1 Introduction

Spam emails are a common problem faced by consumers who have an email account to both send emails and receive emails from others. Spam emails take a variety of forms that target the consumer for a variety of reasons although all are unsolicited. These reasons include the advertising of products, the sending of viruses and attempting to scam the consumer. Many companies currently have their own email service and are thus trying to compete for consumers to use their product. As such services with a more accurate spam detection technique do more to protect their consumers, thus consumers are more likely to use them. As such a database which is used to detect possible emails which contain spam is the Spambase dataset, which describes features of the email as classifies whether it is spam or not.

The pre-processing of data is a crucial aspect of a neural network to learn. Normalization of the input data prior to the training process of the network is both important to receive good results and to reduce the calculation time of the network. [1] An important problem faced when creating a neural network as it needs to be determined before modelling whether data needs to be normalized or not. Normalization within a neural network help remove some geometrical biases within some of the data vectors. [2]

In this report we focus on implementing a neural network to identifying spam emails using the Spambase dataset. In section 2, the architecture of the neural network is explained which includes the normalization of the data, the sparse auto-encoder and dropout. In section 3, we analyse our results and compare then to Dimitrakakis and Bengio paper. [3] In section 5, potential future works on the paper are suggested.

2 Method

2.1 Data Analysis

The raw data used in this report is called 'Spambase' and originates from Hewlett-Packard. This information is represented within a grid of 266858 values. Each vector in the image contains 58. This dataset was chosen for investigating the effects of normalizing data is it contained 4601 instances and 57 attributes. [4]

Table 1. Information on data shown in the Spambase folder

Column Numbers	Description	Data Type
1-48	The percentage of times that a word occurs within the e-mail.	Continuous real [0, 100]
49-54	The percentage of times that a character occurs within the e-mail.	Continuous real [0, 100]
55	The average length of uninterrupted sequences of capital letters within the e-mail.	Continuous real [1, …]
56	The length of the longest uninterrupted sequence of capital letters within the e-mail.	Continuous real [1, …]
57	The total number of capital letters within the e-mail.	Continuous real [1, …]
58	The target, 0 – denotes it is not spam. 1 – denotes it is spam.	Nominal {0,1}

This paper uses dataset to investigate on which pieces of data need to be normalized. The data was analysed to work out which data should be normalized. This is shown in the tables below.

Average Statistics regarding the number of words and characters within an e-mail

Table 2. Statistics for all data which involve the number of words and characters within the email.

Method	Value
Maximum	12.08
Minimum	0
Mean	0.18
Median	0.03
Standard Deviation	0.61

As shown in the statistics the values found within the data are moderately small. This is as the data is a percentage with its maximum value of 12.

Average Length of Uninterrupted Sequence of Capital Letters:

Table 3. Statistics for Average Length of Uninterrupted Sequence of Capital Letters

Method	Value
Maximum	1102.50
Minimum	1
Mean	5.19
Median	2.28
Standard Deviation	31.7294487

Length of Longest Uninterrupted Sequence of Capital Letters:

Table 4. Statistics for Length of Longest Uninterrupted Sequence of Capital Letters

Method	Value
Maximum	9989.00
Minimum	1
Mean	52.17
Median	15
Standard Deviation	194.89131

Total Number of Capital Letters:

Table 5. Statistics for Total Number of Capital Letters

Method	Value
Maximum	15841
Minimum	1
Mean	283.29
Median	95
Standard Deviation	606.347851

As shown in the three tables above, the pieces of data had a much larger variation than the rest of the data. This suggests that more pre-processing needed to be done on these pieces of data.

The data in the network was also split into a learning set and a test set. It was separated by having 80% of the data within the learning set and 20% of the data within the test set.

2.2 Normalization

The normalization method used to normalise the data within the network was the min-max normalization technique [1]. In the min-max normalization approach the data normalized is scaled to a fixed range between 0 and 1. This thus results is a much smaller standard deviation, which can remove much of the biases caused by outliers. [5] The formula for the min-max normalization technique is shown in equation 1 below.

$$x = (x - \min(x)) / (\max(x) - \min(x)) . \quad (1)$$

Not all the data within the dataset needs to be normalized. For example, in certain circumstances such as already small spread of data normalizing the data can thus decrease the accuracy of the neural network learning of the data. As such it was found on the Spambase dataset, it was optimal just to normalize columns 55 – 58. By normalizing these columns individually the network learnt at a much faster rate and had an increased final accuracy.

2.3 Sparse-Autoencoder

The deep-learning method within a neural network is the implementation of a sparse-autoencoder. A sparse auto encoders can learn from a range of features, which include audio, text and etc. [6] An auto encoder is a learning algorithm that is unsupervised, it applies backpropagation, thus sets the target values to be equal to the inputs. [6] An example diagram of an autoencoder is shown in figure 1. A sparse-autoencoder was implemented once per epoch within the neural network on the Spambase dataset.

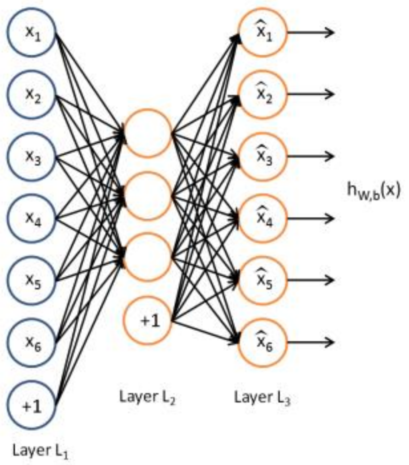

Fig. 1. An example of a sparse autoencoder. [6]

2.4 Dropout

Overfitting, is a major problem for large neural networks as many relationships between each of the layers exist. Dropout is technique that prevents overfitting and makes a ways of combining many neural network architectures efficiently. [7] The choice of which units to temporarily remove from the network is random and the units chosen to dropout are both hidden and visible in the neural network. Due to there being a large number of layers within a Deep learning network. Dropout can significantly reduce the number of connected neurons within the network. Figure 2, shows the effects that a dropout function has on a network compared to a network without dropout.

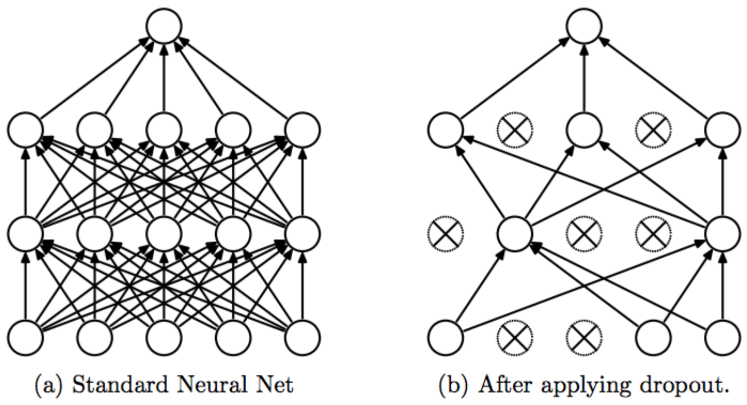

Fig. 2. An example of a standard neural network (a) compared to that of one with dropout (b). [7]

3 Results and Discussion

The neural networks were train for 1000 epochs at a learning rate of 0.5. The effects of adding a deep learning network with both a spare auto-encoder and dropout function were testing with that of a simple 3 layer neural network.

When a 3 layer neural network was used with the normalized Spambase data set a final testing accuracy after 1000 epochs of 90.73% was recorded. After 100 epochs an accuracy of 74.80% was record. Figure 3, shows a graph of the accuracy of the network on the test data after each epoch.

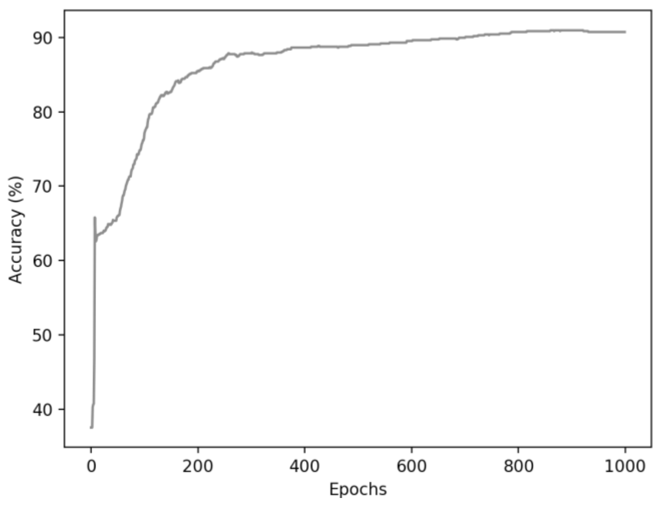

Fig. 3. The accuracy of the neural network on the test data at a varied number of epochs when normalising rows 55-57 of the data.

From figure 3, It was found that for the majority of the epochs the accuracy steadily increased. With a high increase in accuracy at the beginning and it started to approach a maxima around 91% towards the end. There was a low amount of fluctuations within the accuracy between the epochs which may have been because all the data was used for each layer within the network.

When more layers were added to the neural network, combined with a sparse auto-encoder and dropout, the final accuracy after 1000 epochs was found to be 93.72%. After 100 epochs it was 87.78%. Figure 4, shows a graph of the accuracy of the network on the test data after each epoch.

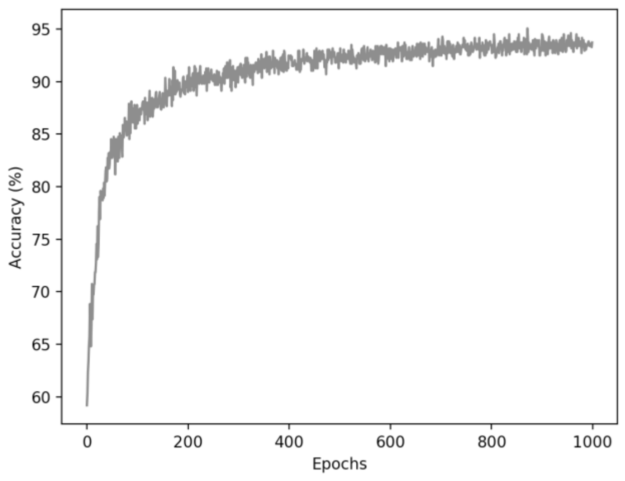

Fig. 4. The accuracy of the neural network on the test data at a varied number of epochs when normalising rows 55-57 of the data and adding a sparse-autoencoder and dropout.

It was found the neural network was a lot slower when a sparse auto-encoder and dropout was added, thus it took a lot longer to read 1000 epochs. The accuracy of the neural network generally increased after each epoch, although at the start it increased at a much greater rate. Although there were large fluctuation in the accuracy of up to 5%. This may have been caused by the dropout algorithms as some units which may have been important to keeping a high accuracy may have been removed temporarily.

It was found that a deep learning network with dropout and a sparse auto-encoder produced better results than a simple 3 layer network as expected. The accuracy after 1000 epochs was approximately 3% larger, and it also increased at a faster rate. Although the deep learning network had much larger fluctuations, which sometimes caused the accuracy to degrease between epochs.

4. Comparison

The results from the testing were compared to those of a Dimitrakakis and Bengio's *Online Policy Adaption for Ensemble Classifiers* paper. [3] As such the paper recorded their results as how much classification error their results had for each method which they used. As such, they record an 8.33% error for multi-layer perceptrons (MLP), a 6.48% error for AdaBoost (Boost), a 7.75% error for the mixture of experts architecture (MOE) and a 7.71% error for reinforcement learning (RL). [3] The results published showed the errors after 100 epochs, therefore it was reasonable to compare our results at 100 epochs to Dimitrakakis and Dengio's results.

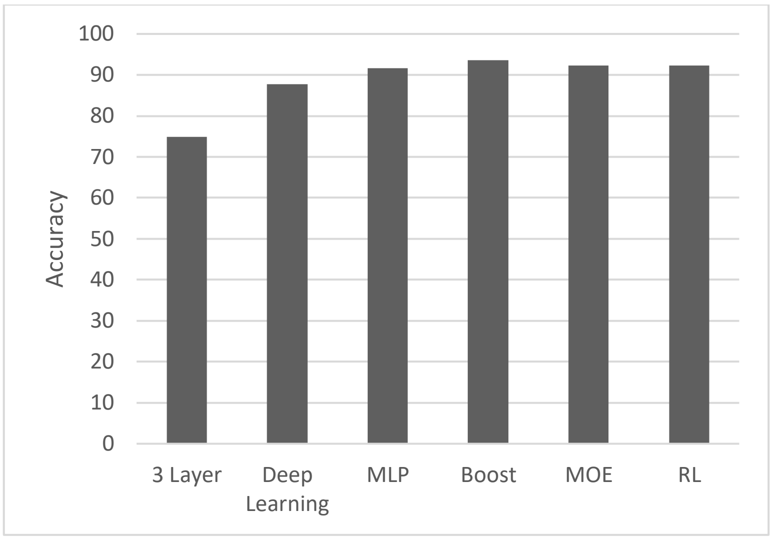

Fig. 1. Comparison of our results with those of Dimitrakakis and Bengio at 100 epochs. [3]

It was found that at 100 epochs both the 3 Layer and the Deep Learning approaches had a lower accuracy than those found by Dimitrakakis and Bengio. [3] It was found that the 3 layer network had a significantly lower accuracy than the papers results, meanwhile the deep learning neural network with a spare auto-encoder and dropout performed only slightly lower. It may be possible to get our results much closer to the results of the paper by fiddling around with the network to get the optimal variables, thus causing the final accuracy to be higher.

5 Future works

There are many possible ways in which the accuracy of the neural network that detects spam emails can be improved upon. These ways included changes that can be made to both the pre-processing of the data and the architecture of the network itself.

A method in which a future paper could improve upon the results would be to use Z normalization instead of min-max normalization on the Spambase dataset. This minor pre-processing change could help remove some of the biases within the dataset and thus causing the final accuracy to be higher.

There are 57 features current in the Spambase dataset. These features are mostly just the proportion of how many times a word appears within an email. A neural network may be for successful in detecting spam emails if more features were available. This would thus mean there would be more features to learn from with each having a potentially different weight.

A major weakness in the results of the network with the sparse auto-encoder and the dropout function was that there was a large amount of fluctuations with the accuracy of the data. A major improvement would be to decrease the size and rate of these fluctuations. To do this another deep learning technique could be implemented and tested to thus get more accurate results.

6 Conclusion

Overall, it was found that the addition of a sparse auto-encoder and dropout to a neural network on the Spambase dataset greatly increased its accuracy. It was found that via normalization and a 3 layer neural network an accuracy of 87.78 could be obtained. Although, it was found that the addition of a sparse-autoencoder and dropout increased the accuracy to 93.72%. It was found that the results of this paper were slightly less than those found by Dimitrakakis and Bengio. Finally, future works were suggested to increase the identification accuracy of spam emails by adding more features to the dataset, using Z normalization instead of min-max normalization and different methods could be used to increase the accuracy.

7 References

1. Sevilla, J., Sola, J.: Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems. IEEE Transactions on nuclear science. 40. 3. 1464--1468 (1997)
2. Jian, J., Ming, L., Jin, L.: Data Normalization to Accelerate Training for Linear Neural Net to Predict Tropical Cyclone Tracks. Mathematical Problems in Engineering. http://dx.doi.org.virtual.anu.edu.au/10.1155/2015/931629 (2015)
3. Dimitrakakis, C., Bengio.: S. Online Policy Adaption for Ensemble Classifiers. IDIAP. Martigny, Switzerland
4. Hopkins, M., Reeber, E., Forman, G., Suermondt, J.: Spambase Data Set, Hewlett-Packard Labs, https://archive.ics.uci.edu/ml/datasets/spambase (1999)
5. Raschka, S.: About Feature Scalising and Normalization – and the effect of standardization for machine learning algorithms. http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-min-max-scaling (2014)
6. Ng, A.: Sparse Autoencoder, https://web.stanford.edu/class/cs294a/sparseAutoencoder. (accessed 2018)
7. Scrivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929—1958. (2014)