An Implementation of Reduced Neural Network Using Genetic Algorithm for Email Spam Detection

Zhoubao Mai

Research school of computer science, The Australian National University
u6118739@anu.edu.au

Abstract. Email spam detection is one of the most urgent problems on the internet. Many models that detect spam require a large size of neural network, which can consume a lot of memory and time. In this report, a network reduction technique based on the distinctiveness with Genetic Algorithm is implemented to prune the neural network. The goal of this network reduction technique is to reduce the number of hidden units without affecting the accuracy. A backward propagation neural network classifier is used to verify the performance of this technique. Sensitivity and specificity are two primary performance evaluation methods. At the end of the report, comparison of results is shown with the original network reduction technique and the results of another paper based on the same dataset. The result shows that this adaptive neural network technology has about 75% compression rate with a minor loss of performance. This report confirms that this technique can be applied to real-world problems.

Keywords: Email spam detection, Genetic Algorithm, ANN, distinctiveness, Network Reduction Technique, Hidden units

1 Introduction

Email spam refers to emails that are unsolicited and often contains commercial advertising, malicious programs, or sensitive content [5]. With the development of the Internet, Email spam has affected people's daily lives and the security of network systems. Hence, email spam detection technic is invented to filter the junk email.

There have been many recent solutions to classify email spam through Artificial Neural Network (ANN) technology. Usually, these classifiers are done by a huge neural network, which means a large number of hidden units and hidden layers. Training such a large neural network requires a lot of time and memory, and it may overfit the training data set and lose the ability to generalisation [4]. An approach is to prune a neural network by reducing the number of hidden units. Gedeon and Harris [3] introduced a network reduction technique based on the distinctiveness of hidden units, in the paper, they use a heuristic to measure the distinctiveness and prune the network. This network reduction technique can effectively reduce the size of a neural network with minor loss of accuracy. However, the heuristic used in the original paper is rigid and lack of flexibility. Genetic Algorithm (GA) is a global optimisation algorithm borrows from the viewpoint of natural selection and survival of the fittest in the natural world [1]. It is suitable for finding optimal parameters. In the report, I will use GA to optimise the pruning process.

This report aims to implement and verify the performance of the reduced neural network using GA for Email Spam prediction. Dataset provided by UCI [2] contains 4601 instances and 58 attributes. The 4601 instances are Partition into two classes, 1813 instances for spam, and 3788 instances for not spam. A pruning algorithm is applied to reduce the unwanted hidden unit based on the correlation of hidden units. A GA is applied to search the optimal parameters for pruning algorithm. A neural network-based classification algorithm is used to measure reduced neural networks. For measuring the performance, sensitivity, specificity, and accuracy are essential parameters for a spam detection system.

At the end of this report will be compared with an Email Spam detection system [6] based on the same dataset. Through this comparison, the practicality, advantages and disadvantages of applying GA on pruning neural network based on the distinctiveness of hidden units are analysed.

2 Method

2.1 Neural Network Classifier Based on Back Propagation Algorithm

BP algorithm (i.e., error backpropagation algorithm) is a learning algorithm suitable for the multi-layer neural network. It is based on the gradient descent method.

The learning progress of the backpropagation algorithm consists of two part, the forward propagation and the backward propagation. During the forward propagation, the inputs are processing through the hidden layer to the output layer. If the predicted output is not equal to the actual output, calculate the error between the actual output and the

predicted output and back propagate using the error. While the back propagation, partial derivative of the function is calculated through layers to modify the weight in the weight vector. The network will stop learning if the error reaches the desired value.

2.2 Network Reduction Technology Based on Distinctiveness

According to Gedeon and Harris [3], the hidden unit's five properties (sensitivity, relevance, badness, contribution, and distinctiveness) can be used to determine whether the hidden unit is not needed. In this report, distinctiveness is used to determine the undesired hidden units.

The distinctiveness of hidden units is to measure to similarity and dissimilarity of each pair of hidden units. If two hidden units are significantly similar, they produce the similar output with the same pattern. In this case, one of the two hidden units can be removed and add the weight of removed, hidden unit to the remaining one. In contrast, if two hidden units are significantly dissimilar, this pair of hidden units produce no effect or constant effect. In this case, all the two hidden units can be removed.

For every hidden unit, put the activation of each input pattern together to form a vector. According to Gedeon and Harris [3], angles between two vectors are measured for the similarity of two hidden units. The formula to calculate the angle between each pair of vectors is

$$\theta = \arccos\left(\frac{v_1 \cdot v_2}{|v_1||v_2|}\right) \qquad (\mathbf{0})$$

Where v_1, v_2 means the two vectors formed by the pattern space and $\theta(0<\theta<180)$ means the angle between two vectors.

Since it is rare to find two complete identical vector or complementary vector, Gedeon and Harris [3] define the two vectors are similar if the angle between them is less than 15 degree and the two vectors are complementary if the angle between them is more than 165 degree.

2.3 Genetic Algorithm on Pruning Neural Network

Genetic Algorithm simulates the evolution process of an artificial population until it fits an ideal situation. Through selection, crossover and mutation, a set of individuals is retained in each iteration. This process is repeated, and the population evolves over several generations. A basic genetic algorithm works as follows:

1. Select the encoding strategy and convert the parameters into individuals
2. Randomly generated a population of individuals according to the population size N.
3. Calculate the fitness of each based on the fitness function
4. Using selection strategy to copy the individuals as offspring according to the fitness of them
5. Cross the individuals based on the crossover probability
6. Change the values of the individual based on mutation probability
7. Repeat step 3 to step 6 until meeting the stopping criteria or certain generations.

There is two measurements to evaluate the performance of neural network reduction techniques, the compression rate and the accuracy after the pruning. However, the two measurements may sometimes be conflict. Intuitively speaking, if the compression rate is too high, the accuracy after pruning may drop significantly. To handle this multi-objective optimisation problem, a Pareto optimal set is used to find the optimal solution. For such a solution, it is impossible to further optimise one or more ga goals without damaging other goals, the set of this kind of solution is called Pareto optimal solution set. In this report, I choose NSGA-II algorithm to select the Pareto optimal solution set.

2.4 Application

2.4.1 Data Preprocessing

Data processing is one of the essential steps in developing module, using data processing can transfer incomplete and inconsistent data into formatted one. The dataset has 4601 instances and 58 attributes. The dataset is divided into two parts, 20% of it as the test data and 80% of it as the training data.

I drop the “ID” attribute since the id takes no contribution to the prediction. Then, I normalise the input attribute to avoid attributes with a greater range have a more significant effect on the module.

2.4.2 Network Architecture

The artificial neural network is implemented by Pytorch. According to the published paper of Gedeon and Harris [3], the neural network is set to be a three-layer ANN with processing units. That is, the neural network has one input layer, one hidden layer and one output layer. I use all the 57 attributes as the input neuron. There are many rules-of-thumb on defining the number of hidden neurons, the most common one is 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers' [7]. Therefore, I set the number of a hidden neuron as 20. As for the

learning rate, I experiment several times setting the initial learning rate as 0.00003 then increase or decrease it exponentially and finally set learning rate as 0.3. The neural network will be trained with Stochastic Gradient Descent as the optimiser and cross-entropy as the loss function. Since the choice of activation function will not influence the network reduction techniques [3], this artificial neural network uses sigmoid function as the activation function. The training result of the neural network is shown in Figure 1.

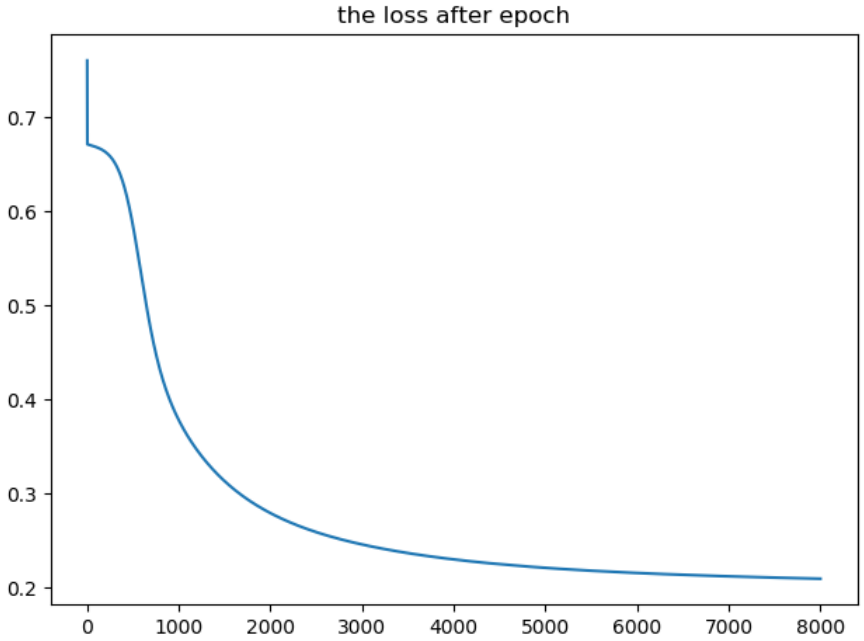

Figure. 1. shows the loss after each epoch, the x-axis is the number of epoch and y-axis is the number of corresponding loss. From figure 1, we can see that the loss is steadily decreasing after each epoch, which shows we have trained an effective neural network. The rapid decrease at the beginning may the caused by the learning rate. If the learning rate is too big, it will decrease rapidly at the beginning but may be difficult to find the best solution later. Hence, I increase the number of epoch to let the network to make useful prediction.

2.4.3 Implementation of Network Reduction Technology

In the implementation, I calculate the angles of each pairs of vector and store them in a matrix where column means the index of a vector and row means the index of another vector. To determine the undesirable hidden units, I create a set to store the index of deleted hidden units. For each pair of hidden units if any of hidden units are in the deleted hidden units set, no further operation will be proceeded since if one of the hidden unit has been deleted, the other hidden unit has no similar or complementary with it. Hence, only the pairs of hidden units which both of them are not in the deleted hidden units set will be measured. If two hidden units are similar, one of them will be deleted and add its weight to the remaining hidden units. If two hidden units are complementary, both of them will be deleted.

In the previous implementation, instead of actual pruning the unwanted hidden units, I set the weight and bias of deleted hidden units as 0. The connections between neurons still exist. Thus the time cost does not decrease. Furthermore, that hidden units which weight is set to be 0 will still be modified if a retraining process after pruning is applied. Therefore, I deleted the hidden units by creating a new neural network module with the number of remaining neurons after pruning as the number of hidden neurons, then copy the parameters of remaining neurons to the new network module.

Although Gedeon and Harris [3] states that retraining is not often required using network reduction technology based on distinctiveness, in practice, I find out that if the compression rate is above 50, then a retraining processing should be applied to compensate the loss of deleted hidden neuron. Thus, I retrain the net with an epoch of 800 (one-tenth of the original epoch) after prune the neural network.

2.4.4 Implementation of Genetic Algorithm

The GA is implemented based on Deap library. In the implementation, I use floating-point representation to encode the parameter since the angle of two vectors is a continuous value. Intuitively speaking, if the angle between two vectors is greater than 90 degree, it is unusual to define this pair of hidden units as similar. Likewise, if the angle between two vectors is less than 90 degree, we may not say this pair of hidden units as complementary. Therefore, I add a distance function as the penalty function to constrain the evolution. The formula for penalty function is as below:

$$\text{penalty}(x, y) = -(x - x_0)^2 - (xy - y_0)^2 \quad \text{if } x > 90 \text{ or } y < 90 \qquad (2)$$

Where x refers to the first parameter of the individual (similarity angle), y refers to the second parameter of the individual (complementary angle), x_0, y_0 refers to the approximate centre of the valid zone. It this case, x_0 is set to be 45 (mean of 0~90) and y_0 is set to be 135(mean of 90~180). This penalty function uses quadratic distance to give a fitness disadvantage to the individuals violate the constraint.

As I mentioned in the sector 2.3, optimise pruning is a multi-objective optimisation problem. Therefore, the fitness function is as below:

$$\text{Fitness(i)} = A\text{cc(i)}, \text{deleted(i)} \quad (3)$$

Where i refers to the individual in the population, Acc refers to the accuracy of the reduced neural network after applying network reduction technology with the individual i and Deleted refers to the deleted hidden neuron after applying the network reduction technology.

As for the operators during GA, Deap provides a variety kinds of methods. After several experiments, I choose Blend crossover as the crossover operators and multi‑polynomial mutation as the mutation operation. As I mentioned below, NSGA-II is applied as the selection operator.

Since the initialisation of hyperparameters varies from problem to problem, for the setting of the initial population, generation, mutation rate and crossover rate, I initially set them as 10, 200, 0.8, 0.01 following the initialisation on Alam, Das & Pant work [8]. Then I experiment several times with a change in those values and check the performance. The result shows that setting initial population, generation, mutation rate and crossover rate as 20, 400, 0.8, and 0.2 has a better performance.

2.5 Performance Evaluation Methods

To evaluate the performance of network reduction technique using GA, I compute the compression rate as it is a direct measurement to evaluate the network reduction. To evaluate the performance of neural network before and after the network reduction technique, I compute the accuracy, sensitivity and specificity. Sensitivity measures the extent to which real Spam are identified, and specificity measures the extent to which regular email is identified. For Email spam detection system, specificity is more important than sensitivity since people may tolerate with junk email, but cannot accept the fact that system classifier their regular email as spam. The formulas to compute accuracy, sensitivity, specificity are as below

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$Sensitivity = \frac{TP}{TP + FN} \quad (5)$$

$$Specificity = \frac{TN}{TN + FP} \quad (6)$$

Where TP is short for true positive, TN is short for true negative; TP is short for false positive and FN is short for false negative.

3 Results and Discussion

3.2 Original Network Reduction Result and Analysis

Table 1. Test results before and after original network reduction with five simulations

	Compression rate (%)	Accuracy (%)			Sensitivity (%)			Specificity (%)		
		Before	After	Retrain	Before	After	Retrain	Before	After	Retrain
1	55.5	88.47	69.18	88.80	89.56	23.90	82.69	87.73	99.81	87.36
2	65	88.42	79.77	88.30	91.16	96.65	92.99	86.72	69.26	85.96
3	50	90.35	75.76	90.77	85.87	38.59	86.41	93.16	99.15	93.50
4	45	92.98	87.53	92.43	89.97	70.82	88.45	94.73	97.19	94.73
5	50	92.14	88.84	91.92	95.36	96.99	95.63	90.09	83.65	89.57
overall	52.5	90.47	80.21	90.44	90.38	65.39	89.23	90.48	89.81	90.2

Table 1 shows the test results before and after network reduction using the original heuristic (15-165) with five simulations. From the results, we can see that there is an approximately 52% compression on the network. Despite that, all the three measurements got a considerable loss after applying network reduction technology and result in a minor loss after the retraining process.

According to Gedeon and Harris [3], the removal of the hidden unit based on distinctiveness will not influence the accuracy of the neural network. However, from the table above we can see that the removal of the hidden units does influence the performance of the neural network without retraining. One possible reason is that the original heuristic is too rigid in measuring the similarity and complementation of the hidden units. That is, it may consider the unique behaviour pair of hidden units as similar or complementary one and prune them. One way to handle this problem is to use adaptive parameters in network reduction technology (Genetic Algorithm will be mentioned in section 3.2). Another possible reason is a large number of deleted hidden units, from the table we can see that the compression rate is above fifty percent. This higher percentage of compression rate may deepen the impact of inaccuracy coursed by pruning the neurons. To support this reason, a retraining process is applied to compensate the inaccuracy after the network reduction. It turns out the accuracy and specificity are approximately the same after comparing to the accuracy and specificity before the pruning whereas the sensitivity gets a minor decrease.

3.2 Network Reduction with GA Result and Analysis

Table 2. Test results before and after network reduction using GA with five simulations

	Compression rate (%)	Angles (degree)		Accuracy (%)			Sensitivity (%)			Specificity (%)		
		Similarity	Complementation	Before	After	Retrain	Before	After	Retrain	Before	After	Retrain
1	90	70.99	122.69	88.47	89.47	89.14	89.56	82.69	93.41	87.73	94.05	96.25
2	80	67.02	167.38	88.42	89.18	88.3	91.16	90.24	94.51	86.72	90.13	84.44
3	75	0	148.94	90.35	60.52	87.47	85.87	0	80.43	83.16	50.34	92.43
4	85	61.92	151.64	90.96	89.46	90.57	94.13	95.31	95.01	89.03	86.39	88.46
5	65	28.53	164.94	92.14	89.57	92.03	95.36	97.57	95.63	90.09	81.22	89.74
overall	79	45.70	151.12	90.07	83.84	89.50	91.22	73.16	91.80	87.35	90.36	90.26

Table 2 shows the test results of applied GA to modify the network reduction. From the table, we can see that the compression rate is significantly large compares to the compression rate using an original heuristic (52.5%) with the accuracy, sensitivity and specificity stay approximately the same except the third simulation.

In the third simulation, the test results show that all the accuracy, sensitivity and specificity get a significant decrease. One possible reason for that is the premature convergence in GA. Since the scale of the population is limited, after the population evolved over several generations, the individual with high fitness will occupy the majority of the population while other individuals will disappear. Therefore, the types of individual will become monotonous and make inbreeding with the group result in the algorithm jump into the local optimal solution. This premature convergence connects to the diversity in the initial population [9], different settings of the initial population will be experimented in the future work.

The table also shows that the angles found out by GA in each simulation is different. There are two possible reasons, one reason is the premature convergence in GA as mentioned above, and the angles we find may be the local optimal solution. Another reason is the weight matrix used in the implementation. In the implementation, only the output weight of the hidden neuron is used to form the behaviour vector, and it may be incomplete to evaluate the distinctiveness of hidden neuron. Different weight matrix such as input matrix or combination of input and output weight will be tested in the future work.

As for the time cost of GA, a comparison of the time cost of original network reduction technology and network reduction technology using GA is as below:

Table 2. Comparison of time cost between the two technologies with five simulations

	Time Cost (s)	
	Original	GA
1	62.58	289.87
2	61.48	299.34
3	65.89	310.14
4	63.22	240.09
5	60.16	296.37
Overall	62.67	287.16

From table 3, the average time cost using GA is four times than using the original heuristic. It is coursed by the slow speed of convergence in GA. Although optimising the hyperparameter of GA can reduce the time cost to some extent, it is no doubt that GA cost a lot more time than the original heuristic and the larger the original neural network, the more significant the difference of the time cost.

3.3 Results of Other Techniques Using Same Data Set

Singh, Chand and Lal [6] have developed a neural network a trained by Memetic Algorithm (MA) to detect the email spam. In the paper, they use MA to train a three-layer ANN and test it on the spambase dataset. The compared test results between ANN using MA and ANN with the reduced hidden unit technique using GA is shown as below:

Table 4. Compared test results with two techniques

	Hidden Neuron	Accuracy (%)	Sensitivity (%)	Specificity (%)
MA	19	94.24	91.74	95.70
Network Reduction with GA	4	90.01	94.64	89.72

From table 4, we can see that using ANN with MA has better accuracy and specificity than using ANN with network reduction technology using GA but has a worse sensitivity and more hidden neuron. As the email spam detection cares more about the specificity (less error in detecting normal email), training ANN with MA has a better performance in the spam detection problem compares to the implementation in this report. One possible reason is the different algorithm in the training process. Singh, Chand and Lal [6] use MA to train the ANN whereas BP with Stochastic Gradient Descent is used in the training process in the report. A significant disadvantage of BP with Stochastic Gradient Descent is that it often jump into local minima. This may result in the performance of the ANN before applying network reduction technology is worse than the performance of ANN with MA. As the network reduction technology does not increase the performance after pruning the network, the performance of using network reduction with GA is worse than the ANN with MA.

It is acceptable for the input features reduction technique has better performance since the aim of network reduction technique is to reduce the size of the network to decrease the memory and time cost. From time four we can see that ANN with MA has a larger number of hidden neuron compares to the network reduction technology with GA. The result supports the view that using Network Reduction Technology with GA can reduce the memory and time cost of ANN.

4. Conclusion and Future work

In this report, I implement a neural network to detect email spam and use network reduction technique based on distinctiveness to reduce the hidden units and use sensitivity and specificity as two major performance measurement method. After that, GA is applied to optimise the network reduction technique, and results of two network reduction technology show that network reduction technique with GA has a better performance than the original network reduction technique. Although the actual test results are not as good as the theory as the sensitivity and specificity will still decrease after applying network reduction technique and retraining, this report shows that this technique can apply to the real-world problems.

For the future work, this network reduction technology is only tested on the single dataset. It needs to be tested on different kinds of the dataset to support this technique. Furthermore, it will be interesting to extend this neural network technology from ANN to Convolutional Neural Network or Recurrent Neural Network.

In this implementation, I only use output weight of the hidden units to measure the distinctiveness. It is worth testing the different performance of using input weight of the hidden units to prune the neural network. Besides, only the distinctiveness between the pair of hidden units is measured in the implementation. The distinctiveness among multi hidden neurons can be used to see if it can improve the compression rate of the network reduction technique.

According to the result, the implementation of GA sometimes get a worse result, and it may be caused by the premature convergence problem. It is worth investigating a way to find the optimal GA structure such as using uniform design method.

References

1. Holland, J. H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press. (1992).
2. Dua, D. and Karra Taniskidou, E. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2017).
3. Gedeon, T. D., and D. Harris. "Network reduction techniques." Proceedings International Conference on Neural Networks Methodologies and Applications. Vol. 1. (1991).
4. Sabo, Devin, and Xiao-Hua Yu. "A new pruning algorithm for neural network dimension analysis." Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. IEEE (2008).
5. Mulligan, G. Removing the Spam: E-Mail Processing and Filtering. Addison-Wesley Longman Publishing Co., Inc. (1999)

6. Singh, S., Chand, A., & Lal, S. P. Improving spam detection using neural networks trained by memetic algorithm. In Computational Intelligence, Modelling and Simulation (CIMSim), 2013 Fifth International Conference on (pp. 55-60). IEEE. (2013)
7. Heaton, J. Introduction to neural networks with Java. Heaton Research, Inc... (2008)
8. Alam, M. N., Das, B., & Pant, V. A comparative study of metaheuristic optimization approaches for directional overcurrent relays coordination. Electric Power Systems Research, 128, 39-52. (2015).
9. Srinivas, M., & Patnaik, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 24(4), 656-667. (1994).