Progressive Distinctiveness Based Reduction for Multilayer Neural Network with Unbalanced Data

Jiamou Sun

Research School of Computer Science, Australian National University,
2601 Canberra, Australia
{u5871153}@anu.edu.au

Abstract. The neural network is an influential classification and regression tool as it does not require the apparent rules from data. One of the defects of the neural network is that the performance is limited by the calculation complexity. The objective of this paper is to apply a progressive distinctiveness based pruning method to reduce the neurons of multi-layer neural networks. Some unbalanced data is chosen and normalized for the real dataset can be unstructured and noisy. The non-normalized dataset and non-progressive pruning method are used as the baselines. From the results, the normalization techniques and the progressive method can hugely improve the function and stability of the network. Before some certain pruning thresholds, the accuracy of pruning network keeps stable while exceeding the thresholds can cause significant reduction of the accuracy.

Keywords: multi-layer network pruning, neural network, distinctiveness based pruning, data normalization

1 Introduction

Traditional ruled based classification method can be hard for application because, in some situation, it is impossible to figure out the rules from the noisy dataset. Compared with it, the neural network has a significant advantage for it can classify data only with the label, and it has good generalization to different kinds of data. It has been used to solve many problems, including solar irradiance prediction and firm rating [1], [2]. However, neural network is criticized for its computation complexity. Although multi-layer network has excellent function potential, it may require long time and more data to train, which is unaffordable for the application under some situation [3], [4]. Fortunately, lots of pruning techniques for reducing the size of the network have been found out, and the distinctiveness based method is one of them. It was made by paper [5], showing good performance for a simple 3-layer network. This method was also applied in many projects to successfully improve the function of the networks [6], [7], [8].

In my previous work, some characters, including the usefulness, of the distinctiveness based pruning have been proved for the single-hidden-layer network [16]. However, many people prefer to use multilayer networks for its good potential to solve problems. Therefore, in this paper, a progressive distinctiveness pruning based method is made for the more complex networks. The multi-hidden-layer pruning is different from the single-hidden-layer network, which can cause accuracy reduction for standard pruning method. During the process of experiment, some of the characters of this kind of technique are revealed. An unbalanced dataset is chosen and normalized because, in the real problem, we cannot approve the cleanliness of data. The non-progressive method and non-normalized data are applied as the baseline in the experiment, and the cross-validation is used to test the accuracy.

The comparison of different pruning techniques and the method of this paper is in section 2, the experiments and the results are in section 3, and the conclusion and discussion are in section 4.

2 Method

2.1 Related Works

Pruning is a popular topic, and many methods have been researched to reduce the size of the neural network. The *sensitivity*, *badness* and *distinctiveness* are three of essential pruning methods.

The *sensitivity* method applies the sensitivity value, which is related to the derivative of error and changes of weights, to remove the nodes. The specific method is described in the paper [9]. The *sensitivity* of a network node is defined as follow:

$$\hat{S}_{ij} = \left[\sum_{0}^{N-1}\left[\Delta w_{ij}(n)\right]^2\right]\frac{w_{ij}^f}{\eta(w_{ij}^f - w_{ij}^i)} \quad (1)$$

The N is number of epochs, the $\Delta w_{ij}(n)$ is the derivative of error, the w^i_{ij} is the initial weight, and the w^f_{ij} is the final weight. By calculation of the sensitivity of the network after training, the weight with least sensitivity value will be removed. The advantage of this technique is that it has little computational overhead as it is a cumulative calculation process [5]. Nevertheless, it has defects. The measurement only considers the independent weights but ignores the effect of multiple weights [10]. If two weights are contradictive to each other, but each of them has high sensitivity, this method will lose efficacy.

The *badness* method is similar to the *sensitivity*, but it focuses on pruning the neurons rather than weights. It is first mentioned in the paper [11], and the formula is presented below:

$$Bad_j^n = \sum_p \left(\sum_k w_{j,k}^{n,n+1} \times \delta_k^{n+1} \right)^2 \quad (2)$$

The p is pattern, and the w is weight of the j^{th} unit in layer n that is linked to layer $n+1$ with k units. After the training, the unit with largest badness units will be eliminated. The *badness* method is simple and can accelerate the speed of convergence of the network [12]. However, as discussed above, this method also only considers the effects of single units rather than the co-effects. Besides, according to paper [5], this method lacks the theoretical support of its rationality, because if the badness value is high, then the adjustment of weights will also be great, which is a good appearance. Thus, this measurement may not get network to the best situation.

This paper applies the *distinctiveness* to reduce the size of network. It is first mentioned in paper [5]. The *distinctiveness* will first calculate the angle between activation outputs of every hidden layer units in pattern space. Then, if some of the angle are too similar or too different, the related units are considered redundant or contradictive, so the unsuitable units will be eliminated. In our previous work, the *distinctiveness* method is proved effective in single hidden layer network within small thresholds [16]. In this paper, the effect of it will be tested in multi-hidden-layer network, and the method will be adjusted to fit the changes of network construction.

2.2 Method Architecture

Fig.1 shows the architecture of the method of this paper. This paper chooses the payment data of Taiwanese customers to predict whether they will pay in the next month. It is first used in paper [13], which will also be applied as baseline in the experiment section. First, the noisy data will be adjusted and normalized by some techniques. Then, a feed-forward neural network with 2 hidden layers is established and used to predict the outcome. The weights are updated by batches rather than the patterns for this method can prevent the network from overfitting. After the training, the angles of second hidden layer's neuron output activation vectors over the pattern presentation are calculated, and the neurons with similar or contradictive function is pruned from the network. Because the second hidden layer's output depends on outputs of previous hidden layers, the multilayer network is more sensitive than the single hidden layer network. Hence, the network is fine-tuned with another 200 epochs before the pruning operation of the first hidden layer. The order of pruning layers is very essential, in this paper, the second hidden layer will be pruned first.

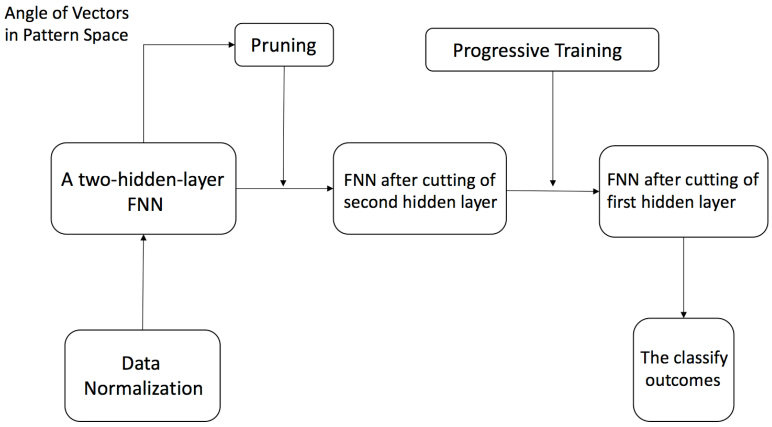


Fig. 1. The mainstream of the progressive pruning

There are 23 attributes in the dataset so the number of input neurons is 23. The first hidden layer contains 92 neurons and second hidden layer contains 23 neurons. It is 2 classification problem so the output layer contains 2 neurons. The *sigmoid* function is applied as the activation function for the hidden layer, and the *softmax* function is applied in the output layer. I make use of decay-learning rate strategy because this can avoid excessive learning rate and fluctuation when the network begins to converge. The initial learning rate is 0.3, and it will halve every 400 epochs. The total training epochs before pruning operation is 1000. The threshold of pruning angle is set to 15 degrees. The specific details of the performance evaluation are in the section 3.

2.3 Data Preparation

Default of Credit Card Clients Dataset, which is found in UCI repository is used as the test set [14]. It wants to apply 23 attributes to predict whether customers will make the payment next month or not. The attributes and their scopes are listed in the table below:

Table 1. Attributes of the dataset

Attributes	Min	Max	Scope
Amount of the given credit	10000	1000000	0+[1]
Gender	1	2	1-2
Education	1	4	1-4
Marital status	1	3	1-3
Age	21	79	0+
History of past payment (from April to September)	-2	8	-2-8
Amount of bill statement (from April to September)	0	1664089	0+
Amount of previous payment (from April to September)	0	1684259	0+
Default Payment	0	1	0-1

It is evident that this data is very unbalanced. By statistics, there are totally 30000 patterns. Among them, the amount of class 0 is 23362, while the number of class 1 is 6638. This kind of data distribution can cause network learning too many features about 0 class, so the risk of misjudgment is high. Another problem of dataset is that most of its figures do not have boundaries. Some of the numbers are too high, while some numbers including education, age are only between 0 to 4. This considerable difference can cause longer time for training the network, and the effect of outliers will be more prominent.

To solve those problems, first the *oversampling* technique is applied. The ratio of two classes is nearly 3:1, so the data of class 1 is duplicated two more times to make the ratio of data balanced. For the problem of figure scopes, the *min-max normalization* technique is applied to the attributes whose scope exceeds 4. The formula for it is shown below:

$$x^* = \frac{x - min}{\max - min} \tag{3}$$

By using this method, those big figures are mapped to the scope of 0 to 1, so the data becomes more structured. The effect of data preparation is tested in section 3.1.

2.4 Progressive Pruning Method

To analyze the effect of pruning, the formula of output of hidden neurons is shown below:

$$O_{\text{hi}} = \sigma\left(\sum_{0}^{N} w_i x_i + b_i\right)$$

[1] There is no upper bound.

(4)

As what paper [5] illustrates, the distinctiveness method adopts a pruning recovering method to decrease the effect of pruning. The recovery technique is that every time the neuron is eliminated their weights will be added to the remaining neuron. From the formula above, if the eliminated neuron's weight and output are w_1 and x_1, and the left neuron's weight and output are w_0 and x_0, then their effect to the next layer's neurons is presented below:

$$O_{\text{hi}} = \sigma\left(\sum_{2}^{N} w_i x_i + (w_0 + w_1)x_0 + b_i\right) \tag{5}$$

Hence, if two neurons' functions are the same, which means their output figures are the same, the pruning process will not affect the input of next layer's neurons. However, the neurons with similar outputs are no equal to the two outputs are same. Besides, the neurons with contradictive function are also considered to be pruned. Thus, this recovering method is not perfect, so the *distinctiveness* pruning method can also cause the change of accuracy to the network. By the experiments of paper [5] and [15], this kind of flaws will not affect the simple hidden layer network hugely. Nevertheless, for the network with two or more hidden layers, the effect is very dramatic. Each hidden layer's input depends on the previous hidden layer's output, which causes the network fragile for the loss of units. If all hidden layers' neurons are pruned together, the accuracy will decrease a lot. Moreover, the order of pruning is also essential, different pruning order of hidden layers can lead to different results, so the performance of different kinds of pruning methods are compared in the experiment section.

To overcome this defect, a progressive strategy is used. The network is pruned layer by layer. After one-layer pruning, the network will be training for another 200 epochs. This can make network recover itself after the elimination, so the pruning for next time will not affect the accuracy too much.

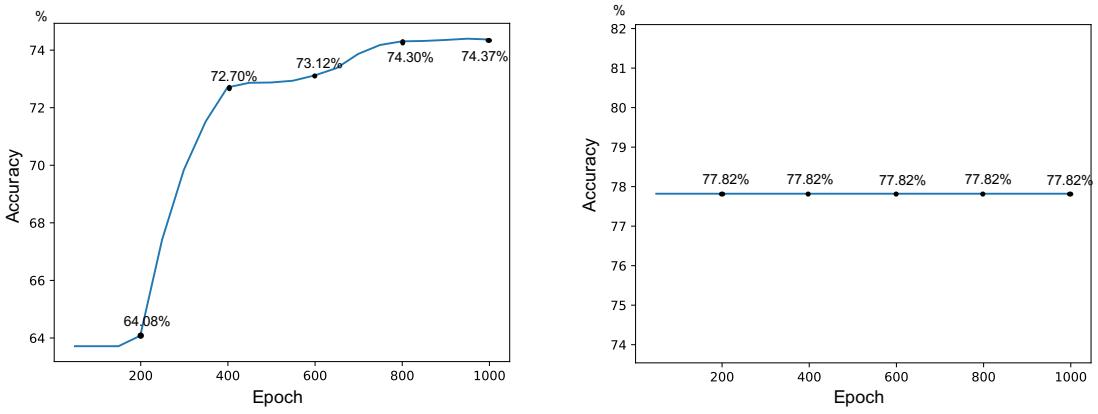
3 Results and Discussion

3.1 Total Accuracy

Through the 5-fold cross-validation, the final accuracy of the network without pruning is 74.46%. Compared with paper [13], whose accuracies were between 74% to 84%, this result is a little bit lower. Note that original paper applied many kinds of classification methods so it had lots of accuracies. However, there was no data normalization techniques in the original paper, and the writer admitted that the data was very unbalanced, so it was not suitable to use error rate as the measurement. Instead, the writer applied a measurement named *Area Ratio* to measure the effect of different kinds of classification techniques, which are hard to compare with the result of this paper.

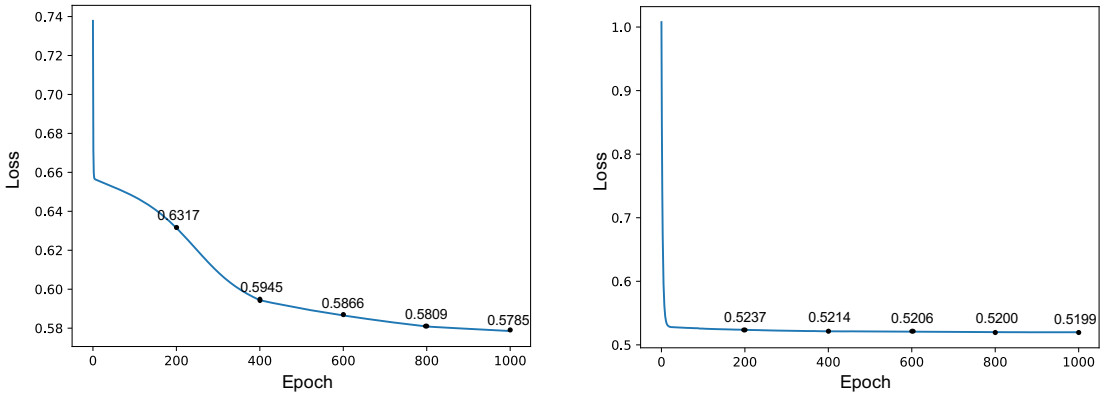
3.2 Data Normalization Assessment

To test the effect of data normalization step, the training result with non-normalized data is used as baseline. The outcome is shown below:



(**a**) Accuracy of normalized data (**b**) Accuracy of non-normalized data

Fig. 2. Accuracy of normalized and non-normalized data training



(**a**) Loss of normalized data

(**b**) Loss of non-normalized data

Fig.3. Loss of normalized and non-normalized data training

As figures show above, network with normalized data can normally work, while the network with non-normalized data totally loses the function, so normalized method in this paper works well for this kind of data. The Fig.2 (b) presents that the accuracy with training sets, which is as high as 77.82%, does not change along with training steps. This happens because the ratio of the negative class and positive class in original data is nearly 3:1, which can lead network only to learn the negative class's features. Thus, the network failed to recognize different patterns. This can be found by looking at Fig. 2 and Fig.3. If the outcome is always 0, by calculation, the result will always be 77.82%, which is the same to the experiment result in Fig.2. Fig.3 (b), which shows the loss of two kinds of training results with unbalanced data, drops sharply at the beginning, and then hardly changes. Because the outcome is always 0, the loss of network drops rapidly at the beginning. However, for most of the data has the features of class 0, it is difficult for network to learning the right features of another class. As the results, the loss decreases slowly with further training, which indicates that the training is utterly useless under the unbalanced data.

To further observe the effect of unbalanced data, the confusion matrix is applied, and the results are presented below:

Table 2. Confusion matrix of two kinds of results

(**a**) The result of normalized data

	Condition Positive	Condition Negative
Test Positive	1146	422
Test Negative	1491	4268

(**b**) The result of non-normalized data

	Condition Positive	Condition Negative
Test Positive	0	0
Test Negative	1313	4687

Those two tables further support the observation above. The Table 2(b) illustrates that the output of network is always 0, so without data normalization, the network cannot work normally. Table 2(a) shows the result of the training after the data preparation. From it, it is obvious that the training effect has been improved a lot. With *oversampling*, the data is more balanced so the network can start to learn the features of another class. With the *min-max normalization*, all the larger data are limited between 0 to 1. Because the outputs of sigmoid activation function have bigger difference in the domain between 0 to 1 than the domain with larger numbers, the normalization can make the features of inputs more prominent, so it is easier for network to learn. Besides, if some inputs are too big, the fixing learning rate may be too bigger for the nodes with big inputs than the nodes with small inputs in the back-propagation steps, so the converging will be slow. The *min-max normalization* makes all the inputs in the similar in similar domain, so the fixing learning rate can have similar effect to each weight in each node, and the network will converge quickly. Therefore, overall, the data normalization methods in this paper is helpful to deal with the unbalanced data.

3.3 Pruning Assessment

3.3.1 Accuracy assessment. To test the effect of the progressive pruning method in the paper, different methods are applied to the 5-fold cross-validation, and their results are compared. The comparison is illustrated below:

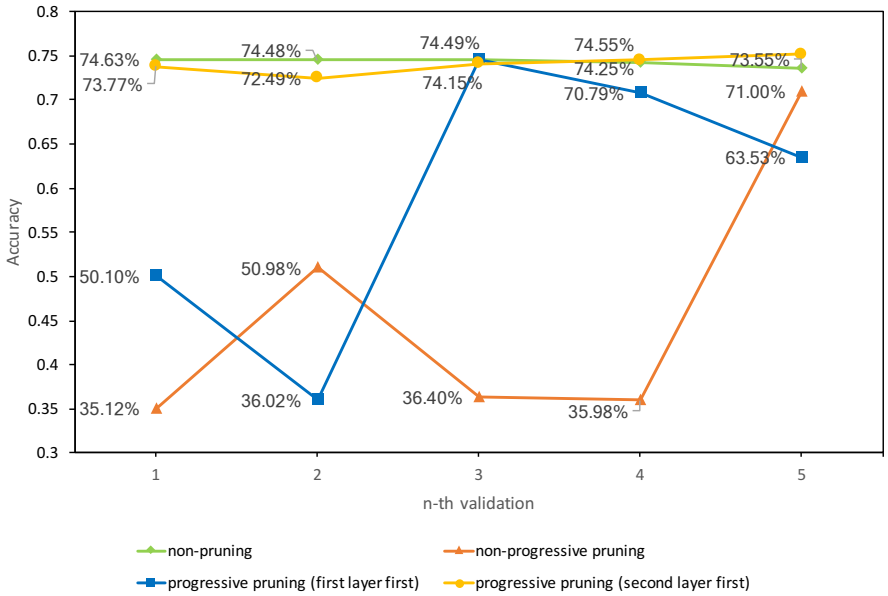


Fig. 4. Comparison of different pruning methods

As it presents, the accuracy of network without pruning, which is shown in green line, is high and stable in this figure. Compare with it, the progressive pruning cutting second hidden layer first, which is shown in yellow, also performs good and stable. However, the network with cutting first hidden layer first pruning and the network without progressive pruning method have poor performance. Their accuracy in average is lower than the other two methods, and they are very unstable for the number of accuracy fluctuate up and down. For the non-progressive pruning method, the effect is bad because the multi-hidden-layer network can be more sensitive to the pruning than the single-hidden-layer network. For the network with multi-hidden layer, the higher level hidden layers have the high concentration characters from the features of bottom hidden layers. Although in ideal state, which means the activation output of pruning nodes are all same of contradictive to each other, the outcome will not be affected. Nevertheless, the threshold is 15 degrees, so the pruning method can always make some changes to the accuracy. If nodes of those higher layers are eliminated, then the highly extracted features can be lost, causing huge decreasing of accuracy of the network. This is also the reason why the pruning method cutting the first hidden layer first cannot perform well comparing to its counterparts. Although there is progressive recovering, only the first hidden layer can get benefits from it. Hence, the progressive pruning method with second hidden layer cutting first can keep the accuracy of the original network.

3.3.2 Stability Assessment. To test the stability of neural network under different thresholds, several 5-fold cross-validation tests are applied, and the average results are shown in the table below:

Table 3. Confusion matrix of two kind of results

Minimum Angle	Accuracy (non-pruning)	Accuracy (pruning)
5	74.18%	74.42 %
15	74.46%	74.02 %
25	74.63%	70.75 %
35	74.31%	65.41 %
45	74.03%	59.57 %
55	74.40%	54.20 %
65	74.25%	62.38 %
75	73.48%	57.81 %

Table 3 presents that with the increase of the threshold, the pruning accuracy decreases, which is similar to what has been gotten in my previous work [16]. There are some figures that do not conform the decrement law, one reason is that the weights of pruning network are different for each test, which means the training of network is a stochastic process, so the accuracy can be high after pruning; another reason is that the network is trained for a two-element classification

problem, so if the outputs are all 0 or 1, the accuracy can still be 50%, but this does not mean the effect of network is good. To further test the performance of the network, the f1-score measurement is applied, the formula is shown below:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{8}$$

The precision is the ratio of the true positive samples to the all positive samples, and the recall is the ratio of the true positive samples to all the true samples. It can be used to measure the quality of the network. If all the outputs are 0, then the accuracy will not drop below 50%, but the f1-score will be 0. The result is listed below:

Table 4. Confusion matrix of two kind of results

Minimum Angle	F1-Score
5	0.567
15	0.561
25	0.554
35	0.529
45	0
55	0
65	0.007
75	0

Combining Table 4 with Table 3, the network still works well before the threshold of 25, but with decreasing of thresholds, it will have a very poor performance. This pattern is similar to the single hidden-layer network, and it proves that the progressive method in this paper will not affect the performance of the network greatly before the certain thresholds.

3.4 Overfitting Assessment

For checking whether the network is overfitting or not, the final result of the network with test sets is compared with the results with training set. Under the threshold of 15 degree, the accuracy for training set is 74.37%, and the accuracy for test set is 73.89%. There are only 0.48% percent differences. Thus, the network in this paper is not overfitting.

4 Conclusion and Future Work

This paper focuses on pruning of multi-hidden-layer networks with unbalanced data. The *oversampling* and *min-max normalization* techniques are applied to reduce the effect of unbalanced data and are proved to be useful by the experiments. After the error becoming stable, the second hidden layer of neural network will be cut first, then a progressive training with 200 epochs will be applied to recover the influence of the pruning step. Finally, the first hidden layer is pruned and there is no further training required. Through experiments, the progressive pruning method has great effect compared to non-progressive pruning techniques. With small angle threshold, the pruning will not affect the function of network greatly, so this method is also useful in the multi-layer network and can observably reduce the size of units.

Nevertheless, there are still many works to do. First, the final outcome compared to the paper [13] is still not ideal, which means there is still space for the data normalization. Second, this method only considered the similarity and contradiction between pairs of neurons. However, the bunch of units can also have same of contrary functions, creating redundancy to the network. Thus, in the future work, more data normalization techniques can be tested to different kinds of noisy data to find the better solutions to clean the data, and the function of bunch of neurons can also be researched to improve the performance of the network pruning.

References

1.Mellit, A., Pavan, A.: A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy. Solar Energy. 84, 807-821 (2010).

2.Albino, V., Garavelli, A.: A neural network application to subcontractor rating in construction firms. International Journal of Project Management. 16, 9-14 (1998).

3.Huang, G., Zhu, Q., Siew, C.: Extreme learning machine: a new learning scheme of feedforward neural networks. 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541). (2004).

4.Heermann, P., Khazenie, N.: Classification of multispectral remote sensing data using a back-propagation neural network. IEEE Transactions on Geoscience and Remote Sensing. 30, 81-88 (1992).

5.Gedeon, T., Harris, D.: Progressive image compression. [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. (1992).

6.Lewis, J.: Probing the critic: approaches to connectionist pattern synthesis. IJCNN-91-Seattle International Joint Conference on Neural Networks. (1991).

7.Namphol, A., Arozullah, M., Chin, S.: Higher order data compression with neural networks. IJCNN-91-Seattle International Joint Conference on Neural Networks. (1991).

8.Cottrell, G., Munro, P., Zipser, D.: Learning internal representations of gray scale images. (1987).

9.Karnin, E.: A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks. 1, 239-242 (1990).

10.Reed, R.: Pruning algorithms-a survey. IEEE Transactions on Neural Networks. 4, 740-747 (1993).

11.Hagiwara, M.: Novel backpropagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection. 1990 IJCNN International Joint Conference on Neural Networks. (1990).

12.Jearanaitanakij, K., Pinngern, O.: An Information Gain Technique for Acceleration of Convergence of Artificial Neural Networks. 2005 5th International Conference on Information Communications & Signal Processing. (2005).

13.Yeh, I., Lien, C.: The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications. 36, 2473-2480 (2009).

14.Dua, D., Karra Taniskidou, E.: UCI Machine Learning Repository, http://archive.ics.uci.edu/ml.

15.Gedeon, T.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. (1995).

16.Sun, J.: Application of Network Reduction in Handwritten Letter Recognition Problem. ABCs2018. (2018).