Network Reduction Based on Hidden Neuron Distinctiveness

Jun Ji

Research School of Computer Science, Australian National University,
Canberra ACT, Australia

u6268344@anu.edu.au

Abstract. Neural Network using feed forward and back propagation for training could result in slow performance issue because the back propagation algorithm passes the error backwards from output to input and does the calculation for each neuron in the network. This performance issue could become more serious when the network turns to have lots of neurons and cost longer training and prediction time. Meanwhile, some of the neurons in the network are not necessary to maintain the prediction accuracy. To solve this, approach suggested in this paper is to determine the unnecessary neurons by comparing their functionalities when facing different patterns, then remove the neurons having similar functions. Results based on the MNIST Gisette data set proved that after pruned unnecessary hidden neurons, the accuracy of the network have not been affected.

Keywords: Classification, Neural network, Network pruning, Neuron distinctiveness, Convolutional neural network, Feature extraction.

1 Introduction

There are debates around the number of neurons should be used in a neural network, some researchers state to add more neurons when the network is not performing well. However, some of the neurons may not be necessary to perform good prediction accuracy, more seriously, too many neurons may even result in overfitting issue. Therefore, we need to find a solution to detect the duplicate neurons and then remove them from the network to see if the network performance stays the same. In this paper, I am using distinctiveness (Gedeon, 2010) to determine unnecessary hidden neurons in the network.

1.1 Choice of Data Set

The data set used in this paper is the 'MNIST Database of handwritten digits' (LeCun, 2008). The data set has a training set of 60,000 examples, and a test set of 10,000 examples. Results in the paper 'Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark' (Guyon, 2007) are used here as reference.

The network inputs are images of size 28 x 28 pixel. Those images are digits written by 500 different writers consist of high school students and company employees, which increases the diversity of the data set. The network outputs are digits corresponding to the images. After training, the network should predict what is the digit in the input image as output.

Because the emphasis of this paper is to check if the network could perform the same after pruning, but not compete with the existing results, for the efficiency purpose, I am using part of the data set. Based on the results in published papers, it takes more than 2 minutes to train and verify a network with entire data set. With the idea from the same resource on UCI Machine Learning Repository website, which consists of only two kinds of digit, highly confusable digits '4' and '9', the number of examples are reduced to 13,500.

1.2 Investigation Outline

The aim of the investigation is to prove that when there are hidden neurons with similar functionalities in the network, removing the 'duplicated' neurons won't decrease the accuracy of the network.

To illustrate this, a convolutional neural network is implemented along with unnecessary hidden neuron detecting method based on the distinctiveness of neurons. The next few sections will describe the structure of the network used in this paper, method of calculating neuron distinctiveness, and comparison between the network before and after pruning.

2 Methods

Because the data set consists of images, a normal fully connected neural network is not the best option for this task due to the fact that the data in the image is spatially related. In this section, details of the network, method used for network validation and neuron distinctiveness are introduced.

2.1 Network Structure

When facing spatially related data like image, a convolutional neural network is normally more favorable than simple fully connected neural network. The convolutional neural network extracts features from the image for easier

classification when comparing to directly using original image. In this paper, we use two convolutional layers, the first convolutional layer extracts 10 features from an original image with kernel size 5, which produces 10 of 24 x 24 pixel images. After max pooling with kernel 2 to further extract the data, the output size of the first convolutional layer is 10 x 12 x 12. The second convolutional layer extract 20 features from the previous layer with same kernel size 5. Before going into the max pooling, some of the result are randomly zeroed for regularization and preventing co-adaptation of the neurons, this process call dropout. After max pooling with kernel 2, it produces 20 x 4 x 4 size output. Then, the 20 x 4 x 4 matrix is flattened to 320 to feed into the first fully connected neural net, the output size of it is 50. Finally, after dropout, those 50 outputs are feed into the second fully connected neural net for classification. The output is softmaxed to demonstrate the probability of the input belongs to each class. Because we only use two kinds of handwritten digit in the data set, the final output size of the network is two, which means we are doing binary classification. The activation function used in the network is Rectified Linear Unit (ReLU). With ReLU, there is fewer vanishing gradient compared to using sigmoid. Therefore, the back propagation performs better when the network using ReLU.

2.2 Network Validation

Since the data set has sufficient data both for training and testing, besides, the data set is originally separated into training and test set, therefore, I did not use K-fold validation. The training data is fed into the network by batches, data of number '4' and number '9' are picked from the batches for training. The training loss is recorded, to prove the training process is working properly as reducing the loss. Once training is finished, the network is tested by test data set. Like previous, only data of number '4' and number '9' are picked for testing. The stopping criteria for training is set by test accuracy. From the result on the MNIST official website (LeCun, 2008), the average prediction accuracy of using convolutional neural network is around 99.2%, thus I am using this as the training stopping criteria. Once network reached accuracy of 99.2% or above, the unnecessary hidden neurons of the first fully connected layer are determined. Then, it removes different number of neurons start from one to remove all of them, and test the reduced network. Test loss and test accuracy are recorded, to show the impact on removing unnecessary hidden neurons.

2.3 Neuron Distinctiveness

Neuron distinctiveness is defined as the behavior of the neuron when facing different input patterns (Gedeon, 2010). Those behaviors construct a vector of dimension equals to the number of patterns. It is hard to tell the difference just by vector values, but by calculating the angle between different vectors, it is more intuitive to observe that whether two neurons are similar which means having the small angle, or they are opposite which means having the angle close to 180 degrees. To check the angle between pairs of vectors in Euclidean space, use function:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos(\theta), \tag{1}$$

From the data set used in this paper, there are two kinds of patterns: number '4' and number '9'. Therefore, the vector dimension in this case is two. Furthermore, since ReLU is used as activation function in the network, the output can only be zero or positive. This limits the range of the angle between pairs of vectors into 0 and 90 degrees. As a result, there are no opposite neurons in the network.

The threshold for determine similar neurons also need to be set appropriately. In this paper, I will use the validation method mentioned previously to experiment different setting to find the best parameter for this data set.

2.4 Determine Unnecessary Hidden Neurons

When comparing vectors in pairs, it go through all the combinations of vectors. Angles between pairs of vectors shows if they are similar neurons, but among those similar neurons, not all neurons should be removed, one should be kept in the network. Since it is iterating in numeric order, the method used here is to keep the first neuron in the comparison, and remove all the second neuron in the comparison.

3 Results Analysis

3.1 Training and Testing

As mentioned previously, the training data is fed into the network by batches. In one single epoch, the entire training data set is utilized. Since the size of the data set is big enough, normally the training finishes in one epoch, which means the network reaches 99.2% or above accuracy during testing. Figure 1 shows the cross entropy loss recorded during training.

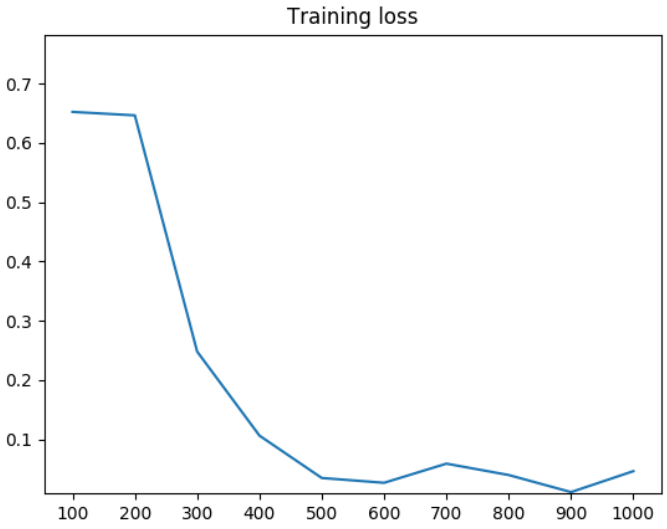

Fig 1. Cross entropy loss during the training

The x axis in Figure 1 is the number of batches, and y axis is the cross entropy loss value corresponding to that batch. It proves that the network is trained to have better performance. The aim of this part is to obtain a network with the similar accuracy level comparing to other networks in published literatures. The comparison between the network in this paper and the convolutional networks in other literatures is in Table 1:

Table 1. Test accuracy comparison.

Literature	Accuracy
This paper	99.60%
LeCun et al.	99.30%
Lauer et al.	99.46%
Simard et al.	99.60%
Ciresan et al.	99.77%

Table 1 shows that the network obtained is good enough to perform the further process.

3.2 Unnecessary Hidden Neuron Detection

By using the neuron distinctiveness, all combinations of the hidden neurons need to be taken into consideration. There are 50 hidden neurons in the first fully connected layer, so only part of the result is shown in Table 2:

Table 2. Hidden neuron vector angles.

Pair of Neurons		Vector Angle
1	3	89.2753
1	4	0.0969133
1	5	8.77615
1	6	77.6165
1	7	42.0997
6	32	76.4108
6	33	12.3263
6	34	77.5105
6	35	76.5986
6	36	3.19666
6	37	12.2686
6	38	5.72858
5	18	1.83984
8	13	0.816613
8	14	77.313
8	15	88.498
8	16	89.5654
8	17	0.0197823
8	18	0.0279765
37	14	0.034571
37	17	89.9175
37	19	0.15017

From Table 2, the vector angle ranges from 0 to 90 degrees as explained previously. After used algorithm and threshold of 5 degrees to select the unnecessary hidden neurons, the filtered vector angle list is shown in Table 3:

Table 3. Unnecessary hidden neuron vector angles.

Pair of Neurons		Vector Angle
1	4	0.0969133
6	36	3.19666
6	38	5.72858
5	18	1.83984
8	13	0.816613
8	17	0.0197823
8	18	0.0279765
37	14	0.034571
37	19	0.15017

From Table 3, we can see after filtering, all pair of neurons in the list has vector angle smaller than 5 degrees. Some of them are even extremely similar, like No. 37 and No. 14 which only has vector angle 0.034571 degree.

3.3 Impact of Removing Unnecessary Hidden Neurons

After obtained the desired network and list of unnecessary hidden neurons, I remove different number of neurons start from one to remove all of them, and test the reduced network. Figure 2 shows the test accuracy and cross entropy loss:

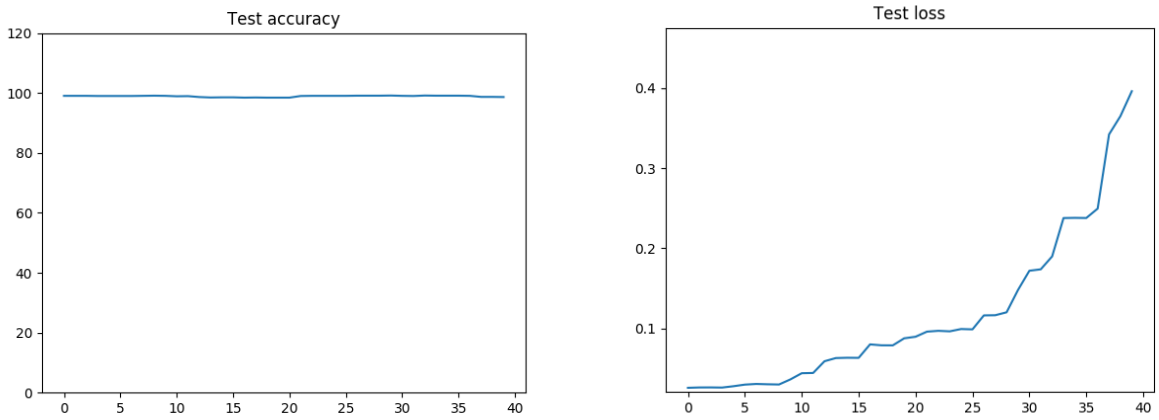

Fig 2. Accuracy and cross entropy loss of removing numbers of neurons (threshold 5 degrees)

The x axis in Figure 2 represents the number of unnecessary hidden neurons removed, and the unit of y axis in Test accuracy graph is percentage, unit in Test loss graph is actual value. From the Test accuracy graph, it shows the accuracy is almost changeless. Even all the 40 unnecessary hidden neurons are removed, the final accuracy is 99.45% comparing to the original 99.60%. But from the Test loss graph, it is clear that despite the cross entropy loss increases slightly, removing neurons does have impact on the network.

The result in Figure 2 is obtained by using threshold of 5 degrees, Figure 3 shows the result of setting the threshold to 30 degrees:

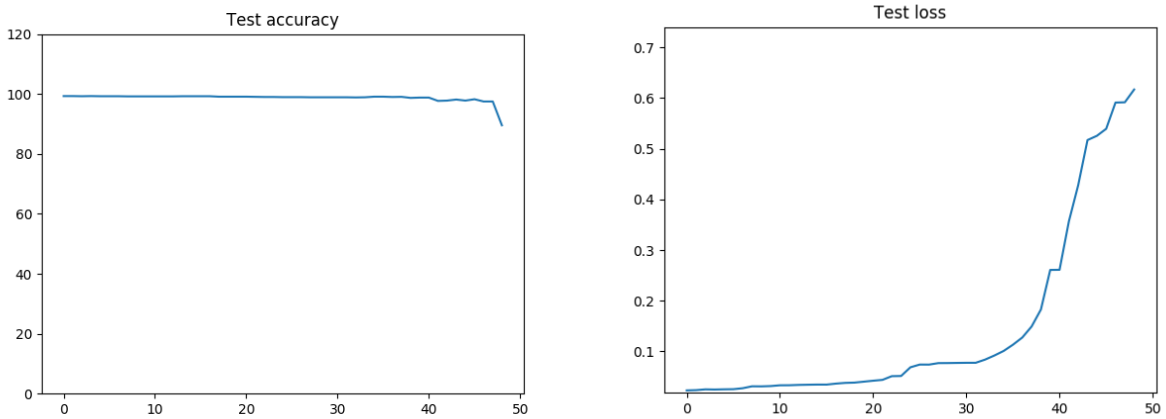

Fig 3. Accuracy and cross entropy loss of removing numbers of neurons (threshold 30 degrees)

In Figure 3, the accuracy drops steeply when removing too much hidden neurons, and values in Test loss are bigger than values in Figure 2. After rounds of experiment, the most suitable threshold for this data set turns out to be 15 degrees, this produce the minimum network size with unaffected prediction accuracy.

4 Conclusion and Future Work

By comparing the neuron distinctiveness, we could successfully identify the unnecessary neurons and reduce the size of the network while keeping the equivalent functionalities. But the threshold for vector angle between neurons need to be carefully defined, otherwise it will end up with an unchanged or even a broken network.

Tool used for implementation in this paper is Pytorch. Due to Pytorch is not allowing neuron removal, the weights of neurons are set to zero to get the equivalent outcome. Because of this, the time efficiency improvement cannot be properly tested in this paper, but it is mostly positive that effective usage of network pruning based on neuron distinctiveness will improve the network efficiency. It is more desirable when the network is utilized in industry with crucial timing requirement. Besides, there are more future works like impact of hidden neuron removal on smaller data sets, etc.

References

1. Gedeon, T. (2010). *NETWORK REDUCTION TECHNIQUES*. London: Department of Computer Science, Brunel University.
2. Guyon, I. (2007). *Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark*. Zurich, Switzerland: Pattern Analysis and Machine Learning Group, Institute of Computational Science.
3. LeCun, Y. (2008, 2 29). *THE MNIST DATABASE of handwritten digits*. Retrieved from THE MNIST DATABASE: http://yann.lecun.com/exdb/mnist/