Research on Pruning Feed Forward Network and Convolutional Neural Network

Yuyang Wang

Research School of Computer Science,
Australian National University
u6342479@anu.edu.au

Abstract. Recently, back-propagation neural network has been used to solve a various of problems in some research. The large network normally needs more training time, and also may more easily suffer from overfitting. For better performance in these neural networks, pruning unimportant or redundant hidden neurons has been used to reduce the training time and improve efficiency. In this paper, I applied the pruning technique on feed forward network and convolutional neural network to solve a simple classification problem and a handwriting recognition problem respectively in order to observe how the network performs. The result is that the performance of both networks is quite similar when pruning the first few neurons. However, when the number of cutting neurons comes to a certain value, the performance of both neuron networks decrease much faster, which is similar to the results in the paper[1]. I conclude that neuron networks pruning is only allowed to a certain extend, and the accuracy of the predictor would decrease when the number of the pruned neurons increases in most cases.

Keywords: Network pruning, Feed Forward Network, Convolutional Neural Network

1 Introduction

1.1 Background and Motivation

Nowadays, the application of neural networks is becoming more and more widespread. In pursuit of better performance, normally we need use larger networks with a huge number of neurons and parameters, which indeed have been proved in many research and experiments. However, the most shortcoming of the large network is that too much time is needed to train these neuron networks due to the great number of neurons and corresponding parameters in most cases[2]. The current trend is to apply more technologies to mobile devices, but the weak point of the large network limits the deployment to some extend. Hence, reducing the size of the network structure seems to be more and more necessary.

Also, it is hard to decide the number of hidden layer neurons in FNN or the fully connected layer in CNN. Sometimes, some of the neurons are extra and mostly do nothing during the training process but only increase the time. Therefore, finding a solution to this problem is our top priority.

Pruning network seems to be a proper option for solving the problems stated before. With the help of pruning technique, the complexity of the network structure and the number of parameters is likely to decrease a lot. In this paper we only care about the performance of pruning. Therefore a simple classification problem and a handwriting recognition problem have been used to investigate the details of pruning network.

1.2 Data set

Due to the fact that FNN and CNN normally solve different kinds of problems, I applied two different data sets on these networks.

To the feed forward network, I use the 'leaf' data set. I decide to use this date set because it has the appropriate number of instances and attributes. This data set is obtained from a URL[3]. There are 340 instances and 16 attributes in 'leaf' data set.

To the convolutional network, the MNIST data set have been applied. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image with the help of preprocessing. This data set is available on a URL[4].

1.3 Experiments Outline

The task of out research is to explore the laws of network pruning. Therefore, these two networks would be trained on their corresponding data set at first. Then, pruning neurons have been applied on the hidden layer of the FNN and the fully connected layer of CNN. After that, evaluate the new network on test set to contrast the performance with the original network with more number of neurons. In this problem, the accuracy of the classification have been used to measure the performance of different classifiers in FNN and CNN.

2 Method

2.1 Network Structure Design

2.1.1 FNN

In the leaf classification application, we use a customized neural network, which is similar to the way use 'torch.nn.sequential' to build the network. There are 14 neurons for the input layer (each neurons represents a feature of leaf), 2 neurons for the output layer (each neuron represents a leaf class(0 or 1)). When it comes to the number of hidden layer neurons, it is really hard to decide because whether too many or few hidden neurons would impact the training performance in a bad way. After testing plenty of times, I decide to use 10 neurons for hidden layers before we prune the network because in this way it is easy to find out the vector between different .

We assume that all connections are from neurons in one previous layer to the following one, with no lateral, backward or multi-layer connections. Each neuron has a ordinary weighted connection from each neuron in the previous layer. Our network will be trained with Stochastic Gradient Descent (SGD) as an optimiser, which will hold the current state and will update the parameters based on the computed gradients. As for activation function, sigmoid function has been used.

2.1.2 CNN

Our CNN model contains two convolution layers, each of them is followed by max pooling and Rectified Linear Unit(ReLU). The network takes a 28 x 28 pixel, greyscale, input image and the picture is fed to the first convolution layer with 10 neurons. The kernel size of the first convolution layer is 5. And it is followed by the 2*2 max pooling with stride 2, then a ReLU activation followed by a dropout layer. The output is send into the second convolution layer. The second convolution layer which owns 20 neurons uses kernel of size 5. It is followed by another 2*2 max pooling with stride 2, then a ReLU layer and a dropout layer. The output will be send into two fully connected layer which are the final layers. The number of first fully connected layer neurons has been set to 50, and the number output neurons in the second fully connected layer is 10 according to the digits 0 to 9 in the data set. Pruning would be only applied to the number of fully connected layers.

2.2 Distinctiveness Measurements

When it comes to pruning methods, I decide to use the distinctiveness angular measure method in the paper[1]. From this paper, we can acquire that the distinctiveness of hidden neurons is determined from the neuron output activation vector over the pattern presentation set, what is, for each hidden neuron we construct a vector of the same dimension as the number of patterns in the training set, each component of the vector corresponding to the output activation of that particular neuron and this vector represents the functionality of the hidden neuron in (input) pattern space[4]. Therefore, the vectors for the neurons with the similar functionality would be same, which means some of these neurons could be pruned.

Then, we calculate the angles between different vectors. Actually, we can obtain the cosine of two vectors with the formula shown below.

$$cdist(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

When the angle is between 0 and 90 degrees, the greater cosine value is, the smaller angle is, and the functionality of two neurons corresponding to the angles is more similar. After we find out the most two similar neurons, one of them should be pruned. Then we are able to renew the weight matrix and test on the test data set again to observe how it performs and the accuracy of the classification.

2.3 Data preprocessing

A number of modifications have been carried out on the leaf data set. Firstly, different column names have been given to each column according to the explanation file along with the data set to make each column more meaningful because there is no attribute name in the original CSV file. Also, the attributes called 'specimen number' has been deleted because this attribute is meaningless in classification training and it may have a negative effect on predictor.

Besides, the class attributes have been moved to the last column and the class number have been divided into two categories(1 and 0) to reduce the difficult of classification. In this project, At the end, I disarrange the index of the original data set to obtain a shuffle data in order to rise the performance of neurons network learning. Since if we just apply all the class 1 leaves at first and then apply all the class 0 leaves, the network would learn the first class quite well at the beginning of training, but forget it gradually and learn the second class. Therefore, this modification is quite important to our network. After all these modification, a new CSV file called leaf_new has been created. The modified data set has been shown below (partially).

Eccentricity	Aspect Ratio	Elongation	Solidity	Stochastic Convexity	Isoperimetric Factor	Maximal Indentation	Lobedness	Average Inter	Average Contrast	Smoothness	Third moment	Uniformity	Entropy	Class
0.72694	1.4742	0.32396	0.98535	1	0.83592	0.0046566	0.0039465	0.04779	0.12795	0.016108	0.0052323	0.00027477	1.1756	1
0.74173	1.5257	0.36116	0.98152	0.99825	0.79867	0.0052423	0.0050016	0.02416	0.090476	0.0081195	0.002708	7.48E-05	0.69659	1
0.76722	1.5725	0.38998	0.97755	1	0.80812	0.0074573	0.010121	0.011897	0.057445	0.0032891	0.00092068	3.79E-05	0.44348	1
0.73797	1.4597	0.35376	0.97566	1	0.81697	0.0068768	0.0086068	0.01595	0.065491	0.0042707	0.0011544	6.63E-05	0.58785	0
0.82301	1.7707	0.44462	0.97698	1	0.75493	0.007428	0.010042	0.0079379	0.045339	0.0020514	0.00055986	2.35E-05	0.34214	1
0.72997	1.4892	0.34284	0.98755	1	0.84482	0.0049451	0.0044506	0.010487	0.058528	0.0034138	0.0011248	2.48E-05	0.34068	1
0.82063	1.7529	0.44458	0.97964	0.99649	0.7677	0.0059279	0.0063954	0.018375	0.080587	0.0064523	0.0022713	4.15E-05	0.53904	1
0.77982	1.6215	0.39222	0.98512	0.99825	0.80816	0.0050987	0.0047314	0.024875	0.089686	0.0079794	0.0024664	0.00014676	0.66975	1
0.83089	1.8199	0.45693	0.9824	1	0.77106	0.0060055	0.006564	0.0072447	0.040616	0.0016469	0.00038812	3.29E-05	0.33696	0

Figure 1

When it comes to the MNIST data set, some preprocessing has also been applied. The original black and white images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio, and the resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm.[3] The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.[3] The part of the MNIST data set has been shown below.

Figure 2(obtained from the website[7])

2.4 Validation

In feed forward network, the cross validation(CV) method has been used to test data set. Due to the fact that the instance is quite few for training(only have 340 instance), I decide to use much more data for train than test. The origin data with 340 instances has been split into two parts. 90% of origin data is used to train, and the other data is used to test classifier and obtain result.

In convolutional neuron network, there are 60,000 samples for training and 10,000 samples for test.

3 Result and Discussion

After pruning some parts of the network, the training and test time of CNN and FNN is decreased in different level. Especially when more and more neurons have been pruned, the total time of training and testing network have a more obvious reduction.

3.1 FNN test results and analysis

For FNN, the pruning step is one, and number of the hidden layer neurons changed from 30 to 17 during the pruning process. I found that the performance is quite stable when I prune the first few neurons, but a instant drop on accuracy occurred beyond a certain point, which is really similar to the result obtained from the paper[5]. During the 'stable' period, the accuracy is still decreasing a little along with the reduction of the number of hidden layer neurons, which means the reduction of hidden layer still impact the performance on accuracy in a bad way. However, the accuracy of the classification in paper which use the same dates set as I do is much higher than mine. The result from the paper[5] is shown below.

Domain	Algorithm	Accuracy		Complexity
	Default rule	56%	1	(always choose most common class)
	Assistant :			
	(no pruning)	79%	41	(41 nodes, inc. 24 leaves)
	(pruning)	78%	36	(36 nodes, inc. 21 leaves)

Figure 3

At first, I thought the reason for this might be more neurons have been use in the paper's network. However, after some more neurons are added to the hidden layer in my network, the accuracy for my network does not show a obvious rise at all. Thus, I hold the idea that this might be caused by the different structure of neuron network(may be more hidden layers have been used in the paper[5]) or the different methods applied . The figure obtained from the result of my classification is shown below.

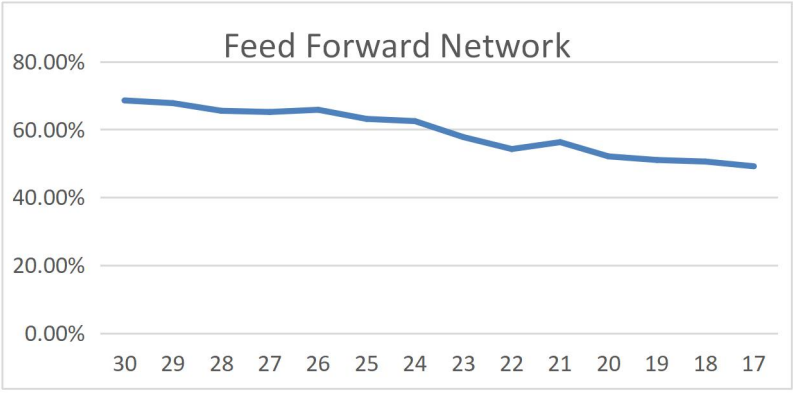

Figure 4(The x axis of the figure is the number of the neurons, and the y axis is the accuracy of the classification)

3.2 CNN test results and analysis

For CNN, the pruning step length is also set to one. However, compared with the FNN, CNN shows a better robustness during the process. When I prune the first few neurons in the fully connected layer, the accuracy of the classification seems to not change at all. After I prune 6 neurons, the accuracy of the network is still greater than 95% (The initial accuracy with no pruning is 98%). When there are about 40 neurons on the first fully connected layer(which means about 10 neurons have been pruned), a slightly obvious downward trend can be observed. And then the accuracy decrease faster and faster. After I prune 15 neurons, the network accuracy drop down to 80%. The figure of the trend is shown below.

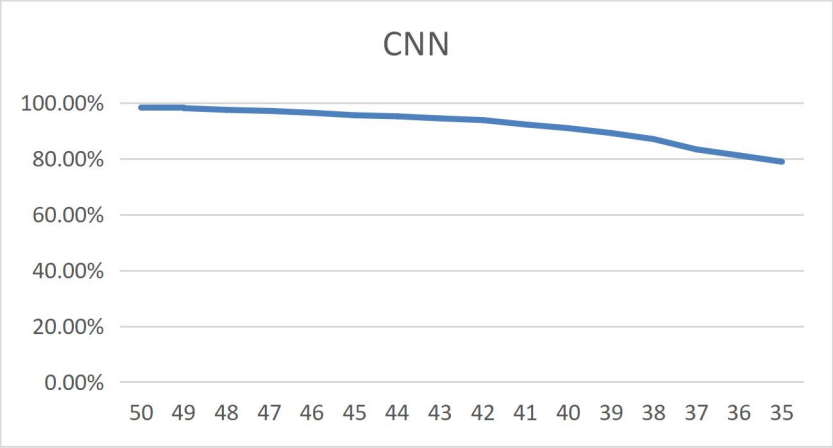

Figure 5 (The x axis of the figure is the number of the neurons, and the y axis is the accuracy of the classification)

In my opinion, the stable attribute of CNN may be due to the complex structure and large number of parameters. In other words, when I prune the convolutional neuron network, only few parts of the neurons have been cut and plenty of parameters are still left.

Besides, there is another paper[8] also use the MNIST data set on CNN. The accuracy of the network in the paper[8] is a little higher than mine with less training iteration. I hold the idea that this may be caused that in their research, they use LeNet-5 architecture and Gaussian connections at the end, which is more complex than my network. The result acquired from paper[8] is shown below.

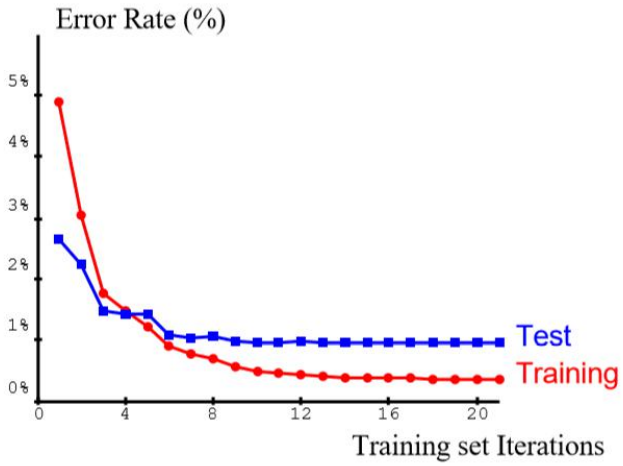

Figure 6

4 Conclusion and Future Work

4.1 Conclusion and improvement

From the result we got above, pruning network would improve the speed of training and not affect the accuracy of classifier badly before a certain point, which means we can obtain benefit from network pruning but too much pruning would decrease the functionality of our network.

Besides, the effect of the pruning to CNN is much smaller than that to FNN because of the different structure complexity. The CNN have more layers and parameters than FNN does, which may lead this phenomenon to some extend.

To improve FNN, some methods could be used like different models, some more features or some more hidden layers. In this project, I just delete a feature and normalize the attributes value. Actually, there are much more pre-processing can be carried out on data set like using the 'percentile' of some attributes, or we can create some new features from the original features which may help a lot in classification. Besides, adding some more hidden layers in neuron network may also improve the accuracy of the classification.

To improve CNN, the layers that apply pruning could be expanded to convolution layer or others. Though these layers have such strong functionality and seems to be not easy to prune, but these layers are also with most parameters. If the pruning technique is able to be deployed on these layers, the efficiency of CNN could be improved on a great level.

4.2 The shortage and future work

For FNN, though the law of pruning neurons have been explored, the balanced accuracy in this classification problem is not desired enough. The performance of my classifier must be improved not only on speed, but also on the accuracy. Because no matter how fast my network could be, the accuracy of the classification is the core of neuron network. Besides, in the future research a polynary classification problem may be adopted to improve the complexity of the network.

For CNN, the problem we solve is quite simple. We have to know that the network with such strong functionality may have millions of neurons, which requests more time to train and test. For example, the Google team train their new network for a whole week. What I want to claim is that only if we apply the pruning technique on these kinds of large networks, the power of technology can be maximized. Therefore, I would prune more complex and more functionally strong neuron network in the future.

5 Citation and Reference:

1. T.D. Gedeon: The paper 'Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour', School of Computer Science and Engineering The University of New South Wales Sydney 2052 AUSTRALIA (1995)

2. T.D. Gedeon: "The major disadvantages of the back-propagation method are that it can be slow to train networks, and that the architecture required for a solution to a problem is not currently determinable a priori." NSW Sydney(1995)

3. UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Leaf (2014)

4. The official website of MNIST: http://yann.lecun.com/exdb/mnist

5. T.D. Gedeon: "The distinctiveness of hidden neurons is determined from the neuron output activation vector over the pattern presentation set. That is, for each hidden neuron we construct a vector of the same dimensionality as the number of patterns in the training set, each component of the vector corresponding to the output activation of that particular neuron. This vector represents the functionality of the hidden neuron in (input) pattern space." NSW Sydney(1995)

6. Clark,P. & Niblett,T. (1987). The paper'Induction in Noisy Domains.' In I.Bratko & N.Lavrac (Eds.) Progress in Machine Learning, 11-30, Sigma Press. (1994)

7. Google picture: https://www.google.com.au/search?q=MNIST&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiD4NWNiKjbAhUDWrwKHWhLCwMQ_AUICigB&biw=1536&bih=734#imgrc=hLiTkQUK7rn9mM:

8. Y.LeCun,L.Bottou,Y.Bengio and P.Haffner: The paper 'Gradient-based Learning Applied to Document Recogniti on'