Pruning Neural network using feature selection and Genetic Algorithm

Tianxiang Zhou
U6124263@anu.edu.au

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

Abstract. Neural networks can obtain good results in classification problems, but it may need large storage size to store the parameters from hidden units and requires massive computations such convolution calculation. Therefore, it is important to optimize the model by pruning the neurons which are redundant and do not have a positive effect on the output. As a result, the Neural network model can be smaller and more efficient because of the reduction of parameters. In this paper, two neural network pruning algorithms are implemented to prune the neurons from hidden layer. The first algorithm is feature selection, it ranks the hidden neurons by calculating the badness of them, and then prunes the worst neurons [1]. The second one is to define the connectivity of network by genetic algorithm and explore the connections to find a combination of connections which have a good performance [2]. In this experiment a 3-layer neural network model was trained on the hand writing digits dataset as the baseline model. And the results of original model and two optimized models are compared at the end of this paper.

Keywords: Model optimization, Hidden neurons pruning, Feature selection, Genetic algorithm

1 Introduction

1.1 Network Pruning

Neural network algorithms have been implemented in different fields such as nature language processing, computer vision and data mining. Because neural networks can address the problems in complex context by training models on datasets. However, it requires many parameters to build the networks and neurons which cost much time to obtain the results. But in some cases, due to the hardware and time limitation, it is impossible to run such a big model. For example, the neural network needs to process the input image from camera and output results real-time in automatic driving system. Therefore, pruning neurons and eliminating unnecessary layers are important to improve the efficiency of neural network model. Many researchers have proposed a lot of network pruning techniques in feature selection, hidden neuron reduction and model simplification.

1.2 Dataset Description

Handwritten digits recognition is a typical multi-class classification problem, which has been used in many machine learning algorithms such as SVM, kmeans. Since Lecun [3] use backpropagation network to recognition the digits, it becomes a very popular practice for the beginners of neural network. These kinds of datasets can be easily understood by beginners because the data structure is quite simple and clear that each dimension of the data is a color value from specific pixel.

This dataset is from UCI Machine Learning Repository [4], there are 1593 instances of handwritten digits from about 80 persons. Then, these digits are scanned and stretched in a square box (16x16) in a gray scale. As a result, each instance has 256 color values and each value is scaled into a boolean value by using the fixed threshold 127. In other word, if the grayscale value is higher than 127 the Boolean value will be 1 and otherwise it will be zero.

1.3 Algorithm Description

The flow char of the algorithm is shown in Fig.1. In this paper, an origin model was trained as the base-line model on the dataset. The algorithm uses a 3-layer neural network model with Relu activate function [5] to generate 10 output units by input 256 units of attributes. In the training part backpropagation and Softmax are implemented, since these two methods are effective the algorithm in classification problems. Softmax will output the probability of the example belongs to each class, then the CrossEntropy loss function can use the probability to calculate the loss. After all, Backpropagation can minimize the loss by Stochastic gradient descent algorithm to get the optimal weights from each layer through several iterations. This origin model has a similar architecture as [6] which works on the same dataset and obtain 88% accuracy on the test results. Furthermore, to demonstrate the effects of pruning neurons, I doubled the number of hidden neurons that is more than necessary to do the job.

The network pruning method are implemented to improve model in reduce the hidden neurons. The first method calculates and ranks the badness of each hidden neuron, so that neurons which are not able to contribute to good results will be found and pruned. The second one is using genetic algorithm (GA) to define the network connectivity and find necessary connections. GA use a population of strings to represent possible solutions, therefore, the connectivity can be represented by a binary string with length of the number of hidden neurons. The optimized results will be found in several generations. In the final part, test results are generated by calculating the accuracy and error in each iteration from test dataset. And a figure of accuracy results from the original model and improved model. This information is important in evaluation the model, when comparing with other results from other research paper.

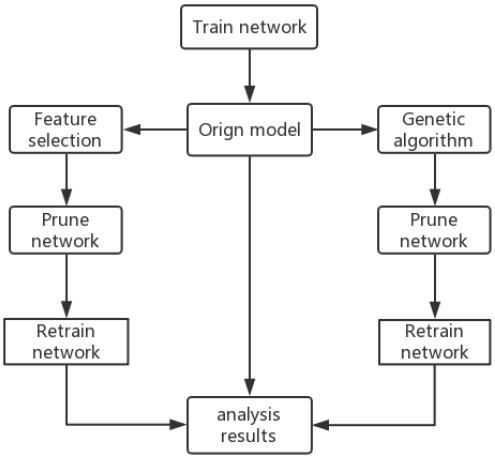

Fig.1. Steps in the algorithm

2 Method

2.1 Data Encoding

Before using the dataset, some preprocessing and data encoding method are implemented to change the format of data in data file.

2.1.1 Preprocessing

In data file, each line of data has 266 boolean values. The first 256 values represent the color values of the pixel form a 16*16 sized digits image, and the last 10 values represent the class this image belongs to. Since the algorithm of feature selection requires to scale the input data to [-1,1]. Thus, the preprocessing is to read each line of the data and map the boolean values to -1 and 1. Moreover, the way of encoding the target are changed from one hot representation to class number by following formula.

$$Argmax([target]) = class \tag{1}$$

After that, the size of input array changed to 1593x257, where the first 256 columns values are training data and the last column the class labels.

2.1.2 Splitting Dataset

Before inputting the data to training algorithm, the data array should be split into two parts: train set and test set. Therefore, a fix number (around 70) of data rows are random selected and put to the train set from each class (700 in sum), the remaining data rows becomes the test set. In this way, the model can be trained on the train set and evaluate on the test set.

2.1.4 Output Encoding

The output units are encoded by Softmax function [7], which can be used to generate a probability distribution over 10 different digits. Softmax function can transform a K-dimensional vector of numbers into a vector of numbers in range (0,1) by following formula:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{for j} = 1, .., \text{K.} \quad (2)$$

Here Z_j represent the j-th value from the K-dimensional vector. And since the model use Cross Entropy Loss, all the probabilities from the output of Softmax are used to calculate the loss.

2.2 Implement Neural Network Model

2.2.1 Network Architecture

I have tested various network architecture in different layers and hidden units and found that the model will be under fitting with few layers and too much layers will lead to over fitting. Here I choose a 3-layer model in Fig 2**.**

This model has 256 input units which takes the 256 Boolean values of input data, and there is one hidden layers with 32 units which is doubled to illustrate the effectiveness of the two network pruning methods. Besides, all the hidden neurons are followed by a Relu function. The Last layer is the Softmax layer, the 10 output unites are encoded by Softmax function.

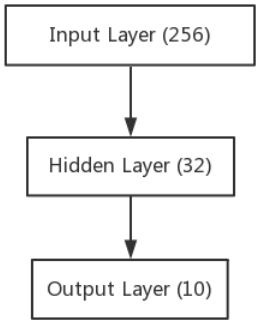

Fig. 2. This shows an architecture of neural network in this algorithm. It consists of 3 layers, with an input layer, a hidden layer and an output layer

2.2.2 Backpropagation

During training the model data are shuffled, and Mini Batch increased the number of training data rows to 20 in each epoch. From the output layer, 10 probabilities of 10 classed are obtain, and they are input to calculate the Cross-Entropy Loss function [8].

$$E_t(y) \coloneqq -\sum_i (t_i \log(y_i) + (1 - t_i) \log(1 - y_i)) \quad (3)$$

Here y_i is the predicted probability value for class i and t_i is the true probability for that class. In order to get optimal weights in each layer, mini batch gradient descent is used in backpropagation stage. The Cross-Entropy Loss will be decent in each batch and epoch correspondent to learning rate. And Stochastic gradient decent algorithm [9] can minimize the loss step by step. The step size is proportional to the batch size that can significantly reduce the training time, because it can process many input examples and converge faster. Here the learning rate is set to 0.03, and the momentum rate is set to 0.5, these two parameters are tested to be efficient in training model on this dataset.

2.3 Feature Selection

Though gradient decent algorithm can minimize the loss in several iterations, it takes too much time and sometimes the final model may be not efficient since there are too much unnecessary neurons increased the complexity of computation. Therefore, this algorithm is implemented to pruning the hidden neurons [1]. The main aim of this algorithm is to find and rank the bad neurons from the last hidden layer and eliminate them.

Firstly, the paper defines the badness of hidden neurons is how much error generated by it in the output layer. Then the paper use following formula to calculate the error in output layer:

$$\delta_j^k = (t_j - o_j^k) f'(net_j^k) \quad (4)$$

$$o_j^k = f(net_j^k) \quad (5)$$

Here output unit has k layers, and o_j is the value from j-th output unit, t_j is target value of that unit. Since softmax is used in the output layer, the derivation of output layer is that:

$$f'\left(net_j^k\right) = f\left(net_j^k\right)\left(1 - f\left(net_j^k\right)\right) = o_j^k\left(1 - o_j^k\right) \tag{6}$$

After that the error of hidden layer units is defined:

$$\delta_j^{k-1} = \left(\sum_l w_j^{k-1}\,\delta_j^k\right) f'\left(net_j^{k-1}\right) \tag{7}$$

Here l is the number of units from hidden layer and w is the weights, and Relu is the activation function in hidden layer, the derivation of hidden layer is that:

$$f'\left(net_j^{k-1}\right) = 1, net_j^{k-1} > 0 \tag{8}$$

$$f'\left(net_j^{k-1}\right) = 0, otherwise \tag{9}$$

Using above formulas, we can get 32 values which represent the badness of the last hidden layer. According to the algorithm, top k units are selected and pruned. However, it is hard set the k value, because we do not know the exact number of unnecessary hidden neurons especially when facing real problems. So, I test the algorithm with k in [1,31], to find which k is suitable for this case.

2.4 Genetic Algorithm

The disadvantage of Rank Hidden Neurons methods is hard to define the number of unnecessary neurons, while it is one of the strength of genetic algorithm. GA is based on natural selection, which is similar to biological evolution. The algorithm modifies a population of individual solutions and selects individuals at random from the current population to be parents and uses them to produce the children for the next generation in each step. After several steps, the population will converge toward an optimal solution. Thus, connections can be both removed and reintroduced through recombination by GA, rather than tell the number of unnecessary neurons directly.

In the implementation, a new model is generated and evaluated based on the original model with 32 hidden neurons and the binary string. The binary string including 32 binary values is used to represent the connectivity of the hidden layer[2]. Value 1 means the neuron is preserved and 0 means neurons are pruned. In processing genetic algorithm, the existing population is selected to breed a new generation. The probability of selecting the population is generated by the fitness, which is derive from the accuracy of the new model. In addition, Genetic Algorithm will change the population by mutation and crossover. And finally, the algorithm will terminal when fixed number of generations reached.

2.5 Evaluation

After two network pruning methods, the model will be retrained for 20 times, and then it is evaluated by calculating the accuracy of test result on each digit class by (10). Because the selected research is also use accuracy to evaluate the performance of the model. In addition, the average accuracy in each iteration are recorded to show how the model changed step by step. And an evaluation matrix is shown in the results.

$$Accuracy(c) = \frac{num(y_i == c\ and\ t_i == c)}{num(t_i == c)} \tag{10}$$

Secondly, two network pruning method are evaluated separately. The feature selection method is evaluated by picking different k values to analysis the effect of reducing different number of hidden neurons. As for the results of GA, we concern about how many neurons reduced when the model gets a high accuracy. So, I aggregated the population appeared in the whole algorithm, and plot the number of each aggregation in a bar chart. For example, [1,1,1,0] and [0,1,1,1] will be aggregated to group 1 since both of them pruned one neuron.

3 Results and Discussion

3.1 Test Result Compared with Another Research

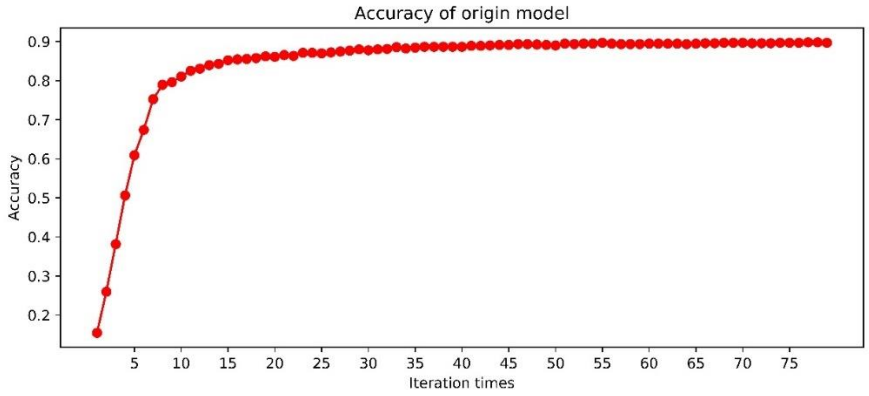

Fig. 3. Figure of how accuracy changed in training the origin model

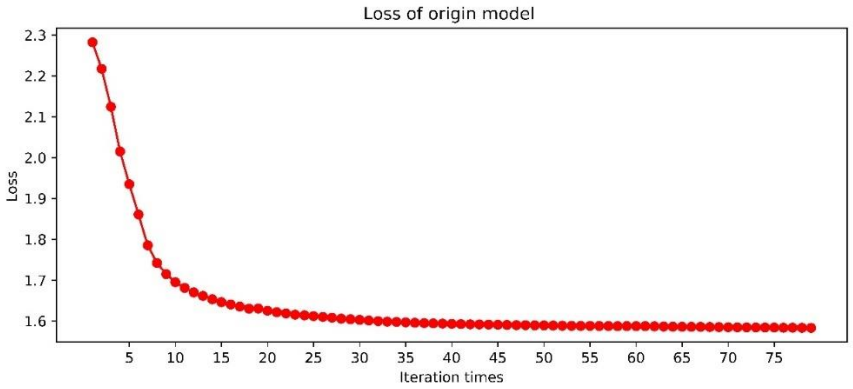

Fig. 4. Figure of how loss changed in training the origin model

From Fig. 3 and Fig. 4, we can find that the original model has already reached a local minimum, and with the increasing of iterations, the accuracy and loss changed little. Therefore, it is useless to optimize the model by adding the training epochs.

Table 1. Test results of accuracy of each class

Number	Results from Paper	My Model
Zero	96.25%	92.31%
One	88.89%	92.39%
Two	88.75%	92.13%
Three	88.61%	78.65%
Four	82.72%	91.21%
Five	89.87%	93.26%
Six	95.06%	97.80%
Seven	79.75%	82.95%
Eight	83.12%	84.70%
Nine	87.34%	78.41%
Total Accuracy %	88.07%	88.47%

Table.1 is the test results of accuracy of each class are compared with the result from paper [1]. According to the result, my model gets different accuracy results in most classes, but the total accuracy is almost the same. The only difference between two models is that my model has doubled the number of hidden neurons. There are 16 hidden neurons in the model from paper, while my model has 32 hidden neurons. As we can see in the results, with 16 additional hidden neurons, the model did not obtain a much better result. A complicate model with more weights and hidden neurons can get a relatively higher accuracy on results, but it increases the compute cost and sometimes it is not worth to use too many weights in neural network to gain a small improvement in accuracy. In addition, it also faces the risks of getting over fitting. Therefore, the origin model is not efficient since there are at least 16 unnecessary hidden neurons.

3.2 Results of Pruning Neurons by Feature Selection

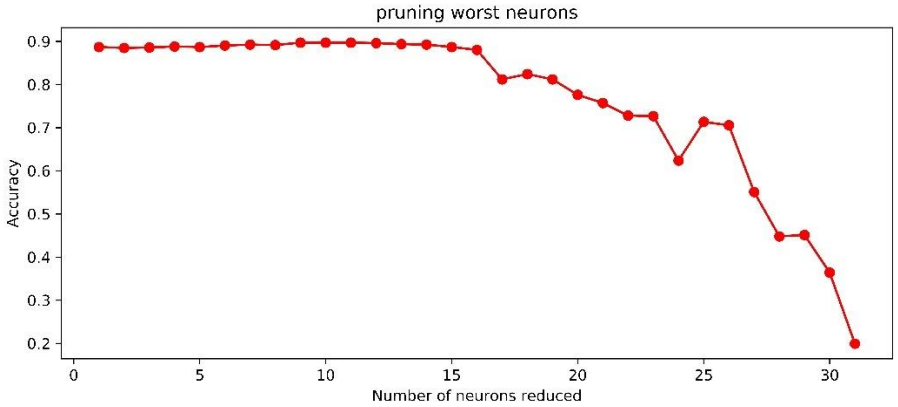

Fig. 5. The figure shows the changes of accuracy when picking different k values in feature selection algorithm

According to the Fig.5, we can see a rapid decrease when 17 neurons are reduced. It means when 16 hidden neurons are eliminated from the origin model, the accuracy will not change a lot, but it is not suitable to prune one more neuron, because the accuracy will decrease by about 10%. In addition, the highest accuracy is obtained when 11 neurons are pruned.

3.3 Results of Pruning Neurons by Genetic Algorithm

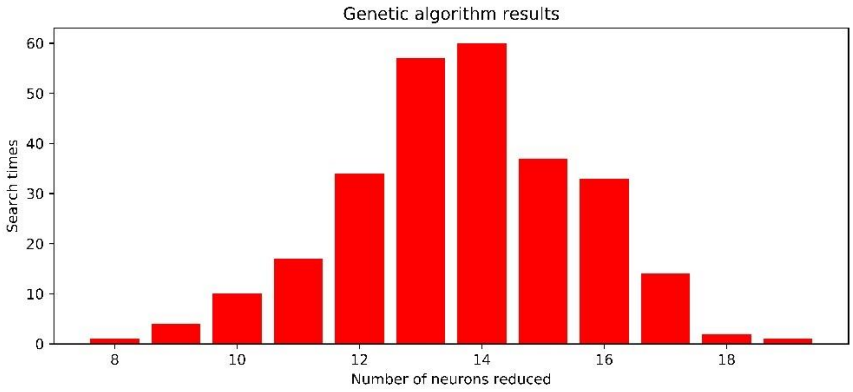

Fig. 6. This figure shows the aggregated results of population

Fig. 6 shows that the population pruned about 13 or 14 neurons are more likely to be selected in Genetic Algorithm, and according to the test result of my program, the best result pruned 12 neurons with the accuracy of 90.75%. The results indicate that compared with the origin model, these population can get higher accuracy by pruning neurons. Therefore, Genetic Algorithm is also effective in optimization the network.

4 Conclusion and Future Work

In summary, this paper investigates two neural network pruning algorithms by using feature selection and Genetic Algorithm. The origin neural network model has 256 input neurons, 32 hidden neurons and 10 output neurons, and the hidden neurons are designed be doubled which are more than necessary to do the job. The model is trained on hand-written digits dataset and obtain almost the same average accuracy with another research on the same dataset.

Feature selection method can find and rank the neurons by calculating the badness of each of them. But the disadvantage is that it is hard to make decision on how many neurons should be pruned. After test the algorithm on pruning different number of neurons, the accuracy will not change a lot until 17 neurons are pruned. Also, there is a trade off in accuracy and efficiency. Because highest accuracy is obtained when 11 neurons are pruned but reducing 16 neurons is a better choice to get an efficient model.

The Genetic Algorithm encoded the connectivity of hidden neurons, and then generate and select possible populations to search the optimal solution to the pruning problem. In the results, it converged in pruning 13 or 14 neurons and the best result came from pruning 12 neurons. The advantage of the Genetic Algorithm is that the number of reduced neurons are picked automatically and it will return the best model after training. But the drawback is that it tasks too much time in training each model to get the accuracy.

In the future, several methods should be implement to improve the model and the algorithm. Firstly, some weight initial algorithm and converge method can be added in training processing. Because sometime the model cannot reach the global minima, the results always changed in the experiment resulting unstable accuracy. Another method is to complement the feature selection and genetic algorithm's advantages. I think combine the badness and accuracy to calculate the fitness in genetic algorithm may reduce the time of training and obtain a stable result.

References

1. Hagiwara, Masafumi. "Novel backpropagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection." *Neural Networks, 1990., 1990 IJCNN International Joint Conference on.* IEEE (1990)

2. Whitley, Darrell, Timothy Starkweather, and Christopher Bogart. "Genetic algorithms and neural networks: Optimizing connections and connectivity." Parallel computing 14.3 (1990): 347-361.

3. LeCun, Yann, et al. "A theoretical framework for back-propagation." *Proceedings of the 1988 connectionist models summer school.* CMU, Pittsburgh, Pa: Morgan Kaufmann (1988)

4. UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets.html

5. Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *Proceedings of the 27th international conference on machine learning (ICML-10).* (2010)

6. Buscema, Massimo. "Metanet*: The theory of independent judges." *Substance use & misuse* 33.2: 439-461. (1998)

7. Elfadel, Ibrahim M., and John L. Wyatt Jr. "The" softmax" nonlinearity: Derivation using statistical mechanics and useful properties as a multiterminal analog circuit element." *Advances in neural information processing systems.* (1994)

8. Nasr, George E., E. A. Badr, and C. Joun. "Cross entropy error function in neural networks: Forecasting gasoline demand." *FLAIRS Conference.* (2002)

9. Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." *Proceedings of COMPSTAT'2010.* Physica-Verlag HD, 177-186. (2010)