Explore and compare the genetic algorithms and distinctiveness method for pruning applied to Brainwave-Music Effect Dataset

Bowei Wang

Research School of Computer Science
The Australian National University
Canberra, Australia
U7102686@anu.edu.au

Abstract. Neural Network is one of widely used model to solve classification problem. Generally, more neurons in hidden layer will extract more features for classification, but at the same time it will also increase the amount of calculation and cause over-fitting. This requires a pruning algorithm to balance the number of neurons and the accuracy of the model. In the previous study [2], the author has used the distinctiveness pruning method, and its effect is not quite obvious. This survey uses genetic algorithm to do pruning. This method also shows the effect of avoiding overfitting. The basic task of this research is using a three layers neural network with genetic algorithm as a pruning method to classify which type of music the participant is listening. The data set used in this research is a small part of the original paper data set [1]. And the final classification accuracy can achieve 57.4%.

Keywords: Genetic algorithm, Distinctiveness method, pruning, NN, music.

1 Introduction

Neural network (NN) is a popular machine learning technology. Its invention is inspired by the biological neural network of animal brains [6]. It can be used to deal with a variety of problems in the scientific field, such as classification and regression. When a given data set contains labels, the neural network can be trained as a classifier.

The basic Neural network has one input layer, one hidden layer and one output layer. The layers between are fully connected. As the number of the hidden layer usually is a hyperparameter, there many methods work for solving pruning issues. From Tom Gedeon [2], the angle of activation vectors between the neurons in hidden layers can be a measurement to define how similar of two neurons. And which means one of them can be removed to improve the performance of the NN model. But after many experiments using this method, there are some evidence shows that only using the angle of activation vectors between neurons is not enough to decide whether they are redundant or not.

An effective method to determine how many hidden layer neurons are redundant is using genetic algorithms. Mathematically, this algorithm essentially selects the best combination through multiple attempts. In our experiment, this method also showed the effect of avoiding overfitting.

In recent years, there have been many studies on the influence of music on the human brain. For example, music therapy has become a popular psychological treatment method. But how music affects people's mood is still under research. The purpose of this task is to explore whether the human brainwave signals are affected by music. If so, we can use the brainwave signals to determine what type of music the participant is listening to.

The data set includes 25 brain wave features and 1 music category label. The given parameter is a single-channel EEG signal, including 25 features collected by Emotiv-EPOC headphones [3], and the label includes 3 different types of music, namely classical music, instrumental music and pop music. In other words, given 25 different characteristics, the neural network is used to predict what kind of music the participants are listening to. These data sets are a subset of the original data set [1]. The original data set in paper contains 150 features, and the performance of the neural network is quite good, reaching 97.5%.

The basic neural network of this research consists of one input layer, a hidden layer and an output layer. The initial hidden layer is composed of 30 neurons. But after using genetic algorithm as pruning method, there were only 11 neurons left in hidden layer. The accuracy can reach 57.4%. Also, we compared the effects of distinctness [2] and genetic algorithm for neural network pruning, the result was genetic algorithm had a better performance for neurons pruning and also could avoid overfitting. Besides, we analyzed the time complexity of genetic algorithm for pruning.

2 Method

This part can be divided into four parts, which are preprocessing and measurement, neural network structure, distinctiveness pruning and genetic algorithm used for pruning.

2.1 Preprocessing and measurement

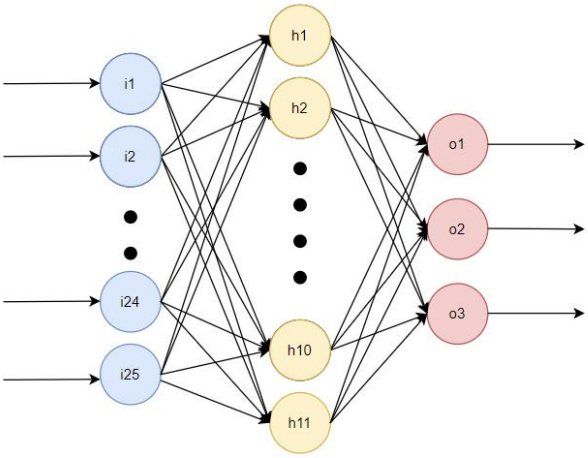
The data was preprocessed by removing ID of the participant first, and then using the Min-Max normalization to get all value be bounded in 0 to 1 in order to reduce the impact of some input features with high value. The dataset was divided into two subset, training set (0.8) and testing set (0.2).

As for evaluation method, we used testing set accuracy and confusion matrix to do the analysis.

2.2 Neural Network structure

The final structure of Neural network was consisted of one input layer, one hidden layer with size 11 and an output layer with 3 neurons which represent the possibility of the data belongs to three different music types. The structure you can find in Figure 1.

Fig. 1. Structure of neural network



The network used CrossEntropy as loss function, Adam as an optimizer function. And epoch number was 1000, batch size was 32 and learning rate was 0.01.

The number of neurons in hidden layer was defined by genetic algorithm. At the beginning, the hidden layer had 30 neurons. After training 2000 epoch we used genetic algorithm and then only 11 neurons were retained.

2.3 Distinctiveness pruning

Distinctiveness pruning method based on paper (Gedeon and Harris, 1991) [2] which was using activation vectors of hidden units to select which neurons were redundant. If the angles of the two activation vectors are less than 15 degree or lager than 165 degrees, it means that these two vectors have similar effects on the neural network, namely, one of them is redundant, so we can delete one of them and add its weight to the remaining neurons.

It also can be applied into image pression [2]. Using distinctiveness of units, we can ensure maximal functionality of compression layer units to improve image compression for a given image quality [4]. We can get these activation vector in Pytorch by using `nn.functio nal.relu(net.sigmoid(net.fc1(x)))`.Each element in an activation vector of a neuron is an output of this neuron given by all input.

The reason we used sigmoid function was that we wanted the all elements in that activation matrix were are constrained to the range 0 to 1. Thus, we can normalize them to 0.5 which will give us the angular range of 0-180 degree rather than 0-90 degree.

2.4 GA pruning

The idea of genetic algorithm is inspired by the evolution of biological populations under the theory of natural selection [6]. Genetic algorithm (GA) is an effective bio-inspired computing algorithm which is capable to find a "relatively good" answer in a large solution space within a short time limit.

Encoding for chromosome and the definition of fitness function is the key point of GA. These two functions finally force evolution to move toward the expected optimization goal.

First, the chromosome is designed as a vector with binary values, consisting of a string of 0s and 1. In our task, the chromosome is the mask of the weight matrix of neurons in hidden layer. 0 indicates the neuron should be removed, and 1s means the neuron to keep.

In order to "simulate" the evolutionary process, there are three steps in each iteration, which are selection, crossover, and mutation. Selection means that chromosomes with high fitness score are more likely to survive, so that the population has a higher proportion of chromosomes with higher fitness score. Crossover enables the transmission of information between chromosomes, which greatly enhances the generalization ability of the model. Mutation usually refers to a random change on the chromosome, which introduces randomness into the entire model to facilitate exploration. And the change is meaningful for jumping out of the potential local optimal solution.

3 Results and Discussion

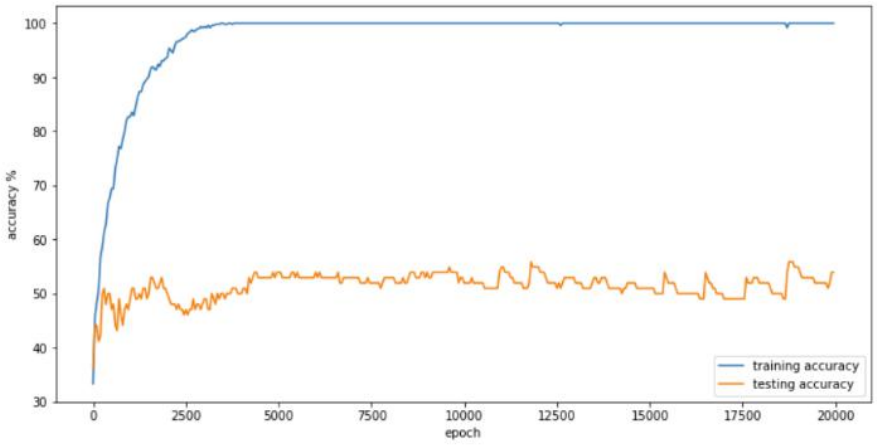
This part will be divided into three sub-parts. First is the experiment on basic one hidden layer NN and second is about the parameters choosing in distinctiveness pruning method. And 3.3 is the result of using genetic algorithm as a pruning method, also the result has compared with distinctiveness pruning method.

3.1 Experiment on basic NN

The neural network model has a hidden layer containing 30 neurons at the beginning, because this is the parameter of the number of hidden neurons mentioned in the data set paper [1]. In this part, we only consider the basic three-layer neural network model, without using different pruning method. The pruning process will be mentioned in 3.2 and 3.3. The epoch was set to 5000. If the number of epochs is small, the model may be underfitting, but if the number of epochs is large, overfitting may occur.

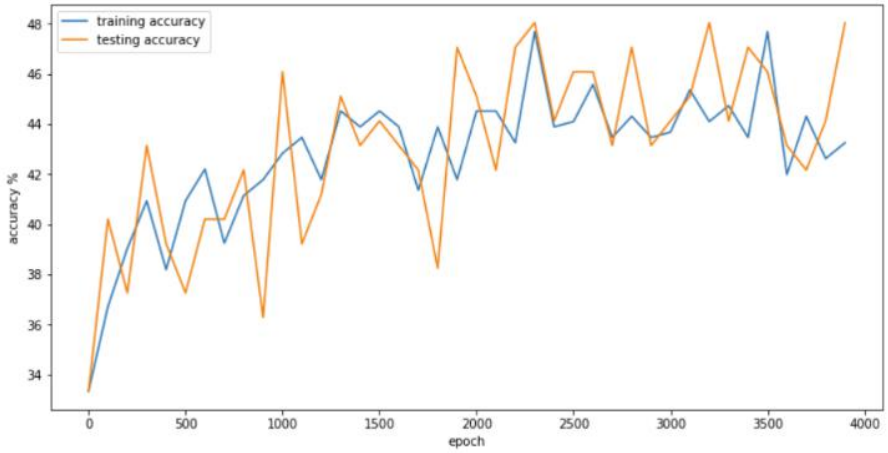
In Fig.2, we can find that when the number of epochs is around 5000, the test accuracy achieves a peak which is around 50% and the training accuracy is about 100%. As for the parameters for Figure 2, we used Adam as an Optimizer, CrossEntropy as loss function and the learning rate is 0.01. If we trained my model continually after 5000 epochs, the accuracy of training and testing won't change a lot. So, in the later experiment, we set the number of epochs to 5000, which can make the model be trained to a relatively stable level.

Fig. 2. Without dropout



As you can see from the figure 1, the model has a serious overfitting situation. So, we used dropout [5] as a way to prevent overfitting. After added a dropout(p=0.5) after hidden layer and also changed learning rate to 0.0001, the result is shown below.

Fig. 3. With dropout



In Fig.3 we can see that the two lines are basically very close, which proves that we had solved the overfitting problem. And when epochs number was about 2000, the model tended to be stable. After that, we took the average accuracy of ten attempts, which was 43.7%. The testing accuracy could hardly exceed 50%. Also, the accuracy curve was always fluctuating. Through confusion matrix, there was no obvious misclassification.

3.2 Experiment on distinctiveness pruning method

The parameters we want to adjust include defining the angle range of similar activation vectors. In the paper [2], authors set the angel between two similar activation vectors are 0-15 degree and 165-180 degree. And they pruned the network after each 200 epochs. But when we used the same parameters in our NN model, there are few neurons need be pruned. When we set initial hidden neurons number to 30, we deleted one neuron as an average number in ten attempts and all of them had been pruned after the first 200 epochs. The result for test sets had not been decreased which was still around 45%. To see if 29 neurons in the hidden layer is enough for this task, we add the number of neurons to 40, 50, 60 at the beginning.

The parameters for this were: hidden size = n, learning rate = 0.01, batch size = 32, num epochs = 5000
, is pruning(bool) = True

Table 1. The angle range was defined as 15 degree

Number of initial neurons in hidden layer	Number of similar neurons detected by distinctiveness pruning method	Number of remaining neurons	Traning accuracy after 5000 epochs	Testing accuracy after 5000 epochs
30	1	29	67.67%	45.38%
40	3	37	74.04%	44.06%
50	8	42	69.42%	43.23%

The reult shown in Table 1 was not expected, because the number of remaning neurons was expected to be the same number as the task and the dataset had not been changed. But basicly we found that as the number of initial neurons increased, the number of pruned neurons also incresed.

We increased the angle to see the relationship with angle and the number of neuron pruned. The parameters for this were: hidden size = n, learning rate = 0.01, batch size = 32, num epochs = 5000, is pruning(bool) = True.

Table 2. The angle range was defined as 30 degree

Number of initial neurons in hidden layer	Number of similar neurons detected by distinctiveness pruning method	Number of remaining neurons	Traning accuracy after 5000 epochs	Testing accuracy after 5000 epochs
40	29	21	62.94%	46.08%
60	37	23	53.04%	45.10%
80	54	26	61.05%	46.12%

From the above Table 2, we can see that by stretching the angle range to 30 degree. After pruning, the number of remaining neurons was basically similar which was around 25, and the accuracy of testing sets did not show any obvious downward trend.

Through the analysis, we can get the conclusion that 25 neurons in hidden layer is enough to get an average performance.

Compared with the original paper, our result was quite poor. This paper [1] could get an accuracy at 97.5%. The main reason was that we only used a small part of the original dataset, that was, only 25 features. And these 25 features were not the most obvious features of the first 25, only 25 features of a specific part of the brain from F& channel. Therefore, the result of neural network training fluctuates greatly.

In addition, we found that for different task conditions, we can set different sizes of angle range to compress our neural network to the greatest extent. For this task, we can expand the angle range to 30 degree to compress our model so that it contains only about 25 neurons in the hidden layer.

And by comparing the change of testing sets accuracy, after pruning, the accuracy would drop suddenly, and then after continuous training, it would return to the original level.

3.3 Experiment on GA algorithm for pruning

We want to perform pruning because we want to simplify the complexity of our model as much as possible while ensuring that the accuracy is basically the same, and this also helps to avoid the occurrence of overfitting. The distinctiveness method we used previously only relied on the activation vector of two neurons. This method has been proven in some papers and is not sufficient to explain whether two neurons have the same and redundant role in the model.

But if we use genetic algorithms, we can get truly accurate answers, based on constant iterations and a large number of attempts. Based on different goals, we have two fitness functions. If the goal is to prune to a specific size, the fitness function is:

$$F(x) = Acc(x) - |sum(x) - n| \tag{1}$$

Among them, term ($-|sum(x) - n|$) is to prevent all genes on the chromosome from being 1, which will not achieve the purpose of pruning to a specific size. And n equals the specific size that we want for hidden layer. Term Acc(x) is the accuracy of test set when we use a chromosome x. And sum(x) means the number of ones in chromosome x. As the term ($-|sum(x) - n|$) has a lager impact than Acc(x), this fitness function can have the ability to prune the neural network to a fixed size.

If our goal is to prune the network in order to prevent overfitting regardless how many neurons we want to keep, then our fitness function is:

$$F(x) = Acc(x) \tag{2}$$

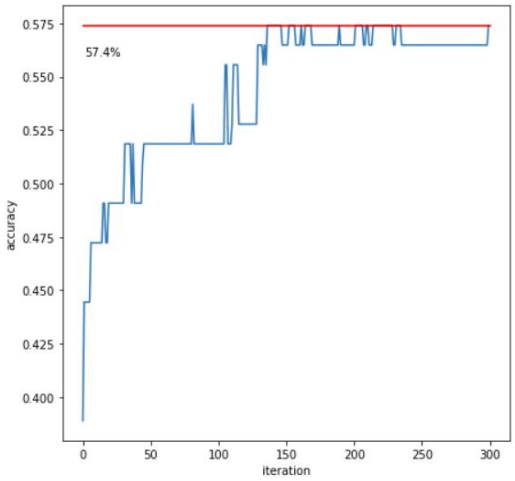
where $Acc(x)$ is the accuracy on the test set.

After training 3000 epochs and we used one-shot genetic algorithm pruning, the outcome can be seen in Fig.4. The parameter for Fig.4 is

```
generation_num = 100
DNA_size = hidden_size              # number of bits in DNA (hidden_size)
pop_size = 100                      # population size
cross_rate = 0.8                    # DNA crossover probability
mutation_rate = 0.002               # mutation rate
```

When the number of iteration lager than 150, the accuracy became stable which was around 57.4%.

Fig. 4.



The final best chromosome is [0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1], which only contains 11 neurons in hidden layer. To compare it with distinctiveness method, we calculated the angle between all of these 30 neurons. However, there were only two of the neurons have an angle which was smaller than 20 degrees. Namely, if we use distinctiveness pruning method, we only need to cut one of thirty hidden neurons, and the result of it was just 43.5 % as for the accuracy which was much lower than 57.4%.

The advantage of using genetic algorithm for pruning is that it is using the test accuracy result, which can have a similar outcome of exhaustive search. The time complexity of genetic algorithm is

$$\mathrm{T} = \mathrm{O}(generation\_num \times pop\_size) \tag{3}$$

. The time complexity of the exhaustive search is

$$\mathrm{T} = \mathrm{O}(2^{DNA\_size}) \tag{4}$$

. When the number of genes, that is, the length of chromosomes, is small, the time complexity of genetic algorithm is higher than that of exhaustive search. When *DNA_size* is large, the time complexity of genetic algorithm will be much higher than that of exhaustive search, which is its advantage.

In the experiment, we found that the use of genetic algorithm can also achieve the effect of avoiding overfitting. This effect is better than using the dropout function on the test set. Dropout is based on the algorithm used in the training set to avoid over-fitting, and the genetic algorithm we use is based on the test set, which means it can be an effective algorithm to avoid over-fitting.

Compare to the previous test, using GA to do pruning can easily achieve a high accuracy which is larger than 50%. But by using dropout, or using distinctiveness the accuracy is difficult to exceed 50%.

4 Conclusion and Future Work

4.1 Conclusion

In this experiment, we explored the influence of different parameters on a simple neural network with only one hidden layer. Since the data set contains only a small number of features, it is easy to overfit. By adding a dropout layer after the input hidden layer, the accuracy of the training set and the testing set will be roughly the same, but it cannot improve the accuracy in test set.

In the process of using different pruning methods, we used two different pruning methods, one is distinctiveness pruning method, and the other is pruning method based on genetic algorithm. Through comparison, we find that the results of the distinctiveness pruning method cannot improve the accuracy of in the test set, but when the genetic algorithm was used, the test accuracy can be approached to the global optimal solution. The final accuracy of classficatrion can be 57.4% working in music-eeg-features dataset. When we use genetic algorithms for pruning, we get the best combination of hidden layer neurons based on accuracy, which allows us to avoid overfitting. In this way, neurons with good generalization ability can be retained, so that the model has better generalization ability.

Experiments have proved that it is not enough to decide whether pruning is needed only by activation vetor.

4.2 Future work

For future work, we can compare more pruning methods and effects of avoid overfitting. On the other hand, we can continue to optimize the fitness function of the genetic algorithm to achieve higher accuracy. We can try to experiment with a complete data set.

Although we have good results using genetic pruning, this method does not give the real reason why some neurons are pruned. We only know that the pruning will have higher accuracy. In the future, we can further explore the relationship between these pruned neurons, not only by comparing the angle of the activation vector between them.

5 References

[1] Rahman, J.S., Gedeon, T., Caldwell, S. and Jones, R., 2020, July. Brain Melody Informatics: Analysing Effects of Music on Brainwave Patterns. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[2] Gedeon, T.D., Catalan, J.A. and Jin, J., 1997, September. Image compression using shared weights and bidirectional networks. In Proceedings 2nd International ICSC Symposium on Soft Computing (SOCO'97) (pp. 374-381).

[3] Emotiv pro academic license. [Online]. Available: https://www.emotiv.com/product/emotivpro/

[4] Gedeon, T.D. and Harris, D., 1992, June. Progressive image compression. In [Proceedings 1992] IJCNN International Joint Conference on Neural Networks (Vol. 4, pp. 403-407). IEEE.

[5] Thakur, A., 2021. Weights & Biases. [online] W&B. Available at: <https://wandb.ai/authors/ayusht/reports/Dropout-in-PyTorch-An-Example--VmlldzoxNTgwOTE> [Accessed 28 April 2021].

[6] En.wikipedia.org. 2021. Genetic algorithm - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Genetic_algorithm> [Accessed 31 May 2021].