Indicators of Hidden Neuron Functionality(Extend): Performance between Back-Propagation Network, Feed Forward Network with pruning and Evolutionary Algorithm with pruning

Hao Jun Chong
School of Computer Science and Engineering
Australian National University
ACT 2600 Australia

Abstract. There are many ways to differentiate neurons with respect to their functionality hidden layer of neural networks. By using the evolutionary algorithm as an optimizer, we can prune redundant or insignificant hidden neurons, which is helpful when comes into performance, to base on its fitness value. With a measure that can numerically calculate hidden neurons as the fitness value of network, we can visualize the training process, grow networks by adding neurons when more functionality is required, and prune the useless hidden neurons.

In this paper we are using only the number of hidden neurons as chromosome and get it fitness value when going through the network. The first is to get the performance in the back propagation neural network that is without pruning and feeding the data measure its quality and performance as well as the feed forward network with pruning based on the previous paper[4]. After that is to get a network with a genetic algorithm optimizer and get the network fitness value for pruning to get the quality and performance as well and compare with the back propagating neural network.

We conclude that the pruning by using genetic algorithm as an optimizer to prune the hidden neuron is getting a better performance than a neural network with back propagation without pruning as well as pruning in feed forward network.

Keywords: neural network, feed-forward, back propagation, evolutionary algorithm, pruning

1 Introduction

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. As a result, to decide the number of hidden neurons which is most difficult. Many papers have mentioned that to train networks

successfully or at an effective rate, more hidden neurons are required than the minimal number. As a result, extra neurons which end up duplicating the functionality of the existing hidden neurons and they will do nothing but decrease the speed of the network by increasing its size. This will end up most of the time is wasted in restarting the training process from scratch. Previously, there are papers that mentioned we can use brute force methods to find minimal size networks by eliminating randomly chosen neurons from trained networks[2]. However, this kind of method is still waste time as it will randomly prune neurons but not sure whether the correct neuron is prune. Therefore, in this paper, we are trying to prune the hidden neurons by using generic algorithm as an optimizer and evaluate the fitness of each network with different amount of hidden neurons, get the best few networks and improve it with few generations.

In the paper[5], it mention that the evolutionary algorithm concept such that those individuals best adapted to their environments are more likely to survive and reproduce. Where this theory is described as“ survival of the fittest”. Those who are fittest than others have the chance to survive in this evolution[6]. As such this what the genetic algorithm all about. It mimics the natural selection process and find the best solution in the problem through several generation of evaluation. As such, in this paper, we are pruning the hidden neuron by giving random number of hidden neurons in a network and let the natural selection process to find out the best network according to the different neuron assigned.

2 Method

To start with this problem, we use the data similar to the previous paper in order to be fair when comparing the methods with each other[1] and predict a user preferences in using what method to study as well as when to study to act as a performance measurement between these 3 types of neural network. Which is we are going to predict groups of 4 preferences (audio, ppt, video, reading) in 12 contexts, that is total up of coming out 48 predictions. Since we previously have created two types of neural network that is back propagation neural network and feed forward network with pruning, mentioned in this paper[3], we only created the third type, that is by using genetic algorithm as an optimizer in neural network with pruning the hidden neurons. To recap in previous neural network setups, for the input layer, we have both to have 51 of input neurons as there is 51 of features after dropping out the insignificant features given by the paper[3]. As for the hidden layer, we consist of putting a 30 hidden neuron for the back propagation and feed forward network with pruning as putting a lot more of hidden neuron will cost very time consuming. As for output layer, we putting 48 neurons as we are predicting 48 preferences. So that each neuron is indicating each preferences from the user.

For the activation function in hidden layer and output layer, we decided to use ReLU as activation function as we are doing a classification prediction for example, it the user prefer to have an audio study alone, the neuron will come out with 1 instead of 0.

Hence, getting the output as ReLU activation function and round up to 0 or 1, we can easily predict what is the user preference.

For the back propagation, we are using BCEWithLogitsLoss as we are doing binary classification in this network. Also we are setting up a 500 epoch to let the neural network to learn.

After that, we are doing a k-fold validation with k = 10 to ensure that the performance is reliable. Lastly, we are comparing both performance by checking out their accuracy and loss as well as the a graph plotting the loss across the number of epoch.

Meanwhile, in evolutionary algorithm with pruning hidden neurons are using a set of different hidden neurons. We gather a set of 10 different amount of hidden neuron and test into the neural network. Where the set of different amount of hidden neuron is called population, and each different amount of hidden neuron is represented in binary representation which is called chromosome. After that, scores each member of the population based on our measurement. This score is called a fitness function. Where the most common fitness functions in our classification neural network is the accuracy, which is mentioned in this paper[5]. After getting all the performance from each member, we start to select some of the member breed them in order to produce more like them.

```
Procedure: Linear rank-based roulette wheel selection
  While population size < pop_size do
        Sort population according to rank
        Assign fitnesses to the individuals according to linear rank function
        Generate pop_size random number (r)
        Calculate cumulative fitness, total fitness and sum of proportional fitness (Sum)
        Spin the wheel pop_size times
        If Sum < r then
          Select the first chromosome, otherwise, select jth chromosome
        End If
  End While
  Return chromosomes with fitness value proportional to the size of selected wheel section
End Procedure
```

Figure1: The pseudocode implementation for Rank Based selection[7]

Where for the selection method, we use the Rank Based selection method. This method is to rank the member by their fitness value in order. The higher the fitness value, the better the candidates is and the bigger probability to be selected when ongoing the roulette wheel selection based on ranking as mention in this paper[7]. The “Figure1” is showing the pseudocode implementation of how rank based selection is performed. This is paper, we tend to select out 2 pair members in the population and breed them for our next generation. Note that the chromosome might undergo mutation with our fixed probability of 0.2% as it can help us increase the diversity in the population as well as increasing the chance in finding out even better candidates in the population. Lastly, it kills all the rest that is not picked during the selection as to maintain the size of the population in 10 members and repeat the whole

step start from evaluate the performance of each member until several conditions[8]. Where in this paper, the condition is given that only fix it to have 10 evolution in the genetic algorithm.

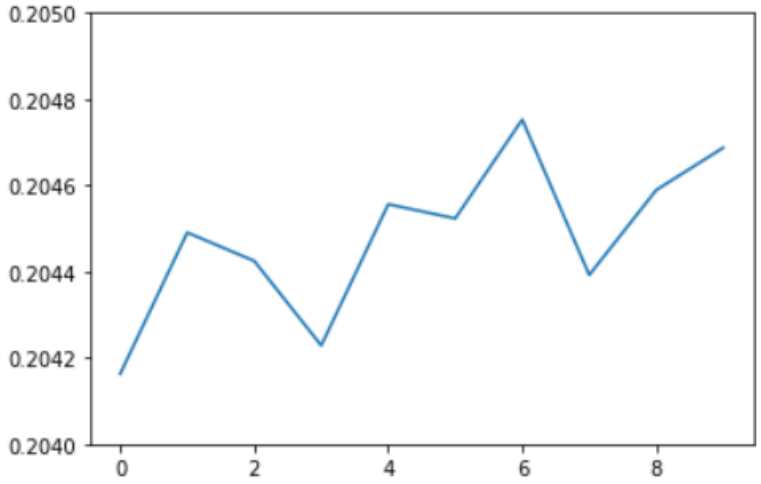
2 Result and Discussion

After implemented the evolutionary algorithm with pruning hidden neuron in Neural Network. The performance is showing in "Table1" and it is shown that it have a different performance compared to both of the previous implemented neural networks.

Network Type	Back-Propagation Neural Network	Feed Forward Neural Network with pruning	Evolutionary Algorithm optimizer with pruning in Neural Network
Testing Accuracy	0.2041	0.1583	0.2047

Table1: The overall performance in accuracy between each type of neural network

In "Table1", we can see that there is a performance improve when implementing the genetic algorithm as an optimizer in the neural network and when selecting which is the best fitted amount of hidden neuron is the respective problem. This is because back propagation does not have the optimal amount of hidden neurons as their hidden neuron is fix, but not same when comes to evolutionary algorithm as it can be auto adjust the number of hidden neurons and find out which amount will get the best performance in the neural network. While feed forward also do the same thing, that is to prune the neural network, but it does make it worse as it only remove the hidden neurons but not increasing it. This will end up in decreasing the performance of a neural network when some neurons is removed with mistakes. Further that, the "Graph1" shows the performance through the generation in neural network with evolutionary algorithm.



Garph 1: The average fitness value of each generation in Evolutionary Algorithm optimizer with pruning in neural network

As shown in "Graph1", it is the average of all the population members' fitness value throughout each generation. In this graph, we can see that the overall of the fitness value is increasing throughout the generation. This means that by using the genetic algorithm as an optimizer, the neural network is getting better at assuming the number of hidden neurons. This will be the case is because the member which having a bad in assuming the amount of hidden neurons has been eliminated throughout the generation where for the member with having a better skills in assuming the number of hidden neuron survived to the next generation and breed the same or better members throughout the generation in the population.

Besides, through the "Graph1", we can conclude that having a bit drop performance in some generation will result in getting more better than ever in their next generation. This is because it diversify when some chosen to be breed with big difference gene members due to the random members probability given. It increases the diversity when there is a highly difference gene in both member and they breed with a child compare to breeding with the members that having a lot of similarity with each other. Apart from that, there is also a chance that having mutation in a child while breeding will increase the diversity in the population. As such, having increase in diversity have a better chance to find a better candidates in getting higher the fitness value as well as performance.

2 Conclusion and Future Work

In a nutshell, after all the discussion we have made and the process we have gone through, it can be seen that there is an ideal performance increase compare to what we are expected. We can also by using the genetic algorithm as an optimizer to our neural network, the performance is better compared the one without using it.

As such, having this kind of technique is adequate enough to getting a better performance to a neural network when adjusting the amount of hidden neurons in the neural network. We can also clearly see that this technique is way more better than the feed forward neural network with pruning. This is because genetic algorithm have the benefits that the feed forward neural network with pruning does not have, that is it can recreate the hidden neurons when knowing the performance is decreasing. Yet the feed forward neural network can only be removing the hidden neurons but cannot reverse all the actions. Therefore, we are prefer to have this kind of technique compared to the previous paper with both neural network given[3], even though it is still somehow a more expensive approaches as it keeps testing out all the kind of possibilities.

In conclusion, for the future work and further investigation is required as the generation use in this evolutionary algorithm is still very limited due to the time cost. The performance tested now is not sufficient to prove that the evolutionary algorithm as an optimizer do really boost the performance compared to without using it. I would suggest that gather a smaller data and better CPU performance will help in saving the time cost and extend the limitation in the numbers of generation to be evolve in genetic algorithm, so that we can have a clearer and more convince performance to show their results.

References

1. Al-Ismail, M., Gedeon, T. & Yamin, M. Effects of personality traits and preferences on M-learning. Int. j. inf. tecnol. 9, 77–86 (2017). https://doi.org/10.1007/s41870-017-0012-0
2. Sharma, Aditya, Nikolas Wolfe, and Bhiksha Raj. "The incredible shrinking neural network: New perspectives on learning representations through the lens of pruning." arXiv preprint arXiv:1701.04465 (2017).
3. T. D. Gedeon, "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, Dunedin, New Zealand, 1995, pp. 26-29, doi: 10.1109/ANNES.1995.499431.
4. H. J. Chong, " Indicators of Hidden Neuron Functionality: Performance between Back-Propagation Network and Feed Forward Network with pruning " Assignment 1, 2020
5. Ahmed Fawzy Gad 'Practical Computer Vision Applications Using Deep Learning with CNNs'. Dec. 2018, Apress, 978-1-4842-4167-7
6. Wikipedia contributors, "Survival of the fittest," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Survival_of_the_fittest&oldid=958550149 (accessed May 31, 2020).
7. Razali, N.M. and Geraghty, J., 2011, July. Genetic algorithm performance with different selection strategies in solving TSP. In Proceedings of the world congress on engineering (Vol. 2, No. 1, pp. 1-6). Hong Kong: International Association of Engineers.
8. Gedeon, T., 2020. Introduction To Computational Intelligence. [online] Wattlecourses.anu.edu.au. Available at: <https://wattlecourses.anu.edu.au/pluginfile.php/2221643/mod_resource/content/5/EA1-%20chapter1%2C8.pdf> [Accessed 31 May 2020].