The reduction pruning method of hidden neuron and evolution algorithm in Feed Forward Neural Network of cleveland heart disease data set

Shichun Yan

Research School of Computer Science, Australian National University
E-mail:u6023257@anu.edu.au

Abstract. In this paper we focus on the musk data set. This report implements the method of using evolution algorithm and the method of using reduction pruning algorithm to train the neural network. The main purpose of using evolution algorithm is to select the best performed feature whose NN perform a better accuracy in the test part. After the investigation, this kind of method can generally improve the performance of the neural network in training accuracy.
Keywords: evolution algorithm, reduction method, neural network

Introduction:

In recent years, with the introduction of the deep learning and machine learning, people start to realize the important meaning of using artificial intelligence to help people predict several diseases which is based on the patients' pattern performance. The neural network is one of the essential part to learn. That's the key motivation to choose this data set for this assignment. The data set is chosen from the UCI Machine learning which mainly describes 13 different kinds of patients' pattern and 5 different kinds of disease symptom which are the classifications of the data set. There are 6 missing value but I replace them with integer 0, so it won't have a significant effect on the neural network training.

To specific the method and algorithm, I first use the evolution algorithm at the first to select the best match of the features. Then the "Network Reduction Technique" [1] is used with the selected features instead all the features to train the neural network. However, there are several disadvantages in these methods. Firstly, high time consuming in evolution algorithm. Secondly, the backwards of using back propagation for they cannot decide the exact number of hidden neurons need to be trained in the neural network. This paper aim to implement the evolution algorithm in the neuron network and discover the effect and the performance on the NN which use the hidden neurons reduction, which will be introduce in the Method and Result analyze part. All the code work is based on Python and PyTorch libraries.

In the Discussion and Result Part, I will present the confusion matrix and accuracy of the paper. And I will compare my NN performance and test accuracy to the result from the article of Efficient Heart Disease Prediction System [2]. The accuracy of this paper and their result will be analyzed and make a conclusion in the Conclusion and Feature Work. What's more some personal though on improving the NN's performance and the limitation will be listed too.

Method:

A. Heart disease data set preprocessing

There are 303 number of instance with 14 attributes. The previous 13 number of attribute are set for describe the game field condition and the last attribute is to determine which type of disease or symptom the patient is having. There are 6 missing value which is distributed mainly in column 12 and 13. In order to decrease the effect which caused by missing value to influence the NN performance, I replace them with integer 0. I also use the following function to preprocess the input data

$$X_i = \frac{X_i - \sum_{n=0}^{N} X_n}{\sqrt{1/N \sum_{n=0}^{N} (X_n - mean)^2}}$$

The network architecture uses 14 input, one hidden layer with 80 number of hidden neurons, and 5 output which represent 5 class.

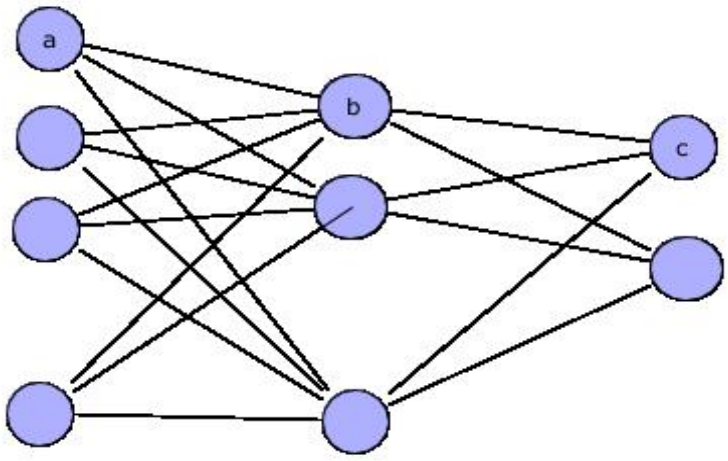

Fig1. network architecture

The main reason to choose the feed-forward network is that it's can perfectly apply the back-propagation. Ans all the afterwards work is based on the back-propagation method. The network should be retraining because some units will be reduced.

B. Reduction hidden neuron

The reduction hidden neuron works on the feed-forwards network in the same way as working in the auto-associative network. The main idea is to delete the hidden unit which is too similar and far too opposite for all pattern. The general process is as following. Firstly, calculate all the possible hidden unit vectors' angle. Secondly, after normalize all the angle to 0.5, we choose those which is bigger than 150 degrees as the complementary units and also filter those whose smaller than 15 degrees as the similar units. These two kinds of hidden units need to be remove from the neural network

C. Evolution algorithm

We use this method at the very beginning to select the best feature which to help us to make a better prediction in the test part. Firstly, I initialize several numbers of 0,1 array whose length equals to the number of the data set's attributes (1 means use this position of attribute, 0 means don't use that one in a neural network) and each array represent a neural network. Then, I define a F function to evaluate the accuracy performance in each neural network. After that, I use F function output as the standard to select the best neural network which has the special kinds of attributes fit in the current environment. we also need to imply some mutation and

cross in each generation. Finally, the last generation of match in features is the beat match to predict the expect classicization in the test set. We then use these features to train the neural network. A simple flow diagram can be described below.

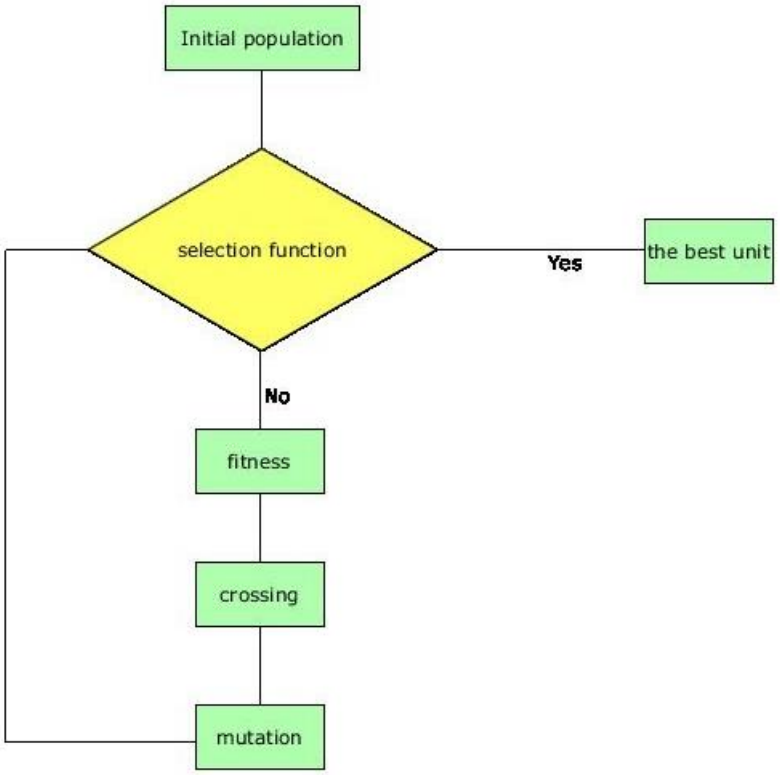

Fig2. Flow chart

D. RESULT EVALUATE TYPE

As to the result evaluate type, I use a confusion matrix to evaluate. Confusion matrix is a visualized display tool for evaluating the quality of classification models. Among them, every column of the matrix represents the sample situation of the model prediction; the real condition of the sample represented by each row of the matrix.

Results and Discussion

All the presented data is the average number of the 20 times running result.

Table 1. Accuracy of using different type of method.

Type of NN	Test Accuracy (%)	Test time (s)
NN	**55**	0.0034
NN after Reduction	**70**	0.0053
NN with EA and Reduction	**78.4**	1.0023
NN of Reference	**80**	0.0459

As the chart above indicates that the test accuracy is improved by using the method of EA. One big problem is that we sacrifice considerable of running time to archive the high test accuracy. One of the reason I believe is that in the part of EA, I define a *'Fun'* function which output the test accuracy of a NN which contain a set of features. To compute that result I train each NN 50 times. However, I initialize 20 population of NN. In all, the process of computing the result of *'Fun'* is very time consuming. Compare to the result of reference NN's performance, mine is not satisficing. After analyze their paper, the advantage of using different optimizer function and the implement of multilayers NN can improve the NN's performance to a new level.

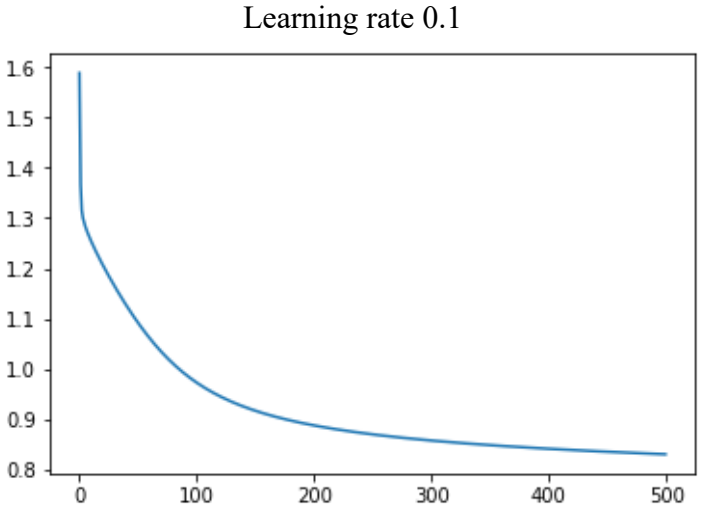

Fig3 loss function in each training epoch (500 in all)

The Fig3 can help us to get a further insight about the struct of the NN. After the 500 numbers of training the loss is come close to 0 as we expected which also indicate one thing that the too many times of training may nor help the performance of the NN.

```
Confusion matrix for training:tensor([[ 13.,   5.,   5.,   0.,  22.],
                                      [  7.,  11.,   7.,   0.,   5.],
                                      [  7.,   7.,  11.,   0.,   2.],
                                      [  5.,   3.,   3.,   0.,   1.],
                                      [  7.,   2.,   0.,   0., 125.]])
Testing Accuracy: 68.18 %
Confusion matrix for testing:tensor([[ 4.,  0.,  1.,  0.,  5.],
                                     [ 3.,  0.,  1.,  0.,  2.],
                                     [ 3.,  5.,  0.,  0.,  0.],
                                     [ 1.,  0.,  0.,  0.,  0.],
                                     [ 1.,  1.,  0.,  0., 28.]])
```

Fig4. Confusion matrix

In all, the performance is improved by the EA.

Conclusion and Future Work

As we discuss and present before either the performance and the test accuracy has improved. However, personally, the accuracy for the network is not good enough, or in another way, the range of increasing is not good enough. One of the reason, I think, is using too less number of hidden layers. To future work, we should improve the training time. And we should also enforce the bidirectional network which improve the efficiency in both connection between the neuron. And in the part of using EA, the average training time is too large. Another thing that worry me is that in the EA part. Even we initial a random mix of the 0,1 arrays to represent the different match of features, we can still not guarantee that after the EA the result is the best. So, in the future, if the situation is allowed, I want to test all the match possible for example in my data set, 2^{13} number of possible match.

Reference:

1. UCI – Maching Learning Repository: Brest Cancer Wisconsin (Diagnostic) Data Set. https://archive.ics.uci.edu/ml/datasets/Heart+Disease

2. *Gedeon, T.D. and Harris, D. (1991) "Network Reduction Techniques," Proceedings International Conference on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 1: 119-126*