Improving Neural Networks with a Pruning Methodology and Genetic Algorithm

Ruiqi Wang
Research School of Computer Science,
Australian Nathional University,
Canberra, ACT 2601
u6342380@anu.edu.au

Abstract. Designing an artificial neural network is a subjective process, especially when it comes to select the hyperparameters and appropriate size of the network. Over the years, many researchers have made attempts to optimize the structure of ANNs. In the paper, I present my experiment and results on some breast cancer diagnostic data and the task was to predict the type of the breast mass. To increase the prediction accuracy, I have present an optimized neural network which implements a genetic algorithm to optimize the initial weights and biases and a network pruning method to remove the excess units by identifying the activity of them. By implementing the genetic algorithm, the networks accuracy increases from 94.64% to 96.1%. After pruning, the test accuracy has a slight improvement and the learning time is shortened, which shows that a reduced network is more efficient. During the pruning process, I also found that deleting too many excess units might cause the test accuracy to drop significantly as the network isn't able to learn well. The same dataset has also been utilized by several studies with the accuracy beyond 97% and the results are compared with my study.

Keywords: Artificial Neural Networks, Pruning, Genetic Algorithm, Vector.

1 Introduction

In the last decade, with the in-depth study of neural networks, Artificial Neural Networks (ANNs) have successfully solved a great number of practical problems in the fields of financial forecasting, pattern recognition, biology, medicine and etc. which modern computers have been struggled with. There is no doubt that ANNs will be used to solve more complex real-world problems in the future. Therefore, increasing the prediction accuracy of ANN models is of great importance. Many attempts have been made over the years to build more accurate and efficient neural networks. The objective of this paper is to present a GA-ANN hybrid model and then refine it with a pruning methodology called distinctiveness analysis.

Back-propagation algorithm is a widely applied learning algorithm for neural networks. The weights and biases of the network are assigned initial values first, and then the error between the target output and the actual output is calculated and back-propagated through the network to update the weights and biases [1]. Although BP algorithm is the most commonly used training algorithm, it has two significant limitations. It is usually unable to escape local optima and also works very slow in convergence. The genetic algorithms (GAs) are then proposed to overcome these drawbacks.

Genetic algorithms, developed by John Holland in 1970, are commonly used to generate high-quality solutions by simulating natural selection, mutation and crossover. They belong to a larger class of evolutionary algorithms and are good at finding optimal solutions by searching large and complex space in a relatively short time. There are many studies have found that GA can significantly increase the accuracy of the ANN models. Garima Singh and Laxmi Srivastava built a genetic algorithm based back propagation neural network (GABPNN) in 2011 for voltage stability margin estimation [2]. Their model turned out to be very efficient. It learned faster and provided more accurate estimations compared to gradient based back propagation neural networks [2]. In 2012, Yu-Tzu Chang and Jinn Lin also proposed an optimized neural network using GA to predict the probabilities of hip fractures [3]. In their model, the genetic algorithm was used to find the optimal set of initial weights. And the result showed that the performance of their model was highly enhanced by implementing the genetic algorithm [3].

Many applications of neural networks have also emphasized the size of network needed to solve a particular problem. In order to keep the size of network as small as possible without affecting the performance, pruning methodology is introduced to identify and automatically delete redundant neurons. The very first network pruning method was brought out in 1988 by Sietsma and Dow [4]. They analysed the output of each hidden unit to determine whether they contribute to the solution or not. The duplicated neurons were removed along with nonfunctional ones. However, the two stages pruning process operated by inspection they proposed is very complex even on small dataset. Many other researchers also have made attempts to solve this problem and they have defined a number of important concept to do that such as relevance [5], contributions [6], badness [7] and distinctiveness [8]. In this paper, I have implemented the network pruning method discussed by Gedeon and Harris in " Network reduction techniques" [8].

The dataset I use is the Diagnostic Wisconsin Breast Cancer Dataset, which has 32 features and 569 instances. The motivation for choosing this dataset is that breast cancer is one of the major malignant tumors in the world and it has threatened many women's lives. Highly accurate neural network models are required in the diagnosis of breast cancer. The features of each instance in this dataset are calculated on a digital image of a fine needle aspirate (FNA) of a breast lump, which describe characteristics of the cell nuclei present in the image. The dataset is linear separable by all input features and the output results are divided into two categories – benign or malignant. When massaging the dataset, I drop the column of ID number, convert the target values in the dataset into 0s (benign) and 1s (malignant) as we need numeric values for training a neural network, and then split the pre-processed dataset into training set and test set using 10-fold cross validation.

The remainder of this paper is organized as follows. Section 2 describes the ANN model trained by back propagation algorithm and the improvements using genetic algorithm. Section 3 introduces the pruning method used to prune the network's hidden layer. Section 4 evaluate the results with a comparison against several studies using the same dataset. Finally, section 5 presents the conclusion and future works.

2 The Network

2.1 Initialize the ANN Model

In the paper, I have initially built a fully connected three-layer feed-forward network with 30 input neurons corresponding to 30 input features, 20 hidden neurons and 2 output neurons separating the input patterns into two classes -- benign and malignant. All connections in the network are only from one layer directly to the layer above it, without lateral, backward connections. Each unit has a simple weighted connection from each unit in the layer above. The network is trained using the training set with the desired targets and uses back-propagation algorithm to do error propagation [9].

The neural network usually adopts the sigmoid function as the activation function as it is easy to calculate and saves time. However, if we build a deep network with multiple hidden layers, the sigmoid function will squeeze the error in the backpropagation process at each layer, which means some big error may not be able to affect synapse weights of neurons from a shallow layer. Due to that, the result we get may not be ideal as other types of nonlinear activation functions. In my neural network, I use leaky rectified linear unit as the hidden units' activation function [10] and a standard SoftMax classifier serves as the output layer. Leaky ReLU activations basically allow a small, positive gradient if the input is negative and if the input is positive, the output is equal to the input. Thus, for positive inputs, the derivative is always 1, which won't cause the squeezing effect like sigmoid function. Research has shown that leaky rectified linear units can speed up the learning process for large networks [11]. The SoftMax function in the output layer will output a categorical probability distribution which tells us the probability that any of the classes are true.

The loss function is defined as cross entropy without regularization [12]. The network is optimized by stochastic gradient descent (SGD) with momentum given a weight of 0.8.

2.2 Improve the Model Using a Genetic Algorithm

Due to the drawbacks of BP algorithm, a genetic algorithm is implemented to enhance the model's performance. A genetic algorithm usually starts from a population of candidate solutions and iterates generation by generation. In each generation, the fitness of each solution is evaluated and only the more fitted ones are selected and mutated to produce the next generation. After many generations, the population with the best characteristics are formed and the best solution is found.

In this paper, the genetic algorithm is utilized to find the optimal initial weights and biases of the network. Then the network is trained using back propagation algorithm with these determined hyperparameters. The process flow of the hybrid GA and BP algorithm is showed in figure 1. Basically, a standard GA algorithm works as follows [1]:

Step 1. The weights and biases of the network are encoded into a list of numbers to generate a collection of chromosomes which is also called a population. Each of the chromosome is a set of weights and biases.

Step 2. Assign each set of weights and biases to the connections of the network, run the network using back propagation algorithm over the training patterns and return the sum of the squares of the errors. And the fitness value of each chromosome is defined to be the inversely proportional to the error.

Step 3. Rank all chromosomes based on their fitness values and discard those with low fitness values. Pass the rest chromosomes to the next generation.

Step 4. Perform the crossover and mutation operations to generate new offspring. Evaluate the fitness value of the new chromosomes and replace worse chromosomes with more fitted new ones. Then we get a new generation.

Step 5. Repeat step 2-4 until reach the maximum number of generations or a satisfactory fitness level for the population.

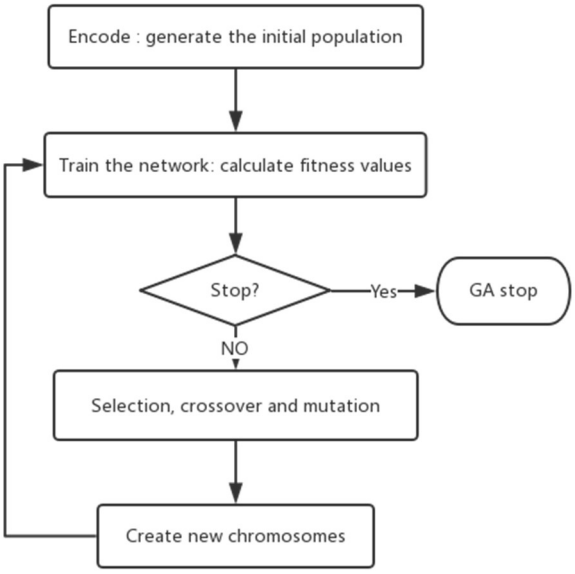

Fig.1 Process flow of the hybrid GA and BP algorithm.

3 Network Pruning Method

The number of neurons in the hidden layer is related to the efficiency of the network. Therefore, it needs to be determined carefully. If the number is too small, the network will not be able to study well. Even if it is possible to learn, it will take a long time to train and the accuracy is very likely to be low. And if the network includes more neurons than it actually needs, the learning process will be long and other problems might occur such as overfitting. In this paper, I focus on using the distinctiveness to determine redundant units and remove them automatically.

The definition of a hidden neuron's distinctiveness is its output activation vector over the pattern presentation set [8]. So, for each neuron in the hidden layer, I compute a vector with the same dimensionality as the number of patterns in the training set and that vector represents the functionality of the hidden neuron in the input pattern space. Each component in that vector is related to the output activation of that neuron. Therefore, no matter what the neuron's output is, the duplicated neurons' vectors are always identical. And neurons with short vectors will be recognized as unimportant to the network's functionality and can be removed.

Unfortunately, it is hard to see a neuron's behavior in the network by only looking at its activation vector. It will be easier if the hidden neurons are compared pair by pair. Showing the similarity of two neurons can be done by calculating the angle between their vectors. If the angle is too small, the two neurons are considered to have similar functionality and one of them can be deleted. And if the vector angle is larger than a given value, both of them can be removed for being complementary. The results are evaluated and discussed in the next section.

4 Evaluation and Discussion

In conducting my experiment, I first trained the initial neural network using BP algorithm without optimizing initial weights and biases. Then after recording the results, I adjusted the weights and biases of the initial neural network using genetic algorithm and tested the performance of the GA-ANN model. Finally, I pruned the network by analyzing the distinctiveness of the hidden units to improve its efficiency.

4.1 Comparison of the performances between the first two experiments

In my first experiment, I set the initial number of hidden neurons as 20 with the learning rate 0.05 and train the simple three-layer network using back propagation algorithm for 1000 epochs, resulting in a test accuracy of 94.64 %. To evaluate the training process, I have printed out the loss and training accuracy after each 50 epochs and also plotted the historical losses which is calculated by the cross-entropy loss function. The result is demonstrated in figure 2 and 3.

```
Epoch [1/1000] Loss: 0.7655  Accuracy: 37.04 %
Epoch [51/1000] Loss: 0.7070  Accuracy: 37.04 %
Epoch [101/1000] Loss: 0.6754  Accuracy: 82.65 %
Epoch [151/1000] Loss: 0.6551  Accuracy: 65.89 %
Epoch [201/1000] Loss: 0.6388  Accuracy: 64.13 %
Epoch [251/1000] Loss: 0.6236  Accuracy: 64.91 %
Epoch [301/1000] Loss: 0.6077  Accuracy: 67.25 %
Epoch [351/1000] Loss: 0.5898  Accuracy: 71.15 %
Epoch [401/1000] Loss: 0.5698  Accuracy: 76.41 %
Epoch [451/1000] Loss: 0.5487  Accuracy: 80.12 %
Epoch [501/1000] Loss: 0.5265  Accuracy: 84.21 %
Epoch [551/1000] Loss: 0.5034  Accuracy: 87.33 %
Epoch [601/1000] Loss: 0.4796  Accuracy: 87.91 %
Epoch [651/1000] Loss: 0.4556  Accuracy: 89.67 %
Epoch [701/1000] Loss: 0.4319  Accuracy: 91.42 %
Epoch [751/1000] Loss: 0.4088  Accuracy: 92.59 %
Epoch [801/1000] Loss: 0.3868  Accuracy: 92.98 %
Epoch [851/1000] Loss: 0.3660  Accuracy: 93.18 %
Epoch [901/1000] Loss: 0.3466  Accuracy: 93.37 %
Epoch [951/1000] Loss: 0.3287  Accuracy: 93.57 %
```

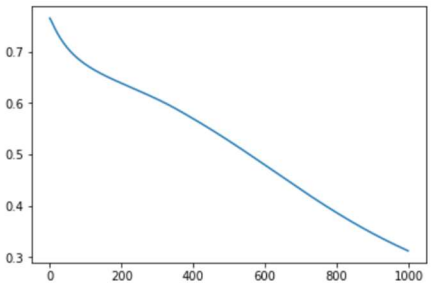

Fig.2 Training loss and accuracy of the initial network. Fig.3 Historical loss during training.

In artificial intelligence, confusion matrix is a useful visual tool to compare the classification results with the actual values, especially for supervised learning. It allows more detailed analysis for the neural network than just the training accuracy. I have computed the confusion matrix and calculated the number of correct predictions, false positive predictions and false negative predictions for training and test (Fig.4). As we can see from the table, among all 513 images in the training set, the network successfully identified 475 of them, and 36 benign tumors were identified as malignant while 2 malignant ones were predicted to be benign. As for the test set, the network only got three wrong result and all three were false positives, which were better than false negatives.

	Correct	False-Positive	False-Negative
training	475	36	2
test	53	3	0

Fig.4 Confusion matix for initial network

In the second experiment, the initial weights and biases of the network were optimized by the GA algorithm introduced in section 2. The parameters of the GA algorithm are selected by trial and error through experiment for good performance. The initial population size is set to be 100, the probability of crossover is 0.7 and the probability of mutation is 0. Figure 5 and 6 present the training results after optimizing the hyperparameters of the network. As we can see, the

training accuracy was improved significantly compared to the first experiment.

```
Epoch [1/1000] Loss: 0.8142  Accuracy: 38.79 %
Epoch [51/1000] Loss: 0.5248  Accuracy: 89.67 %
Epoch [101/1000] Loss: 0.2172  Accuracy: 94.35 %
Epoch [151/1000] Loss: 0.1452  Accuracy: 96.88 %
Epoch [201/1000] Loss: 0.1152  Accuracy: 97.08 %
Epoch [251/1000] Loss: 0.0987  Accuracy: 97.66 %
Epoch [301/1000] Loss: 0.0885  Accuracy: 98.05 %
Epoch [351/1000] Loss: 0.0818  Accuracy: 98.05 %
Epoch [401/1000] Loss: 0.0770  Accuracy: 98.05 %
Epoch [451/1000] Loss: 0.0734  Accuracy: 98.05 %
Epoch [501/1000] Loss: 0.0705  Accuracy: 98.05 %
Epoch [551/1000] Loss: 0.0682  Accuracy: 98.25 %
Epoch [601/1000] Loss: 0.0663  Accuracy: 98.25 %
Epoch [651/1000] Loss: 0.0646  Accuracy: 98.25 %
Epoch [701/1000] Loss: 0.0631  Accuracy: 98.25 %
Epoch [751/1000] Loss: 0.0618  Accuracy: 98.25 %
Epoch [801/1000] Loss: 0.0607  Accuracy: 98.25 %
Epoch [851/1000] Loss: 0.0596  Accuracy: 98.25 %
Epoch [901/1000] Loss: 0.0587  Accuracy: 98.25 %
Epoch [951/1000] Loss: 0.0578  Accuracy: 98.25 %
```

Fig.5 Training loss and accuracy of GA-ANN model

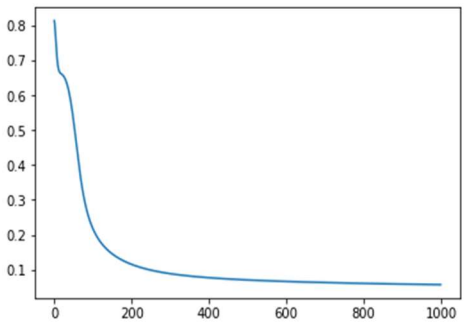

Fig.6 Historical loss

After 10 times of training and testing, the average test accuracy increased to approximately 96.1%. Figure 7 shows the comfusion matrix of the GA-ANN hybrid model.

	Correct	False-Positive	False-Negative
training	493	16	4
test	54	2	0

Fig.7 GA-ANN model

Compared to figure 4, we can see that the performance of the network has been improved. Among 513 images in the training set, the network correctly classified 493 of them along with 16 benign tumors which were diagnosed as malignant and 4 malignant as benign. And in the test set, there were only two incorrect diagnoses. This agrees with my initial hypothesis that genetic algorithm can improve the network's prediction accuracy. However, this model now has a risk of overfitting the data.

4.2 Improvement after pruning

In the third experiment, the pruning method was implemented to prevent overfitting and the similarity of pairs of vectors was recognized by calculating the angle between them in pattern space.

During the pruning process, the network would iteratively delete one of the two neurons with vector angle less than 5° and both neurons with vector angle larger than 175°. As I remove hidden units from the network, the mean squared error increases significantly, which shows that initial model overfits the data. After several iterations, the initially 20 hidden neurons were reduced to 5 and all of their vector angle were in the range of 5° to 175°. However, when I run the network over the test patterns with only 5 hidden neurons, the test accuracy dropped down to 85%. This agrees with my hypothesis that removing too many hidden units may affect the network's learning ability. I then added two more redundant neurons into the hidden layer and test again. The mean test accuracy now is approximately 96.3%. Compared to the result before pruning, the network has a slight improvement by reducing the hidden layer size. In figure 8, I present the final vector angles for the 7 hidden units.

Pair of units		Vector Angle
1	2	41.486813
1	3	25.969786
1	4	134.155365
1	5	35.463394
1	6	21.452768
1	7	138.014679
2	3	15.517054
2	4	175.642105
2	5	6.023466
2	6	20.034050
2	7	179.501480
3	4	160.125122
3	5	9.493616
3	6	4.517049
3	7	163.984467
4	5	169.618744
4	6	155.608124
4	7	3.859351
5	6	14.010639
5	7	173.478104
6	7	159.467468

Fig.8 Vector angles for pairs of the hidden units after pruning

Another advantage of using pruning methodology is that reducing the size of the network can reduce the learning time. Before pruning the network, the learning time of the network with 20 hidden neurons is 0.489 seconds, and with 7 hidden units, the network only needs 0.415 seconds to complete the training phase. It may seem not a big deal, but with a really large dataset with millions of patterns, the learning time will be significantly shortened.

4.3 Comparison with similar studies

In order to find deficiencies of my work, I look up some previous papers which uses the same dataset and compare our results.

In 1993, three researchers from the University of Wisconsin first created the Wisconsin Diagnostic Breast Cancer dataset and they achieved a ten-fold cross-validation accuracy of 97% [10]. They used an MSM method for the classification procedure which iteratively places a series of planes to separate features in the feature space. And in order to increase the prediction accuracy, they intended to minimize both the separating plane and the number of features by using a combination of a few features which contributed most to the classification problem. After experimenting, they chose three of the thirty features: mean texture, worst area and worst smoothness and successfully separated 97% cases using a single separating plane. The reason why my result is not as good as theirs is that I input all features instead of choosing the most important ones.

Grąbczewski Krzysztof and Włodzisław Duch presented a method for creating heterogeneous forests of decision trees based on Separability of Split Value (SSV) criterion to improve the accuracy of classification problems [13]. In their study, they applied the method on the Wisconsin breast cancer dataset and got a surprisingly good result with 97.4% accuracy.

Hussein A. Abbass presented an evolutionary artificial neural network (EANN) approach based on the pareto-differential evolution (PDE) algorithm for the prediction of breast cancer [14]. His approach had better generalization and much lower computational cost compared against an evolutionary programming approach and standard backpropagation. And the best performances for the breast cancer dataset achieved an average accuracy of 0.981 ± 0.005. The evolutionary algorithm they implemented is more efficient than a standard GA algorithm, therefore their result is much better than mine.

5 Conclusion and Future Work

In this study, I implement two techniques to improve the performance of the network. I adjust the initial weights and biases of the neural network using genetic algorithm and the result shows that the GA-ANN model can provide better test accuracy. I then iteratively prune the model's hidden layer by calculating the distinctiveness of the hidden units and find that network reduction can prevent overfitting and improve the performance of the network by shortening the learning phase. But deleting all redundant neurons may result in low test accuracy as the network won't have enough neurons to

learn. So, keeping the size fit is important. Sometimes it's necessary not to delete all functionless units and leave one or two, they can be very helpful in increasing the test accuracy. By comparing the result against similar studies, I find that the prediction performance of this study may be improved further in three means. The first method is to implement other pruning methodologies such as relevance analysis and etc. The second method is to prune the input layer instead of the hidden layer and see how that will affect the test accuracy. Third, we can also try other optimal methods other than genetic algorithm to optimize the hyperparameters of the network.

Reference

1. Qiu, M. and Song, Y.: Predicting the direction of stock market index movement using an optimized artificial neural network model. PloS one, 11(5), p.e0155133. (2016)
2. Garima Singh, Laxmi Srivastava, Singh, G., Srivastava, L.: Genetic algorithm-based artificial neural network for voltage stability assessment. Advances in Artificial Neural Systems, 2011, 4. (2011)
3. Chang, Y.T., Lin, J., Shieh, J.S., Abbod, M.F.: Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction. Advances in Fuzzy Systems, 2012, p.6. (2012)
4. Sietsma, J., Dow, R.J.: Neural net pruning-why and how. In: IEEE international conference on neural networks, pp. 325-333. (1988)
5. Mozer, M.C., Smolensky, P.: Using relevance to reduce network size automatically. In: Connection Science, 1(1), pp.3-16. (1989)
6. Sanger, D.: Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. In: Connection Science, 1(2), pp.115-138. (1989)
7. Hagiwara, M.: Novel backpropagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection. In: Neural Networks, International Joint Conference, pp. 625-630. IEEE. (1990)
8. Gedeon, T. D., D. Harris.: Network reduction techniques. Proceedings International Conference on Neural Networks Methodologies and Applications. Vol. 1. 1991.
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science (1985)
10. Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. In: Biomedical Image Processing and Biomedical Visualization (Vol. 1905, pp. 861-871). International Society for Optics and Photonics. (1993)
11. Maas, Andrew L., Awni Y. Hannun, Andrew, Y. Ng.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml, vol. 30, no. 1, p. 3. 2013. (2013)
12. Rubinstein, R.Y. and Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Springer-Verlag, New York (2004)
13. Grąbczewski, K. and Duch, W.: Heterogeneous forests of decision trees. In: International Conference on Artificial Neural Networks (pp. 504-509). Springer, Berlin, Heidelberg. (2002)
14. Abbass, H.A.: An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial intelligence in Medicine, 25(3), pp.265-281. (2002)