Neural Network Structure Optimization Based on Genetic Algorithm

Ziying Guo

Research School of Computer Science
Australian National University
Canberra, Australia
u7150131@anu.edu.au

Abstract. The construction and adjustment of a neural network is very complicated. Because it involves many hyperparameters, such as the structure of the neural network, including the number of hidden layers and hidden neurons, as well as the learning rate and so on. Adjusting a neural network would be very time-consuming, but these parameters greatly affect the effect of model training. This article aims to propose a method to automatically optimize the network topology. We use a binary vector to represent the structure of a neural network and optimize the topology of the neural network through genetic algorithm. The experiments on the depression dataset [1] prove that the network optimized by the genetic algorithm can not only improve the classification accuracy, but also greatly accelerate the convergence speed. Besides, we found the design tactics of GA's fitness function.

Keywords: Neural Network; Classification; Genetic Algorithm; Depression

1 Introduction

With increasing social pressure, the prevalence of depression is increasing year by year. Depression is a kind of affective disorder [2], and it is a serious chronic health condition. The symptoms are mainly long-lasting depression, unable to experience happiness in any activities. In severe cases, it may also cause suicidal attempts [3]. Therefore, an effective diagnosis of depression is the key to avoiding tragedies. But most of the existing diagnosis methods are based on self-reported questionnaires, that are not objective enough. Hence, it is an objective and efficient way to use neural networks to identify and classify different levels of depression.

The development of neural networks has improved the quality of human life in various aspects, and the research on improving the performance of neural networks has never stopped. Nowadays, neural networks have a very wide range of applications in all aspects. However, NNs have very complicated configurations including network topologies and several hyperparameters, which greatly affect the performance of classification effect based on neural networks. [4] Usually, people need to estimate the complexity of the task to estimate the approximate topology of the neural network. Then repeated testing and adjusting the network based on the results, which is troublesome and time-consuming. This paper aims to overcome this problem by finding a way to automatically optimize the network structure without human labor.

Genetic algorithm (GA) is a meta-heuristic inspired by the process of natural selection and are often viewed as function optimizers [5]. This algorithm represents the candidate solution as a vector called chromosome, and continuously optimizes the population genes according to the fitness function through biologically inspired operators such as mutation, crossover and selection, which is very suitable for solving NP problems.

In this paper, we conducted experiments on the depression dataset [1]. We trained several multi-class classifiers based on the neural networks with four different structures. Two of them were estimated by people according to experience, which were regarded as the baselines. The other two were optimized by genetic algorithm with different fitness functions. And proved the usability of our method by comparison these results.

2 Dataset

In this paper, we used the depression dataset from Xuanying Zhu et al.'s work [1]. This dataset records the 192 biological signals of 12 testers (with no prior knowledge of depression recognition) during they are watching 16 depression videos. The biological signals contain *Galvanic Skin Response (GSR)* with 23 features, *Skin Temperature (ST)* with 23 features, *Pupillary Dilation (PD)* with 39 features, and all features *(ALL)* with 85 features. This dataset divides the depression level into 4 categories: *0* for no or minimal depression, *1* for mild depression, *2* for moderate depression, and *3* for severe depression.

3 Methodology

3.1 Data preprocessing

Normalization. Different features often have different measurement standards and units, which will affect the results of data analysis. In order to eliminate this kind of impacts, the data needs to be normalized to be restricted to the same scale. In this paper, we use Z-score normalization:

$$x^* = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean, and σ is the standard deviation of the original data. For the depression dataset, the biological signal of each individual has subtle differences, normalization can also weaken these differences between different testers.

Features Selection. When dealing with data containing a large number of features, features selection becomes very necessary. Because these features are likely to contain duplicate information and information irrelevant to classification, which will affect classification performance. Also, high-dimensional data is not only not conducive to model convergence, but also computationally expensive. Hence, dimensionality reduction cannot be ignored. In this paper, we use Genetic Algorithm (GA) to perform features selection.

We use a binary array whose length is the number of features to represent a chromosome. 1 means the corresponding feature is retained while 0 means removed. All the values of the chromosomes of the initial population are 1, which means use all features. Through repeated fitness-based selection, crossover and mutation, the chromosomes of each individual in the population would be very similar and tend to no longer change. Finally perform feature selection based on the chromosome array of the individual with best fitness.

In this paper, we continue to use the main parameters in Xuanying Zhu et al.'s work [1] as Table 1. In addition, we define the fitness function based on Linear Discriminant Analysis (LDA). The core idea of LDA is minimizing the within-class scatter and maximizing the between-class scatter. LDA could easily handle the case where the within-class frequencies are unequal [6].

We calculate the within-class scatter matrix S_W and the between-class scatter matrix S_B according to the following formulas:

$$\mathrm{S}_W = \sum_{\mathrm{i}=1}^{\mathrm{C}} \sum_{\mathrm{x}\in\mathrm{N_i}} (\mathrm{x} - \mathrm{m_i})(\mathrm{x} - \mathrm{m_i})^{\mathrm{T}} \tag{2}$$

$$S_B = \sum_{i=1}^{C} \sum_{x\in N_i} (m_i - m)(m_i - m)^T \tag{3}$$

where C is number of classes, m is the mean of the whole data, N_i is the number of samples of class i and m_i is the mean of class i. Therefore, the fitness F is defined as:

$$F = \frac{tr(S_B)}{tr(S_W)} \tag{4}$$

Finally, we got 16 *GSR* features, 7 *ST* features, 22 *PD* features, and 45 for *ALL* of these features.

Table 1. Implementation settings for GA

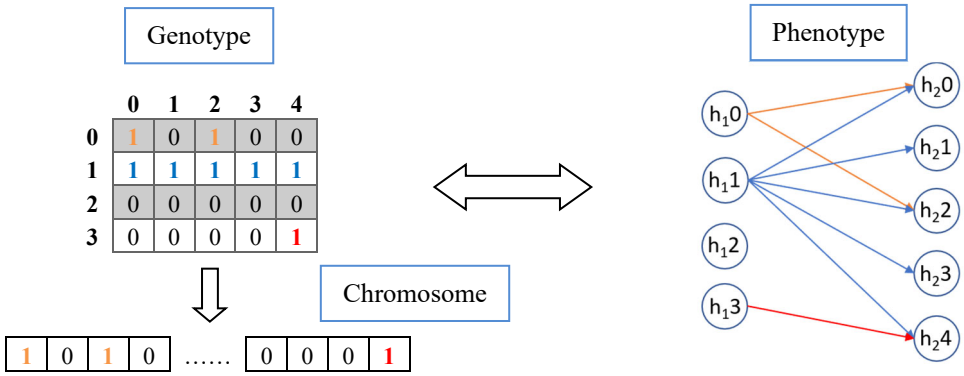
GA Parameter	Value
Chromosome encoding	Binary
Chromosome length	Number of features
Chromosome initialization	All 1s
Population size	100
Crossover rate	0.8
Mutation rate	1 / (length of chromosome)
Crossover type	Uniform crossover
Mutation type	Uniform mutation
Selection type	Roulette wheel selection
Number of generations	1000

3.2 Genetic representation of structured Neural Networks

When applying structure optimization to Neural Networks through Genetic Algorithm, one issue is how to represent the structure of a neural network as a chromosome [4]. In this paper, we do not consider cross-layer connections. This means that the number of connections should be at most mn in two layers that respectively contain m and n neurons, which is fully connected. And our target is to find which connections to be deleted to obtain the network structure with the best classification effect.

To overcome this problem, we consider the connections between any two layers of a neural network as a two-dimensional matrix, called connection matrix C. Element c_{ij} in C represents the connection between neuron i in the current layer and neuron j in the next layer. If c_{ij} equals 1, connection exists, otherwise, the connection does not exist [4]. The relation between the genotype and phenotype are shown in Fig 1. Flatten the connection matrix C could obtain a binary vector of length mn, which should be the chromosome.

Fig. 1. Example of the relation between the genotype and phenotype



In this paper, we constructed a Neural Network with two hidden layers, the number of hidden layer neurons were 100 and 50, which means this network should contains 3 genotype connection matrices. They were respectively:

- C_1: a $N \times 100$ matrix, represents connections between input neurons and neurons in the first hidden layer.
- C_2: a 100×50 matrix, represents connections between neurons in the first and second hidden layer.
- C_3: a $50 \times M$ matrix, represents connections between neurons in the second hidden layer and output neurons.

where N is number of input features, M is number of depression levels, which is 4. So the chromosome would be the result of flattening and concatenating these matrices.

3.3 Genetic algorithm

After encoding the chromosomes, the next step is to initialize the population. In this paper, we set the population size to 500 and initialize the population chromosome sequences randomly. After that, this population would undergo 20 rounds of evolution, including selection, crossover and mutation, and finally form a new population with better chromosomes.

Fitness function. Optimizing the topology of a neural network is an NP problem, which means that it is difficult to solve, but its solutions could be evaluated relatively easier. Fitness function is used to measure the quality of genes, that is, the solutions. The main purpose of optimizing the network structure is to improve classification testing accuracy and reduce testing loss. So we built two fitness functions in accordance with these two ideas.

In this paper, we used leave-one-participant-out cross-validation to train 12 models for 5 epochs with a learning rate of 1e-3. The structure of the network was represented by the chromosome. And calculated the average testing accuracy and loss of the models. Then the two fitness functions were defined as:

$$F_{acc} = Accuracy_5 \tag{5}$$

$$F_{loss} = \frac{1}{Loss_5} \tag{6}$$

Selection. This step is to imitate natural selection in nature. Individuals with higher fitness have a greater chance of surviving. In order to achieve a better selection strategy with less complexity, we used Binary Tournament Selection. Each time randomly selecting two individuals from the entire population with equal probability, then choose the individual with the higher fitness of the two to enter the offspring population [7].

Crossover. This step is to imitate gene recombination and crossover in nature. This is a process that recombine and distribute genes from two parental chromosomes and form progeny chromosomes. There are two key points in this

process, that is, crossover rate and crossover type. In this paper, we used adaptive crossover rate and single point crossover.

In order to avoid the situation that individuals with high fitness cannot enter the progeny population, we let individuals whose fitness are higher than the average fitness of the population have a lower crossover rate [8]. The crossover rate could be expressed as:

$$P_c = \begin{cases} \dfrac{k_1(f_{max} - f)}{f_{max} - f_{avg}} & f \geq f_{avg} \\ k_2 & f < f_{avg} \end{cases} \tag{7}$$

where f_{max} is the maximum fitness in the population, f_{avg} is the average fitness in the population, and f is the greater fitness of the two individuals to be crossed. k_1 and k_2 are constants, and $k_1 < k_2$ [8]. In this paper, we set k_1 as 0.6 and k_2 as 0.9.

Mutation. This step is to imitate gene mutation in nature. This process aims to randomly change part of the chromosomes to maintain the diversity of the population. There are also two key points in this process, that is, mutation rate and mutation type. In this paper, we used adaptive mutation rate and bit-flip mutation.

This adaptive method of adjusting mutation rate is also to retain individuals with higher fitness. The mutation rate could be expressed as:

$$P_m = \begin{cases} \dfrac{k_3(f_{max} - f')}{f_{max} - f_{avg}} & f' \geq f_{avg} \\ k_4 & f' < f_{avg} \end{cases} \tag{8}$$

where f_{max} is the maximum fitness in the population, f_{avg} is the average fitness in the population, and f' is the greater fitness of the individual to be mutated. k_3 and k_4 are constants, and $k_3 < k_4$ [8]. In this paper, we set k_3 as 0.3% and k_4 as 0.8%.

3.4 Neural Networks Based Classification Models

In this paper, we build four Neural Networks with different topologies:

- *NN*: a fully connected neural network with a single sigmoid hidden layer. The size of the hidden layer is 50.
- *NN2*: a fully connected neural network with two hidden layers. The first hidden layer is a ReLU hidden layer of size 50, and the second hidden layer is a sigmoid hidden layer of size 10.
- *NN2_acc*: a non-fully connected neural network with two hidden layers, whose topology is calculated by genetic algorithm using F_{acc} as the fitness function. The first hidden layer is a ReLU hidden layer of size 100, and the second hidden layer is a sigmoid hidden layer of size 50.
- *NN2_loss*: a non-fully connected neural network with two hidden layers, whose topology is calculated by genetic algorithm using F_{loss} as the fitness function. The first hidden layer is a ReLU hidden layer of size 100, and the second hidden layer is a sigmoid hidden layer of size 50.

All NNs were trained using backpropagation with the Cross-Entropy loss function, and the output layer had 4 output neurons, representing the four depression levels. In addition, we tried a variety of optimizers, and finally we chose the Adam optimizer with weight decay. In order to avoid the situation that some of the 12 models suffer from overfitting while others are still underfitting in the 12 models, we use early stopping. If the validation loss improves in 10 consecutive iterations, then stop training.

We use leave-one-participant-out cross-validation for the depression dataset [1]. The data of 11 of the 12 testers is used for training, and the data of the remaining 1 tester is used for testing, and repeat for all. In this way, 12 models will be trained. Finally, we average the 12 models to get the final results.

3.5 Evaluation Measures

To validate and compare the effectiveness of our models, we use *accuracy* for multi-class classification to measure the comprehensive performance of the model to classify all four classes. Since the depression dataset is uniform distributed for the four different classes, *accuracy* could be meaningful. Additionally, we will also compare the loss curves and the number of epochs required for the model to converge during training to evaluate the model's stability and convergence speed.

4 Results and Discussion

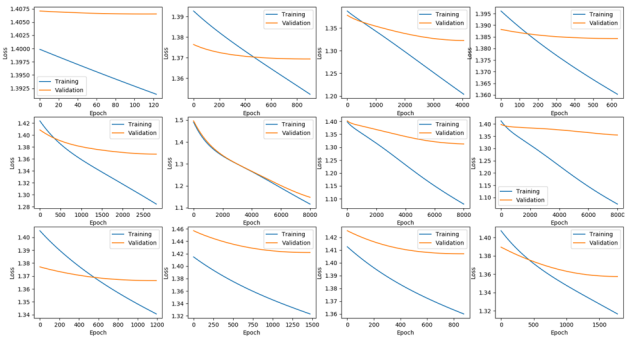
4.1 Model stability and convergence speed

We trained *NN2_acc* and *NN2_loss* with a learning rate of 1e-3 for at most 800 epochs. And tried to train *NN2* with the same hyperparameters, but the model was always early stopped very early (at most 120 epochs) and fallen into overfitting without getting a good result. To solve this problem, we had to reduce the learning rate to 1e-5 and increase training epochs to 8000. The loss curves during training are shown as Fig 2 (the loss curves of *NN2_acc* are similar to those of *NN2_loss*), and the classification accuracy of the models are shown as Table 2. It could be seen that the convergence situation of *NN2* is far worse than that of *NN2_loss*, whose structure has been optimized by GA. Because in the left image, the training loss continues to decline at the end of the training. And from observing the validation loss, the models either did not have time to learn, which is underfitting, or triggered early stopping due to overfitting. In contrast, the models in the right image basically fully converge within 800 epochs. Therefore, the network whose structure was optimized by genetic algorithm could be not only more stable, but also converged faster.

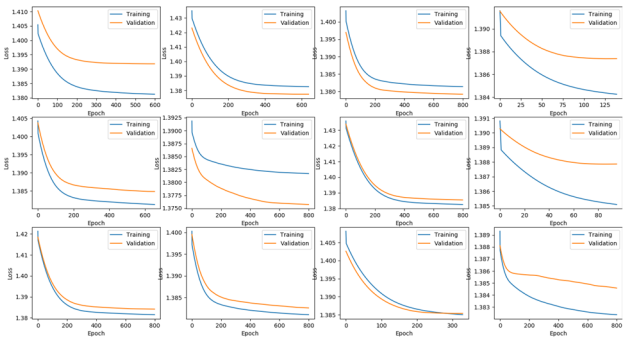
Table 2. Accuracy of models trained by *NN2* and *NN2_loss* with different topologies on *ALL* features

Models	Accuracy
NN2 (lr = 1e-3)	25.000
NN2 (lr = 1e-5)	33.333
NN2_loss	**35.417**

Fig. 2. The loss curves during training *NN2* and *NN2_loss* with *ALL* features



(a) The loss curves for training *NN2*



(b) The loss curves for training *NN2_loss*

4.2 Classification effect

Table 3 and Fig 3 shows the classification accuracy of the models trained by neural networks with different topologies on different features. It could be seen that the accuracy of *NN1* on *GSR*, *PD* and *ALL* features is higher than *NN2*, which indicates for classification on depression dataset, simpler network would be more effective in most cases. This may because this classification task is relatively simple, a network with a simple structure could work, while a complex network structure would easily cause overfitting instead. However, for *ST* feature, the accuracy of *NN2* is higher than that of *NN1*, this may because the relationship between *ST* feature and depression level is more complicated. A simple network is not enough to explore the connections, so a complex network structure works better. This shows that it is difficult for these fixed fully connected networks to meet tasks of different difficulty in a balanced manner.

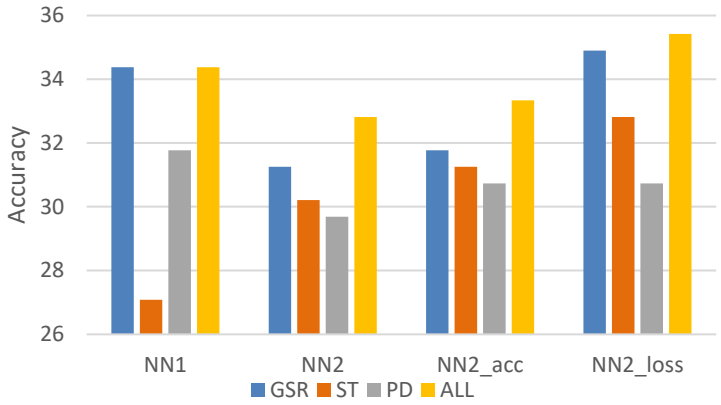
Besides, accuracy of *NN2_acc* and *NN2_loss* is generally higher than that of *NN2*, and more balanced in the performance of all four kind of features than *NN1*. This shows that networks optimized by GA could not only provide relatively better results, but also be robust to tasks of different difficulty, which bring a more balanced result.

In addition, the accuracy of *NN2_loss* is higher than *NN2_acc*. Both of the structures of these two networks had been optimized by GA, but used different fitness functions. This shows F_{loss} has a better evaluation ability of the quality of the network structure than F_{acc}. This may be due to the accuracy changes are not continuous, while loss changes are continuous, which means F_{loss} is more sensitive to network structure changes. For example, if predict value for a certain class of a model is gradually approaching ground truth, then we could say the model is actually developing in a good direction. In this process, its loss would definitely decrease, and F_{loss} would increase. However, the classification accuracy may not change, since the predict value is not close enough to ground truth. This means that the fitness of better chromosomes does not changed, and the population needs much more attempts to improve a little F_{acc} than F_{loss}, that is, it is more difficult for the population to find the evolutionary direction.

Table 3. Accuracy of models trained by NNs with different topologies on different features

Models	GSR	ST	PD	ALL
NN1	34.375	27.083	**31.771**	34.375
NN2	31.250	30.208	29.688	32.813
NN2_acc	31.771	31.250	30.729	33.333
NN2_loss	**34.896**	**32.813**	30.729	**35.417**

Fig. 3. Accuracy of models trained by NNs with different topologies on different features



5 Conclusion and Future Work

Our work shows that genetic algorithm can be used to automatically optimize the structure of neural network according to task requirements. This can save a lot of time for people to try different network topologies, and perform better than the topological structure estimated by people based on experience. Traditional fully connected networks may contain redundant connections, affecting the performance of the model, or causing overfitting. Genetic algorithm is equivalent to an intelligent automatic pruning the network. Our experiments have proved that this method effectively improved the classification effect based on neural network in terms of model stability, convergence speed, and classification accuracy. Besides, we found that the fitness function of GA should be sensitive to chromosome changes. Otherwise, it will be difficult for the population to find an evolutionary direction.

For future work, more research could be used to explore the impact of GA on other hyperparameters in NN, such as learning rate, activation function, optimizer parameters, and so on. Also, the influence of GA parameters on the optimization effect of NN structure is also worth studying, such as population size, number of evolutionary generations, selection, crossover and mutation type, crossover and mutation rate, and so on. In addition, in this paper, we only discussed the case of no cross-layer connection, while cross-layer connections could provide more flexible network structures. This can also be a potential research direction.

References

1. X. Zhu, T. Gedeon, S. Caldwell and R. Jones, “Detecting emotional reactions to videos of depression”. In 23rd International Conference on Intelligent Engineering Systems (INES), pp. 000147-000152. (2019)
2. N. Cummins, S. Scherer, J. Krajewski, S. Schnieder, J. Epps, and T. F. Quatieri, “A review of depression and suicide risk assessment using speech analysis”. Speech Commun. vol. 71, pp. 10-49. (2015)
3. K. Hawton, C. C. i Comabella, C. Haw, and K. Saunders, “Risk factors for suicide in individuals with depression: a systematic review,” J.Affect. Disord. vol. 147, no. 1-3, pp. 17-28. (2013)
4. T. Shinozaki and S. Watanabe, “Structure discovery of deep neural network based on evolutionary algorithms”. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4979-4983. (2015)
5. Whitley, D. “A genetic algorithm tutorial”. Statistics and Computing. Vol. 4, pp. 65-85. (1994)
6. S. Balakrishnama and A. Ganapathiraju. “Linear discriminant analysis-a brief tutorial”. Inst. Signal Inf. Process. Vol. 18, pp. 1–8. (1998)
7. Miller, B. and D. Goldberg. “Genetic Algorithms, Tournament Selection, and the Effects of Noise”. Complex Syst. 9. (1995)
8. Wen-Yang Lin, Wen-Yung Lee and Tzung-Pei Hong. “Adaptive Crossover Rate and Mutation Rate in Genetic Algorithms”. Journal of Information Science and Engineering, Vol. 19 No. 5, pp. 889-903. (2003)