Noname manuscript No.
(will be inserted by the editor)

Automation Design of Artificial Neural Network by Genetic Algorithm

Zhang Huiyao[1] · Atsushi Yokoyama [1,*]

Received: date / Accepted: date

Abstract Based on Ockham's razor, the more complexity your model is, the greater the possibility for errors are. to keep the model as simple as possible. In this paper, genetic algorith method is used to search the simplest neural network strucutre for a specific problem. The experiment result has shown that a simpler neural network can always be found, and the neural network can also be used to compress original data.

Keywords Genetic Algorithm · Neural Network Design · Automation

1 Introduction

The structure of a neuron, as shown in Figure.1, is pretty simple and intuitive, it is consists of dendrites, synapse, cell body, and axon. The number of neurons in human body is about 10^{11}, based on the repeat of this simple neuron pattern, a human being can do reasoning, learning, memorizing, and visualizing. To mimick the behavior of neuron, many neural network have been designed. Hammming network is used to solve binary pattern recognition problem. Hopfield[1] network is used to solve memorizing problem. Grossberg Network[2] is used to solve vision recognition problem. So many amazing neural networks have been developed, however, there isn't a systematic approach to guide the design of neural network. Murata[3] proposed pruning method to decide whether or not should include a neuron. Some researchers[4] use growing methods to build the neural network.

In this paper, global search method is used to find the simplest neural network that can explains the data.

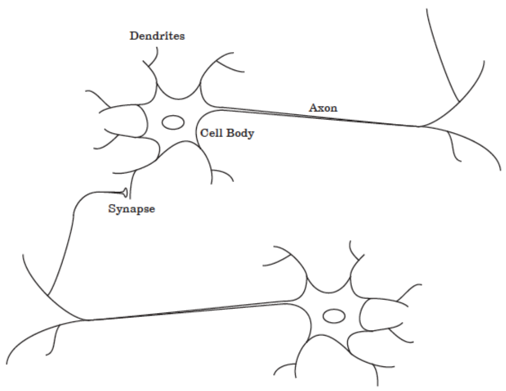


Figure 1: Schematic Drawing of Biological Neurons

Zhang Huiyao
Room 203,Bulding 3,Kyoto Institue of Technology
Matsugasaki,Sakyo-ku,Kyoto,606-8585,JAPAN
E-mail: zhanghy1012@gmail.com

S. Author
second address

Genetic algorithms (GAs) are first introduced in 1975 by Holland[5] because of its powerful search ability, which was widely been used to solve many search, optimiation, and classification problem. Compared with other random search algorithm, for example, particle swarm optimization, ant colony optimization, and simulated annealing algorithm, GAs are not easily trapped in local optima, and obtain the global optimal.

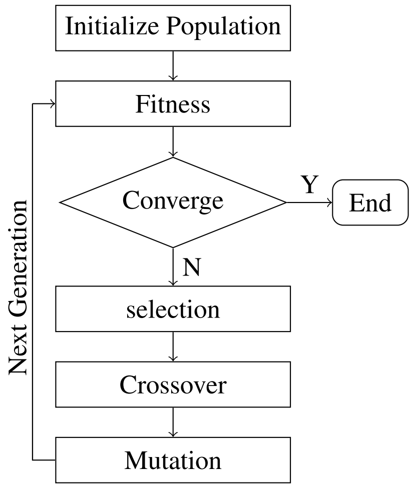


Figure 2: Genetic Algorithm

2 Problem-oriented Neural Network Model

For a specific engineering problem, the inputs and outputs of the neural network are fixed. The design problem unsolved is the number of hidden layers , the number of nodes in each hidden layer, and the transfer functions. Accordint to Ockham's razor, the simple model are more likely to be the right one, so in this paper, one principal is to keep the neural network as simple as possible. The more number of hidden layers, the more complexity the model is. Actually, a two-layer network can approximate any practical function, so there is only one hidden layer in the neural network model.

According to Karlik's[6] research, the performance of neural network not only based one the architecture, but also related to the transfer fucntions. Mostly used transfer functions such as sigmoid, relu, tanh and softmax function are discussed in this paper. In order to discuss transfer function in genetic algorithm, different bianry codes are used to indicate transfer function. '00' stands for sigmoid function, '01' for relu function, '10' for tanh function, and '11' for softmax function, as shown in Table 1

Table 1: Transfer Function

Name	Equation	Binary code
sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	00
relu	$f(x) = \max\{0, x\}$	01
tanh	$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	10
softmax	$f(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{K} e^{x_i}}$	11

Each neuron in hidden layer can be treated as a feature extractor, which is randomly connected with the inputs. As the output of the neural network fully depend on the features that the neural network learnt, the output layer and the hidden layer should be full connected. So the potential neural network model is as shown in Figure.3

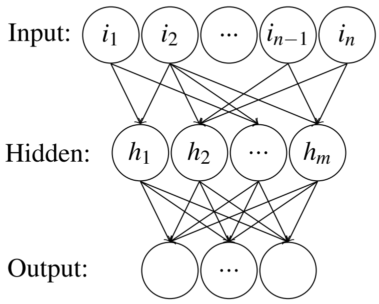


Figure 3: Neural Network Model

As shown in Figure.3
1. Neurons in the output layer are full connected with the neurons in the hiddern layer.
2. Every neuron in the hiddern layer is partially connected with the input, for example b_1 only connects with a_1, a_2
3. The number of neurons m is randomly initialized and m is greater then 1 and less then n

3 Genetic Algorithm Procedure

3.1 Population Initialization

As showned in figure.3, neurons in hidden layer are randomly connected with the inputs. If there is a connection between previous neuron and current neuron, it is indicated by a constant numberic value 1, otherwise the number is 0 which means no connection. This process can be divided into two steps:

1. random generate the number of nodes in the hidden layer, denoted by m.
2. generate the chromosome, the length is $m \times l$, l denotes the lenght of each locus.

For example, assuming the random number m is 4 and the corresponding chromosome is 110100 011101 010101 010110. The chromosome can be divided into four parts, each part corresponds to the locus of one node in the hidden layer; each part can be divided into two subparts, the length of each locus is 6, the beginning four parts indicate the connection, the last 2 bits stand for transfer function. So the architecture of this neural network is as shown in Figure 4(a).

In Figure 4(b), there are two nodes in the hiddern layer, so the beginning four parts of locus of neuron h_1 and h_2 are 1101, 0011.

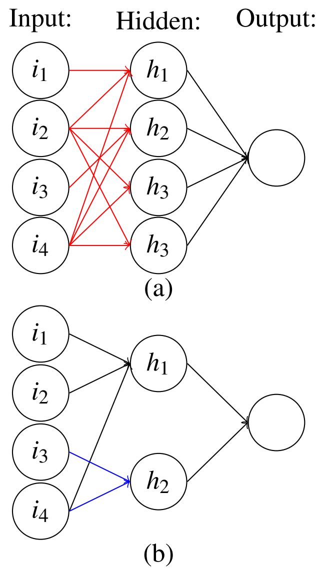


Figure 4: Neural Network Example

3.2 Selection

Ranking selection is taken to choose the parents of next generation from high to low according to their fitness. The individual with best fitness are choosen as parents, repeat the process until you get the desired amount of population.

The input data is splitted into two parts, training data and evaluation data, traing data is used to adjust the parameters in the neural network, and evaluation data is used to measure the performance of the neural network. The performance of the neural network on evaluation data is treated as the fitness of the neural network, as shown in Figure 5.

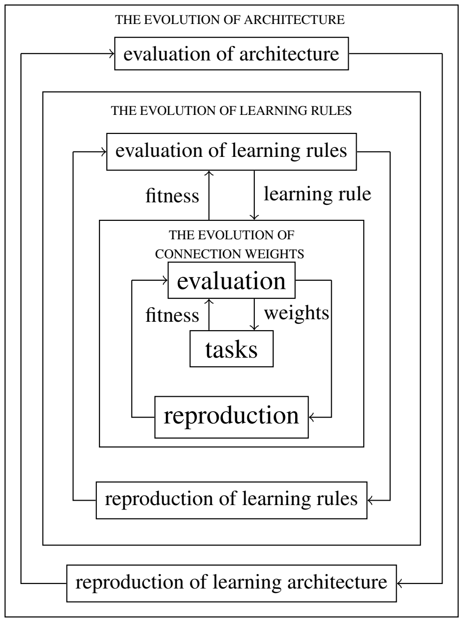
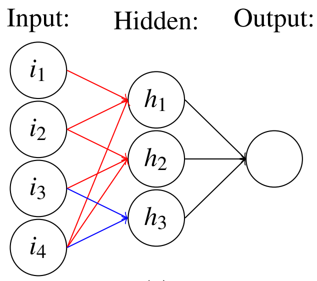


Figure 5: Genetic Algorithm and Neural Network

3.3 Crossover

The crossover is a little bit different from traditional genetic algorithm, because the length of chromosomes which are chosen as parents maybe not equal. So the following rule is used for crossover operation, take half number of loci in each chromosome, and combine these loci into a new chromosome
For example, networks as shown in Figure 4(a) and Figure 4(b) , the number of loci in the hidden layer in each network are 4 and 2, so the number of loci of their offspring is 3. At the same time, the offspring also inherit connection relationship

and transfer function from their parents. As shown in Figure 6, the red color connection inheirts from network in Figure 4(a) and the blue connection inherits from network in Figure 4(b)



(a)

Figure 6: Offspring

4 Experiment and Results

Heart disease date set is used to check the performance of this neural network design method. This dataset was obtained from UCI machine learning benchmark repository

4.1 Data Set

The data set is used to explore the relationship between the symptoms of a patient and the presence of heart disease. There are 75 attributes for each patient, 13 attributes are used as the inputs of the neural network, and 1 is as the output of neural network. The number of the database is 303, 240 of them are used as the train data, the remaining is used as the evaluation data.

4.2 Experiment

Because data set has 13 attributes as inputs, so each locus has 15 bits. The beginning 13bits represent the connection, the last two bits stand for transfer function, relevant parameters as shown in Table 2

Parameter	Value
population	10
locus length	15
encoding method	binary
crossover strategy	one-point
mutation strategy	random

Table 2: Genetic Parameters

4.3 Results

The top four neural networks ranked by accurate, as shown in Table 3

Table 3: Result

Network Name	Number of Hidden Node	Transfer Function	Accuracy
n_1	4	R[1]F[2]FF	0.817
n_2	4	T[3]S[4]RF	0.817
n_3	3	TSR	0.800
n_4	6	FFTRFF	0.800

[1] R denotes relu function, [2] F denotes softmax function, [3] T denotes tanh function, [4] S denotes sigmoid function.

Table 4: Chromosome n_1

locus	Binary Presentation
l_1	0 0 1 0 1 0 1 1 0 0 0 0 1 1 0
l_2	0 1 0 1 0 1 1 0 0 0 0 1 1 0 0
l_3	1 0 1 0 1 1 0 0 0 0 1 1 0 0 1

As shown in Table 4, the two columns which are highlighted are all zero, it means no neuron in the hidden layer is connected with these two inputs.

5 Conclusion

Through automation design of neural network, a simpler architecture can always be found which can explain the data well. According to the final neural network be obtained, it can be used to reduce the dimensionality of the input, namely, data compression.

References

1. John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
2. Stephen Grossberg. Adaptive pattern classification and universal recoding: Ii. feedback, expectation, olfaction, illusions. *Biological cybernetics*, 23(4):187–202, 1976.
3. Noboru Murata, Shuji Yoshizawa, and Shunichi Amari. Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE transactions on neural networks*, 5(6):865–872, 1994.
4. Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. *Neural network design*. Martin Hagan, 2014.
5. David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.
6. Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.