A Framework for an Automatic Generation of Neural Networks

Belal Al-Khateeb[1], and Maha Mahmood[2]

**[1] College of Computer, Al-Anbar University, Iraq
Ramadi.**

**[2] College of Computer, Al-Anbar University, Iraq
Ramadi.**

Abstract

The automatic generation of neural network architecture is a useful concept as in many applications while the optimal architecture is not a priori known. Often trial and error is done before a satisfactory architecture is found. Construction deconstruction algorithms can be used as an approach but they have several drawbacks. Sometimes an evolutionary computation and evolutionary algorithms are used but the idea in this paper is reserved for a special kind of evolutionary algorithms. So in this paper we proposed framework for neural networks which try to get best solution for problems by automatic generation technique. The obtained results are promising, suggesting many other research directions..

Keywords: *neural network, evolutionary algorithms, genetic programming, genetic algorithms.*

1. Introduction

Neural network performance is depending on its architecture and on a given task, includes properties like learning speed and generalization capability. On our world most of the application of evolutionary computation in the generations of neural network architecture has a significant influence on the performance of the network. It is the usual practice to use trial and error to find suitable neural network architecture for a given problem. This method is not only time consuming but may not generate an optimal network. Therefore, the use of evolutionary computation is a step towards automation in neural network architecture generation. This paper seek preferable solution for any problem by random operations on the neural network architecture, those operations are add layer, add node, delete layer, delete node, keep the architecture with no change and update weights [1].

2. Background

Work on artificial neural networks, commonly referred to as neural networks, has been motivated right from its inception by the recognition that the brain computes in an entirely different way from the conventional digital computer. In the human brain, many neurons are arranged together as a complex network. A single neuron acts as valve to change the flow of information propagating through the NN. Neurons will strengthen some signals and limit others in order to produce the final resulting output. In a brain, neurons collect inputs like water flowing into a dam. When the water attains a certain predefined level, it empties everything into its outputs and starts over again [2][3]. There may be many different Neural Networks (NN) for solving every particular problem and network designers usually face questions such as “How could we find the size of a NN?”, “Is the selected architecture an appropriate one?”, “How could one design an optimal network?”. These questions usually lead the designer to optimization methods to find the desired network. In order to avoid the local minima encountered in most of the optimization methods, scientists tend to use random search methods such as evolutionary algorithms (EA) and Genetic algorithms (GA) to find an optimal network [7]. In 1997, Vonk et. al. [1] used automatic generation of a neural network architecture using evolutionary computation. Koza and Rice [3] used genetic generation of both the weights and architecture for a neural network shows how to find both the weights and architecture for a neural network (including the number of layers, the number of processing elements per layer, and the connectivity between processing elements).In 2007 Fiszelew [5] used automatic generation of neural networks based on genetic algorithms

for finding optimal neural network architectures to learn particular problems. In 2009 Nadi [7] used evolution of neural network architecture and weights using mutation based genetic algorithm. In this paper we present a new approach for evolving optimized neural network architecture for a three to five layer feedforward neural network with a mutation based genetic algorithm.

3. Neural networks and Evolutionary Algorithms

Neural networks are widely used in applications such as pattern recognition, classification, clustering, prediction, and so on. These networks are trained using the application data. The generalization capability in these networks directly depends on the training, architecture, the number of layers and the number of neurons in each layer. If the number of neurons in the network is increased, the network attracts to over fit the training set, and thus the interpolation capability will be decreased. On the other hand if the number of neurons is less than the necessary number, the network cannot learn all the data. Therefore, for every application, there are a particular number of neurons which keep the best interpolation generalization balance. The designer needs some method for finding the appropriate choice for keeping this balance [4]. EAs are some kind of random search algorithms which use natural evolution to solve optimization problems. EAs include different categories such as genetic algorithm, evolutionary strategy, evolutionary programming, and genetic programming. Although there are some differences between these methods, but they have a lot of common traits. An EA is applied to a population which is a presentation of the optimization problem. The representation of problem could be as simple as a series of 0's and 1's or as complicated as a computer program. The initial population could be defined completely random or based on prior knowledge. This algorithm will evaluate the population based on a goal function and specify how much each agent is close to the goal of problem. The goal function is different for each individual problem and should be defined by the user. There are several methods to produce the next generation of the solutions from current population. In one formal method, agents with better fitness will be selected as the parents for the next generation. There are several operators which are applied to chromosomes to produce the next generation. They are called Genetic operators. The most important operators are mutation, crossover and combination. The new population is produced and its fitness will be computed. This loop continues until a desired answer or the maximum epoch assigned is reached [1][4].

4. The Proposed Framework

The network in the framework has an input layer with any number of neurons that can be specified by the user, one to three number of hidden layers with three neurons as minimum and twenty neurons as maximum, finally the network has an output layer with one neuron. The input layer receives a vector values (x1…….xp) and distributes those values to each of the neurons in the hidden layer, also there special neuron called the bias that is used to control the network and if fed to each of the hidden layers, the basis is multiplied by a weight and added to the sum going into the neuron. To get the output for the neurons in the hidden layer, the value from each input neuron is multiplied by a weight (wij), and the resulting weighted values are added together producing a combined value uj. The weighted sum (uj) is fed into transfer function which outputs a value hj, the outputs from the hidden layer are distributed to the next layer. Also to get the output of the neurons in the output layer, the value from each hidden layer neuron is multiplied by the weight (wki), and the resulting weighted values are added together producing a combined value vi that is fed into a transfer function, which outputs value (y) that is the value of output layer of the network.The following algorithm is proposed to get the neural network architecture automatically:

1. A population of a given number (n) of neural networks was initialized at random. All of the weights and biases of each network were initialized uniformly over [-0.5, 0.5].
2. Each strategy has an associated self-adaptive parameter vector s_i, i=1,…,n initialized to 0.05.

3. For each network do the following:-
   - With 1/3 probability choose the add operation then go to step 4.
   - With 1/3 probability choose the delete operation then go to step 5.
   - With 1/3 probability keep the architecture as it is.
4. - With 1/2 probability choose a hidden layer to be added to the neural network at a random position, such that the total number of the hidden layers is not greater than the maximum number of hidden layers in the network. If this is not the case then do not add the new hidden layer. Then go to step 6.
   - With 1/2 probability randomly choose the hidden layer in order to add a number of nodes to it, such that the total number of the nodes is not greater than the maximum number of nodes in the hidden layer. If this is not the case then add the number of nodes that makes the total number of nodes equal the max number of nodes in the hidden layer. Then go to step 6.

5. - With 1/2 probability choose a hidden layer to be deleted from the neural network at a random position, such that the total number of the hidden layers is not less than the minimum number of hidden layers in the network. If this is not the case then do not delete the hidden layer.
   - With 1/2 probability randomly choose the hidden layer in order to delete a number of nodes from it, such that the total number of the nodes is not less than the minimum number of nodes in the hidden layer. If this is not the case then delete the number of nodes that makes the total number of nodes equal the min number of nodes in the hidden layer.
6. Calculate the output (fitness) of each network.
7. Select the best n/2 networks that have the highest scores as parents and retained for the next generation. Those parents are then mutated to create another n/2 offspring using the following equations:

$$s_i(j) = s_i(j)\exp(\, tN_j(0,1)\,),\ j = 1, \ldots, N_w$$
$$w_i(j) = w_i(j) + s_i(j)N_i(0,1),\ j = 1, \ldots, N_w$$

where N_w is the number of weights and biases in the neural network, $t = \frac{1}{\sqrt{2\times\sqrt{N_w}}}$, and $N_j(0,1)$ is a standard Gaussian random variable resembled for every j.

8. Repeat steps 3 to 7 for k generations.

5. Results

In this section we present the initial results of our proposed framework that is used to select the best architecture of the multi-layer perceptron neural network. Tables 1 through 3 show sample results of the algorithm.

Table 1: Add operation (add new nodes and hidden layer)

Initialization …… Input Nodes = : 15 Input Value = { 0 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , } ---------- : ---------- Layers Count= 1 : Layer = : 1 Nodes = : 6 weights = : 15 ---------- : ---------- Procedure : Add New Nodes 18 : Nodes have been added ..	Output = : 0.598036690303204 Old Output = : 0 ---------- : ---------- Layers Count= 1 : Layer = : 1 Nodes = : 24 weights = : 15 ---------- : ---------- Procedure : Add New Hidden Layer in position 0 Layers after adding = : 2 Output = : 0.638789646760108 Old Output = : 0.598036690303204

Table 2: No change operation

Layers Count= 2 : Layer = : 1 Nodes = : 3 weights = : 15 Layer = : 2	Nodes = : 24 weights = : 3 ---------- : ---------- Procedure: No Change.. Output = : 0.76474504467588 Old Output = : 0.638789646760108

Table 3: Delete operation (layer and nodes) and adding new hidden layer

Layers Count= 2 : Layer = : 1 Nodes = : 3 weights = : 15 Layer = : 2 Nodes = : 24 weights = : 3 ---------- : ---------- Procedure : Delete Layer R = 0 Layers after deleting = : 1 Output = : 0.84588784258043 Old Output = : 0.76474504467588 ---------- : ---------- Layers Count= 1 : Layer = : 1 Nodes = : 24 weights = : 15 ---------- : ---------- Procedure : Add New Hidden Layer in position 0 Layers after adding = : 2	Output = : 0.860533295142764 Old Output = : 0.84588784258043 ---------- : ---------- Layers Count= 2 : Layer = : 1 Nodes = : 7 weights = : 15 Layer = : 2 Nodes = : 24 weights = : 7 ---------- : ---------- Procedure : Delete Nodes 3: Nodes have been deleted... R = 0 : R + 1 = 1 R = 0 : R + 1 = 1 R = 0 : R + 1 = 1 Output = : 0.957570492384686 Old Output = : 0.860533295142764

The results in the above tables show all the proposed cases for the addition and deletion operations, in all the tables one can easily notice that the output values are enhanced over the old output, which can be considered as a success for the proposed framework.

6. Conclusions and future work

This paper have shown the designing of an automatic generation of neural networks that performs random operation within hidden layers (generating layer, generating nodes and connections; deleting layer, deleting nodes and connections) in order to get best possible architecture for finding the best solutions. Although the obtained results are initial but they were promising as shown in tables 1 through 3 as the tables showed all the applied cases for the addition and deletion operations with an output enhancing. The work needs some additional tests therefore the following can be done as a future work: applying the framework to a real problems such as computer games (like tic–tac-toe, checkers and chess) and pattern recognitions.

References

[1] E. Vonk, L.C. Jain, L.P.J. Veelenturf and R. Johnson'' Automatic Generation of a Neural Network Architecture Using Evolutionary Computation'' 1997.

[2] Koza, John R., Genetic Programming, On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, 1992.

[3] Koza J. R. and Rice, J. P I "Genetic Generation of both the Weights and Archilecture for a Neural Network", IEEE lnrernnrional Joinr Conference on Neural NerrtorXs, 1991.

[4] Freeman J.A., Skapura D.M.: Neural networks - Algorithms, applications, and programming techniques, Addison-Wesley, Reading, MA 1991

[5] Fiszelew, A., Britos, P., Ochoa, A., Merlino, H., Fernández, E., García-Martínez, R. '' Finding Optimal Neural Network Architecture Using Genetic Algorithms'' *Research in Computing Science*, 2007.

[7] A. Nadi, S. S. Tayarani-Bathaie and R. Safabakhsh ''Evolution of Neural Network Architecture and Weights Using Mutation Based Genetic Algorithm'' Proceedings of the 14th International CSI Computer Conference (CSICC'09), 2009.

[8] Alejandro Correa, Banco Colpatria, used Genetic Algorithm Optimizatin for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network, 149-2011.

First Author Belal Al-Khateeb received the B.Sc. (honors) (first class) degree in computer science from Al-Nahrain Universi-ty, Baghdad, IRAQ, in 2000, and the M.Sc. degree in com-puter science from Al-Nahrain University, Baghdad, IRAQ, in 2003, and the Ph.D. degree from the School of Computer Science, University of Nottingham, Nottingham, U.K., in 2011. He is currently a lecturer at the College of Computer, Al-Anbar University. He has published over 15 refereed journal and conference papers. His current research interests include evolutionary and adaptive learning particularly in computer games, expert systems, and heuristics and me-ta/hyper-heuristics. He has a particular interest in computer games programming. Dr. Al-Khateeb is a reviewer of two international journals (including one IEEE Transaction) and four conferences .

Second Author Maha Mahmood received the B.Sc. (first class) degree in computer science from Al-Anbar Universi-ty, Ramadi, IRAQ.