Optimize BpNN Using New Breeder Genetic Algorithm

Maytham Alabbas[1], Sardar Jaf[2(✉)], and Abdul-Hussein M. Abdullah[3]

[1] Department of Computer Science, College of CS and IT,
University of Basrah, Basrah, Iraq
ma@alumni.manchester.ac.uk
[2] School of Engineering and Computing Sciences,
Durham University, Durham, UK
sardar.jaf@durham.ac.uk
[3] Department of Computer Science, College of Science,
University of Basrah, Basrah, Iraq
abdo60_2004@yahoo.com

Abstract. In this paper, the ability of genetic algorithms in designing artificial neural network (ANN) is investigated. The multi-layer network (MLN) is taken into account as the ANN structure to be optimized. The idea presented here is to use the genetic algorithms to yield contemporaneously the optimization of: (1) the design of NN architecture in terms of number of hidden layers and of number of neurons in each layer; and (2) the choice of the best parameters (learning rate, momentum term, activation functions, and order of training patterns) for the effective solution of the actual problem to be faced. The back-propagation (BP) algorithm, which is one of the best-known training methods for ANNs, is used. To verify the efficiency of the current scheme, a new version of the breeder genetic algorithm (NBGA) is proposed and used for the automatic synthesis of NN. Finally, several problems of the experiment were taken and the results show that the back-propagation neural network (BpNN) classifier improved the current scheme has higher accuracy of classification and greater gradient of convergence than other classifiers, which have been proposed in the literature.

Keywords: Breeder genetic algorithm · Back-propagation network · Artificial neural network · Multi-layer neural network

1 Introduction

The back-propagation (BP) algorithm, despite having proved useful in a number of problems, still presents a certain range of difficulties as a network structure, convergence, calculation time, and teaching method.

There is neither theoretical result nor even a satisfactory empirical rule suggesting how a network should be dimensioned to solve a particular problem. Should the network use one hidden layer or more? How many neurons should there be on the hidden layer? What is the relationship between the number of training examples and the number of classes to separate these examples into? What should be the overall size of the network?

A.E. Hassanien et al. (eds.), *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, Advances in Intelligent Systems and Computing 533, DOI 10.1007/978-3-319-48308-5_36

Several researchers used guesswork or trial and error to determine the number of layers, number of neurons in each layer, and the connection neurons for a certain problem. Also, many of the network parameters (learning factor, momentum factor… etc.) were determined empirically. This approach needs a long time to obtain good results, so genetic algorithms are used for solving these difficulties.

The rest of this paper is organized as follows: Sect. 2 will describe the breeder genetic algorithm and a brief explanation of the neural network will be given in Sect. 3. Section 4 explains the current scheme and in Sect. 5 results of our experiments are compared with those obtained by other researchers' results for the same problems through computer simulations. Finally, Sect. 6 presents the conclusion of this paper.

2 Breeder Genetic Algorithm

The breeder genetic algorithm (BGA) was designed by Heinz Mühlenbein in Germany at the beginning of the 1990s. It lies somehow in between the genetic algorithms (GA) and evolution strategies (ESs), in the sense that they borrow from each of them some basic ideas. The basic scheme for BGA is illustrated in Fig. 1 [1].

```
Procedure BGA;
 Begin
   t=0;
   Initialize randomly P(t) with N individuals;
   While (termination criterion not fulfilled) do
     Evaluate goodness of each individual;
     Save the best individual in the new population;
     Select the best T% individuals;
     For I= 1 to N-1 do
       Select randomly two elements within the best T% in P(t);
       Recombine them so as to obtain one offspring;
       Perform mutation on the offspring;
       Insert it in P'(t);
     End For;
     P(t+1)=P'(t);
     t=t+1;
     Update variable for termination;
    End while;
End;
```

Fig. 1. Breeder genetic algorithm (BGA).

3 Back-Propagation Neural Network

Neural Networks (NNs) are algorithms for optimization and learning based loosely on concepts inspired by researches conducted on the nature of the brain [2]. An NN is simply a set of interconnected individual computation elements called neurons. In the

case of the multi-layer neural networks (MLNs), the neurons are arranged in a series of layers. A layer is usually a group of neurons, each of which is connected to all neurons in the adjacent layer [3].

The back-propagation neural network (BpNN) is MLN. During the learning phase; input patterns are presented to the network in some sequence. Each training pattern is propagated forward layer by layer until an output pattern is computed. The computed output is then compared to the desired one and an error value is determined. The errors are used as input to feedback connections from which the adjustment is made to the synaptic weights layer by layer in a backward direction. The backward linkages are used only for the learning phase, whereas the forward connections are used for both the learning and the operational phases [4].

4 The Current Approach

The basic idea is to provide an automatic technique to define the most appropriate NN structure for a given problem. The optimization of MLN, which must be trained to solve a problem, is characterized by the need to determine: the architecture (the number of hidden layers (*NHL*) and number of neurons (HN_i) for each layer i), the activation function (Lf_i) to be used for each layer i, the momentum term (α), the initial temperature (t_0), the initial learning term (η_0), the bias cell (b), and the order of training patterns that are being presented during training (*PatOrd*). Choosing a suitable initial value for these parameters is a fundamental decision faced by all NN's users, but it is a problem. Often, the choice of these parameters can have a significant impact on the effectiveness of the NN. The choice needs to be tuned for efficiency.

New breeder genetic algorithm (NBGA) has been used to determine the appropriate set of parameters listed above. NBGA is an updated version of BGA (see Fig. 2) which is characterized by the following properties compared with BGA:

1. Its ability to determine the population size depending on the value of T, which represents the best individuals which are selected.
2. It has an efficient selection method, which prevents the repetition of the selected parents, and this implies to increase the diversity of the population.
3. It is capable of extending the population size by the Eq. 1. to move the best k individuals to the next generation instead of the best individual only.

$$N = \frac{T(T-1)}{2} + k, 1 \leq k \leq T, \tag{1}$$

4. Its ability to manipulate chromosomes with different length, which ensures solving different problems.

- Encoding

The NN is defined by "genetic encoding" in which the genotype is the encoding of the different characteristics of MLN and the phenotype is the MLN itself.

```
Procedure NBGA;
  Begin
    t=0;
    Input T & k; // where 1≤k≤T
      N = T(T−1)/2 + k
    Initialize randomly P(t) with N individuals;
    While (termination P(t) criterion not fulfilled) do
      Evaluate goodness of each individual;
      Save the best k individual(s) in the new population;
      Select the best T individuals;
      For i=1 to T-1 do
        For j=i+1 to T do
          Recombine chromosome i & j in P(t) to obtain one offspring;
          Perform mutation on the offspring;
          Insert it in P' (t);
        End For;
      End For;
      P(t+1)=P' (t)
      t=t+1;
      Update variable for termination;
    End While;
  End
```

Fig. 2. New breeder genetic algorithm (NBGA).

The NBGA considered the chromosome structure C = {G_1,…,G_8} as reported in Fig. 3.

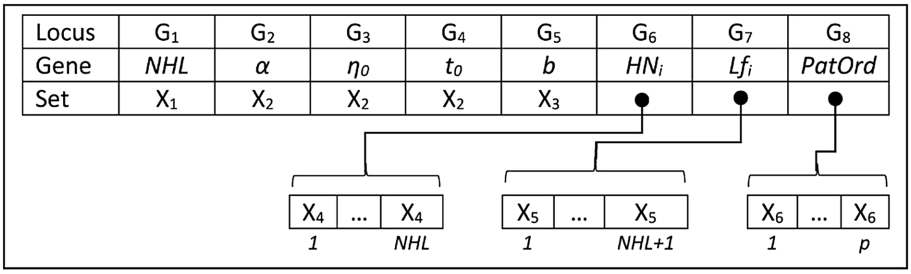

Fig. 3. The hierarchical chromosome representing an NN.

The loci are defined within the following subsets:
$X_1 \in \{1, \ldots, 4\}$
$X_2 \in (0, 1)$
$X_3 \in \{0 = \text{without bias}, 1 = \text{with bias}\}$
$X_4 \in \{1, .., M\}$, where M: maximum number of neurons.
$X_5 \in \{1 \equiv f_1, 2 \equiv f_2, 3 \equiv f_3, 4 \equiv f_4\}$, where:

Sigmoid function (f_1):

$$f_1(x) = \frac{1}{1+e^{-kx}}, k > 0 \tag{2}$$

Tanh function (f_2):

$$f_2(x) = \frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}} \tag{3}$$

Hyperbolic Tanh function (f_3):

$$f_3(x) = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}} \tag{4}$$

Semi-linear function (f_4):

$$f_4(x) = \begin{cases} 1 & if\ x > 1 \\ x & if\ -1 \le x \le 1 \\ -1 & if x < -1 \end{cases} \tag{5}$$

X_6 = the set {1, .., p} in permutation form, where p: number of training patterns.

- The fitness function

To evaluate the goodness of an individual, the network is trained with a fixed number of patterns and then evaluated according to Eq. 6 to determine parameters. The following function is used:

$$f = k_1 E + k_2 \frac{ep}{epmx} + k_3 \sum_{i=1}^{NHL} \frac{HN_i}{M} \tag{6}$$

Where,
E: (Mean Square Error) the error result from BpNN,
ep: current epoch,
$epmx$: maximum epochs,
NHL: number of hidden layers,
HN_i: number of neurons in hidden layer i,
M: maximum neuron,
$k_1, k_2, k_3 \in (0, 1)$ and $k_1 + k_2 + k_3 = 1$.

- Crossover

The uniform crossover (UX) [5] is used on the genes (NHL, α, η_0, t_0) because of its ability to yield an offspring more different from his parents and this implies to population variety. Depending on the value of NHL, which is inherited from the offspring, the UX used for NH and Lf genes in the case where NHL of parents is different. There are two cases, as shown in Fig. 4.

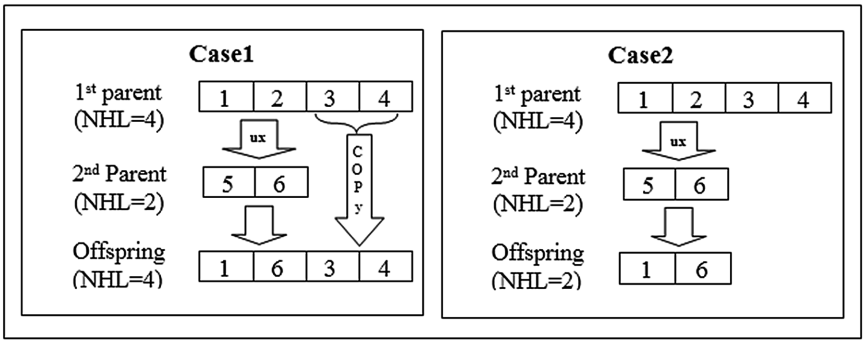

Fig. 4. UX for different chromosome length.

For the last part (*PatOrd*), multi-cycle crossover (MCX) is used because of its capability to prevent the repetition of a certain value for more than one gene. MCX is an updated version of cycle crossover (CX) [6] that builds offspring from ordered individuals by identifying cycles between two parents. To form the offspring, the cycles are copied from the respective parents. The basic scheme for MCX is explained in Fig. 5.

```
Procedure MCX(P1,P2);
  Begin
     i=1;
     j=2;
     C1 & C2 are empty chromosomes
     While (child chromosome has empty position) do
        x=random position from empty positions in the child
        While (C1(x) is empty) do
           C1(x) = Pi(x)
           C2(x) = Pj(x)
           x = the position of the gene Pj(x) in Pi(x)
        End while;
        Swap(i,j)
     End while;
Return C1&C2;
```

Fig. 5. Multi-cycle crossover (MCX).

A chromosome structure (genotype) for instance can look like the chromosome in Fig. 6.

- Mutation

A certain value is added (or deleted) from the genes (α, η_0, t_0) in certain probability in order to keep their values in the range (0, 1). If one of these values exceeds the range, the operations repeated until it becomes at the proper value. In addition, the mutation

NHL	α	η_0	t_0	*b*	HN_1	HN_2	Lf_1	Lf_2	Lf_3	*PatOrd*			
2	0.5	0.9	0.7	0	4	4	1	3	1	2	3	1	4

Fig. 6. A chromosome structure (genotype) for the current technique.

operator 1 m [5] is used for the genes (*Lf*, *HN*) because each of (*Lf*, *HN*) has minimum and maximum value. For the part (*PatOrd*), the mutation operator 2 m [5] is used to prevent presenting training pattern more than once to the network. The gene (*NHL*) is left unchanged, because any change on this gene will change the value of *Lf* and *HN*.

5 Results

In order to check the effectiveness of the current technique, the following problems were chosen because they allow us to compare our results with the previously published results. The tables below show the strength of the current scheme compared with the previous works. The following parameters were selected for each test:

- Maximum number of hidden neurons (*M*) = 5,
- Number of maximum learning trails (*epmx*) = 500,
- Number of genetic trails = 2000,
- Accepted error = 0.025,
- A number of best chromosomes move to the new population (*k*) = 1.

5.1 Artificial Problem

- Bit parity Problem

Input cell = 6, output cell = 1, T = 7, number of patterns = 64. The result is shown in Table 1.

Table 1. The result of the current scheme compared with other works for 6 bit-Parity problem

Method	Purpose	Pop. size	No. of Gen. Trails	Structure (# connections)	No. of Learning Trail	Error
Yu [7]	Train	*Not Reported*	*Not Reported*	6-8-5-1 (49)	4000	0.075
Najim [8]	Design	100	300	6-7-1 (39)	4000	0.526
Al-Fadhly [9]	Design	25	215	6-7-1 (49)	319	0.053
Current Scheme	Train Design	22	250	6-6-1 (49)	1823	0.023

The parameters are: $\alpha = 0.6$, $\eta_0 = 0.429$, $t_0 = 0.29$, $b = 1$. Activation function: Tanh, Sigmoid. Pattern order: 1, 2, 3, 55, 51, 26, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 6, 32, 20, 21, 22, 23, 24, 25, 18, 27, 38, 29, 30, 31, 19, 48, 34, 35, 36, 37, 28, 39, 40, 45, 42, 43, 44, 41, 46, 47, 33, 49, 50, 5, 52, 53, 58, 4, 56, 57, 54, 59, 61, 62, 63, 64.

5.2 Realistic Problem

The artificial problem presented in Sect. 5.1 is not large enough to challenge the effectiveness of the current scheme. A number of realistic problems that consist of a real world data are used for this purpose in this section. The results of two of these problems are stated below.

- Breast Cancer

Diagnosis of breast cancer: try to classify a tumor as either benign or malignant based on cell descriptions gathered by microscope examination. Input attributes are for instance the clump thickness, the uniformity of cell size and cell shape, the amount of marginal adhesion, and the frequency of bare nuclei.[1]

Input cell = 9, output cell = 2, T = 10, number of patterns = 350. The result is illustrated in Table 2.

Table 2. The result of the current scheme compared with other works for Breast Cancer problem.

Method	Training set	Testing set	Structure	Learning percentage	Generalization percentage
Prechelt [10]	350	349	9-4-2-2	*Not Reported*	94.25 %
Engelbrecht [11]	450	170	9-8-2	97.6 %	95.7 %
Khaudeyer [12]	250	433	9-18-2	100 %	96.3 %
Current Scheme	350	349	9-6-2	100 %	96 %

- Iris Plants

This problem is a common benchmark problem in pattern recognition and classification studies. This dataset contains 150 instances of four attributes (sepal length, sepal width, pental length, pental width) from each of three classes (setosa, versicolor, virinica). The first class is separated from others clearly, while the other two classes overlap slightly. This dataset is publically available from the UCI Repository of Machine Learning Databases and Domain Theories.[2]

[1] This dataset is publically available at ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz.

[2] This dataset is publically available at http://ftp.ics.uci.edu/pub/machine-learning-databases/iris/.

Table 3. The result of the current scheme compared with other works for Iris Plants problem.

Method	Training set	Testing set	Structure	Learning percentage	Generalization percentage
Swain [13]	75	75	4-3-1	*Not Reported*	96.66 %
Weihong [14]	90	60	4-25-3	100 %	96.67 %
Current Scheme	75	75	4-7-3	100 %	97.1 %

Input cell = 4, output cell = 3, T = 10, number of patterns = 150 (50 instances in each of three classes). The result is shown in Table 3.

As it can be seen from the tables above, the current scheme results have been compared with the results of other methods in the literature that provided to solve the same problems. These results show that the current scheme is very efficient and takes less time to solve the problems.

6 Conclusions and Future Work

In this paper, a new improved version of breeder genetic algorithm (BGA), which is called new breeder genetic algorithm (NBGA), was introduced. NBGA was used in designing MLN network to yield contemporaneously the optimization of the design of a neural network architecture in terms of the number of hidden layers and number of neurons in each layer, and the choice of the best parameters (learning rate, momentum term, activation functions order of training patterns, and the order of training patterns) for the effective solution of the actual problem to be addressed. The BpNN algorithm was used as a classifier. NBGA added new useful characteristics to BGA to increase the diversity of the population and manipulate the chromosomes with different length, which ensures solving different problems.

The current scheme was tested and the results were compared with previously published results on solving the same problems. The experimental results presented in this paper have demonstrated the effectiveness of NBGA for BpNN optimization. They have also proved the strength of NBGA in terms of solution quality and speed of conversion.

Based on the findings in this work, two further research tasks have been identified: (i) GA-based ANN model, construction and optimization, is computation intensive and could take quite a long time to process. In order to improve the performance of the presented scheme in terms of execution efficiency, we will work on a low-cost general-purpose graphics-processing unit (GPGPU), specifically, the NVIDIA graphics card, to adopt the ANN model training and validation; and (ii) integrate the artificial bee colony (ABC) algorithm [15] with NBGA algorithm. First, the ABC will be applied to derive an optimal set of initial weights from enhancing the accuracy of ANNs. Then, these weights will be used as the starting points for the NBGA evolution procedure.

References

1. Stoica, F., Boitor, C.: Using the breeder genetic algorithm to optimize a multiple regression analysis model used in prediction of the mesiodistal width of unerupted teeth. Int. J. Comput. Commun. Control **9**, 62–70 (2014)
2. Heaton, J.: Deep Learning and Neural Networks. CreateSpace Independent Publishing Platform (2015)
3. Souza, A., Soares, F.: Neural Network Programming with Java. Packt Publishing Ltd. (2016)
4. Rashid, T.: Make Your Own Neural Network. CreateSpace Independent Publishing Platform (2016)
5. Jacobson, L., Kanber, B.: Genetic Algorithms in Java Basics. Springer, New York (2015)
6. Simon, D.: Evolutionary Optimization Algorithms. Wiley, Berlin (2013)
7. Yu, X.-H., Chen, G.-A.: Efficient backpropagation learning using optimal learning rate and momentum. Neural Netw. **10**, 517–527 (1997)
8. Najim, S., Al-Sharibini, M.: Enhancement neural networks design by general genetic algorithm. Basrah J. Sci. **1**, 46–54 (2003)
9. Al-Fadhly, A.: A study of a neuro-genetic system performance. Department of Computer Science, Ph.D. thesis, University of Basrah (2004)
10. Prechelt, L.: Proben1: a set of neural network benchmark problems and benchmarking rules (1994)
11. Engelbrecht, A., Cloete, I.: Selective learning using sensitivity analysis. In: The 1998 IEEE International Joint Conference on Neural Networks, Proceedings of the 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, USA, vol. 2, pp. 1150–1155. IEEE (1998)
12. Khaudeyer, R.: Hybrid approaches: neuro-fuzzy and geno-neuro-fuzzy hybrid system for solving some classification and functions approximation problems. Department of Computer Science, Ph.D. thesis, University of Basrah (2003)
13. Swain, M., Kumar Dash, S., Dash, S., Mohapatra, A.: An approach for IRIS plant classification using neural network. Int. J. Soft Comput. (IJSC) **3**, 79–89 (2012)
14. Weihong, Z., Shunqing, X.: Optimization of BP neural network classifier using genetic algorithm. Intell. Comput. Evol. Comput. AISC **180**, 599–605 (2013)
15. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**, 21–57 (2014)