Universal Journal of Applied Mathematics 9(2): 17-24, 2021 http://www.hrpub.org
DOI: 10.13189/ujam.2021.090201

A Comparison Between Two Approaches to Optimize Weights of Connections in Artificial Neural Networks

Aydin TEYMOURIFAR

Centro de Estudos em Gestão e Economia (CEGE), Católica Porto Business School, 4169-005, Porto, Portugal
Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), 4200-465, Porto, Portugal

Received May 5, 2021; Revised May 13, 2021; Accepted July 19, 2021

Cite This Paper in the following Citation Styles

***(a)**: [1] Aydin TEYMOURIFAR, "A Comparison Between Two Approaches to Optimize Weights of Connections in Artificial Neural Networks," Universal Journal of Applied Mathematics, Vol.9, No.2, pp. 17-24, 2021. DOI: 10.13189/ujam.2021.090201*

***(b)**: Aydin TEYMOURIFAR, (2021). A Comparison Between Two Approaches to Optimize Weights of Connections in Artificial Neural Networks. Universal Journal of Applied Mathematics, 9(2), 17-24. DOI: 10.13189/ujam.2021.090201*

Abstract Artificial neural networks (ANNs) have been used for estimation in numerous areas. Raising the accuracy of ANNs is always one of the important challenges, which is generally defined as a non-linear optimization problem. The aim of this optimization is to find better values for the weights of the connections and biases in ANN because they seriously affect the efficiency. This study uses two approaches to do such optimization in an ANN. For this aim, we create a feed-forward backpropagation ANN using the functions of MATLAB's deep learning toolbox. To improve its accuracy, in the first approach, we use the Levenberg — Marquardt algorithm (LMA) for training, which is available in MATLAB's deep learning toolbox. In the second approach, we optimize the values of weights and biases of ANN with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), available in MATLAB's global optimization toolbox. Then, we assess the accuracy of estimation for the trained ANNs. In this way, for the first time in the literature, we compare these methods for the optimization of an ANN. The used data sets are also available in MATLAB. Based on the acquired results, in some data sets, training with LMA, and for some others training with PSO cause the best results, however, training with LMA is faster, significantly. Although the used approaches and the obtained conclusions are beneficial for researchers that work in this field, they have some limitations. For instance, since only the functions and data sets from MATLAB are used, it can only serve as an example for researchers.

Keywords Artificial Neural Networks, Genetic Algorithm, Particle Swarm Optimization, MATLAB's Toolboxes

1 Introduction

Artificial neural networks (ANNs) are inspired by the biological structure of animal brains, and have several applications in manufacturing [1-3], process control [4], pattern recognition [5], medical diagnosis [6], and environmental problems [7].

An ANN has components like neurons, connections and propagation function. In this study, we generate a feed-forward backpropagation ANN using MATLAB's deep learning toolbox to do estimations on data sets available in the same toolbox. In ANNs, a weight is assigned to each connection. The values of weights and biases seriously affect the performance of ANN. These values can be defined by solving an optimization model. We improve the performance of ANN in two ways. As the first approach, we use the Levenberg–Marquardt algorithm (LMA) available in MATLAB's deep learning toolbox. In the second approach, we define the values of weights and biases by using Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), which are available in MATLAB's global optimization toolbox.

In the literature, evolutionary algorithms (EAs) have been used to find the weights and biases of ANNs [8-12]. Different from them, in this study, we compare the performances of LMA, GA and PSO available in MATLAB's toolboxes, for the training of ANNs. So the main goal of this study can be summarized like this: we use LMA, GA and PSO to improve the accuracy of a feed-forward backpropagation ANN and make a comparison between their performances.

In other sections of the work, structure of ANN and its training are summarized. The used functions are explained briefly. Then experimental results are presented and compared. Conclusion is the last section of the article.

2 General Structure of ANN, ANN+GA, and ANN+PSO

Each of the used data set is divided into training and test sets, whose inputs and outputs are normalized. The ANN is generated using *newff* function [13]. In different versions of MATLAB, there are similar ones to this function, which can also be used. In the syntax of the *newff* function, there is a matrix in size of $I \times 2$, denoted as *PR*, in which *I* is the number of inputs and shows the minimum and maximum values of the inputs. In all rows, the first and second columns of the *PR* matrix are -1 and 1, respectively. *tansig* is used as the transfer function, which is defined as $f(x) = \frac{1-e^{-x}}{1+e^{-x}}$.

A created ANN is shown in Figure 1.

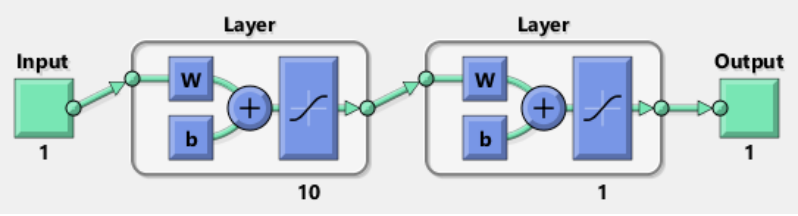

Figure 1. Structure of an ANN with one input, one output, and 10 neurons in the layer. The last layer is the output of the network.

Subsequently, the formed ANN is trained with training data set. During this process, weights and biases are determined. The objective function of the training process is defined as in Equation 1.

$$Minimize\ (MSE(Y_T^{Tr} - Y_P^{Tr})) = Minimize\ \frac{1}{d}\sum_{i=1}^{d}(y_{it}^{Tr} - Y_{ip}^{Tr})^2 \tag{1}$$

where, d, Y_T^{Tr}, and Y_P^{Tr} are the number of data, target outputs, and predicted outputs in the training set, respectively. MSE stands for mean square error.

As mentioned before, two approaches are used for training. In the first one, the *train* function is applied from MATLAB's deep learning toolbox. This process is shown in Figure 2, which is for *vinyl* dataset that has 16 inputs and 1 output [15]. As seen in Figure 2, there are 10 neurons in the layer, LMA is used for training, and performance is calculated based on MSE.

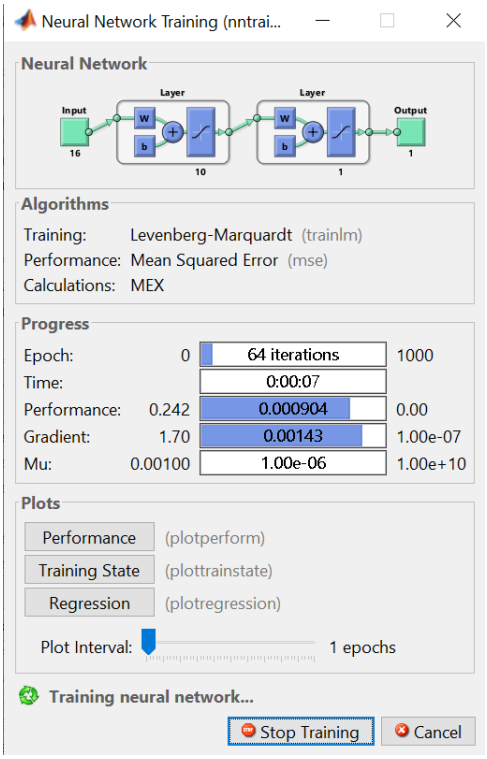

Figure 2. Training of ANN using *train* function from MATLAB's deep learning toolbox [14].

In the second approach, after the ANN is created, the values of its weights and biases are improved with GA and PSO. For this aim, these commands are used: *Network.IW* to access the input weight matrix, *Network.LW* to access layer weight matrix and *Network.b* to access bias vectors [16]. If there are *I* inputs and layer contains *N* neurons, the size of the input weight matrix is $I \times N$. We use only one layer, and the size of its weight matrix is $N \times O$, where *O* is the number of outputs. Each ANN has biases as many as neurons. Hence, in the optimization problem there are $nvars = I \times N + N \times O + N$ decision variables, which are defined as a vector.

In GA, the vector of decision variables is initialized as follows:

$$lb = -1 * ones(1, nvars);$$
$$ub = +1 * ones(1, nvars);$$
$$X = rand(1, nvars). * (ub - lb) + lb;$$

lb and *ub* represent the lower and upper bounds of decision variables, respectively. The objective function of the training process with GA is defined as: $fun = @(X)mse(Y_T^{Tr} - Y_P^{Tr})$ [17]. The *sim* function is used to get Y_P^{Tr} [18].

Optimization with GA is done with the next commands:

$$options = optimoptions('ga',' PlotFcn',"gaplotbestf");$$
$$rngdefault$$
$$[x, fval] = ga(fun, nvars, [], [], [], [], [], [], [], options);$$

The optimization process of weights and biases in ANN+GA and the related fitness calculation are summarized in Figures 3 and 4.

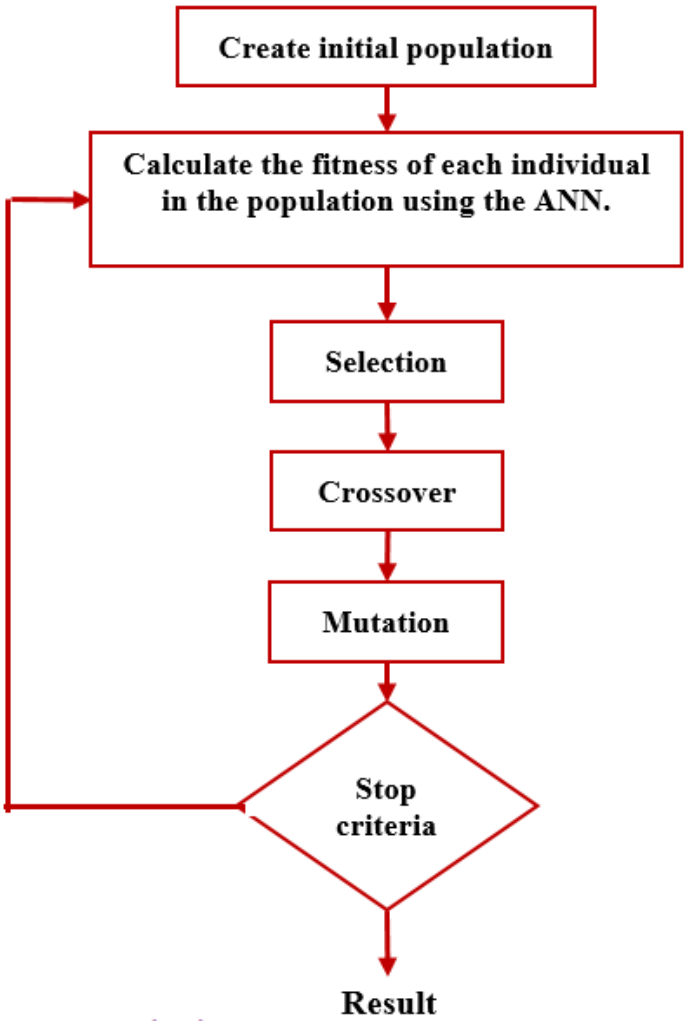

Figure 3. Flowchart of optimization of weights and biases in ANN+GA.

Each individual in the population consists of weights and biases. For example, if there are *n* weights and *m* biases, then each individual consists of *n*+*m* values, as:

$$(w_1, \dots, w_n, b_1, \dots, b_m)$$

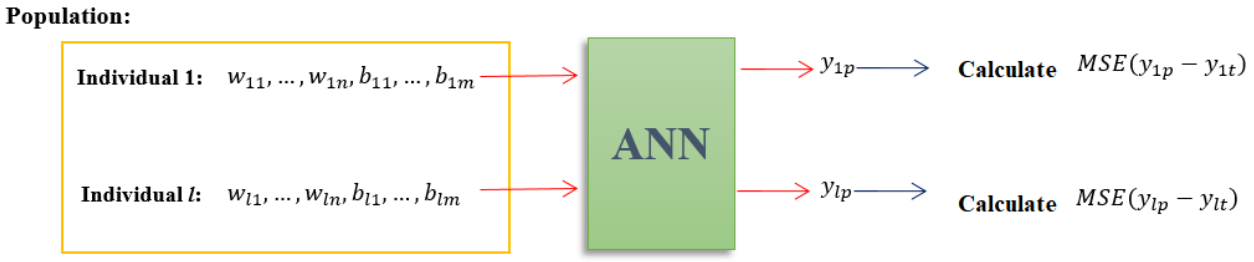

Figure 4. Fitness calculation of GA.

In a similar way, in PSO, the optimization is done with the following commands [19]:

$options = optimoptions('particleswarm',' PlotFcn',' pswplotbestf');$
rng default
$[x, fval, exitflag, output] = particleswarm(fun, nvars, lb, ub, options);$

As mentioned before, each data set is divided into two parts as training and test data. ANN's accuracy is assessed using the test data, over the outputs. It's calculated MSE between the *target* (*actual*) and *predicted* (*output*) values. The corresponding objective function is defined as in Function 2. Similar to the training process, the *sim* function is used to obtain ANN's outputs [18].

$$Minimize\ (MSE(Y_T^{Ts} - Y_P^{Ts})) = Minimize\ \frac{1}{d}\sum_{i=1}^{d}(y_{it}^{Ts} - Y_{ip}^{Ts})^2 \tag{2}$$

3 Experimental Results

The used data sets are *simple fitting, abalone shell rings, body fat percentage, building energy, chemical sensor, cholesterol, engine behavior,* and *vinyl bromide*, which are available in MATLAB. The dimensions of the inputs and outputs of the data sets are given in Table 1.

Table 1. The dimensions of the inputs and outputs of the used data sets [15].

Data set	Input	Output
simplefit	94×1	94×1
abalone	4177×8	4177×1
bodyfat	252×13	252×1
building	4208×19	4208×3
chemical	498×8	498×1
cho	264×21	264×3
engine	1199×2	1199×2
vinyl	68308×16	68308×1

The percentage of the data which is used for training is 75%. As an example, the input and output sizes of the *abalone* data set are 4177×8 and 4177×1, respectively, [15]. In this case, there are 4177 data, each has eight inputs and one output. So, the sizes of the input and output data sets used for training are 3132×8 and 3132×1, respectively. Assessment is done using the rest of the data, i.e. 25%. In fact, data of size 1045×8 is used as input from trained networks, and outputs of size 1045×1 are obtained. Assessment is done based on Equation 2.

Default options are used for selection, crossover and mutation in GA [20]. We use *MaxStallGenerations* and *MaxGenerations* in options. *MaxStallGenerations* checks the number of generations of GA without improvement to see whether there is progress or not, which is defined as equal to 500. The maximum number of generations is controlled by *MaxGenerations* that is set up as equal to 1000. Similar values have also been used for the options of PSO. Other options for both algorithms are as defaults, determined in the toolbox of global optimization.

We use a system with an Intel Core i7 processor, 1.8 GHz with 16 GB of RAM. Convergences of GA and PSO for *vinyl* are shown in Figure 5. Similar situations are valid for the other sets.

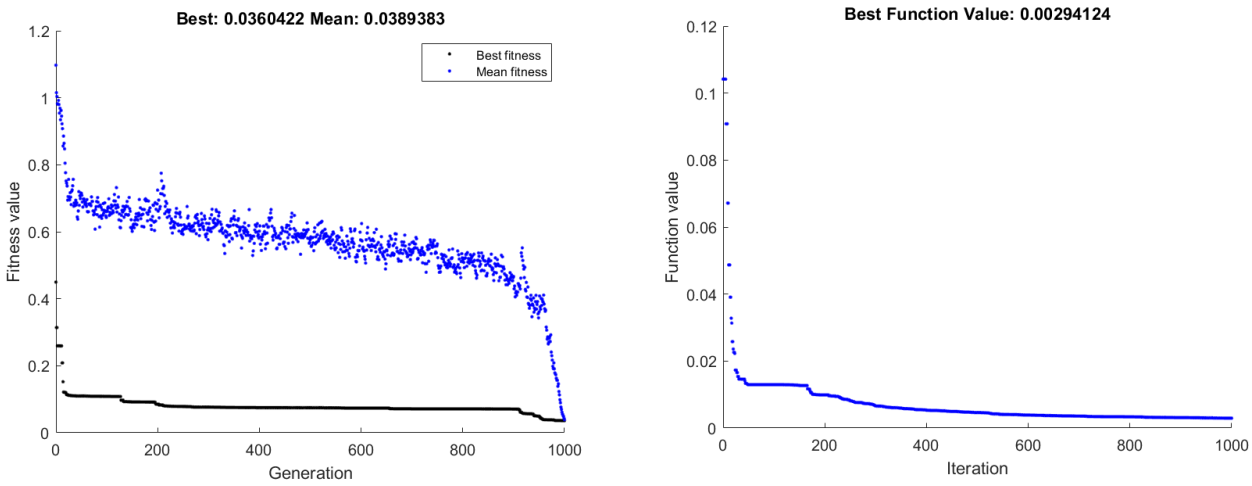

Figure 5. Convergence of the algorithms.

Results are presented in Table 2, in which the best results are bold. As seen, the best results are acquired by ANN for some data sets and also by ANN+PSO for some others, however, there is not much difference between the results. The computation time for ANN is less than ANN+GA and ANN+PSO. ANN+GA has not achieved the best result for any data set and has the highest computation times.

Considering the computation time for the used data sets, it may seem more reasonable to apply LMA for ANN optimization. But it should be noted that the sizes of the data sets used are not very large, and the other approach may provide better results for different ones.

Table 2. Obtained results. CPU times are in seconds.

	ARNN		ARNN+GA		ARNN+PSO	
	MSE	CPU time	MSE	CPU time	MSE	CPU time
simplefit	**0.001**	1.46	0.035	3309.98	0.084	1646.59
abalone	**0.021**	5.10	0.040	1722.87	0.024	907.61
bodyfat	0.136	2.16	0.168	1536.22	**0.052**	807.36
building	**0.036**	9.16	0.200	1903.38	0.045	966.16
chemical	0.013	2.14	0.055	1548.61	**0.009**	815.99
cho	0.093	4.10	0.092	1923.02	**0.028**	815.14
engine	**0.005**	3.10	0.056	1584.72	0.019	854.46
vinyl	**0.001**	15.05	0.036	4037.56	0.003	2299.53

4 Conclusion

In this study, to do a better estimation, the optimization of an ANN was done with two different approaches. In the first one, optimization was done with the LMA, while in the second approach, the values of weights and biases of ANN were optimized with EAs such as GA and PSO. These methods were demonstrated with ANN, ANN+GA, and ANN+PSO. Deep learning and global optimization toolboxes of MATLAB were used. The implementations were briefly described. Different sizes of data sets from MATLAB were used for training and testing. According to the acquired results, ANN achieved the best results for some data sets and also ANN+PSO for some others. However, there were no serious differences between the results. The ANN was the fastest approach while ANN + GA had the highest computation times.

An important point to note is that the obtained results depend on the established options, performance evaluation criteria, the used data set, and especially their size. For example, there are functions in MATLAB's deep learning toolbox different from the LMA that may provide different results. Therefore, it is planned to carry out more comprehensive studies by taking these factors into consideration, in future works. For this aim, a statistical experiment will be designed. Also, parallel programming

possibilities of the used toolboxes will be utilized for large benchmarks.

REFERENCES

[1] A. Teymourifar. Dinamik atölye çizelgeleme problemleri için taslim zamanı belirleme ve yeni sevk etme kuralları, Master's thesis, Anadolu Üniversitesi, 2014.

[2] A. Teymourifar, G. Ozturk. New dispatching rules and due date assignment models for dynamic job shop scheduling problems. International Journal of Manufacturing Research, 13(4), 302-329, 2018.

[3] A. Teymourifar, G. Ozturk. A Neural Network-based Hybrid Method to Generate Feasible Neighbors for Flexible Job Shop Scheduling Problem. Universal Journal of Applied Mathematics, 6(1), 1-16, 2018.

[4] S. N. Deepa, N. Y. Jayalakshmi. Optimized Fuzzy-Based Wavelet Neural Network Controller for a Non-Linear Process Control System. IETE Journal of Research, 1-10, 2021.

[5] T. Zarra, M. G. K. Galang, F. C. Ballesteros, V. Belgiorno, V. Naddeo. Instrumental Odour Monitoring System Classification Performance Optimization by Analysis of Different Pattern-Recognition and Feature Extraction Techniques. Sensors, 21(1), 114, 2021.

[6] V. A. Akimov,V. A. Minkin. Determination of Significant Behavioral Parameters on COVID-19 Diagnosis by Artificial Neural Networks Modeling. In Preprint of the 4th International Open Science Conference VIBRA2021: Modern Psychology. The Vibraimage Technology, 2021.

[7] V. Gholami, H. Sahour, M. A. H. Amri. Soil erosion modeling using erosion pins and artificial neural networks. Catena, 196, 104902, 2021.

[8] D. J. Montana. Neural network weight selection using genetic algorithms. Intelligent Hybrid Systems, 8(6), 12-19, 1995.

[9] Y. T. Chang, J. Lin, J. S. Shieh, M. F. Abbod. Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction. Advances in Fuzzy Systems, 2012.

[10] S. Khademikia, A. Haghizadeh, H. Godini, G. S. Khorramabadi. Artificial neural network-cuckoo optimization algorithm (ANN-COA) for Optimal control of Khorramabad wastewater treatment plant, Iran. Civil Engineering Journal, 2(11), 555-567, 2016.

[11] A. Lee, Z. W. Geem, K. D. Suh. Determination of optimal initial weights of an artificial neural network by using the harmony search algorithm: application to breakwater armor stones. Applied Sciences, 6(6), 164, 2016.

[12] D. K. Bui, T. N. Nguyen, T. D. Ngo, H. Nguyen-Xuan. An artificial neural network (ANN) expert system enhanced with the electromagnetism-based firefly algorithm (EFA) for predicting the energy consumption in buildings. Energy, 190, 116370, 2020.

[13] newff Create a feed-forward backpropagation network. - Obsoleted in R2010b NNET 7.0., online available from https://www.mathworks.com/matlabcentral/answers/77741-newff-create-a-feed-forward-backpropagation-network-obsoleted-in-r2010b-nnet-7-0, access date: April 2021.

[14] Function Approximation and Nonlinear Regression, online available from https://www.mathworks.com/help/deeplearning/function-approximation-and-nonlinear-regression.html?s_tid=CRUX_lftnav, access date: April 2021.

[15] Sample Data Sets for Shallow Neural Networks, online available from https://www.mathworks.com/help/deeplearning/gs/sample-data-sets-for-shallow-neural-networks.html, access date: April 2021.

[16] How to Show the Weight or Bias in a Neural Network?, online available from https://www.mathworks.com/matlabcentral/answers/2356-how-to-show-the-weight-or-bias-in-a-neural-network, access date: April 2021.

[17] Find minimum of function using genetic algorithm, online available from https://www.mathworks.com/help/gads/ga.html#mw_f69991d0-5b23-4023-8c8e-e2890f2474d5, access date: April 2021.

[18] Simulate neural network, online available from https://www.mathworks.com/help/deeplearning/ref/sim.html, access date: April 2021.

[19] Particle swarm optimization, online available from https://www.mathworks.com/help/gads/particleswarm.html, access date: April 2021.

[20] How the Genetic Algorithm Works, online available from https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.htm, access date: April 2021.