Journal of Theoretical and Applied Information Technology
15th July 2020. Vol.98. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TRAINING STRATEGY FOR A NEURAL NETWORK USING A MODIFIED GENETIC ALGORITHM

[1]HOLMAN MONTIEL A., [2]FREDY H. MARTÍNEZ S., [3]EDWAR JACINTO G.

[1, 2, 3] Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Bogotá D.C., Colombia

E-mail: [1]hmontiela@udistrital.edu.co, [2]fhmartinezs@udistrital.edu.co, [3]ejacintog@udistrital.edu.co

ABSTRACT

There are different computer models to classify groups of data, among which are neural networks, vector support machines, numerical methods, among others. However, in some cases these strategies consume a large amount of computer resources, reducing the speed of operation during their execution in the various electronic development devices. In this work, a partial solution to this limitation is proposed. The algorithm developed is a classifier that incorporates a backward-propagation neural network, which is trained by means of a modified genetic algorithm that is in charge of finding the appropriate set of weights for the neural network to classify a given group of random numbers. The proposed optimization will allow the use of this algorithm in various classification problems, not only in conventional computing units, but also on various technological platforms with reduced properties (embedded systems), maintaining an optimal balance between the use of resources and the speed of response of the device used.

Keywords: *Neural Network; Classifier; Machine Learning; Genetic Algorithms; Correlation.*

.

1. INTRODUCTION

Automatic learning is one of the pillars studied in computer science, since, through supervised and unsupervised learning techniques, it generalizes processes with the ability to solve problems like a living being. In certain cases, such processes require efficient classification methods, since the amount of computer resources they consume must be optimized to avoid cost overruns during their operation or actual implementation [1-3].

Despite this, machine learning techniques are making a large-scale foray into industrial automation; some of the applications found are based on pattern recognition, raw material sorting or fault prediction. This incursion has turned the pattern classification into a research problem in the area of automation and control, in which researchers are given the task of finding efficient methodologies to design machine learning algorithms, in order to optimize the computer cost they may have [4-5].

A trend to optimize resources in general is based on co-evolution, which could be defined as an evolutionary change between interrelated species guided by natural selection. Thanks to this phenomenon there is a great diversity of species which has helped nature in the forging of life, in addition to the interactions that make it possible to conserve genetic diversity.

This concept has become part of the family of evolutionary techniques and is inspired by the principles of natural evolution. Co-evolutionary models are very similar to conventional evolutionary algorithms, individuals are coded and evolved by genetic operators, and are selected according to their level of aptitude. However, they differ because in co-evolution, interactions between individuals are required [6-8].

These interactions are represented by the concepts of cooperation and competition for the exploitation and exploration of the regions of interest. Co-evolution is used to solve optimization problems, when there are large and complex search areas, since, thanks to this methodology, these areas can be divided into smaller and more easily soluble areas [9-10].

More specifically, collaborative strategies allow individuals from a solution set to cooperate with each other to solve a problem [9]. The opposite is true for models based on competitive co-evolution, since individuals in a solution set compete to find a partial solution to a problem. However, the search for communities in this type of algorithms has been a research topic, to improve the individuals in the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

solution sets and increase their capacity to solve different types of problems [10].

A limitation of this type of technique is that it lacks memory, therefore, in some cases learning models are implemented to store information of interest for a certain amount of time. For example, the implementation of intelligent classifiers that discard defective products. However, this technique requires prior training of the classifier reducing its ability to adapt and learn from the process. Some improvements to these classifiers have incorporated learning processes based on least squares that are executed online, that is, while the process is running the classifier is modified to adapt to the new characteristics of the process [11-12].

The results obtained by the above-mentioned training algorithms and some others are very good, however, they do not allow optimizing their consumed computational resources by reducing the speed of operation in the system where they are executed [12].

Consequently, this paper proposes a learning strategy that reduces the computational cost by training a neural network-based classifier. The strategy combines a modified genetic algorithm, which incorporates several crossing and mutation mechanisms to find the most appropriate combination of weights to allow the network to classify random data series.

2. MATERIALS AND METHODS

Neural networks and bio-inspired optimization algorithms are computational models that are built to emulate certain characteristics of human cognitive behavior. The definition and behavior of these strategies is presented in detail below.

2.1 Artificial Neural Networks (ANN)

Neuronal networks are inspired by neurons and the nervous system of living beings, in the same way that synapses between neurons are made, approximate models of neurons collaborate with each other to modify a set of weights and generate an output stimulus [12-13].

In some cases, the neurons in these network systems are grouped in three different layers, where the first layer represents the input data that is sent to the second layer, which performs the synapse process to send an output parameter to the third layer. The synapse is performed by modifying a series of parameters called weights, which vary according to the complexity of the neural network and can be grouped in more than one layer (see Figure 1) [13].

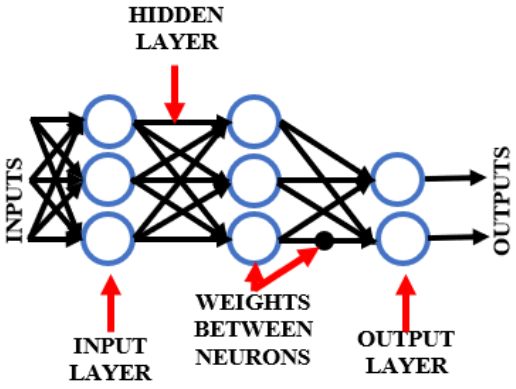


Figure 1: Topology of a multi-layered neural network (Based on [14, 16]).

Finding approximate values for the weights is a very complex task, since as the size of the network increases, so do the number of possible solutions it may have. Normally, the weights are updated using a back-propagation algorithm which uses a calculation based on the descending gradient to estimate the weight values, reducing the mean square error between the training sample set and the result obtained, as close to zero as possible from a set of validation samples [14-15].

It can be said that the combination of these algorithms allows the creation of new types of network, such as the so-called "back-propagation network" which is a multi-layer network that uses the back-propagation algorithm through a generalization of the least-squares algorithm. By combining the two algorithms the learning of the network is done based on the mean square error and the idea is that the learning is supervised, that is, a set of samples should be provided for the updating of the weights and to estimate an output value [14-15].

The simplest retro-propagation network has three layers, e.g. the one described in Figure 2, which has 2 inputs and one output. The figure shows that each neuron is composed of two layers, where the first is the sum of the inputs by their respective weights and the second contains the activation function. The activation function is in charge of defining the output of a node from an input, in other words, if the output of the network is represented as a sum $y = f(h)$where, y would be the approximate output provided by the node and h the trigger signal.

Journal of Theoretical and Applied Information Technology
15th July 2020. Vol.98. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

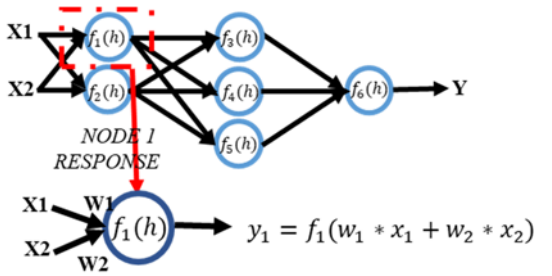


Figure 2: Response of a two-layer network.

The input data sets, such as the x_1 y x_2 that are assigned to expected outputs yThe data sets are used for network training through an iterative process. Each time an iteration is executed, an attempt is made to modify the weights of the network to obtain a better approximation of the output value. This is done by propagating the information and the result shown in Figure 3 [14, 16] is obtained.

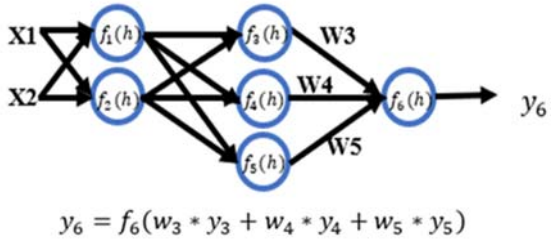


Figure 3: Response of the first sweep of the network.

Considering that the inputs to each node are multiplied and added up at the end of the operations between all the nodes, the output value obtained is compared with the expected value. The difference between these two values is known as the "signal error δ"and spreads back to all neurons, starting with the last neuron that was the exit from the network (see Figure 4) [14, 16].

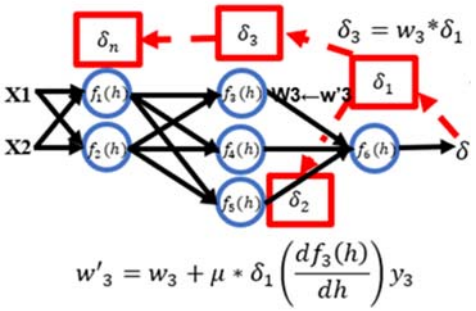


Figure 4: Error propagation.

By propagating the error each new weight that is assigned has a learning rate assigned μ. The learning rate can be estimated by various methods or be fixed with a constant value. For example, the value of μ is minimizing its magnitude exponentially as training iterations progress.

As the training process progresses, the weights change as shown in Figure 4, the weight w_3 changes to w'_3 (in each iteration) which depends on the learning rate μthe error of the signal δ and the derivative of the activation function $\left(\frac{df_3(h)}{dh}\right)$ multiplied by the appropriate entry [14, 16].

An activation function is the function that can represent the output of a neuron from the input data, such functions are usually expressed as mathematical functions, such as: S-shaped, Gaussian, rectangular or trapezoidal [14, 16].

2.2 Genetic Algorithm

Unlike neural networks, bio-inspired optimization algorithms try to find the best value among a given set of values using various techniques, most notably genetic algorithms. These algorithms emulate the natural selection of species to find a solution to a problem, make use of crossing and mutation methods, in order to generate new species and thus improve the search process [17].

The conventional genetic algorithm starts with a selection process by which it generates a new set with the best individuals from the solution set based on their aptitude value (Result of the evaluation of the individual). The crossing process consists of taking 2 or more individuals from the solution set and combining their characteristics to generate new individuals and incorporate them into the set. Finally, the mutation process oversees generating a new individual and changing it for one from the set of solutions, as presented in the following algorithm [16-17].

Algorithm 1. *Conventional Genetic Algorithm*

```
Start Variables;
P0=Generate initial solution set ();
/*The initial population (Solution Set) is generated randomly*/
While Termination condition = True to do
        Fitness=Evaluate Function (P0);
        Children=Selection (P0, Fitness)
        Sons= Crossing (Sons);
        Sons=Mutation (Sons);
        P0=Sons;
End While
```

The operation of these approaches is based on the theory of schemes proposed by Holland [19]. This theory says that by coding everyone in binary (genetic chain) it converges to a solution by performing the different operations (Assessment of fitness, selection, mutation and crossing) on each bit of the chain (chromosome) directly. Where the

Journal of Theoretical and Applied Information Technology
15th July 2020. Vol.98. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

mathematical expressions of these operations are described taking into account the following: L is the size of the population, K is the size of the individual d or of the string and $\varphi(E,t)$ is the number of individuals in the population over time t described by the solution set E [16-19].

The suitability of the solution set in time t, $evaluate(E,t)$ is defined as the average of the aptitude of all the individuals who make up the solution set. If there are several strings n described by the set of solutions over time t, the result is shown in equation 1 [16-19].

$$evaluate(E,t) = \sum_{i=1}^{n} \frac{evaluate(d_i)}{n} \quad (1)$$

When performing the selection operation, each string is copied according to its suitability value, whose probability is expressed as shown in equation 2.

$$p_i = \frac{evaluate(d_i)}{F(t)} \quad (2)$$

Where $F(t)$ is the sum of all the skills of the individuals in the solution set as shown in equation 3.

$$F(t) = \sum_{i=1}^{L} evaluate(d_i) \quad (3)$$

At the end, you get several individuals $\varphi(E,t+1)$ described by the solution set E, this can be represented by the mathematical expression of equation 4.

$$\varphi(E,t+1) = \varphi(E,t)\frac{evaluate(E,t)}{F(t)} \quad (4)$$

The above shows that the number of individuals given by the solution set E at $t+1$, is represented by: The number of individuals is equal to that of time t with characteristics like those of the time t. The suitability of the solution set in t and the average fitness of the population.

These characteristics allow the selected individuals in time $t+1$ have two characteristics; individuals with a proficiency value higher than the population average will have a higher probability of replicating themselves in the next generation. The opposite case occurs when the skill value of an individual has a value below the population average, since, it will not have a probability of replicating itself in the next generation (they tend to disappear).

Other effects that should be considered during the execution of this algorithm are crossover and mutation. Crossing allows to generate new individuals from two existing individuals (as shown in Figure 5) [16-19].

Figure 5: Single point crossing strategy.

Assuming that two individuals were randomly selected and break at a random crossing point, as observed, crossing both individuals results in two new individuals like their parents, which replace the parent individuals. Normally the crossing point is selected using a uniform distribution with a maximum value of $K-1$ where K is the length of the individual; in other words, the individual d has a probability of disappearing or prevailing as indicated by equations 5 and 6 respectively ($\delta(E)$ represents the individual coded in real numbers).

$$p_{de}(E) = \frac{\delta(E)}{m-1} \quad (5)$$

$$p_{sob}(E) = 1 - \frac{\delta(E)}{m-1} \quad (6)$$

However, some authors propose a probability for the occurrence of a crossing event pc; In that case it is already known that the individual will survive and is represented as indicated in equation 7.

$$p_{sob}(E) = 1 - pc\frac{\delta(E)}{m-1} \quad (7)$$

The combined effect of crossover and selection affects generational replacements as shown in equation 8. This allows the suitability value of above-average schemes to be increased and thus to have a greater probability of replication and passage to the next generation [16-19].

$$\varphi(E,t+1) = \varphi(E,t)\frac{evaluate(E,t)}{F(t)}\left[1 - pc\frac{\delta(E)}{m-1}\right] \quad (8)$$

Unlike the crossover, the mutation must be very careful, since, the change in many bits can destroy the individual or the changes made will not benefit the overall solution set. If the probability of modifying a bit is p_m, This means that the probability of survival will be its inverse, where the operation results in a new individual (see Figure 6) [16-19].

Journal of Theoretical and Applied Information Technology
15th July 2020. Vol.98. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 6: Example of a mutation

Each mutation can be represented by equation 9 and each mutation affects the behavior of the solution set as shown in equation 10.

$$p_{sob}(E) = E * N[0,1]$$
$$where\ N\ represents\ a\ normal\ distribution \quad (9)$$

$$\varphi(E,t+1) = \varphi(E,t)\frac{evaluate(E,t)}{F(t)}\left[1 - pc\frac{\delta(E)}{m-1}\right] * p_m \quad (10)$$

3. IMPLEMENTATION

The algorithm developed is a classifier that incorporates a retro-propagation neural network, which is trained by a modified genetic algorithm that is responsible for finding the appropriate set of weights for the neural network to classify a given group of random numbers. The design of this algorithm (see Figure 7) is based on a conventional genetic algorithm but differs in that the single-point cross operator was replaced by an operator that reverses the positions of the weight vector.

In other words, the proposed algorithm has two mutation operators, the first depends on a crossing probability and the second incorporates an operation with a Gaussian random number generator centred on 0 with a variance of 1, which reduces the size of the mutations (with increments of 0.01 in each generation) as the process advances, therefore, it was not made dependent on some probability. The structure of the proposed algorithm is shown in Figure 8.

Once the neural network is created (see Figure 9) a solution set or initial population is created, in which its elements (individuals) change using two types of mutation, where the first mutation depends on a probability threshold set at 0.3. When this threshold is exceeded by a randomly generated number, the mutation is executed.

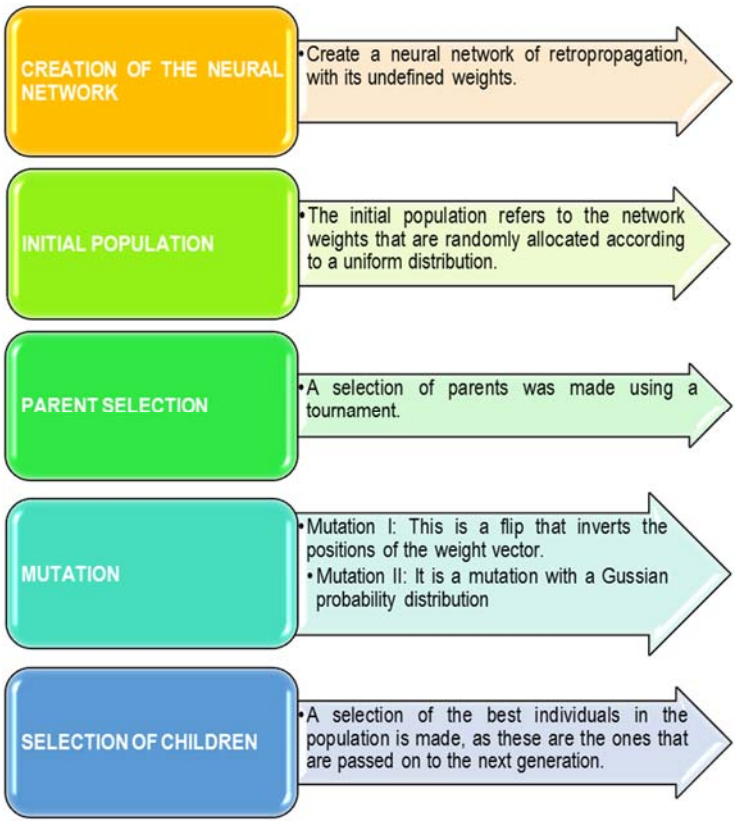


Figure 7: Training strategy

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

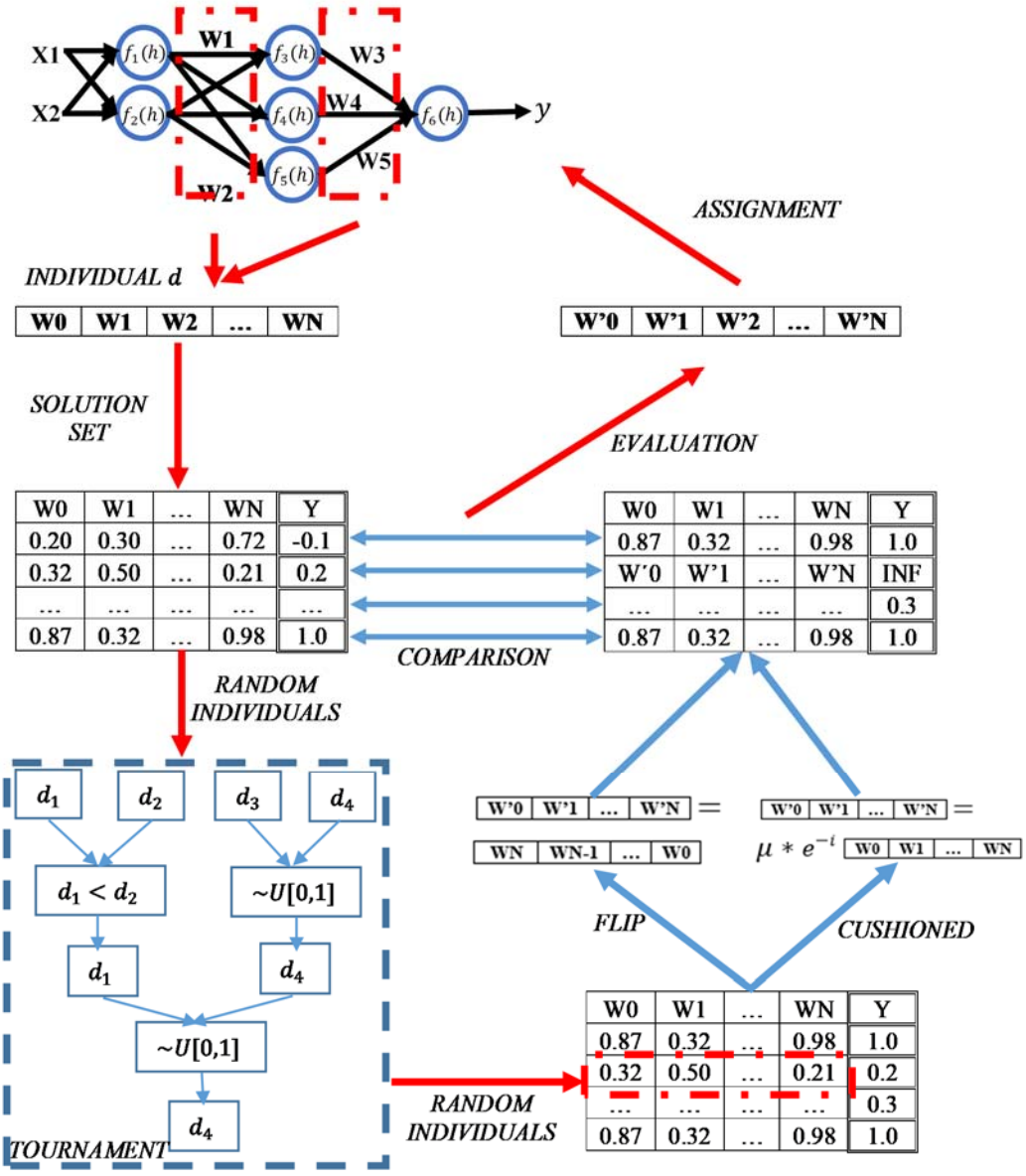


Figure 8: Optimization routine

Considering the structure of the genetic algorithm, a selection per tournament was incorporated, which selects four individuals at random and compares them to each other in three different ways to take the best one. The first phase of the tournament selects the individual with the highest aptitude value, in the same parallel phase an individual is selected at random. The individual is selected by assigning a probability to each one according to its aptitude value, then a random number with uniform distribution is generated and the individual with the aptitude value closest to the generated number is selected.

In the second phase of the tournament the same selection mechanism is used generating a random number with uniform distribution. However, to improve the effectiveness of the algorithm a steady state generation replacement was implemented, which allows new generations of individuals to prevail only if proficiency value is better. The aptitude value of each set of weights is evaluated by replacing in the network all the existing weights by those estimated with the algorithm, unlike the descending gradient the weights are estimated and replaced without taking into account the error; as shown in the Algorithm 2.

Journal of Theoretical and Applied Information Technology
15th July 2020. Vol.98. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

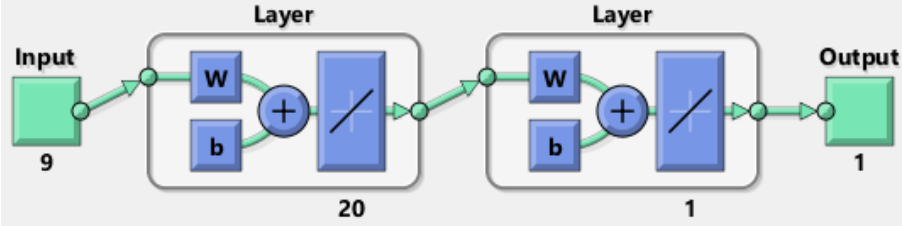


Figure 9: Structure of the network

Algorithm 2. *Proposed Genetic Algorithm*

```
Modified AG program ()
    Start Variables;
    Network=Create network (weights, hidden layers);
    /*Create neural network*/
    P0=Initial population U~ [0. weights];
    /*Start grid weights*/
    While != Stop condition
        Proficiency=Testing network weights ();
        Parents= Selection (P0, proficiency);
        /*Apply selection per tournament*/
        If R~U [0.1] <Pc
         /*If the random number exceeds the Pc threshold, the
         mutation is activated*/
         Children=Flip (P0, Parents);
         Sons=Mutation (Sons, sigma);
        /*This mutation performs the following operation
         H=H*sigma*/
         P0=Selection (Children, fitness);
        /*A selection is made to take the best individuals*/
        /*And make the selection*/
        sigma=sigma*e^(-1/iteration);
      End While
End of program
```

The Gaussian mutation used to modify the value of the weights is represented by multiplying a scalar value provided by equation 11 and the aptitude value of each individual is the correlation (Relationship between variables) between the output estimated by the network and the training data, is calculated using equation 12.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma}} * e^{-\frac{(x_i-\bar{x})^2}{2\sigma^2}} \quad (11)$$

$$r = \frac{(\sum_{i=1}^{N}(x_i-\bar{x})*(y_i-\bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i-\bar{y})^2}} \quad (12)$$

Where $\mathbf{P(x)}$ is the probability value, $\mathbf{x_i}$ is a value in the data range, $\boldsymbol{\sigma}$ is the standard deviation, $\boldsymbol{\sigma^2}$ is the variance and $\bar{\mathbf{x}}$ is the mean or average of the measured data [16].

4. RESULTS

The neural network created to implement the classifier has 20 hidden layers and a linear-type activation function in each layer. The training data of the network are composed by a set of data available in a repository, which proposes a problem to test this kind of computational learning strategies [20].

In this case, the validation and testing of the proposed strategy was carried out by seeking the solution to a classification problem of three wines (3 classes) with nine different characteristics. The characteristics are available as a 9-column arrangement and the wine to which they belong in a separate column or class, i.e. each row represents the characteristics of a wine, see Table 1. The total number of samples used for the validation of this example was 1926, based on 9 different characteristics stored in 214 records.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Example of Samples Used

	Features									
#Reg	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***Class***
1	152.101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.00	**1**
2	151.761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	**1**
3	151.618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	**1**
4	151.766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	**1**
5	151.742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	**1**
71	151.574	14.86	3.67	1.74	71.87	0.16	7.36	0.0	0.12	**2**
72	151.848	13.64	3.87	1.27	71.96	0.54	8.32	0.0	0.32	**2**
73	151.593	13.09	3.59	1.52	73.10	0.67	7.83	0.0	0.00	**2**
74	151.631	13.34	3.57	1.57	72.87	0.61	7.89	0.0	0.00	**2**
75	151.596	13.02	3.56	1.54	73.11	0.72	7.90	0.0	0.00	**2**
147	151.769	13.65	3.66	1.11	72.77	0.11	8.60	0.0	0.00	**3**
148	151.610	13.33	3.53	1.34	72.67	0.56	8.33	0.0	0.00	**3**
149	151.670	13.24	3.57	1.38	72.70	0.56	8.44	0.0	0.10	**3**
150	151.643	12.16	3.52	1.35	72.89	0.57	8.53	0.0	0.00	**3**
151	151.665	13.14	3.45	1.76	72.48	0.60	8.38	0.0	0.17	**3**

Table 2 shows the average data obtained in 50 runs of the algorithm and their respective margin of error, then in Table 3 presents the behavior of genetic algorithms during their evolution (after fifty generations).

The problem was solved with 3 different algorithms, initially tested with a neural network trained with the descending gradient strategy, then trained with the conventional genetic algorithm and finally with the proposed algorithm. The performance of the neural network was measured with a correlation between the training data and the data obtained by modifying its weights, i.e. the graphs in Table 3 represent the output behavior of the network and if this value is close to one, it can be said that the network has a large number of hits in the prediction of wines from its characteristics.

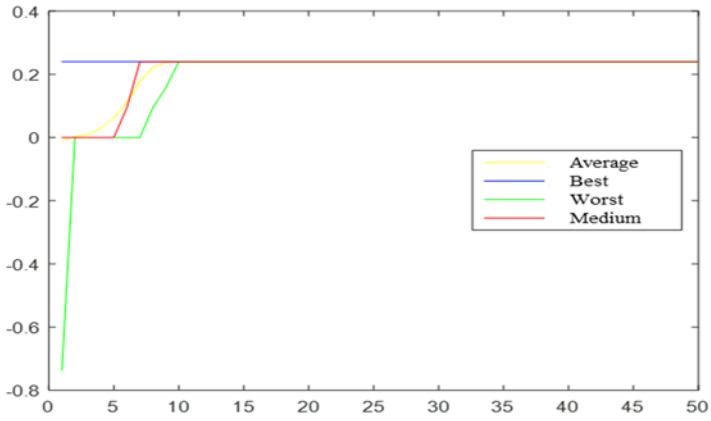
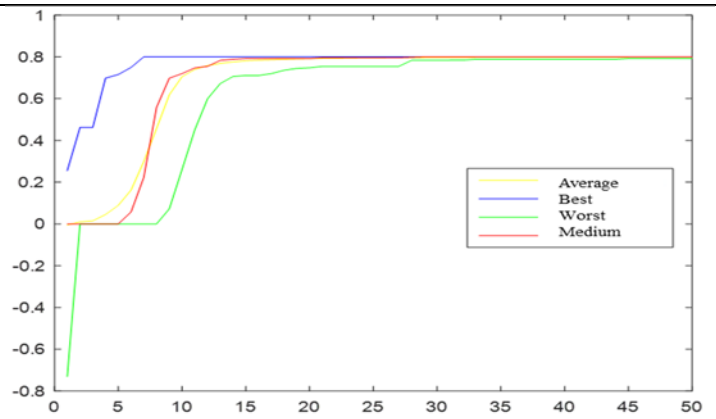
It is important to emphasize the execution times achieved, because a substantial improvement is obtained in this process, since with the modified algorithm it only takes an average of 42.48 milliseconds to obtain the analysis of the samples (class classification) while only with the neural network it takes a time of 249.38 milliseconds. The software validation tests were performed on a development interface on MATLAB 2015a, which runs on a 64-bit Windows 10 operating system with an Intel Core I7-4702MQ 2.2 GHz CPU and 8 Gb of RAM.

Table 2: Statistical Results

	Best	***Worst***	***Average***	***Medium***	***Learning time***
Descending gradient	0.75±0	0.75±0	0.75±0	0.75±0	4 hours 55 min
Algorithm genetic conventional	0.2397±0.12	-0.651±0.13	0.2397±0.1	0.2397±0.05	5 hours 20 min
Algorithm genetic Modified	0.8124±0.08	-0.0050±0.01	0.8123±0.09	0.8074±0.01	4 hours 45 min

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3: Results Graphs

Conventional Algorithm	
Modified Algorithm	

5. CONCLUSIONS

The implemented strategy improved the performance of the conventional genetic algorithm, because the single point crossing does not allow the population to diversify as much as the *flip* operator and the Gaussian mutation combined. This is shown in Tables 2 and 3 where it is appreciated that the correlation value found by the proposed strategy is higher than the one found using the conventional genetic algorithm, according to these results we can say that the solution of the optimized algorithm proposed by this work reflected an increase in performance of 82.97 % (42.48 milliseconds), with respect to the response time associated with the solution of classification problems using only neural networks and a conventional genetic algorithm (249.38 milliseconds). By omitting the error calculation as done with the back-propagation network and replacing them with genetic operators, it is possible to optimize the amount of resources consumed by the computer during neural network training, increasing the processing speed.

This proposed optimization will allow the use of this algorithm in various classification problems, not only in conventional computer units (PCs), but can also be used on various technological platforms with reduced properties (embedded systems), maintaining an optimal balance between the use of resources and the speed of response of the device used.

REFERENCES

[1] M. H. Roy-Cardinal, F. Destrempes, G. Soulez and G. Cloutier, "Assessment of carotid artery plaque components with machine learning classification using homodyned-K parametric maps and elastograms," in *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2019, pp. 493-504.

[2] K. Bakshi and K. Bakshi, "Considerations for artificial intelligence and machine learning: Approaches and use cases," in 2018 *IEEE Aerospace Conference*, Big Sky, MT, USA, 2018, pp. 1-9.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[3] F. Elghibari, R. Elouahbi and F. El Khoukhi, "An Automatic Updating Process to Control The E-learning Courseware." *International Journal on Advanced Science, Engineering and Information Technology*, vol 7, no. 2, pp. 546-551, 2017.

[4] Q. Liu, Y. Zhao, Y. Zhang, D. Kang, Q. Lv and L. Shang, "Hierarchical context-aware anomaly diagnosis in large-scale PV systems using SCADA data," in 2017 *IEEE 15th International Conference on Industrial Informatics (INDIN)*, Emden, 2017, pp. 1025-1030.

[5] R. Samdarshi, N. Sinha and P. Tripathi, "A triple layer intrusion detection system for SCADA security of electric utility," in *2015 Annual IEEE India Conference (INDICON)*, New Delhi, 2015, pp. 1-5.

[6] S. Sen, E. Aydogan and A. I. Aysan, "Coevolution of Mobile Malware and Anti-Malware," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2563-2574, Oct. 2018.

[7] J. i. Matsuoka, Y. Nakashima and S. Ono, "A preliminary study on designing a benchmark problem for analysis of sparsely-synchronized heterogeneous coevolution," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, 2017, pp. 1-8.

[8] E. Batot, W. Kessentini, H. Sahraoui and M. Famelis, "Heuristic-Based Recommendation for Metamodel — OCL Coevolution," in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, 2017, pp. 210-220.

[9] C. H. Chen and C. B. Liu, "Reinforcement Learning-Based Differential Evolution with Cooperative Coevolution for a Compensatory Neuro-Fuzzy Controller," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4719-4729, Oct. 2018.

[10] R. Janssen, S. Nolfi, P. Haselager and I. Sprinkhuizen-Kuyper, "Cyclic Incrementality in Competitive Coevolution: Evolvability through Pseudo-Baldwinian Switching-Genes," *Artificial Life*, vol. 22, no. 3, pp. 319-352, Aug. 2016.

[11] Y. Liang, J. Wang and C. Chen, "Research on neural network control method with end point bias," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Wuhan, China, 2018, pp. 153-157.

[12] M. L. Tej and S. Holban, "Determining optimal neural network architecture using regression methods," in *2018 International Conference on Development and Application Systems (DAS)*, Suceava, Romania, 2018, pp. 180-189.

[13] A. Mahdi and J. Qin, "Bottom up saliency evaluation via deep features of state-of-the-art convolutional neural networks," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, Las Vegas, NV, 2018, pp. 247-250.

[14] H. Moayedi et al. "A systematic review and meta-analysis of artificial neural network application in geotechnical engineering: theory and applications." in *Neural Computing and Applications*, pp. 1-24, 2018.

[15] D. Ulinic, J. Casas. "Why neural networks and deep learning are the future in machine learning", B. Eng. thesis, Universitat Autónoma de Barcelona, Spain, Mar. 2018.

[16] Z. Luo et al. "Structure-property relationships in graphene-based strain and pressure sensors for potential artificial intelligence applications." In *Sensors*, vol. 19, no 5, p. 1250, 2019.

[17] T. Barman and N. Deb, "State of the art review of speech recognition using genetic algorithm," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, Chennai, India, 2017, pp. 2944-2946.

[18] M. Shnan, T. H. Rassem & N. S. Zulkifli, "Facial image retrieval on semantic features using adaptive mean genetic algorithm". *Telkomnika*, vol.17, no. 2, pp. 882-896, 2019.

[19] Liang Ming, Yu-Ping Wang and Yu-ming Cheung, "A new schema theorem for uniform crossover based on ternary representation," in *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference*, 2004, pp. 235-239.

[20] D. Dua and T. Karra Taniskidou. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2017.