Evolutionary Algorithm for Optimal Connection Weights in Artificial Neural Networks

G.V.R. Sagar *nusagar@gmail.com*
Assoc. Professor,
G.P.R. Engg. College.
Kurnool AP, 518007, India.

Dr. S. Venkata Chalam *sv_chalam2003@yahoo.com*
Professor,
CVR Engg. College,
Hyderabad AP, India

Manoj Kumar Singh *mksingh@manuroresearch.com*
Director,
Manuro Tech. Research,
Bangalore, 560097, India

Abstract

A neural network may be considered as an adaptive system that progressively self-organizes in order to approximate the solution, making the problem solver free from the need to accurately and unambiguously specify the steps towards the solution. Moreover, Evolutionary computation can be integrated with artificial Neural Network to increase the performance at various levels; in result such neural network is called Evolutionary ANN. In this paper very important issue of neural network namely adjustment of connection weights for learning presented by Genetic algorithm over feed forward architecture. To see the performance of developed solution comparison has given with respect to well established method of learning called gradient decent method. A benchmark problem of classification, XOR, has taken to justify the experiment. Presented method is not only having very probability to achieve the global minima but also having very fast convergence.

Keywords: Artificial Neural Network, Evolutionary Algorithm, Gradient Decent Algorithm, Mean Square Error.

1. INTRODUCTION

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if “questions. ANN can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neurons outputs and neuron inputs. Based on the connection pattern (architecture), ANN can be grouped into two categories: (a) Feed Forward Networks allow signals to travel one-way only,

from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer as shown in Fig.(1) (b)Recurrent Networks can have signals traveling in both directions by introducing loops in the network

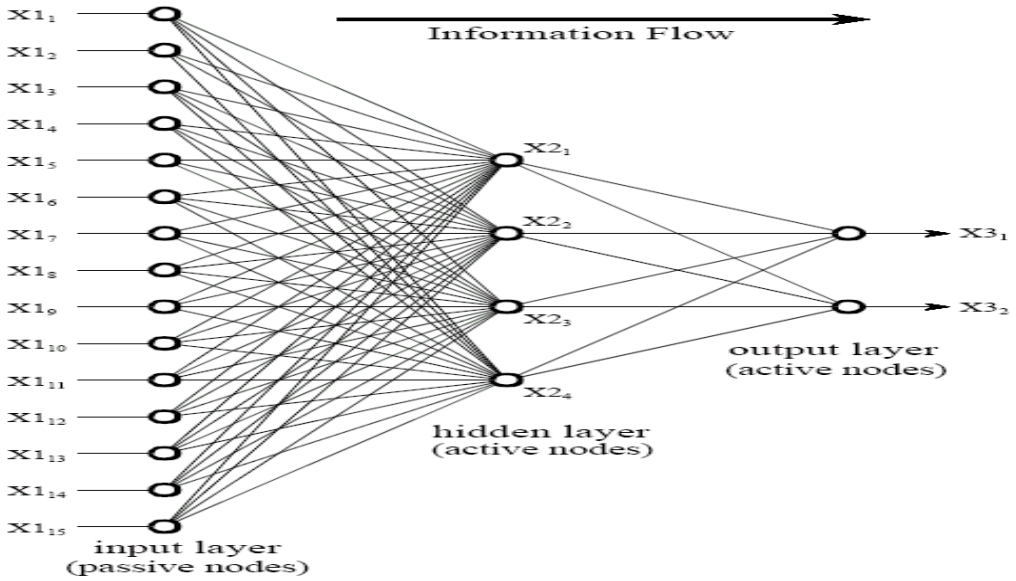

FIGURE 1: Feedforward architecture

Learning in artificial neural systems may be thought of as a special case of machine learning. Learning involves changes to the content and organization of a system's knowledge, enabling it to improve it's performance on a particular task or set of tasks. The key feature of neural networks is that they learn the input/output relationship through training. There are two types of training/learning used in neural networks, with different types of networks using different types of training. These are Supervised and Unsupervised training, of which supervised is the most common for feed forward architecture training modes. Supervised Learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights, which minimizes the error. One well-known method, which is common to many learning paradigms, is the gradient decent based learning.

The idea behind learning in Neural Network is that, the output depends only in the activation, which in turn depends on the values of the inputs and their respective weights. The initial weights are not trained with respect to the inputs, which can result in error. Now, the goal of the training process is to obtain a desired output when certain inputs are given. Since the error is the difference between the actual and the desired output, the error depends on the weights, and we need to adjust the weights in order to minimize the error.

2. GRADIENT DESCENT LEARNING

Gradient descent is a first order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent. Gradient descent is also known as steepest descent, or the method of steepest descent. A gradient descent based optimization algorithm such as back-propagation (BP) [6] can then be used to adjust connection weights in the ANN iteratively in order

to minimize the error. The Gradient descent back-propagation algorithm [7] is a gradient descent method minimizing the mean square error between the actual and target output of multilayer perceptrons. The Back-propagation [8], [9] networks tend to be slower to train than other types of networks and sometimes require thousands of epochs. When a reduced number of neurons are used the Error Back-propagation algorithm cannot converge to the required training error. The most common mistake is in order to speed up the training process and to reduce the training errors, the neural networks with larger number of neurons than required. Such networks would perform very poorly for new patterns nor used for training[10]. Gradient descent is relatively slow close to the minimum. BP has drawbacks due to its use of gradient descent [11, [12]. It often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or non differentiable. A detailed review of BP and other learning algorithms can be found in [13], [14], and [15].

3. EVOLUTIONARY ARTIFICIAL NEURAL NETWORK

Evolutionary artificial neural networks (EANN's) refer to a special class of artificial neural networks (ANN's) in which evolution is another fundamental form of adaptation in addition to learning [2] – [5]. Evolutionary algorithms (EA's) are used to perform various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialization, rule extraction from ANN's, etc. One distinct feature of EANN's is their adaptability to a dynamic environment. The two forms of adaptation, i.e., evolution and learning in EANN's, make their adaptation to a dynamic environment much more effective and efficient. Evolution has been introduced into ANN's at roughly three different levels: connection weights, architectures, and learning rules. The evolution of connection weights introduces an adaptive and global approach to training, especially in the reinforcement learning and recurrent network learning paradigm where gradient-based training algorithms often experience great difficulties. The evolution of architectures enables ANN's to adapt their topologies to different tasks without human intervention and thus provides an approach to automatic ANN design as both ANN connection weights and structures can be evolved.

4. EVOLUTION OF CONNECTION WEIGHTS

Weight training in ANN's is usually formulated as minimization of an error function, such as the mean square error between target and actual outputs averaged over all examples, by iteratively adjusting connection weights. Most training algorithms, such as BP. and conjugate gradient algorithms are based on gradient descent. There have been some successful applications of BP in various areas .One way to overcome gradient-descent-based training algorithms' shortcomings is to adopt EANN's, i.e., to formulate the training process as the evolution of connection weights in the environment determined by the architecture and the learning task. EA's can then be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information. The fitness of an ANN can be defined according to different needs. Two important factors which often appear in the fitness (or error) function are the error between target and actual outputs and the complexity of the ANN. Unlike the case in gradient-descent-based training algorithms, the fitness (or error) function does not have to be differentiable or even continuous since EA's do not depend on gradient information. Because EA's can treat large, complex, nondifferentiable, and multimodal spaces, which are the typical case in the real world, considerable research and application has been conducted on the evolution of connection weights .The aim is to find a near-optimal set of connection weights globally for an ANN with a fixed architecture using EA's. Comparisons between the evolutionary approach and conventional training algorithms, such as BP, will be made over XOR classification problem.

4.1 Evolution of Connection Weights Using GA

% initialization of population

1. sz = total weights in architecture;
2. ***For*** i = 1: popsize;
3. pop(i)=sz number of random number;

```
4.  End

% offspring population creation
5.  For j=1: popsize/2;
6.    pickup two parents randomly through uniform distribution;
7.    cp=cross over position defined by randomly pickup any active node position;
8.    To create offspring, exchange all incoming weights to selected nodes cp between parents;
9.        For each offspring;
10.,            place of mutation, mp = randomly selected active node;
11.               For all incoming weights w to selected node mp;
12.                     w=w+N (0, 1);
13.               End
14.       End
15. End

16. Offspring population, off_pop available;
17. npop= [pop; off_pop];

% Define fitness of each solution,
18. For i=1:2*popsize;
19.    wt=npop(i);
20.    apply wt to ANN architecture to get error value;
21.    define fitness as fit(i)=1/error;
22.  End

% Tournament selection
23. For  r =1:2*popsize;
24.  pick P number of Challengers randomly, where P = 10% of popsize;
25.  arrange the tournament w.r.t fitness between rth solution and selected P challengers.;
26.  define score of tournament for rth solution
27. End
28.  Arrange score of all solution in ascending order;
29.  sp=pick up the best half score position ;
30. select next generation solution as solution corresponding to position sp;

31. repeat the process from step 5 until terminating criteria does not satisfy
32. final solution=solution with maximum fitness in last generation.
```

5. EXPERIMENTAL SETUP

A fully interconnected feed forward architecture of size [2-2-1] / [2 3 1] designed .transfer function in the active node is taken as unimodel sigmoid function. Initial random weights are upgraded by gradient decent and genetic algorithm respectively. Various learning rates have applied to capture performance possibilities from gradient decent. To increase the learning and efficiency 'bias' in architecture and 'momentum' in learning have also included when learning given by gradient decent. Population size in GA taken as 20 and 10 independent trails have given to get the generalize behavior. Condition of terminating criteria is taken as fixed iteration and it is equal to 500 for GA. Because GA works with a population at time where as gradient decent takes only one solution in each iteration hence to nullify the effect , more number of iterations have given to gradient decent learning and it is taken as 20*500;

5.1 Performance Shown by Gradient Decent Learning

With the defined size of architecture, bias has applied with +1 input for hidden layer and output layer. Various learning rate taken from 0.1 to 0.9 with the increment of 0.1 along with momentum constant as 0.1.in the Fig(2) performance has shown for architecture size [2 2 1] where as in Fig(3) with size [2 3 1].Mean square error obtained after 10,000 iterations has shown in Table 1 and in Table 2.

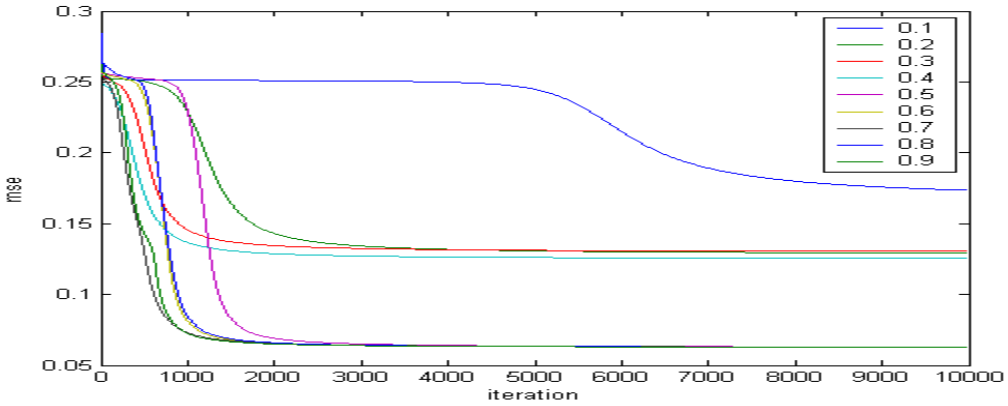

FIGURE 2: MSE performance with various learning rate .

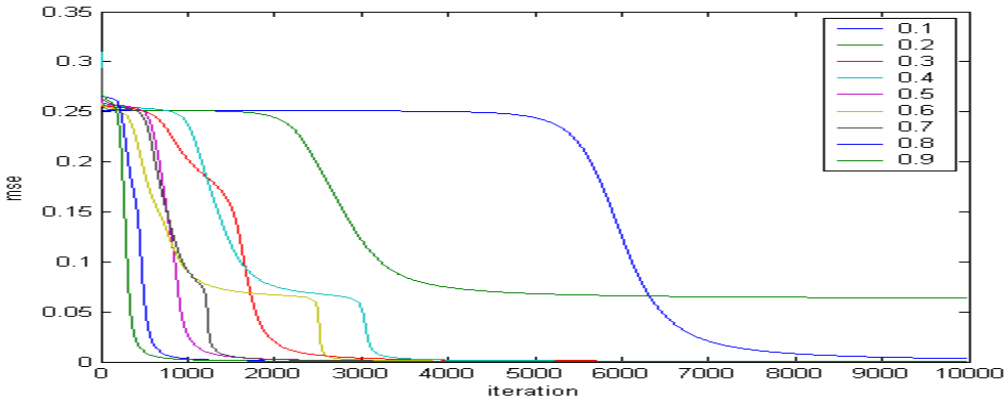

FIGURE 3: MSE performance with various learning rate

Learning rate	MSE([2 2 1])	MSE([2 3 1])
0.1	1.7352 e-001	3.2613 e-003
0.2	1.2920 e-001	6.3886 e-002
0.3	1.3042 e-001	4.4409 e-004
0.4	1.2546 e-001	3.7232 e-004
0.5	6.2927 e-001	2.6024 e-004
0.6	6.2825 e-001	2.3994 e-004
0.7	6.2915 e-001	2.0970 e-004
0.8	6.2868 e-001	1.3542 e-004
0.9	6.2822 e-001	1.2437 e-004

TABLE 1: performance shown by gradient decent

5.2 Performance Shown by GA Based Learning

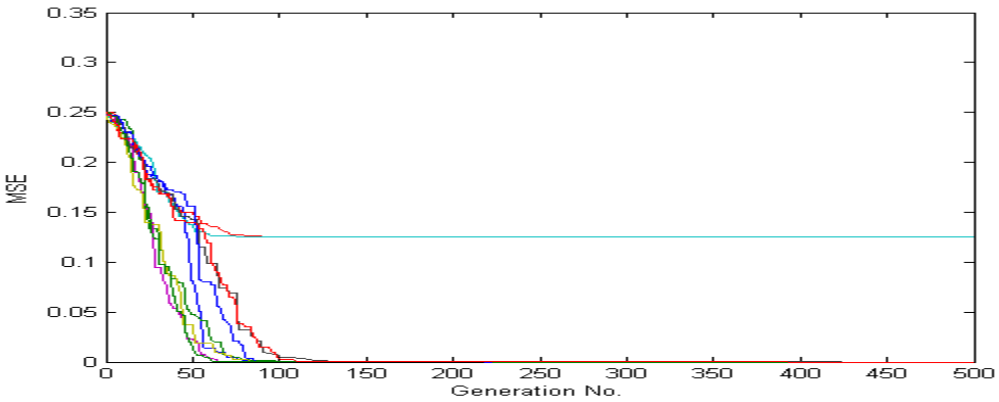

FIGURE 4: MSE performance by GA for different trails in [2 2 1]

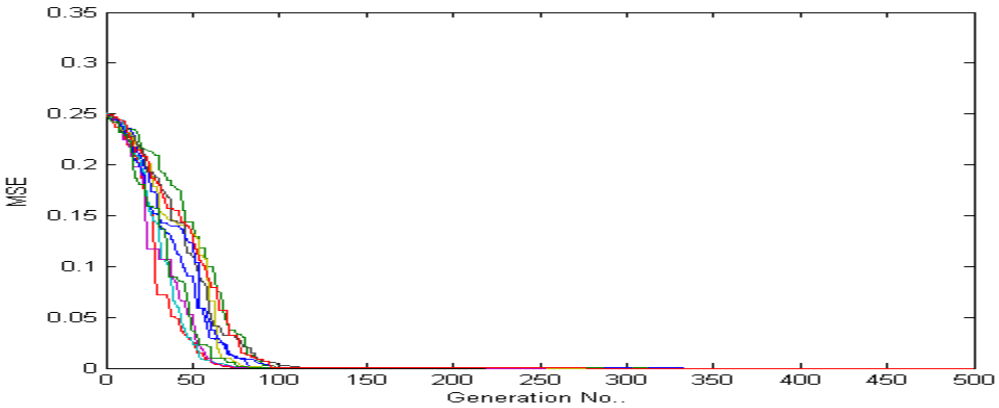

FIGURE 5: MSE performance by GA for different trails in [2 3 1]

Trail No.	MSE([2 2 1])	MSE([2 3 1])
1	8.8709 e-055	3.6112 e-028
2	2.5941 e-022	4.6357 e-031
3	***1.2500 e-001***	7.5042 e-032
4	***1.2500 e-001***	4.2375 e-037
5	3.2335 e-044	6.0432 e-049
6	2.5765 e-041	1.1681 e-035
7	9.6010 e-022	9.3357 e-032
8	3.5481 e-047	4.4852 e-030
9	3.9527 e-050	1.1725 e-033
10	6.3708 e-023	2.5171 e-036

TABLE 2: performance shown by GA for different trails

Results shown in Table1 indicate the difficulties associated with gradient decent based learning rule. Performance is very poor with the architecture size [2 2 1] for all learning rates. In fact learning failed for this case. This is indication of stuckness in local minima. For architecture [2 3 2] there is an improvement in reduction of mean square error, and with higher value of learning

rate equal to 0.9, best performance has obtained. Convergence characteristics for both cases have shown in Fig (1) and in Fig (2). convergence characteristics performance of developed form of GA for weight adjustment shown in Fig(4) and in Fig(5).in both cases it is very clear that very fast convergence with high reliability can be achieve by GA (except for trail number 3 and 4)as shown in Table 2.

6. CONCLUSION

Determination of optimal weights in ANN in the phase of learning has obtained by using the concept of genetic algorithm. Because of direct form realization in defining the solution of weights there is no extra means required to represent the solution in population. Proposed method of weights adjustment has compared with the gradient decent based learning and it has shown proposed method outperform at every level for XOR classification problem. Even with lesser number of hidden nodes where gradient decent method is completely fail for learning, proposed method has shown very respectable performance in terms of convergence as well as accuracy also. Defined solution of learning has generalized characteristics from application point of view and having simplicity in implementation.

7. REFERENCES

[1] X. Yao, “Evolution of connectionist networks,” in Preprints Int. Symp. AI, Reasoning & Creativity, Queensland, Australia, Griffith Univ., pp. 49–52. 1991.

[2] “A review of evolutionary artificial neural networks,” Int. J. Intell. Syst., vol. 8, no. 4, pp. 539–567, 1993.

[3] “Evolutionary artificial neural networks,” Int. J. Neural Syst., vol. 4, no. 3, pp. 203–222, 1993.

[4] T. Dartnall, Ed. Dordrecht, “The evolution of connectionist networks,” in Artificial Intelligence and Creativity, The Netherlands: Kluwer, pp. 233–243, 1994.

[5] A. Kent and J. G. Williams, “Evolutionary artificial neural networks,” in Encyclopedia of Computer Science and Technology, vol. 33, , Eds. New York: Marcel Dekker, pp. 137–170, 1995.

[6] G. E. Hinton, “Connectionist learning procedures,” Artificial Intell., vol. 40, no. 1–3, pp. 185–234, Sept. 1989

[7] Rumelhart D. E., Hinton G. E., Williams R. J. “Learning representations by back propagating errors”, .Nature, 323, 533-536, 1986.

[8] Rumclhart D. E., Hinton G. E., Williams R. J.: “ Learning errors” , Nature, Vol. 323, pp. 533-536, 1986.

[9] Wcrobs P. J,: ” Back-propagation: Past and Future”, Proc. Neural Networks, San Diego, CA, 1, 343-354, 1988.

[10] Wilamowski B. M,: “ Neural Network Architectures and Learning Algorithms: How not to be Frustrated with Neural Networks, IEEE Industrial Electronics Magazine “, Vol. 3 No. 4, pp. 56-63, Dec. 2009.

[11] R. S. Sutton, “Two problems with back-propagation and other steepest-descent learning procedures for networks,” in Proc. 8th Annual Conf. Cognitive Science Society. Hillsdale, NJ: Erlbaum, pp. 823–831, 1986

.

[12] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," Parallel Comput., vol. 14, no. 3, pp. 347–361, 1990

[13] J. Hertz, A. Krogh, and R. Palmer, Introduction to the Theory of Neural Computation. Reading, MA: Addison-Wesley, 1991.

[14] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," IEEE Signal Processing Mag., vol. 10, pp. 8–39, Jan. 1993.

[15] Y. Chauvin and D. E. Rumelhart, Eds., Back-propagation: Theory, Architectures, and Applications. Hillsdale, NJ: Erlbaum, 1995.

[16] J. Hertz, A. Krogh, and R. Palmer, Introduction to the Theory of Neural Computation. Reading, MA: Addison-Wesley, 1991.

[17] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," IEEE Signal Processing Mag., vol. 10, pp. 8–39, Jan. 1993.

[18] Y. Chauvin and D. E. Rumelhart, Eds., Backpropagation: Theory, Architectures, and Applications. Hillsdale, NJ: Erlbaum, 1995.

[19] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural Networks, vol. 6, no. 4, pp. 525–533, 1993.

[20] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," Neural Networks, vol. 3, no. 1, pp. 33–43, 1990.

[21] S. Knerr, L. Personnaz, and G. Dreyfus, "Handwritten digit recognition by neural networks with single-layer training," IEEE Trans. Neural Networks, vol. 3, pp. 962–968, Nov. 1992.

[22] S. S. Fels and G. E. Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," IEEE Trans. Neural Networks, vol. 4, pp. 2–8, Jan. 1993.

[23] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," Parallel Comput., vol. 14, no. 3, pp. 347–361, 1990.

[24] D. Whitley, "The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best," in Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, pp. 116–121, 1989.

[25] P. Zhang, Y. Sankai, and M. Ohta, "Hybrid adaptive learning control of nonlinear system," in Proc. 1995 American Control Conf. Part 4 (of 6), pp. 2744–2748, 1995.