Current Development in Artificial Intelligence.
ISSN 0976-5832 Volume 3, Number 1 (2012), pp. 9-21

http://www.irphouse.com

Evolutionary ANN Learning Algorithm on Benchmark and Real Time Dataset Classification Problems

[1]G.V.R. Sagar and [2]S. Venkata Chalam

[1]Assoc. Professor, G.P.R. Engg. College, Kurnool, A.P., India
E-mail: nusagar@gmail.com
[2]Principal, A.C.E. Engg. College, Ghatkesar, A.P., India
E-mail: sv_chalam2003@yahoo.com

Abstract

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. Evolutionary algorithms operate on a population of potential solutions applying the principal of survival of the fittest to produces better and better approximations to solutions. 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, Parity2[26], Parity4 (XOR with 4 inputs), and two real world data problems like heart and Pima-India-diabetes, 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.

Introduction

An Artificial Neural Network (ANN) 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. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. 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

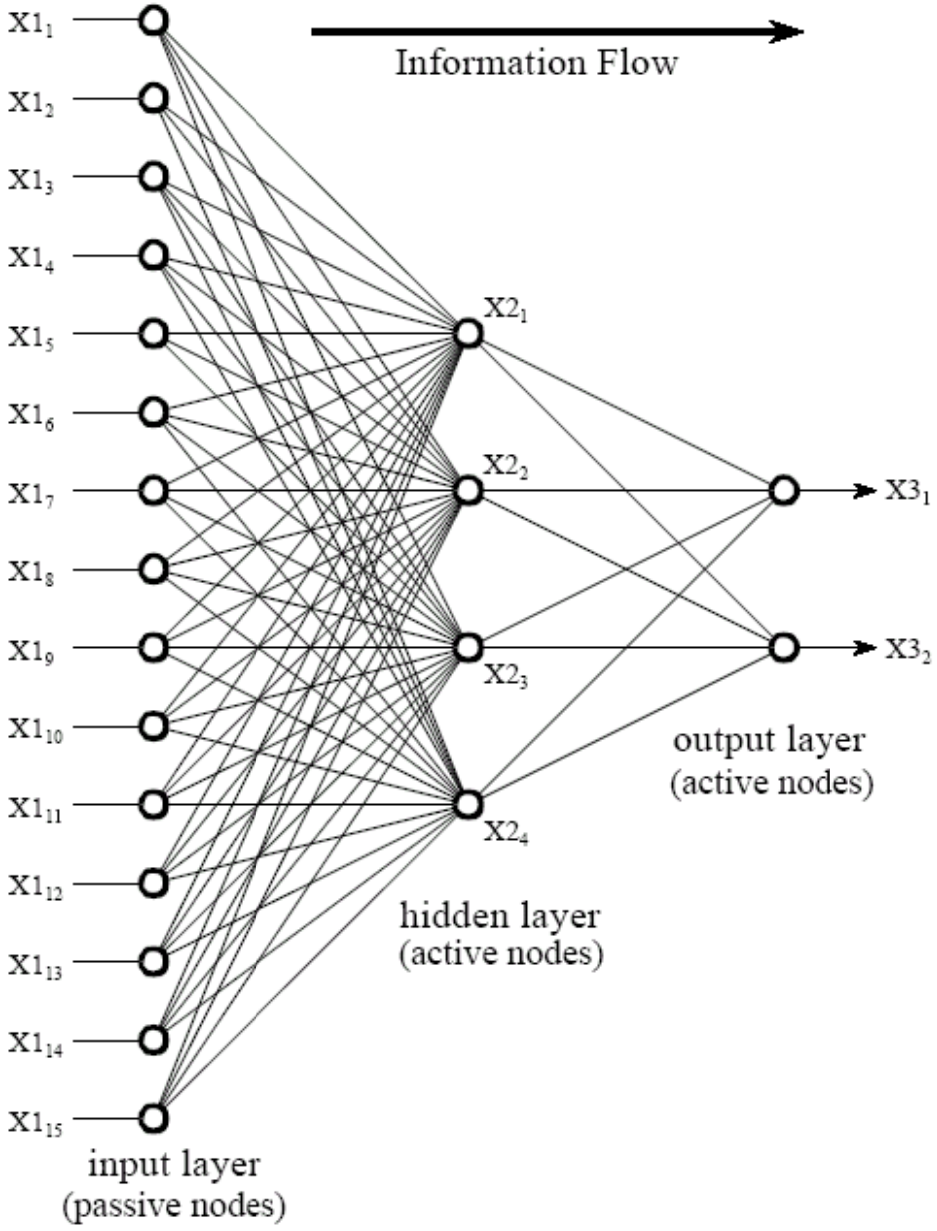

Figure 1: Feed forward architecture

Learning in artificial neural systems 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. 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. During the learning process global information may be required. 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 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.

Gradient descent Back-Propagation learning

The supervised learning Paradigm 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. In the previous paper[x] described the benchmark problem of parity2 (xor with two inputs) for different learning factors for better convergence. 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 initial weights influence the net reaches a global minima or only local mina of error and if so how rapidly it converges. To get the best result the initial weights are random numbers set between -0.5 and +0.5 or between -1 and +1. The initialization of weights (bias) can be done randomly. A high learning rate leads to rapid learning but weights are oscillate, while a lower learning rate leads to slower learning. A simple way is to increase the learning rate in order to improve performance and to decrease the learning rate in order to worsen the performance, or to double the learning rate until the error value worsens. The learning in the Back-Propagation uses the batch learning in which the weights are updated only after the entire set of training network has been presented to the network. Thus the weight update is only performed after every iteration. A Feed-forward Back-propagation network with one hidden layer of N nodes is used to train the benchmark problems like Parity2 [26] and parity4. Parity4 has four inputs and one output the truth table is given in the table1.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. A generalized number of hidden nodes are (2n+1), where n is the number of inputs.

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. 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].

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. The fitness function is shown in Fig1(b).

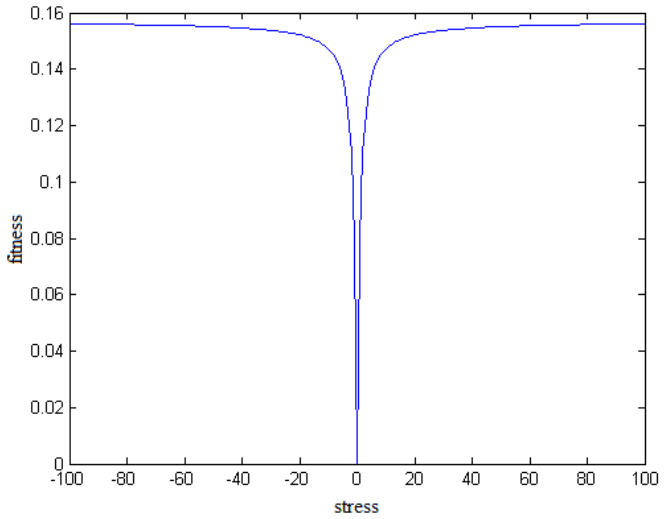

Figure 1(b): fitness function

Evolutionary 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 be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information. .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 Parity4 (XOR), two real time data set like heart deices, Pima-India-Diabetes data sets classification problem.

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.

Experimental setup

A feed forward architecture with fully interconnected network 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 for parity4 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 1000 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*1000 and run it for different network sizes. Population size for real time data classifiers like heart deices and pima-india-diabetes in GA taken as 100 and 200 and run it for different network sizes.

Performance shown by gradient decent learning

With the defined size of architecture, bias has applied with +1 input for hidden layer and output layer. For parity4 (xor with 4 inputs and 1 output):Various learning rate taken from 0.1 to 0.9 with the increment of 0.1 along with momentum constant as 0.1.shown in the Fig(2.0) and various momentum rate taken from 0.1 to 0.9 with the increment of 0.1 along with learning constant as 0.9 in Fig(3) for a network size of [4 4 1]. The general criteria for maximum number of hidden nodes in gradient algorithm is (2n+1), where n is the number of inputs. The performance has also shown in Fig(4) for architecture size [4 9 1] with size [4 5 1]. Mean square error obtained after 20,000 iterations has shown in Table 1.

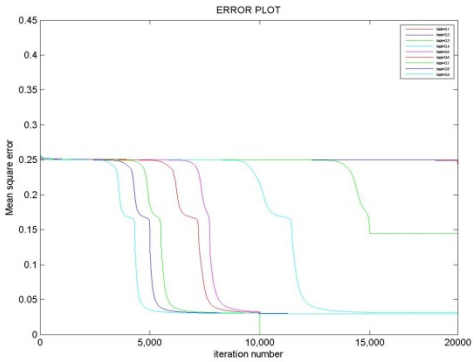

Figure 2(a): with network size[4 4 1]

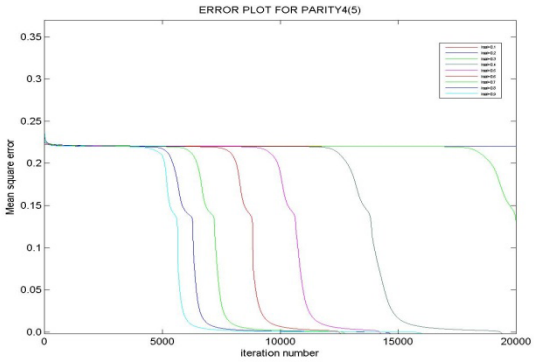

Figure 2(b): with network size[4 5 1]

Figure 2: Results of Gradient based learning with different learning rates (0.1 -0.9) keeping momentum rate at 0.1 on different network sizes

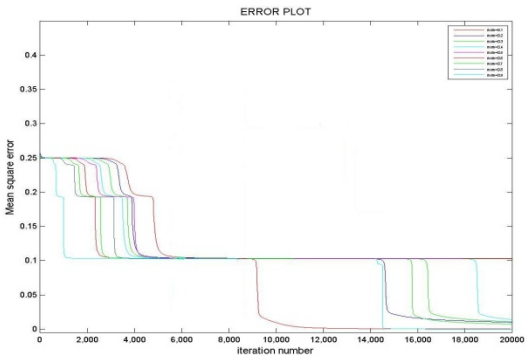

Figure 3(a): with network size[4 4 1]

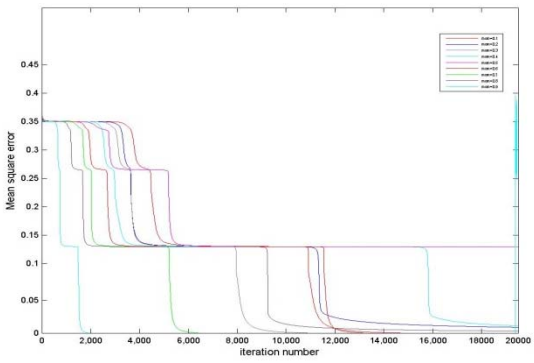

Figure 3(b): with network size[4 5 1]

Figure 3: Results of Gradient based learning with different momentum rates keeping learning rate at 0.9 on different network sizes

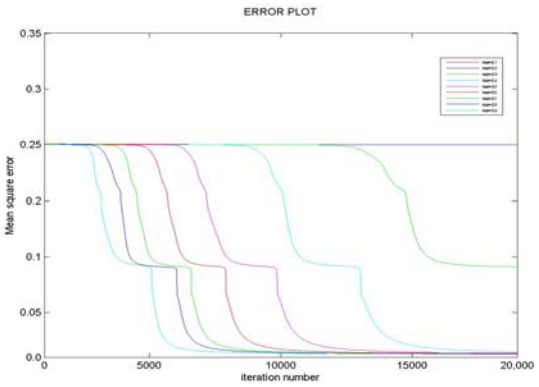

Figure 4(a): Different learning rate (0.1-0.9) with momentum is constant at 0.1

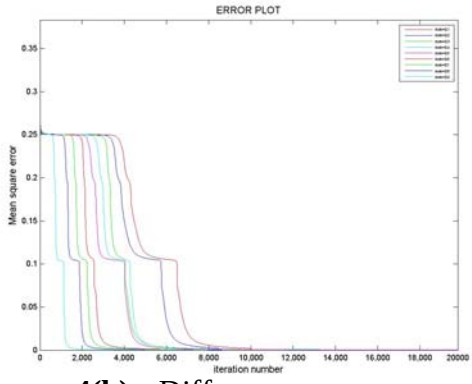

Figure 4(b): Different momentum rate (0.1-0.9) with learning is constant at 0.1

Figure 4: Results of Gradient based learning with network size[4 9 1]

Table 1: performance shown by gradient decent

Learning rate	MSE([4 4 1])	MSE([4 5 1])	MSE([4 9 1])
0.1	5.0013e-001	5.0017e-001	5.0023e-001
0.2	5.0006e-001	5.0008e-001	5.0005e-001
0.3	3.9440e-001	4.1291e-001	2.8197e-001
0.4	2.8178e-001	1.8251e-001	1.2974e-001
0.5	2.8247e-001	1.3025e-001	1.2729e-001
0.6	2.8148e-001	1.2823e-001	1.2650e-001
0.7	2.8047e-001	**1.2770e-001**	1.2612e-001
0.8	**2.7983e**-001	1.2828e-001	1.2594e-001
0.9	2.7976e-001	1.2886e-001	**5.4989e-003**

Results shown in Table1 indicate the difficulties associated with gradient decent based learning rule. Performance is very poor with the architecture size [4 4 1] for all learning rates. In fact learning failed for this case. This is indication of stuckness in local minima. For architecture [4 5 1] 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 at momentum rate equal to 0.1 with early convergence as shown in fig3(b). Convergence characteristics for both cases have shown in Fig(2) and Fig(3). The performance of gradient algorithm is further improved using the network size [4 9 1], with higher value of learning rate(0.9) and lower momentum factor of 0.1 the convergence is very fast and best reduced mean square error values compare to the network size [4 5 1] as shown in Fig4(b).

Performance shown by GA based learning

With the defined size of network architectures bias has applied with +1 input for hidden layer and output layer. For parity4 the developed form of GA results for ten different runs are shown in the Fig(5).Mean square error obtained after 1000 iterations has shown in in Table 2.

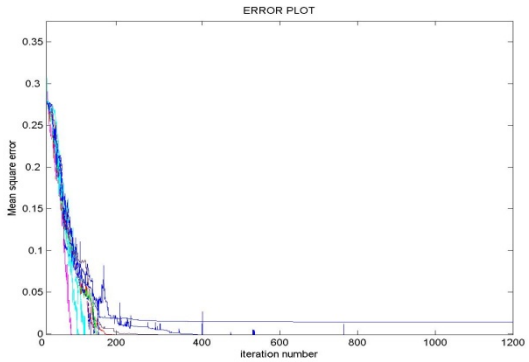

Figure 5(a): network size of [4 4 1]

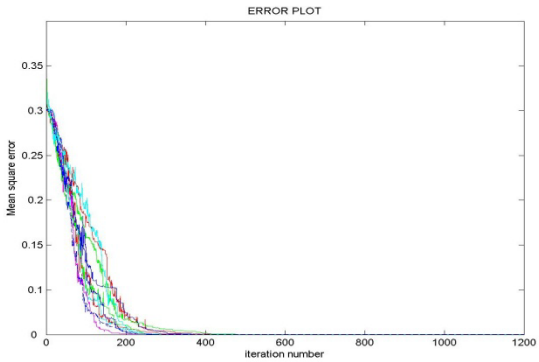

Figure 5(a): network size of [4 5 1]

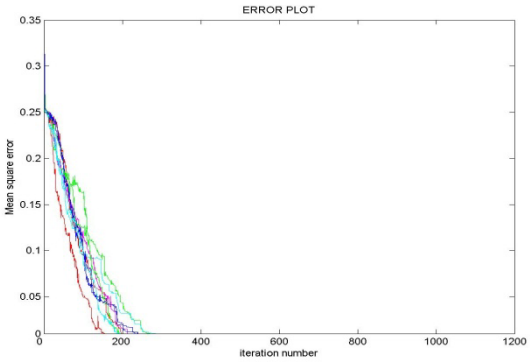

Figure 5(c): network size of [4 9 1]

Figure 5: MSE performance by GA for different trails

Table 2: performance shown by GA for different trails

Trail No.	MSE([4 4 1])	MSE([4 5 1])	MSE([4 9 1])
1	***9.0084e-003***	***4.6115e-006***	7.0489e-015
2	2.1219e-026	9.4020e-028	5.0245e-037
3	2.0416e-014	9.6960e-032	7.8364e-029
4	***1.3406e-003***	6.0631e-037	8.8598e-033
5	3.2323e-022	6.9969e-049	3.6283e-038
6	4.6980e-044	8.3418e-033	9.0794e-049
7	2.2361e-047	7.7779e-028	7.3994e-047
8	2.3923e-022	7.4838e-018	9.2225e-032
9	1.8086e-041	8.3266e-030	8.3001e-035
10	7.1106e-055	7.3821e-047	6.4088e-049

Convergence characteristics performance of developed form of GA for weight adjustment shown in Fig(5). For all the network cases it is very clear that very fast convergence with high reliability can be achieve by GA (expect for the two trail numbers one and four in [4 4 1] network and trail number one in [4 5 1] network) as shown in Table 2. But when compared to parity2 (xor2) [26] the convergence for parity4 takes more iterations as shown in Fig.5.

Benchmark Real data classification Problems

Using the feed-forward defined size of architecture, bias has applied with +1 input for hidden layer and output layer. For real data classification data problems, two types data sets has considered i) SPECT Heart data set ii) Pima-India-Diabetes data set problems. The data available for these problems had been uploaded into the UCI repository in 1990.

Pima Indians Diabetes

The Pima Indians Diabetes database data refers to a medical problem, in which the diagnosis was carried out on several patients to investigate whether a patient shows signs of diabetes. This dataset composed of 8 attributes plus a binary class value to show the signs of diabetes which corresponds to the target classification value and includes 768 instances shown in Table3.

Table 3: Set of features considered for PIMA Indian Diabetes problem.

Number	Attribute
1	Number of times pregnant
2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	Diastolic blood pressure(mm Hg)
4	Triceps skin fold thickness(mm)
5	2-Hour serum insulin (mm U/ml)
6	Body mass indesx, with weight expressed in Kg and height expressed in m(Kg/m2)
7	Diabetes pedigree function
8	Age (years)
9	Class variable (0 or 1)

Results of this data sets and different parameters are shown in Table4 and Pima India diabet achieved good percentage of accuracy and consistency.

SPECT Heart Data

For Single Proton Emission Computed Tomography (SPECT) Heart deceases data set, only 13 attributes are used as input parameters to classify the problem and a total of 267 instances. The target value has stored at 14^{th} parameter in the data set. These data sets are normalized before applying to the network fixed feed-forward architecture, one hidden layer of N number of nodes and one output node. The Table4 summaries the parameters and results are achieved up to good percentage of accuracy and consistency.

Table 4: performance shown by GA for different real time datasets.

Parameter	**Pima India Diabet**		**SPECT Heart Data**	
Number of inputs	09	09	13	13
Number of outputs	01	01	01	01
Number of hidden nodes	**02**	**03**	**02**	**03**
Max. Iteration taken	**102**	**61**	**120**	**77**
Number of Training patterns	500	500	200	200
Number of Test patterns	268	268	67	67
Percentage of Accuracy	**81.5**	**81.5**	**88.6**	**88.6**

Performance of developed form of GA for weight adjustment real world classification problems are shown in Table4, which shows the results of 2 and 3 hidden nodes with a population of 50. Pima India Diabet problem takes 500 data sets for Training and 268 data sets for test pattern. This classification achieved the very good consistence with minimum mean square error of **7.2346e-002** for different runs and the average percentage of accuracy is **81.5**. Similarly for SPECT Heart Data classification problem 200 datasets are used for training and 67datasets are used as test pattern with the same 2 and 3 hidden nodes networks with a population of 50. This problem achieved the excellent consistency with minimum mean square error of **4.1722e-002** for different runs and the average percentage of accuracy is **88.6.**

Comparison of the literature

Table 5: Shows the caparison of accuracy of the literature

Method	**Pima India Diabet**	**SPECT Heart Data**
Neuro genetic approach	75.5%	--
ADAP Algorithm	76%	--
EPNet	77.6%	--
ANN Development by means of GP with Graph Codification	--	81.11
Proposed Method	**81.5%**	**88.6**

In the above Table 5 shows the developed evolutionary genetic algorithm has given the excellent performance on accuracy and less mean square error compare to the ADAP, Neuro genetic approach and EP Net algorithm.

Conclusion

Determination of optimal weights in ANN in the phase of learning has obtained by using the concept of evolutionary 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 first compared with the gradient decent based learning and it has shown proposed method outperform at every level for parity4 classification problem. Even with lesser number of hidden nodes where gradient decent method is completely fail for learning, and it has been shown excellent convergence as well as accuracy also. Second, it has been applied on real time data sets classification problems has shown, the proposed method outperform when compared with some literature works. Defined solution of learning has generalized characteristics from application point of view and having simplicity in implementation.

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.

[26] G.V.R.Sagar, Dr. S. Venkatachalam, & Mannoj kumar sing” Evolutionary Algorithm for Optimal Connection Weights in Artificial Neural Networks” in International Journal of engineering(IJE), vol:5, issue:5, pp.333-340, 2011.