A Hybrid Advanced PSO-Neural Network System

Talha Ali Khan, Khawaja Zain-Ul-Abideen, Sai Ho Ling

Abstract— **In this paper, a combination of Advanced Particle Swarm Optimization and Neural Network are presented to compensate the drawbacks of both the techniques and utilize the strong attributes to form a hybrid system called Hybrid Advance Particle Swarm Optimization-Neural Network System (HAPSO-NNS). APSO is used for the training of the neural network. In the initial phases of the search, PSO has swift convergence for global optimum, but later it suffers from slow convergence around the global optimum position. On the contrary, the gradient method attains prior to convergence around the global optimum point, therefore, attaining better accuracy in terms of convergence. This paper elucidates the usage of APSO applied to feedforward neural network to improve the classification accuracy of the network and also decreases the network training time.**

I. Introduction

A neural network is considered as a set of systems where the neurons adjust their weights and biases according to the output of the network. The number of nodes in the input and output layer of the neural network is fixed according to the application however the number of nodes in the hidden layer are subject to variation. This variation in the number of nodes of the hidden layer is to get the optimum results based on hit and trial [1]. The learning process of neural networks where there is an update in the weights and biases continues until the change in the weights and biases are too small to make a significant difference [2]. The neural network has a good capability of faster convergence however, the neural network sometimes gets trapped in the local optima [3].

Swarm intelligence is a term inspired by the social behavior of the animals where they try to find the shortest path between their habitat and the food. In the case of a flock of birds, every group member gets its neighbors information and the decision is made using the current neighbor's experience as well as the previous experience [4]. Particle Swarm Optimization Algorithm is a smart computational technique based upon the methodology that is not predominantly affected by the size and nonlinear nature of the problem and can simply be converged to an optimum in most of the problems where other methods are unsuccessful to give an optimal solution [5].

With the passage of time, different algorithms have been proposed for optimizing the neural networks these include genetic algorithms, gravitational search algorithms but results obtained are not of high accuracy [6]. This is because by using these algorithms the neural network algorithm is trapped at the local minima and unable to get out of it. Therefore, bio-inspired algorithms are used to train the neural network. These algorithms have the capability to solve nonlinear and multimodal equations, therefore, the training of neural networks using swarm intelligence resulted in better optimum finding capability and reduction in a number of iterations required for training. Several metaheuristic algorithms have been proposed for training a neural network based in local search, population, and other search algorithms. Some bio-inspired evolutionary algorithms have also been proposed to train a neural network. Most of the research done until this point has focused on the weights of neural connections between different layers. More research needs to be done on the neural functions that transform the input of the neural network to output [7].

Therefore, to resolve the problem of these small setbacks, the hybrid of particle swarm optimization and feedforward neural network is presented in this paper. This hybrid system results in better optimum finding capability and the time to train the data is also reduced. As a result, there is a reduction in the number of training cycles to get to a standard mean square error for the hybrid system compared to the simple backpropagation neural network. It is an important challenge in the field of biomedical engineering where the task to correctly classify the iris type has not yet been resolved fully. Therefore, to validate the performance of the system Iris classification problem is used. Four features of IRIS namely petal length, sepal length, petal width, sepal length, and sepal width are used.

II. ADVANCED PARTICLE SWARM OPTIMIZATION

PSO is a simple concept taken from nature such as a problem-solving capability of social animals like a swarm of fish or a flock of birds finding for food. This concept was first proposed by Jing Wang and Professor Gerardo Beni in 1989. This idea was further developed by Dr. Eberhart and Dr. Kennedy in 1995. This falls under the domain of swarm intelligence where the group of ants in the ant colony trying to find the smallest pathway between their habitat and their food. One point of emphasis here is that they all find the optimum point which is their food [8]. In this paper, the initialization method for APSO is similar to the standard PSO using the same parameters. The main modification is in the velocity equation, which is also the significant measure of the PSO algorithm. The last term that is additional in the velocity equation of the PSO is used to reduce the positions of the particles through the iterations so the velocity will be improved and the algorithm will be reached to the optimal solution more rapidly [9]. In addition, the inertia weight is also used as a factor to the particle location. The velocity and position equations will be

$$v_{id} = wv_{id} + c_1R_1(p_{id} - x_{id}) + c_2R_2(p_{gd} - x_{id}) + w\left(\frac{c_1}{c_2}\right)(p_{id} - p_{gd}) \quad (1)$$

where R_1, R_2 are two random numbers with uniform distribution in the range of [0-1], x_{id} is the current position of the particle, v_{id} is the velocity of the particle, p_{id} is the local position of the particle, p_{gd} is the global best position of the particle, k is the iteration number, w is inertial weight and c_1,

T.A.Khan, Khawaja.Z.A, S.H.Ling, are with School of Biomedical Engineering, University of Technology Sydney, Ultimo, Australia.

c_2 are two positive acceleration constants numbers. The generation of random numbers makes sure that if the set of particles are stuck in between the local minima, these can push the particles out of that to a better optimum point. The new position of the particle can be updated using the position update equation

$$x_{id}(k+1) = wx_{id}(k) + v_{id} \quad (2)$$

(2) Illustrates the position of the particle d in the i^{th} iteration. The following relation is being used for inertial weight control:

$$w = w_{max} - \left(\frac{w_{max}-w_{min}}{FE}\right) * i \quad (3)$$

In Eq (3) [10], , FE is the Function Evaluation and i is the current function evaluation. The value of w_{max} and w_{min} has been optimized to achieve the best results. Constriction factor has also been introduced in this paper as a modification made by [11]. The constriction factor is calculated as follows when $\boldsymbol{\Phi}$ is set at 4.1;

$$chi = \frac{2}{\Phi - 2 + \sqrt{\Phi^2 - 4*\Phi}} \quad (4)$$

This constriction factor is used to modify inertial weight. The following control is incorporated in the inertial weight.

$$w = chi * \left(0.0005 + w * \left(\frac{FE-(i-30)}{FE}\right)\right) \quad (5)$$

A maximum number of function evaluation is used as the stopping criteria. In this paper, the value of $\boldsymbol{\Phi}$ is set at 4.1.

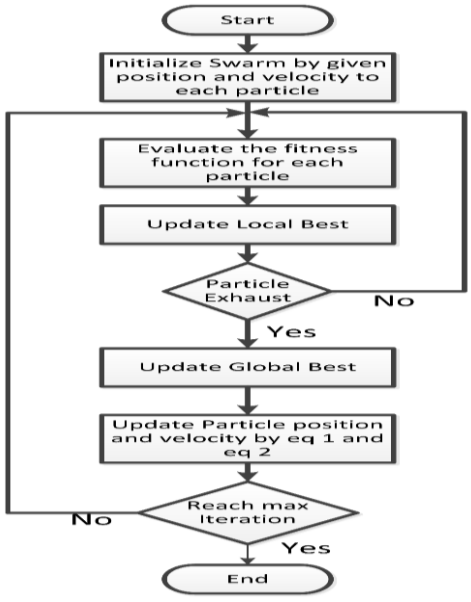


Fig 1. Shows the flowchart of APSO

III. HYBRID APSO-NEURAL NETWORK SYSTEM

APSO is used to train the neural network and each particle's position in a swarm is encoded into weights and biases for the current loop. The dimensions of each particle are the weights of the neural network. The particle moves within the D-dimensional space searching for the global optimum. In each epoch, the particle searches for the global optimum reducing the mean square error. Each epoch updates the position and velocity of the particle. New particle position represents the new sets of weights and biases. This loop is iterative unless the particle with the lowest mean square error is achieved. This training continues unless the termination criteria for the lowest mean square error is reached. The same sets of weights and biases are then used to test the network with the test patterns. Figure 2 shows a feedforward neural network that uses PSO algorithm.

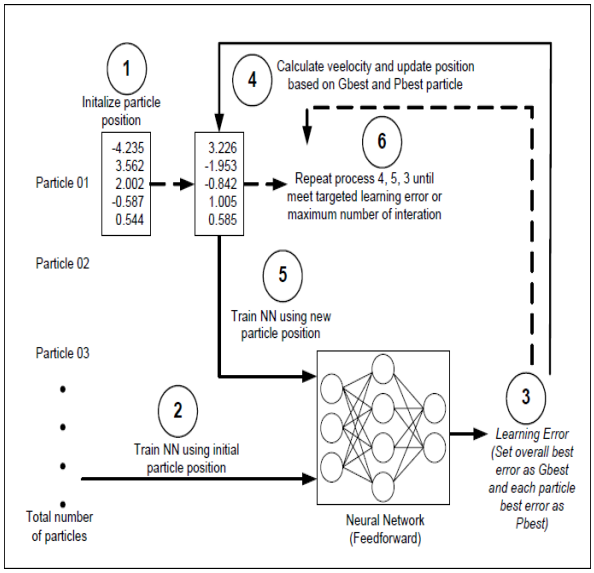


Fig 2. PSO-FFNN learning process

In fig.2 there is no back propagation of the error from the neural network. New P_{best} and G_{best} are set by the learning error produced by the feed-forward neural network. The new velocity is obtained by applying P_{best} and G_{best} to equation (1). The new D-dimensional velocity vector is added to the position of the previous iteration using equation (2) to produce a new particle position.

The new particle position is used to set the new weights and biases for the feedforward neural network. Fig.3 shows the complete algorithm implementation

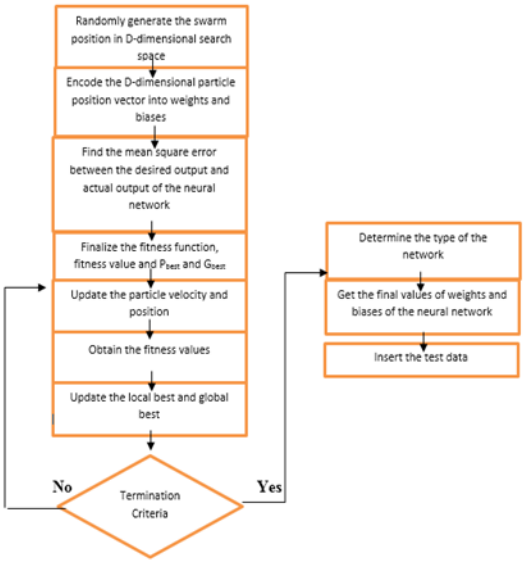


Fig 3. Flow chart for PSO-Feed forward neural network

The steps explain how the PSO-feedforward neural network system works

1st Step. Determine the number of inputs, hidden and output layers of the neural network

2nd Step. Randomly generate the particles within the D-dimensional search space

3rd Step. Encode the D-dimensional initial position of the particle into the weights and biases of neural networks

4th Step. Initialize the parameters of PSO

5th Step. Input the training data

6th Step. Calculate the output of the neural network, compare it with the desired output and find the fitness value

7th Step. Use the output from the fitness function to get local best and the global best position

8th Step. Update the particle's velocity and position vector

9th Step. Check for the maximum number of iterations and termination criteria. If reached, go to next step else go back to step no 6

10th Step. The termination criteria give the optimal weights and biases of the neural network.

11th Step. Input the test data to obtain the classification accuracy of the hybrid PSO-Feed forward neural network.

IV. IRIS CLASSIFICATION

This training data is used in the algorithm to obtain the optimum sets of weights and biases. This trained neural network with optimum weights and biases is used on the test data and the classification accuracy is obtained. The IRIS classification problem has been widely used in the field of biomedical engineering. The problem is to distinguish between the three classes of iris namely Versicolor, Setosa and Virginia. This classification has to be done using the four features of the IRIS namely petal length, sepal length, petal width, and sepal width.

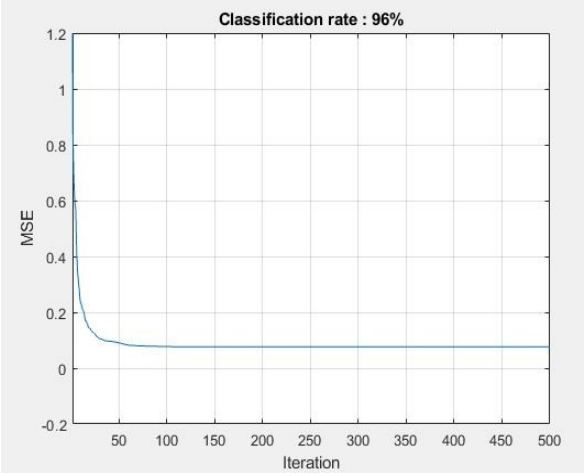
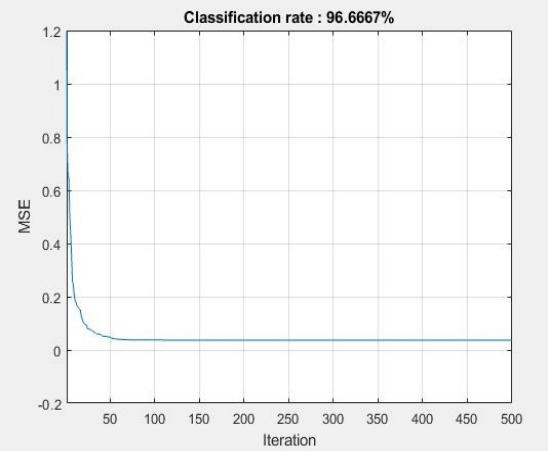
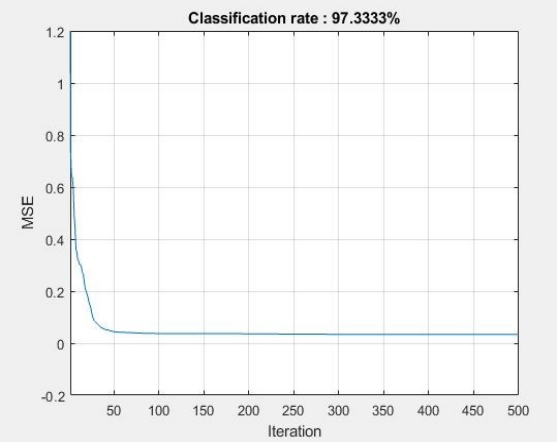


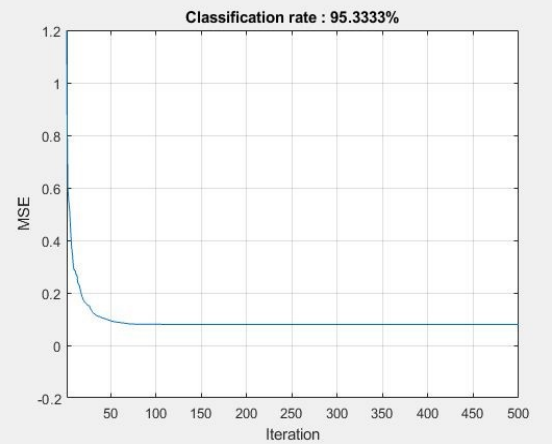
Fig 4. (A)



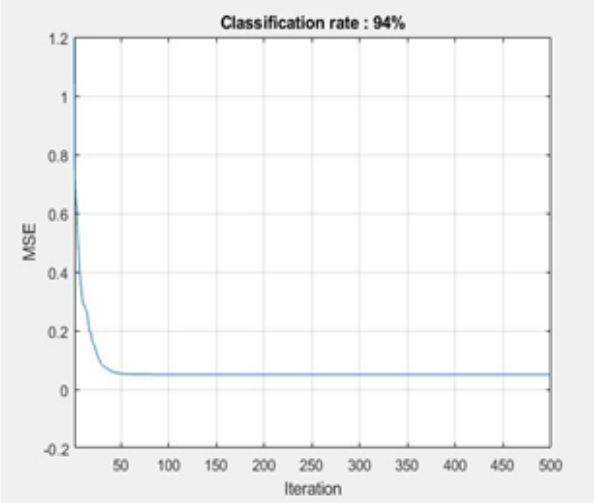
(B)



(C)



(D)



(E)

As a result, the neural network will have the structure with 4 input neurons, 3 output neurons and the number of hidden neurons will be determined by the hit and trial method. This neural network structure can be represented as 4-S-3 where S represents the number of neurons in the hidden layer. This number of hidden layers (S) varies from 7 to 15 in this task to look at which number gets the highest classification accuracy. The graphs below represent the highest outputs achieved by varying the hidden number of neurons

Fig 4 shows the convergence curves of PSO-Feed forward neural network based on MSE for all training samples for iris classification problem. (A), (B), (C), (D), (E) are convergence curves for S= 7, 9, 11, 13, and 15, respectively. Fig 5. shows how the values of classification vary with the number of hidden layers of the feed-forward neural network. It shows the variation of the classification accuracy against the number of neurons in the hidden layer. The highest accuracy is achieved for 11 neurons.

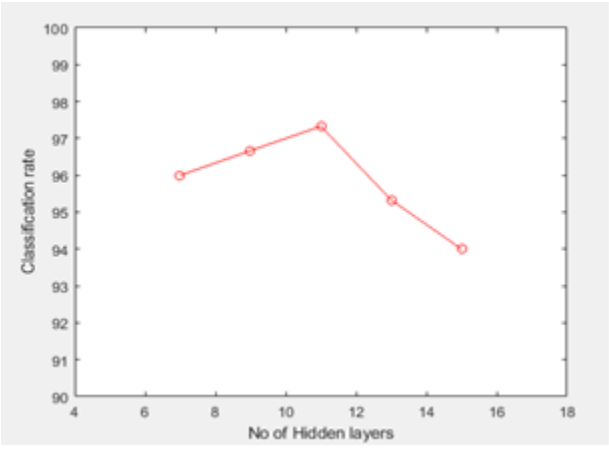


Fig 5: Hidden layer neurons vs classification rate

Table 1 shows the values of the classification shown above by the individual graphs of the Feedforward neural network.

TABLE 1. *Classification accuracy against the number of neurons*

Number of neurons	***Classification accuracy (%)***
7	96.0
9	96.67
11	97.33
13	95.33
15	94.0

V. PSO-FEEDFORWARD NEURAL NETWORK VS NEURAL NETWORK

To elaborate on the performance of the hybrid APSO-feedforward neural network comparison is made against the Backpropagation. The hybrid algorithm's performance is better than the performance of the back propagation neural network. The neural network has a fixed classification accuracy whereas the classification accuracy of the hybrid algorithm varies between certain percentages. This variation is because the initial generation of particles affects the overall accuracy. Figure 6 shows the comparison of the classification accuracy of the back propagation neural network and the classification accuracy of the hybrid PSO-Feed forward neural network.

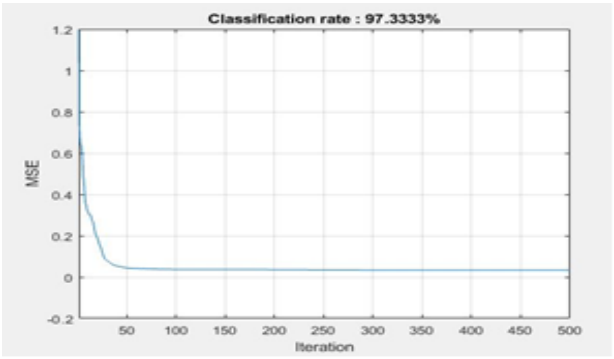
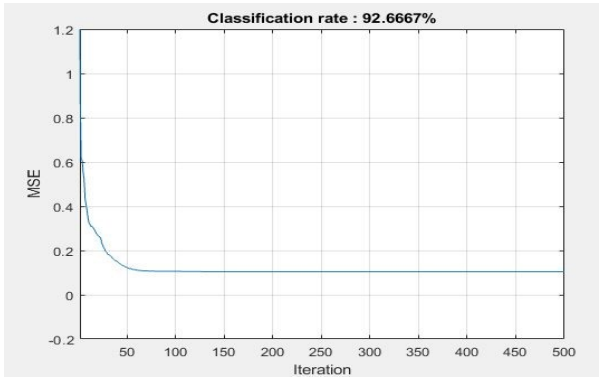


Fig 6 (A)



(B)

Fig 6(A) and 6(B) show that the overall classification of the PSO-Feedforward neural network is better than the

classification accuracy of the back propagation neural network. The convergence of the hybrid network is also faster in around 25 iterations around the global optimum compared to the back propagation neural network where it takes around 60 iterations to converge around the global optimum. The algorithm has been run for 500 iterations.

VI. CONCLUSION

APSO is used in training the neural network. The back-propagation neural network that is commonly used, in this paper a novel approach of feeding the position of the particle to the feed-forward neural network is presented. The output of the neural network is fed to the fitness function and fitness values are used to update the particle's local best and the global best position. This algorithm achieves the potential to achieve better results than the simple neural network as demonstrated by the classification accuracy of the neural network and the hybrid algorithm. It can also be seen that the convergence of the hybrid algorithm is faster than the back propagation neural network where it takes around 25 iterations in the hybrid algorithm whereas it takes around 60 iterations in the back propagation neural network to converge to the optimum point.

VII. REFERENCES

[1] D. Lyridis, P. Zacharioudakis, P. Mitrou, and A. Mylonas, *Forecasting Tanker Market Using Artificial Neural Networks*. 2004, pp. 93-108.

[2] G. Bo, H. Hejuan, H. Hui, and R. Yan, *Hybrid PSO-BP neural network approach for wind power forecasting*. 2017, pp. 211-222.

[3] Z. Yuan, L. Huang, D. Hu, and B. Liu, "Convergence of Nonautonomous Cohen–Grossberg-Type Neural Networks With Variable Delays," *IEEE Transactions on Neural Networks,* vol. 19, no. 1, pp. 140-147, 2008.

[4] D. Pal, P. Verma, D. Gautam, and P. Indait, "Improved optimization technique using hybrid ACO-PSO," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, 2016, pp. 277-282.

[5] H. Xue, Y. Bai, H. Hu, T. Xu, and H. Liang, "A Novel Hybrid Model Based on TVIW-PSO-GSA Algorithm and Support Vector Machine for Classification Problems," *IEEE Access,* vol. 7, pp. 27789-27801, 2019.

[6] L. Gao, C. Zhou, H.-B. Gao, and Y. Shi, *Combining Particle Swarm Optimization and Neural Network for Diagnosis of Unexplained Syncope*. 2006, pp. 174-181.

[7] M. Farahnakian, M. Reza Razfar, M. Moghri, and M. Asadnia, *The selection of milling parameters by the PSO-based neural network modeling method*. 2011, pp. 49-60.

[8] Y. Zhang, W. H. Li, Y. Meng, Z. J. Tan, and Y. J. Pang, *Fast image mosaics algorithm using particle swarm optimization*. 2006.

[9] T. Ali Khan, S. Ho Ling, and A. Sanagavarapu Mohan, *Advanced Particle Swarm Optimization Algorithm with Improved Velocity Update Strategy*. 2018, pp. 3944-3949.

[10] M. Alhussein and S. I. Haider, "Improved Particle Swarm Optimization Based on Velocity Clamping and Particle Penalization," in *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, 2015, pp. 61-64.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 69-73.