Hybrid Algorithm with Variants for Feed Forward Neural Network

Thinakaran Kandasamy[1] and Rajasekar Rajendran[2]
[1]Sri Venkateswara College of Engineering and Technology, Anna University, India
[2]Excel Engineering College, Anna University, India

Abstract: *Levenberg-Marquardt back-propagation algorithm, as a Feed forward Neural Network (FNN) training method, has some limitations associated with over fitting and local optimum problems. Also Levenberg-Marquardt back-propagation algorithm is opted only for small network. This research uses hybrid evolutionary algorithm based on Particle Swarm Optimization* **(***PSO) in FNN training. This algorithm includes a number of components that gives advantage in the experimental study. Variants such as size of the swarm, acceleration coefficients, coefficient constriction factor and velocity of the swarm are proposed to improve convergence speed as well as to improve accuracy. The integration of components in different ways in hybrid algorithm produces effective optimization of back propagation algorithm. Also, this hybrid evolutionary algorithm based on PSO can be used for complex neural network structure*.

Keywords: *Back propagation, hybrid algorithm, levenberg-marquardt, Particle swarm optimization, variants of PSO algorithm.*

Received August 31, 2014; accepted April 12, 2015

1. Introduction

The aim of this paper is to show that integration of components in different ways will produce effective optimization of algorithm. Stand alone algorithm are used as components of hybrid Particle Swarm Optimization (PSO) algorithm [7]. Our comparison focuses on updating particles velocity and different population size. This algorithm has been applied in non linear function optimization. Similar to the Genetic Algorithm (GA), the PSO algorithm is an optimization tool based on population and the system is initialized with a population of random solutions and is searched for optima by updating the generations.

The performance of the PSO algorithm can be improved through the inertia weight [3, 8, 10]. The PSO search is performed by tracing P_b best position in its history. PSO have tested with different communication of circles, stars and randomly assigned edges [12]. In "speciation based PSO" dynamical change of size of particle used to increase the convergence speed [14]. The adaptive tuning of parameters of PSO can improve the convergence speed [19]. The PSO algorithm use a time varying population topology to increase the convergence speed [6]. The Fully Informed PSOs (FIP), highly connected topology, has quick convergent behavior using a fixed number of function evaluations [17]. The TRIBES PSO [5] is also an adaptive PSO algorithm which can adaptively tune the number of particles. In HRCGA algorithm different population size was suggested to get convergence speed [18]. To achieve the best performance, it is necessary to tune the selection of the mutation. The Back propagation algorithm will easily get trapped in local minima then itidentify the local optima value [2]. To improve the performance of the Back propagation algorithm, the people concentrated on two things:

1. Selection of energy function [20].
2. Selection of dynamic learning rate [9, 16, 22].

GA needs encoding operator and decoding operator i.e., selection, mutation and crossover. Particles use mutation to jump out of local optima. Of course mutation can also help particle to explore the search space. To reduce the data dimensions, fused features are passed to hybrid PSO-GA that eliminates irrelevant features [18]. When the FNN becomes complex then the Genetic algorithm convergent speed will become slow. PSO applied in real world problems with promising output [13]. When the FNN becomes complex then the Genetic algorithm convergent speed will become slow. PSO applied in real world problems with promising output [15].

FNN training by the hybrid evolutionary algorithm is testified by using Iris data classification and the result shows that the proposed hybrid algorithm possesses good result to find the global optimum compared to the LM algorithm. This paper is organized as follows: section 2 describes PSO algorithm and Levenberg-Marquardt algorithm. Section 3 describes that in XOR problem, the convergence speed is better in Hybrid evolutionary algorithm compared to Levenberg Marquardt algorithm. Section 4 describes parameters setting for the hybrid PSO-FNN algorithm to solve Iris problem. Section 5 describes about the result. Section 6 concludes hybrid evolutionary algorithm based on PSO in FNN training includes a number of components that gives advantage to increase converge speed with accuracy.

2. Particle Swarm Optimization Algorithm

Particle swarm optimization is a heuristic global optimization method put forward in the year 1995. It is based on the research of bird movement behavior. When searching for food, the birds go together before they locate the place where they can find the food. Birds are communicating the information, while searching the food; the birds will eventually flock to the place where food can be found. The food resource is equal to the most optimist solution during the whole course. This algorithm can be used to work out the complex optimist problems. Due to its easy implementation, the algorithm can be used widely in the fields such as function optimization, the model classification, neutral network training, the signal procession, automatic adaptation.

In the basic particle swarm optimization algorithm, particle swarm consists of "n" particles, and the position of each particle stands for the potential solution in d-dimensional space. The particles change its condition according to the following three principles:

1. To keep its inertia.
2. To change the condition according to its most optimist position.
3. To change the condition according to the swarm's most optimist position.

The position of each particle in the swarm is affected both by the most optimist position during its movement and the position of the most optimist particle in its surrounding. When the whole particle swarm is surrounding the particle, the most optimist position of the surrounding is equal to the one of the whole most optimist particle; this algorithm is called the PSO. Each particle is defined by its current speed and position. To optimize a d-dimensional continuous objective function f:R->R, a population of particles={p1,….pn} is initialized. At anytime‘t’, a particle pi has an associated position vector pb_i^t. This position vector contains the best position the particle has ever visited. PSO algorithm updates the particles velocities and positions.

Let us assume that φ1 and φ2 are two parameters called acceleration coefficients. These are generated at every iteration. We describe the variants that are selected to be part of our study in the following paragraphs.

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1^k (pbest_{id}^k - x_{id}^k) + c_2 r_2^k (gbest_d^k - x_{id}^k) \quad (1)$$

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

In this, v_{id}^k and x_{id}^k stands for the speed of the particle ‘i’ at its ‘k’ times. d is dimension of its position. $pbest_{id}^k$ Represents the d-dimension of the individual ‘i’ at its most optimist position at its ‘k’ times. $gbest^k_d$ is the d-dimension of the swarm at its most optimist position.

In order to avoid particle being far away from the searching space, the speed of the particle created at its each direction is confined between -vdmax and vdmax. If the number of vdmax is too big, the solution is far from the best. If the number of vdmax is too small, the solution will be the local. c_1 and c_2 represent the speeding figure, regulating the length when flying to the most optimist individual particle. If the figure is too small, the particle is probably far away from the target field. If the figure is too big, the particle may be flying to the target field suddenly. The proper figures for c_1 and c_2 can control the speed of the particle's flying and the solution will be the global optima. Usually, c_1 is equal to c_2 and they are equal to 2; r_1 and r_2 represent random fiction. Each particle pursuits the optimist particle in its surrounding to regulate its speed and position. Next we see about the variants which are used in this research. They are constricted particle swarm optimizer, time varying acceleration coefficients and also discussed about topology modification.

2.1. Constricted Particle Swarm Optimizer

Constriction factor of particles velocity avoids the unlimited growth of the particles' velocity [4].

$$v_{id}^{k+1} = x(v_{id}^{k} + c_1 r_1^k (pbest_{id}^k - x_{id}^k) + c_2 r_2^k (gbest_d^k - x_{id}^k)) \quad (3)$$

Where x value is set to 0.729. This will be referred to as constricted PSO.

2.2. Time Varying Acceleration Coefficients

PSO can be optimized with time varying acceleration coefficients. A local search behavior is amplified by linearly adapting the value of the acceleration coefficients φ1 and φ2

2.3. Topology Modification

Adaptive algorithm can be used to manage the Exploration and exploitation behavior of the PSO. At each iteration, a child particle updates its velocity by considering the best performance of its parent. A low branching degree has a more exploratory behavior than with a high branching degree.

Algorithm1: PSO Algorithm

```
For each particle
        Initialize particle
End
Do
        For each particle
```

```
    Calculate fitness value
        If fitness value is better than its best
            Set current (pBest) as the new pBest
  End
     Choose best fitness value of all particles as gBest
For each particle
     Calculate particle velocity according to equation
     Update particle position according to equation
     End
```

While maximum iterations or minimum error criteria is not attained Particles' velocities on each dimension are clamped to a maximum velocity Vmax

2.4. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) is one of the fastest and accurate learning algorithms for small to medium sized networks. The advantage of the LM algorithm decreases as the number of network parameters increases. The LM algorithm can find a solution of a system of non-linear equations, y=φx, by finding the parameters, φ, that link variables, y, to variables, x, by minimizing an error of a function of said system by using error gradient information for every parameter considered in the system. The LM algorithm in (4), finding the appropriate change, Δφ, leading to smaller errors. The LM-algorithm depends on error, E, the Hessian matrix H, the gradient of the error, ∇ J, a scalar μ which controls the trust region, and I is the identity matrix.

$$\begin{gathered} H = J \times J \\ \nabla J = J \times E \\ \Delta\varphi = -(H + I\mu)^{-1}\nabla J \end{gathered} \tag{4}$$

Algorithm 2: Levenberg-Marquardt algorithm

1. *While i < Max Iteration*
2. *Output of first layer is calculated as below*
3. *Output after hidden layer is* $X_{Hidden} = f(Net_{Hidden})$
   $$Net_{Hidden} = W_1 \times \begin{bmatrix} X_0 \\ 1 \end{bmatrix}$$
4. *Network output is*
   $$Net_{output} = W_2 \times \begin{bmatrix} X_{Hidden} \\ 1 \end{bmatrix}$$
5. *Error in output layer is*
   $$E = T\arg et - Net_{output}$$
6. *Weight vector is*
   $$\theta = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$$
7. *Jacobian matrix is*
   $$J(\theta) = \begin{bmatrix} f'_{linear}(X_{Hidden}) \\ W_2 \times f'_{logistics}(X_{Hidden}) \end{bmatrix}$$
8. *Error gradient is*
   $\nabla J = J \times E$
9. *Hessian matrix is*
   $H = J^T \times J$
10. *Updating Hessian matrix*
   $H = H + \lambda \times diagonal(H)$
11. *Weight change is*
   $\Delta\theta = H^{-1} \times \nabla J$
12. *New weight vector*
   $\theta_{new} = \theta_{old} + \Delta\theta$
13. *New hidden-output layer weights*
   $W_{2new} = W_{2old} + \theta_{new}(W_2)$
14. *New input-hidden layer weights*
   $W_{1new} = W_{1old} + \theta_{new}(W_1)$
15. *Updating* λ
16. *Calculating update conditions*
   $L = \Delta\theta^T \nabla J + \Delta\theta^T \Delta\theta\lambda_{old}$
17. *New lambda*
   $$\lambda = \begin{cases} \frac{\lambda}{2} if\ 2N(MSE - MSE_{new}) > 0.75L \\ 2\lambda if\ 2N(MSE - MSE_{new}) \le 0.25L \end{cases}$$
18. *Check if training conditions are still true. If true, repeat or go to step 10. Otherwise exit training*

Starting weights are adjusted by a learning algorithm to reach the desired state with the lowest errors Equation.

3. Hybrid Evolutionary Algorithm

The PSO is used to find global optimistic result. The hybrid algorithm is created using the PSO with Back propagation. Here the PSO is employed to increase the search speed.

Algorithm 3: hybrid algorithm

1. *Initialize the positions and velocities of particles.* P_b *is positions of the current particles.* P_g *is the best position of the initialized particles.*
2. *At the Maximum generation, go to step 5 else go to 3.*
3. *The positions and velocities of all particles are updated, then group of new particles are generated.*
4. *If the ith particle's new position is better than* P_{ib}*, then* P_{ib} *is set as the new position of the ith particle. If the best position of all new particles is better than* P_g*, then* P_g *is updated.*
5. *If* P_g *is unchanged, then output* P_g *else go to step 2.*
6. *Use Steepest Descent method to search around* P_g*, if search result is better than Pg , output the current search result*

3.1. Neural Network Training

We use an FNN with the structure of 2-2-1 to address XOR problem. We use the sigmoid functions for the hidden layer. For our analysis, we considered the below sigmoidal activation function to generate the output

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

Where n is the number of the input node. w_{ij} is the weight of the connection from the nth node of input layer to the jth node of hidden layer. The output of the kth output layer is

$$y_k = \sum_{i=1}^{n} w_{kj} \cdot f(s_j) - \theta_k \tag{6}$$

Where w_{kj} is the connection weight from the jth hidden node to the kth output node. θ_k is the threshold of the kth output unit. The learning error E can be calculated by the following formulation:

$$E_k = \frac{1}{2}(T_k - O_{ok}) \tag{7}$$

Where T_k -O_{ok} is the error of the actual output and desired output of the ith output unit. When the PSO algorithm is used in evolving weights of feed forward neural network, we need to decode each particle into weight matrix.

4. PSO-FNN Algorithm Parameter Setting

We use an FNN with the structure of 4-6-3 to address Iris problem. Suppose that the hidden layer has 6 neurons. Here we only evolve the network weights. So the particle will be a group of weights. There are 4*6+6*3=42 weights. So, the particle consists of 42 real numbers. We apply PSO-FNN algorithm in Iris classification. The comparison is carried out in our benchmark suite and analyzed. The Iris data has 135 samples evenly distributed in three classes, called iris-setosa, iris-versicolor and iris-virginica. Each sample has four features: sepal length x_1 and width x_2, petal length x_3 and width x_4. The samples evenly distributed in the three classes are used to train the FNN. 45 samples are used to test the generalization ability. We considered two population sizes 12 and 30 with fully connected population. In our experimental setup we used the parameter settings listed in Table 1.

Table 1. Hybrid parameter settings.

Hybrid Algorithm	**Parameter Settings**
Constricted	Acceleration Coefficients φ1=φ2=2.06, Coefficients Constriction factor x=0.729, Maximum velocity Vmax=±Xmax
Stochastic IW	Acceleration Coefficients φ=1.494, inertia weight in the range [0.6,1], velocity Vmax=±Xmax
Decreasing IW	Acceleration Coefficients φ1=φ2=2, decreasing inertia weight from 0.9 to 0.4, velocity Vmax=±Xmax

5. Analysis of Results

It is said that Leverberg-Marquardt is one of the fastest and accurate learning algorithms for small to medium sized networks. But Hybrid algorithm is faster than Levenberg-Marquardt algorithm as per our study. From the below Table 2 observation, you can see the time taken to converge in Hybrid algorithm is less when compared to the Levenberg-Marquardt algorithm. This shows that Hybrid algorithm is faster than Levenberg-Marquardt algorithm. In general, the standard LM algorithm does not perform as well on pattern recognition problems as it does on function approximation problems. The advantage of the LM algorithm decreases as the number of network parameters increases.

Table 2. Performance comparsion.

Hybrid Algorithm			**LMBP**		
epoch	**MAE**	**Time**	**epoch**	**MAE**	**Time**
14	0.9501	0.01375	1	0.9408	.02394
28	0.1680	0.01890	2	0.4981	.058795
42	0.1330	0.02403	4	0.4828	.060723
56	0.1055	0.03010	6	0.3818	.061394
70	0.0874	0.03477	8	0.3379	.061927
84	0.0806	0.03985	10	0.3359	.062461
98	0.0781	0.04497	12	0.3347	.063103
			14	0.3344	.063843
			16	0.2821	.064894
			18	0.1466	.065640
			20	0.0896	.066202

When you see Mean Absolute Error (MAE) between Hybrid algorithm and Levenberg-Marquardt in Table 2, you can notice that the mean correct recognition rate of the trained samples for the Hybrid algorithm is higher than Levenberg-Marquardt. The Hybrid algorithm can achieve 96% while LMBP algorithm can only reach 88%.

A graph is drawn considering Time and MAE for Hybrid and LMBP algorithms. The graph is given below in Figure 1. We can know that the Hybrid algorithm is much more accurate and stable than the LMBP algorithm

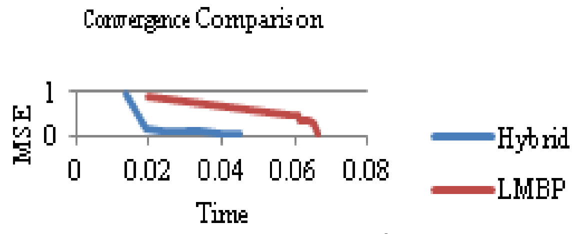

Figure 1. Convergence comparison for LM & hybrid algorithms.

The Hybrid algorithm is apparently better than the PSO algorithm. The Hybrid algorithm traces the global optimum using gradient descending method. We focused on varying population topologies and different strategies for updating a particle's velocities. We used fully connected topology, in which every particle is neighbor of four particles. The less connected topology delays the propagation. Thus, low connected topologies result in more exploratory behavior than highly connected ones [17].

Next we considered population size in Hybrid algorithm. Here, the hitting times for the Hybrid algorithm for different population sizes are obtained. The results obtained using different population size are tabulated and compared in Table 3. As seen from the Table 3, the Hybrid algorithm converges quickly when the population size increases. This shows that the population size is important for the algorithm to converge quickly. The stagnation tendency is smaller when using large population sizes.

Table 3. Performance comparsion with different popultion size.

Population Size 12			Population Size 20		
Time	epoch	MAE	Time	epoch	MAE
0.0272	14	0.37525	0.0393	14	0.41199
0.0429	28	0.2605	0.0664	28	0.33352
0.059	42	0.22904	0.0943	42	0.33352
0.0744	56	0.22081	0.1202	56	0.33335
0.0911	70	0.18765	0.1459	70	0.07862
0.1067	84	0.16583			
0.1223	98	0.16583			
0.1387	112	0.14419			
0.1542	126	0.0888			

In Table 4, we are evaluating Hybrid performance using Root Mean Square Error (RMSE), MAE and Mean Square Error (MSE). This table enables us to conclude that mean absolute error is more robust indicator of Hybrid performance.

Table 4. Hybrid performance using RMSE, MAE and MSE.

Execution Using RMSE			Execution Using MAE			Execution Using MSE		
Time (ms)	Epoch	RMSE	Time (ms)	Epoch	MAE	Time (ms)	Epoch	MSE
13	14	0.974	12	14	0.95	14	14	3.8
18	28	0.409	17	28	0.168	19	28	0.672
23	42	0.364	21	42	0.133	24	42	0.532
30	56	0.324	28	56	0.105	30	56	0.422
34	70	0.295	32	70	0.087	35	70	0.349
39	84	0.283	36	84	0.08	40	84	0.322
44	98	0.282	41	98	0.079	44	98	0.319
49	112	0.19				49	112	0.15
53	126	0.16				54	126	0.11
59	140	0.14				59	140	0.08
64	154	0.12				64	154	0.06
68	168	0.12				69	168	0.05
73	182	0.1						
77	196	0.08						
83	210	0.07						
88	224	0.06						
93	238	0.06						

The goal of comparison presented in this section is to identify algorithm components that provide good performance under different operating conditions. In fact some works are already exploring these issues [11, 21]. The results obtained using different parameter settings mentioned in Table 1 are tabulated and compared. As you see in the Table 5, Decreasing IW converges very quickly. When the acceleration coefficients value is φ1=φ2=2, the Hybrid algorithm gives better performance than the other value of acceleration coefficients. This proves that parameter settings are also important for algorithm to converge quickly.

Table 5. Performance comparsion with different parameter settings.

Constricted		Stochastic IW		Decreasing IW	
Time	MAE	Time	MAE	Time	MAE
0.038	0.412	0.039	0.411	0.0388	0.178
0.063	0.333	0.066	0.333		
0.09	0.333	0.094	0.333		
0.115	0.333	0.120	0.333		
0.141	0.078	0.145	0.078		

6. Conclusions

In this paper, we used a Hybrid algorithm that is PSO with Feed forward Neural Network. We did a comparison between Hybrid algorithm and Levenberg-Marquardt algorithm. The results show that Hybrid algorithm is better than Levenberg-Marquardt algorithm in terms of convergence speed and mean correction rate. And also we discussed the limitations of Levenberg-Marquardt algorithm. Next, we considered few variants in our study to improve the performance of the Hybrid algorithm. In our implementations we considered most promising PSO variants. The variants considered in our study are population size, acceleration coefficient, coefficient constriction factor and velocity of the swarm. The Hybrid algorithms such as Constricted algorithm, Stochastic IW algorithm, Decreasing IW algorithm are considered. These hybrid algorithms are formed with different value for the variants. Results implies that Decreasing IW is most promising than the other two algorithms. Thus we say that these variants play major role in improving convergence speed as well as in improving accuracy of the algorithm. Also it is proved that the integration of components in different ways in hybrid algorithm produces effective optimization of back propagation algorithm.

References

[1] Angeline P., Sauders G., and Pollack J., "An Evolutionary Algorithm that Construct Recurrent Neural Network," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54-65, 1994.

[2] Behera L., Kumar S., and Patnaik A., "On Adaptive Learning Rate that Guarantees Convergence in Feed Forward Networks," *IEEE Transactions Neural Networks*, vol. 17, no. 5, pp. 1116-1125, 2006.

[3] Clerc M., *Particle Swarm Optimization*, Wiley online Library, 2006.

[4] Clerc M. and Kennedy J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[5] Cooren y., Clerc M., and Siarry P., "Performance Evaluation of TRIBES, an Adaptive Particle Swarm Optimization Algorithm," *Swarm Intelligence*, vol. 3, no. 2, pp. 149-178, 2009.

[6] De Oca M., Stutzle T., Birattari M., and Dorigo M., "Frankenstein's PSO; A Composite Particle Swarm Optimization Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120-1132, 2009.

[7] Fan S. and Zahara E., "A Hybrid Simplex search and Particle Swarm Optimization for Unconstrained Optimization," *European Journal of Operational Research*, vol. 181, no. 2, pp. 527-548, 2007.

[8] Gori M. and Tesi A., "On the Problem of Local Minima in Back-propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76-86, 1992.

[9] Jacobs R., "Increased Rates of Convergence through Learning Rate Adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295-307, 1988.

[10] Janson S. and Middendorf M., "A Hierarchical Particle Swarm Optimizer and its Adaptive Variant," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1272-1282, 2005.

[11] Jordan J., Helwig S., and Wanka R., "Social Interaction in Particle Swarm Optimization, the Ranked FIPS and Adaptive Multi Swarms," *in Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, pp. 49-56. 2008.

[12] Kennedy J., "Small Worlds and Mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance," *in Proceedings of Congrees on Evolutionary Computation*, Washington, pp. 1931-1938, 1999.

[13] Khan S., Nazir M., Riaz N., and Khan M., "Optimized Features Selection using Hybrid PSO-GA for Multi-view Gender Classification," *The International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 183-189, 2014.

[14] Li X., "Adaptively Choosing Neighborhood Bests Using Species in a Particle Swarm Optimizer for Multimodal Function Optimization," *in Proceedings of Genetic and Evolutionary Computation Conference*, Seattle, pp. 105-116. 2004

[15] Liang J., Qin A., Suganthan P., and Baska S., "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, 2006.

[16] Magoulas G., Plagianakos V., and Vrahatis M., "Globally Convergent Algorithm with Local Learning Rate," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 774-779, 2002.

[17] Mendes R., Kennedy J., and Neves J., "The Fully Informed Particle Swarm: Simpler, Maybe Better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204-210, 2004.

[18] Nguyen Q., Ong Y., and Lim M., "A Probabilistic Memetic Framework," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 604-623, 2009.

[19] Wu Z. and Zhou J., "A Self Adaptive Particle Swarm Optimization Algorithm with Individual Coefficient Adjustment," *in Proceedings of International Conference on Computational Intelligence and Security*, Harbin, pp. 133-136, 2007.

[20] Yao X., "A Review of Evolutionary Artificial Neural Network," *International Journal Intelligent Ssystem*, vol. 8, no. 4, pp. 539-567, 1993.

[21] Yisu J., Knowles J., Hongmei L., Yizeng L., and Kell D., "The landscape Adaptive Particle Swarm Optimizer," *Applied Soft Computing*, vol. 8, no. 1, pp. 295-304, 2008.

[22] Yu X., Chen G., and Cheng S., "Acceleration of Backpropagation Learning using Optimized Learning Rate and Momentum," *Electronics Letters*, vol. 29, no. 14, pp. 1288-1290, 1993.

Thinakaran kandasamy received M.E., degree in computer science from Mahendra Engineering College which is affiliated to Anna University, Coimbatore, Tamilnadu in 2009. He is currently working toward the Ph.D. degree at the Anna University. He is currently an Assistant Professor in Computer Science Engineering, Sri Venkateswara College of Engineering & Technology, Thiruvallur, India. His current research interests include Neural Network and Data Mining.

Rajasekar Rajendran received his doctorate from Department of Aeronautics, Imperial College, London, UK. His aeronautical masters' degree was from IIT, Madras. He is currently working as the Professor and Head of Aeronautical Engineering Department, Excel Engineering College, Erode, India. (an affiliated college under Anna University, Chennai). His specialization and research interests are aerodynamics,Neural Network and its applications.