International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 10 (July 2013), PP. 01-10

Training of Radial Basis Function Using Particle Swarm Optimization

Jagriti Sharma[1], Sunali Mehta[2]
[1.2] Department of Computer Science & Engineering

Abstract:- Particle swarm optimization (PSO) is a population based optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. It is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It is an optimization of continuous nonlinear functions that are used to find the best solution in problem space.. In this paper, this nature inspired behaviour has been considered to optimise the neural network i.e Radial Basis Function by PSO. The RBF is a Neural Network. It is based on moving or directed along the radius. Radial Basis Function emerged as a variant of artificial neural networks in late 80's. RBF's are embedded as two layer neural network, where each hidden unit implements a radial activated function. The output units implement a weighted sum of hidden unit outputs. In order to use a Radial Basis Function Network we need to specify the hidden unit activation function, the number of processing units, a criterion for modeling a given task and a training algorithm for finding the parameters of the network. Finding the RBF weights is called network training. If we have at hand a set of input-output pairs, called training set, we optimize the network parameters in order to fit the network outputs to the given inputs. The fit is evaluated by the cost function. By means of training the neural network models the underlying the function of the certain mapping. In order to model such a mapping we have to find the network weights. In Radial Basis Function parameters are found such that they minimize the cost function.

Keywords:- Radial Basis Function, Particle Swarm Optimization, Particle best, Global best, Local best

I. INTRODUCTION

PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds don't know where the food is. But they know how far the food is in each iteration. So what's the strategy to find the food? The effective one is to follow the bird which is nearest to the food.

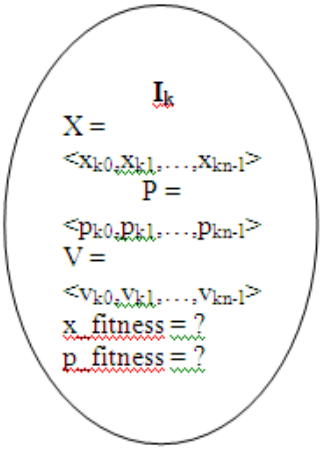
PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far.(the fitness value is also stored) .This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding, the two best values, the particles update its velocity and positions. . Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has achieved so far by that particle. This value is called personal best, pbest. A particle (individual) is composed of:

– Three vectors:
- The **x-vector** records the current position (location) of the particle in the search space,
- The **p-vector** records the location of the best solution found so far by the particle, and
- The **v-vector** contains a gradient (direction) for which particle will travel in if undisturbed.

– Two fitness values:
- The **x-fitness** records the fitness s of the x-vector, and
- The **p-fitness** records the fitness of the p-vector.



In PSO, particles never die! Particles can be seen as simple agents that fly through the search space and record (and possibly communicate) the best solution that they have discovered. So the question now is, "How does a particle move from on location in the search space to another?" This is done by simply adding the v-vector to the x-vector to get another x-vector ($X_i = X_i + V_i$). Once the particle computes the new Xi it then evaluates its new location. If x-fitness is better than p-fitness, then $P_i = X_i$ and p-fitness = x-fitness.

A. Algorithm

For each particle
Initialize particle
Do
For each particle
Calculate fitness value
If the fitness value is better than the best personal fitness value in history, set current value as a new best personal fitness value
Choose the particle with the best fitness value of all the particles, and if that fitness value is better than current global best, set as a global best fitness value
For each particle
Calculate particle velocity according to the change in velocity
Update particle position according to the change in position
End

Fig.1: Pseudo code for PSO

In order to understand it clearly let us consider a scenario given ,Consider this search space with random set of particles:

(iteration0)

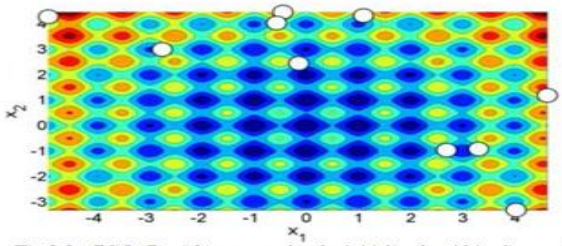


Fig.1.1: PSO: Particles are randomly initialized within the search space (iteration0)

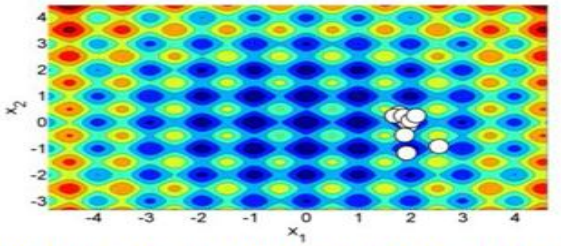


Fig.2: PSO: Particles are converging to local minimizer [2,0] via their attraction to the global best

• The basic concept of PSO lies in accelerating each particle toward its pbest and the gbest locations, with a random weighted accelaration at each time step as shown in Fig.

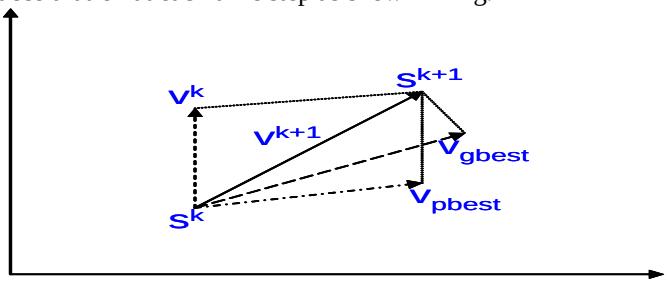


Fig.3: Concept of modification of a searching point by PSO

s^k: current searching point.
s^{k+1}: modified searching point.
v^k: current velocity.
v^{k+1}: modified velocity.
v_{pbest} : velocity based on pbest.
v_{gbest} : velocity based on gbest

Each particle tries to modify its position using the following information:

- the current positions,
- the current velocities,
- the distance between the current position and pbest,
- the distance between the current position and the gbest.

B. Problem in PSO- Stagnation

Stagnation- Stagnation means when the particles remains still or cant move from its its position. It is the main problem in PSO in which the particles donot move from their local positions to global position

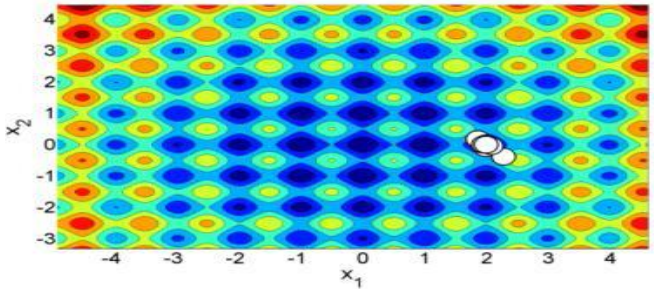


Fig.4: PSO: Premature convergence to a local minimum begins

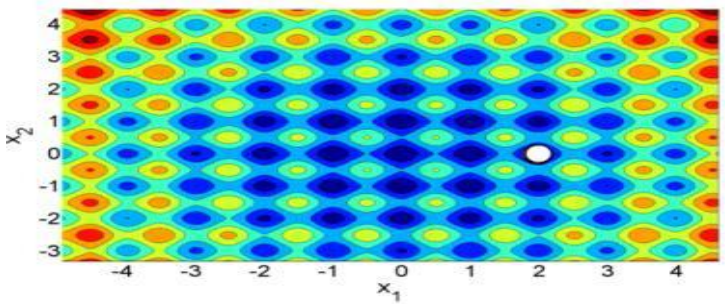


Fig.5: The local minimum is being honed in on, but no progress is being made toward a better solution since the swarm has stagnated.

The swarm is said to have converged prematurely when the proposed solution approximates a local rather than global, minimizer and when progress toward a global minimizer has ceased so that continued activity could only hope to refine a local minimize .Stagnation is a result of premature convergence. Once particles have converged prematurely, they continue converging to within extremely close proximity of one another so that the global best and all personal bests are within one small region of the search space. Since particles are continually

attracted to the bests in that same small vicinity, particles stagnate as the momentum from their previous velocities vanishes. While particles are technically always moving, stagnation can be thought of as a lack of movement discernable on the large scale, from which Perspective the stagnated swarm will appear as one static dot.

Section II reviews the idea of neural network. Section III discusses how neural network can be trained with Particle Swarm optimization.

II. NEURAL NETWORK - RBF

Neural Network is developed by McCulloch and Pitts in 1943.Neural Network has certain processing capabilities of the human brain that tries to simulate its learning process. An Artificial Neural Network, often just called a neural network, is a mathematical model inspired by biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. Thus the term has two distinct usages:

- **Biological Neural Networks** is an information processing paradigm, inspired by biological system, composed of a large number of highly interconnected processing elements (neurons) to solve specific problems. Learning in biological system involves adjustment to the synaptic connections that exist between the neurons.

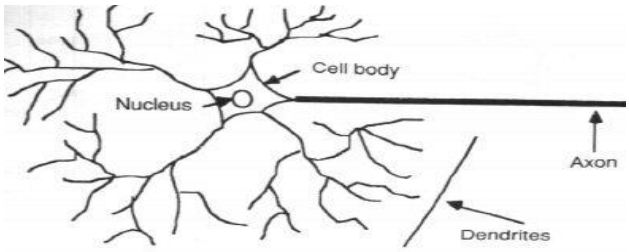


Fig.6:structure of biological neural network

- **Artificial Neural Networks** is an artificial representation of the human brain that tries to simulate its learning process. An artificial Neural network is often called a Neural network or simply Neural net. It is a mathematical model or computation model.

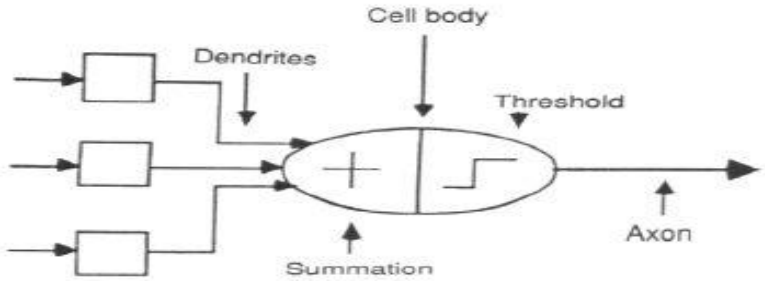


Fig.7: structure of biological neural network

From Human Neurons to Artificial Neurons

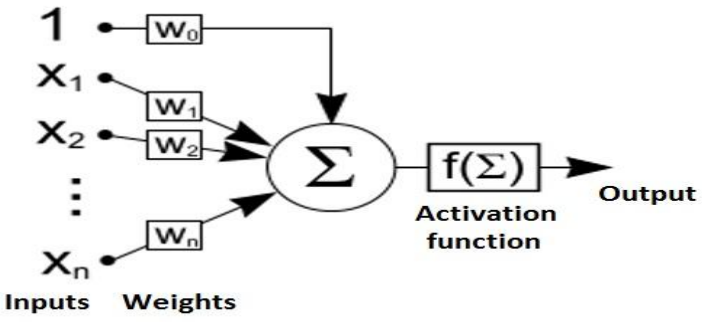


Fig.8: The Neuron model

Various Neural Networks have been developed. But in this paper Radial Basis Function Neural Network is used to train the PSO.

Broomhead and Lowe in 1988 use the RBF in the design of Neural Network. It is based on moving or directed along the radius. Radial Basis Function emerged as a variant of artificial neural networks in late 80's. RBF's are embedded as two layer neural network, where each hidden unit implements a radial activated function. The output units implement a weighted sum of hidden unit outputs. The input into an RBF unit is nonlinear while the output is linear. Due to their non linear approximation properties, RBF network are able to model complex mappings.

In order to use a Radial Basis Function Network we need to specify the hidden unit activation function, the number of processing units, a criterion for modeling a given task and a training algorithm for finding the parameters of the network. Finding the RBF weights is called network training. If we have at hand a set of input-output pairs, called training set, we optimize the network parameters in order to fit the network outputs to the given inputs. The fit is evaluated by the cost function. By means of training the neural network models the underlying function of the certain mapping. In order to model such a mapping we have to find the network weights. There are two categories of training algorithms: supervised and unsupervised. RBF networks are used mainly in supervised applications. In a supervised application we are provided with a set of data samples called training set for which the corresponding network outputs are known. In Radial Basis Function parameters are found such that they minimize the cost function.

Radial functions are a special class of function. Their characteristic feature is that their response decreases or increases monotonically with distance from a central point. The centre, the distance scale, and the precise shape of the radial function are parameters of the model, all fixed if it is linear.

Gaussian Function

- Different types of Radial basis Functions could be used, but the most common is the Gaussian Function.
- It shows the graphical representation of Gaussian functions. It is a bell shaped curve.

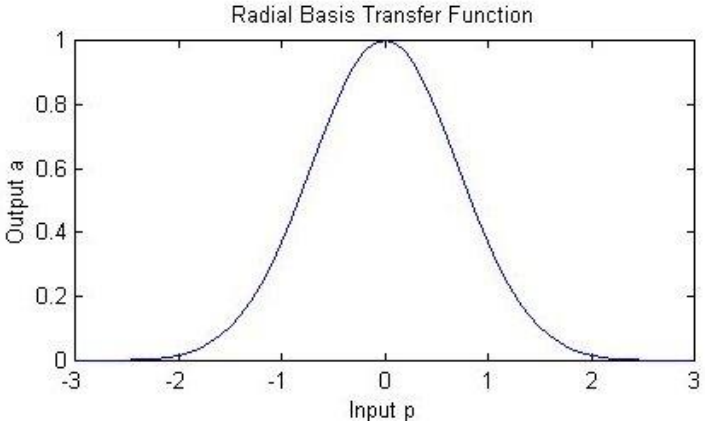


Fig.1.9: Graphical representation of Gaussian Function

A typical radial function is the Gaussian which, in the case of a scalar input, is

$$h(x) = \exp(-(x-c)^2/r^2)$$

Its parameters are its centre c and its radius r. Figure 1.7 illustrates a Gaussian RBF with centre c = 0 and radius r = 1.

A Gaussian RBF monotonically decreases with distance from the centre. In contrast, a multiquadric RBF which, in the case of scalar input, is

$$h(x) = \sqrt{r^2 + (x-c)^2}/r^2$$

monotonically increases with distance from the centre see Figure 1.7 Gaussian-like RBFs are local (give a significant response only in a neighborhood near the centre) and are more commonly used than multiquadric-type RBFs which have a global response. They are also more biologically plausible because their response is finite.

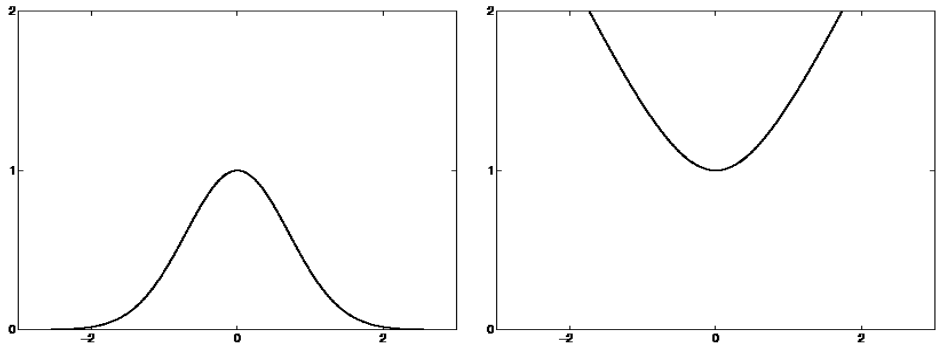


Fig.10: Gaussian (left) and multiquadric RBF.

A. Architecture of RBF

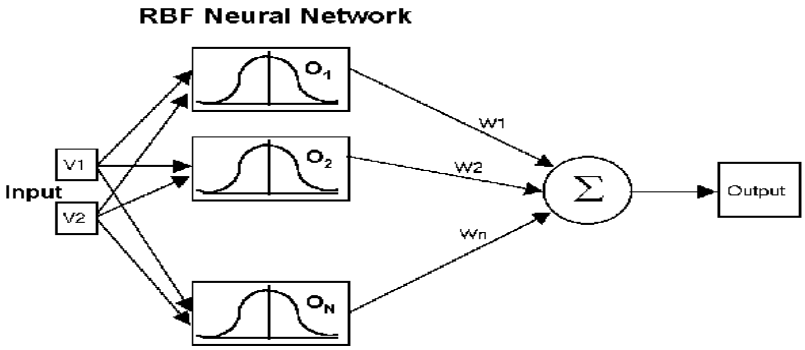


Fig.11: Architecture of RBF

- Input layer – There is one neuron in the input layer for each predictor variable. In the case of categorical variables, N-1 neurons are used where N is the number of categories..
- Hidden layer – This layer has a variable number of neurons (the optimal number is determined by the by the training process). Each neuron consists of a radial basis function centered on a point with as many dimensions as there are predictor variables.
- Summation layer – The value coming out of a neuron in the hidden layer is multiplied by a weight associated with the neuron (W1, W2, ...,Wn in this figure) and passed to the summation which adds up the weighted values and presents this sum as the output of the network.

B. How RBF networks work

Although the implementation is very different, RBF neural networks are conceptually similar K-Nearest Neighbor (k-NN) models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have close values of the predictor variables.

The Algorithm on how to compute the K-nearest neighbors is as follows:

Algorithm

Step 1. Determine the parameter K = number of nearest neighbors beforehand. The value is all up to you.

Step 2. Calculate the distance. You can use any distance algorithm.

Step 3. Sort the distances for all the training samples and determine the nearest neighbor based on the K-th minimum distance.

Step 4. Since this is supervised learning, get all the Categories of your training data for the sorted value which fall under K.

Step 5.Use the majority of nearest neighbors as the prediction value.

Consider this figure:

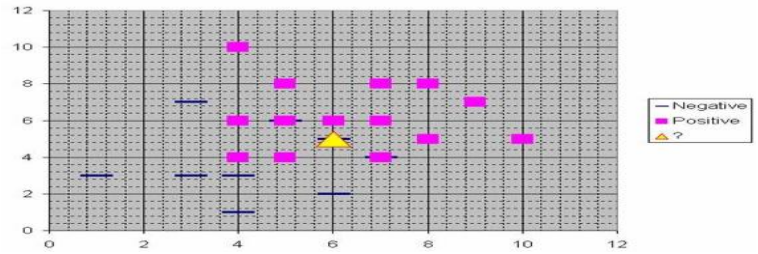


Fig12: K-means Clustering

Assume that each case in the training set has two predictor variables, *x* and *y*. The cases are plotted using their *x*,*y* coordinates as shown in the figure. Also assume that the target variable as two categories, positive which is denoted by a square and negative which is denoted by a dash. Now, suppose we are trying to predict the value of a new case represented by the triangle with predictor values *x*=6, *y*=5.1. Should we predict the target as positive or negative? Notice that the triangle is position almost exactly on top of a dash representing a negative value. But that dash is in a fairly unusual position compared to the other dashes which are clustered below the squares and left of center. So it could be that the underlying negative value is a odd case. The nearest neighbor classification performed for this example depends on how many neighboring points are considered. If 1-NN is used and only the closest point is considered then clearly the new point should be classified as negative since it is on top of a known negative point. On the other hand, if 9-NN classification is used and the closest 9 points are considered, then the effect of the surrounding 8 positive points may overbalance the close negative point.

An RBF network positions one or more RBF neurons in the space described by the predictor variables (x,y in this example). This space has as many dimensions as there are predictor variables. The Euclidean distance is computed from the point being evaluated (e.g., the triangle in this fig) to the center of each neuron, and a radial basis function (RBF) (also called a kernel function) applied to the distance to compute the weight (influence) for each neuron. The radial basis function is so named because the radius distance is the argument to the function.

III. HOW RBF IS TRAINED WITH PSO

Various ways for removing the stagnation

1) terminate the search and accept the best decision vector found as the proposed solution
2) allow the search to continue and hope that the swarm will slowly refine the quality of the Proposed solution, though it is likely only an approximation of a local minimizer rather than the desired global minimize
3) restart the swarm from new locations (i.e. start from scratch) and search again to see if a better solution
4) reinvigorate the swarm by introducing diversity so the search can continue more or less from the current location without having to restart and re-search low quality regions of the search space.

Particle swarm optimization (PSO) is known to suffer from stagnation once particles have prematurely converged to any particular region of the search space. The proposed regrouping PSO (RegPSO) avoids the stagnation problem by Radial Basis Function that automatically triggers swarm regrouping when premature convergence is detected. This mechanism liberates particles from sub-optimal solutions and enables continued progress toward the true global minimum. Particles are regrouped within a range on each dimension proportional to the degree of uncertainty implied by the maximum deviation of any particle from the globally best position. This is a computationally simple yet effective addition to the computationally simple PSO algorithm. Experimental results show that the proposed RegPSO successfully reduces **st**agnation.

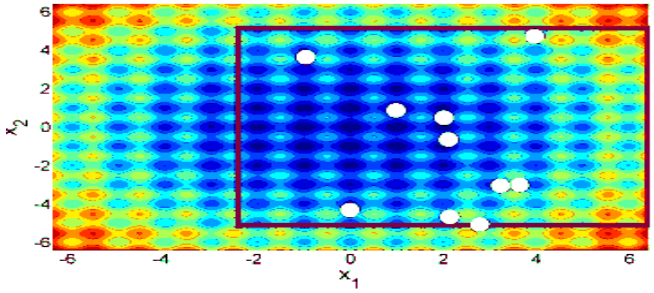


Fig.13: RegPSO: Regrouping PSO detects premature convergence at iteration 102

of last fig once gbest PSO has stagnated. The swarm is regrouped.

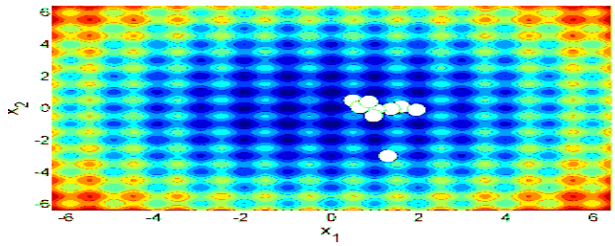


Fig.14: RegPSO: The swarm is migrating toward a better position found by one of the particles near position [1, 0]. (Iteration 123).

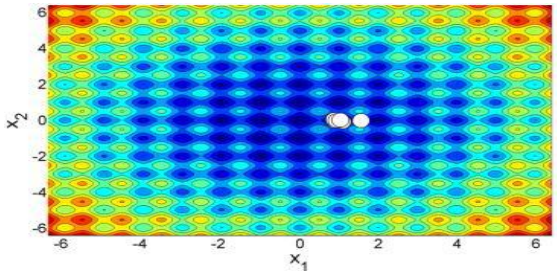


Fig.15: RegPSO: The swarm is prematurely converging to a new local minimum. (Iteration 143)

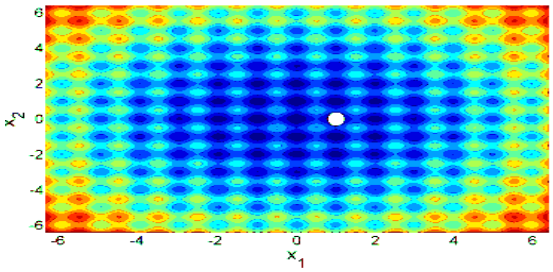


Fig16: RegPSO: The swarm has stagnated at local minimizer [1, 0] (Iteration 219)

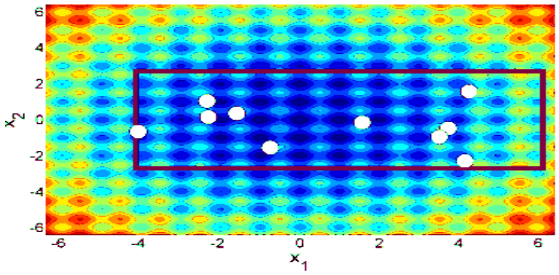


Fig.17: RegPSO: A second regrouping is triggered. (Iteration 220)

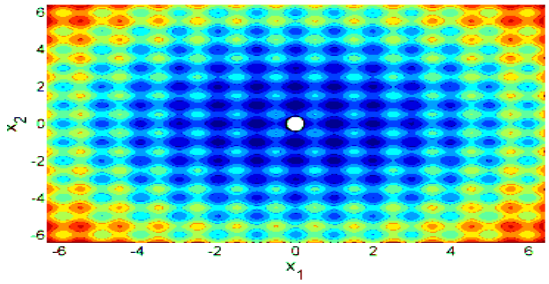


Fig.18: RegPSO: The global minimum has been found successfully. (Iteration 270)

The main objective is that it allows regrouping and allows the swarm to escape from local wells efficiently and continue in its search rather than stagnating or restarting the search entirely.

C. FLOWCHART OF WORK

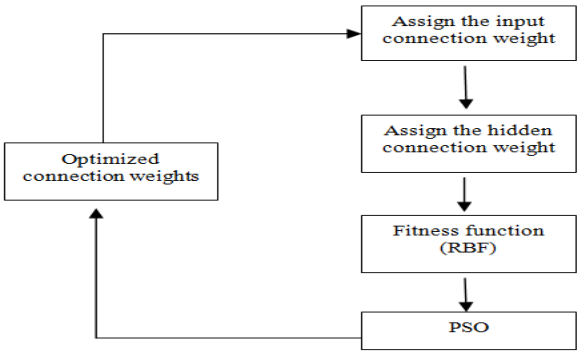


Fig.19: Flowchart of the present work

In this figure the PSO algorithm optimizes the connection weights between layers in Radial basis function. A Radial Basis Function is able to work parallel with input variables and consequently handle large sets of data swiftly. The connection weights in RBF are optimized by using PSO. It has three phases in the first phase; the PSO searches the optimal or near optimal connections weights and threshold.

The connection weights and threshold are initialized in two random values before search process. It needs three sets of parameters.

(a) The first set is the set of connection weights of input layer and the hidden layer.
(b) The second is the set of connection weight between the hidden layer and the output layer.
(c) The third step represents the threshold. a fully connected feed forward network with one hidden layer and one
output is shown above.

D. Result

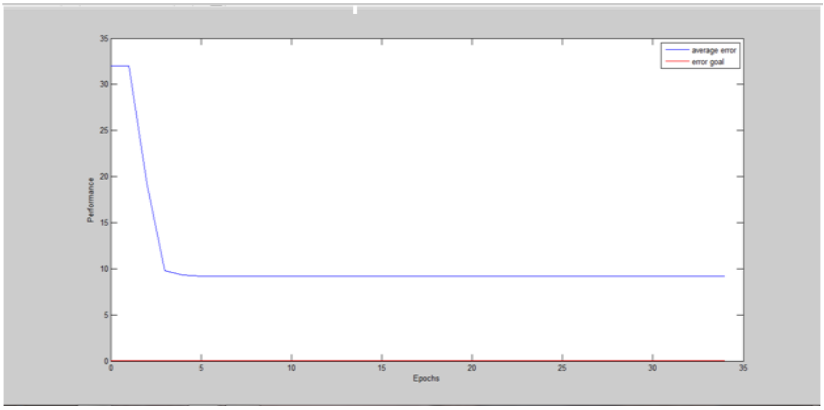


Fig.20: Performance versus epochs

IV. CONCLUSION AND FUTURE SCOPE

A. Conclusion

- It automatically triggers swarm regrouping when premature convergence is detected
- It is an approach for overcoming the stagnation problem
- This regrouping helps liberate the swarm from the state of premature convergence in order to enable continued progress toward the global minimum.

B. Future Scope

- The future work can be extended in following directions:
- In an attempt to provide a standard language for expressing predictive models, the Predictive Model Markup Language (PMML)
- Most important attribute can be found for this work can be extended to further programming languages.
- It can be used for Pattern Recognition.

REFERENCES

[1]. Kennedy, J. and Eberhart, R.C.(1995), "*Particle Swarm Optimization*", In Proceedings of IEEE International Conference on Neural Networks, New Jersey, USA, pp. 1942-1948.

[2]. Kennedy, J. (1997). "*The particle swarm: social adaptation of knowledge*". *Proceedings of IEEE International Conference on Evolutionary Computation*. pp. 303–308.

[3]. Poli, R. (2008). *"Analysis of the publications on the applications of particle swarm optimisation*". Journal of Artificial Evolution and Applications **2008**: 1–10. doi:10.1155/2008/685175.

[4]. Qinghai Bai ,"*Analysis of Radial Basis Function*",computer and information science, vol.3, no.1,February 2010.

[5]. Meissner.M., Schmuker.M., Schneider.G., [2006] "*Optimized Particle Swarm Optimization and its application to artificial neural network training*", published in BMC Bioinformatics, vol.7, doi:10.1186/1471-2105-7-125

[6]. D. Simon, "*Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*", John Wiley & Sons, 2006

[7]. D. Simon, "*Training Radial Basis Neural Networks with the Extended Kalman Filter,*" *Neurocomputing*, vol. 48, no. 1–4, pp. 455–475, October 2002

[8]. C. Yang and D. Simon, "*A New Particle Swarm Optimization Technique*," *International Conference on Systems Engineering*, Las Vegas, NV, pp. 164–169, August 2005.

[9]. *M. Ovreiu and D. Simon, "Cardiomyopathy Detection from Electrocardiogram Features," in Cardiomyopathies – From Basic Research to Clinical Management (J. Veselka, editor) InTech, pp. 117–134, 2012* .

[10]. Bratton, D. and Kennedy, J. (2007), "*Defining a Standard for Particle Swarm Optimization*", In Proceedings of IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, USA, pp. 120-127.

[11]. Aleksander, I. and Morton "*An introduction to neural computing*",. 2nd edition Neural Networks, Eric Davalo and Patrick Naim.

[12]. Y. Liu, J. A. Starzyk, Z. Zhu, "*Optimized Approximation Algorithm in Neural Network without Overfitting*", IEEE Trans. on Neural Networks, vol. 19, no. 4, June, 2008, pp. 983-995.

[13]. Esprit, I.F.Croall, J.P.Mason, "*Industrial Applications of Neural Networks*".

[14]. Eric Davalo and Patrick Naim Klimasauskas, CC.,1989 "*DARPA Neural Network Study*" (October, 1987-February, 1989). MIT Lincoln Lab. Neural Networks,

[15]. Hammerstrom, D. (1986), "*The 1989 Neuro Computing Bibliography*". A Connectionist/Neural Network Bibliography.

[16]. Broomhead,D.S,Lowe(1988) "*Multivariable functional interpolation and adaptive networks"*, complex system, vol 2 , pp.321-355.

[17]. Park, J, Sandberg, J.W(1991) "*Universal approximation using radial basis functions network*," Neural Computation, vol 3, pp.246-257.

[18]. Ali Idri, Abdelali Zakrani and Azeddine Zahi IJCSI , vol.7, issue 4, No.3, July 2010.

[19]. Adrian, G., [2001] "*the introduction of Radial Basis Function Networks*", online symposium for Electronics Engineers, issue.1, vol.1, pp 1-7

[20]. Haykin , S. (1994) Neural Network: "*A comprehensive foundation*". Upper saddle river, NJ: prentice hall.

[21]. Orr, M.J.L., [1996], "*Introduction to radial basis function networks*," http://www.anc.ed.ac.uk/~mjo/papers/intro.psor http://www.anc.ed.ac.uk/~mjo/papers/intro.ps.gz

AUTHORS PROFILE

Jagriti Sharma received her B.Tech in Computer Science & Engineering from Guru Gobind singh College of Engineering & Technology (Punjab Technical University) during the year 2008 and also did M.Tech in computer science & Engineering from Rayat Institute of Engineering & Information technology (Punjab Technical University) during the year 2013.

Sunali Mehta received her B.E in Information Technology from Model Institute of Engineering & Technology (Jammu University) during the year 2008 and also did M.Tech in computer science & Engineering from Rayat Institute of Engineering & Information technology (Punjab Technical University) during the year 2013.