Particle Swarm Optimization for creating self organized Evolutionary Back Propagation Artificial Neural Networks

Mr. Shivnath Ghosh
ISIM(IIIM)
Jaipur, Rajasthan
E-mail: shivnath01@yahoo.com

Dr. Santanu Koley
Dronacharya College of Engineering
Gr. Noida (UP)
E-mail:santanukoley@yahoo.com

Abstract: - **A novel technique for developing a self organized Back Propagation Artificial Neural Networks (BPANNs) or simply Back Propagation Neural Network (BPN) with the help of swarm Intelligence (SI) technique. This paper proposes an overlapping swarm intelligence algorithm for training of neural networks in which a particle swarm is assigned to each neuron to search for that neuron's weights. The training of ANN is a difficult task when it's get into the local minima and causes the low learning rates; it is required of an evolutionary multi dimensional algorithm to seek optimal weight values, the positional optimum values and the dimensional optimum in values in the dynamic problem space. The approach discussed throughout in this paper is the credit assignment process by first focusing on updating weights and biases swarms Intelligence and then evaluating the fitness of the particles using a network. With the proper adaptation of the SI-BPN process, the proposed method can develop an optimum network within an architecture space for a particular problem. Additionally, it provides a class list of all other potential configurations. This algorithm will provide superior learning ability to the traditional Back-Propagation (BP) method in terms of accuracy and speed.**

Index Terms: **Swarms Intelligence, Artificial Neural Networks, Back Propagation Neural Network, Learning rates, Local minima, Problem space.**

I. Introduction

The Artificial Neural Networks (ANNs) are known as the ``universal adaptation model'' and "computational models" with particular characteristics such as the ability to learn, data clustering. A neuron is a smallest processing unit in neural network. It is a node that processes all fan-in from other nodes and generates an output according to a transfer function called as activation function. The activation function presents a linear or non linear mapping form the input to output. Automatic determination of the optimal number of hidden layers and hidden nodes in each hidden layer is the most significant task. On such complex networks proper training can also be highly time-consuming. The optimum number of hidden

nodes/layers might depend on input/output vector sizes, training and test data sizes, more importantly the characteristics of the problem, e.g. its non-linearity, dynamic nature, etc.

Several researchers have attempted to design ANNs automatically with respect to a particular problem. The earlier attempts fall into two broad categories: constructive and pruning algorithms where many deficiencies and limitations have been reported [1][2]. Efforts have been then focused on evolutionary algorithms (EAs) [3], particularly over Genetic Algorithm (GA) [4] and Evolutionary Programming (EP) [5], Most GA based methods have also been found quite inadequate for evolving ANNs mainly due to two major problems: permutation problem and noisy fitness evaluation [6]. Although EP-based methods might address such problems, they usually suffer from their high complexity and strict parameter dependence. While the back propagation (BP) algorithm could be an effective method for training feed forward neural networks, it typically has a slow convergence rate and is known to suffer from local minima [7]. To overcome these limitations alternative approaches such as particle swarm optimization (PSO) algorithms, genetic algorithms, and hybrid approaches which use back propagation combined with evolutionary approaches have been used.

II. Fundamental Concept

A. Artificial Neural network : BPN

The Back propagation (BP) learning algorithm is currently the most popular learning rule for performing supervised learning task [9]. The BP algorithm is a generalization of delta rule known as delta rule [10]. The fundamental concept of BP can be divided into MLP in FNN and Backward pass. In a MLP (FNN mode) for m layers output is given by:

$$\boldsymbol{net_p^{(m)} = [w^{(m-1)}]^T\, o_p^{(m-1)} + \Theta^m} \quad (1)$$

$$\boldsymbol{o_p^{(m)} = \phi^m(net_p^{(m)})} \quad (2)$$

Where Θ^m, o_p^m, ϕ_p^m are bias, output, activation function respectively.

Backward pass for *m=3*.

$$\boldsymbol{\delta_{p,v}^2 = \left(\sum_{w=1}^{j3} \delta_{p,w}^{(3)} w_{v,w}^{(2)}\right) \phi\left(net_{p,v}^{(2)}\right)} \quad (3)$$

$\Delta \boldsymbol{w_{u,v}^{(1)} = \eta \delta_{p,w}^{(3)} o_{p,u}^{(1)}}$, Where (4)

$$0 < \eta < 1, Convergence\ rate.$$

B. Swarm Intelligence : PSO

Particle swarm optimization (PSO) is introduce by Dr. Eberhart and Dr. Kennedy in 1995[8], a randomized variable with stiffness based stochastic optimization technique, stimulated from social behavior of

bird flocking or fish schooling. In the problem search space, each “bird” is directly associated with “particle” through one – one mapping. All of particles have fitness values which are evaluated by the fitness function to be optimized, where fitness function has some specific parameters such as distance, velocities, which direct the flying of the particles. The particles moves through the problem search space by adopting the current optimum particles’ values obtained by fitness function.

PSO considers and have many similarities with evolutionary computation techniques such as Evolutionary algorithm, Intelligent water drop, Genetic Algorithms (GA). The stochastic system is initialized with a identified random solutions and searches for optima by updating best solution. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. PSO is initialized with a group of random particles and then searches for optima by updating generations. In every iteration, each particle is updated by following “best" values. 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 neighbors of the particle. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO).

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

The best part of PSO is comprehension knowledge using very few parameters and its’ adjustment. Particle swarm optimization has been used for serial correlation situation, NP complete problem, as well as for specific applications focused on a specific requirement.

Algorithm 1: Particle Swarm Optimization Process.

For each particle

Randomize : Initialize particle values

End

Do

For each particle

Calculate Position fitness value

If the Position fitness value is better than the best fitness value

Then

pbest ← set current value

End

Choose the particle with the best fitness value of all the particles as the *gbest*

For each particle

Calculate particle velocity

Update particle position

End

While maximum iterations or minimum error criteria is not attained

---

Each particle keeps track of its position for weight updating and its best solution so far achieved. Where a larger value of δw indicates that its value was learned more recently while a smaller value for δw indicates that the weight was learned less recently.

Algorithm 2: Share weight function (Neural Network for each Particle)

Let P_i and P_j be the two Particles that are to share weights

Let nn_i and nn_j be the neural networks of *Pi* and P_j respectively

For each weight *w* in the neural network **Do**

If $pi.\delta w > p_j.\delta w$ **then**

Insert nn_i's value for *w* into nn_j

Increment $p_j.\delta w$

End if

If $p_j.\delta w > p_i.\delta w$ **then**

Insert nn_j's value for *w* into nn_i

Increment $p_i.\delta w$

End if

End for

III. Related work

A stochastic search algorithm in MD problem search space, PSO faces some critical problems such as parameter variation and performance shift [11]. The next problem is being trapped in local optima [12]. Regarding 'gbest', 'pbest' and various modifications have been proposed in order to address the different problem we have [13], [14], [15].

IV. Approach

The goal is, to converge and establishment of the position and velocity dimension of particles in multidimensional search space with none linear function. Instead of fixed dimension, this algorithm is design to work positional and dimensional optima in dynamic environment. Non linear multidimensional PSO process at time t, each particle p in the swarm,

$\chi = \{x_1, x_2 \ldots x_p, \ldots x_s\}$ is represented by following characteristics :

$\boldsymbol{xx_{p,j}^{xd_p(t)}}(t)$: *j*th component(dimension)of the position of particle *p*, in dimension $xd_p(t)$

$\boldsymbol{vx_{p,j}^{xd_p(t)}}(t)$: *j*th component(dimension)of the velocity of particle *p*, in dimension $xd_p(t)$

$\mathbf{xy_{p,j}^{xd_p(t)}}(t)$: jth component (dimension) of the personal best (pbest) position of particle p, in dimension $xd_p(t)$

gbest (d): Global best particle index in dimension d

$\mathbf{x\hat{y}_j^d(t)}$**:** jth component (dimension)of the global best position of swarm, in dimension d.

$\boldsymbol{xd_{p\,(t)}}$**:** Dimension component of particle p

$\boldsymbol{vd_{p\,(t)}}$**:** Velocity component of dimension of particle p

$\boldsymbol{\tilde{xd}_p(t)}$**:** Personal best dimension component of particle p.

Let *f* denotes the fitness function, objective is to find minimum of f in N dimensional problem space, for iteration (t+1)

$$\boldsymbol{y_{p,j}\,(t1)}=$$

$$\left\{\begin{matrix} \boldsymbol{y_{p,j}(t)} & \boldsymbol{if\ f(x_a\,(t+1)) > f(y_p(t)} \\ \boldsymbol{x_{p,j}^{(t+1)}} & \boldsymbol{else} \end{matrix}\right\}$$

J=1,2,…….N.

(6)

P is the element of (1, s) and

$$\boldsymbol{v_{p,j}\,(t+1)} =$$

$$\boldsymbol{w(t)v_{p,j}(t) + c_1 r_{1,j}(t)\left(y_{p,j}(t) - x_{p,j}(t)\right) + c_2 r_{2,j}(t)(\hat{y}(t)\text{-}x_{p,j}(t))}$$

(7)

Where w stand for weight and c_1,c_2 are the acceleration constant.

$$\boldsymbol{x_{p,j}(t+1) = x_{p,j}(t)+v_{p,j}(t+1)} \qquad (8)$$

$$\boldsymbol{if\left(f(xx_p^{xd_a\,(t)}(t))\right)} < min\,(\boldsymbol{f(xy_p^{xd_p\,(t)}(t-1))}$$

$$\boldsymbol{Then\ xy_p^{xd_p\,(t)}(t) = xx_p^{xd_a\,(t)}(t)} \qquad (9)$$

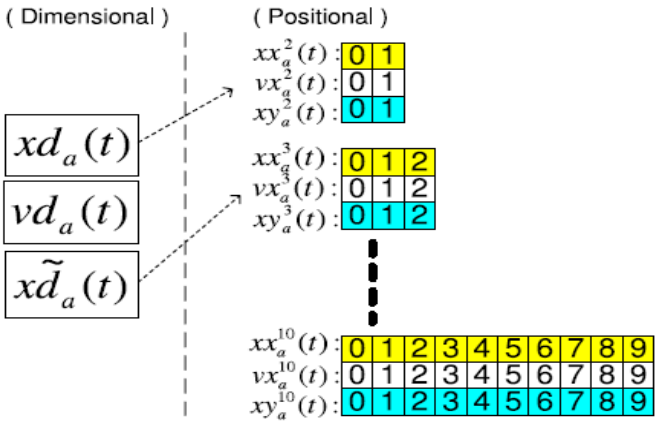

Figure 1: Positional PSO Particle

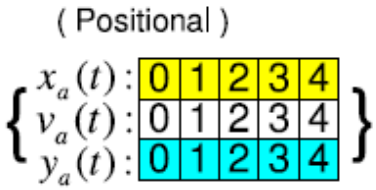

Figure 2: SI-BPN Particles

V. Conclusion

The SI BPN evolutionary technique provides a generic solution for common drawback of PSO. It can efficiently address the new weight for BPN in the environment of insufficient learning inputs and with the proper adaptation of SI BPN can evolve to the optimum network with the architecture search space. It provides a rank list of all others potential and optimum configurations. The proposed method achieves a superior generalization of BPN and PSO method with native feature of having better and faster convergence of optimum solution.

VI. References

[1] Burgess, N. “A constructive algorithm that converges for real-valued input Patterns”, International Journal of Neural Systems, 1994 Vol. 5(1), 59-66.

[2] Angeline P. J., Sauders G. M., & Pollack J. B. “An evolutionary algorithm that constructs recurrent neural networks”, IEEE Transactions on Neural Networks, 1994, 5, 54-65.

[3] Back T. & Schwefel H. P., “An overview of evolutionary algorithm for parameter optimization. Evolutionary Computation”, 1994, 1-23.

[4] Goldberg, D. “Genetic algorithms in search optimization and machine learning” Reading, MA: Addison-Wesley 1989, 1-25.

[5] Fayyad, U. M., Shapire, G. P., Smyth, P. & Uthurusamy R. Advances in knowledge discovery and data mining. Cambridge, MIT Press.

[6] Yao, X., & Liu, Y“A new evolutionary system for evolving artificial neural networks”, IEEE Transactions on Neural Networks, 8(3), 1997,694 -713.

[7] M Gori and A Tesi “On the problem of local minima in back propagation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, 1992, 76–86.

[8] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in Proceedings of IEEE International Conference on Neural Networks, vol. IV, 1995, 1942–1948.

[9] Rumelhart DE, Hinton GE, Williams R J “Learning internal representations by error propagation” MIT Press, Cambridge, 1986, 318-326.

[10] Widrod B, Hoff ME “Adoptive Switching Circuit”, IRE Eastern Electronic Show & Convention (WESCON), 1960, Convention record 4, 96-104.

[11] Lovberg, M., & Krink, T. “Extending particle swarm optimisers with selforganized criticality”, In Proc. of the IEEE congress on evolutionary computation. vol. 2, 2002 1588-1593.

[12] Riget, J., & Vesterstrom, J. S. “A diversity-guided particle swarm optimizer the ARPSO” ,Technical report. Department of Computer Science, University of Aarhus, 2002.

[13] Christopher, K. M., & Seppi, K. D. “A new approach to particle motion in swarm optimization”, In Proc. of the genetic and evolutionary computation conference, 2004, 140-150.

[14] Kaewkamnerdpong B. & Bentley P. J. “Perceptive particle swarm optimization: an investigation”, In Proc. of IEEE swarm intelligence symposium, 2005, 169-176.

[15] Richards, M., & Ventura, D. “Dynamic in particle swarm Optimization”, sixth int. conf. on computational intelligence and natural computing, 2003, 1557-1560.