Jurnal
Teknologi

Full Paper

ENHANCEMENT OF QUANTUM PARTICLE SWARM OPTIMIZATION IN ELMAN RECURRENT NETWORK WITH BOUNDED *VMAX* FUNCTION

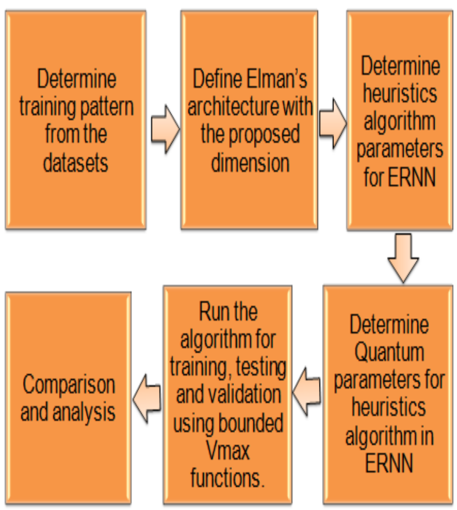
Mohamad Firdaus Ab Aziz, Siti Mariyam Hj Shamsuddin*

Soft Computing Research Group, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

Article history
Received
4 September 2016
Received in revised form
14 November 2016
Accepted
8 November 2016

*Corresponding author
mariyam@utm.my

Graphical abstract



Abstract

There are many drawbacks in BP network, such as trap into local minima and may get stuck at regions of a search space. To solve these problems, Particle Swarm Optimization (PSO) has been executed to improve ANN performance. In this study, we exploit errors optimization of Elman Recurrent Neural Network (ERNN) with a new enhance method of Particle Swarm Optimization with an addition of quantum approach to optimize the performance of both networks with bounded Vmax function. Main characteristics of Vmax function are to control the global exploration of particles in Particle Swarm Optimization and Quantum approach is used to improve the searching ability of the individual particle of PSO. The results show that for cancer dataset, Quantum Particle Swarm Optimization in Elman Recurrent Neural Network (QPSOERN) with bounded Vmax of hyperbolic tangent depicted 96.26% and Vmax sigmoid function with 96.35% which both furnishes promising outcomes and better value in terms of classification accuracy and convergence rate compared to bounded standard Vmax function with only 90.98%.

Keywords: Particle Swarm Optimization, Elman Recurrent Neural Network, Quantum, classification

1.0 INTRODUCTION

Simultaneous recurrent neural networks are class of neural network architetctures where the recurrence is instantaneous [1]. A simple recurrent network (SRN) is sometimes called an "Elman network" created by Jeff Elman and it is avariation on the multi-layer perceptron. A three-layer network is used, added with a set of "context units" in the input layer. There are connections from the hidden layer to the context units fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule called back-propagation is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units. Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that is beyond the power of a standard multi-layer perceptron.

The ability to implement associative memory is another important feature of recurrent neural network [2]. Other associative memory only store patterns but SRN store dynamics which can be stimulated by excitations of similar dynamic. Based on the limitations of the RNN network from previous studies, an enhancement of Elman Recurrent Neural Network was developed and it was successfully integrated with PSO for better classification performance. In this paper, Quantum approach is implemented in PSO to enhance the searching capability for the particles on the Elman Recurrent Neural Network algorithm.

2.0 METHODOLOGY

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [3]. It is a stochastic method that share many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [4]. The advantage of the PSO over many of the other optimization algorithm is its relative simplicity [5]. PSO is an optimization algorithm that using only primitive mathematic calculations for velocity update and particle position update.

To explain how the algorithm works in solving an optimization problem, suppose that we are trying to choose D continuous variables x1, . . . , xD to maximize a function

$$f(x1, . . . , xD).$$

Suppose also that we create a swarm of i = 1, . . . ,N particles. At all points in time, each particle i have

• A current position Xi or Xn = (xi1, . . . , xiD)

• A record of the direction it followed to get to that position Vi or Vn= (vi1, . . . , viD)

• A record of its own best previous position P_{best} = (P_{best} 1, . . . , P_{best} D)

• A record of the best previous position of any member in its group g_{best} = (g_{best} 1, . . . g_{best} D)

Given the current position of each particle, as well as the other information, the problem then becomes one of determining the direction of change for the particles. As mentioned above, this is done by reference to each particle's own experience and the experience of other members of its group. Its own experience includes the direction it came from Vi and its own best previous position. The experience of others is represented by the best previous position for any member in its group. This suggests that each particle might move in

1. The same direction that it came from Vi
2. The direction of its best previous position P_{best} – Xi
3. The direction of the best previous position of any member in its group g_{best} – Xi.

The algorithm supposes that the actual direction of change for particle i will be a weighted combination of these

$$Vn = W \times Vn + C_1 * rand1 * (G_{best,n} - Xn) + c2 * rand2 * (P_{best,n} - X n) \quad (1)$$

Where r1 and r2 are uniform [0,1] random numbers, c1 > 0 and c2 > 0 are constants called the *cognitive* and *social* parameters and w > 0 is a constant called the *inertia* parameter. For their part, n and n+1 index successive periods (generations). Given the direction of change, the new position of the particle will simply be

$$Xn = Xn + Vn \quad (2)$$

Given initial values for Xi, Vi, P_{best} and g_{best}, equations (1) and (2) will determine the subsequent path that each particle in the swarm will follow.

3.0 RESULTS AND DISCUSSION

PSO has a population of random solution, called particle, and is given a random velocity and is flown through the problem space. The particles have memory and each particle keeps track of previous best position and corresponding fitness. The previous best value is called as '*pbest*'. Thus *pbest* is related only to a particular particle. It also has another value called '*gbest*', which is the best value of all the particles *pbest* in the swarm. The basic concept of PSO technique lies in accelerating each particle towards its *pbest* and the *gbest* location at each time step. Acceleration has random weights for both *pbest* and *gbest* locations. The number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (fewer particles) must be sought [7]. PSO with well-selected parameter set can have good performance [6].

In Particle Swarm Optimization Neural Network (PSONN) number of dimension is defined as number of weight and bias that is based on the dataset and network architecture. Equation 3 illustrates the PSONN dimension. For Elman Recurrent Network, new formulation is derived, and this is based on data representation that will be fed to the network (Equation 4). The number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be investigated [5]. This is due to the concept of PSO with well-selected parameter set can have good performance [6].

Dimension original equation for feedforward neural network =
(InputUnitNo*HiddenUnitNo) + (HiddenUnitNo*OutputUnitNo) + HiddenUnitNo+ OutputUnitNo (3)

Dimension propose equation for elman recurrent neural network =
(InputUnitNo*HiddenUnitNo)+(HiddenUnitNo*ContextUnitNo)+(ContextUnitNo*HiddenUnitNo) + (HiddenUnitNo*OutputUnitNo)+HiddenUnitNo+ContextUnitNo+OutputUnitNo (4)

In Quantum Particle Swarm Optimization (QPSO), the particles have quantum behavior. Unlike conventional PSO, QPSO have dynamic behavior for each particle. Therefore, if individual particles in a PSO system have quantum behavior, the performance of PSO will be far better from the classical PSO [8]. The particle moves along a determined trajectory following Newtonian mechanics[9]. In [10], the state of the particle is depicted by wave function Ψ (x, *t*). The probability density function and distribution function are obtained by solving the Schrődinger equation. By using the Monte Carlo method, the position of the ith particle can be obtained using:

$$x_{ij}(t) = p_j \pm L_{ij} \ln\left(\frac{1}{u}\right), \quad (5)$$

where

$$p_j = \frac{p(c_1 P_{id} + c_2 P_{gd})}{(c_1 + c_2)},$$

c_1 and c_2 are random numbers and u is also a random number uniformly distributed between 0 and 1.

The value of L is given by

$$L_{ij}(t+1) = 2 * \beta \left| m_{best_j} - x(t_{ij}) \right|,$$

where

$$m_{best_j} = \frac{1}{M} \sum_{i=1}^{M} p_{ij}, \quad j = 1,2,\ldots,D,$$

and

$$m_{best} = \left(m_{best_1}, m_{best_2}, m_{best_2}, \ldots, m_{best_p}\right),$$

where,

β is Contraction-Expansion coefficient which can be used to control the convergence of the PSO algorithm [9],
M is the size of the population. The new position is now given as:

$$x\,(t+1) = p \;+ \beta * \left| m_{best} - x(t) \right| * \ln\left(\frac{1}{u}\right)$$

The experiment was done according to the following flow diagram

Initialize the Swarm
Do
Calculate m best by equation

$$mbest = \frac{1}{M}\sum_{i=1}^{M} P_i = \left(\frac{1}{M}\sum_{i=1}^{M} P_{i1},\ \frac{1}{M}\sum_{i=1}^{M} P_{i2},\ \cdots,\ \frac{1}{M}\sum_{i=1}^{M} P_{id}\right)$$

Update particles position using equation
x(*t* +1) = *p* + β *| *mbest* − x(*t*) | *ln(1/ *u*)
where
p (c1 *Pid* + c2*Pgd*) / (c1+ c2)
Update Pbest
Update Pgbest
While maximum iteration is reached

Figure 1 Flow of a QPSO algorithm

In particle swarm algorithm, changes in the velocity are stochastic and one undesirable effect of this is that the uncontrolled particle's trajectory can be expanded into wider cycles in the problems space. This can result in swarm explosion and thus divergence. To address this drawback of the original PSO model, a threshold called Vmax on the particles velocity was introduced. The motivation was to dampen the oscillation of the particles by restricting them to a maximum allowed value. The threshold as applied to the particle velocity as follows [15]:

If
$v_{id}(t+1) > V_{max}$
then
$v_{id}(t+1) = V_{max}$ (6)

If
$v_{id}(t+1) < -V_{max}$
then
$v_{id}(t+1) = -V_{max}$ (7)

If Vmax is too big, particles may flurry too far at once and miss good solutions. If Vmax is too small, particles may be limited to a local area. However, the choice of Vmax value is problem dependent. Based on the researches' experience, Vmax can be set at 10 to 20% of the range of each variable or proportion to the range of the problem [16].

On the other hand, the parameters selection of PSONN plays an important role in optimization [17]. A single PSONN parameter choice has a tremendous effect on the rate of convergence. In [18], optimal PSONN parameters are determined by trial and error experimentations. Optimal refers to the set of PSONN parameters that yield faster network convergence. In PSONN, number of dimension is referred to the number of weight and bias that is based on the dataset and ANN architecture. Equation (8) illustrates the calculation of PSONN dimension in this study. Inertia weight, *w* and velocity maximum Vmax control the exploration and exploitation of the search space [18]. If Vmax is too high, then particles will move beyond good solution and if Vmax is too low, then particles will be trapped in local minima [19]. Consequently, bounded function such as sigmoid function and hyperbolic tangent function are proposed in this study. Bounded PSO Vmax function is suggested to control global exploration of particles, increase the convergence and classification rate.

$$Dimension = (input \times hidden) + (hidden \times output) + hidden_{bias} + output_{bias} \quad (8)$$

In this study, Vmax sigmoid function is given as in equation (9) and Vmax hyperbolic tangent function is depicted in equation (10). These bounded functions are implemented for particles global exploration besides standard Vmax function in conventional PSO.

$$v_{id}(t+1) = w \times v_{id}(t) + C_1 \times \left(\frac{1}{1+e^{-rand()}} \right) \times [P_{id}(t) - x_{id}(t)] + C_2 \times \left(\frac{1}{1+e^{-rand()}} \right) \times [P_{gd}(t) - x_{id}(t)] \quad (9)$$

$$v_{id}(t+1) = w \times v_{id}(t) + C_1 \times \left(\frac{e^{rand()} - e^{-rand()}}{e^{rand()} + e^{-rand()}} \right) \times [P_{id}(t) - x_{id}(t)] + C_2 \times \left(\frac{e^{rand()} - e^{-rand()}}{e^{rand()} + e^{-rand()}} \right) \times [P_{gd}(t) - x_{id}(t)] \quad (10)$$

Quantum Particle Swarm Optimization in Elman Recurrent Neural Network (QPSOERN) with bounded Vmax function are developed and tested on XOR, Cancer and Iris data. The results for each dataset are compared and analyzed based on the convergence rate and classification performance. The experiment was done for 10 time for each dataset and the average value was calculated. The network structure for QPSOERN is illustrated in Table 1. Table 2 shows the results of QPSOERN with various bounded vmax function, standard Vmax function, sigmoid function and hyperbolic tangent function. The convergence time for vmax function is the lowest; 56.1 second, sigmoid function is 64.7 second, while hyperbolic tangent is just 55.7 second. However, both algorithms have converged successfully under pre-specified error tolerance. As well, sigmoid function yields better results of 82.9% compared to Vmax function with 80.77 % and hyperbolic tangent 80.22%. Error convergence for sigmoid function is the lowest among the others with 0.4733.

Table 1 Network Structure for QPSOERN on XOR dataset

Subject	Value
Network Size (nodes)	3 input layer 5 hidden layer 5 context unit 1 output layer
Data Pattern	8
The minimum value of weight	-0.50
The maximum value of weight	0.50
PSO parameters	Use 20 particles
C1	2
C2	2
Δt	0.1
problem dimension (total number of weight and bias)	81
stop conditions minimum error	0.005
maximum iteration	500

Table 2 QPSOERN on XOR dataset

	Vmax Function	Sigmoid function	Hyperbolic tangent function
Error Convergence	0.5031	0.4602	0.4733
Convergence Time (Sec)	56.1	64.7	55.7
Classification (%)	80.77	82.90	80.22

Table 3 depicts the network structure for ERNPSO cancer dataset. While Table 4 illustrates the results of all network structures. In Cancer learning, using hyperbolic tangent function it takes 142.3 seconds to converge compared to standard vmax function and sigmoid function which obtains 154.8 second and 155.5 second to converge. For the classification rates, QPSOERN discloses better results with sigmoid function 96.345%, compared to the standard hyperbolic tangent function and vmax function with 96.263% and 90.98%..

Table 3 Network Structure for QPSOERN on Cancer dataset

Subject	Value
Network Size (nodes)	9 input layer , 19 hidden layer 19 context unit 1 output layer
Data Pattern	150
The minimum value of weight	-0.50
The maximum value of weight	0.50
PSO parameters	Use 20 particles
C1	2
C2	2
Δt	0.1
problem dimension (total number of weight and bias)	951
stop conditions minimum error	0.005
maximum iteration	500

Table 4 QPSOERN on Cancer Dataset

	Vmax Function	Sigmoid function	Hyperbolic tangent function
Error Convergence	1.45	1.7	1.5
Convergence Time (Sec)	154.8	155.5	142.3
Classification (%)	90.98	96.35	96.26

Table 5 depicts the network structure for QPSOERN Iris dataset. While Table 6 illustrates the results of all network structures. In Iris learning, vmax function average converge time is 24 second, hyperbolic tangent function takes 25.74 seconds, and sigmoid function which obtains 31.27 second. For the classification rates, QPSOERN discloses better results with hyperbolic tangent function 76.77%, compared to the standard sigmoid function and vmax function with 74.94% and 74.66%.

Table 5 Network Structure for QPSOERN on Iris dataset

Subject	Value
Network Size (nodes)	4 input layer , 9 hidden layer 9 context unit 3 output layer
Data Pattern	120
The minimum value of weight	-0.50
The maximum value of weight	0.50
PSO parameters	Use 20 particles
C1	2
C2	2
Δt	0.1
problem dimension (total number of weight and bias)	246
stop conditions minimum error	0.005
maximum iteration	500

Table 6 QPSOERN on Iris Dataset

	Vmax Function	Sigmoid function	Hyperbolic tangent function
Error Convergence	24	31.27	25.74
Convergence Time (Sec)	110.2	118.2	117.3
Classification (%)	74.66	74.97	76.77

4.0 CONCLUSION

The study is carried out to analyze the effectiveness of QPSO in optimizing the Elman Recurrent Neural Network Structure. Based on the results, it shows that QPSOERN has better accuracy using sigmoid function and hyperbolic tangent then the ordinary vmax function. The accuracy is much higher for both Iris and Cancer dataset in using sigmoid function and hyperbolic tangent, but not in XOR dataset. The accuracy is better with QPSOERN using both function tangent, which indicates that quantum based approach is better for large amount of dataset and vmax function are good for less dataset.

The main motivation of the proposed algorithm is towards its pbest and pgbest location where it is actually locating the optimum best position for the particle. So the larger amount of data depicts that the easier for the algorithm to search for the local and global best position in the neighborhood. Similar to original particle swarm optimization, all particles in the QPSOERN are also converged to the global best position. With the dynamic behavior of quantum it shows that the bigger number of datasets it upfront with the easier it achieve the optimum best position. Furthermore, with the dynamic behavior of quantum approach, the proposed method also converges much faster compared to the conventional PSO.

Acknowledgement

This work is supported by Ministry of Science, Technology and Innovation (MOSTI, SF 79311). Authors would like to thank Research Management Centre (RMC) Universiti Teknologi Malaysia, for the research activities and *Soft Computing Research Group* (SCRG) for the support in making this study a success.

References

[1] Geib, J., & Serpen, G. 2004. Computational Promise Of Simultaneous Recurrent Network With A Stochastic Search Mechanism. Proc. *IEEE International Joint Conference On Neural Networks.* 3: 2239-2244.
[2] Michel, A., & Farrell, J. 1990. Associative Memories Via Artificial Neural Networks. *IEEE Control Systems Magazine.* 10(3): 6-17.
[3] Robinson, B. 2003. *Evolutionary Artificial Neural Networks in Problem Solving*. University of Manitoba,
[4] Luger, G. F. 2002. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 4th ed. Addison Wesley/Pearson Education Limited, Harlow, England. 417-473.
[5] G. Dib. 2000. Artificial Neural Networks. *ELE82 Biomedical Engineering Seminar 1*. Kingston.
[6] Krose, B. and Van der Smagt, P. 1996. *An Introduction to Neural Networks.* 8th ed., University of Amsterdam.
[7] R. C. Eberhart and J. Kennedy. 1995. A New Optimizer using Particle Swarm Theory. *Proceedings of the Sixth International Symposium on Micromachine and Human Science*. 39-43.
[8] F. Van den Bergh. 1999. Particle Swarm Weight Initialization In Multi-Layer Perceptron Artificial Neural Networks. *Accepted for ICAI*. Durban, South Africa. 41-45.
[9] J. L. Elman. 1990. Finding Structure In Time. *Cognitive Science.* 14: 179-211.
[10] S. Ujjin and P. J. Bentley. 2003. Particle Swarm Optimization Recommender System. *IEEE Swarm Intelligence Symposium 2003*. 24-26 April. 124-131.
[11] R. C. Eberhart and Y. Shi. 1998. Comparison between Genetic Algorithms and Particle Swarm Optimization. *Proceedings of the 7th International Conference on Evolutionary Programming VII*, March 25-27 1. 611-616.
[12] Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford,
[13] Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.
[14] F. Van den Bergh, and A. P. Engelbrecht. 2000. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal.* 26: 84-90.
[15] Ahmed, T. 2004. *Adaptive Particle Swarm Optimizer for Dynamic Environments.* The University Of Texas: Master Thesis.
[16] Lin, L. I. 2005. *Particle Swarm Optimization for Solving Constraint Satisfaction Problems*. Simon Fraser University: Master Thesis.
[17] Shi, Y., Eberhart, R. 1998. Parameter Selection in Particle Swarm Optimization. *Proc. Seventh Annual Conf. on Evolutionary Programming*. 591-601.

[18] Shamsuddin, S. M. 2004. *Lecture Note Ad Advanced Artificial Intelligence: Number of Hidden Neurons*. Universiti Teknologi Malaysia. Unpublished.

[19] Shi, Y., Eberhart, R. 1998. A Modified Particle Swarm Optimizer. *IEEE International Conf. An Evolutionary Computation Anchorage, Alaska. USA*. 69-73.