Jurnal Teknologi, 49(D) Dis. 2008: 13–26

PARTICLE SWARM OPTIMIZATION FOR NEURAL NETWORK LEARNING ENHANCEMENT

HAZA NUZLY ABDULL HAMED[1], SITI MARIYAM SHAMSUDDIN[2] & NAOMIE SALIM[3]

Abstract. Backpropagation (BP) algorithm is widely used to solve many real world problems by using the concept of Multilayer Perceptron (MLP). However, major disadvantages of BP are its convergence rate is relatively slow and always being trapped at the local minima. To overcome this problem, Genetic Algorithm (GA) has been used to determine optimal value for BP parameters such as learning and momentum rate and, also for weight optimization. Although GA has successfully improved Backpropagation Neural Network (BPNN) learning, there are still some issues such as longer training time to produce the output and usage of complex functions in selection, crossover and mutation calculation. In this study, Particle Swarm Optimization (PSO) algorithm has been chosen and applied in feedforward neural network to enhance the learning process in terms of convergence rate and classification accuracy. Two experiments have been conducted; Particle Swarm Optimization Feedforward Neural Network (PSONN) and Genetic Algorithm Backpropagation Neural Network (GANN). The results show that PSONN give promising results in terms of convergence rate and classification accuracy compared to GANN.

Keywords: Particle swarm; neural network; genetic algorithm; backpropagation; swarm intelligence

Abstrak. Algoritma rambatan balik telah digunakan secara meluas dalam menyelesaikan pelbagai masalah dengan menggunakan konsep perceptron multi aras. Namun begitu terdapat banyak isu utama pada algoritma ini seperti kadar penumpuan yang lambat dan kekerapan terperangkap di dalam minimum setempat. Bagi mengatasi masalah ini, Algoritma Genetik (AG) digunakan untuk menentukan nilai yang optimal bagi mendapatkan parameter yang sesuai seperti kadar pembelajaran dan kadar momentum serta pengoptimuman pemberat. Walaupun AG berjaya meningkatkan keupayaan pembelajaran bagi Rangkaian Neural (RN) menggunakan rambatan balik, masih terdapat beberapa masalah lain seperti latihan untuk mengeluarkan output mengambil masa yang lama dan penggunaan fungsi yang rumit seperti perhitungan pilihan, silangan dan mutasi. Kajian ini mengemukakan teknik pengoptimuman yang terkini, iaitu pengoptimuman partikel secara berkumpulan yang dicerap di dalam proses pembelajaran RN, bagi meningkatkan masa penumpuan dan ketepatan pengelasan. Dua uji kaji telah dilaksanakan, iaitu RN ke hadapan menggunakan pengoptimuman partikel secara berkumpulan dan RN rambatan balik menggunakan AG. Hasil kajian mendapati bahawa RN ke hadapan dengan pengoptimuman partikel secara berkumpulan memberikan keputusan yang lebih baik dari aspek masa penumpuan dan ketepatan pengelasan, berbanding dengan RN rambatan balik menggunakan AG.

Kata kunci: Partikel berkumpulan; rangkaian neural; algoritma genetik; rambatan balik; kepintaran berkumpulan

[1,2&3]Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia
Email: haza@kuittho.edu.my, mariyam@utm.my, naomie@utm.my

1.0 INTRODUCTION

Artificial Neural Network (ANN) or commonly referred as Neural Network (NN) is an information processing paradigm that is inspired by the way biological nervous systems process information. Processing power in ANN allows the network to learn and adapt, in addition to making it particularly well suited to tasks such as classification, pattern recognition, memory recall, prediction, optimization, and noise filtering [1]. The primary significance for a NN is the ability of the network to learn from its environment and to improve its performance through learning [2]. Learning is a process of modifying the weights and biases to the neurons and continued until a preset condition is met such as using a pre-defined error function. The most familiar technique in NN learning is called Backpropogation (BP) algorithm. However, the major disadvantages of BP are its convergence rate is relatively slow [3] and the solution being trapped at the local minima. There are many solutions proposed by NN researchers to overcome the issues of slow convergence rate and trapping at the local minima. Genetic Algorithm (GA) is one of the algorithms proposed to determine the best BP parameters value and weight optimization. Although GA has successfully improved BPNN learning as demonstrated by many researchers such as Zhang and Ciesielski [4] and Randall and Naheel [5], there are still some issues such as longer training time to produce the output and usage of complex functions in selection, crossover and mutation calculation. According to Song and Gu [6], due to the convenience of realization and promising optimization ability in various problems, Particle Swarm Optimization algorithm has been paid more and more attention to by researchers. Lee *et al.* [7] have used PSO and GA for excess return evaluation in stock market. Based on their experiment, it is proven that PSO algorithm is better compared to GA. Another study by Al-kazemi and Mohan [8] proposed Multi-Phase Particle Swarm Optimization algorithm (MPPSO) to train feedforward neural network. Zhang *et al.* [9] applied PSO in ANN to two real problems in medical domain; breast cancer and heart disease. The result shows that PSO has better accuracy in classifying these data.

Based on the experiments conducted by several researchers, it shows that PSO provides better result. In this study PSO is employed to investigate the convergence speed and the classification accuracy of NN learning compared to GA-based NN.

2.0 ARTIFICIAL NEURAL NETWORK AND SWARM INTELLIGENCE

Artificial Neural Network (ANN) consists of a parallel collection of simple processing units (neurons/nodes) arranged and interconnected in a network topology [10] as shown in Figure 1. Zhang *et al.* [9] addressed that the information processing capability ANN is closely related to its architecture and weights. Figure 2 shows the interconnection between nodes which is usually referred as a fully connected network

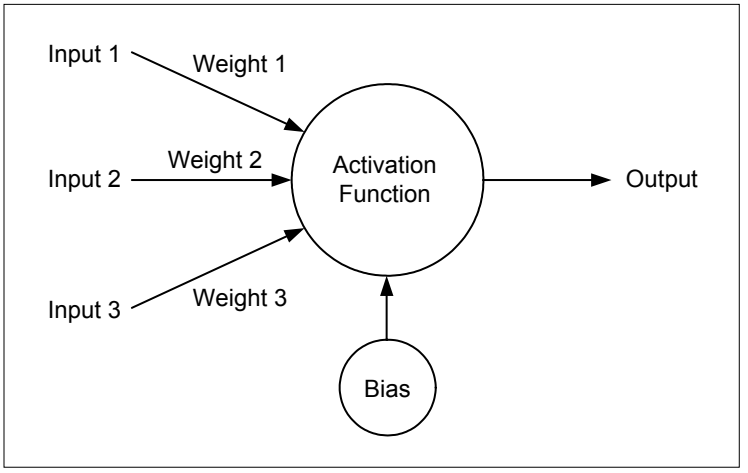

Figure 1 Artificial neural network

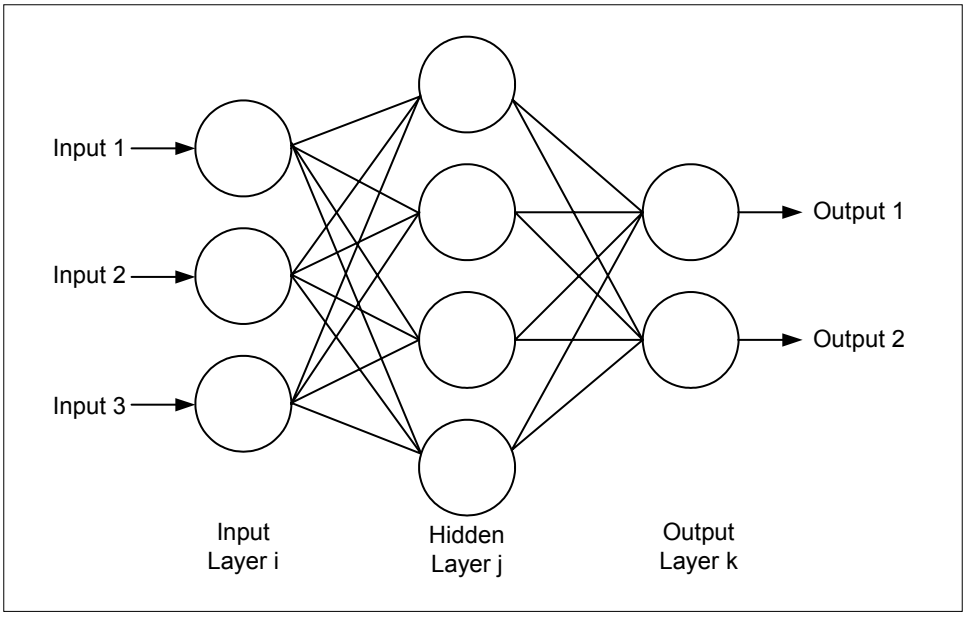

Figure 2 Multilayer perceptron

or multilayer perceptron (MLP). MLP network can be used with great success to solve both classification and function approximation problems [11]. There are two types of learning networks which are supervised learning and unsupervised or self-organizing learning. Supervised learning is when the input and desired output are provided while for unsupervised learning, only input data is provided to the network. Supervised learning has been chosen for this study.

Swarm Intelligence (SI) is the latest soft computing technique introduced in 1995 by Russell C. Eberhart and James Kennedy [12]. SI is defined as any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of the social insect colonies and other animal societies such as ant colonies and bird flocking [13]. There are two major techniques in SI; Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). The ACO algorithm is a probabilistic technique for solving computational problems, which can be reduced to finding good paths through graphs. They are inspired by the behaviour of ants in finding paths from the colony to food. On the other hand, PSO is a technique where several particles (solutions) interacting between each other to find the best solutions. In this study, PSO is chosen as learning algorithm in ANN.

3.0 PARTICLE SWARM OPTIMIZATION AND GENETIC ALGORITHM

The original Particle Swarm Optimization (PSO) algorithm is discovered through simplified social model simulation [14]. PSO is a simple concept adapted from nature decentralized and self-organized systems such as choreography of a bird flock and fish school. PSO is a population-based algorithm in which individual particles work together to solve a given problem. The Population (or swarm) and the member called particle is initialized by assigning random positions and velocities. The potential solutions are then flown through the hyperspace. The particles learn over time in response to their own experience and the experience of the other particles in their group. According to Eberhart and Shi [15], each particle keeps track of its best fitness position in hyperspace that has been achieved. This value is called personal best or pbest. The best value obtained by any particle in the population is called global best or gbest. During each epoch (or iteration), every particle is accelerated towards its own personal best as well as in the direction of the global best position. This is achieved by calculating a new velocity term for each particle based on the distance from its personal best, as well as its distance from the global best position. These two components (personal and global velocities) are randomly weighted to produce a new velocity value for this particle, which will in turn affect the next position of the particle during the next epoch [16]. Based on the velocities calculation, all particles will move to the best solutions and this process is repeated until the stop condition is met.

PSO is an optimization algorithm that using only primitive mathematical calculations. The advantage of the PSO over many of the other optimization algorithms is its relative simplicity [16]. According to Jones [17], there are only two equations in PSO, the movement Equation (Equation (1)) and velocity update Equation (Equation (2)). The movement equation provides for the actual movement of the particles using their specific vector velocity value, while the velocity updates equation provides

for velocity vector adjustment that guide every particle in order to move to the best solution based on gbest and pbest values.

The movement equation is given as:

$$x_n = x_n + v_n * \Delta t \tag{1}$$

where

x_n is the current position value,
v_n is the current velocity value,
Δt is the time interval.

The velocity updating equation is given as:

$$v_n = v_n + c_1 * rand() * \left(g_{best,n} - x_n\right) + c_2 * rand() * \left(p_{best,n} - x_n\right) \tag{2}$$

where,

v_n is the current velocity value,
$g_{best,n}$ is the current gbest value,
$p_{best,n}$ is the current pbest value,
$rand\ ()$ is the random number between 0 –1,
c_1 is the first constant,
c_2 is the second constant,
x_n is the current position value.

Equation (1) is performed for each element of the position (x) and velocity (v) vector. The Δt parameter defines the discrete time interval over which the particle will move. The result is a new position for the particle. In equation (2), it subtracts the dimensional element from the dimension from the best vector and multiplies by a random number (between 0.0 and 1.0) and acceleration constant (C_1 and C_2). The sum of these products is then added to the velocity for the given dimension of the vector. This process is performed for each element of the velocity vector. The random numbers provide an amount of randomness in the path to help the particle move throughout the solution space. The C_1 and C_2 acceleration constant provide some controls to the equation to define which should be given more emphasis on the path (global or personal best).

On the other hand, Genetic Algorithm (GA) introduced by John Holland [18] is a probabilistic optimization algorithm. The original idea came from biological evolution process in chromosomes. GA exploits the idea of the survival of fittest where best solutions are recombined with each other to form new better solutions. There are three processes in GA; selection, crossover and mutation. In standard GA, the population is a set of individual number. Each individual represents the chromosome of a life form. There is a function that determines the fitness of each

individual from the population to reproduce. The two selected chromosomes are combined in crossover process and split into two new individuals, and this is known as mutation. The process is repeated until the stopping condition is met.

4.0 A FRAMEWORK OF PARTICLE SWARM OPTIMIZATION FEEDFORWARD NEURAL NETWORK AND GENETIC ALGORITHM BACKPROPAGATION NEURAL NETWORK

This study is conducted to see the performance of convergence rate and classification accuracy between Particle Swarm Optimization Feedforward Neural Network (PSONN) and Genetic Algorithm Backpropagation Neural Network (GANN) using UCI machine dataset; XOR, Cancer and Iris dataset [19]. Figure 3 is a proposed framework of the study.

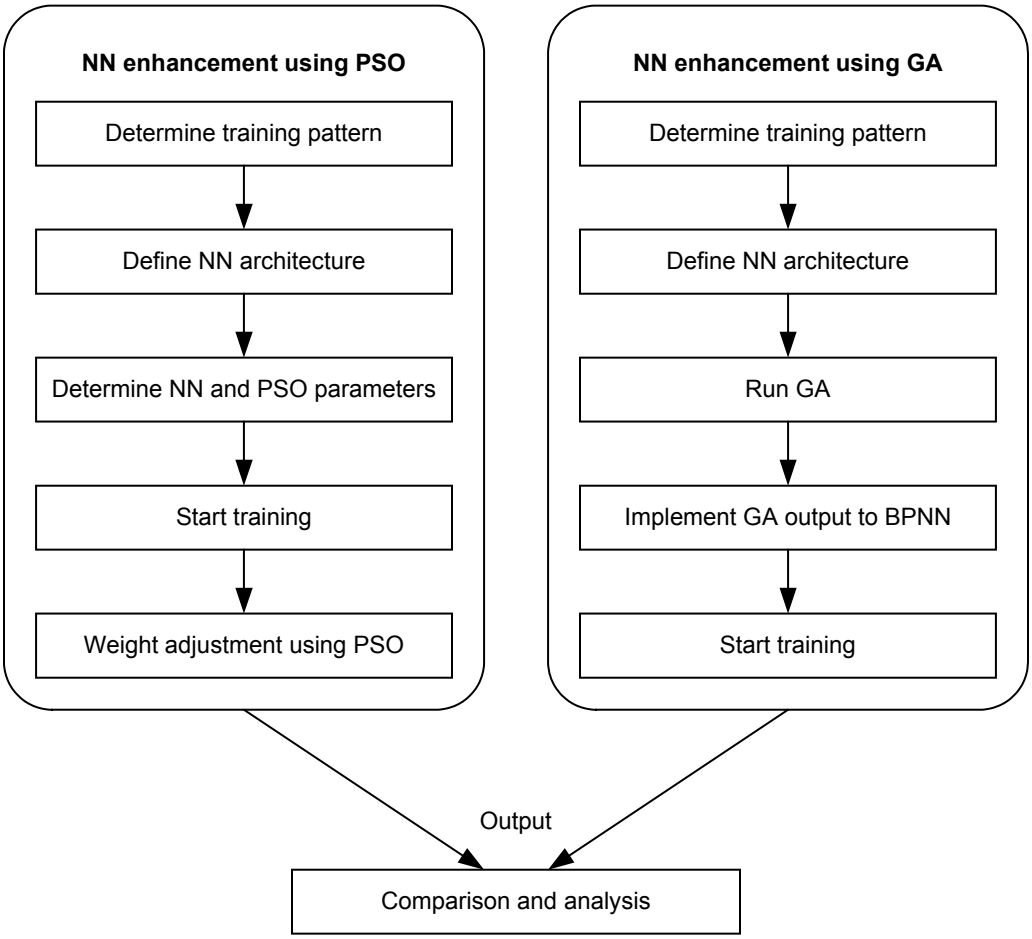

Figure 3 A framework of the study

In this study, both algorithms use the same NN architecture as described below:

(a) 3 Layer of ANN as shown in Table 1.

Table 1 ANN architecture

Type of dataset	Data pattern	Input	Hidden	Output
XOR	4	2	5	1
Cancer	150	9	19	1
Iris	120	4	9	3

(b) There are many suggestions by researcher to determine the suitable number of hidden nodes. In this study, number of hidden nodes are determined using Kolmogorov Theorem as shown in Equation (3):

$$hidden = 2n + 1, \text{ where } n = \text{number of input.} \tag{3}$$

(c) A common activation function in ANN, Sigmoid Activation/Transfer Function has been used as shown in Equation (4).

$$f(x) = \frac{1}{\left(1 + e^{-x}\right)}, \text{ where } x \text{ is an input.} \tag{4}$$

In this study, PSO is applied to feedforward neural network based on Al-kazemi and Mohan [8], where the position of each particle in swarm represents a set of weight for the current epoch or iteration. The dimensionality of each particle is the number of weights associated with the network. The particle moves within the weight space attempting to minimize learning error. Changing the position means updating the weight of the network in order to reduce the error of the current epoch. In each epoch, the particles update their position by calculating the new velocity, and move to the new position. The new position is a set of new weights used to obtain the new error. This process is repeated and the particle with the lowest learning error is considered as the global best particle. The training process continues until satisfactory error is achieved by the best particle or computational limits (maximum iteration) are exceeded. When the training ends, the weights are used to calculate the classification error for the training patterns. The same set of weights is used to test the network using new patterns.

There is no backpropagation concept in PSONN where the feedforward NN produced the learning error (particle fitness) based on set of weight and bias (PSO positions). The pbest value (each particle's lowest learning error) and gbest value (lowest learning error found in entire learning process) are applied to Equation (2)

to produce a value for positions adjustment to the best solution or targeted learning error. The velocity value obtained from equation (2) is added to current position to produce the new set of particle positions using equation (1). This is the concept of particle movement or 'fly' in PSO, where all particles move to new location based on the new particle position. These new sets of positions are used for producing new learning error in feedforward NN. This process is repeated until the stop conditions are met (minimum learning error or maximum number of iteration). Learning errors are calculated in feedforward NN starting from input nodes to hidden nodes and output nodes, and NN make a backward pass from output nodes to hidden nodes and input nodes to produce new set of weights. Figure 4 shows PSONN learning process.

PSO with well-selected parameter set can have good performance [14]. Jones [17] gave four basic parameters in PSO, and this includes:

(a) acceleration constants for gbest (C_1),
(b) acceleration constants for pbest (C_2),
(c) time interval (Δt),
(d) number of particle.

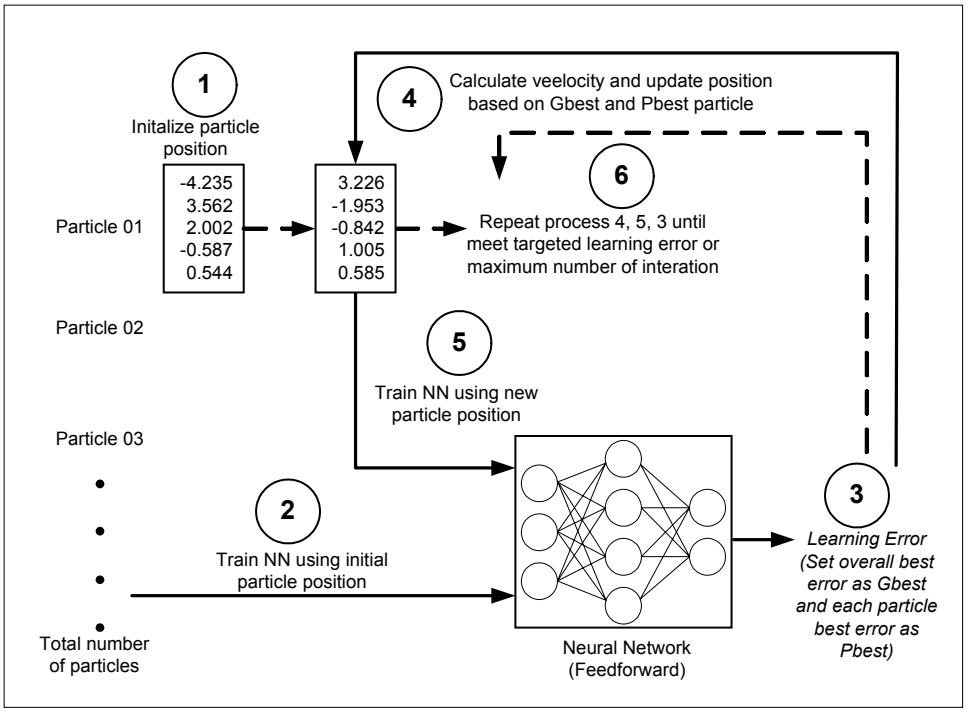

Figure 4 PSONN learning process

The acceleration constants are used to define how particles swarm in the simulation. According to Eberhart and Shi [15], the acceleration constant C_1 and C_2 represent the stochastic acceleration that pulls each particle toward pbest and gbest position. The Δt parameter defines the time interval over which movement takes place in the solution space. For the number of particles in the simulation or swarm, the more particles that are presented, the greater the amount of space that is covered in the problem, thus optimization becomes slower. Besides these basic parameters, there are also some other parameters that depend on the problems such as particle dimension, number of particles and stopping condition. Table 2 shows the parameters that have been used in this study, which are based on Eberhart and Shi [15] and Jones [17].

Table 2 PSONN parameters

Parameter	**Value**	**Notes**
C_1	2.0	Suggested by Eberhart and Shi [15]
C_2	2.0	Suggested by Eberhart and Shi [15]
Δt	0.1	Suggested by Jones [17]
Number of particles	20	Must balance between variety (more particles) and speed (fewer particles) [16]
Problem dimension	XOR : 21 Cancer : 210 Iris : 75	$Dimension=(input*hidden)+(hidden*output)+hidden_{bias}+output_{bias}$
Range of particles	Not specified	Particle free to move anywhere
Stop condition	NN minimum error or max number of iteration	
Initial position	Random	Suggested by Eberhart and Shi [15]
Initial velocity	Random	Suggested by Eberhart and Shi [15]

GA is usually applied in ANN to optimize the network because of its efficiency in giving the best parameters such as learning rate and momentum rate to avoid from being trapped in a local minima and making the convergence speed faster. GA also has been used to produce best NN architecture and for NN weight optimization. According to Randall and Naheel [5], GA starts at multiple random points (initial population) when searching for a solution. Each solution is then evaluated based on the objective function. These solutions are selected for the second generation based on their performance. This operation keeps all the weights that are included in the previous generation but allows them to be rearranged. If the weights are good, they still exist in the population. The next operation is mutation, which can randomly replace the weights in the population, for a solution to escape from local minima.

Upon completion, the generation is ready for evaluation and the process continues until the best solution is found. Figure 5 shows the encoding process from ANN weights to GA chromosome.

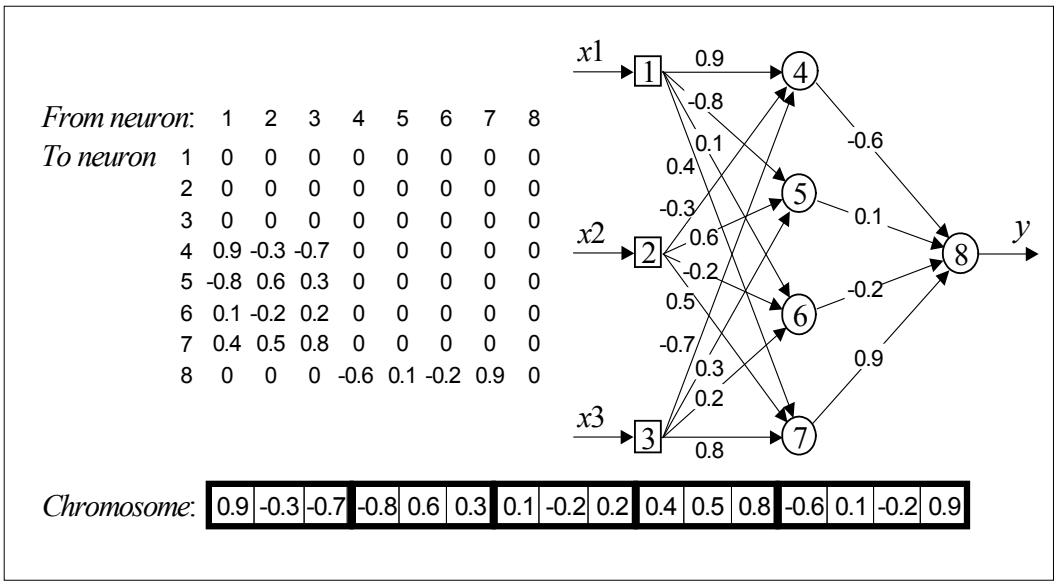

Figure 5 Encoding a set of weights in a chromosome

Weights in BP are replaced with weights from GA, while η and α (learning and momentum rate) value in BP are replaced with value from GA process. GA algorithm is proven to be effective to guide the ANN learning, thus, it is widely used in many real world applications. As a result, PSO technique is proposed to see the performance and the results are compared with GANN learning performance. Table 3 shows the GA parameters that have been used in this study. The parameters value that has been chosen is the optimal value in order to generate result in NN learning.

Table 3 GA parameters

Parameter	**Value**
Maximum population	100
Population size	50
Maximum generation	100
Crossover rate	0.6
Mutation rate	0.02
Reproduction rate	0.6

5.0 EXPERIMENTAL RESULT

Experiments are conducted using three dataset; XOR, Cancer and Iris on both algorithms with pre-defined error is 0.005 as stopping condition. The results for each dataset are compared and analysed based on the convergence rate and classification performance.

Result for XOR is shown in Table 4 and Figure 6. The result shows that PSONN convergence time is 12 seconds at iteration 51 compared to GANN, where it takes 37 seconds for overall learning process. Both algorithms are converged using the minimum error criteria. For the correct classification percentage, it shows that PSONN result is better than GANN with 95.17% compared to 85.66% in GANN. Figure 6 shows the learning process where both algorithms attempt to reach the learning stop condition. In PSONN, 20 particles work together to find the lowest error (gbest) at each iteration and consistently reduce the error at each iteration. While in GANN, it seems that the error starts to decrease at iteration 37, and stop at a specified condition in a short time.

Table 4 XOR learning result

	PSONN	GANN
Learning iteration	51	61
Error convergence	0.00473763	0.04125
Convergence time	12 Sec	37 Sec
Correct classification	95.17%	85.66%

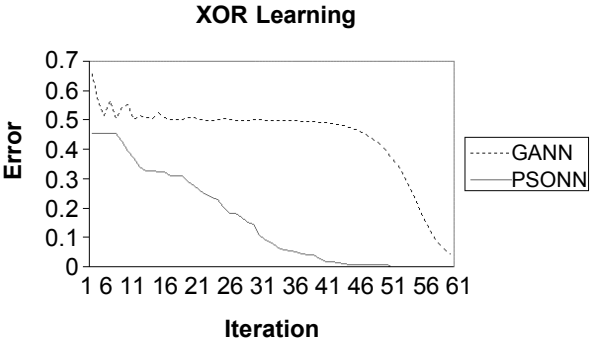

Figure 6 Convergence rates of XOR dataset

In Cancer dataset, PSONN takes 110 seconds compared to 273 second in GANN to converge as shown in Table 5. In this experiment, PSONN manages to converge using minimum error at iteration 196, while GANN converge at a maximum iteration of 10000. For the correct classification percentage, it shows that PSONN result is better than GANN with 99.75% compared to 99.28% in GANN. Figure 7 shows PSONN significantly reduce the error at small number of iteration compared to GANN.

Table 5 Cancer learning result

	PSONN	GANN
Learning Iteration	196	10000
Error Convergence	0.00495528	0.50049
Convergence Time	110 Sec	273 Sec
Correct Classification	99.75%	99.28%

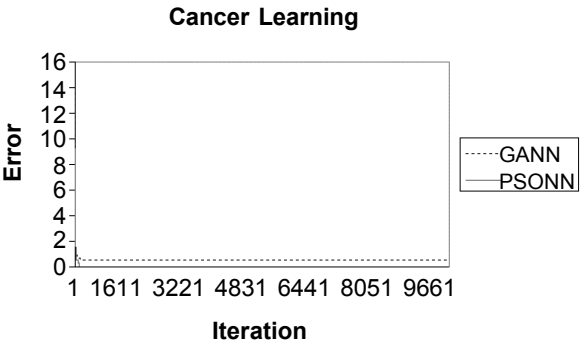

Figure 7 Convergence rates of Cancer dataset

For Iris learning, both algorithms converge using the maximum number of pre-specified iteration. PSONN takes 173 second to converge at minimum error of 4.18404 while minimum error for GANN is 1.88831 at 10000 iterations. Table 6 shows that GANN is better in classification than PSONN with 97.72% compared to 92.11% in GANN. However, PSONN convergence is faster compared to GANN (Figure 8).

Table 6 Iris learning result

	PSONN	GANN
Learning Iteration	1000	10000
Error Convergence	4.18404	1.88831
Convergence Time	173 Sec	256 Sec
Correct Classification	92.11%	97.72%

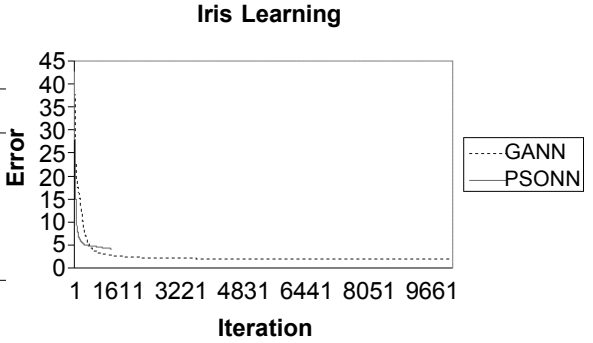

Figure 8 Convergence rates of Iris dataset

6.0 DISCUSSION

The experiments show that PSONN produces acceptable results in terms of convergence time and classification accuracy. However, the convergence time in PSONN can be faster if the number of text file used in the PSONN program is reduced. This is due to the implementation of PSONN program with several text files for checking and verification purposes such as text file for checking gbest error in every iteration and text file to record every particle with current fitness function for graph drawing. This process causes slow convergence time compared to PSONN without those functions. In PSONN, network architecture and selection of network

parameters for the dataset influence the convergence and the performance of network learning. By reducing the hidden nodes, it minimizes the number of weight (position) and also reduces the problem dimension. In this study, several times that PSONN is generated with reduced number of hidden nodes, and the results have proven that the learning becomes faster. However, to have fair comparison, Kolmogorov theorem is chosen, and both algorithms should have similar network architecture. Choosing PSONN parameters also depend on the problem and dataset to be optimized. This parameter can be adjusted to achieve better optimization. For GANN, learning rate and momentum rate which are critical for standard BP learning is provided by GA algorithm with a set of weights. This is to ensure that the convergence time is faster with better results. However, the process including parameters selection in GA takes longer time compared to overall process in PSONN.

7.0 CONCLUSION

In this study, PSO and GA are successfully applied in neural network and has been tested using XOR, Cancer and Iris datasets. The analysis is done by comparing classification results for each dataset produced by PSONN and GANN. Most important finding in this study is PSO is a simple optimization algorithm with less mathematical equation that can be effectively applied in neural network with faster convergence rate and promising classification accuracy compared to GANN. This study also shows that implementation of GA as parameter tuning to BP algorithm does not give better improvement to the convergence rate. For future works, PSO can be enhance by using multiple group of particle as suggested by Al-kazemi and Mohan [8] or modify the velocity update Equation (2) by using the inertia weight as suggested by Eberhart and Shi [15].

REFERENCES

[1] Luger, G. F. 2002. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 4th ed. Harlow, England: Addison-Wesley/Pearson Education Limited.

[2] Haykin, S. 1999. *Neural Network. A Comprehensive Foundation*. 2nd Ed. Upper Saddle River, NJ: Prentice Hall.

[3] Zweiri, Y. H., J. F. Whidborne and L. D. Sceviratne. 2003. A Three-term Backpropagation Algorithm. *Neurocomputing*. 50: 305-318.

[4] Zhang, M. and M. Ciesielski. 1998. Using Back Propagation Algorithm and Genetic Algorithms to Train and Refine Neural Networks for Object Detection. Proceedings of Computer Science Postgraduate Students Conference. Royal Melbourne Institute of Technology.

[5] Randall, S. S. and A. S. Naheel. 2001. Data Mining Using a Genetic Algorithm-Trained Neural Network. *International Journal of Intelligent Systems in Accounting, Finance & Management*. 10: 201-210.

[6] Song, M. P. and G. C. Gu. 2004. Research on Particle Swarm Optimization: A Review. Proceedings of 2004 International Conference on Machine Learning and Cybernetics. Shangai, China. 4: 2236-2241.

[7] Lee, J. S., S. Lee., S. Chang and B. H. Ahn. 2005. *A Comparison of GA and PSO for Excess Return Evaluation in Stock Markets*. Gwangju Institute of Science and Technology, Republic of Korea. Springer-Verlag: 221-230.

[8] Al-kazemi, B. and C. K. Mohan. 2002. Training Feedforward Neural Network Using Multi-phase Particle Swarm Optimization. Proceeding of the 9th International Conference on Neural Information Processing. New York.

[9] Zhang, C., H. Shao and Y. Li. 2000. Particle Swarm Optimization for Evolving Artificial Neural Network. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics 2000. 2487-2490.

[10] Yao, X. 1993. A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems*. 4: 203-222.

[11] Van den Bergh, F. 1999. Particle Swarm Weight Initialization in Multi-Layer Perceptron Artificial Neural Networks. Accepted for ICAI. Durban, South Africa.

[12] Eberhart, R. and J. Kennedy. 1995. A New Optimizer Using Particle Swarm Sixth International Symposium on TheoryMicro Machine and Human Science. Nagoya, Japan. 39-43.

[13] Bonabeau, E., M. Dorigo and G. Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial System*. New York: Oxford University Press.

[14] Shi, Y. 2004. Particle Swarm Optimization. *IEEE Neural Network Society*. 2: 8-13.

[15] Eberhart, R. and Y. Shi. 2001. Particle Swarm Optimization: Developments, Application and Resources. Proc. Congress on Evolutionary Computation 200. Seoul, Korea. 81-86.

[16] Van den Bergh, F. and A. P. Engelbrecht. 2000. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*. 26: 84-90.

[17] Jones M. T. 2005. *AI Application Programming*. 2nd Ed. Hingham, Massachusetts: Charles River Media Inc.

[18] Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press.

[19] Newman, D. J., S. Hettich., C. L. Blake., and C. J. Merz. 1998. UCI Repository of Machine Learning Databases. University of California, Irvine, Department of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html