The Development of Neural Network Models by Revised Particle Swarm Optimization

Peitsang Wu[1*] Chin-Shiuh Shieh[2] Jar-Her Kao[1]

[1] Department of Industrial Engineering and Management, I-Shou University,
Kaohsiung, Taiwan, R.O.C.
[*]Corresponding author: pwu@isu.edu.tw
[2] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences,
Kaohsiung, Taiwan, R.O.C.

Abstract

A novel training paradigm for artificial neural networks had been developed and presented in this article. In the proposed approach, a revised version of particle swarm optimization (PSO) had been employed to find out the optimal connection weights of feed-forward artificial neural networks for given training sets. Literatures reported that conventional particle swarm optimization could easily get stuck at local optima, especially in problem domains with high dimensionality. In our scheme, a re-seeding mechanism will be invoked when the system is under the risk of converging to pre-mature solutions. The incorporation of the concept of mutation had endowed the systems with better capability in escaping local optima and approaching to the global optimum. A series of experiments were conducted to verify the feasibility and effectiveness of the proposed approach, and optimistic results were obtained as expected. In additions, the impact and influence of different parameter settings on system performance was investigated through comprehensive empirical study, as reported in this paper.

Keywords: neural networks, particle swarm optimization (PSO), mutation, re-seeding.

1. Introduction

Rooted back to 1940s, artificial neural networks (ANN) [1]-[2] were originally used as test beds for the study of the operation of human brain. As time went by, an ample amount of different ANN architectures, together with associated learning algorithms, were developed as problem-solving models and had been successfully applied to various disciplines, such as pattern recognition, prediction, control, diagnosis and function approximation. Attributed to its ability in learning, generalization, and adaptation, feed-forward ANN had been well recognized as an essential member in the ANN taxonomy. In this study, we will use feed-forward ANNs for function approximation / prediction and experiment for their performance on historical data of Taiwan's stock market.

Recognizing as a gradient-descent typed method, the back-propagation algorithm (BPA) can effectively approach local optima for a given training set. However, for those problem instances with high dimensionality, there in general exist a vast number of local optima. It is not uncommon that, for randomly selected initial connection weights, the BPA misses the global optimum and gets stuck at a local optimum. Various approaches had been proposed as countermeasures to this problem, such as genetic algorithms (GA), ant colony systems (ACS), and particle swarm optimization (PSO) [3]-[4]. Among other meta-heuristics for the escaping from local optima, the integration of ANN and PSO is supposed to be a promising direction and has received considerable attention in recent years.

Zhang, *et al.* [5] had developed a PSO-ANN system to model the product quality estimator for fractionators of the hydro cracking unit in the oil refining industry. They proposed a new updating policy of the inertia weights and obtained satisfactory results. Peng, *et al.* [6] had applied PSO-ANN to the design of state estimators of battery packs. The inertia weights in their system are composed by learning constants of individuals as well as groups with random coefficients. Their system reported superior performance on real data. Chatterjee and Siarry [7] had suggested a dynamic adaptation scheme for nonlinear updating of the inertia weights. Their proposal had successfully solved several benchmark problems.

Despite these successful works, conventional PSO is still vulnerable to the local optima problem. These observations motivate the development of our revised PSO. We will introduce the concept of mutation into conventional PSO. Basically, we will allow certain number of particles in the population to fly more violently than others when we suspect that the system had converged to a local optimum. This mechanism, in effect, re-initializes part of the population when systems get stuck. It can serve to be an effective operator for escaping from local optima, and hopefully achieve a balance between exploration and exploitation during the course of searching for the global optimum.

2. Particle Swarm Optimization

Some social systems of natural species, such as bird flock and fish school, possess interesting collective behavior. In these systems, globally sophisticated behavior emerges from local, indirect communication among simple agents with only limited capabilities. In an attempt to simulate the flocking behavior by computers, Kennedy and Eberthart [4] realized that an optimization problem can be formulated as that a flock of birds fly across an area seeking for spot with abundant food. This observation, together with some abstraction and modification, leads to the invention of a novel optimization technique – particle swarm optimization.

Particle swarm optimization optimizes an object function by conducting population-based search. The population consists of potential solutions, called particles, which are metaphor of birds in bird flocking. These particles are randomly initialized and then freely fly across the multi-dimensional search space. During the flying, every particle updates its velocity and position based on the best experience of its own and the entire population. The updating policy will drive the particle swarm to move toward region with higher object value, and eventually all particles will gather around the point with highest object value. The detail operation of particle swarm optimization is given bellows:

Step 1 Initialization
The velocity and position of all particles are randomly set to within pre-specified or legal range.

Step 2 Velocity Updating
For each iteration, the velocity of all particles is updated according to the following rule:

$$\vec{v}_i \leftarrow w \cdot \vec{v}_i + c_1 \cdot R_1 \cdot (\vec{p}_{i,best} - \vec{p}_i) + c_2 \cdot R_2 \cdot (\vec{g}_{best} - \vec{p}_i)$$

where $\vec{p}_i$ and $\vec{v}_i$ are position and velocity of particle *i*, respectively; $\vec{p}_{i,best}$ and $\vec{g}_{best}$ are the position with best object value found so far by particle *i* and the entire population, respectively; *w* is a parameter controlling the dynamics of flying; R_1 and R_2 are random variables from the range [0,1]; c_1 and c_2 are factors used to control the related weighting of corresponding terms.

The inclusion of random variables endows the particle swarm optimization with the ability of stochastic searching. The weighting factors, c_1 and c_2, compromises the inevitable tradeoff between exploration and exploitation.

After the updating, $\vec{v}_i$ should be checked and clamped to pre-specified range to avoid violent random walking.

Step 3 Position Updating
Assuming unit time interval between successive iterations, the positions of all particles are updated according to the following rule:

$$\vec{p}_i \leftarrow \vec{p}_i + \vec{v}_i$$

After the updating, the $\vec{p}_i$ should also be checked and clamped to legal range to ensure legal solutions.

Step 4 Memory Updating
Update $\vec{p}_{i,best}$ and $\vec{g}_{best}$ when condition is meet.

$$\vec{p}_{i,best} \leftarrow \vec{p}_i \text{ if } f(\vec{p}_i) > f(\vec{p}_{i,best}),$$
$$\vec{g}_{best} \leftarrow \vec{p}_i \text{ if } f(\vec{p}_i) > f(\vec{g}_{best})$$

where $f(\vec{x})$ is the object function subject to maximization.

Step 5 Termination Checking
Repeats Step 2 to Step 4 until certain termination condition is met, such as pre-defined number of iterations is reached or failed to have progress for certain number of iterations. Once terminated, particle swarm optimization reports the $\vec{g}_{best}$ and $f(\vec{g}_{best})$ as its solution.

3. Revised Particle Swarm Optimization

Ideally, an adequate range of velocity updating should depend on the landscape of the object function under consideration. However, such an assumption is highly impractical in real applications, especially when the object function posses heterogeneous landscape. A too large velocity will lead to random walk, while a too small one will increase the risk of trapping at local optima. Most of the previous works follow a trial-and-error approach. Chatterjee and Siarry [7] had made an attempt by adaptation. Here we will take a quite different approach by incorporating the concept of mutation. The mutation is a background operator used in genetic algorithms to prevent pre-mature convergence. We will follow and extend the same philosophy and apply it to conventional PSO. During the velocity updating phase, some particles in the population will be randomly selected and be assigned with randomly set velocity, while others follow the conventional velocity updating rule in a more conservative manner. By doing so, those particles selected and assigned with random velocity are actually conducting the exploration in unexplored search space, while others are focusing on the locating of the optimum in explored regions. As a result, there will be a coherent balance between exploration and exploitation for the revised PSO.

4. Experiments on Revised PSOANN

A series of experiments were conducted to verify the feasibility and effectiveness of the proposed approach. Historical data collected from Taiwan stock market during November 9, 1999 to March 4, 2000 is used in our experiments. Totally, 120 patterns are collected. The first 100 patterns are used for the training patterns and the last 20 patterns are for the testing patterns. Essential experiment settings are listed bellows:

- Topology of feed-forward ANN: 8x10x1
- Legal range of connection weights: [-2,2]
- $c_1 = c_2 = 3.0$
- Population size: 15
- Mutation rate: 10%
- Mutation range: [-0.03,0.03]
- Number of iterations: 10000

Model	Mutation Rate	Mutation Range	Legal Range	Maximal Velocity
BPANN	-	-	-	-
PSOANN	-	-	[2,-2]	0.03
mPSOANN(1)	0.1	[1,-1]	[2,-2]	0.03
mPSOANN(2)	0.3	[1,-1]	[2,-2]	0.03
mPSOANN(3)	0.1	[2, -2]	[2,-2]	0.03
mPSOANN(4)	0.3	[2,-2]	[2,-2]	0.03
mPSOANN(5)	0.1	[0.03,-0.03]	[2,-2]	0.03
mPSOANN(6)	0.3	[0.03,-0.03]	[2,-2]	0.03

Table 1: Experiment configuration.

After training the 100 training patterns, the best fitness of revised PSO-ANN is 9.51E-5, and that of PSO-ANN is 1.077E-4 and BP-NN (Back-Propagation Neural Network) is 1.6E-4.

To examine the effects of different parameter setting, we had experimented various experiment configuration as listed in Table 1.

From Fig. 1, it can seen that configuration mPSOANN(5) possesses the fastest convergence among all models. Fig. 2 reveals the superiority of the revised PSO-ANN over PSO-ANN and BP-NN in terms of average mean-square-error. Fig. 3 presents the results of forecasting ability on testing patterns for the last 20 patterns which are not trained. The revised PSOANN outperform the other two in most cases.

5. Conclusions

A revised PSO featuring the concept of mutation is presented in this article. With re-seeding mechanism, the proposed approach is more capable in escaping from local optima. Initial study reveals that the revised PSO-ANN has better training performance and forecasting ability than the conventional PSO-ANN. It is worthwhile to pay more efforts to this direction in order to establish a more general framework for more powerful PSO.

6. References

[1] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *IEEE Computer*, pp. 31-44, 1996.

[2] J. A. Freeman and D. M. Skapura, *Neural Networks – Algorithms, Applications, and Programming Techniques*, Addison-Wesley, 1991.

[3] S.-C. Chu, C.-S. Shieh, and J. F. Roddick, "A tutorial on meta-heuristics for optimization," in J.-S. Pan, H.-C. Huang, and L. C. Jain (Eds.), *Intelligent Watermarking Techniques*, World Scientific Publishing Company, Singapore, Chapter 4, pp. 97-132, 2004.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.

[5] C. Zhang, Y. Li, and H. Shao, "A new evolved artificial neural network and its application", *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, vol. 2, pp. 1065-1068, 2000.

[6] J. Peng, Y. Chen, and R. Eberhart, "Battery pack state of charge estimator design using computational intelligence approaches", *Proceedings of the Fifteenth Annual Battery Conference on Applications and Advances*, pp. 173-177, 2000.

[7] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization", *Computers & Operations Research*, vol. 33, pp. 859-871, 2006.

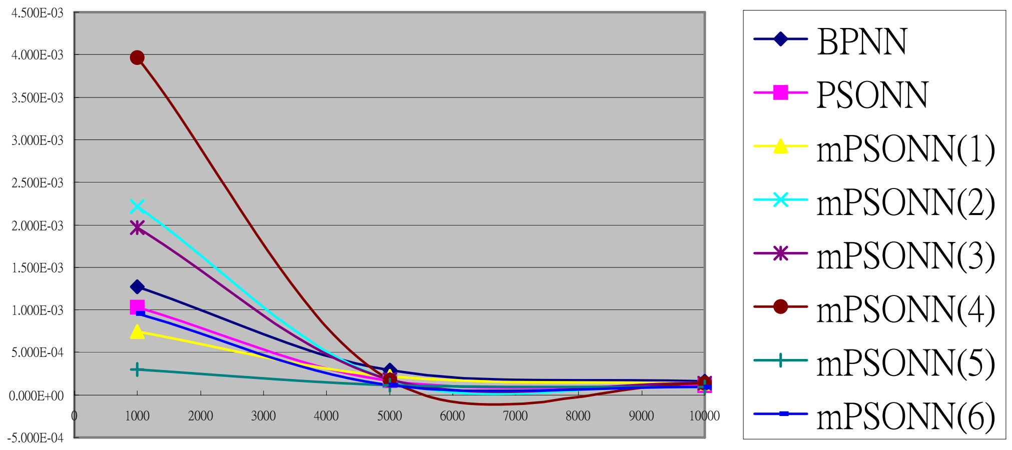

Fig. 1: Minimal MSE.

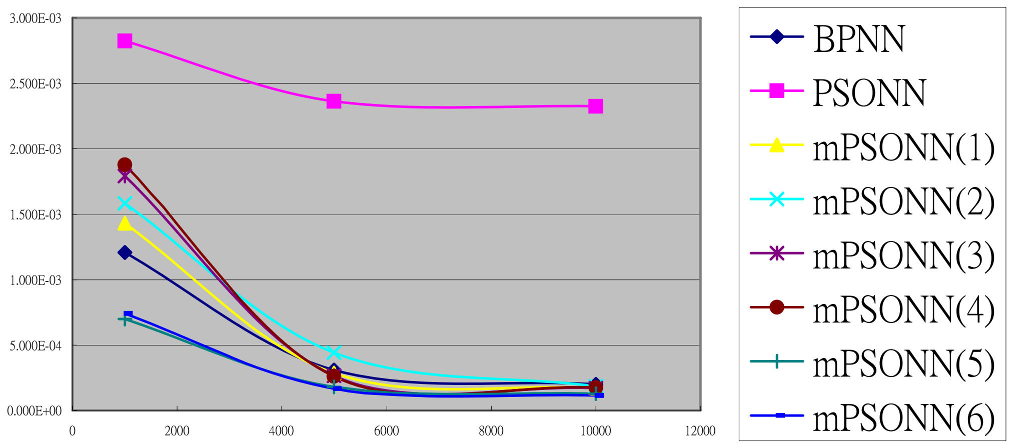

Fig. 2: Average MSE.

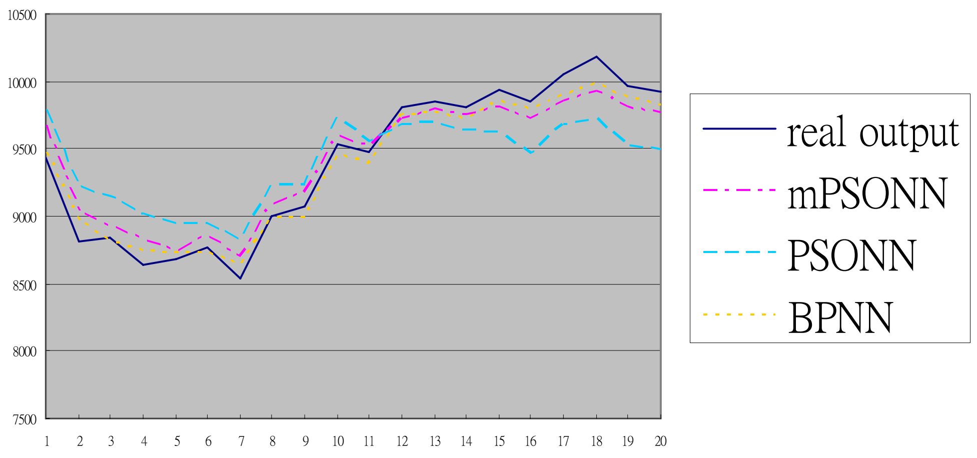

Fig. 3: Prediction on testing patterns.