Journal of Computers Vol. 29 No. 4, 2018, pp. 31-38
doi:10.3966/199115992018082904003

Application of Improved PSO-ELM Algorithm in Optimizing the Path of Robot

Hong-Ge Ren*, Rui Yin, Tao Shi, Fu-Jin Li

College of Electrical Engineering, North China University of Science and Technology
No. 21 Bohai Road, Tangshan 063210, P. R. China
renhg@ncst.edu.cn, 384042235@qq.com, st99@ncst.edu.cn, lfj@ncst.edu.cn

Received 14 November 2016; Revised 30 March 2017; Accepted 26 July 2017

Abstract. In view of the problem of poor stability, large errors in optimizing the parameters of extreme learning machine network (ELM network) by traditional particle swarm optimization algorithm (PSO algorithm), an improved PSO-ELM algorithm (IPSO-ELM algorithm) was proposed in this paper. This algorithm is designed to adjust the inertia factor and learning factors of the PSO algorithm. It chooses the appropriate learning factors and the dynamic inertia factor to improve the optimization performance of PSO algorithm. The core of IPSO-ELM algorithm is to improve the certainty of initial weights and thresholds that belonged to ELM neural network and then train the simples by using ELM neural network for enhancing the generalization ability and stability of system. The simulation experimental results show that the proposed IPSO-ELM algorithm outperforms other similar algorithms with faster convergence speed, better robustness, lower errors, and higher accuracy.

Keywords: extreme learning machine, higher accuracy, improved particle swarm optimization, inertia and learning factors, optimized path

1 Introduction

Nowadays mobile robot has become a hot issue in the field of control research. However, the path planning of robot is an important topic in the field of mobile robot learning. This problem has been described as follows: according to a few optimization methods, mobile robot can find an optimal or near optimal path, enables the robot to reach the target position from the starting position, and can avoid the obstacles on the path in spatial motion [1-2]. As a matter of fact, it is a problem of mobile robot navigation and collision avoidance. Since 1970s, there are mainly three methods on the path planning of mobile robot which can show follow: firstly, it is the mobile robot path planning method based on the environment models, which can deal with the complete consistent environment. That is, the shape and location of the obstacles in a given state, but it can't deal with the unknown environmental information online, the specific methods have grid method, visual map method and topological method; secondly it is the mobile robot path planning method based on the sensor information, the specific methods are artificial potential field method, determine the grid method, fuzzy logic method; at last, it is the mobile robot path planning method that based on behavior. It decomposed navigation problem into multiple independent navigation behavior primitives, such as obstacle avoidance, following, goal guidance, behavior elements, which have the complete motion control unit with sensors and actuators and corresponding navigation function. They coordinate each other, completed the overall navigation tasks.

There are many intelligence algorithms with different performance for solving the optimization problem such as genetic algorithm (GA) [3] and particle swarm optimization (PSO) [4]. Compared with other algorithms, PSO has few parameters to adjust, which is easy to implement. So it is one of the most popular optimization algorithms. Lazzús [5] combined the advantage of BP and PSO for predicting thermal properties of the organic compounds, which can improve the convergence speed and prediction

* Corresponding Author

accuracy of the traditional BP network. Zhang et al. [6] and Li et al. [7] proposed two kinds of improved PSO algorithm for overcoming shortcoming of the traditional BP network, which can adjust the inertia weight coefficients and learning factors adaptively.

In view of the uncertainty of the random input weights and thresholds of extreme learning machine, an improved extreme learning machine based on particle swarm optimization algorithm (IPSO-ELM algorithm) is proposed to optimize the parameters of extreme learning machine. In 1995, Kennedy and Eberhart were inspired by the phenomenon of cluster and foraging activities in flying birds, and a particle swarm optimization algorithm was proposed [8-9]. The utility model has the advantages of small number of individuals and good robustness and great effect on all kinds of optimization problems. And the algorithm uses the particle swarm optimization algorithm to optimize the input weights and thresholds of the extreme learning machine, thus optimizes the output weights. Under the premise of the fast training speed, the training accuracy of the algorithm is greatly improved, and the optimal planning path of the robot is obtained.

2 Traditional and Improved PSO Algorithm

2.1 Traditional PSO Algorithm

In the PSO model, the essence is that each optimization problem can be regarded as a "particle" in the search space. Each particle has a "Fitness value", which depends on the fitness function $F\left(X_j\right)$. All particles are given the memory function. So it can accurately remember the best location of them in the search process. Each particle runs randomly in the solution space. Their motion distance and direction are determined by the same particle velocity. The particles adjust timely according to their own flight experience and their companion's flight experience, and update their position and speed according to the two "extremum" and "Fitness value". One of "extremum" is the optimal solution of the particle named individual best position p_{best}; the other "extremun" is the optimal solution for the whole population named global best position g_{best}.

In D-dimensional space, the population $X=\left(X_1,X_2,\cdots,X_m\right)$ is composed of m particles. Among them, $X_j=\left(x_{j1},x_{j2},\cdots,x_{jD}\right)^T$ indicates the position of the j th particle. $V_j=\left(v_{j1},v_{j2},\cdots,v_{jD}\right)^T$ indicates the velocity of the j th particle. $P_j=\left(p_{j1},p_{j2},\cdots,p_{jD}\right)^T$ indicates the individual optimal position of the j th particle. $P_g=\left(p_{g1},p_{g2},\cdots,p_{gD}\right)^T$ indicates global optimal position of population. In the training process of PSO algorithm, the particle velocity and position are updated by the formula (1) and (2).

$$V_{id}^{u+1}=wV_{id}^{u}+c_1R_1\left(P_{id}^{u}-X_{id}^{u}\right)+c_2R_2\left(P_{gd}^{u}-X_{id}^{u}\right) \tag{1}$$

$$X_{id}^{u+1}=X_{id}^{u}+V_{id}^{u+1} \tag{2}$$

In the formulas, u represents evolutionary iteration number. c_1, c_2 are the acceleration constants, and they are nonnegative number. We usually take $c_1=c_2=2$. R_1, R_2 are two constants which obtain into the interval of (0,1). w represents the inertia factor which is used to adjust the scope of the regulation of solution space. In formula (1), wV_{id}^{u} is called the "momentum" section which represents current velocity of particles. $c_1R_1\left(P_{id}^{u}-X_{id}^{u}\right)$ is called the "cognitive" section which represents current behavior in thinking and memory of particles. $c_2R_2\left(P_{gd}^{u}-X_{id}^{u}\right)$ is called the "society" section which represents information sharing and cooperation among particles.

Algorithm process of PSO is as follows:

Step 1. Initialize the position and velocity of the particle swarm. Take the acceleration constants $c_1=c_2=2$ and set the maximum number $U_{\max}$ of iterations.

Step 2. The fitness value f of each particle is calculated according to the fitness function $F\left(X_j\right)$.

Step 3. Compare the calculated fitness value f with the individual optimal position p_{best}. If the performance of calculated fitness value is better than p_{best}, we replace the previous individual optimal position with X_j. On the contrary, the current optimal position is unconverted.

Step 4. Compare the calculated fitness value f with the global optimal position g_{best} of the individual. If the performance of calculated fitness value is better than g_{best}, we replace the global optimal position with X_j. On the contrary, the current global optimal position is unconverted.

Step 5. Update position and velocity of each particle according to formula (1), (2), and update the next particle in accordance with **Step 2-Step 4** again.

Step 6. Detect whether the adaptation value has reached the specified accuracy. If the specified accuracy is reached the maximum number of iterations $U_{\max}$, and the loop iteration is over. Otherwise it returns **Step2**.

The paper size is A4. The printing margins are: 2cm for up and down margins and 2.5cm for left and right margins. The text should be justified to occupy the full line width, so that the right margin is not ragged, with words hyphenated as appropriate.

2.2 Improved PSO Algorithm

Learning Factors c_1 and c_2. The learning factors c_1 and c_2 which called acceleration constants, are responsible for controlling individual cognitive component $\left[P_{id}^u - X_{id}^u\right]$ and group cognitive component $\left[P_{gd}^u - X_{id}^u\right]$. Due to the lack of methods and formula, the learning factor is usually determined on a general scope. Li et al. [7] utilized formula (1) at the different number of iterations for building a special linear difference equation with constant coefficients and gave the value range of c_1 and c_2. Let $c_1 + c_2 = c$; the range of w and c is shown as follows [8]:

$$\begin{cases} c > 0 \\ 2w - c + 2 > 0 \\ -3 < w - c < 1 \\ 0 < w < 1 \end{cases} \tag{3}$$

In addition, Gao [10] proved that the PSO algorithm ultimately tends to the optimal solution with the help of updating the two factors based on the rule mentioned above. In the proposed algorithm, we also use the rule mentioned above and limit the scope of c, which affects the convergence speed and accuracy. In the subsequent trial, we set $c = 2$.

The Inertia Weight *w*. In all kinds of improved algorithms, the inertia weight w occupies an important role in the PSO, which is the balance point between the global and local search ability. For avoiding PSO algorithm into local optimum, different solutions are put forward for updating w.

$$w = w_{\max} - \frac{k}{k_{\max}} w_{\min} \tag{4}$$

where $w_{\max} = 0.9$. $w_{\max} - \left(k / k_{\max}\right) w_{\min}$ is the linear decreasing function for directed searching of w. k is the number of iterations where $k_{\max}$ is the maximum value. The value range of inertia weight is described by Shi and Eberhart in reference [11]. It dropped from maximum value 0.9 to 0.4. It can get a higher global search capability in the initial stage of this algorithm, and can converge to a smaller search space gradually. It also makes the local search ability strengthen gradually to optimize the accuracy of the solution.

3 The Flow of Improved PSO-ELM Algorihm

ELM neural network [12-15] is a feed-forward network, which is composed of input layer, hidden layer, and output layer. It substantially turns an input/output problem into a nonlinear optimization problem. ELM algorithm depends on learning and training a certain amount of samples for determining weights and threshold of the network. So we use the IPSO algorithm to optimize the initial weights and the thresholds of basic ELM algorithm and then train the samples for enhancing the generalization ability and stability of ELM network. The algorithm flow chart is shown in Fig. 1.

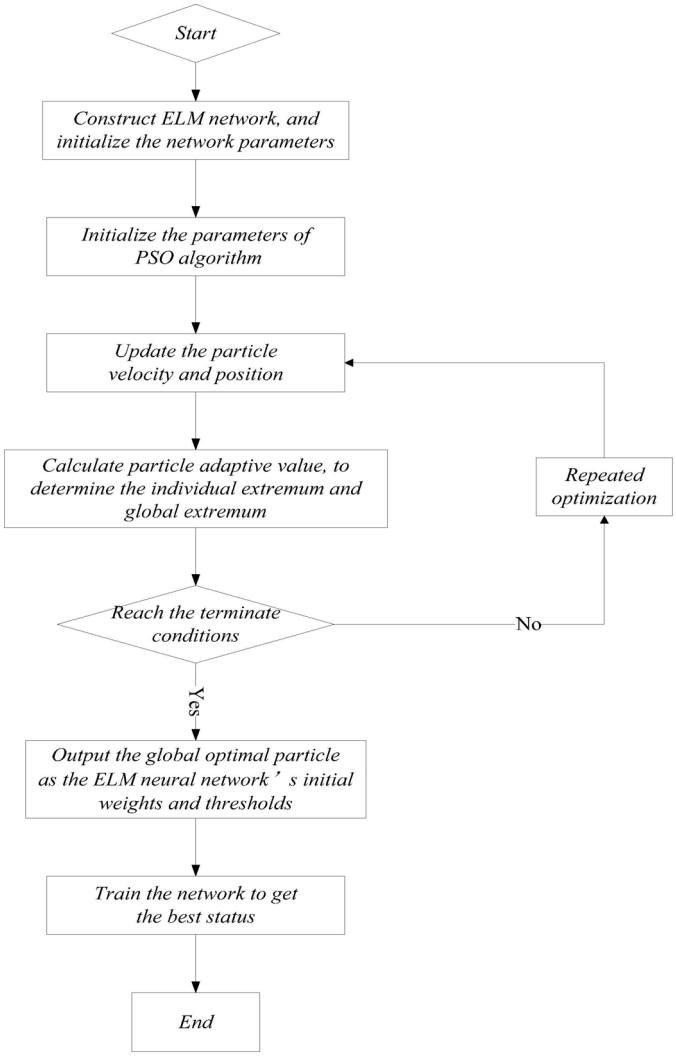

Fig. 1. Flow chart for the IPSO-ELM algorithm

The algorithm steps are shown as follows.

Step 1. Considering the process of initialization such as population size N, particle dimension D, the maximum number of iterations $k_{\max}$, the error precision of particles E, the maximum speed $V_{\max}$, and the position $X_{\max}$.

Step 2. The initial weights and the thresholds of traditional BP algorithm are become column which are the initial population of PSO algorithm. Suppose the number of initial population is t and the number of initial weights and thresholds is n. x_{in} is the initial position of one particle, $0 < i \le t$

Step 3. Calculate the fitness value of every particle and find the best position along with updating particle based on formula (1) and (2) until training the best position of the whole particle swarm.

Step 4. Terminal condition: if the fitness value is less than the error precision of particles E or $k_{\max}$ is reached, the algorithm terminates; otherwise, go to **Step 3**.

4 Experimental and Simulation Results

In order to verify the feasibility of the algorithm, experiments were carried out by Matlab 7.0. The experimental environment is Windows7, 64bit operating system, Core i5-3470 3.2GHz 4G Intel memory.

The position and number of obstacles in the simulation environment were set randomly. We had selected learning factors $c_1 + c_2 = 2$ in section 2.2.1, so we set $c_1 = c_2 = 1$. According to formula (4) in section 2.2.2, the inertia weight was from global optimization to local optimization. The number of particles was 300 and the maximum iteration number was 2000. In this experiment, two groups of obstacles in simulation environment were randomly assigned. One of experiments was simple, and the number of the obstacles was 13. The other was more complex, and the number of obstacles reached to 24. Its path simulation results were shown in Fig. 2 and Fig. 3. And the relationships between the number of iterations and each fitness value were shown in Fig. 4 and Fig. 5. According to the different environments, the robot could find the optimal path and reached the target point quickly.

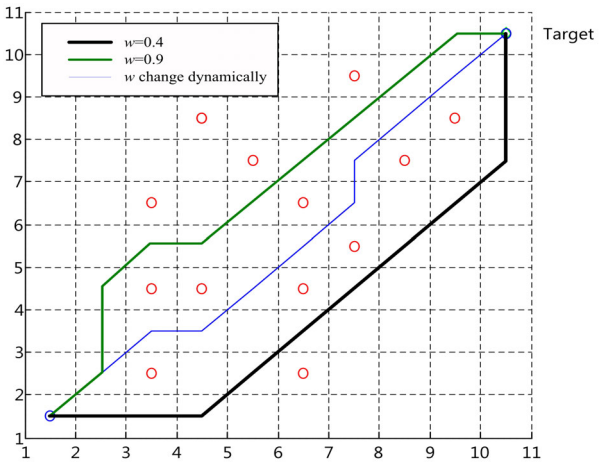

Fig. 2. Paths in simple environment

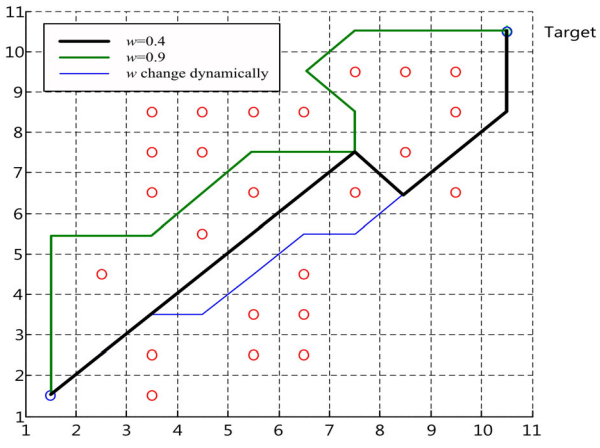

Fig. 3. Paths in complex environment

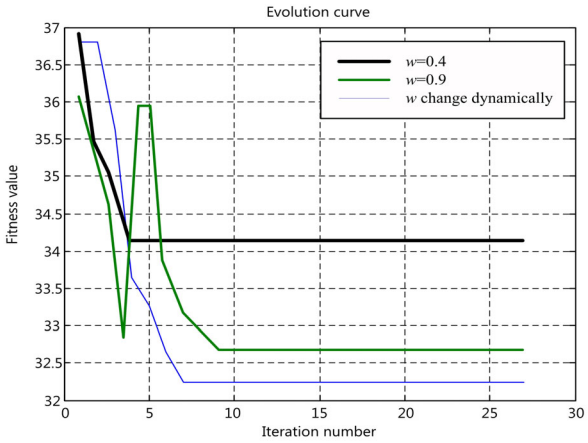

Fig. 4. Relationship between convergent iterative steps and fitness value in the simple environment

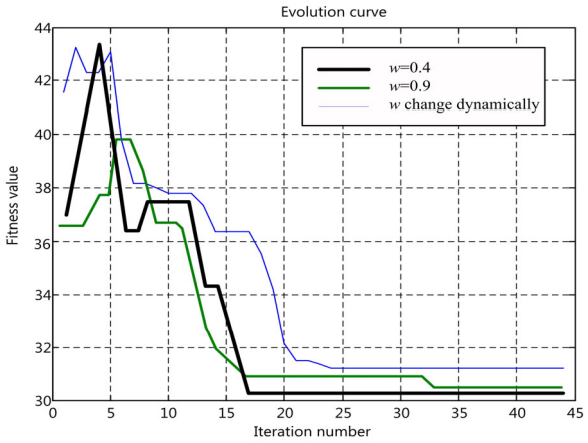

Fig. 5. Relationship between convergent iterative steps and fitness value in the complex environment

Fig. 2 and Fig. 3 respectively were the paths in the simple environment and the complex environment when the inertia weight w was 0.4, 0.9 and dynamic. It could be seen from two figures, in a simple environment, the robot could find relatively short paths; but in complex environment, the path is longer. The path length and the error were shown in Table 1. It can be seen that the path was the shortest when w was dynamic, and the error was the smallest.

Table 1. The path length and the error in different environment

Environment	The shortest path	$w = 0.4$	$w = 0.9$	Dynamic w	Error in $w = 0.4$	Error in $w = 0.9$	Error in dynamic w
Simple	12.73	14.49	13.90	13.31	13.83%	9.19%	4.56%
Complex	12.73	14.73	17.66	13.90	15.71%	38.73%	9.19%

Fig. 4 and Fig. 5 respectively were the relationship between convergent iterative steps and fitness value in the simple environment and the complex environment when the inertia weight w was 0.4, 0.9 and dynamic. It can be seen from the figures, in the same environment but different inertia weight, the convergence of the number of convergence iteration steps was different. Its convergent iteration steps and fitness values were shown in Table 2.

Table 2. Convergent iteration steps and fitness values

	Iteration step in simple environment	Iteration step in complex environment	Fitness value in simple environment	Fitness value in complex environment
$w = 0.4$	4	17	34.1198	30.2478
$w = 0.9$	9	33	32.7624	30.8785
Dynamic w	7	24	32.2441	31.2269

From Table 1 and Table 2, we could be seen that the proposed IPSO-ELM algorithm in this paper not only got the shortest path and the minimum error , but also the iteration number was relatively smaller. That was relatively faster training speed.

We adopt three algorithms such as traditional BP algorithm, ELM network, and PSO-BP algorithm for illustrating the astringency of the proposed approach.

We adopted three-layer topology for building the BP network which selected $Net(2,19,1)$, so did ELM network. Learning factors, the number of populations and iterations of PSO algorithm were unchanging. But inertia weight changed dynamically. The comparison of four algorithms was shown in Fig. 6. As we could be seen in the figure, in the case of more obstacles, the convergence speed of IPSO-ELM algorithm proposed in this paper is faster than others. That is the convergent iteration steps were less than others. It could reflect the fast performance of this algorithm in complex environment.

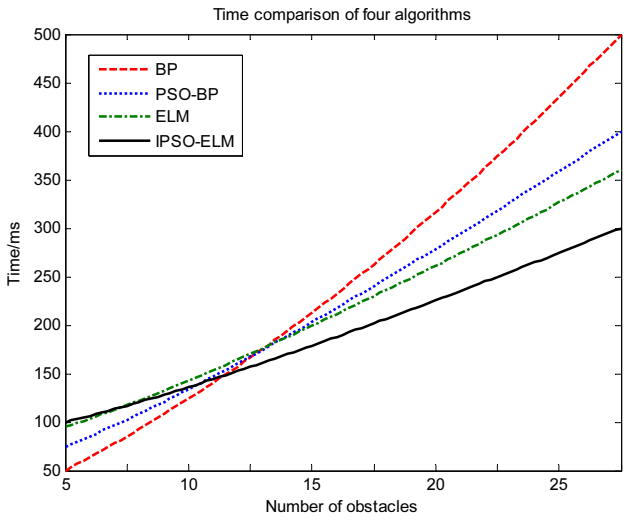

Fig. 6. The comparison of four algorithms

5 Conclusion

An improved PSO-ELM network is proposed in this paper, which is superior to the traditional BP neural network in several aspects such as low study efficiency, slow convergence speed, and being easily jumped out of local optimal. The core of the proposed algorithm is to reduce the uncertainty of selecting the initial values and weights of ELM network for improving the ability of generalization. We apply this algorithm in robot to find shorter path. In this algorithm, it adjusted the inertia weight dynamically to ensure the optimum seeking range. It can find the optimal path faster and more accurately. And it also solves the contingency of the traditional particle swarm optimization algorithm. Compared with the traditional BP, ELM and PSO-ELM, we demonstrate the effectiveness of the proposed approach. The compensation results show that the network optimized by the improved PSO algorithm has better performance with faster convergence speed, lower errors, higher accuracy, and autonomous learning ability.

Acknowledgements

Rui Yin was supported by the National Nature Science Foundation under Grant of China (61203343). The authors would also like to acknowledge the contributions of Hongge Ren to the experiment and technical support from Tao Shi, Fujin Li

References

[1] H.Y. Shi, C.Z. Sun, Motion planning of mobile robot in unstructured environment, Robot 26(1)(2004) 27-31, 2004.

[2] X.P. Fan, S.Y. Li, T.F. Chen, Robot dynamic obstacle avoidance planning based on new artificial potential field function, Control Theory and Application 22(5)(2005) 703-707.

[3] F.S. Wen, Z.X. Han, Fault section estimation in power system using genetic algorithm and simulated annealing, Proceedings of the CSEE 14(3)(1994) 29-36.

[4] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. the IEEE International Conference on Neural Networks, 1995.

[5] J.A. Lazzús, Neural network-particle swarm modeling to predict thermal properties, Mathematical and Computer Modelling 57(9-10)(2013) 2408-2418.

[6] L. Zhang, J.Q. Zhao, X.N. Zhang, S.L. Zhang, Study of a new improved PSO–BP neural network algorithm, Journal of Harbin Institute of Technology (New Series) 20(5)(2013) 106-112.

[7] Q. Li, K.X. Zhou, The research of the pressure sensor temperature compensation based on PSO-BP algorithm, Acta Electronica Sinica 43(2)(2015) 413-416.

[8] X.L. Tang, H. Zhang, Y.Q. Cui, L. Gu, Y.Y. Deng, A nevel reactive power optimization solution using improved chaos PSO based on multi-agent architecture, International Transactions on Electrical Energy Systems 24(5)(2014) 609-622.

[9] B. Fu, Passive targets localisation algorithm based on optimising extreme learning machine with PSO, Computers Applications and Software 32(11)(2015) 325-328.

[10] Z. Gao, X.Z. Liao, Hybrid adaptive particle swarm optimization based on average velocity, Control and Decision 27(1)(2012) 152-160.

[11] Y. Liu, Improvements and Applications of PSO Algorithm, Xidian University, 2012.

[12] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine, a new learning scheme of feed-forward neural networks, in: Proc. IEEE International Joint Conference on Neural Networks, 2004.

[13] B.Y. Wang, S. Zhao, S.M. Zhang, A distributed load forecasting algorithm based on cloud computing and extreme learning machine, Power System Technology 38(2)(2014) 526-531.

[14] W.W. Zong, G.B. Huang, Y.Q. Chen, Weighted extreme machine for imbalance learning, Neurocomputing 101(2013) 229-242.

[15] G.B. Huang, H. Zhou, X. Ding, X. Zhang, Extreme learning machine for regression and multi-class classification, IEEE Trans Pattern Anal Mach Intell 42(2)(2012) 513-529.