A Hierarchical PSO Algorithm for Self-organizing Neural Network Design

Chen Lei

Basic Courses Teaching Department, the Chinese People’s Armed Police Force Academy, China, E-mail: chen_nuli@126.com

Abstract

Particle swarm optimization (PSO) algorithm has come to be widely used as a problem solving method in engineering and computer science. This algorithm is one of recently several highly desirable attributes, including the fact that the basic algorithm is very easy to understand and implement. It is similar in some ways to evolutionary algorithms, but requires less computational bookkeeping and generally fewer lines of code. In the traditional training algorithm, we will determine neuron for number of neural network, but we are unable to guarantee that is feasible to number. So, in this paper, a hierarchical particle swarm optimization (HPSO) is proposed, which can determine the structure of the neural network and tune the parameters in the neural network automatically. Experimental results have shown that the proposed HPSO has a good performance.

Keywords: Particle Swarm Optimization, Neural Network, Genetic Algorithm

1. Introduction

Particle swarm optimization (PSO) was originally introduced by Kennedy and Eberhart in 1995 [1], originated in the social behaviors’ studies of synchronous bird flocking and fish schooling.

In PSO algorithm, the trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of the other particles in the search space. The position vector and the velocity vector of the ith particle in the N-dimensional search space can be represented as 

\[ X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in}) \]

and

\[ V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in}) \]

respectively. According to a user defined fitness function, let us say the best position of each particle (which corresponds to the best fitness value obtained by that particle at time) is

\[ P_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{in}) \]

and the fittest particle found so far at time is

\[ P_g = (p_{g1}, p_{g2}, p_{g3}, \ldots, p_{gn}) \].

Then, the new velocities and the positions of the particles for the next fitness evaluation are calculated using the following two equations [2]:

\[ v_{id} = \omega \times v_{id} + c_1 \times \text{rand}() \times (P_{id} - x_{id}) + c_2 \times \text{rand}() \times (P_{gd} - x_{id}) \]  

(1)

\[ x_{id} = x_{id} + v_{id} \]  

(2)

where, \( c_1 \) and \( c_2 \) are constants known as acceleration coefficients, and rand() are two separately generated uniformly distributed random numbers in the range of \([0,1]\).

The first part of (1) represents the previous velocity, which provides the necessary momentum for particles to roam across the search space. The second part, known as the “cognitive” component, represents the personal thinking of each particle. The cognitive component encourages the particles to move toward their own best positions found so far. The third part is known as the “social” component, which represents the collaborative effect of the particles, in finding the global optimal solution. The social component always pulls the particles toward the global best particle found so far.

A hierarchical structure was proposed by Tang et al. [3]. This idea was originated in biological background of the DNA structure. A DNA structure is made up by regulatory sequences (RSs) and structural genes (SGs). The SGs are coded for polypeptides or RNAs, while RSs serve as the leaders that denote the beginning and ending of SGs, or participate in turning on or off the transcription of SGs, or function as initiation points for replication or recombination.
This architecture is a multiple level of control genes, as shown in Figure 1. The activation of the parametric gene is governed by the value of first-level control gene, which is governed by the second-level control gene.

To indicate the activation of the control gene, an integer 1 is assigned for each control gene that is being ignited where 0 is for turning off. When 1 is signaled, the associated parameter genes due to that particular active control gene are activated in lower level structure. It should be noticed that the inactive genes always exist within the chromosome even when 0 appears. We call this architecture as Hierarchical chromosome structure.

2. Hierarchical PSO Trained Neural Networks

Figure 1 shows the chromosome representation in hierarchical neural network system. Each chromosome consists of control gene and connection gene. The control gene is the binary code and activated hidden layers and hidden neurons of neural network. The connection gene is real-value (weight). A neural network defined by this structure is shown in Figure 2.[4][5].

Within such a specific treatment, a structural chromosome incorporates both active and inactive genes. It should be noted that the inactive genes remain in the chromosome structure and can be carried forward for further generations. Such an inherent genetic variation in the chromosome avoids any trapping at local optima which has the potential to cause premature convergence. Thus it maintains a balance between exploiting its accumulated knowledge and exploring the new areas of the search space.
This structure also allows larger genetic variations in chromosome while maintaining height viability by permitting multiple simultaneous genetic changes.

Figure 3. Adjust the sketch map in chromosome parameter

Figure 3 shows the chromosome. We adjust control genes using GA[6][7][8][9], and adjust parameter genes using PSO. Genetic algorithm is stochastic search procedures based on the mechanics of natural selection, and the concept of “survival of the fittest”. Genetic algorithm use three processes: reproduce, crossover and mutation.

Reproduction is a process in which individual strings are copied according to their fitness value. This operator is an artificial version of neural selection. A fitness value is assigned to each individual using a fitness assignment method in which high numbers denote a good fitness. In this study, we use the roulette-wheel selection method [10][11][12][13].

We use a two-point crossover operation, as shown in Figure 4. It shows new individuals which are created by exchanging the site’s values between the selected sites of parents’ individual. A crossover rate is set to 0.5.

Figure 4. Parametric and Control gene’s crossover operation

Although reproduction and crossover will produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation is an operator that randomly alters the allele of a gene. The mutation operation of an individual is shown in Figure 5. Unlike the traditional symbiotic evolution that generated the mutation value according to the constant range. With mutation, new genetic materials can be introduced into the population. Mutation should be used sparingly because it is a random search operator. In the following simulations, a mutation rate is set to 0.3[14].

Figure 5. Mutation operation
We duplicate better control genes and utilize these better control genes to carry on crossover and mutation, hoping to produce better control genes. Therefore we use GA to get a structure of hierarchical neural network. Figure 6 shows the HPSO algorithm:

**Figure 6. The flowchart of HPSO learning algorithm**

- Step 1: Initially, Particle individuals forming the population should be randomly generated.
- Step 2: Evaluate fitness of particles. The fitness function of the engine power optimization is formulated as follows:

\[ f_i = \alpha \times RMSE + \beta \times Weight\_Ratio \]  

Where, Weight\_Ratio is the proportion of having weight linked of neurons, RMSE represents root-mean-squared error. The characters of \( \alpha \) and \( \beta \) are evaluation ratio and set to 0.7 and 0.3, respectively.
- Step 3: Update local best and global best.
- Step 4: Using GA adjust control genes.
- Step 5: Using PSO adjust parameter genes.
- Step 6: If finished, export best individual; if not finished, return 2.

3. Simulation Results

This section discusses a prediction example that is used to evaluate the HPSO method. For the computer simulation, the initial parameters are shown in Table 1.

### Table 1. The initial parameters before training

<table>
<thead>
<tr>
<th>Trial</th>
<th>Level</th>
<th>Hidden Node</th>
<th>Particles</th>
<th>( \omega )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( V_{max} )</th>
<th>( p_c )</th>
<th>( p_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>10</td>
<td>50</td>
<td>0.6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The Mackey-Glass chaotic time series \( x(t) \) in consideration here was generated from the following delay differential equation:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \]  

Crowder [15] extracted 1000 input-output data pairs \( \{x, y^d\} \), which consist of four past values of \( x(t) \), i.e.

\[
[x(t-18), x(t-12), x(t-6), x(t), x(t+6)]
\]
Where, $\tau = 17$ and $x(0) = 1.2$. There were four inputs to the HPSO method, corresponding to these values of $x(t)$, and one output representing the value of $x(t + \Delta t)$, where $\Delta t$ was a time prediction into the future. The first 500 pairs (from $x(1)$ to $x(500)$) were the training data set, while the remaining 500 pairs (from $x(501)$ to $x(1000)$) were the testing data set used for validating the proposed method[16].

In this simulation result, Table 2 shows the best root-mean-squared error (Best RMSE), averaged RMSE (Ave. RMSE) and the worst root-mean-squared error (Worst RMSE). Table 3 outputs the error of HGA and HPSO.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training cases</th>
<th>Ave. RMSE</th>
<th>Best RMSE</th>
<th>Worst RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPSO</td>
<td>2000</td>
<td>0.009</td>
<td>0.004</td>
<td>0.013</td>
</tr>
<tr>
<td>HGA [17]</td>
<td>2000</td>
<td>0.048</td>
<td>0.021</td>
<td>0.081</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Training error</th>
<th>Forecasting error</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPSO</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>HGA [17]</td>
<td>0.017</td>
<td>0.017</td>
</tr>
</tbody>
</table>

In our best result, the HPSO obtained number of hidden level and hidden node are 1 and 6. The HGA [17] obtained number of hidden level is 2 and number of hidden node are 4 and 5 respectively. HPSO can find the fewer number of hidden level and hidden node, and convergence is better than the HGA, as shown in Figure 7.

In Figure 8, we train 10 times for HPSO method. Initially, the number of hidden level and neuron are set as 1 and 10. We find that the HPSO method training result, the number of neurons is between 5 and 8. In this simulation result, we can know the number of neuron set to 6 or 7 is fine. Figure 9 shows the RMSE value of 10 training times.
4. Conclusion

In this paper, a new hierarchical particle swarm optimization algorithm is proposed for self-organizing neural network design. We can find feasible number of hidden levels and hidden neurons. Simulation results demonstrate that the proposed hierarchical particle swarm optimization for neural network is quite effective in prediction problem.

5. References


