2019 2nd International Conference on Computer Science and Advanced Materials (CSAM 2019)

Software Fault Prediction Analysis under BPSO Dimension Reduction Conditions

Liu Hongqing

Hunan Vocational College of Modern Logistics, Changsha, Hunan, 410131, China

Email:158140027@qq.com

Keywords: bound particle swarm optimization; software fault; deep neural network; dimensionality reduction; bound state

Abstract: The identification of module fault tendency is very important to reduce cost and improve the effectiveness of software development process. The software fault tendency module deep neural networks (DNN) prediction based on the bound particle swarm optimization (BPSO) dimensionality reduction is put forward. Firstly, the calculation framework of the BPSO dimensionality reduction-based software fault tendency module DNN prediction algorithm and the measure indexes of 21 software faults used are given, and the normalized preprocessing method of its index values is also given; secondly, the dimensionality reduction is performed on the software fault dataset by bound particle swarm optimization, and the participle position is represented by a binary (0 or 1) character string to simplify the data processing. Then the prediction of software fault tendency module is realized by the deep neural network algorithm; finally, the performance advantages of the algorithm are verified by simulation experiments on the four standard test sets of PC1, JM1, KC1 and KC3.

1. Overview

Currently, software plays a significant role in all fields, so the software testing is thus recognized as a basic mission for software development [1,2]. Researches show that most faults are prone to occur in several software modules, so what we need to pay attention to is just these modules which are always affected by faults. Compared with traditional algorithm, the algorithm in this article has several creative points, they are :(1)the dimension reduction algorithm based on the Binary Particle Swarm Optimization is introduced to handle with the data of software faults. It can improve the data execution efficiency and is less influenced by testing process;(2)in order to improve the dimension reduction, a kind of Bound-State Particle Swarm Optimization Algorithm is proposed to replace the traditional particle swarm optimization algorithms. It uses the wave function to replace the position and speed the primal particle swarm optimization algorithm to improve the dimension reduction.

2. Dimension Reduction Based on BPSO

2.1 Particle Swarm Optimization Algorithm

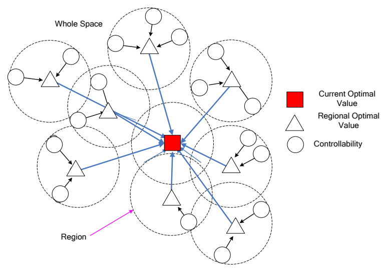

Figure 1 Diagram of the Process of Particle`s Searching for Optimal Solution

In the Particle Swarm Optimization Algorithm, each particle can adjust the position in the search space from time to time based on the flight experience of itself and neighbor nodes. It initializes the population at random and searches for an optimal solution that satisfies some certain performance. The potential solution is called particle which flies through a multidimensional search space. It is shown in Figure 1.

Each particle i has a position vector representing its position X_i. The speed of each particle is represented by a vector of V_i. The formula forms of particle velocity and position are given:

$$\begin{aligned} &V_i(t+1) = \\ &w\cdot V_i(t) + c_1 r_1 (P_{i,best}(t) - X_i(t)) \\ &+c_2 r_2 (P_{global}(t) - X_i(t)) \end{aligned} \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2}$$

In these formulas, t represents the current number of iterations, w represents the inertia weight, c_1 as well as c_2 are positive constants and r_1 and r_2 are random numbers uniformly distributed within the interval of $[0,1]$. And $P_{i,best}$ and P_{global} are the best position for the current visiting particle i and the best value of all particle position values, among which:

$$\begin{cases} X_i(t) = (X_{i1}(t), X_{i2}(t), \cdots, X_{id}(t)) \\ V_i(t) \ = \ (V_{i1}(t), V_{i2}(t), \cdots, V_{id}(t)) \end{cases} \tag{3}$$

In the formula (1), w, c_1 and c_2 are predefined. When the number of iterations is t, the cost value of particle i is as follow:

$$C\left(X_i(t)\right) = \frac{1}{q} \sum_{k=1}^{q} \sum_{j=1}^{d} \left(X_{ij}^{(k)}(t) - \hat{X}_{ij}^{(k)}(t) \right)^2 \tag{4}$$

In this formula, for the particle i, $X_{ij}^{(k)}(t)$ represents the jth output component of the kth sample, and $\hat{X}_{ij}^{(k)}(t)$ represents the jth actual output component of the kth observed sample. In the formula (5), it is better to make the value of $C(\cdot)$ as small as possible. As for the issue of minimization, the smaller the objective function value is, the greater the cost value is. The best position $P_{i,best}$ of particle i can be updated with the following formula:

$$\begin{aligned} &P_{i,best}(t+1) = \\ &\begin{cases} X_i(t),\ if\ \ C\left(X_i(t)\right) < C\left(P_{i,best}(t)\right) \\ P_{i,best}(t),\ if\ \ C\left(X_i(t)\right) \geq C\left(P_{i,best}(t)\right) \end{cases} \end{aligned} \tag{5}$$

When the particle finds a position which is better than the previous best position, it will be stored in memorizer. The algorithm will continue to work until a satisfactory solution is found or the maximum number of iterations is met.

2.2 Bound-State Particle Swarm Optimization Algorithm

In the model of Particle Swarm Optimization Algorithm, the state of a particle is described by a wave function of $\psi(x,t)$ which replaces position and velocity. The dynamic behavior of particle swarms is very different from that of traditional particle swarm optimization algorithms. In this case, the probability density function of particle X depends on the potential field of the particle. The particles move according to the following iterative formula:

$$\begin{cases} X_i(t+1) = p_i(t) + \alpha \left| m_{i,best} - X_i(t) \right| \cdot \ln \dfrac{1}{u_i(t)} \\ ,\ if\ \ s \geq 0.5 \\ X_i(t+1) = p_i(t) - \alpha \left| m_{i,best} - X_i(t) \right| \cdot \ln \dfrac{1}{u_i(t)} \\ ,\ if\ \ s < 0.5 \end{cases} \tag{6}$$

In this formula, the parameter α, which represents the parameter of expansion and shrinkage, is uniform. And u_i and s are uniform random numbers within the interval of $[0,1]$. And there are also two formulas:

$$p_i(t) = \varphi_i(t) \cdot P_{i,best}(t) + (1 - \phi_i(t)) \cdot P_{global}(t) \tag{7}$$

$$m_{i,best} = \frac{1}{Q} \sum_{i=1}^{Q} P_{i,best}(t) \tag{8}$$

In these formulas, φ_i is a uniformly distributed random number in the interval $[0,1]$, Q represents the number of all particles and $m_{i,best}$ is defined as the average value of the best positions of all particles in the population. The termination rule of the Particle Swarm Optimization Algorithm is that if the absolute difference between $C(t+1)$ and $C(t)$ is less than δ which is training threshold for 10 consecutive times, then the algorithm should be stopped; otherwise, it should be stopped until the maximum number of iterations $G_{\max}$ can be satisfied. The flow of the BPSO algorithm is as follows:

```
Algorithm 1: BPSO Algorithm
Initialize the positions of all particles and the position of $P_{i,best}$
   Do
   The formula (9) is used to calculate the $m_{i,best}$ of particle i. $i = 1,2,\cdots,Q$;
  Select the appropriate value of $\alpha$
For particle $i = 1:Q$
   The cost value of particle i can be calculated based on the formula (5);
   The $P_{i,best}$ can be updated based on the formula (6)
   The $P_{global}$ can be updated with the following steps
      If $C(P_{i,best}(t)) < C(P_{global}(t-1))$ then
         $P_{global}(t) = P_{i,best}(t)$;
      Else
         $P_{global}(t) = P_{i,best}(t-1)$;
      Endif
Endfor
do For dimension $1:d$ do
     $\varphi = rand(0,1)$;
     $u = rand(0,1)$;
    If $rand(0,1) \geq 0.5$ do
       The particle position can be updated based on the upper formula of the formula set (7)
    Else
       The particle position can be updated based on the lower formula of the formula set (7)
Endfor
Until the termination conditions are satisfied
```

3. Prediction Method of Tendency of Software Fault

For the prediction performance of the two types of problems, the data in the confusion matrix is usually used for evaluation. It is shown in Table 1.

Table 1 Confusion Matrix

Actual Type	Predicted Type	
	$class = 0$	$class = 1$
$class = 0$	f_{00}	f_{01}
$class = 1$	f_{10}	f_{11}

In Table 2, the actual type of i is predicted to be the number of the type of j by f_{ij}. In the table, the predicted performance indexes are selected as follows: quantitative evaluation of the sensitivity and specificity of the prediction method. These indexes can be calculated based on the confusion matrix, and the calculation forms are:

$$f_{Sensitivity} = \frac{f_{11}}{f_{11} + f_{10}} \tag{9}$$

$$f_{Specificity} = \frac{f_{00}}{f_{00} + f_{01}} \tag{10}$$

In these formulas, the sensitivity is also called the fault detection rate. It is defined as the ratio between the total number of types which are correctly predicted to be prone to be affected by faults and the total number of types which are actually prone to be affected by faults. Specificity is also called correctness. It is defined as the ratio between the total number of types which are correctly predicted to be not prone to be affected by faults and the total number of types which are actually not prone to be affected by faults.

4. Conclusion

This article studies the fault prediction method of application-development software of the hybrid deep neural network and BPSO algorithm. The proposed prediction method can recognize the fault tendency correctly.

References

[1] Liu Jihua; Wang Fengjin; Kong Jie; DNN prediction module based on BPSO dimensionality reduction; computer engineering and design; 2008

[2] Pan Jiansheng; Cheng Shi; Wen Wanzhi; Distance-based software fault analysis method [J]; Wireless interconnection technology; Phase 12, 2017

[3] Fan Zhenyu; Software Fault Measurement Method [J]; Equipment Manufacturing Technology; Issue 08, 2011

[4] Michelle; Ben Kerong; Demand Defect Analysis Based on Behavior Tree and Software Fault Tree [J]; Computer and Digital Engineering; Issue 08, 2010