Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 52 (2018) 12–20

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Cognitive Deep Neural Networks prediction method for software fault tendency module based on Bound Particle Swarm Optimization

Wang Geng

School of Computer Science, Hubei University of Technology, Wuhan 430068, China

Received 15 May 2018; received in revised form 29 May 2018; accepted 2 June 2018
Available online 8 June 2018

Abstract

Identification of module fault tendency is greatly important for cost reduction and software development effectiveness. A DNN (Deep Neural Networks) prediction method for software fault tendency module based on BPSO (Bound Particle Swarm Optimization) dimensionality reduction was proposed in the paper. Firstly, the calculation framework of the DNN prediction algorithm for software fault tendency module based on BPSO dimensionality reduction and 21 software fault measurement indexes as well as the normalization processing method of these index values were provided in the paper; then, the particle swarm optimization algorithm was adopted for the dimensionality reduction of software fault data set, and the particle position was represented by binary (0 or 1) character string to simplify data processing; then, the DNN algorithm was adopted to predict software fault tendency module; finally, the simulation experiments were implemented in four standard test sets—PC1, JM1, KC1 and KC3 to verify the performance advantage of the algorithm.

Keywords: Particle swarm optimization algorithm; Software fault; Deep neural network; Dimensionality reduction; Bound

1. Introduction

At present, software plays an important role in various fields, and software testing is regarded as a basic task of software development (Chan, Zhang, & Uhrich, 2015). Relevant research shows that most faults usually only occur in several software modules, so people only need to concern several software fault tendency modules.

The software fault tendency prediction method at early period is based on statistics, and the prediction performance is dissatisfactory. Recently, the machine learning technologies, including data mining, Naive Bayesian algorithm, DNN, fuzzy logic, etc. (Ghebrebrhan et al., 2017; Malarkodi et al., 2013), have been introduced. Although the software fault is researched in these technologies, yet the data dimensionality is not reduced in above algorithms during fault data set processing, so these algorithms have excessively high calculation complexity, especially in large software projects. The DNN prediction model adopted in the paper also has the problem of long calculation time for massive data analysis. The most common dimensionality method used in software engineering is the principal component analysis (PCA). For example, in literature (Stephygraph & Arunkumar, 2016), on the basis of PCA method for software fault data for dimensionality reduction, SVM (support vector machine) was adopted for the statistical analysis of the data and ant colony algorithm was adopted for optimizing the parameters of the SVM algorithm; in literature (Ghebrebrhan et al., 2017), similarly on the basis of PCA method for software fault data

E-mail address: shanshando9@163.com

https://doi.org/10.1016/j.cogsys.2018.06.001
1389-0417/

for dimensionality reduction, a two-stage fault prediction technology was designed. However, PCA has a disadvantage that opposite to the original input variable, the derivative dimensionality may not have intuitional explanation. The other common dimensionality reduction method is PLS (Partial Least Squares). At present, PLS has wide application in dimensionality reduction field. For example, in literature (Kurup et al., 2017), the least square algorithm was adopted for software fault data for dimensionality reduction, and the artificial neural network was adopted for the statistical analysis of the data; in literature (Arunkumar et al., 2017), the least square algorithm was also adopted for software fault data for dimensionality reduction and for the statistical analysis of the fault distribution units of the complex software system, etc. There are many similar literatures, but the classical PLS implementation process is based on nonlinear iteration, and original sample data usually need to be assumed or converted for convenient use of traditional estimation methods. In practical conditions, original sample data may be influenced by various factors such as operating environment, testing strategy and resource allocation. Therefore, it is difficult to meet such hypotheses in practical problems.

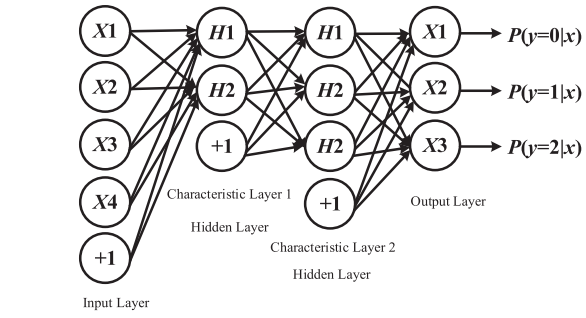
Compared with traditional algorithm, the paper has following innovations: (1) particle swarm data dimensionality reduction algorithm is introduced for software fault data for improving data execution efficiency and reducing testing process influence; (2) in order to improve data dimensionality reduction effect, a bound particle swarm optimization algorithm is proposed to replace traditional particle swarm algorithm; position and velocity of original particle swarm optimization algorithm are replaced by wave function to improve data dimensionality effect.

2. Model method and software index

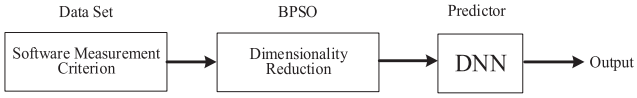
2.1. Model method

DNN (Zhang et al., 2016) is a learning algorithm structure formed by adding multiple hidden web-based learning units between output and input network layers. The hidden layers have fewer nodes than the input layer of the encoder and the output layer of the decoder. Meanwhile, the second-order optimization method is introduced to realize DNNs network training. The DNN structure is as shown in Fig. 1a.

The DNN with dual-hidden-layer network is adopted in the paper. Specifically, field theory is introduced into the traditional particle swarm optimization algorithm as the searching method for ensuring global convergence (Arunkumar & Mohamed Sirajudeen, 2011; Ashokkumar, Arunkumar, & Don, 2018; Hussein). Therefore, BPSO algorithm is adopted for dimensionality reduction. By the combination of hybrid deep neural network algorithm and particle swarm optimization algorithm, an effective software defect prediction method was proposed in the paper. The prediction method is as shown in the block diagram in Fig. 1b.



(a) Deep neural network structure



(b) Prediction algorithm framework

Fig. 1. DNN structure and algorithm framework.

2.2. Software measurement index

At present, software indexes are used in most research for identifying the fault tendency module. Relevant research shows that software indexes are very useful for fault tendency prediction (Sarvaghad-Moghaddam et al., 2018). Specifically, McCabe, Butler, Halstead criterions, etc. (Arunkumar & Mohamed Sirajudeen, 2011; Ashokkumar et al., 2018) are used in the paper. The selected indexes include code line, circulation complexity, basic complexity, design complexity, etc., and are described in Table 1. In the experiment of the paper, the software fault tendency of each software module is represented by 21 indexes.

2.3. Preprocessing process

For DNN algorithm, a data set $(x^{(i)}, y^{(i)})$ $(i = 1, 2, \ldots, q)$ is preset, wherein $x^{(i)} \in R^d$ is the vector of the software index value for quantifying the index value of the ith class, and q is the total number of the data samples. The output value of the ith expectation of the output nerve cell is $y^{(i)}$, value "1" for corresponding breakpoint tendency and "0" for corresponding non-breakpoint tendency. For DNN training, each input is normalized into the same interval number, which is favorable for improving the training process behaviors and ensuring the equal importance of each initial input. Notably, the upper limit of the software index is usually unlimited in the value range thereof. For standardization, it is necessary to obtain the upper and lower limits of the software index value range. For specific data set and software index, we can obtain each index value according to the data set. Each index value is provided in each data set, so it is easy to obtain the maximum value and the minimum value.