Research on software defect predict model based on JCUDA_BP algorithm

Chun Shan, Lei Dong, Changzhen Hu, Liang Zhang, Jingfeng Xue

School of Software, Beijing Institute of Technology

Beijing, China

sherryshan@bit.edu.cn, 294055164@qq.com, chzhoo@bit.edu.cn, 56279329@qq.com, xuejf@bit.edu.cn

Keywords: security; defect predict; JCUDA_BP algorithm

Abstract. For the long time computation and the poor learning effect in application of BP algorithm in software defect predict model, an improved BP algorithm is presented in this paper. The improved algorithm combines the JCUDA technology and the BP algorithm, by learning the sample decomposition, multithreading and CPU-GPU processing mode, both learning speed and predicting time are optimized for the software defect predict model based on BP algorithm. The experimental results show that the improved JCUDA_BP algorithm optimizes the learning process, especially for the larger samples. Also, when the hidden node number reaches a certain number, the effect of the optimized algorithm is much better than the traditional BP algorithm.

Introduction

Software defect refers to software error, which will cause a reduction in functionality or performance of the software system so that the software can't meet of the users' basic requirement. Software defect prediction means predicting the software defect that may exist, it judges which module of the software system contains error that not having been found [1]. Recent years, with the software system being widely used continuously, an increasing number of unknown errors in software system are exposed. Meanwhile, the enormous challenges facing software reliability also prompting researchers to focus on research for software defect prediction. Therefore, how to establish reasonable software defect predict model becomes the focus of recent research.

Software defect prediction technology can be divided into static type and dynamic type [2]. Currently, the research on software defect predict model is deepening in the major research institution. Parag C. Pendharkar proposed a learning problem in a software defect predict model(SDPMLP, a software defect prediction model learning problem)[3], the research shows SDPMLP is a complex optimization problem in factorial calculation. For simple software defect predict model learning problems, PNN (probabilistic neural network) has better effect. For complex software defect predict model learning problems, the PNN algorithm based on improved simulated annealing algorithm has better effect. Kazu Okumoto presents a software defect predict model, using the method of modifying feature weighting to optimize the software defect predict result[4]. Stefan Lessmann and some other people use NASA software information dataset to test and evaluate software defect predict result [5]. The research shows that the distribution of software defect can be acquired in various software modules based on a measure of software attributes for expression. The research shows the expression of software attributes based on measure has access to the distribution of software defects in various software modules. From the above three typical study, currently, the project based on software defect predict model mainly aims at deeper research and verification on the neural network and multilayer perceptron network. In addition, for these traditional methods, some algorithms like PNN algorithm and simulated annealing algorithm is usually being introduced to improve the traditional algorithms.

Since 2012, there are more and more research on software defect predict model research based on BP algorithm technology, researchers made varying degrees of transformation of BP algorithm, also, they are trying to adjust algorithm to ensure BP algorithm can be applied to the software defect

predict model readily and effectively. Currently, studies on BP algorithm have received certain recognition in software defect predict model, but how to improve the software defect predict model based on BP algorithm reasonably will be the research focus.

Previous studies have shown that during the process that applying BP algorithm to software defect predict model, BP algorithm will produce excessive number of iterations and long computation time [6]. Therefore, this paper puts forward using JCUDA technology to improve the BP algorithm and the new method will speed up the calculation of BP algorithm effectively ,also it will optimize the BP algorithm based software defect predict model effectively.

Software defect predict model based on BP algorithm.

Software defect predict model based on BP algorithm is a software defect predict model which is established on the basis of software information and artificial neural network, reflecting if the software module information has defective estimate and judgment for the defect [7]. Software defect predict model based on BP algorithm consists of an input layer, a hidden layer and an output layer. The model is shown in Fig. 1.

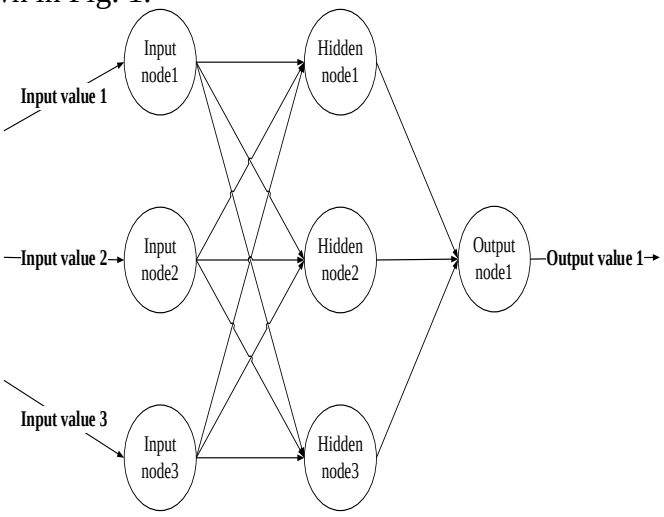

Figure 1. Software defect predict model network structure based on BP

As the Fig. 1 shows, the input node information represents information about the software modules, such as recurring number and number of judgement, the output indicates whether the module has a defective state. The input node information is shown in (1).

$$R = (r_1, \cdots, r_i)^T \quad (1)$$

R represents a set of learning sample information, r_i represents the value of the module information that is described by the i-th software information.

The output node information is shown in (2).

$$O = (o_1)^T \quad (2)$$

In the process that based on software defect predict model application, output node only has one output information, the information represents if there is an error in the module, thus output node has only one output value. In practical application process, research on software defect predict model based on BP algorithm needs to determine whether the node has an error on the basis of the output value. Calculation result closing to 0 indicates no error and the result closing to 1 indicates some error.

The learning process of defect predict model based on BP algorithm is on basis of the learning sample S, and it acquires the stable weight between the input layer and the hidden layer and the stable weight between the hidden layer and the output layer. In the implementation process, the hidden layer uses S-function as stress function setting, as (3) shows.

$$S(z) = \frac{1}{1+e^{-z \cdot k}} \quad (3)$$

As Equation (3) shows, k represents a positive integer parameter that controls the steepness of S

function, z represents the input value using this function. By the S function, extrusion effect can be achieved effectively and larger input value can be mapped to a small range conveniently, effectively and quickly. S function is also known as extrusion function as its effect. In addition, BP algorithm requires using the derivative in the process of adjusting weight reversely. S function can get the derivative conveniently, and the derivative of the S function can associate with the S input value accurately. Derivative of S function is shown in (4).

$$\frac{dS(z)}{dz} = S(z) \cdot (1 - S(z)) \tag{4}$$

As Equation (4) shows, S(z) can be obtained with the inputted computed result, and it achieves convenient calculation of its derivative.

The defect predict model algorithm based on BP algorithm uses the data of software defect model as the learning sample of BP algorithm. With this BP neural network learning data, the prediction for unknown software data will get better effect.

Software defect predict model based on JCUDA_BP algorithm.

Software defect predict model based on JCUDA_BP algorithm is an attempt to combine the JCUDA technology with BP algorithm. By learning sample decomposition and multi-threaded processing and CPU-GPU processing mode, the learning speed of the learning software defects model process is optimized.

CPU/GPU technology is a kind of collaborative computing method that has huge advantage currently, generally there are two ways, one is managing GPU with CPU, the other way is making GPU and CPU calculate together and GPU and CPU will undertake a part of the complex calculation respectively. The former way makes full use of the computational advantages of GPU, but it is a waste of computing resources of CPU, while the later way releases the computing power of the GPU and CPU well. The release is the future direction of the development of collaborative parallel computing [8].

Therefore, while using JCUDA to make BP algorithm optimization design, we need to consider collaborative way of CPU/GPU. Obviously, the latter that both the CPU and GPU undertake some part of the computational work has larger advantage than the other method. The computation with strong parallel is handles by GPU. CPU is responsible for both managing GPU and undertaking part of the non-parallel computation. Thus, the computing power of GPU / CPU can be stimulated effectively and the software defect predict model based on JCUDA_BP algorithm has higher computational efficiency.

First of all, JCUDA_BP algorithm optimizes the inputting stage of learning sample. If the learning sample is S and the number of the sample is N for software defect predict model, it can be shown in (5).

$$S = (s_1, \cdots, s_N) \tag{5}$$

JCUDA_BP algorithm divides learning samples into x groups in CPU, and the learning sample S can be written as (6).

$$S = S' = (s'_1, \cdots, s'_x) \tag{6}$$

In Equation (6), each s'_i corresponds to x/N learning samples. When the calculated value is not an integer, the grouping stage will automatically round down the value and get a parameter called n to represent the value. In Equation (6), the i-th group of data samples can be expressed using (7).

$$s'_i = (s_{1i}, \cdots, s_{ni}) \tag{7}$$

When it comes to the last group in the S sample, it will take all remaining samples automatically to ensure the accuracy and integrity of grouped data.

Then, software defect predict model based on JCUDA_BP algorithm starts multiple threads on the GPU, calculating the average error value respectively on the sample set S that is shown in (7). Error of all samples is expressed as (8).

$$\Delta e = \frac{\sum_{j \le x}^{j=1} \Delta e_j}{x} \tag{8}$$

In Equation (8), Δe_j represents the calculated error value after the combination and calculation of the j-th sample group, Δe represents the calculation error of the mean value of the whole sample set.

Then, basing on JCUDA_BP algorithm, adjust BP neural network weights with the reversal error of BP algorithm. After the adjustment, verify if the range of error value meets the requirement. If the error still cannot meet the accuracy requirement of the BP algorithm, the iteration will repeat until the result reaches best learning effect.

The software defect predict algorithm that combined with JCUDA_BP algorithm uses multi-threading and data grouping to speed up the effect of BP algorithm. However, in the actual use of the process, realizing the algorithm also need a computing monitor thread to ensure the calculation effect of each thread grouping sample. Once there is a sample that can't calculate the result, the monitor will discover, reminding CPU that do not calculate the average error for this sample. If the average error is greater than the error standard, then give up the thread sample calculation result and adjust the value of the weight of neural network. On the contrary, the thread waits for calculation results of the learning sample thread. If this thread has no response for a long time, then restart the calculation thread to ensure the accuracy of final average error value calculation.

Software defect predict model based on JCUDA_BP algorithm uses similar network structure to BP network, optimizes and simulates the runtime of the software defect prediction by the JCUDA algorithm. The process is shown in Fig. 2. In Fig. 2, Thread 1 to thread m is calculated in the GPU. CPU undertakes the task of error solution and verification, and also realizes the work of GPU running monitoring.

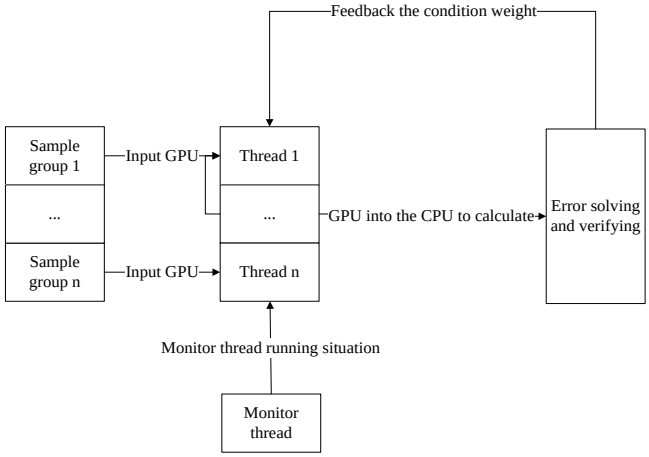

Figure 2. The process diagram based on JCUDA_BP.

The learning process of the study of software defect information based on JCUDA_BP algorithm is as follows:

a) *Complete BP neural network initialization in CPU and GPU.*

b) *Realize learning process with BP by JCUDA programming technology.*

c) *Using the calculation results of CPU storing BP algorithm.*

d) *Analysis and monitors the effect of result of the data in GPU by CPU*

Realizing BP algorithm with JCUDA is to make the iteration calculation process inside GPU, use CPU to analysis the error of the learning process, and adjust edge weight in GPU neural network topology.

Experimental analysis

B. Experimental data and environment

This paper mainly analyzes the computation efficiency of software defect prediction for standard BP algorithm and use GPU to deal with the experimental data. In order to obtain more obvious experiment effect and reflect the GPU processing effect for large data, NASA'S MDP (metric data program) dataset which is being used widely in the software defect research is chosen to be the experimental data. The data that being used in the experiment is shown as the table I:

TABLE I. EXPERIMENTAL DATA OF SOFTWARE DEFECT PREDICT MODEL BASED ON BP ALGORITHM AND JCUDA_BP ALGORITHM

Name	Training File Size (KB)	Number of sample data	Validation file size (KB)	Number of validation samples chosen
CM1	47	344	45	327
JM1	722	9593	607	7782
KC1	141	2096	88	1183
KC3	29	200	28	194
MC1	946	9277	234	1988
MC2	19	127	19	125
MW1	35	264	34	253
PC1	99	759	92	705
PC2	181	1585	90	745
PC3	145	1125	139	1077
PC4	169	1399	158	1287
PC5	1637	17001	219	1711

In general, the study data's learning number of input is different in software defect models, while the output has 2 types indicating if there is defect. Therefore, the iterations and the number of hidden nodes in the second layer have greater influence on the software defect predict model based on BP. The experiment aiming at BP algorithm efficiency needs to adjust the number of hidden nodes and investigate how the number works for final calculation time. In the experiment, the number of hidden nodes is set to 1024, 2048, 8192, 10384, 32768. The number of hidden nodes affects the final number of iterations. Then collect the runtime respectively to analysis of the influence of different virtual nodes number on BP model.

The experiment uses Windows2007 operating system and the memory size is 4GB. It uses Eclipse as programming environment, realizes the GPU and CPU programming under JCUDA technology based on Java. The experiment combines GPU and CPU and increase the processing efficiency of BP algorithm. GPU's maximum number of multi-thread in each module is 10536 and maximum number of threads in each module is 1024, the largest memory pool is 2147483647 bytes.

C. Experiment result and analysis

This paper writes from the following two aspects, one is "the contrastive analysis between the software defect predict based on BP and the software defect predict based on JCUDA_BP", the other is "the analysis of the software defect predict influence of the different hidden node number on JCUDA_BP ", illustrates the different between software defect predict model based on JCUDA_BP algorithm and software defect predict model based on BP algorithm, elaborates the

problems and the influence that may be produced in the process of using JCUDA_BP.

1) Experimental results and analysis of rate improving

The experiment shows, under the same experimental data, if the computation time of different dataset will change obviously using BP algorithm that is improved with JCUDA technology. The computation time, computation time difference and the optimized ratio is shown in Table II.

TABLE II. COMPARISON BETWEEN THE SOFTWARE DEFECT PREDICT TIME BASED ON BP AND THE SOFTWARE DEFECT PREDICT TIME BASED ON JCUDA_BP

Name	Numb-er of sample data	CPU computati-on time (ms)	GPU comput-ation time (ms)	Computa-tion time difference (ms)	Optim-ized ratio (%)
CM1	344	29914	31709	-1795	-6.00
JM1	9593	1064892	851914	212978	20.00
KC1	2096	272858	226472	46386	17.00
KC3	200	26150	26935	-785	-3.00
MC1	9277	1329638	1077007	252631	19.00
MC2	127	19456	19651	-195	-1.00
MW1	264	35943	36123	-180	-0.50
PC1	759	93170	76399	16771	18.00
PC2	1585	210509	176828	33681	16.00
PC3	1125	137306	119456	17850	13.00
PC4	1399	174587	141415	33172	19.00
PC5	17001	3563642	2815277	748365	21.00

As Table II shows, for dataset PC5, the computation time with software defect predict model optimized by JCUDA time is smaller than software defect predict model based on BP algorithm. As for the sample CM1, using the software defect predict model optimized by JCUDA makes the learning time longer. The number of learning sample of PC4 and PC2 is closing and optimization of learning process is also closing..

According to the Table II, when using JCUDA to improve BP algorithm, the relationship of the learning ratio, sample number, algorithm are as follows:

(1) When the number of learning samples of software defect data is large, the learning time of software defect predict model based on BP and the learning time of software defect predict model based on JCUDA_BP will both increase with the adding of learning samples.

(2) When the number of learning samples of software defect data is small, the learning time of software defect predict model based on BP will smaller than the learning time of software defect predict model based on JCUDA_BP.

(3) With the adding of the learning sample of the software defect model, the optimized ratio of the learning process of the modified BP algorithm based on JCUDA will increase. It means that the effect of BP algorithm modified by JCUDA is better with more learning samples.

(4) During the process of achieving BP algorithm based on CPU, excessive learning sample will cause learning failure. This phenomenon is mainly due to the presence of memory overflow in calculation process of program execution. Realizing BP algorithm with JCUDA technology will effectively reduce the probability of the phenomenon of memory overflow. To solve the memory overflow, distribute the software defects module learning sample information to different threads during JCUDA programming.

With the above experiment, for the MDP software model information dataset, BP algorithm modified by JCUDA can optimize the learning process for a certain degree and acquire better optimization. However, this optimization process performs obviously when the number of software defect model learning sample is large. The optimization is not good if the number is small.

2) The result and analysis for hidden nodes experiment

When the number of initial input nodes is same, sample data is same and learning efficiency set is same, if the number of hidden nodes based on JCUDA_BP model increases, then use JCUDA_BP

algorithm to learn the data of PC5 learning dataset. Execution time is shown in Table III.

TABLE III. COMPARISON OF PROCESSING TIME OF GPU\CPU WITH THE NUMBER OF HIDDEN NODES IN PC5

The number of hidden nodes	Start-up time of GPU（ms）	Runtime of GPU（ms）	Time of CUP（ms）
1024	4732	46601	37392
2048	3162	51516	50027
8192	3910	170976	176077
16384	5375	323316	337000
32768	4929	628398	740592

As seen from Table III, when increasing the number of hidden nodes, the time of running BP software defect simulation algorithm on GPU and CPU will increase. When the number of hidden nodes is fewer, CPU processing time will be shorter. The GPU computation also need the start time of GPU, which will extend the final running time of GPU. However, after subtracting the start time of GPU, the running time of GPU on BP algorithm is close to the CPU time. When the number of hidden node increases again, GPU start time will fluctuate, but still remaining at a relatively stable range. With the increase of the number of hidden nodes, GPU computing time will be better than CPU time. When the number of hidden nodes reached an order of magnitude, the learning effect of optimized BP algorithm under GPU is much better than BP algorithm under CPU.

Conclusion

According to “software defect predict model experiment based on BP/JCUDA_BP” and “software defect predict model experiment using JCUDA_BP based on different number of hidden nodes”, software defect predict model based on JCUDA_BP can handle multiple learning samples of training samples quickly and accurately and it will also speed up the computation time effectively. When increasing the number of hidden nodes, JCUDA_BP algorithm is still able to obtain a shorter processing time and speed up the operation efficiency. The research on software defect predict model based on JCUDA_BP combines the new thoughts of parallel processing and GPU technology, making the use of BP algorithm in the process of software defect prediction more quickly and efficiently, enabling JCUDA_BP’s rapid learning of software defect data and ensuring the availability between BP algorithm and software defect prediction model.

Acknowledgment

This work was supported by the Key Project of National Defense Basic Research Program of China (Grant No. B1120132031) and the Ph.D. Programs Foundation of Ministry of Education of China (Grant No. 20131101120043).

References

[1] S. Naeem, K. M. Taghi, H. V. Jason. Predicting faults in high assurance software [C]. In Proceeding of the 12th IEEE International High Assurance Systems Engineering Symposium, San Jose, USA, pp.26-34, 2010.

[2] W. Qing, W. Shujian, L. Mingshu. Software defect prediction [J]. Journal of Software, vol 19(7): 1565-1580, 2008. .DOI:10.3724/SP.J.1001.2008.01565. (In Chinese)

[3] P. C. ParaC. Exhaustive and heuristic search approaches for learning a software defect prediction model[J]. Engineering Applications of Artificial Intelligence, vol 23(1): 34-40, 2010.

[4] O. Kazu. Software defect prediction based on stability test data [C]. Proceedings of 2011 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering, pp.385-387, 2011.

[5] L. Stefan, B. Bart, M. Christophe, P. Swantje. Benchmarking classification models for software defect prediction: A proposed framework and novel findings [J]. IEEE Transactions on Software Engineering, vol 34(4): 485-496, 2008. doi: 10.1109/TSE.2008.35

[6] Y. Anlei, P. Dechang. Software defect prediction model based on PSO-BP [J]. Computer Engineering and Applications, vol 49(7): 64-67, 2013. DOI:10.3778/j.issn.1002-8331.1208-0533.(In Chinese)

[7] Yinran. Research on software defect predict model based on SAPS0-BP [D]. South-West University, 2014.

[8] F. F. Igual, R. Mayo, E. S. Quintana-orti. Attending High Performance in General-purpose Computations on Current Graphics Processors [C], VECPAR, LNCS 5336.2008:406-419, 2008.