文章编号:1671-4598(2017)03-0123-03 DOI:10.16526/j.cnki.11—4762/tp.2017.03.034 中图分类号:TP3 文献标识码:A

基于增强 GA—BP 神经网络的软件错误定位方法

张 蓓,张树东
(首都师范大学 信息工程学院,北京 100048)

摘要:在软件开发和后期维护的过程中,进行软件调试来定位错误并修正错误是其中最复杂且成本最大的一部分;文章针对现有基于神经网络的软件错误定位方法中的权值和阈值设定不方便、鲁棒性差等问题,结合正交实验设计思想和遗传算法(Genetic Algorithm),提出了一种基于增强遗传 BP 神经网络的软件错误定位方法;并将其同基于 GA—BP 神经网络的和基于 BP 神经网络的定位方法都在 MATLAB 上进行了实验,实验数据来源西门子测试集,从结果上看,基于增强 GA—BP 神经网络的软件错误定位方法在定位错误的效率和精确度上都有一些进步。

关键词:错误定位;GA—BP 神经网络;正交实验设计

Fault Localization Method Based on Enhanced GA—BP Neural Network

Zhang Bei, Zhang Shudong
(College of Information Engineering, Capital Normal University, Beijing 100048, China)

Abstract: In the process of software development and maintenance, software debugging is the most complicated and the most expensive part. During the period of traditional software debugging, programmers have to locate mistakes by browsing codes, this is a time—consuming and laborious work. There has been a great need for fault localization techniques that can help guide programmers to the locations of faults. In recent years, automated software fault localization technology has attracted many scholars' attention, various approaches have been proposed. In this paper, a technique named EGA—BPN is proposed which can propose suspicious locations for fault localization automatically without requiring any prior information of program structure or semantics. EGA—BPN is a software fault localization method based on enhanced Genetic Algorithm—Back Propagation neural network. Firstly, through processing running traces of the program, covering information of test cases are converted as the training samples of neural network; secondly, the data are input into neural network in training orderly, the initial weights of neural network are computed by GA, then test matrix is calculated by the neural network to count the suspiciousness of each statement, and using orthogonal experimental design to adjust the parameters of neural networks; finally, the fault is located at the statements with higher suspicious value. Through experiment on the proposed method and GA—BPN and BPN were compared, the results show that the enhanced GA—BP neural network—based fault localization technology has certain validity.

Keywords: software fault localization; GA—BP neural network; orthogonal experimental design

0 引言

随着时代的进步,计算机正渗入社会的各个角落,逐步改变着我们的生活,手机、电脑等电子产品使人与人之间的距离变短,就算相隔千里都变成了一根网线的距离;在冰箱、空调、电视等家电上安装的智能控制系统使这些电器变得更加“听话”;物联网、智能社区、智慧城市等技术的发展使人类的生活变得更加信息化、智能化。然而,计算机技术的发展不仅能带来便利快捷的一面,还会产生不好的影响,例如黑客利用网站的漏洞来攻击某些网站,妨碍网站的正常运营或者获取用户资料来进行一些非法活动;游戏代码存在漏洞,会造成一些外挂软件的产生,从而造成一个不公平的游戏环境,损害游戏玩家的利益。所以,改进软件可靠性迫在眉睫。

为了提高软件的质量,就必须尽可能多地找到软件中存在的问题,才能解决问题,而发现问题是其中最难的一个环节。在传统的错误定位方法中,都是靠工作人员手工调试代码来找出错误的所在位置,主要方法是在代码中设置断点或者插入某一段测试程序,跟踪程序运行步骤,根据程序运行状态是否正常,捕捉到的中间值是否同预期相符合来进行错误定位,这种方式不仅花费时间多、效率低下,还很枯燥乏味,还可能在代码中注入其他语句,造成不可预知的影响。如果能实现自动化错误定位,就能将工作人员从繁琐的工作中解放出来,提高软件调试效率,降低软件开发成本。

近几年,中外学者对软件错误定位的自动化做了各种研究,成果丰富,根据原理不同可以大致分为基于切片、基于程序光谱和基于状态修改的错误定位方法。

基于程序切片(slicing—based)[1-3]的错误定位方法:程序切片,顾名思义,就是缩小范围,该方法的主要思想是构建一个程序中可能同错误输出有关的代码的集合,包括两个内容:错误所在的语句和同错误语句相关的,可供工作人员理解程序的调试上下文,尽可能的缩小可疑语句存在的范围,减少需要检测的代码行数,提高软件调试的效率。这种方法可适用于规模较大程序的测试,但是由于需要事先对程序进行依赖关系分析,过程复杂,需要耗费大量资源,而且往往处理后的代码量

收稿日期:2016-01-18; **修回日期**:2016-02-26。

作者简介:张 蓓(1992-)女,硕士研究生,主要从事软件测试方向的研究。

张树东(1969-)男,教授,博士,主要从事计算机网络和分布式计算等方向的研究。

还是很大，冗余度很高，需要做进一步的简化。

基于程序光谱（spectrum－based）的错误定位方法：程序光谱是指在程序执行测试用例过程中统计的信息[4]。可以通过在程序中定位不同的元素，例如可执行语句或语句块[5]、谓词[6]、方法[7]、信息流路径[8]，来获取不同的程序光谱，这个方法的主要思想是利用成功执行案例的程序光谱信息同失败执行案例的程序光谱信息之间的差异来定位错误，基于统计数据或者各种数学计算方法获得程序中的每个元素的可疑度并排序，排名越靠前，可疑度越高，该元素越可能出现错误。这种方法的复杂度低，并且工作人员可以直接根据语句的可疑度的不同，从可疑度最大的语句开始排查，从而能更快的定能错误，但是它没有分析程序元素间的关系，并且结果容易受到某些特殊性质的测试数据的干扰。

基于程序状态（state－based）[9-10]的错误定位方法：基于程序状态的错误定位方法的中心思想是首先统计正确测试用例执行过程中的运行状态和故障测试用例执行过程中的运行状态，将两者作对比得出差异结果，再按照不同的规则对失败测试用例的运行状态进行修改，根据修改后得出的测试结果来找出错误所在关键语句的位置。这种方法的复杂度低，且能有效减少需要被检测的运行状态的数量，效率高，但是能定位的错误的类型很少。

除了传统的错误定位方法外，近些年，国内外有些学者提出了将神经网络应用于软件错误定位领域中，例如在 2009 年，Wong 等提出的基于 BP 神经网络的错误定位算法[11]。由于神经网络算法中含有大量的参数，这些参数数值的设置对错误定位的最终结果的准确度有很大的影响。本文基于这些算法，融合遗传算法思想（Genetic Algorithm）和正交试验设计的思想（Orthogonal Experimental Design，OED），提出了一种新的软件错误定位方法——基于增强遗传算法和 BP 神经网络的软件错误定位方法 EGA－BPN，利用遗传算法来计算 BP 神经网络参数的最初设定值，并且采用正交试验设计方法来调整 EGA－BPN 中的参数值，使 EGA－BPN 的更适用于错误定位。

1 增强 GA－BP 神经网络模型（EGA－BPN）

BP 神经网络（Back Propagation）分为输入层、隐含层和输出层，结构如图 1 所示，其中隐含层可能是一层，也有可能是多层，每一层中都含有神经元，神经元的数量不固定，大多数情况下只能通过尝试的方法来确定[12]，它对于非线性或者其他更复杂的关系的识别需要训练很多的次数，而且无法排除局部极小点，算法时间消耗大。针对 BP 神经网络的这些缺点，国内外学者做了很多研究，其中一种改进方法是利用遗传算法能“优胜劣汰，适者生存”的能力克服 BP 神经网络算法的缺陷和不足，从而进行一定的优化升级。

EGA－BPN 的主要思想是用 GA 来处理神经网络各层之间的连接权重和阈值，筛选出其中的最佳个体，即误差最小的一组，将其作为神经网络的权值和阈值的最初设定值，将输入值代入神经网络后，用网络的预测结果同预期输出值作比较，根据误差继续利用 OED 调整网络参数值，再重新执行直到神经网络训练完成。将测试数据作为网络输入，并对输出结果分析总结。图 2 是 EGA－BPN 模型执行流程图：

EGA－BPN 模型训练过程如下：

1）首个任务是编码来获得遗传初始种群，每个编码串也

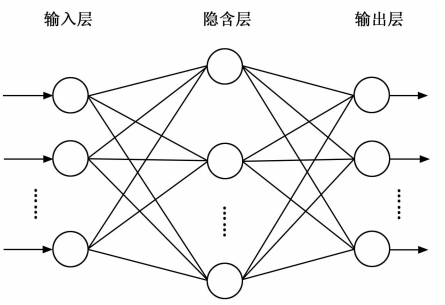

图 1 BP 神经网络结构

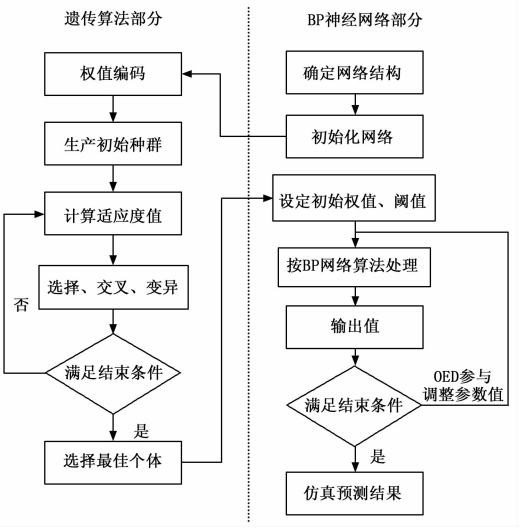

图 2 EGA－BPN 流程图

就是每个染色体包含了神经网络各层之间的连接权值和阈值，通过随机函数产生 N 个。

2）确定神经网络的结构，分别计算染色体的基因遗传给后代的可能性，也就是其适应性值 F，F 是根据第 i 个训练数据经网络处理的训练预测值 y_i 同其期望输出值 o_i 之间的误差来定义的，为了使结果更精确和易于判断，通常采用误差的绝对值来作为判断依据，适应值越大的染色体，其基因遗传下去的可能就越大，F 的计算公式如下：

$$F = k\left(\sum_{i=0}^{n} abs(y_i - o_i)\right) \tag{1}$$

其中：n 为网络输出节点数，k 为系数。对染色体进行交叉、变异操作，将经过层层筛选得出最终个体作为神经网络权值和阈值的最初设定值。

3）设神经网络的输入层、隐含层、输出层的节点数量分别为 m、p 和 q，三层神经元之间的连接权值分别是 w_{ij} 和 v_{jk}，隐含层阈值为 α_i，输出层阈值为 β_k，确定学习参数以及神经元激励函数 $f(x)$，该函数一般用 sigmoid 传递函数。将网络输入数据用矩阵表示为：

$$X = [X_1, X_2, \cdots X_m] \tag{2}$$

4）隐含层输出 Y 可以由输入向量 X、w_{ij} 和 α_i 计算得出，具体公式如下：

$$Y_j = f\left(\sum_{i=1}^{m} w_{ij} X_i - \alpha_i\right) j = 1,2,\cdots p \tag{3}$$

5）输出层结果 Z 可以由 Y、v_{jk} 和 β_k 计算得出，具体公式

如下：

$$Z_k = f\left(\sum_{j=1}^{p} v_{jk} Y_j - \beta_k\right) k = 1,2,\cdots,q \tag{4}$$

6）若神经元 X_i 的期望输出值是 O_k，那么可以根据如下均方误差函数公式求出误差 E：

$$E = \frac{1}{2}\sum_{k}(O_k - Z_k)^2 \tag{5}$$

7）将误差 E 同预设值作对比，如果小于预设值或者训练次数已达到预设次数，则代表训练完毕，反之，要按照如下公式进行误差修正：

$$\begin{cases} w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij} \\ v_{ik}(n+1) = v_{ik}(n) + \Delta v_{ik} \end{cases} \tag{6}$$

其中：n 表示网络的训练次数，Δw_{ij} 和 Δv_{ik} 分别表示权值阈值的修正值，公式如下：

$$\begin{cases} \Delta w_{ij} = -\sigma \dfrac{\partial E}{\partial w_{ij}} \\ \Delta v_{jk} = -\sigma' \dfrac{\partial E}{\partial v_{jk}} \end{cases} \tag{7}$$

8）在公式（7）中，σ 和 σ' 表示学习参数，结合 OED 来调整学习参数的值[13]。将修正后的参数作为新的网络连接权值和阈值，返回第（3）步，直到达到理想状态或者训练次数达到设定值。

2　基于 EGA－BPN 的软件错误定位算法

基于 EGA－BPN 的软件错误定位算法流程如下：

1）获得程序执行信息。对于一个存在错误并且能正常执行的程序 P，例如表 1 中两数相加求和程序[14]，有 9 条语句，按照语句顺序将其编号为 S_1、S_2……S_9，其中存在错误语句 S_3。对于一个测试用例 $t_k = <in_k, out_k, C_k, flag>$，其中，$in_k$ 表示其输入信息；out_k 表示其输出信息；C_k 表示其程序覆盖信息，在 t_k 运行过程中，执行了语句 S_i，则将 C_k 中标识该语句的覆盖标识符 C_{ki} 设为 1，否则设为 0；$flag$ 是程序执行结果标识符，如果 out_k 等于 in_k 期望输出值，那么测试用例 t_k 执行成功，将其 $flag$ 的值设为 0，反之 out_k 不等于 in_k 期望输出值，那么 t_k 执行失败，将其 $flag$ 的值设为 1。对于一些规模较大的程序，很难实现按照语句的顺序来得到测试用例的覆盖信息，可以更换程序元素，换成谓词、方法等，减少神经网络处理的数据量，提高错误定位效率。

表 1　获取语句覆盖信息实例

	程序	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
S_1	read(a,b);	1	1	1	1	1	1	1	1
S_2	if (a<10&&b<10)	1	1	1	1	1	1	1	1
S_3	result=a−b;//应该为 a+b	1	1	1	1	0	1	1	1
S_4	if(result>0)	1	1	1	1	0	1	1	1
S_5	print(“positive”);	1	0	0	0	0	1	1	0
S_6	else if (result==0)	0	1	1	1	0	0	0	1
S_7	print(“zero”);	0	1	0	0	0	0	0	0
S_8	else print(“negative”);	0	0	1	1	0	0	0	1
S_9	else print(“invalid input”);	0	0	0	0	1	0	0	0
$flag$		1	1	1	1	0	0	0	0

2）编码神经网络各层之间的权值阈值，形成遗传算法的进化初始种群，利用交叉、变异算子计算适应度最高的染色体，遗传给下一代，挑选出最优个体，最终得出的结果就是 BP 神经网络参数的最初设定值；

3）接下来就是训练神经网络，将测试用例 t_k 的程序覆盖信息 $C_k = <C_{k1}, C_{k2}, \cdots, C_{kn}>$ 作为训练神经网络的输入层神经单元，表 1 中的实例，训练数据有 8 组，每组有 9 个输入神经元。再利用三层之间的连接参数按照一定的公式对输入数据进行一系列的计算，得到输出值，将实际输出值同期望输出值 $flag$ 作比较，得出误差，如果误差值大于设定误差值，则利用 OED 调整参数值。循环这个过程，直到误差值小于等于设定误差值或者循环次数已经超出最大循环值，循环停止，神经网络训练完成。

4）一旦神经网络训练完成，就在输入数据和输出数据之间建立了良好的计算网络，用一组虚拟测试用例 v_1，v_2，…，v_m，每一个测试用例 v_i 依次覆盖一条语句 s_i，测试用例的覆盖信息是 C_{v_1}，C_{v_1}，…，C_{v_m}，用矩阵的形式表示如下[15]：

$$\begin{bmatrix} C_{v_1} \\ C_{v_2} \\ \vdots \\ C_{v_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{8}$$

将虚拟测试用例数据赋值给网络的输入层，预测结果为 f_1，f_2，…，f_m。f_i 的值越靠近 0，那么表示该测试用例 v_i 运行结果很可能成功，其覆盖语句 s_i 出错的可能性较小，反之 f_i 的值越靠近 1，那么表示 v_i 的运行结果很可能失败，s_i 也可能存在错误，所以可以将输出结果 f_1，f_1，…f_m，看成覆盖语句 s_1，s_2，…s_m，的错误可疑率，可疑率越大的语句，出错概率越高，需要被检查的优先级也要提前。

5）将神经网络的输出结果，即语句的错误可疑率按序进行排列，给工作人员提供参考，工作人员可以直接从可疑率最大的语句进行错误排查，从而节省了很多时间，提高工作效率。

3　实验结果与分析

EGA－BPN 算法的实验需要测试用例的 4 个元素：输入数据、输出数据、覆盖信息和运行结果标识符。如果测试程序语句量较大，首先要对程序分块，将块作为实验最小单位，再执行测试用例，生成覆盖文件，对覆盖文件进行过滤、分离、抽取等操作，得到测试用例的覆盖信息，最后将系统运行得到的输出数据同预期值进行比对，得出运行结果标识符，最后将这些信息进行整合，归纳，筛选，得出类似表 1 的数据文件，将其作为 EGA－BPN 的输入数据，赋值给之前已经训练完成的神经网络，将网络的预测结果按序排列，排名越靠前的代码块越需要被检查是否有错误。将表 1 中的示例程序按照 EGA－BPN 算法计算出的结果如下：

表 2　示例执行结果

语句	可疑度	语句	可疑度	语句	可疑度
S_1	0.8356	S_4	0.35486	S_7	0.00915
S_2	0.9775	S_5	0.09429	S_8	0.0756
S_3	0.9998	S_6	0.68125	S_9	0.0996

按照表 2 的数据我们看出，错误存在的语句 S_3 的可疑度数值最大，最可能出现错误。为了进一步论证本文方法的有效性，引入了此西门子套件集（Siemens suit）作为进一步实验的

（下转第 129 页）

电力信息化，2012，10（2）：65－68.
[2] 朱碧琴. 不动产登记信息数据整合及管理基础平台建设［J］. 低碳世界，2016，5（13）：145－202.
[3] 王春海，刘晓辉，白风涛. 信息资源整合中的风险及其防范策略［J］. 电力信息化，2009，7（9）：43－48.
[4] 高　磊，肖建飞，范俊杰，等. 企业门户与目录系统的研究与应用［J］. 电气信息化，2011，9（2）：138－143.
[5] 李立博. 面向服务的多源异构数据整合平台的设计［J］. 计算机工程与设计，2011，32（1）：141－144.

（上接第 125 页）

数据来源，西门子套件集是由 Software－artifact Infrastructure Repository（SIR）[16] 提供，大部分由 C 语言编写，每个程序都有正确版本、错误版本和测试用例，并且错误都是由人工添加进程序的，有语句赋值错误、谓词错误等，错误类型都和实际工程中可能发生的类型相似。但是，并不是套件集所有的错误版本都适合本次实验，例如 print _ tokens 的错误版本 4 的错误语句存在于头文件中，不能统计错误语句的覆盖信息；例如 replace 的错误版本 27 和错误版本 32，在执行测试用例过程中会发生错误，从而会产生程序异常终止，不能得出输出结果数据。

为了检验算法的有效性，引入定位有效率作为评价指标，算法的定位有效率是指在定位错误过程中所花的精力，也就是检测出错误百分比同已检测代码百分比的比值，假设 A 算法和 B 算法同样检测出了 55％的错误，但是 A 算法已检测代码占总代码的 30％，而 B 算法已检测代码占同代码的 50％，那么显然，A 算法的定位有效率的值小于 B 算法的定位有效率的值，A 算法的计算效率比 B 算法的计算效率更高，性能更优。图 3 展示了 EGA－BPN 同 GA－BPN 和 BPN 的定位有效率的比较。

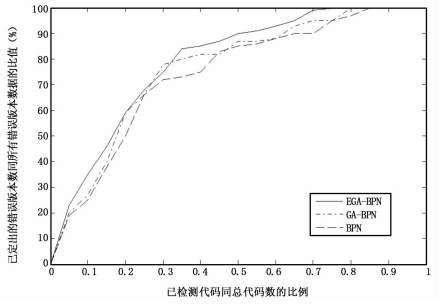

图 3　EGA－BPN、GA－BPN、BPN 的定位有效率比较

根据图 3 数据，我们可以看出，在已检测同样的代码数的情况下，EGA－BPN 能定位更多的错误，同样，在定位同样多错误的情况下，EGA－BPN 只需检测更少量的代码，相比较而言，EGA－BPN 具有更高的效率和精确度。

4　总结与展望

本文将遗传算法、BP 神经网络和正交试验设计融合，应用于错误定位领域，提出了增强 GA－BP 神经网络的软件错误定位算法，实验证明，该算法在定位错误的效率和精确度方面确实比 GA－BPN 和 BPN 具有一定的提高。虽然该错误定位算法取得了良好的实验结果，但是该算法还存在很多不足，例如：实验结果只是给出了错误语句的可疑度排名，还需要工作人员逐一排查，手动修改错误并再次执行程序以确认错误是否已修改；另外，神经网络预测的准确性同训练数据有较大关系，训练数据越多，预测结果相对来说就更准确，但是在实际工程中，由于程序结构或者逻辑都比较复杂，获得程序的测试用例、覆盖信息、执行结果等信息需要耗费大量的时间，会降低测试效率。这些不足在后续研究中会继续优化改进。

参考文献：

[1] Sterling C D，Olsson R A. Automated bug isolation via program chipping［J］. Software：Practice and Experience，2007，37（10）：1061－1086.
[2] Zhang X，Gupta N，Gupta R. Locating faulty code by multiple points slicing［J］. Software：Practice and Experience，2007，37（9）：935－961.
[3] Wong W E，Qi Y. Effective program debugging based on execution slices and inter－block data dependency［J］. Journal of Systems and Software，2006，79（7）：891－903.
[4] Abreu R，Zoeteweij P，Van Gemund A J C. On the accuracy of spectrum－based fault localization［C］. Testing：Academic and Industrial Conference Practice and Research Techniques Mutation，2007：89－98.
[5] Jones J A，Harrold M J，Stasko J. Visualization of test information to assist fault localization［C］. Proceedings of the 24th international conference on Software engineering，ACM，2002：467－477.
[6] Masri W. Fault localization based on information flow coverage［J］. Software Testing：Verification and Reliability，2010，20（2）：121－147.
[7] Murtaza S S，Madhavji N，Gittens M，et al. Diagnosing new faults using mutants and prior faults（NIER track）［C］. IEEE 33rd International Conference on Software Engineering，2011：960－963.
[8] Yu K，Lin M，Gao Q，et al. Locating faults using multiple spectra－specific models［C］. Proceedings of the 2011 ACM Symposium on Applied Computing，2011：1404－1410.
[9] Zhang X，GuPta N，GuPta T. LocatingFaultsthroughAutomatedPredicate Switching［C］. Inthe28thInternational Conference onSoftwareEngineering（ICSE. 06），May2006：272－281.
[10] Cleve H，Zeller A. Locating Causes of Program Failures［C］. In the 27th International Conference on Software Engineering（ICSE. 05），2005：342－351.
[11] Wong W E，Qi Y. BP neural network－based effective fault localization［J］. International Journal of Software Engineering and Knowledge Engineering，2009，19（4）：573－597.
[12] 史忠值. 神经网络［M］. 北京：高等教育出版社，2009.
[13] 张　柯，张德平，汪　帅. 基于增强径向基函数神经网络的错误定位方法［J］. 计算机应用研究，2015，3.
[14] Eric Wong W，VidrohaDebroy and et al. The DStar Method for Effective Software Fault Localization［J］. IEEE Transactions on Reliability，2014，63（1）：290－308.
[15] Wong W E，Debroy V，Thuraisingham B and et al. RBF Neural Network－based fault location［J］. Technical report UTDCS－20－10，2010.
[16] SIR，http：//sir. unl. edu/portal/index/php［EB/OL］.