

文章编号: 1006-2475(2015) 11-0027-05

基于遗传神经网络的软件错误定位方法

吕琼帅, 单冬红, 申 远

(平顶山学院软件学院, 河南 平顶山 467000)

摘要: 软件错误定位的准确率有助于提高软件调试的效率。通过对软件运行时信息的分析, 提出一种利用遗传神经网络来提高软件错误定位准确率的方法。首先, 采集程序运行过程中的相关信息并将其编码, 将此编码作为遗传神经网络的训练数据集; 然后, 构建虚拟测试集, 并根据虚拟测试集计算程序语句的可疑度, 再根据可疑度的大小来定位软件错误; 最后, 利用 Siemens Suite 中 132 个预先植入错误的程序进行实验。实验结果表明, 遗传神经网络能够在一定程度上提高软件错误定位的准确率, 对软件调试工作起到一定的帮助作用。

关键词: 错误定位; 遗传神经网络; 可疑度; 软件调试

中图分类号: TP311

文献标识码: A

doi: 10.3969/j.issn.1006-2475.2015.11.006

Software Fault Localization Based on Genetic Neural Network

LYU Qiong-shuai, SHAN Dong-hong, SHEN Yuan

(Software College, Pingdingshan University, Pingdingshan 467000, China)

Abstract: Software fault localization accuracy helps to improve software debugging efficiency. According to the analysis of software runtime information, a method is proposed to improve the accuracy of software error locating by using genetic neural network. First, the information is collected and encoded while the program is running. The encoding is considered as a genetic neural network training data set. Then, building a virtual test set, the program statement suspicious degree is calculated according to the virtual test set and the software error is positioned according to the size of the suspicious. Finally, this paper conducts on the Siemens Suite of 132 programs with injected bugs. The experimental results show that the genetic neural network can improve the accuracy of locating software faults to some extent and is helpful for software debugging.

Key words: fault localization; genetic neural network; suspicious degree; software debugging

0 引 言

软件行业的快速发展使得软件的规模不断扩大, 功能不断完善。软件在设计和开发过程中难免会出现人为错误或逻辑错误, 这就增加了软件系统测试和维护的成本。随着软件系统复杂性的不断增大, 在发现和定位软件错误时所花费的时间和人力成本也逐渐增加。因此, 如何有效地提高软件错误定位技术的效率和准确率, 降低软件调试和维护成本, 已成为软件故障领域研究的热点问题。

所谓软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态^[1-2]。为了提高软件故障定位的效率, 国内外学者从不同角度提出了很多行

之有效的方法。如陈浩等提出了利用硬件故障定位法结合遗传算法来进行快速的软件故障定位^[3]。Jone 等提出了根据统计程序实体可疑度定位软件故障的 Tarantula 方法^[4-5]。Wong 等人提出了一种基于 RBF 神经网络的软件错误定位方法^[6]。秦兴生等提出了利用神经网络集成的软件故障预测方法^[7]。张柯等提出了基于增强径向函数神经网络的软件错误定位方法^[8]。

本文从软件错误定位的轻量级^[9]的角度出发, 通过收集测试程序的覆盖信息, 利用机器学习法进行软件错误定位。与文献[10]不同的是, 本文提出一种利用遗传算法来优化 BP 神经网络的方法——遗传神经网络(GABP)来定位软件的错误。最后, 通过

收稿日期: 2015-04-03

基金项目: 河南省科技厅项目(132102310516); 平顶山学院青年基金重点项目(PDSU-QNJJ-2013002)

作者简介: 吕琼帅(1985-), 男, 河南平顶山人, 平顶山学院软件学院讲师, 硕士, 研究方向: 机器学习, 智能算法; 单冬红(1976-), 女, 副教授, 硕士, 研究方向: 软件工程, 网络安全; 申远(1983-), 男, 讲师, 硕士, 研究方向: 信息安全及密码学, 软件工程。

实验对比表明,该方法能够较为准确地进行软件错误定位。

1 遗传神经网络

1.1 遗传算法

遗传算法(Genetic Algorithms)是由美国 Michigan 大学 Holland 教授于 1975 年提出的模拟自然界生物遗传机制和生物进化论而形成的一种并行随机搜索优化方法^[11]。遗传算法没有复杂的求导和函数连续性要求,采用概率化的寻优方法能够自适应地调整搜索方向指导搜索空间,因此具有更好的全局搜寻能力。基于这些优点,遗传算法已经广泛应用于智能优化、机器学习、车间调度和信号处理等方面。

遗传算法的基本过程描述如下:

- 1) 初始化阶段。设置种群规模,进化代数,交叉概率,变异概率,随机生成初始种群。
- 2) 评价体制。计算初始种群中每个个体的适应度。
- 3) 运算阶段。选择算子:根据评价体制选取种群中较为优化的个体遗传到下一代;交叉算子:将种群中的个体按照交叉概率进行交叉操作;变异算子:将种群中的个体按照变异概率对某些基因座上的值进行变动。
- 4) 算法终止。如果达到种群的最大进化代数,则选取种群中具有最大适应度的个体作为最优解,算法结束;反之,继续迭代。

1.2 神经网络

神经网络模型已经广泛应用于解决软件风险评估^[12]、模式识别^[13]和软件可靠性预测^[14]等方面。其中,BP神经网络能够较好地解决复杂的非线性问题,是神经网络模型中比较简单也是应用较广的模型之一。BP神经网络的核心是信号正向传播,误差反向传播。文献[15]中证明了3层BP神经网络能够拟合任何一种非线性关系。其结构如图1所示。

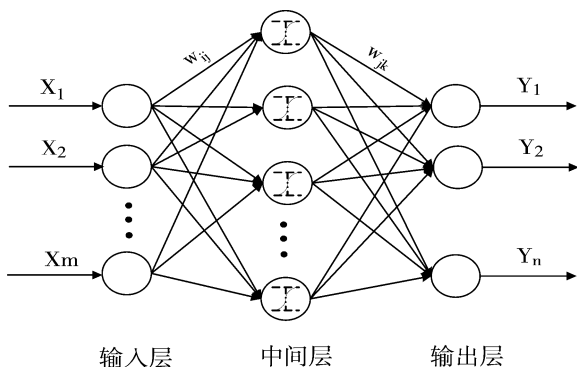


图1 3层BP神经网络结构图

在3层的BP网络中,网络的实际输出为:

$$Y(n) = [x_1, x_2, \dots, x_m] \quad (1)$$

其中, x 表示输入样本向量,其数量 m 由每个样本的属性决定。 Y 表示网络的实际输出向量,其数量 n 由问题结果决定。网络的期望输出为:

$$D(n) = [x_1, x_2, \dots, x_m] \quad (2)$$

其中, x, m, n 含义同公式(1), D 表示样本中的目标结果。

误差信号定义为:

$$e(n) = D(n) - Y(n) \quad (3)$$

通过不断地训练与迭代,直到网络实际输出值在误差允许的范围内无限地逼近训练样本的真实值则表明BP网络模型已经训练完成。

1.3 遗传算法优化的神经网络

遗传神经网络是将遗传算法与BP神经网络进行有机的结合,通过利用遗传算法对BP神经网络进行优化,从而在一定程度上避免BP神经网络收敛速度慢、容易陷入局部极小值和易震荡等缺陷。利用遗传算法优化BP神经网络的方法描述如下,具体过程如图2所示。

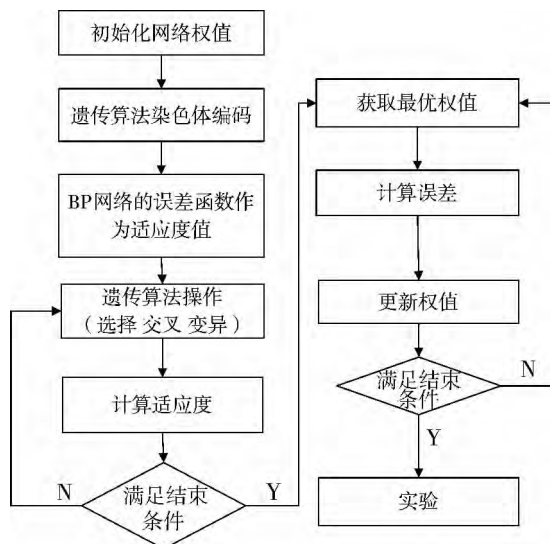


图2 遗传算法优化BP神经网络流程图

- 1) 初始化神经网络结构及遗传算法参数。
- 2) 根据适应度函数计算每个个体(神经网络权值及阈值的组合)的适应度。
- 3) 通过遗传算法中运算阶段的选择、交叉、变异操作来进行不断的迭代,并记录每次迭代后种群中每个个体的适应度。
- 4) 判断是否达到最大迭代次数。如果满足终止条件,算法终止,选择种群中适应度最好的个体,并将此个体作为最优的权值和阈值赋给神经网络;否则,继续迭代。

2 问题描述

软件是用来解决某一方面问题的工具,通常包含 3 个关键的部分,即输入、处理和输出。在进行软件错误定位时,假定 t_i 为软件的第 i 个测试用例, T 为软件输入的测试用例集合, o_i 为第 i 个测试用例的目标输出, a_i 为第 i 个测试用例的实际输出。如果实际 a_i 与目标 o_i 不符,则 t_i 为执行失败的测试用例;反之, t_i 为执行成功的测试用例。因此,软件的错误定位是通过软件的运行收集软件故障信息,再通过收集的故障信息去寻找软件故障位置。

对于软件错误定位问题的一般描述为:

- 1) 有 m 条语句的程序 P , 每条可执行的语句记为 s_j ;
- 2) T 为程序 P 的测试用例集合, 共有 n 组, r_i 为每组测试用例执行的结果, 其中有 k 组成功的测试用例, 那么, 剩下的 $n-k$ 组为失败的测试用例;
- 3) c 为程序 P 在测试用例集合 T 上执行时的轨迹或覆盖信息。

3 软件错误定位方法

在使用测试用例集 T 对程序进行测试时, 1 表示程序语句被覆盖到, 0 表示未被覆盖到。对于第 i 个测试用例 t_i 的执行结果来说, 0 表示第 i 个测试用例执行成功, 1 表示失败。具体详见表 1。

表 1 程序语句的覆盖情况

	s_1	s_2	s_3	s_4	s_5	s_6	r_i
t_1	1	1	1	0	0	1	0
t_2	0	0	0	0	1	0	0
t_3	1	1	0	1	1	1	0
t_4	0	0	1	0	0	1	1
t_5	1	1	1	1	0	0	1

在表 1 中, s_j 表示程序的语句, t_i 表示测试用例, 如果 (t_i, s_j) 位置上为 1, 表示语句 s_j 被测试用例 t_i 覆盖到, 0 表示未被覆盖到; r_i 表示在测试用例上的执行结果, $t_1 \sim t_3$ 的执行结果为 0 表示执行成功, t_4, t_5 为 1 表示执行失败。

在使用遗传神经网络进行软件错误定位时, 将表 1 中的语句覆盖情况作为网络的输入, r_i 作为目标输出, 通过这样一组映射关系对遗传神经网络进行训练。然后, 将虚拟测试集作为训练好的遗传神经网络的输入, 通过不断地迭代产生每条语句的可疑度值, 根据语句的可疑度值的大小进行软件错误定位的预测。

虚拟测试集是一个单位矩阵, 在每一个虚拟的测试案例中只覆盖了一条程序中的语句。如文献[10]中所描述的, 如果测试样例执行结果失败, 则这个被

覆盖的语句 s_j 很有可能错误。因此, 应该优先检测在虚拟测试集上可疑度值较高的程序语句。对于表 1 经遗传神经网络训练后, 再使用单位矩阵作为输入, 可以得到的每条语句 s_j 的可疑度, 见表 2。

表 2 语句可疑值表

语句	可疑度	语句	可疑度
s_1	0.041 6	s_4	0.357 1
s_2	0.115 7	s_5	0.003 1
s_3	0.068 2	s_6	0.039 1

根据表 2 中每条语句的可疑度按照降序的方式排序, 为 $s_4, s_2, s_3, s_1, s_6, s_5$ 。然后按照可疑度从大到小逐一一对程序中的语句进行检测。

在进行软件错误定位时, 具体步骤如下:

- 1) 建立遗传神经网络。初始化遗传神经网络, 遗传神经网络的输入层神经元个数需要根据测试程序的语句个数进行设置, 为避免计算时间过长, 中间层神经元个数设置为 5, 输出层神经元个数为 1, 遗传算法的交叉概率和变异设置为 0~1 之间, 进化代数设为 20, 种群规模为 100, 遗传神经网络的训练目标为 0.001。

- 2) 使用收集的语句执行信息训练遗传神经网络, 如图 3 所示。用类似于表 1 中的数据作为遗传神经网络的输入数据, r_i 作为期望输出, 通过遗传神经网络进行不停的迭代训练, 直到达到迭代满足训练目标程序终止。

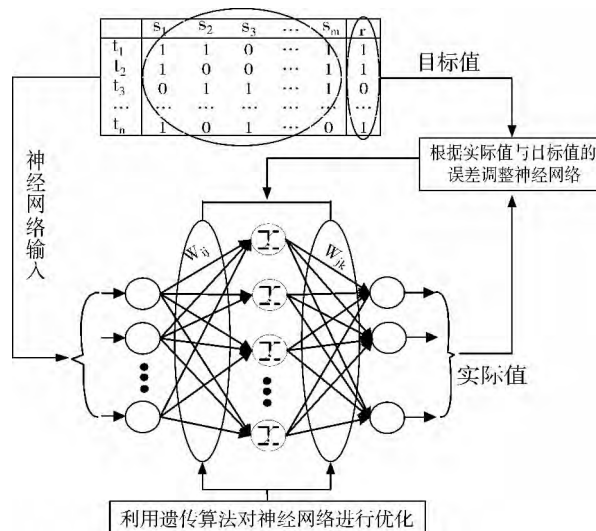


图 3 遗传神经网络训练过程

- 3) 设计一组虚拟测试用例作为训练好的遗传神经网络的输入来进行模拟仿真, 从而得到与虚拟测试用例对应的输出, 见图 4。

- 4) 对虚拟测试用例得到的结果进行降序排序。

- 5) 根据排序结果从大到小逐一进行软件错误定位。

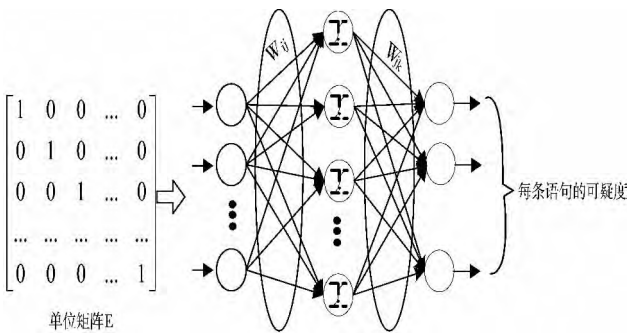


图4 用虚拟单位矩阵 E 仿真测试

4 实验分析

本实验使用遗传神经网络来对软件错误进行定位,测试程序使用的是由西门子研究院开发的 Siemens Suite,该套件是最常用的软件错误定位测试程序^[16-17]。在 Siemens Suite 中共有 7 个程序包,分别是 print_tokens 和 print_tokens2 词法分析器;replace 实现模式匹配和替换;schedule 和 schedule2 优先等级调度程序;tcas 简单的飞行轨迹冲突避免系统;tot_in-fo 信息度量程序。详细信息见表 3。

表 3 Siemens Suite

Program	Language	Description	Lines of Code	Faulty version	Test cases
print_tokens	C	Lexical analyzer	565	7	4 130
print_tokens2	C	Lexical analyzer	510	10	4 115
replace	C	Pattern replacement	563	32	5 542
schedule	C	Priority scheduler	412	9	2 650
schedule2	C	Priority scheduler	307	10	2 710
tcas	C	Altitude separation	173	41	1 608
tot_info	C	Information measure	406	23	1 052

实验硬件平台采用的是 Intel Core i7 CPU 2.13 GHz 8 G 内存,操作系统为 Ubuntu 12,程序的编译及程序运行时数据的收集采用的是 GCC4.4.6 及其组件 Gcov。使用 Matlab R2011a 构建遗传神经网络及对数据进行处理。

为了更好地与其他的软件错误定位方法进行比较,本文采用的程序语句的评价标准与文献[5]中的方法一致,并给出采用同样评价标准的 Tarantula、NN/perm、NN/binary、CT、CT/relevant、CT/infected、Intersection、Union 等技术的实验结果。GABP 与其他技术比较的结果见表 4。

表 4 在每个评分阶段中测试版本所占的比例

Score/%	GABP	Tarantula	NN/perm	NN/binary	CT	CT/relevant	CT/infected	Intersection	Union
99-100	19.87	13.93	0	0	4.65	5.43	4.55	0	1.83
90-99	29.51	41.8	16.51	4.59	21.71	30.23	26.36	0.92	3.67
80-90	11.54	5.74	9.17	8.26	11.63	6.2	10.91	0	0.92
70-80	10.71	9.84	11.93	4.59	13.18	6.2	13.64	0	0.92
60-70	9.72	8.2	13.76	3.67	1.55	9.3	4.55	0	0
50-60	6.82	7.38	19.27	7.33	6.98	10.08	6.36	0	0
40-50	3.46	0.82	3.67	9.17	3.1	3.88	1.82	0	0
30-40	2.32	0.82	6.42	13.76	7.75	10.08	3.64	0	0
20-30	3.8	4.1	1.83	13.76	4.65	3.1	7.27	0	0
10-20	1.21	7.38	0	6.42	6.98	10.85	0	0	0
0-10	1.03	0	17.43	28.44	17.83	4.65	20.91	99.08	92.66

图 5 将表 4 中各种技术的信息表达出来。在图 5 中横坐标表示评价标准,表示程序中有多少比例的语句不需要被审查即可定位到错误,以 10 个百分点作为等级划分点,99% ~ 100% 这个级别除外,因为 100% 的评分是不可能实现的,不可能任何一个语句都不需要测试就能定位到程序的错误。评分 99% 已经表示该方法几乎准确地命中该程序的错误,因此,将 99% ~ 100% 这个级别单独列出来表明有多少比例的版本可以准确命中程序的错误。纵坐标表示有多少比例的被测试版本的评分值是在该横坐标所给定的评分等级之内。从图中可以得知,Tarantula 方法

在每一个评分区间上都要比 NN/perm、NN/binary、CT、CT/relevant、CT/infected、Intersection、Union 等方法要好。例如,Tarantula 方法,横坐标为 90% ~ 99%,纵坐标为 55.73%,表示对于比例为 55.73% 的测试版本,通过搜索少于 10% 的语句就可以定位到软件错误,而本文采用的 GABP 方法在此区间上如要达到同样的效果,测试版本的比例为 49.38%,由此可知,在 90% ~ 99% 区间上,GABP 方法不如 Tarantula 方法,而在其它区间上基本接近于或优于 Tarantula 方法。

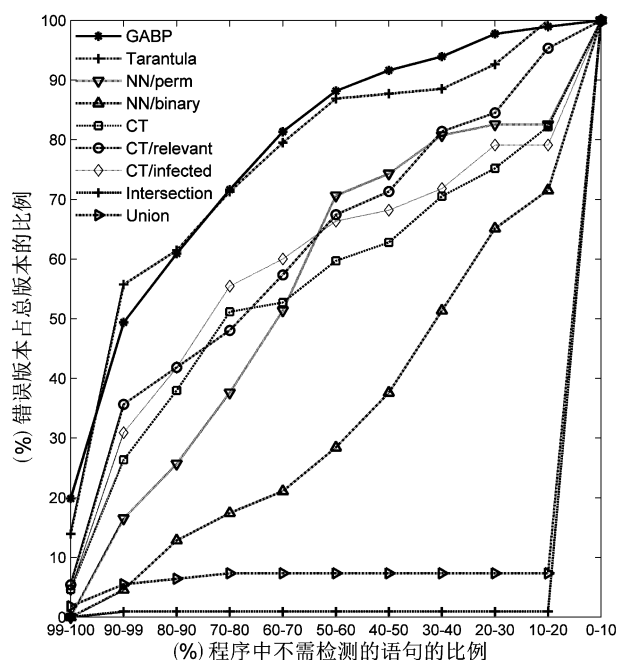


图5 各种软件错误定位技术的效果图

5 结束语

本文研究了利用机器学习的方法来进行软件错误定位的技术。由于传统的 BP 神经网络自身存在收敛速度慢和易于陷入极小值的缺陷, 本文利用遗传算法来对 BP 神经网络进行优化, 并将遗传神经网络技术应用于软件的错误定位。实验表明, 本方法在一定程度上提高了软件错误定位的效率, 与其它方法相比具有较好的性能, 能够对软件调试工作起到较大的帮助作用。

参考文献:

- [1] 虞凯, 林梦香. 自动化软件错误定位技术研究进展[J]. 计算机学报, 2011, 34(8): 1411-1422.
- [2] 谭德贵, 陈林, 王子元, 等. 通过增大边际权重提高基于频谱的错误定位效率[J]. 计算机学报, 2010, 33(12): 2335-2342.
- [3] 陈浩, 刘海涛, 杨根庆. 用遗传算法进行软件的快速故障定位[J]. 微电子与计算机, 2000(5): 35-37.
- [4] James A Jones, Mary Jean Harrold, John Stasko. Visualization of test information to assist fault localization[C]// Proceedings of the 24th International Conference on Software Engineering. 2002: 467-477.
- [5] Jones J A, Harrold M J. Empirical evaluation of the tarantula automatic fault-localization technique[C]// Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. 2005: 273-282.
- [6] Wong W E, Debroy V, Golden R, et al. Effective software fault localization using an RBF neural network[J]. IEEE Transactions on Reliability, 2012, 61(1): 149-169.
- [7] 秦兴生, 胡觉亮, 丁佐华. 基于神经网络集成的软件故障预测及实验分析[J]. 计算机工程与应用, 2014, 50(10): 44-47.
- [8] 张柯, 张德平, 汪帅. 基于增强径向函数神经网络的错误定位方法[J]. 计算机应用研究, 2015, 32(3): 781-785.
- [9] 曹鹤玲, 姜淑娟, 鞠小林. 软件错误定位研究综述[J]. 计算机科学, 2014, 41(2): 1-6.
- [10] Ericwong W, Yu Qi. BP Neural network-based effective fault localization[J]. International Journal of Software Engineering and Knowledge Engineering, 2009, 19(4): 573-597.
- [11] 吉根林. 遗传算法研究综述[J]. 计算机应用与软件, 2004, 21(2): 69-73.
- [12] Neumann D E. An enhanced neural network technique for software risk analysis[J]. IEEE Transactions on Software Engineering, 2002, 28(9): 904-912.
- [13] 吴鸣锐, 张钹. 一种用于大规模模式识别问题的神经网络算法[J]. 软件学报, 2001, 12(6): 851-855.
- [14] Su Yu-shen, Huang Chin-yu. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models[J]. Journal of Systems and Software, 2007, 80(4): 606-615.
- [15] Ito Y. Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory[J]. Neural Networks, 1991, 4(3): 385-394.
- [16] Aristotle Analysis System. Siemens Programs [DB/OL]. <http://pleuma.cc.gatech.edu/aristotle/Tools/subjects/>, 2015-02-04.
- [17] Wong W E, Debroy V, Li Yihao, et al. Software fault localization using DStar(D*) [C]// 2012 IEEE 6th International Conference on Software Security and Reliability. 2012: 21-30.