

文章编号:1001-9081(2010)01-0153-03

BP 人工神经网络在 bug 分析中的应用

王 雷, 杨小虎

(浙江大学 计算机科学与技术学院, 杭州 310015)

(wingwanglei@hotmail.com)

摘 要:应用于金融领域的软件系统,由于其包含复杂的商业逻辑导致此类系统不但庞大而且逻辑复杂。在此类系统的开发和升级过程中,系统缺陷及错误的寻找、分析常常非常困难且费时,在通常情况下,它往往成为整个项目中后期的瓶颈。运用 BP 人工神经网络的算法,设计并实现了针对某银行网上交易系统的缺陷及错误分析系统,并且通过实验证实该系统能帮助开发人员提高寻找、分析系统缺陷及错误的效率,进而加快整个项目的进度。

关键词:BP 人工神经网络; 系统升级; bug 分析; 金融软件系统

中图分类号: TP301.6 **文献标志码:** A

Application of back propagation artificial neural networks in bug analysis

WANG Lei, YANG Xiao-hu

(College of Computer Science and Technology, Zhejiang University, Hangzhou Zhejiang 310015, China)

Abstract: Due to the complex business requirements, financial software system could be giant in scope and complex in system logic. During the development and maintenance of such software system, the detection and analysis of the system defects and errors are difficult and time-consuming in most of the circumstances, which usually is deemed as the bottleneck of the whole project after the mid-stage. In this paper, by implementing the Back Propagation (BP) algorithm, a defect analysis system was designed for the enhancement project of an Internet-based transaction trading system. The experimental results show that it does improve the developers' efficiency and productivity in defect detecting and fixing, which consequentially accelerates the project progress.

Key words: Back Propagation Artificial Neural Network (BPANN); system upgrade; bug analysis; financial software system

0 引言

网上金融交易系统的出现和普及极大地简化了投资人和投资信托机构的工作。现在买卖双方完全可以足不出户,仅仅通过点击鼠标来完成交易,这种方式不但更加方便、快捷而且更加安全。金融交易系统不但要实现各种类型的交易中所包含的每一条商业逻辑,还要保证系统具有高度的精确性、足够的稳定性、出色的强壮性和可靠的安全性。

当这样的系统需要升级或者重构时,面临的困难也是非常大的。从需求分析到代码重构都包含了巨大的挑战,但是最大的困难还是代码编写完成后的测试和 bug 修正。测试工作虽然有很大的难度,但是可以通过过饱和的平行测试来解决:将整个系统视为一个黑盒,使用自动化测试工具导入海量的测试数据测试重构后的新系统,然后对新、旧系统运行的结果做比较。

但在目前,还没有成熟的软件或者工具可以辅助程序员寻找、分析 bug;而且对于 bug 处理的研究更多的是基于如何从设计上剔除 bug^[1]。因此,开发人员只能通过分析代码、分析数据、打印日志消息或者 Debug 等几种传统方法来寻找出 bug,然后分析并且修正它们。使用这样纯人工方法定位 bug 产生位置在小规模的系统中并无太大缺陷,但是这种方法在一个庞大的系统中就往往力不从心了。

1 bug 定位方法

1.1 人工 bug 定位方法

人工 bug 定位方法是最为普遍使用的方法,但缺陷很多。以某银行的网上外汇交易系统的系统重构工程为例,程序员在以纯人工方式寻找和分析 bug 过程主要会遇到以下的困难:

1) 金融应用类系统对系统的强壮性要求很高,系统不会轻易终止对任何一条数据的处理。因此,如果不能定位 bug 产生的位置,那么开发人员要分析的代码将会是整个系统!

2) 系统经过重构以后,即使最稳定的存储层变动也是巨大的。而且,仅仅靠开发人员肉眼寻找、比较这些数据几乎不可能分析出 bug 产生的原因,因此程序员通常只能通过分析代码寻找 bug。

3) 由于金融应用的系统包含了丰富的商业逻辑,因此很多逻辑过程复杂而且难以理解。复杂的商业逻辑无疑进一步加大了开发人员寻找和分析 bug 的难度。

4) 通常平行测试只给出新系统和旧系统最终输出结果的比较。因此,分析 bug 时只能采用回溯的方法。

所以开发人员只能采用从下游模块到上游模块的“接力”分析模式人工寻找 bug。这样的 bug 分析模式有很大的缺陷:

1) 下游的模块的开发人员将承担更加繁重的任务,因为

收稿日期:2009-06-30;修回日期:2009-08-17。

作者简介:王雷(1982-),男,浙江金华人,硕士,主要研究方向:计算机软件应用; 杨小虎(1966-),男,浙江绍兴人,副教授,博士,主要研究方向:软件工程。

每次的 bug 分析工作都由他们开始;

2) 大量的任务被积压在下游模块;

3) 人力资源浪费严重, 因为很多情况下, 下游模块的开发人员在重复的而且完全可以避免的工作。

显而易见, 单纯依靠程序员人工分析代码的 bug 定位方法不但效率低、工作分配不合理还浪费了大量人力资源。

1.2 使用软件定位 bug 方法

目前出现了一些用于分析和寻找 bug 的软件, 例如针对 Java 程序的 FindBugs 等。但是此类软件的实质上只是对程序静态扫描、分析。因此, 这类软件对只有在动态运行中才能体现出来的 bug 束手无策。而其他的诸如 BugFree 类的软件实现的是 bug 的管理而非分析。

因此, 目前并没有一种现成的工具软件能有效地帮助开发人员在本文举例的项目过程中有效地分析和定位 bug。

1.3 BP 神经网络 bug 定位方法

首先, 金融应用系统对数据完整性有很高的要求。因为即使系统运行过程中遭遇严重错误导致崩溃, 该系统也要能将系统正确地恢复到事故前的状态。因此系统在设计之初就要求在各模块的数据库表中存储足够的数据以恢复任何一条记录。所以, 每个模块的数据库表中有足够的数据能确定任何一条交易记录在该模块的运行状态。因此, 通过对数据库表数据的分析来判断某条交易数据在该模块执行后的状态是否是正确是完全可行的。

其次, 系统升级必须保证升级以后的新系统对旧系统的数据完全支持。因此无论新系统和旧系统在系统架构、实现技术、实现方法上有多大的差异, 在存储层上的差异还是相对小的。在本文描述的重构项目中, 旧系统中有 95% 以上的数据库的列元素能在新系统的数据库表中找到对应的列。

综上两点, 通过对照分析新系统和旧系统中任何模块的数据库表元素的值, 来预测该模块是否正确运行是完全可行的。因为每个模块的数据库表的列元素值和整个模块最终运行结果之间存在的联系实际上就是我们在程序中实现的商业逻辑。

BP 神经网络擅长的是处理那些规律隐藏在一大堆数据中的映射逼近问题, 特别是需要通过学习自适应可调的问题^[2]。因此, 本文使用 BP 人工神经网络的算法, 针对该网上外汇交易系统系统升级这一项目中寻找 bug 困难的问题, 设计了一个 bug 分析系统。

2 基于 BP 神经网络的 bug 分析系统

2.1 系统设计

bug 分析系统设计如图 1 所示。

2.2 输入数据的处理

本文中作为分析对象的网上外汇交易系统的数据库表里包含的数据类型有字符型、字符串型、布尔类型还有整数和浮点小数。必须对这些不同类型的数据做预处理才能得到适合 BP 神经网络处理的输入数据。此外, 更大的难点在于, 新、旧两个系统中数据库表并非完全对应。根据数据来源对应关系的不同, 做以下 3 类预处理操作:

1) 数据来源的数据表列在新系统和旧系统的数据库表中有直接的对对应关系。这类数据的预处理只需要把取自 2 个系统的值进行简单的比较, 如果比较结果为“匹配”则赋值“1”, 反之赋值“0”。

2) 数据来源在新系统和旧系统的数据库表中为 1:n 或

者是 n:1 的对应关系(例如旧系统中把交易额表示为“货币类型+交易数目”存为一列, 而新系统中把货币类型和数额分为两列存储)。这类数据的处理方法是把包含信息多的数据列拆开, 再一一对应, 然后按照 1) 处理。

3) 数据来源在新系统和旧系统数据库表中为 m:n 的关系。对这类数据源, 处理的方法是先把相关数据值拼接, 再按照最小粒度分割, 把分割以后的单元一一对应按照 1) 处理。

经过以上处理后得到的就是适合 BP 神经网络处理的“0”和“1”串了。但是在实际应用中, 0 和 1 都是使用 sigmoid 单元的人工神经网络所无法达到的权值^[3], 因此本文在实际实现过程中用 0.1 代替 0; 用 0.9 代替 1。

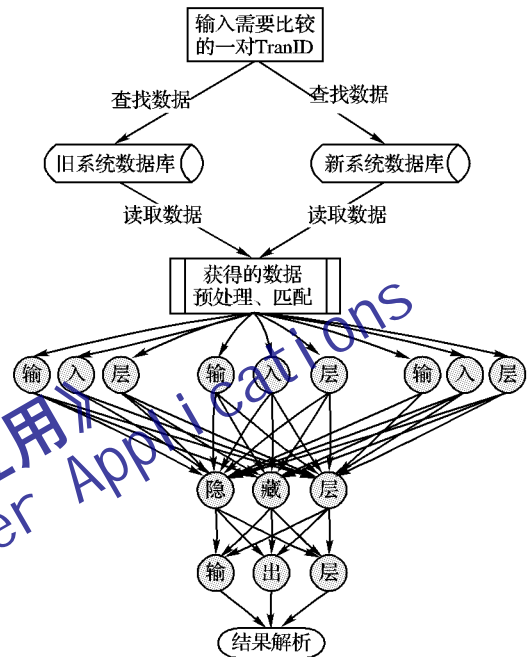


图1 bug 分析系统设计

2.3 神经网络的设计

已有理论研究证明: 只要有一个隐含层的三层 BP 神经网络, 就可以逼近任何映射函数以完成给定的映射任务^[4]。在三层 BP 网络的基础上再增加一层隐含层, 虽然能进一步提高逼近的精度, 但是会导致网络复杂度激增和训练时间的大量增加。所以本文选择 3 层的 BP 神经网络。

输入层单元的数目由需要处理的数据元素数量决定。在本文中, 输入层单元的数量等于经过预处理后得到的单元数。

隐藏层节点的作用是从样本中提取并存储其内在规律。因此, 隐藏层单元数目对网络性能至关重要, 如果隐藏层单元太少, 则无法产生足够的连接权组合数来满足样本对学习的要求; 如果隐藏层单元过多, 则学习以后网络泛化能力变差^[2]。隐藏层节点数 m 的确定方法, 有以下 3 个经验公式^[5]:

$$k < \sum_{i=1}^n C_i^m \quad (1)$$

其中: K 为样本数, n 为输入层单元数。当 $i > m$ 时, $C_i^m = 0$ 。

$$m = \sqrt{n + o + a} \quad (2)$$

其中: o 为输出层单元数, a 是一个 $[1, 10]$ 内的常数。

$$m = 1.5n \quad (3)$$

为了使该网络有足够的隐藏层节点来产生连接权组合数来满足样本对学习的要求, 又不会因为隐藏层节点数目过多而影响网络的泛化能力, 本文设计的 bug 分析系统输入层单元数量大约为 340。在本例中选取式 (2) 来确定隐藏层节点

数目,本文取隐藏层单元数目为60。

输出层单元的数目由输出信息的表达需要所决定。在本文的例子中,作为分析对象的网上外汇交易系统按照功能划分,由上游到下游可以分为如下5个功能模块:处理输入信息的 Inbound 模块;验证交易信息完整性的 Trade Capture 模块;处理具体商业逻辑的 Trade Processing 模块;做交易结算的 Settlement 模块;往下游系统发送消息的 Outbound 模块。

本文实现的 bug 分析系统最主要的目的是把产生 bug 的代码定位到某个模块(如果有超过一个模块有 bug,则报告最先产生 bug 的模块)。因此3个输出层单元就能表示所有7种情况了,具体对应关系如表1所示。

表1 输出层单元信息真值表

编码值	代表状态
111	执行正确,无 bug
110	Inbound 产生 bug
101	Trade Capture 产生 bug
100	Trade Processing 产生 bug
011	Settlement 产生 bug
010	Outbound 产生 bug
001	N/A
000	输入信息缺失

最后,在每个输入层单元、隐藏层单元之间建立连接;每个隐藏层单元、输出层单元之间也建立连接完成该 BP 神经网络的构建。

2.4 算法和关键参数的选择

本文选取随机梯度下降的包含双层 Sigmoid 单元的 BP 网络反向传播算法,算法简单描述如下^[3]:

建立并初始化网络,为了加快训练速度,把网络权值初始化为 $(-2.4/N, 2.4/N)$,其中 N 为该层单元数量^[6]。

在遇到终止条件前做以下训练:

对于训练样例 $\langle X, T \rangle$,把输入按照 Sigmoid 函数映射关系沿网络前向传递:

1) 把实例 X 输入网络,并计算网络中每个输出单元的输

出值 O_k ;使误差沿网络反向传播:

2) 对于网络每个输出层单元 k ,计算它的误差项 δ_k :

$$\delta_k = o_k(1 - o_k)(t_k - o_k) \quad (4)$$

3) 对于网络的每个隐藏层单元 h ,计算它的误差项 δ_h :

$$\delta_h = o_h(2 - o_h) \sum_{k \in \text{outputs}} w_{kh} \delta_k \quad (5)$$

4) 更新每个网络权值 w_{ij} :

$$w_{ij} = w_{ij} + \Delta w_{ij} \quad (6)$$

为了防止网络在训练时收敛于局部最小值,并且加快训练时的收敛速度,按如下公式加入冲量项^[3]:

$$\Delta w_{ij}(n) = \eta_i \delta_i x_{ij} + \alpha \Delta w_{ij}(n-1) \quad (7)$$

式中, η_i 为学习步长,是该算法中非常重要的一个参数。学习步长太小会导致训练时收敛速度太慢;学习步长过大则会导致学习时预测值发生振荡。因此本文采用变步长算法^[7]:

$$\begin{cases} \eta = \eta \times 1.05, & E(t) < E(t+1) \\ \eta = \eta \times 0.07, & E(t) > 1.04 \times E(t+1) \\ \eta = \eta, & \text{其他} \end{cases} \quad (8)$$

使网络在训练中能尽量比较平滑而迅速地收敛。

3 实验结果

有理论研究表明:神经网络中连接总数和训练该网络所需要的样本数有如下的近似关系:

$$N = W/\varepsilon \quad (9)$$

其中 ε 为接近10的常数^[5]。因此,本文设计的网络大约需要1800个训练样本才能达到稳定。

本文描述的该外汇交易系统的重构项目在过去的2个半月,Inbound 模块修正76个 bug;Trade Capture 模块修正47个 bug;Trade Processing 模块修正317个 bug;Settlement 模块修正114个 bug;Outbound 模块修正211个 bug,总计765个 bug。

本文针对每个 bug 取3组数据,得到2300个样本。再根据训练样本:测试样本为7:3的比值从各个模块的样本中按这一比例选取1600个训练样本和700个测试样本。使用这1600个训练样本完成对该网络的训练。此后再从训练样本集中选取700个训练样本,用这些训练样本和700个测试样本分别对该网络进行测试。测试结果如表2所示。

表2 bug 分析系统测试结果 %

bug 所在模块	训练样本准确率	测试样本准确率
Inbound	92.6	86.8
Trade Capture	97.6	92.9
Trade Processing	90.5	81.8
Settlement	90.3	84.5
Outbound	94.2	87.9

实验结果可以看出,该 bug 分析系统对训练样本的识别率均在90%以上,对测试样本的识别率也均在80%以上。若将各个模块发现 bug 数量定义为权值,则该系统整体的 bug 定位准确率为:对训练样本为92.2%,对测试样本为85.0%。因此,完全可以认为本文提出的 bug 分析系统对该网上外汇交易系统重构项目中的 bug 有很高的识别率,从而在 bug 修正过程中能帮助程序员节省大量寻找 bug 的时间。

4 结语

本文针对某银行的网上外汇交易系统系统重构这一项目过程中,开发人员在项目中后期寻找和分析 bug 的过程中遇到的困难,提出了一种基于 BP 人工神经网络的 bug 定位方法。通过实验证明,使用该方法能有效地定位 bug 产生的模块,从而大大节省开发人员花费在寻找 bug 上的时间。

参考文献:

- [1] BROOKS F P. The mythical man-month[M]. 汪颖,译.北京:清华大学出版社,2007.
- [2] 张立明. 人工神经网络的模型及其应用[M]. 上海:复旦大学出版社,1993.
- [3] MITCHELL T M. Machine Learning[M]. 曾华军,张银奎,译.北京:机械工业出版社,2003.
- [4] 靳蕃. 神经计算智能基础原理·方法[M]. 成都:西南交通大学出版社,2000.
- [5] 张立明. 多层神经网络泛化特性分析[C]//1997年中国神经计算科学大会论文集. 北京:人民邮电出版社,1997:36-39.
- [6] 罗四维. 大规模人工神经网络基础[M]. 北京:清华大学出版社,2004.
- [7] 胡伍生. 神经网络理论及其工程应用[M]. 北京:测绘出版社,2006:63-72.
- [8] HOPFIELD J. Artificial neural networks[J]. IEEE Circuits and Devices Magazine, 1998, 4(5): 3-10.
- [9] VELASCO T, ROWE M R. Back propagation artificial neural network for the analysis of quality control charts[J]. Computers and Industrial Engineering, 1993, 25(1/4): 397-400.