偏相关方法在软件缺陷预测中的应用 *

马 樱, 罗光春, 李 炯, 陈爱国
(电子科技大学 计算机科学与工程学院, 成都 611731)

摘 要: 为了提高预测模型的性能,解决不同属性子集带来的分歧,提出了基本偏相关方法的预测模型。首先,该方法在公开数据集上分析出代码静态属性与缺陷数之间存在偏相关关系;然后基于偏相关系数值,计算出代码复杂性度密度属性值;最后基于该属性值建立新的缺陷预测模型。实验表明,该模型具有较高的召回率和很好的 F-measure 性能,从而进一步证实了代码属性与模块缺陷之间的偏相关性是影响软件质量预测性能的重要因素的结论。该结论有助于建立更加稳定可靠的软件缺陷预测模型。

关键词: 软件缺陷预测; 代码静态属性; 实证; 复杂度; 机器学习; 偏相关
中图分类号: TP301.5 **文献标志码**: A **文章编号**: 1001-3695(2012)02-0594-03
doi:10.3969/j.issn.1001-3695.2012.02.051

Partial correlation analysis for software defect prediction

MA Ying, LUO Guang-chun, LI Jiong, CHEN Ai-guo
(*School of Computer Science & Engineering, University of Electronics Science & Technology of China, Chengdu* 611731, *China*)

Abstract: In order to improve the performance of predictors, and reduce the dissention, this paper propsed a new predict model based on partial correlation analysis. Firstly, different to prior works, analyzed the correlation between attributes and defects. Then computed code complexity density values. Based on these values, built a new predictor. Experiments were performed on the public Eclipse dataset. This predictor had a good performance with high recall rates and substantially high F-measure values. The satisfactory results also confirm the partial correlation is a very important factor in software quality analysis. This conclusion is helpful for building more stable defect predictors.

Key words: software defect prediction; static code attributes; empirical; complexity; machine learning; partial correlation

缺陷预测的目的是对模块给出测试的优先次序,从而避免全面的测试。缺陷预测模型可以帮助项目管理者有效地分配测试资源。如果提供了模块相关的软件特征(度量属性),预测模型就可以预测这些模块的缺陷趋向程度。许多研究人员使用静态属性来进行软件质量预测,并提出了许多以此为基础的质量预测模型[13]。但是同时出现了一些问题,如基于复杂度度量与代码行缺陷预测之间优点的争论。不同研究可以得出类似于这样的结论,相对 McCabes 复杂性属性,行代码是一个更好的或更糟糕的预测。

1 相关工作及本文贡献

研究者基于商业软件的实验数据得出支持面向对象设计的复杂性度量,尤其是 CK 度量属性,在确定软件缺陷中起到了决定作用[4]。而文献[5]通过四种评估模型则发现,类耦合度量(CBO)似乎是最好的模块缺陷预测属性。代码行度量(LOC)性能较好,而且由于它可以很容易地计算,因此它似乎是适合快速故障预报。但是对于细粒度分析,多变量模型的性能更好,而继承深度(DIT)与子类个数(NOC)度量属性基本不能应用在缺陷预测中。文献[6]在美国宇航局数据集上进行了广泛的研究,他们使用 25 种分类技术和包括代码行的 21 个代码度量属性,得出的预测结果并不令人满意。文献[7]表明对缺陷预测的最佳属性集在不同的数据集上会发生巨大的变化,得出最佳的属性子集很脆弱,即当变化数据集时该模型可能不再适用。

而另一些研究者则认为代码行属性就能达到很好的预测性能。文献[8]表明,McCabe 提供的静态属性,只是比代码行提供更多的无意义的属性。文献[9]对一个大型通信系统的连续两个版本分析了预发布缺陷数和发布后缺陷数,评估了缺陷数和代码行相关的基本假设范围。文献[10]则重复了文献[9]的研究,其结果证实了缺陷分布的 Pareto 原则,但并未最终支持代码行的缺陷预测能力。而文献[11]更进一步在 Eclipse 数据集上利用统计相关系数证明了代码行与缺陷数的相关性,并进一步使用威布尔函数形式化描述了模块缺陷分布,得出简单的代码静态属性代码行是有用的软件质量指标。但是该论文没有考虑到其他代码属性的影响,如代码复杂度属性。

本文在研究模块代码属性与模块缺陷之间关系的基础上,着重研究代码静态属性与模块缺陷之间的关系问题。分析了 Eclipse 三个版本(2.0、2.1 和 3.0)的缺陷数据集,包括预发布和发布后缺陷数据。本文的结论如下:

a)通过假设检验,实验得出的相关性统计量表明代码属

收稿日期: 2011-07-15; **修回日期**: 2011-08-22 **基金项目**: 新世纪优秀人才支持计划资助项目(NCET-10-0298);四川省科技支撑计划资助项目(2011GZ0192);中央高校基本科研业务费专项资助项目(ZYGX2009J066)

作者简介: 马樱(1982-),男,博士研究生,主要研究方向为数据挖掘、软件工程(maying 9012@ sohu.com);罗光春(1974-),男,教授,博导,主要研究方向为软件中间件技术、计算机网络;李炯(1979-),男,博士研究生,主要研究方向为软件工程;陈爱国(1981-),男,博士,主要研究方向为软件分析.

性与缺陷数量之间存在正相关。大规模、复杂度高的模块往往具有更多的缺陷。不同于文献[11]只对代码行属性的考虑,本文在 Eclipse 数据集上验证了很大一部分的代码静态属性与软件模块缺陷数是相关的。同时在 Eclipse 数据集上,计算属性与缺陷数量的相关性系数,支持了文献[7]的"最佳"数据集属性个数大于 1 的结论。

b)从统计学意义上证明了预测属性本身之间的关系问题。在一定的限制上减少代码属性是可能的,比如软件代码的一个属性的预测效果可能是假象。实验获取的偏相关性系数值表明,代码行数与模块缺陷数的相关性是由于块嵌套深度复杂性属性协同作用的结果。

c)基于属性偏相关特性的融合属性建立的预测模型预测效果不错。本文融合了代码复杂度属性与代码行数属性,计算出复杂密度属性,在此基础上应用五种机器学习的算法预测了模块的缺陷趋向性,得出了令人满意的效果。例如,对发布前的 Eclipse 3.0 缺陷预测,得到的召回率超过 95% 和精确度超过 64%,F-measure 值大于 76%。同时该预测性能也证明了代码属性之间是存在偏相关关系的。

2　代码属性的偏相关分析

2.1　偏相关系数

偏相关系数[12]是在多元回归分析中,在消除其他变量影响的条件下,所计算的某两变量之间的相关系数。对任意变量 X_i、X_j、X_k,如果考虑以 X_k 为控制变量,X_i 与 X_j 的一级偏相关系数计算公式为

$$r_{ij,k}=\frac{r_{ij}-r_{ik}r_{jk}}{\sqrt{(1-r_{ik}^2)(1-r_{jk}^2)}} \tag{1}$$

其中:r_{ij}、r_{ik}、r_{jk}分别表示 X_i、X_j、X_j 之间的单相关系数。本文采用 t 检验法进行显著性检验。

2.2　偏相关实验分析

本研究中使用由 Saarland 大学收集的 Eclipse 公共数据。Eclipse 是一种广泛使用的集成开发平台,用于创建 Java、C++ 和 Web 应用程序。公开的 Eclipse 度量和缺陷数据集包含 Eclipse 的版本 2.0、2.1 和 3.0(表 1)。本研究的 Eclipse 的系统是相当大的软件系统,它们平均包含 1 030 k 代码行,8 403 个类,491 个包。该缺陷数据是从 Eclipse 的缺陷数据库和版本库挖掘获得的。它有两种缺陷,即预发布缺陷(最后发布前 6 个月的缺陷报告)和发布后缺陷(发布后 6 个月的缺陷报告)。更多有关数据收集过程中的细节可参见文献[13]。

表 1　Eclipse 缺陷数据集

版本	文件数	缺陷数(文件级)	
		发布前	发布后
2.0	6 729	7 635	1 692
2.1	7 888	4 975	1 182
3.0	10 593	7 422	2 679

文献[5]指出在不损害预测性能的情况下,大量减少属性数量是可能的。使用 2 或 3 个属性与使用 38 个属性工作性能一样,但是只使用 1 个属性将导致性能很大程度上地下降。该文没有给出合理的解释,在此讨论了不同属性与缺陷数量之间的关系,对文献[7]的结果给出了相关的解释。表 2 为代码行数(TLOC_sum)与缺陷数的块嵌套深度(NBD_sum)偏相关系数。通过表 2 可以看出,当除去块嵌套深度 NBD_sum 的影响后,代码行与缺陷数量的相关性不明显;当存在块嵌套深度的影响时,代码行与缺陷数之间的相关度是显著的,表明只使用 1 个属性将导致性能很大程度上地下降。

表 2　代码行数(TLOC_sum)与缺陷数的块嵌套深度(NBD_sum)偏相关系数

版本	控制变量	统计	发布前	发布后
2.0	NBD_sum	Correlation	-0.046	-0.093
		Sig.(2-tailed)	0.373	0.073
	NULL	Correlation	.585(*)	.564(*)
		Sig.(2-tailed)	0.000	0.000
2.1	NBD_sum	Correlation	.075	-.007
		Sig.(2-tailed)	.121	.891
	NULL	Correlation	.575(*)	.471(*)
		Sig.(2-tailed)	0.000	0.000
3.0	NBD_sum	Correlation	0.044	0.139
		Sig.(2-tailed)	.264	.000
	NULL	Correlation	0.581(*)	0.559(*)
		Sig.(2-tailed)	0.000	0.000

注:Correlations significant at the 0.01 level marked with(*)

3　基于偏相关的属性融合预测模型

3.1　基于偏相关的属性融合

在缺陷预测模型 $L=\{(x_1,c_1),(x_2,c_2),\cdots,(x_n,c_n)\}$ 表示训练集,$c_i\in(\text{false},\text{true})$,其中 false 标志无缺陷模块,true 标志有缺陷模块。模块 $x_i=\{a_{i1},a_{i2},\cdots,a_{ik}\}$,其中 a_{ij} 为第 i 个模块的第 j 个属性。本文的预测模型就是从给定的函数 $f(x,\alpha)$,α 为该函数的参数,选择出能够最好地逼近训练集标记的函数,即使得风险函数[14] $R(\alpha)$ 最小。

$$R(\alpha)=\int L(y,f(x,\alpha))\,\mathrm{d}F(x,y) \tag{2}$$

其中:

$$L(y,f(x,\alpha))=\begin{cases}0,\text{若 } y=f(x,\alpha)\\1,\text{若 } y\neq f(x,\alpha)\end{cases} \tag{3}$$

基于偏相关系数,定义基于偏相关性的融合属性 $\rho_{ij}=\frac{\ln a_{ij}}{\ln a_{id}}$,$d\neq j$。假设所有的属性都存在偏相关,该融合属性将有 C_n^2 个。不失一般性,针对代码行数,定义复杂密度属性 ρ_{density} 为

$$\rho_{\text{density}}=\frac{\ln a_{ij}}{\ln \text{TLOC\_sum}} \tag{4}$$

其中:TLOC_sum 为代码行数。当出现 ln 0 时,取 ln 0.000001。本文以复杂密度作为缺陷预测模型的属性值,该属性结合了代码行和复杂度度量属性。

3.2　性能评估

本文选择了五个常用的分类技术,如表 3 所示,以便于其他学者做对比研究。本文使用了 10 倍交叉验证评估分类模型,即整个训练数据随机地划分为 10 部分,每个部分轮流进行数据测试,而由其余 9 部分训练分类模型。

表 3　缺陷预测使用的分类器

分类器	描述
NaiveBaye	基于标准概率论的贝叶斯模型
BayesNet	基于贝叶斯网络的分类器
Kstar	基于实例学习与熵度量的分类器
J48	决策树分类器
DecisionTable	基于决策规则的分类器

为了评估预测模型,本文使用召回率(recall)、精确率(precision)、F-measure 和准确率(Acc)。这些指标是软件缺陷

领域广泛使用的精确性度量指标[1517]。

缺陷预测结果示例如表 4 所示,它们被定义为:A 表示本来有缺陷的模块被预测为有缺陷的模块数;B 表示本来有缺陷的模块被预测为无缺陷的模块数;C 表示本来无缺陷的模块被预测为有缺陷的模块数;D 表示本来无缺陷的模块被预测为无缺陷的模块数。

表 4　预测模型的结果

分类	预测为有缺陷	预测为无缺陷
真实有缺陷	A	B
真实无缺陷	C	D

召回率是正确被预测为有缺陷模块数与真实缺陷模块数的比值,被定义为

$$\text{recall} = \frac{A}{A+B} \tag{5}$$

精确率是正确被预测为有缺陷模块数与预测为缺陷模块数的比值,被定义为

$$\text{precision} = \frac{A}{A+C} \tag{6}$$

一个很好的预测模型应达到较高的召回率和高精确度。然而高召回率的获得往往以低精确度为代价的,反之亦然。因此,F-measure 常常用于将召回率与精确度结合起来评价。它定义为精确度与召回率的调和平均:

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{7}$$

准确率是正确预测的模块数与总模块数之比,被定义为

$$\text{Acc} = \frac{A+C}{A+B+C+D} \tag{8}$$

其中:recall、precision、F-measure 和 Acc 值在 01,数值越高代表模型在该方面的性能越好。

3.3　实验结果

表 5、6 表明了使用 Eclipse 3.0 的复杂密度构造缺陷预测模型的 10 倍交叉验证结果。对于预发布有缺陷的模块,所有分类技术获得良好的一致的 95% 以上的召回率,64% 左右的精确度,F-measure 值大于 76%,除 Kstar 算法的 Acc 值接近 60% 外,Acc 值大于 63%。这些结果可认为是令人满意的。

表 5　基于 $\rho_{density}$ 的发布前缺陷预测结果

分类器	recall	precision	F-measure	Acc
NaiveBaye	0.976	0.647	0.778	0.672
BayesNet	0.998	0.64	0.78	0.747
Kstar	0.954	0.636	0.763	0.585
J48	0.959	0.645	0.771	0.633
DecisionTable	0.998	0.64	0.78	0.747

表 6　基于 $\rho_{density}$ 的发布后缺陷预测结果

分类器	recall	precision	F-measure	Acc
NaiveBaye	0.99	0.495	0.66	0.716
BayesNet	0.767	0.591	0.668	0.656
Kstar	0.7	0.628	0.662	0.665
J48	0.728	0.608	0.663	0.658
DecisionTable	0.744	0.601	0.665	0.657

对于发布后的数据,召回率为 70% 以上,除 NaiveBaye 外,精确度范围为 60% 左右。对于 NaiveBaye 算法而言,高召回率(99%)是以低精确度为代价的,但是可以在高危性软件中应用此模型。F-measure 值为 66%,Acc 值都大于 65%,笔者认为这些结果相当不错。其他 Eclipse 版本也获得了相似结果。虽然预测模型还不是很完善(特别是发布后的缺陷),但该模型可以帮助管理者作出明智的决定,通过缺陷预测对模块分配有限的资源。

4　结束语

究竟使用代码静态属性的哪些子集来预测缺陷引起了广泛争议。本文在分析了公共缺陷数据集 Eclipse 数据集的基础上,研究了静态代码属性与缺陷数之间的相关关系,同时还研究了不同代码属性与缺陷数之间的偏相关关系。然后在存在偏相关的属性上计算出复杂度密度值属性,并以此属性建立了分类器,得到了令人满意的预测效果。

先前的研究者在选取了最优属性集上比较预测效果,但是不同的数据集会产生不同的结论。即最佳的属性子集很脆弱,当变化数据集和分类器时可能不再适用。本文分析了存在这种情况的原因,并利用统计学意义上的偏相关系数分析了静态代码属性之间存在偏相关关系。即可能产生某个数据集上的某个属性集最优的假象。在此基础上,利用五类分类器,进一步以复杂度密度值属性进行缺陷预测,令人满意的结果进一步证实了该偏相关关系的存在。笔者希望本文的工作可以帮助人们更好地理解静态代码度量属性与软件质量之间的关系。

这项研究结果是在 Eclipse 数据集上实验获得的。如果这些数据集是有严重缺陷的(如错误数据收集和记录有大问题),本文的结果可能是无效的。今后笔者一方面将在不同的大型项目上进一步评估本文的研究结果,另一方面将基于本文的研究结论,建立能较目前方法更好的缺陷预测模型。

参考文献:

[1] CATAL C. Software fault prediction: a literature review and current trends[J]. Expert Systems with Applications: An International Journal,2011,38(4):4626-4636.

[2] LIU Yi, KHOSHGOFTAAR T M, SELIYA N. Evolutionary optimization of software quality modeling with multiple repositories[J]. IEEE Trans on Software Engineering,2010,36(6):852-864.

[3] MENZIES T, MILTON Z, TURHAN B, *et al*. Defect prediction from static code features: current results, limitations, new approaches[J]. Automated Software Engineering,2010,17(4):375-407.

[4] AUBRAMANYAN R, KRISHNAN M S. Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects[J]. IEEE Trans on Software Engineering,2003,29(4):297-310.

[5] GYIMOTHY T, RERENC R, SIKET I. Empirical validation of object-oriented metrics on open source software for fault prediction[J]. IEEE Trans on Software Engineering,2005,31(10):897-910.

[6] KHOSHGOFTAAR T M, SELIYA N. The necessity of assuring quality in software measurement data[C]// Proc of the 10th International Conference on Software Metrics. 2004:119-130.

[7] MENZIES T, GREENWALD J, FRANK A. Data mining static code attributes to learn defect predictors[J]. IEEE Trans on Software Engineering,2007,32(11):2-13.

[8] SHEPPED M, INCE D. A critique of three metrics[J]. Journal of Systems and Software,1994,26(3):197-210.

[9] FENTON N, OHLSSON N. Quantitative analysis of faults and failures in a complex software system[J]. IEEE Trans on Software Engineering,2000,26(8):797-814.

[10] ANDERSSON C, RUNESON P. A replicated quantitative analysis of fault distributions in complex software systems[J]. IEEE Trans on Software Engineering,2007,33(5):273-286.

(下转第 613 页)

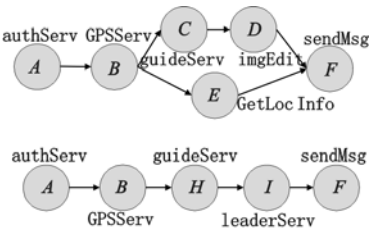

图3　Web服务组合模型示例

表 2　候选服务的 QoS 属性

service list	price	response time	reliability
$C[1]$	41	[72,80]	0.64
$C[2]$	34	[76,93]	0.65
$D[1]$	38	[83,89]	0.75
$D[2]$	31	[78,101]	0.74
$E[1]$	36	[62,88]	0.84
$E[2]$	29	[56,93]	0.83
$H[1]$	47	[127,139]	0.78
$H[2]$	40	[142,150]	0.78
$I[1]$	36	[140,159]	0.83
$I[2]$	31	[151,178]	0.89

表 3　候选服务组合的 QoS 属性

scheme list	price	response time	reliability
$C[1]D[1]E[1]$	129	[210,234]	0.68
$C[1]D[1]E[2]$	112	[189,224]	0.69
$C[1]D[2]E[1]$	107	[207,240]	0.70
$C[2]D[1]E[1]$	100	[235,248]	0.71
$C[1]D[2]E[2]$	96	[230,255]	0.72
$C[2]D[2]E[1]$	87	[212,266]	0.72
$C[2]D[1]E[2]$	86	[242,272]	0.73
$H[1]I[1]$	94	[282,300]	0.81
$H[1]I[2]$	92	[290,313]	0.81
$H[2]I[1]$	88	[300,314]	0.85
$H[2]I[2]$	67	[310,334]	0.86

为验证求解成功率,测试服务分别设置为不同数量来比较。为了验证服务组合的求解算法 GDT 组合求解效率,本文借助两个服务组合算法,即简单的基于图的服务组合算法(WSCG)和基于图的局部最优服务组合算法(WSCGLO)。本文生成了多个相关服务,分别进行五次实验,可选服务分别为 100、200、500、800 和 1 000 个测试服务依次五种情况,GDT 算法和另外两个算法的成功率比较实验结果如图 4 所示。

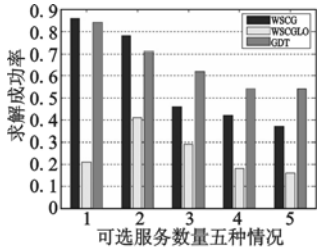

图4　求解成功率对比

4　结束语

本文研究工作主要针对即时任务服务组合的求解,提出一种基于领域本体概念和决策理论的服务组合方法。该方法从服务体系结构的抽象层角度,根据用户的功能性需求或粗粒度服务的输入和输出,利用领域本体概念求解满足功能需求的服务;而后对多个相似服务依据其与用户 QoS 需求的相近度排序,根据多个不同相似度服务组合成多个方案;同时在多个服务组合方案选取上,采用综合评估计算各方案优良度,根据优良度高低选择符合客户 QoS 需求的服务组合。通过实际分析,该方法能在数量较大的情况下能很好地适应,算法的时间消耗也在允许范围内,进一步证明了方法的有效性和可行性。

参考文献:

[1] 高俊,沈才梁,陈暄.一种面向服务体系结构的服务组合方案求解方法[J].计算机应用研究,2011,28(11):4184-4187.

[2] 崔晓峰, 孙艳春, 梅宏. 以决策为中心的软件体系结构设计方法[J]. 软件学报,2010,21(6):1196-1208.

[3] MEDJAHED B, BOUGUETTAYA A, ELMAGARMID A K. Composing Web services on the semantic Web[J]. The VLDB Journal, 2003,12(4):333-351.

[4] KO J M, KIM C O. Quality of service oriented Web service composition algorithm and planning architecture[J]. Journal of Systems and Software,2008,81(11):2079-2090.

[5] 范小芹,蒋昌俊,王俊丽,等.随机 QoS 感知的可靠 Web 服务组合[J]. 软件学报,2009,20(3):546-556.

[6] 李研,周明辉,李瑞超,等.一种考虑 QoS 数据可信性的服务选择方法[J]. 软件学报, 2008,19(10):2620-2627.

[7] CANFORA G, Di PENTA M, ESPOSITO R, *et al*. An approach for QoS-aware service composition based on genetic algorithms[C]//Proc of Conference on Genetic and Evolutionary Compulation. Washington DC: IEEE Computer Society,2005: 1069-1075.

[8] 孙吉贵,刘杰,赵连宇.聚类算法研究[J]. 软件学报,2008,19(1):48-61.

[9] 尹可挺.Internet 环境中基于 QoS 的 Web 服务组合研究[D]. 杭州:浙江大学,2010.

[10] ZHAO Zeng-liang, BOUALEM B, NGU A H H, *et al*. QoS-aware middleware for Web services composition[J]. IEEE Trans on Software Engineering,2004,30(5):311-327.

[11] Di NITTO E, GHEZZI C, METZGER A, *et al*. A journey to highly dynamic, self-adaptive service-based applications[J]. Automated Software Engineering,2008,15(3):313-341.

[12] Di MODICA G, TOMARCHIO O, VITA L. Dynamic SLAs management in service oriented environments[J]. Journal of Systems and Software,2009,82(5):759-771.

(上接第 596 页)

[11] ZHANG Hong-yu. An investigation of the relationships between lines of code and defects[C]//Proc of the 25th IEEE International Conference on Software Maintenance. 2009:274-283.

[12] FISHER R A. Statistical methods for research workers[M]. 10th ed. Edinburgh: Oliver and Boyd, 1946.

[13] ZIMMERMANN T, PREMRAJ R, ZELLER A. Predicting defects for Eclipse[C]// Proc of the 3rd International Workshop on Predictor Models in Software Engineering. 2007.

[14] VAPNIK V N. 统计学习理论的本质[M]. 张学工, 译. 北京:清华大学出版社,2000.

[15] KIM S, ZHANG Hong-yu, WU Rong-xin, *et al*. Dealing with noise in defect prediction[C]//Proc of the 33rd International Conference on Software Engineering. 2011:21-28.

[16] ZHANG Hong-yu, WU Rong-xin. Sampling program quality[C]// Proc of the 26th IEEE International Conference on Software Maintenance. 2010:1-10.

[17] JIANG Yuan, LI Ming, ZHOU Zhi-hua. Software defect detection with ROCUS[J]. Journal of Computer Science and Technology,2011,26(2):328-342.