基于决策树的软件可维护性预测方法研究

朱佳俊[1],王 炜[1,2],李 彤[1,2],唐 季[1]
(1. 云南大学 软件学院,云南 昆明 650091;
2. 云南省软件工程重点实验室,云南 昆明 650091)

摘 要:软件维护是软件全生命周期中一项高难度、高成本、长周期的活动,准确预测软件可维护性对降低软件维护成本、提高软件可用性具有重要意义。软件可维护性分析历经 20 多年的研究,当前的预测分析性能和准确率仍然不高,甚至达不到模型预测是准确的标准;而总结相关研究发现,软件可维护性数据还普遍存在数据分布不均衡问题,该问题将直接影响到模型预测的性能。针对上述问题,基于采样方法利用决策树建立软件可维护性预测模型,并通过 UIMS 和 QUES 数据集对模型进行实验验证。结果表明,与基线方法和现有的可维护性预测方法相比,文中方法在 UIMS 数据集和 QUES 数据集的平均误差率(MMRE)分别提高了 84% 和 61%,且 Pred(0.25)都达到了该评价标准的最优值 1,表明该方法具有更优的综合性能。

关键词:面向对象;软件可维护性;可维护性预测;采样方法;决策树
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2018)08-0066-05
doi:10.3969/j.issn.1673-629X.2018.08.014

Research on Software Maintainability Prediction Based on Decision Tree

ZHU Jia-jun[1], WANG Wei[1,2], LI Tong[1,2], TANG Ji[1]
(1. School of Software, Yunnan University, Kunming 650091, China;
2. Key Laboratory for Software Engineering of Yunnan Province, Kunming 650091, China)

Abstract: Software maintenance has been one of the most difficult, costly and long-term tasks in the software development lifecycle. Accurate prediction of software maintainability can be useful to reduce the cost and improve the usability of software. The process of software maintainability research has gone through more than 20 years, but predicting performance and accuracy is still not high, and even fail to reach the standards that the model prediction is accurate; and summary of the study found that the common problem of imbalance of data distribution exist in software maintainability, which will directly affect the performance of the model prediction. For this, we apply decision tree to construct the software maintainability prediction model based on the sampling method, and use the UIMS and QUES datasets to conduct the experiment. The result shows that compared with the baseline method and the existing maintainability prediction method, the MMRE of the UIMS data set and QUES data set is improved by 84% and 61% respectively. And both of datasets achieve the optimum value 1 of Pred (0.25). The result suggests the proposed method has a better comprehensive performance.

Key words: object-oriented; software maintainability; maintainability prediction; sampling method; decision tree

0 引 言

软件维护是软件全生命周期中一项高难度、高成本、长周期的活动,将花费软件总资源的 60% ~ 80%[1]。预测软件产品的可维护性,能有效支持软件维护工作,例如决策支持、资源分配、成本控制等。预测软件可维护性对降低软件维护成本、提高软件可用性具有重要意义。

1993 年 Li 和 Henry[2] 定义了用软件维护期间每个类的代码修改行数来表示面向对象软件的可维护性,用 CHANGE 变量表示。近年来,随着机器学习的发展,陆续提出了一些使用机器学习的方法来预测面向对象软件的可维护性。基于机器学习的方法通常希望构建一个较好的模型,以更好地刻画可维护性数据,从而提高软件可维护性的预测准确率。已有模型包括

收稿日期:2017-08-29 **修回日期**:2018-01-10 **网络出版时间**:2018-04-28
基金项目:国家自然科学基金(61462092,61379032);云南省自然科学基金重点项目(2015FA014)
作者简介:朱佳俊(1991-),女,硕士研究生,CCF 会员(58960G),研究方向为软件工程、软件演化、数据挖掘;王 炜,博士,副教授,CCF 会员,研究方向为软件工程、软件演化、数据挖掘;李 彤,博士,教授,博导,CCF 高级会员,研究方向为软件工程、信息安全。

神经网络模型[3-5]、贝叶斯模型[6]、向后消除[6]、逐步选择[6]、MARS[7]、SVM[7-8]、GMDH[9]、GA[9]、PNN[9]等。但是,软件可维护性预测研究经过 20 多年的发展,其预测精度始终不高,较好的平均预测误差仅能达到 0.210[9]。

软件可维护性预测精度涉及两个核心问题:训练数据好坏和预测模型。针对这两个问题,文中基于抽样方法采用决策树对面向对象软件可维护性进行研究,构建了相应的预测模型,并利用可维护性数据集对构建的模型进行实验验证。

1　文中方法

文中方法的框架如图 1 所示,包含三部分:数据集获取与预处理、决策树建模和可维护性预测。决策树建模是主要部分,文中按模型的要求将获取的数据集进行预处理形成输入,经过训练数据生成预测模型,最后使用验证集对生成的模型进行验证,完成面向对象软件可维护性预测过程。

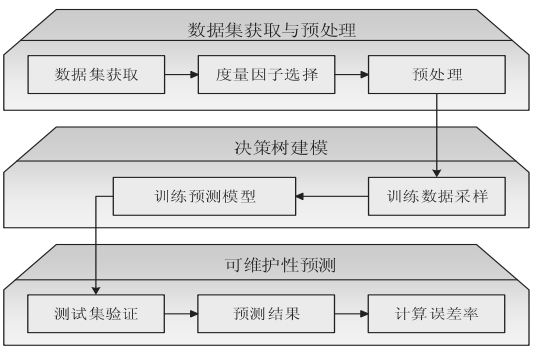

图 1　文中方法框架

1.1　数据集获取与预处理

(1)数据集获取。

数据集获取是所有工作的基础。为了保证文中方法的客观性,采用该领域比较流行的两个面向对象系统数据集:UIMS(user interface system)和 QUES(quality evaluation system)[2]。这两个数据集采用 Ada 语言编码实现以类为单位,一个类代表一个实例,其中 UIMS 系统有 39 个类,QUES 系统有 71 个类,共 110 个类。

(2)度量因子选择。

度量因子定义了如何描述软件可维护性,表 1 给出了各度量标准的详细描述。文中采用 11 个度量因子,其中五个 C&K[10] 度量因子:DIT、NOC、RFC、LCOM、WMC,四个 L&H[2] 度量因子:MPC、DAC、NOM、$SIZE_2$,以及一个传统代码行数度量因子 $SIZE_1$。目标变量可维护性度量因子用 CHANGE 表示,代表了每个类在维护期间代码更改(增加或删除)行数。UIMS 数据集和 QUES 数据集都采用了上述 11 个度量因子进行描述。

表 1　度量准则

度量因子	描述
DIT	(depth in the inheritance tree)表示类的继承层次
NOC	(number of children)表示一个类的直接子类数
RFC	(response for class)度量类的响应基数
LCOM	(lack of cohesion of methods)度量低内聚的方法
WMC	(weighted method complexity)度量所有方法的静态复杂度
MPC	(message-passing coupling)度量类间消息传递的复杂度
DAC	(data abstraction coupling)度量由抽象数据类型(ADT)引起的耦合复杂度
NOM	(number of methods)度量一个类中所定义方法的数量
$SIZE_2$	表示一个类中的属性和方法数
$SIZE_1$	表示代码行数,用分号来计算
CHANGE	表示在维护期间代码的更改(增加或删除)行数

(3)预处理。

数据预处理的目的是把原始数据按照模型要求经过归一化和数据集划分,将其形成模型的特定输入。经过分析原始数据,发现存在同一度量因子的不同样本值或同一样本的不同度量因子值相差很大的情况。为了缩小数据样本各度量因子间的数值差异性,采用归一化方法将所有有用数据映射到[0,1]区间。

$$X^* = \frac{X - \min}{\max - \min} \tag{1}$$

1.2　决策树建模

(1)训练数据采样。

可维护性训练集的划分能直接影响所构建预测模型的精度和性能,好的训练集能覆盖所有样本空间,训练出的预测模型更符合问题域,也能得到更符合实际的预测结果。但是软件可维护性数据集中普遍存在数据分布不平衡问题,在可维护性数据集中,无需演化的模块数量远比需要演化的模块数量多,因此随机划分训练集和测试集存在划分不均衡的情况,导致建立的预测模型在预测可维护性行为时偏向数量多的模块,而对数量少的模块预测精度偏低。然而这并不符合软件演化的实际情况,而且对可维护性的错误预测代价较高。为了缓解可维护性数据的分布不平衡问题,提高所建立模型的预测准确率,采用聚簇再划分的方式,利用 K-means 算法把数据集样本先聚成 K 簇,再分别从 K 簇中提取相应比例的样本构成训练集和测试集。算法伪代码描述如下:

输入:可维护性数据集 $D = \{(x_1^*, y_1), (x_2^*, y_2), \cdots, (x_t^*, y_t)\}$,聚类数 K,采样比例 P;

输出:训练数据集。

过程:

For D 中每个数据

K-means 聚成 K 类

End For

For 每个类簇

按比例 P 抽取训练集

End For

(2)训练预测模型。

决策树(decision tree,DT)[11]是一个树结构(可以是二叉树或非二叉树),表示了对象属性和对象值之间的一种映射,该结构由树的分支来对对象的属性进行分类和预测[12]。决策树相比其他机器学习算法不仅易于理解和实现,可读性好,效率高,而且在进行学习任务时也表现出了强大的性能,因而在各领域应用广泛。

具体来说,假设训练数据集 $D = \{(x_1^*, y_1), (x_2^*, y_2), \cdots, (x_t^*, y_t)\}$,那么对于回归问题,即是从一个 n 维的输入变量 x 预测一个目标变量 y 值的过程,如果输入空间的划分给定,即决策树结构给定,且最小化平方和误差函数,则任意一个区域预测目标变量的最优值就落在这个区域数据点 y 值的平均上。

一棵决策树的生成过程主要分为 3 个部分:

①特征选择:指从训练数据的众多特征中选择一个特征作为当前节点的分裂标准,根据选择不同特征量化评估标准,衍生出不同的决策树算法。

②决策树生成:根据选择的特征评估标准,从上至下递归地生成子节点,直到数据集不可分则决策树停止生长。显然,决策树的生成过程是一个递归过程。

③剪枝[13]:决策树容易过拟合,一般需要剪枝缩小树结构规模,缓解过拟合。剪枝技术有预剪枝和后剪枝两种。剪枝遵循的准则是在残留误差与模型复杂度之间进行平衡。

然而在实际应用中通常会先构建一棵较大的树,使用与叶节点关联的数据点数量的停止准则,然后再剪枝,生成最终的决策树。因此,生成决策树最重要的过程变成了剪枝。假设 T_0 代表剪枝开始的树,对于 $T_i \in T_0$,如果 T_i 能够从 T_0 剪枝(即通过合并对应区域来收缩内部节点)得到,那么它就被定义为 T_0 的一棵子树。假设叶节点为 $L_i(i = 1,2,\cdots,m)$,L_i 表示具有 N_i 个数据点的区域 R_i。那么区域 R_i 给出的最优预测为:

$$y_i' = \frac{1}{N_i} \sum_{x_n \in R_i} y_n \tag{2}$$

y_i' 对残留的平方和误差为:

$$Q_{L_i}(m) = \sum_{x_n \in R_i} (y_n - y_i')^2 \tag{3}$$

因此,剪枝标准为:

$$C(m) = \sum_{i=1}^{m} Q_{L_i}(m) + \lambda m \tag{4}$$

正则化参数 λ 确定了整体的残留平方和误差与模型复杂度的折中,模型复杂度用叶节点的数量 m 表示,它的值通过交叉验证的方式确定。通过使用上述方法构建的决策树,可以高效地对未知数据进行预测,决策树生成伪代码如下:

输入:训练数据集 $D = \{(x_1^*, y_1), (x_2^*, y_2), \cdots, (x_t^*, y_t)\}$,特征集 $\alpha_i(i = 1,2,\cdots,n)$;

输出:预测结果。

过程:

特征选择:

For 每个特征

For 每个特征值

将数据集切分为两部分

计算切分误差

If 当前误差<当前最小误差

将当前切分设定为最佳切分并更新最小误差

End for

End for

返回最佳切分的特征和阈值 α_i

构建决策树:

For 数据集中每个样本

If 该节点不能再分

则把当前节点的数据均值作为叶节点

Else

寻找当前最佳待切特征和特征值并返回

If $X_i > \alpha_i$

在左子树调用构建决策树方法

Else if $X_i \leqslant \alpha_i$

在右子树调用构建决策树方法

End for

剪枝:

For 所构建决策树中的每个节点

If T_i 是一棵树

计算将当前两个叶节点合并后的误差

计算不合并的误差

If 合并误差<不合并误差

将叶节点合并

End for

1.3 可维护性预测

利用 1.2 生成的模型进行可维护性预测是一个回归过程,具体为将可维护性数据按度量因子方法进行度量,然后按相同预处理方法进行处理,形成可维护性预测模型的输入数据,通过模型预测出对应的代码更改行数 CHANGE 值,完成可维护性预测过程。

2 实验验证

2.1 评价方法

为了验证该方法的有效性和客观性,采用该领域常用的评价标准绝对残差、相对误差率和 Pred(q)对实验预测结果进行度量和评价。

(1)绝对残差(Ab. Res.)[6]:表示可维护性预测值和实际值差的绝对值。文中用绝对残差的中位数来度量残差分布的集中趋势,表示为 Med. Ab. Res. 。Med. Ab. Res. 值越小,说明预测值和实际值的差异越小,预测越准确。

$$\text{Ab. Res.} = | \text{实际值} - \text{预测值} | \tag{5}$$

$$\text{Med. Ab. Res.} = \text{Ab. Res.}_{0.5} \tag{6}$$

(2)相对误差率(MRE)[4,6-7,9,14-15]:表示实际值和预测值间的归一化程度。文中用 MRE 的最大值 Max. MRE 和平均值 MMRE 来度量可维护性预测精度。MMRE 用来度量实际值和预测值之间的平均差异,MMRE 是主要评价标准;Max. MRE 用来度量实际值和预测值之间的最大差异。

$$\text{MRE} = \frac{| \text{实际值} - \text{预测值} |}{\text{实际值}} = \frac{\text{Ab. Res.}}{\text{实际值}} \tag{7}$$

$$\text{MMRE} = \overline{\text{MRE}} \tag{8}$$

$$\text{Max . MRE} = \text{MRE}_{\max} \tag{9}$$

MMRE 值越小,说明实际值和预测值之间平均差异越小,预测准确率越高;Max. MRE 值越小,说明实际值和预测值之间最大差异越小,预测准确率越高。

(3)Pred(q)[4,6-7,9,14-15]:度量所有预测值中 MRE 值小于或等于特定值的比例[16],值在[0,1]区间。

$$\text{pred}(q) = \frac{K}{N} \tag{10}$$

其中, q 为特定值;K 为预测值中 MRE 值小于或等于 q 的数量; N 为预测数据集总个数。Pred(q)值越小,说明预测准确的比例越大。文中用 Pred(0.25)来度量。

2.2　实验过程

数据是实验的基础,文中使用 UIMS 数据集的 39 个类和 QUES 数据集的 71 个类进行实验。根据数据集的数据分布情况,分别将 UIMS 数据集和 QUES 数据集聚成 3 簇和 4 簇,然后采用随机抽样的方式,分别从每个簇中抽取了 80% 的数据样本构成训练集,剩下 20% 的数据样本作为测试集。

实验使用 Python3. 5 实现,并采用 Scikit - learn 0.18.1的 Tree 包创建决策树模型。模型的好坏一方面与划分训练集的合理性有关,另外还与模型参数设置的值有关,模型参数设置的合理与否能影响预测结果的精确度。文中实验将所有的参数设置为默认值。

2.3　实验结果与分析

2.3.1　基线方法选择

为了验证文中方法的准确性,实验需选择基线方法进行对比。文献[6]建立了贝叶斯模型、回归树模型、向后消除和逐步选择模型,通过实验得出贝叶斯模型的预测**而而数据**于另外三个模型。文献[4]采用人工神经网络(ANN)进行研究,得出 ANN 模型能够用于软件可维护性预测。文献[9]采用数据分组处理(GMDH)方法、遗传算法、概率神经网络进行研究。文献[14]采用多层感知机、支持向量机、径向基网络及异构集成方法进行研究,发现基于最好训练(BT)集成方法的预测准确率优于单模型方法。

由于文献[4]和文献[9]未给出具体 UIMS 和 QUES 数据集的实验结论,因此文中选择文献[6]和文献[14]的实验结果作为基准方法进行侧面比对。另外,为了使实验更公平客观,设置了一组对照实验,采用当前取得较好预测精度的 GMDH 模型作为基线方法,在同等环境下对相同数据集进行测试。

2.3.2　预测结果分析

(1)UIMS 数据集。

表 2 为采用 UIMS 数据集实验结果对比,图 2 为 UIMS 数据集的 MMRE 和 Pred(0.25)实验结果对比。

表 2　UIMS 数据集实验结果

模型	MMRE	Max. MRE	Pred(0.25)	Pred(0.3)	Med. Ab. Res.
文中方法	0.045	0.128	1	1	0
基线方法	0.286	0.754	0.571	0.571	7.125
文献[6]	0.972	7.039	0.446	0.469	10.550
文献[14]	0.970	–	–	0.641	–

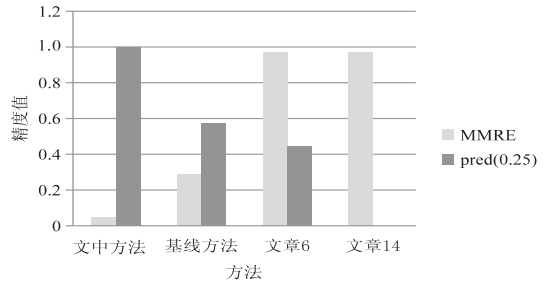

图 2　UIMS 数据集实验结果

根据 UIMS 实验结果可以得出:文中方法得到的 MMRE 和 Max. MRE 最小,Pred(0.25)达到最优值 1;对比基线方法,文中方法的预测结果其 MMRE、Max. MRE 和 Pred(0.25)分别提高了 84%、83% 和 75%;对比文献[6]和文献[14],文中方法得到的 MMRE 提高了 95%,Pred(0.3)分别提高了 113% 和 56%;Med. Ab. Res. 为 0,对比基线方法和文献[6],文中方法在该指标上提高了 100%;通过对 UIMS 数据集的实验结果分析,可以得出文中方法的预测性能更好,预测准确率更高(MMRE 为 0.045)。

(2)QUES 数据集。

表 3 为采用 QUES 数据集的实验结果对比,图 3 为 QUES 数据集的 MMRE 和 Pred(0.25)实验结果对比。

根据 QUES 实验结果可以得出:文中方法得到的 MMRE 和 Max. MRE 最小,Pred(0.25)达到最优值 1;

对比基线方法,文中方法的预测结果其 MMRE、Max. MRE 和 Pred(0.25)分别提高了 61%、73% 和 30%;对比文献[6]和文献[14],文中方法得到的 MMRE 分别提高了 81% 和 79%,Pred(0.3)分别提高了 132% 和 18%;Med. Ab. Res. 为 0,对比基线方法和文献[6],文中方法在该指标上提高了 100%;通过对 QUES 数据集的实验结果分析,可以得出文中方法的预测性能更好,预测准确率更高(MMRE 为 0.084)。

表 3 QUES 数据集实验结果

模型	MMRE	Max. MRE	Pred(0.25)	Pred(0.3)	Med. Ab. Res.
文中方法	0.084	0.2	1	1	0
基线方法	0.216	0.747	0.769	0.846	8.228
文献[6]	0.452	1.592	0.391	0.430	17.560
文献[14]	0.410	–	–	0.845	–

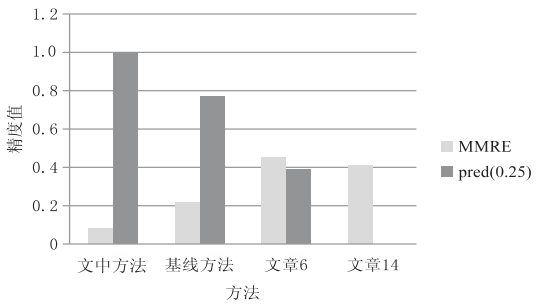

图 3 QUES 数据集实验结果

综上可得,在面向对象系统可维护性预测上,文中方法的性能和准确率更高,而且 Pred(0.25)都达到了该评价标准的最优值,表明该模型所有预测结果的误差率都小于 0.25,达到了最优状态,验证了该方法能够显著提高软件可维护性预测能力。

3 结束语

基于采样方法利用决策树算法对 UIMS 数据集和 QUES 数据集分别构建了面向对象软件可维护性预测模型,并通过测试集对构建的预测模型进行了验证。通过设定基线方法进行对照实验,同时与文献[6]和文献[14]的研究结果进行侧面比对,并采用该领域常用的评价标准-绝对残差、平均误差率和 Pred(0.25)对实验预测结果进行度量和评价。结果表明,该方法的预测性能更好,准确率更高。另外,该方法对两个数据集预测结果的 Pred(0.25)指标都达到了该评价标准的最优值 1,表明该模型所有预测结果的误差率都小于 0.25,达到了最优状态。

但是该方法也存在一些问题:模型的参数设置为默认值,有一定局限性,参数设置的好坏与模型的性能有很大关系,未来将针对参数的设置问题进行探讨;另外,采用的 UIMS 和 QUES 数据集发布时间较早且数据实例少,未来研究中,可以探索更有利于该领域研究的数据集。

参考文献:

[1] BHATT P, WILLIAMS K, SHROFF G, et al. Influencing factors in outsourced software maintenance[J]. ACM SIGSOFT Software Engineering Notes, 2006, 31(3): 1-6.

[2] LI Wei, HENRY S. Object-oriented metrics that predict maintainability[J]. Journal of Systems & Software, 1993, 23(2): 111-122.

[3] THWIN M M T, QUAH T S. Application of neural networks for software quality prediction using object-oriented metrics[J]. Journal of Systems & Software, 2003, 76(2): 147-156.

[4] AGGARWAL K K, SINGH Y, KAUR A, et al. Application of artificial neural network for predicting maintainability using object oriented metrics[J]. Enformatika, 2006, 15: 285.

[5] 管 笛,周明全,林晓燕,等. 多层感知器在提高软件可维护性上的应用[J]. 计算机应用与软件, 2009, 26(11): 4-6.

[6] VANKOTEN C, GRAY A R. An application of Bayesian network for predicting object-oriented software maintainability[J]. Information & Software Technology, 2006, 48(1): 59-67.

[7] ZHOU Yuming, LEUNG H. Predicting object-oriented software maintainability using multivariate adaptive regression splines[J]. Journal of Systems and Software, 2007, 80(8): 1349-1361.

[8] 陈雪娟,潘梅森,雷超阳. 基于 SVM 的软件可维护性评估模型研究[J]. 计算机工程与设计, 2008, 29(3): 566-569.

[9] MALHOTRA R, CHUG A. Software maintainability prediction using machine learning algorithms[J]. Software Engineering: An International Journal, 2012, 2: 19-36.

[10] CHIDAMBER S R, KEMERER C F. Towards a metrics suite for object oriented design[C]//Proceedings on object-oriented programming systems, languages, and applications. Phoenix, Arizona, USA: ACM, 1991: 197-211.

[11] SAFAVIAN S R, LANDGREBE D. A survey of decision tree classifier methodology[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1991, 21(3): 660-674.

[12] 杨学兵,张 俊. 决策树算法及其核心技术[J]. 计算机技术与发展, 2007, 17(1): 43-45.

[13] 李道国,苗夺谦,俞 冰. 决策树剪枝算法的研究与改进[J]. 计算机工程, 2005, 31(8): 19-21.

[14] ELISH M O, ALJAMAAN H, AHMAD I. Three empirical studies on predicting software maintainability using ensemble methods[J]. Soft Computing, 2015, 19(9): 2511-2524.

[15] JIN Cong, LIU Jinan. Applications of support vector mathine and unsupervised learning for predicting maintainability using object-oriented metrics[C]//Second international conference on multimedia and information technology. Kaifeng, China: IEEE, 2010: 24-27.

[16] FENTON N, PFLEEGER S L. Software metrics: a rigorous & practical approach[M]. 北京: 清华大学出版社, 2003.