基于代价敏感集成分类器的长方法检测

刘丽倩 董 东
(河北师范大学数学与信息科学学院 石家庄 050024)

摘 要 长方法(Long Method)是由于一个方法太长而需要重构的软件设计的问题。为了提高传统机器学习方法对长方法的识别率,针对代码坏味数据不平衡的特性,提出代价敏感集成分类器算法。以传统决策树算法为基础,利用欠采样策略对样本进行重采样,进而生成多个平衡的子集,并将这些子集训练生成多个相同的基分类器,然后将这些基分类器组合形成一个集成分类器。最后在集成分类器中引入由认知复杂度决定的误分类代价,使得分类器向准确分类少数类倾斜。与传统机器学习算法相比,此方法对长方法检测结果的查准率和查全率均有一定提升。

关键词 长方法,代码坏味,代价敏感,认知复杂度
中图法分类号 TP311 **文献标识码** A

Long Method Detection Based on Cost-sensitive Integrated Classifier

LIU Li-qian DONG Dong
(College of Mathematics and Information Science, Hebei Normal University, Shijiazhuang 050024, China)

Abstract Long method is a software design problem that requires refactoring because it is too long. In order to improve the detection rate of traditional machine learning approaches on long method, a cost-sensitive integrated classifier algorithm was proposed from the viewpoint of unbalanced sample data of code smell. Based on the traditional decision tree algorithm, the under-sampling startegy is used for resampling, then a plurality of balanced subsets are generated. These subsets are trained to generate a plurality of same base classifiers. Finally, the mistaken classification cost determined by the cognitive complexity is complemented to the integrated classifier. The cost makes the classifier inclined to the accuracy rate of the minority categories. Compared with the traditional machine learning algorithm, this method has improved the precision and recall for detection result of long methods.

Keywords Long method, Code smell, Cost-sensitive, Cognitive complexity

1 引言

Fowler 等[1]于 1999 年定义了 22 种软件设计错误,称为代码坏味,指在面向对象的软件设计中因违背软件设计原则而导致的设计错误。长方法是 22 种代码坏味中的一种,指一个方法的代码行数太多,过于复杂,几乎集中了该方法所在类的大多数功能,让人难以理解,且访问了其他类的大量成员,耦合度太高[2],违反了面向对象的软件设计的单一性原则。

识别和检测代码坏味对于代码重构具有重要的意义。Rao 等[3]提出了基于逻辑的检测技术,通过运用包含大量类属性和类之间关系的模糊逻辑规则,并用逻辑规则对候选坏味进行排序。Moha 等[4]研究了基于表征的检测方法,要求把不同的代码坏味的征兆转换成检测算法,并选择合适的阈值。针对运用这种方法处理相同的征兆的代码坏味,不同学者具有不同看法,准确率较低。Kosba 等[5]提出了基于语义的检测技术,将风险概念应用于代码坏味检测过程中,并开发了一个基于风险的代码坏味自动检测工具。刘秋荣[6]提出了面向长方法检测的阈值动态优化方法,该方法根据长方法的判断检测历史,利用程序员的反馈,自动动态优化阈值,从而提高结果的查准率。

越来越多的学者研究用机器学习的方法来检测代码坏味。Kreimer[7]提出用决策树来检测长方法和上帝类。Fontana 等[2]针对长方法在内的 4 种代码坏味的检测,运用朴素贝叶斯、随机森林等多种机器学习算法做了大量实验,并对比了实验结果。Maiga 等[8]提出基于 SVM 的检测反模式的方法,这种方法主要检测 Blob, Functional Decomposition 等坏味。Khomh 等[9]介绍了一种在开源项目中用 BDTEX (Bayesian Detection Expert)算法来检测代码坏味,这种方法是利用反模式的定义来建立贝叶斯网络,并用特定的代码坏味来验证该方法。Khomh 等[10]提出了用贝叶斯途径来检测 Blob 坏味的发生。

然而,上述基于机器学习的方法都是在假设正负样例的数量相差很少的情况下才会取得理想的结果,但事实上代码坏味数据集是不平衡的,负样例的数量远远大于正样例的数量[11]。这就导致了传统的机器学习算法为了提高总体分类准确率,会牺牲少数类的分类准确率,而本文重点研究的长方法属于少数类。

本文考虑代码坏味数据不平衡的特性,设计了代价敏感集成分类器,并使用长方法数据集进行有效性检验。以传统决策树算法为基础,利用欠采样策略对样本进行重采样,进而生成多个平衡的子集,并把这些子集训练生成多个基分类器,将这些基分类器组合形成一个集成分类器。最后在集成分类

刘丽倩(1991—),女,硕士生,主要研究方向为经验软件工程,E-mail:1437784143@qq.com;董 东(1971—),男,硕士,副教授,CCF 会员,主要研究方向为经验软件工程,E-mail:dongdong@hebtu.edu.cn(通信作者)。

器分类属性选择中引入误分类代价，其中误分类代价是由认知复杂度决定的，使得分类器能向准确分类少数类倾斜。实验结果表明，该方法在检测准确率方面得到了显著提高。

本文第 2 节主要介绍所提方法涉及到的代价敏感的基本概念；第 3 节介绍所提方法的主要步骤和原理；第 4 节展示实验结果；最后总结全文。

2 代价敏感学习的基本概念

代价敏感学习[12]对分类中不同的误分决策会分配不同的代价。Bahnsen 等[13]提出了代价敏感贝叶斯最小风险分类器，用于监测信用卡诈骗，通过引入真实经济代价度量，设置欺诈交易被检测成合法交易所付出的代价远远大于合法交易被检测成欺诈交易，从而提高监测诈骗交易的准确性。陶新民等[14]提出代价敏感的支持向量机算法，用于轴承故障检测，该算法首先对边界样本点进行数据过抽样，利用 K 最近邻构造代价矩阵，并利用每个样本的代价函数来消除噪声样本对 SVM 算法分类精度的影响。Kai 等[15]提出了一种用实例加权法使决策树代价敏感的方法。Liu 等[16]提出改变训练集类别分布的代价敏感学习算法。类似地，在医疗诊断中，将患癌病人诊断为健康人所付出的代价远远大于将健康人诊断为患癌病人。代价敏感主要用于减小误分少数类带来的影响，但误分的少数类的影响远远大于误分多数类的影响。

代价是代价敏感学习的基础，认知学家 Feldman[17]曾指出，在人类概念的学习过程中，概念的复杂度会随着该概念中蕴含关系前键个数的增多而成比例增加。

对于代码坏味来说，不管是正例预测为负例，还是负例预测为正例，都需要付出人工脑力劳动，把这种人工脑力劳动量化为认知复杂度，因此把认知复杂度作为代价的度量。分别从空间方面和结构方面来分析认知复杂度[18]。

面向对象的源代码空间复杂度(Code Spatial Complexity，CSC)由方法的定义与调用之间的距离来度量，距离用定义与调用之间的源代码行数表示。每个方法的空间复杂度[18](Code-spatial Complexity of a Method，MCSC)定义为：

$$MCSC=\frac{\sum_{i=1}^{n}D_i}{n} \tag{1}$$

其中，n 代表调用的次数，D_i 代表方法定义与调用之间的源代码行数。具有 m 个方法的类的源代码空间复杂度[18]为：

$$CSC=\frac{\sum_{i=1}^{m}MCSC_i}{m} \tag{2}$$

当方法的定义和调用在同一个编译单元中时，距离就是两者之间的绝对距离；当两者不在同一个编译单元中时，距离[18]定义为：

D=当前编译单元顶部的方法调用的源代码行数+包含定义的编译单元顶部的定义方法的源代码行数+0.1*(剩余编译单元的全部代码行数)/2

由于认知复杂度也受到控制语句和数据类型的影响，因此源代码认知复杂度取决于定义与调用之间的距离，控制语句的类型、方法和参数的类型，因此定义方法的认知复杂度[18](Method Cognitive Complexity，MCC)为：

$$MCC=W_c*D+\sum_{i=1}^{N_{ip}+N_{op}}W_{p_i} \tag{3}$$

其中，W_c 表示方法调用控制语句的认知权重，D 表示式(1)中方法定义和调用之间的距离，N_{ip} & N_{op} 表示方法输入和输出参数的数量，W_{p_i} 表示参数 p_i 的认知权重。式(3)中，MCC 由两部分构成，第一部分表示认知复杂度取决于空间距离，第二部分表示认知复杂度受到方法输入输出参数的影响。控制语句和参数类型的认知权重因子如表 1 所列。

表 1 认知权重因子

类别	基本控制结构	权重
顺序	顺序	1
分支	If-then-else	2
	case/switch	3
迭代	for-do，while，do-while	3
	嵌套	4
常量	字面量	1
	枚举型/静态变量	1
变量	基本数据类型	1
	引用类型	2
	群(collection)	3
	多维数组等	4

因此，认知复杂度可以计算所有方法调用的 MCC 平均值 CCC[18]。

$$CCC=\frac{\sum_{j=1}^{m}MCC(MC_j)}{m}=\frac{\sum_{j=1}^{m}(W_c*D(MC_j)+\sum_{i=1}^{N_{ip}+N_{op}}W_{P_i}(MC_j))}{m} \tag{4}$$

其中，m 表示软件所有方法调用的数量，MC_j 表示第 j 次调用。

3 代价敏感集成分类器

本节主要介绍代价敏感集成分类器的设计。

3.1 欠采样

欠采样是通过删除多数类中的一些样例，与少数类形成一定的比例，最终到达平衡状态。Tahir 等[19]提出一种新的随机欠采样的方法，基本思想是首先对负样例进行多次独立随机抽样，把每个负样例子集与正样例组成一个训练集，在每个训练集中找到线性的正负样例划分规则。Phua 等[20]提出了一种最近邻规则欠抽样方法，该方法删除其最近的 3 个近邻样本中的 2 个或 2 个以上类别不同的样本。但由于大多数的多数样本附近的样本都是多数类，因此该方法所能删除的多数类样本十分有限。Laurikkala 等[21]针对训练样本集中的每个样本找出其 3 个最近邻样本，若该样本是多数类样本且其 3 个最近邻中有 2 个以上是少数类样本，则删除它；反之，当该样本是少数类，并且其 3 个最近邻中有 2 个以上是多数类样本，则去除近邻中的多数类样本。

本文通过改变正负样例的不同比例和决策树的实现，来最终选择一个实验结果最好的比例。因为长方法是小类，为了提高小类的分类效果，通过计算正负样例不同比例的分类效果，选择 1∶5 作为基分类器的样例比例，如图 1 所示。

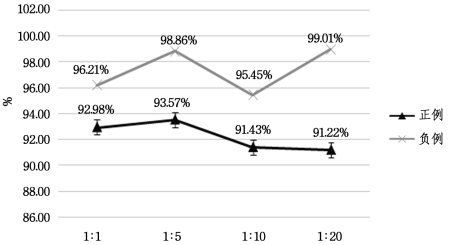

图 1 欠采样不同比例分类查准率

3.2 集成分类器

从多数类中独立、随机抽取若干子集，并将每个子集与少数类数据联合起来训练生成多个基分类器，最终将这些基分类器组合形成一个集成分类器。集成分类器示意图如图 2 所示。

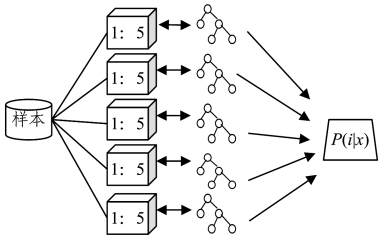

图 2　集成分类器示意图

通过重采样得到 5 个训练子集，并训练生成 5 个分类模型。根据得到的模型计算训练集中每个样本属于每个类别的概率 $P(i|x)$。

3.3 代价敏感集成分类算法

代价敏感算法通过指定不同误分产生不同的代价并选择误分代价最小的策略来使传统的机器学习算法代价敏感，从而提高小类的分类准确率。

集成分类器中可得到训练集中每个样本属于每个类别的概率 $P(i|x)$，按照贝叶斯最小风险决策 $R(i|x)=\sum_j P(i|x)Cost(i,j)$，其中 $R(i|x)$ 表示将样本 x 分成第 i 类的期望代价，如果期望代价越小，样本 x 属于第 i 类的概率就越大，计算每个样本属于各个类别的期望代价 $R(i|x)$，并将训练集中的每一个实例重标记为代价最优的类。

3.4 代价矩阵的表示形式

代价矩阵是代价敏感算法的基础，给定一个具体的样例，该样例应该被预测为具有最小贝叶斯风险的类别。通常用 $Cost(i,j)$ 表示把类别 j 预测为类别 i 的代价。对于一个两类的问题，假设用 0 表示负类，用 1 表示正类，此外负类为多数类，正类为少数类，则 $Cost(0,1)$ 表示把真实的正类预测为负类的代价，$Cost(1,0)$ 表示把真实的负类预测为正类的代价，而 $Cost(0,0)$ 和 $Cost(1,1)$ 表示正确分类负类和正确分类正类。一般情况下，正确分类正类和正确分类负类的代价都为 0，而误分的代价可以根据第 2 节介绍的认知复杂度来决定。通过认知复杂度的计算，可以得出具体代价矩阵，如表 2 所列。

表 2　代价矩阵

	真实正类	真实负类
预测正类	$Cost(1,1)=0$	$Cost(1,0)=1$
预测负类	$Cost(0,1)=5$	$Cost(0,0)=0$

代价矩阵中正确预测样例的代价为 0，负例被预测为正例的代价是正常阅读代码需要的代价，指定为 1，通过给定的认知复杂度的计算式(4)，计算长方法数据集中所有方法的平均复杂度，得到正类预测为负类的代价为 5。

对于给定的代价矩阵，样例的最优预测为该预测具有最小的贝叶斯风险。

$$R(i|x)=\sum_j P(j|x)Cost(i,j)$$

其中，$P(j|x)$ 为把样例类别划分为 j 的后验概率。

完整的代价敏感集成分类器算法如算法 1 所示。

算法 1　代价敏感集成分类器算法

输入：带有类别标签的方法集 S，代价矩阵 C，重采样次数 5

输出：分类模型 M

1. for j=1,2,…,5；
2. 从 S 中随机重采样，得到 5 个训练集 S_i；
3. 分类器对每个训练集进行学习，得到 5 个分类模型 M_i；
4. for 训练集每个样本 x；
5. 计算样本属于每个类别的概率 P(i|x)；
6. 计算样本 x 属于每个类别的期望代价 R(i|x)；
7. 根据最小期望代价，修改每个样本的类别标签；
8. 训练修改过的数据集，得到新的分类模型。

4 实验设计和结果分析

为验证算法的有效性，本节主要介绍检测代码坏味所用到的数据集，以及代价敏感集成分类器与其他机器算法性能的比较。

4.1 实验数据集

实验选取开放数据集 Landfill，Landfill 是一个基于 Web 平台收集共享代码坏味的公共数据集。该数据集包含 40 个存在代码坏味的软件项目，并对最常见到的几种代码坏味进行了标记。

如表 3 所列，本课题实验选取的软件项目是 Apache Hive 和 Apache Lucene。通过抽象语法树共提取了 91020 个方法，并对其中存在的 94 个 Long Method 进行了标记。

表 3　实验数据

项目	LOC	个数	网址
Apache Hive	60664	60	http://www.sesa.unisa.it/landfill/index.jsp
Apache Lucene	30356	34	http://www.sesa.unisa.it/landfill/index.jsp

4.2 数据预处理

对于机器学习而言，学习算法需要形式化输入和输出数据，因此本文计算了面向对象的度量，这些度量被认为是独立变量。

针对 Long method 的检测，从 Landfill 获取 Apache Hive 和 Apache Lucene 项目，运用抽象语法树，在项目中计算出了每个方法的长度(MethodLine)、圈复杂度(McCabe)、LCOM 等度量值，并标记项目中的所有长方法，以便后续识别。

4.3 实验设计

本文方法是以传统的决策树为基础，首先对数据进行多次采样，训练生成基分类器，并组合成集成分类器，然后引入代价矩阵，使得传统的机器学习算法代价敏感。机器学习算法需要输入输出数据，因此需要将软件项目中的所有方法的特征值提取出来。本次实验采用了抽象语法树算法，通过遍历树中的节点得到方法的特征值。

对于决策树算法，使用 R 语言中 party 包中的函数 *ctree*()来建立决策树，用函数 *predict*()对新数据进行预测。

4.4 评价标准

传统机器学习算法的性能都是从整体分类情况考虑的，但由于数据的不平衡性，整体分类性能并不能体现出本文所关注的少数类的分类效果。采用查准率(Precision)和查全率(Recall)两个指标来评价代价敏感集成分类器的分类性能。其中，查准率用于考查结果的准确性，查全率用于考查结果的

完整性。定义如下：

$$查准率=\frac{正确识别出的长方法总数}{识别出的长方法的总数}$$

$$查全率=\frac{正确识别出的长方法总数}{数据集中实际存在的长方法总数}$$

4.5 实验结果分析

从表 4 中可以看出，传统的机器学习算法（如决策树、随机森林）在长方法检测结果中由于重点关注了总体的分类性能，因此反例的查准率已经非常高，但正例的查准率却很低，误分代码坏味的代价非常高。本文提出的代价敏感集成分类器设定误分少数类的代价远远高于误分多数类的代价，分类器为了使总代价最低，必然会向提高少数类的查准率和查全率倾斜。实验结果表明，代价敏感集成分类器显著提高了正例的分类查准率和查全率，同时反例的查准率和查全率也保持了较高的水平。

表 4 3 种不同方法的结果对比

（单位：%）

方法类别	决策树			随机森林			代价敏感集成分类器		
	查准率		查全率	查准率		查全率	查准率		查全率
长方法	正例	反例		正例	反例		正例	反例	
	13.2	99.99	46.67	22.64	99.97	64.32	100	99.88	88.46

结束语 本文在传统机器学习算法的基础上提出一种代价敏感集成分类器对软件中存在的长方法进行检测。传统机器学习算法在不平衡数据下的分类效果并不理想，为了降低不平衡数据造成的这种影响，首先对数据欠采样，选取多数类的子集和少数类组成相对平衡的数据，通过多次欠采样形成多个数据集，并训练生成多个基分类器，组成集成分类器。为了进一步提高少数类的分类性能，引入以认知复杂度为基础的代价矩阵，并选择分类代价最小的决策。实验结果表明，代价敏感集成分类器大大提高了长方法的检测性能；同时，本文着重介绍了长方法的检测算法，后续也会研究能否使用该方法检测其他代码坏味。

参 考 文 献

[1] FOWLER M. Refactoring: Improving the Design of Existing Code [M]. Lecture Notes in Computer Science, 1999: 256.

[2] FONTANA F A, ZANONI M, MARINO A. Comparing and Experimenting Machine Learning Techniques for Code Smell Detection[J]. Empirical Software Engineering, 2016, 21(3): 1143-1191.

[3] RAO A A, REDDY K N. Detecting Bad Smells in Object Oriented Design Using Design Change Propagation Probability Matrix [M]. Lecture Notes in Engineering & Computer Science, 2008.

[4] MOHA N, GUEHENEUC Y G, DUCHIEN L, et al. DECOR: A Method for the Specification and Detection of Code and Design Smells[J]. IEEE Transactions on Software Engineering, 2010, 36(1): 20-36.

[5] KOSBA E, ABDELMOEZ W, IESA A F. Risk-Based Code Smells Detection Tool[C]// International conference on Computing Technology and Information Management. 2014.

[6] 刘秋荣. 面向代码坏味检测的阈值动态优化方法[D]. 北京：北京理工大学，2016.

[7] KREIMER J. Adaptive Detection of Design Flaws[J]. Electronic Notes in Theoretical Computer Science, 2005, 141(4): 117-136.

[8] MAIGA A, ALI N, BHATTACHARYA N, et al. Support Vector Machines for Anti-pattern Detection[C]// IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012: 278-281.

[9] KHOMH F, VAUCHER S, SAHRAOUI H. BDTEX: A GQM-based Bayesian Approach for the Detection of Antipatterns[J]. Journal of Systems & Software, 2011, 84(4): 559-572.

[10] KHOMH F, SAHRAOUI H. A Bayesian Approach for the Detection of Code and Design Smells[C]// International Conference on Quality Software. IEEE, 2010: 305-314.

[11] MALHOTRA R, KHANNA M. An empirical study for software change prediction using imbalanced data[J]. Empirical Software Engineering, 2017, 22(6): 1-46.

[12] ELKAN C. The Foundations of Cost-Sensitive Learning[C]// Seventeenth International Joint Conference on Artificial Intelligence. 2001: 973-978.

[13] BAHNSEN A C, STOJANOVIC A, AOUADA D, et al. Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk[C]// International Conference on Machine Learning and Applications. IEEE, 2014: 333-338.

[14] 陶新民，刘福荣，童智靖，等. 不均衡数据下基于 SVM 的故障检测新算法[J]. 振动与冲击，2010，29(12)：8-12.

[15] KAI M T. Inducing Cost-sensitive Trees via Instance Weighting [C]// European Symposium on Principles of Data Mining and Knowledge Discovery. Berlin Heidelberg: Springer-Verlag, 1998: 139-147.

[16] LIU X Y, ZHOU Z H. The Influence of Class Imbalance on Cost-Sensitive Learning: An Empirical Study[C]// International Conference on Data Mining. IEEE Computer Society, 2006: 970-974.

[17] FELDMAN J. An Algebra of Human Concept Learning[J]. Journal of Mathematical Psychology, 2006, 50(4): 339-368.

[18] CHHABRA J K. Code Cognitive Complexity: A New Measure [M]. Lecture Notes in Engineering & Computer Science, 2011, 2191(1).

[19] TAHIR M A, KITTLER J, MIKOLAJCZYK K, et al. A Multiple Expert Approach to the Class Imbalance Problem Using Inverse Random under Sampling[C]// International Workshop on Multiple Classifier Systems. Berlin Heidelberg: Springer-Verlag, 2009: 82-91.

[20] PHUA C, ALAHAKOON D, LEE V. Minority Report in Fraud Detection: Classification of Skewed Data[J]. Acm Sigkdd Explorations Newsletter, 2004, 6(1): 50-59.

[21] LAURIKKALA J. Improving Identification of Difficult Small Classes by Balancing Class Distribution[C]// Conference on AI in Medicine in Europe: Artificial Intelligence Medicine. Berlin Heidelberg: Springer-Verlag, 2001: 63-66.