跨项目软件缺陷预测方法研究综述

陈 翔[1),2)] 王莉萍[1)] 顾 庆[2)] 王 赞[3)] 倪 超[2)] 刘望舒[2)] 王秋萍[1)]

[1)](南通大学计算机科学与技术学院 江苏 南通 226019)

[2)](南京大学计算机软件新技术国家重点实验室 南京 210023)

[3)](天津大学软件学院 天津 300072)

摘 要 软件缺陷预测首先通过挖掘与分析软件历史仓库,从中抽取程序模块并进行类型标记.随后通过分析软件代码的内在复杂度或开发过程特征,设计出与软件缺陷存在强相关性的度量元,并对这些程序模块进行度量.最后借助特定的机器学习方法基于上述数据构建出缺陷预测模型.因此该方法可以在项目开发的早期阶段,通过预先识别出项目内的可疑缺陷模块,达到优化测试资源分配的目的.但在实际软件开发场景中,需要进行缺陷预测的项目可能是一个新启动项目,或这个项目的历史训练数据比较稀缺.一种简单的解决方案是利用其他项目已经搜集的训练数据来构建缺陷预测模型.但不同项目之间因所处的应用领域、采用的开发流程、使用的编程语言、开发人员经验等并不相同,因此对应数据集间会存在较大的分布差异性并造成该方案的实际性能并不理想,因此如何通过有效迁移源项目的相关知识来为目标项目构建预测模型,吸引了国内外研究人员的关注,并将该问题称为跨项目软件缺陷预测问题.论文针对该问题进行了系统综述.根据预测场景的不同,将已有方法分为3类:基于有监督学习的方法、基于无监督学习的方法和基于半监督学习的方法.其中基于有监督学习的方法主要基于候选源项目集的程序模块来构建模型.这类方法根据源项目与目标项目采用的度量元是否相同又可以细分为同构跨项目缺陷预测方法和异构跨项目缺陷预测方法.针对前者,研究人员主要从度量元取值转换、实例选择和权重设置、特征映射和特征选择、集成学习、类不平衡学习等角度展开研究.而后者更具研究挑战性,研究人员主要基于特征映射和典型相关分析等方法展开研究.基于无监督学习的方法直接尝试对目标项目中的程序模块进行预测.这类方法假设在软件缺陷预测问题中,有缺陷模块的度量元取值存在高于无缺陷模块的度量元取值的倾向.因此研究人员主要基于聚类方法展开研究.而基于半监督学习的方法则会综合使用候选源项目集的程序模块和目标项目中的少量已标记模块来构建模型.这类方法通过尝试从目标项目中选出少量模块进行标记,以提高跨项目缺陷预测的性能.研究人员主要借助集成学习和TrAdaBoost方法展开研究.论文依次对每一类方法的已有研究成果进行了系统梳理和点评.随后论文进一步总结了跨项目缺陷预测研究中经常使用的性能评测指标和评测数据集,其统计结果可以辅助研究人员针对该问题进行合理的实验设计.最后总结全文,并分别从数据集搜集、数据集预处理、模型构建和评估、模型应用这4个维度对未来值得关注的研究方向进行了展望.

关键词 经验软件工程;软件缺陷预测;跨项目软件缺陷预测;迁移学习;实证研究

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2017.00000

A Survey on Cross-Project Software Defect Prediction Methods

CHEN Xiang[1),2)] WANG Li-Ping[1)] GU Qing[2)] WANG Zan[3)]

NI Chao[2)] LIU Wang-Shu[2)] WANG Qiu-Ping[1)]

[1)](*School of Computer Science and Technology, Nantong University, Nantong, Jiangsu* 226019)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

[3)](*School of Computer Software, Tianjin University, Tianjin* 300350)

Abstract Software defect prediction firstly analyzes and mines software historical repositories to extract program modules and label them. It secondly designs novel metrics, which have strong

收稿日期:2016-07-18;在线出版日期:2017-06-20.本课题得到国家自然科学基金(61202006,61202030,61373012,61402244,61602267)、南京大学计算机软件新技术国家重点实验室开放课题(KFKT2016B18)、江苏省高校自然科学研究项目(15KJB520030,16KJB520038)资助.**陈 翔**,男,1980年生,博士,副教授,硕士生导师,中国计算机学会(CCF)会员,主要研究方向为软件缺陷预测、回归测试、软件缺陷定位和组合测试.E-mail:xchencs@ntu.edu.cn.**王莉萍**,女,1992年生,硕士,主要研究方向为软件缺陷预测.**顾 庆**,男,1972年生,博士,教授,博士生导师,主要研究领域为软件质量保障.**王 赞**,男,1979年生,博士,副教授,硕士生导师,主要研究领域为软件测试.E-mail:wangzan@tju.edu.cn.**倪 超**,男,1990年生,博士研究生,主要研究方向为软件缺陷预测.**刘望舒**,男,1987年生,博士,主要研究方向为软件缺陷预测.**王秋萍**,女,1993年生,硕士研究生,主要研究方向为软件缺陷预测.

correlation with defects, based on the analysis on code complexity or development process. Then it uses these metrics to measure these program modules. It finally uses a specific machine learning algorithm to construct software defect prediction models, which are trained on these datasets. Therefore software defect prediction can optimize the software testing resource allocation by identifying the potential defect modules in advance. However in real software development, a project, which needs defect prediction, maybe a new project or it maybe has less training data. A simple solution is directly using training data from other projects to construct the model. However application domain, development process, used programming language, developer experience of different projects may be not same. This will cause the distribution of corresponding datasets to be large and result in the poor performance of defect prediction. Therefore, how to effectively transfer the knowledge of the source project to build a defect prediction model for the target project has attracted the attention of researchers, and this problem is called cross-project defect prediction (CPDP). We conduct a comprehensive survey on this topic and classify existing methods into three categories: supervised learning based methods, unsupervised learning based methods, and semi-supervised learning based methods. In particular, the supervised learning based methods will use the modules of the source project to construct the model. These methods can be further classified into two categories: homogeneous cross-project defect learning and heterogeneous cross-project defect prediction based on whether the source project and the target project use the same metric set. For the former, researchers design novel methods by using metric value transformation, instance selection and weight setting, feature mapping and selection, ensemble learning, class imbalance learning. For the latter, the issue is more challenging and researchers design novel methods by using feature mapping and canonical correlation analysis. The unsupervised learning based methods will attempt to make a prediction on the modules of the target project immediately. The assumption of these methods is that the metric value of defective modules has the tendency to be higher than the metric value of non-defective modules. Researchers design novel methods by using cluster algorithms. The semi-supervised learning based methods will use the modules of the source project and some labeled programs in the target project together to construct the model. These methods try to improve the performance of CPDP by identifying some representative program modules in the target project and label them manually. Researchers design novel methods by using ensemble learning and TrAdaBoost. We summarize and comment the existing research work for each category in sequence. Then we analyze the commonly used performance metrics and benchmarks in empirical studies in CPDP for other researchers to better design empirical studies. Finally we conclude this paper and discuss some potentially future research work from four dimensions: dataset gathering, dataset preprocessing, CPDP model construction and evaluation, and CPDP model application.

Keywords empirical software engineering; software defect prediction; cross-project defect prediction; transfer learning; empirical studies

1 引 言

软件在人们日常生活中的作用日益重要,而隐含缺陷的软件在部署后可能会产生意料之外的结果或行为,严重的时候会给企业带来巨额的经济损失,甚至有时候会引发人员伤亡.但在软件的整个开发生命周期中,检测出内在隐含缺陷的时间越晚,修复这些缺陷的代价也越高,尤其在软件完成部署后,检测和移除缺陷的代价将急剧增加.

软件缺陷预测(software defect prediction)[1-3]是当前软件工程数据挖掘领域[4]中的一个研究热点,其希望能够在项目开发的早期阶段,预先识别出项目内的潜在缺陷程序模块,并对这类程序模块分配足够的测试资源以确保可以进行充分的代码审查或单元测试,最终达到提高软件产品质量的目的.具体来说:软件缺陷预测首先通过挖掘与分析软件历史仓库(software historical repositories),从中抽取程序模块并进行类型标记.程序模块可以根据实际缺陷预测的需要,将粒度设置为函数(function)、类(class)或者包(package)等.目前研究人员经常挖掘的软件历史仓库包括版本控制系统(例如 SVN、CVS 或 Git 等)、缺陷跟踪系统(例如 Bugzilla、Mantis 或 Trac 等)以及开发人员相互之间发送的电子邮件等.随后通过分析软件代码的内在复杂度或开发过程特征,设计出与软件缺陷存在强相关性的度量元(metrics)[3,5],并借助这些度量元对已抽取的程序模块依次进行软件度量.通过对这些程序模块的软件度量和类型标记,可以构建出用于模型训练的缺陷预测数据集.最后基于特定的机器学习方法(例如 Logistic 回归、朴素贝叶斯、决策树、支持向量机等)构建出缺陷预测模型,并用于对项目中的新程序模块进行预测,其预测目标可以是模块内是否含有缺陷、含有的缺陷数或缺陷密度等.其具体流程如图 1 所示.

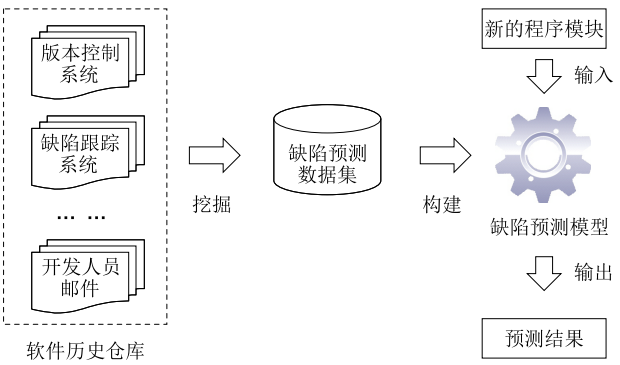


图 1　软件缺陷预测的流程图

目前大部分研究工作都集中关注同项目缺陷预测(Within-Project Defect Prediction,WPDP)问题,即选择同一项目的部分数据集作为训练集来构建模型,并用剩余未选择的数据作为测试集来获得模型的预测性能.但在实际的软件开发场景中,需要进行缺陷预测的目标项目可能是一个新启动项目,或这个项目已有的训练数据较为稀缺.目前在缺陷预测训练数据的搜集过程中,虽然借助一些软件度量工具(例如 Understand 工具①)可以较为容易地自动搜集到项目内程序模块的软件度量信息,但在随后分析这些模块内部是否含有缺陷时,则需要领域专家深入分析项目缺陷跟踪系统中的缺陷报告信息和版本控制系统中的代码修改日志,因此存在模块类型标记代价高昂且容易标记出错等问题.

一种简单的解决方案是直接使用其他项目(即源项目)已经搜集的高质量数据集来为目标项目构建缺陷预测模型.但不同项目的特征(例如所处的应用领域、采用的开发流程、使用的编程语言或开发人员的经验等)并不相同,所以源项目与目标项目的数据集存在很大的度量元取值分布差异,难以满足独立同分布的假设.因此在缺陷预测模型构建时,如何从源项目中迁移出与目标项目相关的知识是其面临的研究挑战,吸引了国内外研究人员的关注,并称该问题为跨项目缺陷预测(Cross-Project Defect Prediction,CPDP)问题.在跨项目软件缺陷预测时,若源项目与目标项目不是来自于同一企业,则该问题有时又被称为跨企业缺陷预测(cross-company defect prediction)问题.

针对该问题,大部分研究人员借助机器学习领域中的迁移学习(transfer learning)方法[6-7]来设计解决方案.迁移学习是运用已有知识,对具有一定相关性的领域的问题进行求解的一种机器学习方法,其目的是迁移已有知识来解决目标领域仅有少数已标记实例甚至没有的问题.因此 CPDP 问题可以视为迁移学习在软件缺陷预测领域中的一个重要应用场景.

为了对该研究问题进行系统的分析、总结和比较,我们首先在国内外重要的学术搜索引擎(例如 Google 学术搜索、DBLP、CNKI 等)中搜索与该综述主题相关的论文,选择的英文关键词包括"cross-project defect prediction"和"cross-company defect prediction".随后通过分析论文的标题、发表源、摘要、关键词来过滤掉与综述主题无关的论文.接着,我们通过分析选中论文的相关工作描述,来补充遗漏的论文.最后,我们通过 Google 学术、ACM、IEEE、Elsevier、Springer、Wiley、CNKI 等数据库中的论文引用分析和相关研究人员的已发表论文清单,来进一步补充遗漏的论文.最终我们确定了与该综述主题直接相关的论文共 56 篇(截止到 2016 年 7 月).

我们首先将每年的累计发表论文总数进行了统

① https://scitools.com/

计,最终结果如图 2 所示.不难看出,累计发表的论文数呈多项式增长.在 2012 年之前,每年发表的论文数并不多(介于 0 到 4 篇之间),但从 2012 年开始,随着共享出来的缺陷预测数据集的不断增多,针对 CPDP 问题的研究逐渐成为软件缺陷预测领域内的一个研究热点,每年发表的论文数至少达到 7 篇,甚至在 2015 年达到 13 篇.

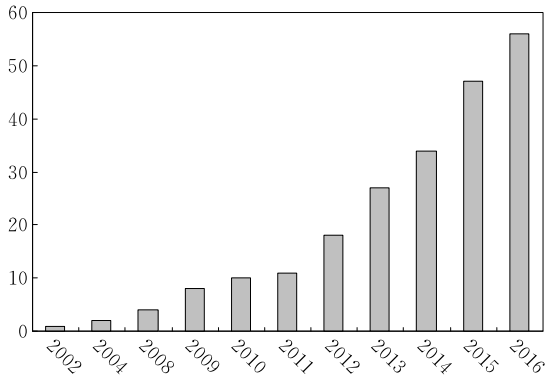


图 2 每年累计发表论文数的统计

随后我们从论文发表源这一角度,对论文的发表数进行了统计(仅统计了软件工程领域的权威期刊和会议),并按照论文数从大到小进行排序,最终结果如表 1 所示.其中排名前五的发表源分别是 TSE、ESE、ICSE、FSE/ESEC 和 IST.其中 ICSE 和 FSE/ESEC 是中国计算机协会推荐的软件工程领域中的 A 类会议,TSE 是 A 类期刊,而 ESE 和 IST 均是 B 类期刊.

表 1 论文的发表源统计

发表源简称	发表源全称	类型	论文数
TSE	IEEE Transactions on Software Engineering	期刊	6
ESE	Empirical Software Engineering	期刊	6
ICSE	International Conference on Software Engineering	会议	5
FSE/ESEC	ACM SIGSOFT Symposium on the Foundation of Software Engineering/European Software Engineering Conference	会议	4
IST	Information and Software Technology	期刊	4
MSR	Working Conference on Mining Software Repositories	会议	3
PROMISE	International Conference on Predictive Models and Data Analytics in Software Engineering	会议	3
ASE	International Conference on Automated Software Engineering	会议	2
ESEM	International Symposium on Empirical Software Engineering and Measurement	会议	2
ICSME	International Conference on Software Maintenance and Evolution	会议	1
ICST	The IEEE International Conference on Software Testing, Verification and Validation	会议	1
COMPSAC	International Computer Software and Applications Conference	会议	1
HASE	International Symposium on High Assurance Systems Engineering	会议	1
ASE	Automated Software Engineering	期刊	1
STVR	Software Testing, Verification and Reliability	期刊	1
SQJ	Software Quality Journal	期刊	1
JCST	Journal of Computer Science and Technology	期刊	1

基于上述分析,不难看出跨项目软件缺陷预测问题是近些年来软件缺陷预测领域中的一个研究热点.Hall 等人[1]在 2012 年重点分析了在 2000 年到 2010 年间发表的 36 篇文献,对影响缺陷预测模型性能的因素(例如模型的上下文、模型考虑的度量元以及建模方法等)进行了系统的分析,王青等人[2]在 2008 年对当时主流的软件缺陷预测技术进行了分类和比较.但这两篇综述并未对跨项目缺陷预测问题进行过分析.我们[3]在 2016 年提出了一种软件缺陷预测的研究框架,并对框架内的 3 个重要影响因素(度量元的设定、缺陷预测模型的构建方法以及缺陷预测数据集的相关问题)进行了深入分析.但该综述仅在缺陷预测数据集的相关问题中对跨项目缺陷预测问题进行过简单分析.为了对该问题的已有研究工作进行更为深入的总结,相对于综述文献[3],我们进一步补充了 30 余篇相关文献,提出了研究框架,并对框架内 3 种不同类型的方法进行了系统的分析和比较,同时总结了该研究问题常用的性能评测指标和评测数据集等.

本文第 2 节分析跨项目软件缺陷预测问题的研究框架;第 3 节到第 5 节分别总结基于有监督学习、无监督学习和半监督学习的跨项目软件缺陷预测方法的已有研究成果;第 6 节介绍常用的性能评测指标;第 7 节介绍常用的评测项目数据集;最后总结全文并对未来可能值得关注的研究问题进行了展望.

2 跨项目软件缺陷预测问题研究框架

基于已有跨项目软件缺陷预测的研究成果分析,可以简单总结出如下 3 个场景:

(1)场景 1. 仅存在与目标项目 T 相关的候选源项目的集合 $S=\{S_1,S_2,\cdots,S_n\}$. 其中 T 对应的数据集为 t，候选源项目 S_i 对应的数据集为 s_i.

(2)场景 2. 不仅存在与目标项目 T 相关的候选源项目的集合 S，而且还在 T 中存在少量已标记的目标项目模块，并将这些已标记的模块记为 t_l.

(3)场景 3. 难以找到与目标项目 T 相关的候选源项目，同时目标项目中也不存在已标记的模块.

针对上述 3 个不同的场景，目前已有的跨项目缺陷预测方法可以简单分为 3 类：

(1)基于有监督学习的方法. 这类方法主要针对场景 1，其基于候选源项目集 S 内的模块来训练缺陷预测模型，并对目标项目 T 中的程序模块进行预测.

(2)基于半监督学习的方法. 这类方法主要针对场景 2，其基于候选源项目集 S 内的模块和目标项目 T 中的少量已标记模块 t_l 来进行缺陷预测模型的训练，并用于预测目标项目中的未标记程序模块(即 $t-t_l$).

(3)基于无监督学习的方法. 该类方法主要针对场景 3，其直接尝试对目标项目 T 中的程序模块进行预测.

基于上述方法分类，在实际进行跨项目软件缺陷预测时，其研究框架可如图 3 所示.

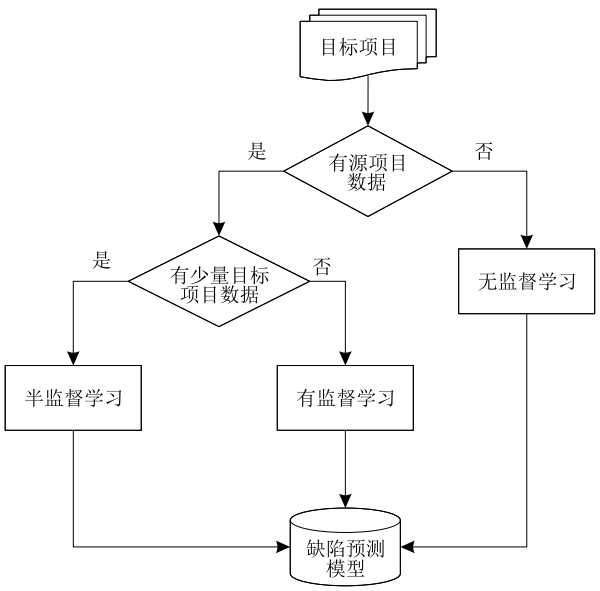


图 3　跨项目缺陷预测的研究框架图

基于上述方法分类，我们对相关论文数进行了统计，最终结果如图 4 所示. 不难看出，大部分研究工作(79%)都关注的是有监督学习方法. 但近些年来，研究人员对无监督学习和半监督学习方法的研究逐渐增多.

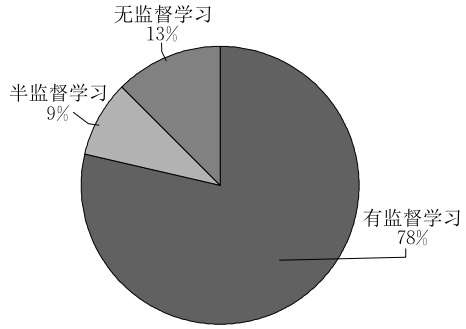


图 4　不同类型方法的论文数统计

3　基于有监督学习的方法

根据需要预测的目标项目和选择的源项目是否采用了相同的度量元(即是否具有相同的特征空间)，这类方法又可以细分为两类. 其中第一类假设源项目与目标项目采用了相同的度量元，第二类则假设源项目与目标项目采用的度量元并不完全一致. 研究人员将前一类问题称为同构跨项目缺陷预测(homogeneous cross-project defect learning)，而将后一类问题称为异构跨项目缺陷预测(heterogeneous cross-project defect prediction)，不难看出后一类问题更具研究挑战性.

3.1　同构跨项目缺陷预测方法

该小节首先分析了针对同构跨项目缺陷预测可行性调研的多个实证研究结果，由于考虑的评测数据集和评测指标等并不完全一样，因此造成一些研究工作之间的结论并不一致. 随后从度量元取值转换、实例选择和权重设置、特征映射和特征选择、集成学习、类不平衡学习等角度对已有方法进行总结.

3.1.1　可行性分析

据我们所知，Briand 等人[8]最早对 CPDP 的可行性展开了调研，他们主要借助 Logistic 回归和 MARS(Multivariate Adaptive Regression Splines)方法，尝试基于 Xpose 项目来构建缺陷预测模型，并对由同一开发团队开发(但采用了不同的设计策略和编码规范)的另一个项目 Jwrite 进行预测，他们发现 CPDP 模型在性能上虽然优于随机模型，但仍低于同项目缺陷预测的性能，因此他们认为项目间存在的差异性不可忽略.

Zimmermann 等人[9]随后针对 CPDP 的可行性展开了更大规模的实证研究. 他们首先对 Firefox 项目和 Internet Explorer 项目进行了 CPDP 分析，这两个项目虽然同属浏览器软件领域，但它们采用

的开发流程却存在较大的差异.在进行 CPDP 时,他们发现:基于 Firefox 项目构建的预测模型可以在 Internet Explorer 项目上取得很好的预测效果(其查准率为 76.47%,查全率为 81.25%),反之则不然(其查全率仅为 4.12%).他们推测其中的一个可能原因是从 Firefox 项目中搜集到的数据集规模要大于从 Internet Explorer 项目中搜集到的数据集的规模.随后他们又额外选择了来自微软公司和开源社区的 10 个大规模项目,累计分析了 622 对 CPDP,他们发现仅有 21 对(即 3.4%)可以取得满意的缺陷预测性能(即查准率、查全率和准确率均超过 75%).

He 等人[10]则假设如果 CPDP 的查全率超过 70%并且查准率超过 50%,则认为 CPDP 可以取得满意的预测性能,他们的假设标准要稍低于 Zimmermann 等人[9]提出的满意标准.他们选择了 10 个开源项目,基于 6 种不同分类方法的结果表明:在 160586 对 CPDP 中,仅有 0.32%到 4.67%的 CPDP 可以取得满意的缺陷预测性能.

Jureczko 和 Madeyski[11]从代码所有权(code ownership)角度,对 CPDP 的可行性进行了分析.他们考虑了 3 种不同类型的代码所有权,分别是来自工业界、开源界和学术界的项目,基于 35 个项目的累计 83 个版本的实证研究表明:基于一种类型的项目上构建的缺陷预测模型,在其他类型的项目上的缺陷预测效果并不理想.

但有些研究人员认为借助基于信息检索的度量指标(例如查准率、查全率、AUC 或 f-measure 等)来评估 CPDP 的性能并不合理.在实际软件测试中,受到测试资源的约束,测试部门无法对所有预测为有缺陷的程序模块进行单元测试或代码审查,因此他们认为更加合理的方式是从成本收益(cost-effectiveness)角度对 CPDP 的可行性进行分析.其中成本是需要审查的代码比例,收益是可以检测出的缺陷比例.Briand 等人[8]首次从这个角度出发,发现他们构建的模型在 CPDP 时,具有很好的经济可行性.随后 Rahman 等人[12]进一步基于其他开源项目进行了验证,即 CPDP 的性能并不一定比 WPDP 的性能差,并且要显著优于随机预测模型.

Fukushima 等人[13-14]则从跨项目缺陷预测角度对基于代码修改的缺陷预测问题[15]展开了研究.与传统的缺陷预测问题不同,该问题需要预测提交的代码修改内部是否含有缺陷,因此该问题将程序模块的粒度设置为代码修改.基于 11 个开源软件项目的实证研究,他们同样发现 CPDP 的性能要低于 WPDP 的性能.

Turhan[16]通过引入数据集漂移(dataset shift)的概念,深入分析了造成 CPDP 性能不理想的深层次原因.他们将数据集漂移的类型细分为 6 类:

(1)协变量漂移(covariate shift);

(2)先验概率漂移(prior probability shift);

(3)采样选择偏差(sample selection bias);

(4)类不平衡数据(imbalanced data);

(5)领域漂移(domain shift);

(6)源组件漂移(source component shift).

基于对上述数据集漂移类型的深入分析,他们推荐了两类解决方法:基于实例的方法和基于分布的方法.其中基于实例的方法主要包括:异常点检测、关联过滤、为实例设置权重等.而基于分布的方法主要包括:分层采样、成本曲线、混合模型等.上述分析为 CPDP 的后续研究提供了重要的理论支撑.

3.1.2 度量元取值变换方法

由于同一度量元在不同项目上的取值分布差异性较大,这类方法借助数据取值变换,尝试缩小源项目与目标项目数据集间的分布差异,以提高 CPDP 的性能.其取值变换的主要目标包括:使得度量元取值处于同一量纲中,度量元的取值分布更为对称或更接近正态分布,减少数据集中的异常点等.

假设将源项目 S 中第 j 个度量元视为向量 $\boldsymbol{s}^j$,则其第 i 个程序模块对应的度量元取值为 s_i^j.将目标项目 T 中第 j 个度量元视为向量 $\boldsymbol{t}^j$,则其第 i 个程序模块对应的度量元取值为 t_i^j.有的研究人员[17-18]针对源项目数据和目标项目数据同时进行取值变换,主要考虑的方法包括 min-max 标准化方法和 z-score 标准化方法.这些方法可以使得不同项目的度量元取值可以处于同一量纲中,且可以与原有的取值分布特征保持一致.以源项目的数据取值预处理为例,其中 min-max 标准化方法的计算公式为

$$\tilde{s}_i^j=\frac{s_i^j-\min(\boldsymbol{s}^j)}{\max(\boldsymbol{s}^j)-\min(\boldsymbol{s}^j)} \tag{1}$$

其中 $\max(\boldsymbol{s}^j)$和 $\min(\boldsymbol{s}^j)$分别为向量 $\boldsymbol{s}^j$中的最大值和最小值.

z-score 标准化方法的计算公式为

$$\tilde{s}_i^j=\frac{s_i^j-\mathrm{mean}(\boldsymbol{s}^j)}{\mathrm{std}(\boldsymbol{s}^j)} \tag{2}$$

其中 $\mathrm{mean}(\boldsymbol{s}^j)$和 $\mathrm{std}(\boldsymbol{s}^j)$分别为向量 $\boldsymbol{s}^j$的均值和标准差.

也有研究人员[19-20]将呈偏态分布(skew distribution)的度量元进行取对数转换，即 $\tilde{s}_i^j=\ln(s_i^j+1)$. 其中对原有度量元取值加 1 是为了确保预处理后的取值大于 0. 除此之外，Zhang[21] 还进一步考虑了 rank 转换和 Box-Cox 转换. 其中 rank 转换会将原有的度量元取值用对应的 rank 值进行替换，该转换方法尤其适用于取值呈重尾分布(heavy-tailed distribution)的度量元，其转换公式为

$$\tilde{s}_i^j=\begin{cases}1, & \text{若 } s_i^j\in[0,Q_1]\\ k, & \text{若 } s_i^j\in(Q_{k-1},Q_k]\text{且 } k\in\{2,\cdots,9\}\\ 10, & s_i^j\in(Q_9,Q_{10}]\end{cases}\quad(3)$$

其中 Q_k 是 $\boldsymbol{s}^j$ 中 $k\times10\%$ 的分位数.

Box-Cox 的转换公式为

$$\tilde{s}_i^j=\begin{cases}(\tilde{s}_i^{j\lambda}-1)/\lambda, & \text{若 } \lambda\neq0\\ \ln(s_i^j+1), & \text{若 } \lambda=0\end{cases}\quad(4)$$

其中 λ 是 Box-Cox 转换中的参数，其具体取值需要通过分析数据集的特征来确定.

不难看出，取对数转换、rank 转换和 Box-Cox 转换的主要目的是使得度量元的取值与正态分布更为接近.

Watanabe 等人[22]提出一种不同的度量元取值修正(metric compensation)方法. 该方法根据源项目度量元取值对目标项目度量元取值进行修正，其计算公式为

$$\tilde{t}_i^j=(t_i^j\times\text{mean}(\boldsymbol{t}^j))/\text{mean}(\boldsymbol{s}^j)\quad(5)$$

借助上述方法，他们希望可以有效提高源项目与目标项目的相似度. 他们分析了 Java 项目和 C++ 项目之间的 CPDP 效果. 发现该方法可以有效提高 CPDP 的查准率和查全率，其中在查全率上的提高程度更大.

3.1.3　基于实例选择的方法

目前执行跨项目缺陷预测时，一般存在多个候选源项目. 早期主要存在两种使用方式：1 对 1 方式和多对 1 方式，具体如图 5 所示. 其中 1 对 1 方式仅考虑使用单个源项目来构建模型，而多对 1 方式则将所有候选源项目的数据集汇总为一个数据集，并基于该数据集来构建模型.

但是上述两种方式均存在一定的局限性，例如：如果采用 1 对 1 方式，则可能存在选择的源项目与目标项目的相关度不高，或训练数据过少等问题，而如果采用多对 1 方式，则虽然增加了训练数据，但也引入了一些无关实例，因此会造成模型具有很高的误报率(即将很多无缺陷模块误预测为有缺陷模块)，从而造成了测试资源的浪费[23]. 为了更好地提升跨项目缺陷预测性能，一种更为合理的方式是在多对 1 方式的基础上，从训练数据集中识别并移除与目标项目无关的实例，这样可以确保充分利用训练集中的有用信息. 一些研究人员，借助实证研究，对这种方式的合理性进行了验证. Jureczko 和 Madeyski[24] 推断相似项目之间的预测效果会更好，他们借助 *K*-means 和 Kohonen 神经网络等聚类算法对一系列项目进行了聚类分析，并验证了他们的推断. Menzies 等人[25-26]验证了软件项目中的缺陷分布存在多个局部区域，因此通过对各个区域构建局部模型，其预测性能要优于全局模型. 具体来说，他们首先基于源项目和目标项目的数据进行聚类分析，并确保每个聚类中会同时包含这两个项目的数据，随后他们基于聚类中的源项目数据构建模型，并用于预测聚类中目标项目的实例. Bettenburg 等人[27-28]则对全局模型和局部模型进行了更为深入的分析. 他们认为局部模型对特定数据子集有效，而全局模型则更具有一般性，因此更为实用.

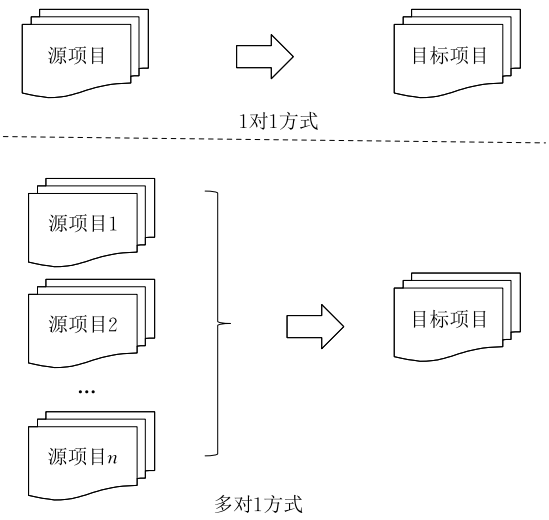


图 5　两种不同的跨项目缺陷预测方式

在实例选择时，目前主要存在 3 类方法：仅从项目角度进行筛选，仅从实例角度进行筛选，以及两阶段筛选(即先从项目角度进行筛选，随后针对选中的项目，从实例角度进行进一步筛选).

(1) 仅从项目角度进行筛选

Zimmermann 等人[9]从项目的上下文因素(例如项目所处领域、考虑的开发流程等)角度出发，共提出了 40 种不同的项目上下文因素来计算项目间的相似性. 基于实证研究中考虑的 622 对 CPDP 结果和对应项目间的相似度，构建出决策树，为候选源项目的选择提供了一系列指南.

He 等人[10]尝试借助度量元的数据分布特征来为目标项目选择合适的源项目.他们考虑了 16 种不同的分布特征,其中 mode、median、mean 和调和均值(harmonic mean)等特征可以描述度量元取值的中心趋势,而 range、变化率(variation ratio)、interquartile、range、variance、标准差(standard deviation)和变异系数(coefficient of variation)等特征可以描述度量元取值的分散程度.基于已有的 CPDP 预测结果,他们首先构造出训练集.具体来说针对每对 CPDP,其源项目每个度量元的 16 种分布特征、目标项目每个度量元的 16 种分布特征、以及是否可以取得满意的 CPDP 性能可构成一个实例.基于上述训练集,他们借助决策树生成了一系列源项目选择准则.

同样 Herbold[29]首先计算出每个项目的度量元分布特征向量,随后提出两种选择策略.其中第一种策略是根据分布特征向量对所有项目进行聚类分析,并选出与目标项目处于同一簇的源项目.第二种策略则基于 K 近邻法,即选出与目标项目距离最为接近的前 k 个源项目.基于 14 个开源项目的 44 个版本,他们发现上述选择策略可以有效提升 CPDP 性能,但仍与 WPDP 性能存在一定的差距.

Fukushima 等人[13-14]从不同角度提出了一种项目间相似度的计算方法.他们首先基于源项目数据集,计算出各个度量元与类型特征之间的 Spearman 相关系数.随后他们选择取值最高的前 3 个度量元(q_1,q_2,q_3),并在目标项目中也选择同样的 3 个度量元(r_1,r_2,r_3),随后在源项目上计算 q_1 和 q_2、q_1 和 q_3 以及 q_2 和 q_3 之间的 Spearman 相关系数,并得到向量 $\boldsymbol{Q}=(Q_1,Q_2,Q_3)$,同样在目标项目上也可以得到向量 $\boldsymbol{R}=(R_1,R_2,R_3)$,最后通过计算向量 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 的欧氏距离,得到两个项目间的相似度.

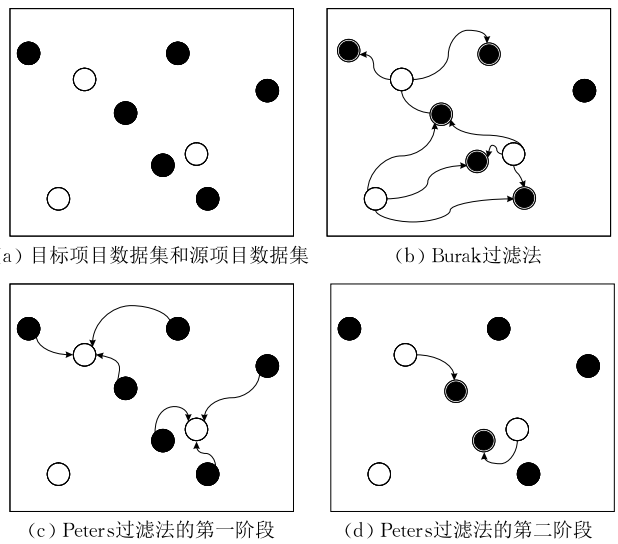
(2)仅从实例角度进行筛选

这类方法一般建立在多对 1 方式的基础之上,即首先将所有源项目中的实例汇总为一个候选训练集,随后根据需要预测的目标项目,从候选训练集中选出相关实例来训练模型.

Turhan 等人[23]提出了 Burak 过滤法.具体来说,他们首先计算出目标项目中实例与候选训练集中实例的欧氏距离.随后依次为目标项目中的每一个实例,从候选训练集中选出距离最近的 $k(k=10)$ 个实例并添加到最终的训练集中.假设目标项目中含有 N 个实例,则最终会选出 $k\times N$ 个实例,但候选训练集中可能会存在一个实例被多次选中,则最终仅选择一次.Burak 过滤法虽然可以提升 CPDP 的性能,但仍低于 WPDP 方法.

Turban 等人从目标项目出发进行实例选择,该方法假设目标项目含有足够的实例.而 Peters 等人[30]则认为候选训练集中包含的信息更多(即若考虑了很多候选源项目,则含有的实例数更多,因此与缺陷相关的信息也会更多).因此他们提出了 Peters 过滤法.具体来说:首先针对候选训练集中的每个实例,从目标项目中识别出与之距离最近的实例并进行标记.随后对于目标项目中已标记的实例,从候选训练集中选出与之距离最近的实例并添加到最终的训练集中.

Burak 过滤法[23]和 Peters 过滤法[30]的区别如图 6 所示.其中白圈为目标项目中的实例,黑圈为候选训练集中的实例(如子图(a)所示).对于 Burak 过滤法来说,针对每个目标项目实例,会从候选训练集实例中选出与之距离最近的 3 个实例(假设 $k=3$)(如子图(b)所示).而 Peters 过滤法则相反,首先从候选训练集实例中选出距离最近的一个实例(子图(c)所示).随后针对选中的目标项目实例,再从候选训练集实例中选出距离最近的一个实例(如子图(d)所示).基于图 6 所示,最终 Burak 过滤法从候选源项目中选出 5 个相关实例,而 Peters 过滤法仅选出 2 个相关实例.



(a)目标项目数据集和源项目数据集 (b)Burak过滤法 (c)Peters过滤法的第一阶段 (d)Peters过滤法的第二阶段

图 6 Burak 过滤法与 Peters 过滤法的区别

(3)两阶段筛选

He 等人[31]首次提出了一种两阶段筛选方法,该方法包括如下两个阶段:① 首先计算出候选源项目与目标项目间的相似度,并选出前 K 个最为相似

的源项目；② 随后对每一个选出的源项目，通过与目标项目的分析，移除掉一些不稳定的特征来减少两个项目间的数据差异性；③ 最后借助 bagging 集成学习来得到最终的缺陷预测结果. 实证研究表明：他们的方法在预测性能和计算开销上要优于 Turhan 等人[23]提出的 Burak 过滤法.

MA 等人[32]同样提出了一种两阶段筛选方法 TDS. 他们的方法首先在第一阶段根据需要预测的目标项目，借助项目度量元的分布特征取值(考虑的是 max、min、median、mean 和标准差)，根据欧氏距离，选出前 K 个候选源项目，这些项目的分布特征与目标项目最为接近. 随后在第二阶段，根据 Burak 过滤法[23]或 Peters 过滤法[30]，从第一阶段中选出的候选源项目中选出与目标项目最为接近的实例. 随后李勇等人[33]提出了相似的两阶段筛选方法，他们首先基于项目的分布特征向量(考虑的是 max、min、mean 和标准差)，根据余弦距离，选出与目标项目最为相关的前 k 个源项目，随后进一步借助 Peters 过滤法[30]选出相关实例.

基于实例选择的方法是同构跨项目软件缺陷预测研究中的一种主流方法，因此研究成果也相对较多. 从早期仅从项目选择角度或实例选择角度进行筛选，逐渐发展到更为合理的两阶段筛选(即先项目筛选后实例筛选). 基于已有研究工作的分析，不难看出，在项目选择角度，主要通过分析项目间的度量元取值分布来衡量不同项目间的相似度，而在实例选择时，主要通过计算实例间的距离(例如欧氏距离、余弦距离等)来衡量不同实例间的相似度. 但上述方法均是启发式方法，在实际应用中，在项目相似度计算方法和距离公式的选择上还需要进行针对性的选择或定义新的计算方法，以确保更好的预测效果.

3.1.4　基于实例权重设置的方法

Ma 等人[34]从为源项目中的实例设置权重出发，提出一种新颖的跨项目缺陷预测方法 TNB(Transfer Naive Bayes)，该方法通过对目标项目数据的分布进行预测，来为候选源项目中的实例设置权重. 但该方法仅适用于朴素贝叶斯分类方法. 同样程铭等人[35]也提出一种新的加权贝叶斯迁移学习方法，该方法首先计算出源项目和目标项目的特征信息，随后计算特征差异，将源项目与目标项目数据之间的差异转化为源项目实例中的权重，最后基于这些权重信息建立跨项目缺陷预测模型.

3.1.5　基于特征映射和选择的方法

Nam 等人[17]借助特征映射，即采用 TCA(Transfer Component Analysis)方法[36]来进行跨项目缺陷预测. TCA 方法在保持原有数据属性的基础上，借助对数据分布间的距离最小化，来为目标项目和源项目寻找潜在特征空间，并将这两个项目映射到该空间中. 在实证研究中，以 ReLink[37]和 AEEEM[38]作为评测数据集，他们发现数据标准化方法的选择(例如 z-score 标准化方法、min-max 标准化方法等)对 TCA 方法的性能存在一定的影响，因此他们又提出了 TCA+方法，该方法可以在分析源项目和目标项目的特征后，借助预先学习的一系列规则，自动选出最优的数据标准化方法.

实践表明缺陷预测数据集一般存在维数灾难(curse of dimensionality)问题，即数据集中含有的冗余特征或无关特征会造成缺陷预测模型的性能下降、模型的复杂度过高或模型训练时间过长等问题. 特征选择是缓解维数灾难的一种有效方法. He 和 Ma 等人[19]从这个角度出发展开了研究，他们首先通过分析不同数据集，借助特征选择方法选出不同特征子集. 随后选出出现次数最多的前 k 个特征，并构造出 top k 特征子集，最后他们针对每个数据集，通过进一步分析该子集中特征之间的相关性，从而移除其中的冗余特征，并构造出最小化特征子集. 而 Amasaki 等人[39]则从目标项目入手，尝试移除其中的无关特征和无关实例. 由于目标项目中的实例并未标记，因此他们借助无监督学习方法来识别出这些无关特征和无关实例.

3.1.6　基于集成学习的方法

集成学习(ensemble learning)通过构建并结合多个个体分类器来完成最终缺陷预测模型的构建. 集成学习若能取得好的性能，需要满足两个条件：(1)个体分类器需要具备较好的性能，即至少要优于随机预测；(2)不同个体分类器的预测结果需要具备多样性.

Liu 等人[40]借助基于搜索的策略，使用遗传规划(genetic programming)，通过考虑多个源项目数据，来提高模型性能. 他们提出了 3 种不同的策略：baseline 策略，validation 策略和 validation-and-voting 策略. 结果表明最后一种策略的效果更好，不易产生过拟合问题.

Panichella 等人[18]基于主成分分析(principal component analysis)对 6 种不同分类方法的等价性

进行了深入分析,他们发现不同分类方法预测出的有缺陷模块并不完全等同.随后他们提出了一种基于集成学习的方法 CODEP(Combined Defect Predictor),具体来说:首先基于 6 种分类器得到每个程序模块的预测值,随后这 6 种预测值作为该程序模块的新特征值,来训练出最终的分类器.在第二个阶段他们考虑了两种不同的分类方法(即 logistic 回归和贝叶斯网络).

随后 Zhang 等人[41]在 Panichella 等人[18]的研究工作基础上,进一步考虑了更多的集成学习方式,例如:平均投票、最大投票、Bagging、Boosting 和随机森林等,来尝试进一步提高跨项目缺陷预测的性能.

Zhang[21]从度量元取值变换角度出发,他们发现不同转换方法的效果并不等价,因此借助集成学习方法,提出一种组合方法,来进一步提高 CPDP 的性能.

Fukushima 等人[13-14]在集成学习的时候,考虑了简单的投票法.他们发现了一个有意思的现象:在集成学习时,若移除一些相似度低的候选源项目,却并不一定能够提高集成学习方法的性能.因此他们认为在跨项目缺陷预测时,搜集更多的源项目数据比移除相似度低的源项目更有意义.

戴翔和毛宇光[42]首先借助 Burak 过滤法[23],从源项目中选出相关实例,随后借助类不平衡学习方法(例如 SMOTE 方法[43]),使得训练集可以达到类平衡状态.最后针对类别平衡的数据集,综合运用了决策树、朴素贝叶斯、随机森林、支持向量机等 7 种常见的分类方法,并借助投票方式进行集成学习.

上述研究工作表明集成学习是进一步提升 CPDP 性能的一种有效方法.但集成学习要取得更好的预测性能,需要满足两个前提条件,而目前仅文献[18,21]对这两个前提条件进行过分析.除此之外,大部分已有研究工作考虑的集成学习方法还比较简单(例如投票法、Bagging、Boosting 或随机森林等),因此需要研究人员面对该问题,去针对性地设计出更加新颖有效的集成学习方法.

3.1.7 考虑类不平衡问题的方法

类不平衡问题[44]是软件缺陷预测数据集内的常见问题,其产生的根源是软件缺陷在被测项目内的分布大致符合帕累托原则,即 20%的程序模块内会含有大概 80%的缺陷.因此在缺陷预测数据集内,有缺陷模块(少数类)的数量要远远少于无缺陷模块(多数类)的数量,并造成模型对少数类的预测精度较低.目前已有的类不平衡方法[44]可以简单分为两类:(1)代价敏感的学习方法,该方法为不同类型的预测错误设置不同的代价;(2)采样方法,包括随机欠采样、随机过采样和 SMOTE 方法[43]等.

Fukushima 等人[13-14]在训练集中借助随机欠采样方法来减少多数类实例的数量,从而使得多数类实例的数量与少数类实例的数量保持一致.

Ryu 等人[45-46]针对跨项目缺陷预测中的类不平衡问题,提出了 VCB-SVM(Value-Cognitive Boosting with Support Vector Machine)方法.该方法首先基于目标项目的数据集,依次计算出源项目中每个实例的相似性权重,随后基于支持向量机,使用 boosting 方法来构建缺陷预测模型.随后 Ryu 等人[47]进一步提出了 HISNN(Hybrid Instance Selection using Nearest-Neighbor)方法.他们首先借助海明距离(Hamming distance)来计算目标项目实例和源项目实例之间的相似度.随后针对每个目标项目实例,首先计算出其最小海明距离 Min-Ham,即以该目标实例为圆心,Min-Ham 为距离的圆内,至少会含有一个源项目实例.随后则可能存在 3 种情况:(1)在圆内,仅有一个源项目实例;(2)在圆内,含有多个源项目实例,且均属于同一类型;(3)在圆内,含有多个源项目实例,但并不属于同一类型.其中情况 2 可以借助局部知识来直接决定目标项目实例的类型,而情况 1 和 3 则需要进一步借助全局知识来进行类型标记.

3.1.8 基于其他类型的方法

Wang 等人[48]考虑了表示学习(representation learning)算法,他们认为基于表示学习分析出的语义特征,可以更好的捕获缺陷的共有特征,并用于 CPDP.他们提出了 DBN-CP 方法,给定源项目和目标项目,DBN-CP 基于源项目训练深度信念网络(deep belief network),并生成同时针对源项目和目标项目的语义特征,随后训练出 ADTree,并用于预测目标项目.

Canfora 等人[49-50]将跨项目缺陷预测问题建模为多目标优化问题,他们主要考虑了两个可能存在冲突的优化目标:尽可能多地识别出被测项目内的缺陷数和尽可能少的代码审查量.这两个目标存在明显的冲突,若需要识别出更多的缺陷,一般需要审查更多的代码.他们提出了 MODEP 方法,该方法借助一种经典的多目标遗传算法(NSGA-II),针对 Logistic 回归或决策树模型中的参数取值生成 Pareto 前端(Pareto front).

3.2　异构跨项目缺陷预测方法

上述研究工作适用于对采用相同度量元的软件项目进行跨项目软件缺陷预测，但不同类型的项目间采用的度量元可能并不相同. 例如 Jing 等人[51]分析了软件缺陷预测研究中经常使用的 4 组数据集：NASA[52]、Softlab、Relink[37]和 AEEEM[38]. 他们发现：NASA 与 Softlab 这两组数据集中共有的度量元数有 28 个，NASA 与 Relink、Softlab 与 Relink 之间共有的度量元数仅有 3 个，而其他任意两组数据集之间则没有相同的度量元. 研究人员将上述问题称为异构跨项目软件缺陷预测.

针对上述问题，一种简单的方法是在构建 CPDP 模型的时候，仅考虑源项目与目标项目间共有的度量元. 例如 Turhan 等人[23]在进行 CPDP 时，考虑了 NASA 和 Softlab 这两个数据集共同采用的度量元. 但该方法存在如下不足，一方面项目间共有的度量元普遍很少，甚至有时候没有. 其次仅考虑共有的度量元，会遗漏大量有用的信息. 随后研究人员针对该问题提出了多种有效的解决方法.

He 和 Ma 等人[53]提出了 CPDP-IFS 方法，他们将每个实例视为一个向量，并计算其分布特征取值. 该方法的研究动机是：一个实例，如果其度量元取值均处于正常取值范围之内，则其含有缺陷的可能性会较低. 而如果该实例的一些分布特征取值存在异常，则其含有缺陷的可能性会更高. 基于上述观察，他们将目标项目和源项目中的实例投射到一个潜在空间，该空间由实例的分布特征指标构成. 因此该方法可以确保 CPDP 在同一特征空间内运行. 在实证研究中，他们重点考察了 median、mean、min、max 和方差这 5 种指标.

Nam 和 Kim[54]提出了 HDP(Heterogeneous Defect Prediction)方法. 该方法包括特征选择和特征映射两个阶段. 具体来说：首先借助特征选择方法从源项目中选出与类标强相关的特征. 随后借助特征映射方法将为源项目选出的特征与目标项目的特征进行映射. 最后基于映射的特征构建缺陷预测模型. 其中特征选择方法通过移除数据集中的无关特征(即度量元)和冗余特征来保留含有有用信息的特征，他们主要考虑了基于增益比(gain ratio)、Chi-square、Relief-F 等的特征选择方法. 在特征映射的时候则考虑了 3 种度量元相似度匹配方法.

Jing 等人[51]针对该问题，首先提出一种 UMR (Unified Metric Representation)表示. 该表示包含 3 种不同类型的度量元：源项目与目标项目共有的度量元、源项目特有的度量元和目标项目特有的度量元. 为源项目构建 UMR 表示时，将目标项目特有的度量元取值设置为空. 而为目标项目构建 UMR 表示时，将源项目特有的度量元取值设置为空，具体如图 7 所示.

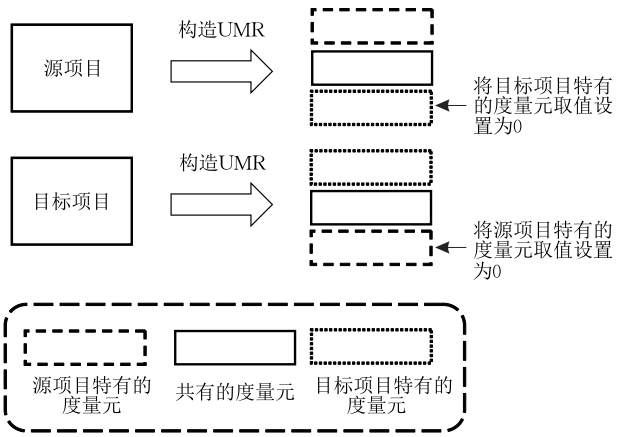


图 7　分别为源项目和目标项目构造 UMR

随后基于上述 UMR 表示，他们借助典型相关分析(Canonical Correlation Analysis，CCA)方法，来减少源项目与目标项目数据集分布间的差异程度. CCA 通过为源项目和目标项目寻找一个共有空间，使得投影到该共有空间的两个数据集间的相关性最大化. 结果表明：若源项目与目标项目之间共有的度量元数越多，该方法的性能越好. 同时他们也发现多对 1 的方式要优于 1 对 1 的方式.

针对该问题的研究有助于更加充分的利用研究人员已经搜集到的高质量缺陷预测数据集. 但目前与之相关的研究工作还较少，并且实证研究结论是否具有一般性还需要在更多数据集上进行验证. 除此之外，一些方法能够取得好的预测性能是基于一定的假设，例如 Nam 和 Kim[54]提出的 HDP 方法，需要源项目与目标项目成功映射后的特征的取值具有一致的缺陷倾向性(defect-prone tendency)，但由于目标项目中实例是否含有缺陷是未知的，因此我们无法提前获知目标项目中特征的缺陷倾向性，最终造成该方法在一些情况下，预测性能并不理想.

3.3　常用分类方法总结

该节对这类方法中常用的分类方法进行总结. 表 2 统计了已有研究中经常使用的分类方法(即使用次数超过 1 次)，对累计使用次数和首次使用时间(包括对应的参考文献)进行了统计，并按照累计使用次数从大到小进行排序. 从表 2 中可以看出，目前研究人员在实证研究中使用最多的前 5 种

分类方法是:Logistic 回归、朴素贝叶斯、决策树、支持向量机和随机森林.除此之外,也有一些研究人员会使用到集成学习、决策表(decision table)、贝叶斯网络(Bayesian network)、K 近邻、多层感知器(multi-layer perceptron)和 RBF 网络(Radial Basis Function network).

表 2 分类方法使用统计

分类方法	累计使用次数	首次使用时间
Logistic 回归	19	2002[8]
朴素贝叶斯	17	2009[23]
决策树	11	2008[22]
支持向量机	8	2012[10]
随机森林	7	2013[29-31]
集成学习	6	2010[40]
决策表	5	2012[10]
K 近邻	4	2013[30]
贝叶斯网络	4	2013[29]
多层感知器	3	2013[29]
RBF 网络	2	2014[18]

在软件缺陷预测研究中,不同分类方法的性能是否存在显著性差异是一个争议性问题.Lessmann 等人[55]早期基于 NASA 数据集,系统地比较了 22 种不同的分类方法的预测性能(基于 AUC 值),他们发现最优的 17 种分类方法间的性能差异并不显著.随后 Ghotra 等人[56]在 2015 年,同样基于 Lessmann 等人使用的 NASA 数据集,并考虑了更多新颖的分类方法.他们对 Lessmann 等人的实证研究[55]进行了再现,并得出了相同的结论.但是在来自 Promise 库的 10 个开源项目数据集上和去除噪音的 NASA 数据集[57]上,他们发现不同的分类方法之间的确存在显著的性能差异.Tantithamthavorn 等人[58]随后在 2016 年,进一步分析了分类方法中的参数取值优化对缺陷预测性能的影响,结果表明这种影响不可忽略.基于上述研究工作,我们认为在 CPDP 的后续研究中,研究人员也需要深入地分析不同分类方法和具体分类方法中的参数取值对 CPDP 性能的影响,而不是仅仅将分类方法中的参数取值设置为默认值.

4 基于无监督学习的方法

虽然基于有监督学习的跨项目缺陷预测方法可以有效提高 CPDP 的性能,但在实际情况中,源项目与目标项目数据集间的分布差异仍然很大.因此一些研究人员尝试通过仅关注目标项目中的未标记数据,借助无监督学习方法来缓解该问题.

Zhong 等人[59]借助聚类方法(k-means 和 Neural-Gas),对程序模块进行聚类.随后他们从每一个簇中选出典型模块,并将一些辅助统计信息(例如度量元的均值、最大值、最小值、中位数等)提供给专家,最终由专家完成簇的标记.Catal 等人[60]提出一种基于度量元阈值的方法,当实例中任何一个度量元的取值高于指定阈值时,则认为该实例是缺陷实例.他们在阈值设定时主要基于专家知识或对已有文献的分析.Bishnu 和 Bhattacherjee[61]在 k-means 聚类时,借助四叉树(quad trees)来初始化簇类中心.Abaei 等人[62]则使用了基于 SOM(Self-Organizing Maps)和给定阈值的无监督学习方法.Yang 等人[63]则尝试使用近邻传播聚类算法(affinity propagation clustering algorithm).但上述方法在聚类标记或阈值设定时,需要依赖专家知识.

Nam 和 Kim[64]提出了一种自动方法 CLA 和 CLAMI.其中 CLA 方法包含两个阶段,在阶段 1,首先依次计算每个度量元的中位数,随后对每个实例,依次对高于中位数的度量元(即异常度量元)进行标记,并计算出每个实例含有的异常度量元数 K.最后将具有相同 K 值的实例归入到同一聚类内.在阶段 2,将具有更大 K 值的簇类实例标记为有缺陷模块,而将具有更小 K 值的簇类实例标记为无缺陷模块.他们随后通过进一步引入特征选择和实例选择这两个阶段来移除数据集中的噪声,并提出了 CLAMI 方法.

Zhang 等人[65]则借助谱聚类(spectral clustering)方法.传统的基于距离的聚类方法,一般借助欧氏距离完成对数据集中实例的划分.而谱聚类则基于实体间的连通性完成对数据集的划分.针对该问题,可以将模块设置为实体,而实体间的联系强度,则可以通过度量元取值的相似度来确定.随后他们借助谱聚类方法,将所有程序模块划分到两个聚类.在聚类标记时,对于每一个聚类,首先计算出聚类内每个实例的所有度量元值的总和 rs,随后计算出聚类中所有实例的 rs 值的均值 ars.最后将具有更高 ars 值的聚类内的所有模块标记为有缺陷模块,而将另一个聚类内的所有模块标记为无缺陷模块.

有时候由于源项目和目标项目的考虑的开发流程和编程语言、开发人员的经验、项目在软件度量时考虑的度量元等存在较大的差异性,造成基于有监督学习的方法难以取得令人满意的性能.在这种场景下,无监督学习方法由于不需要训练数据,因此可以有效避免上述问题.这类方法中的大部分研究工

作，一般借助聚类分析技术来将程序模块划分到不同的簇内. 在对不同簇内的模块进行标记时，一般基于如下假设：在软件缺陷预测问题中，有缺陷模块的度量元取值存在高于无缺陷模块的度量元取值的倾向. 因此如果需要预测的目标项目，其使用的度量元不满足上述假设，则这类方法的有效性将无法得到保障.

5　基于半监督学习的方法

这类方法会综合考虑大量已标记的其他源项目数据和少量已标记的目标项目数据.

Turhan 等人[66]从综合使用目标项目数据和源项目数据角度出发，对该问题进行了研究. 他们通过实证研究发现，如果目标项目的训练数据比较充足，则可以不考虑源项目的训练数据，但如果目标项目中仅有少量训练数据，则综合使用一些源项目的数据来构建缺陷预测模型是比较合理的，因此他们建议在项目的早期开发阶段，可以考虑这种混合方法.

一些研究工作借助集成学习的方法. Ryu 等人[67]提出一种 TCSBoost(Transfer Cost-Sensitive Boosting)方法，该方法首先将源项目中的实例和目标项目中的少数已标记实例汇总为训练集，随后迭代 M 次，每次基于训练集中的实例权重构建一个基分类器，随后根据该基分类器在训练集上的预测结果来更新实例的权重，其权重更新公式不仅考虑了实例间的相似度也考虑了不同预测错误类型的开销. 最后他们将这 M 个基分类器进行加权集成学习，其中每个基分类器的权重由该基分类器在训练集上的预测性能决定.

Xia 等人[68]提出了 HYDRA 方法，该方法包括两个阶段：遗传算法阶段和集成学习阶段. 在遗传算法阶段，该方法会基于多个候选源项目的数据 $\{s_1, s_2, \cdots, s_n\}$ 和少量已标记目标项目数据 t_l，构建 $N+1$ 个分类器，并借助遗传算法来搜索这些分类器的最优权重取值并输出一个组合分类器(即 GA 分类器). 在集成学习阶段，该方法会对第一阶段迭代多次，并生成大量 GA 分类器，借助 AdaBoost 方法，基于上述 GA 分类器构建出最终的缺陷预测模型.

也有一些研究工作[69-70]基于 Dai 等人提出的 TrAdaBoost 方法[71]，该方法受 AdaBoost 方法的启发，在每一轮迭代时，根据源项目数据和目标项目已标记实例 t^l，以及这些实例的权重来构建模型，并根据模型在源项目实例中的预测结果来更新相应权重，即若预测错误，则认为该实例与 t^l 在数据分布上存在较大的差异并减小其权重取值，否则若预测正确，则认为该实例与 t^l 在数据分布上存在较小的差异并增加其权重取值.

Chen 和 Fang 等人[69]从减少源项目内的负面实例(negative instances)权重角度出发，这类实例与目标项目的实例在数据分布上存在较大的差异. 他们首先对候选源项目的训练数据，借助 Burak 过滤法[23]和 SMOTE 类不平衡方法[43]进行预处理. 随后他们提出了 DTB(Double Transfer Boosting)方法，该方法有两个阶段：首先借助 DGM(Data Gravitation Method)方法[34]，通过分析目标项目已标记实例 t^l 的特征来确定源项目实例的权重，即与 t^l 的相似度越高，其实例权重取值也越高，随后借助 TrAdaBoost 方法[71]来进一步减小源项目中的负面实例的权重. 该方法通过 DGM 方法可以为随后 TrAdaBoost 方法提供更好的训练实例权重初始值.

但上述方法基于 1 对 1 方式，因此其 CPDP 性能与选择的源项目密切相关. 而沈备军等人[70]则考虑了多对 1 方式，并提出了两种基于 TrAdaBoost[71]的改进方法. 第一种方法 MergeTrAdaBoost 借助聚类分析，构造出与目标项目实例数相当，且关联性高的候选源项目训练集，随后执行 TrAdaBoost 方法. 第二种方法 MultiTrAdaBoost 则首先在每个候选源项目上执行 TrAdaBoost 方法，随后借助集成学习来构造出最终的预测模型.

由于为目标项目中的实例进行标记的代价较高且容易出错，因此这种方法通过尝试为目标项目中的少数实例进行标记，以提高 CPDP 的性能. 目前一部分研究工作[67-68]基于集成学习方法，通过分析基分类器的预测结果来调整基分类器的权重. 而另一部分研究工作[69-70]则基于 TrAdaBoost 方法[71]，在每次迭代时，基于该轮分类器的预测结果来尝试不断降低下一次迭代时负面实例的权重取值. 基于半监督学习的方法在近两年逐渐得到研究人员的关注，但研究成果不多，因此仍有一定的性能提升空间.

6　性能评测指标分析

绝大部分的已有研究工作将跨项目缺陷预测问题视为二分类问题. 若将有缺陷模块设为正例，无缺

陷模块设置为反例，则可以将程序模块根据其真实类型与模型的预测类型的组合划分为真正例(True Positive)、假正例(False Positive)、真反例(True Negative)和假反例(False Negative)这4种情形. 令 TP、FP、TN、FN 分别表示对应的模块数，则分类结果的混淆矩阵(confusion matrix)可如表3所示.

表3 混淆矩阵

真实类型	预测类型	
	有缺陷模块	无缺陷模块
有缺陷模块	TP	FN
无缺陷模块	FP	TN

基于上述混淆矩阵，我们将已有研究工作中常用的模型性能评测指标总结如下：

(1) 准确率. 该指标返回的是正确预测的程序模块占所有程序模块的比例，其计算公式为

$$accuracy=\frac{TP+TN}{TP+FN+FP+TN} \tag{6}$$

(2) 查准率. 该指标返回的是预测为有缺陷的模块中，真实缺陷模块所占的比例，其计算公式为

$$precision=\frac{TP}{TP+FP} \tag{7}$$

(3) 查全率. 该指标在一些文献中又被称为 pd (probability of detection)，其返回的是所有缺陷模块中，被预测为有缺陷模块所占的比例，其计算公式为

$$recall=\frac{TP}{TP+FN} \tag{8}$$

(4) pf (probability of false alarm). 该指标返回的是所有无缺陷模块中，被预测为有缺陷模块所占的比例. 若 pf 取值过高，则意味着需要投入更多资源到无缺陷模块的代码审查或单元测试，因此会造成测试资源的浪费. 其计算公式为

$$pf=\frac{FP}{TN+FP} \tag{9}$$

(5) $f\text{-}measure$. 该指标是查准率和查全率的调和平均数，可以对查准率和查全率这两个指标进行有效的平衡，其计算公式为

$$f\text{-}measure=\frac{2\times precision\times recall}{precision+recall} \tag{10}$$

(6) $g\text{-}measure$. 该指标则是 pd 和 $(1-pf)$ 的调和平均数，其计算公式为

$$g\text{-}measure=\frac{2\times pd\times(1-pf)}{pd+(1-pf)} \tag{11}$$

(7) $balance$. 该指标由 Menzies 等人提出[52]，计算的是点 (pd,pf) 到点 $(1,0)$ 的欧氏距离，其计算公式为

$$balance=1-\frac{\sqrt{(0-pf)^2+(1-pd)^2}}{\sqrt{2}} \tag{12}$$

(8) $g\text{-}mean$，该指标主要用于类不平衡学习场景，其计算公式为

$$g\text{-}mean=\sqrt{pd\times(1-pf)} \tag{13}$$

(9) MCC，该指标被一些研究人员[51,69]使用，其计算公式为

$$MCC=\frac{TP\times TN-FN\times FP}{\sqrt{(TP+FN)\times(TP+FP)\times(FN+TN)\times(FP+TN)}} \tag{14}$$

不难看出，MCC 指标的取值范围介于 -1 到 1 之间，其取值越大越好.

上述指标的计算与阈值的设定有关，即如果缺陷预测模型对新程序模块的预测值高于该阈值时，则该模块被预测为有缺陷模块，否则被预测为无缺陷模块. 因此上述指标的取值与实际阈值的设定密切有关. 而 AUC(Area Under ROC Curve)指标的计算则与阈值设定无关，该指标主要基于 ROC 曲线. ROC 曲线的全称是受试者工作特征(receiver operation characteristic)曲线. 它源于二战中用于敌机检测的雷达信号分析技术，随后被引入机器学习领域. 其横轴为假正例率(False Positive Rate，FPR)，纵轴为真正例率(True Positive Rate，TPR). 所有曲线都通过点(0,0)和点(1,1). 最优模型对应的 ROC 曲线接近于直线 $y=1$，而随机模型对应的 ROC 曲线为直线 $y=x$. AUC 值对应的是 ROC 曲线下的面积，可用于模型的性能评估. AUC 的取值范围介于 0 到 1 之间，取值越高表示模型的性能越好.

上述性能指标均假设测试人员可以审查完所有可疑程序模块. 但在实际的软件测试场景中，在测试成本受限的情况下，可以测试完的程序模块数是有限的.

Arisholm 等人[72]提出了新的评测指标 AUCEC (Area Under the Cost-Effectiveness Curve). 该指标主要基于成本收益曲线，其中横轴表示可以审查的代码比例，纵轴表示可以检测出的缺陷比例. 在曲线绘制时，需要基于模型的预测结果，将所有程序模块按照含有缺陷的概率，从高到低进行排序. AUCEC 则是成本收益曲线下的面积. 图8中的上

边子图分别表示了最优模型 O、一般模型 P 和随机模型 R. 除此之外，也可以事先指定可以审查的代码比例. 但不同的代码审查比例在一些场景下会得到相互矛盾的结论，图 8 中的下子图给出了两个不同模型 $P1$ 和 $P2$ 的成本收益曲线. 若仅考虑 20% 的代码审查比例，我们会发现模型 $P2$ 的性能要优于模型 $P1$ 的性能，但若考虑 60% 的代码审查比例，则我们会发现模型 $P1$ 的性能要优于模型 $P2$ 的性能. 由于缺陷在项目中的分布大致符合帕累托原则，即 80% 的缺陷集中分布于 20% 的代码内，因此 Xia 等人[41,68]在他们的研究工作中将代码审查比例设置为 20%，并将该指标称为 PofB20 指标.

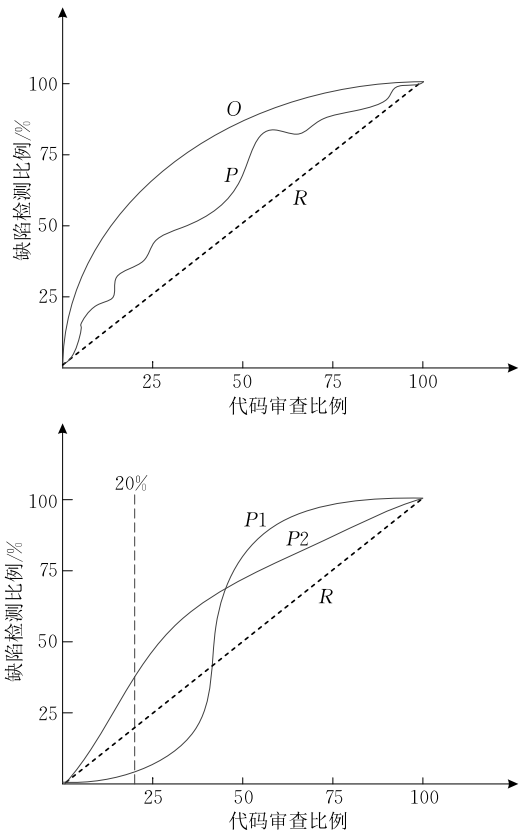


图 8　成本收益曲线

论文对已有 CPDP 研究中常用的性能评测指标的累计使用次数进行了统计，并从大到小进行排序，同时给出了每个指标的首次使用时间（包括对应的参考文献），最终结果如表 4 所示，从表中不难看出经常使用的评测指标是：查准率、查全率、*f-measure*、*pf* 以及 AUC. 其中 AUC 指标由于在计算时与阈值的设定无关，因此近些年来被更多的研究人员所使用. 除此之外，我们还发现，一些研究人员逐渐开始从成本收益角度来评估模型的预测性能，主要考虑的指标包括 AUCEC 和 PofB20.

表 4　性能评测指标统计

指标	累计使用次数	首次使用时间
查全率	32	2002[8]
pf	19	2004[59]
f-measure	20	2012[10,12,34]
查准率	17	2002[8]
AUC	16	2012[12]
g-measure	9	2013[31]
准确率	7	2004[59]
balance	4	2009[23]
MCC	4	2014[73]
AUCEC	2	2012[12]
PofB20	2	2015[41]
g-mean	1	2016[67]

7　评测数据集分析

这一节对 CPDP 研究中经常使用的评测数据集的累计使用次数和首次使用时间进行了总结，最终结果如表 5 所示（表 5 仅列出累计使用次数超过 5 次的评测数据集）.

表 5　评测数据集的使用情况统计

数据集名称	累计使用次数	首次使用时间
PROMISE	24	2010
NASA	11	2004
SoftLab	10	2009
AEEEM	6	2013
Relink	6	2013

早期由于数据集并未共享，造成相关研究工作难以重现，因此相关论文数较少. 例如 Briand 等人最早进行跨项目缺陷预测可行性分析时，使用的是两个中等规模的 Java 项目 Xpose 和 Jwriter. 但这两个数据集并未进行共享. 从 2009 年开始，随着 NASA 和 Softlab 数据集的共享，对 CPDP 问题的研究开始逐步升温. 尤其在 2010 年以后，研究人员通过挖掘开源软件又陆续共享了 Promise、Relink 和 AEEEM 数据集，最终使得 CPDP 问题成为当前软件缺陷预测领域的一个研究热点，并产生了很多高质量的研究成果. 接下来，论文将对这些数据集的特征进行简要介绍. 目前这些数据集均可以在 PRMISE 库（http://openscience.us/repo）中进行下载.

NASA 数据集来自美国国家航空和宇宙航行局（NASA）的项目，其中项目类型比较多样，与卫星仪器、地面控制系统、飞行控制模块相关. 而 Softlab 数据集来自土耳其一个家电制造商的项目，这些项目与洗衣机、洗碗机和冰箱的控制软件有关. 这些项目在度量元的选择时，主要关注代码的复杂度，其度

量元与模块的代码行数、Halstead 复杂度、McCabe 环路复杂度等有关. Shepperd 等人[57]随后发现之前共享的 NASA 数据集中存在诸多质量问题,例如部分数据的取值不一致或取值存在遗失,部分实例重复等. 因此他们对 NASA 数据集进行了清理,并建议随后的研究人员在实证研究中考虑使用已经数据清理后的 NASA 数据集.

PROMISE 数据集由 Jureczko 和 Madeyski[24]搜集. 该数据集来自 10 个不同开源项目(例如 ant、log4j、lucene、poi 等)的多个版本. 程序模块的粒度设置为类,考虑了 20 种度量元,这些度量元均关注的是代码复杂度,其中包括由 Chidamber 和 Kemerer[74]提出的 CK 度量元. 这些度量元综合考虑了面向对象程序固有的封装、继承、多态等特性.

AEEEM 数据集由 D'Ambros 等人[38]搜集,他们分析的项目包括 Eclipse JDT Core、Eclipse PDE UI、Equinox framework、Mylyn 和 Apache Lucene. 其考虑了 61 种度量元,这些度量元不仅考虑了软件代码的复杂度,也考虑了软件开发过程. 重点分析了代码修改特征、历史缺陷检测信息、模块复杂度等.

Relink 数据集由 Wu 等人[37]搜集,他们分析的项目是 Apache HTTP Server、Safe 和 ZXing. 该数据集共考虑了 26 种度量元,这些度量元均关注的是代码复杂度,并借助 Understand 工具进行搜集. 他们随后通过手工验证和校对等方式进一步提高了数据集的质量.

8 总结与展望

一方面对于一些新启动的软件项目或大部分中小规模企业来说,搜集充足的高质量缺陷预测数据集较为困难. 这也是目前软件缺陷预测研究成果难以成功应用到企业软件质量保障流程的主要阻碍之一,另一方面研究人员已经通过挖掘开源项目,搜集了很多高质量的缺陷预测数据集,并共享到 Promise 库中. 因此针对跨项目软件缺陷预测问题的研究具有丰富的理论研究价值和工业界应用前景,并日益得到了学术界和工业界的关注. 虽然国内外研究人员已经取得了一定的研究进展,但我们认为,CPDP 仍然是今后软件缺陷预测研究中值得关注的一个开放性研究课题,其还存在很多值得国内研究人员关注的研究问题. 论文将依次从数据集的搜集、数据集的预处理、模型的构建和评估以及模型的应用这 4 个维度对未来研究工作进行展望.

(1)针对数据集搜集维度的研究展望

在评测数据集的总结中,我们发现目前绝大部分实证研究选择的数据集都集中于 PROMISE、NASA、Softlab、AEEEM 和 Relink,其研究结论具有一定的偏向性. 因此我们需要通过多种渠道来搜集更多的数据集,来对已有研究结果的一般性进行验证.

一方面随着开源项目的蓬勃发展,研究人员可以从一些开源项目的托管网站(例如 Google Code、SourceForge 或 GitHub 等)上搜集大量开源项目的历史开发数据. 另一方面,研究人员也可以尝试与企业进行合作来搜集数据集.

从开源社区获取项目数据相对容易,但从工业界中获取数据则比较困难,因为数据集中一般含有商业敏感信息,并容易被攻击者获取. 因此数据所有者在共享数据集的时候,希望能够不违反任何数据隐私法律或公司隐私政策. 但数据集的隐私保护和数据集的可用性存在一定的折衷. 即加强数据集的隐私保护可能会降低数据集的可用性,即减弱数据集的跨项目缺陷预测效果. 因此针对该问题的研究挑战是:首先防止共享数据集中的一些敏感度量元取值不会被泄露,其次是对数据集进行隐私保护后,仍能够有效的进行 CPDP. Peters 等人[75-77]在这方面进行了尝试,并提出了一系列数据集隐私保护方法. 其中 CLIFF 方法尝试找出具有很好类型区分能力的度量元取值子区间. 他们认为若模块含有这类取值子区间的数量较少,则这些模块构建模型的时候用处不大,并将这些模块进行移除. MORPH 方法则通过对实例的特征取值尝试进行随机搅动,使得该实例一方面在 n 维空间中可以移动到一个新的位置,另一方面又可以确保上述移动不会跨越不同类间的边界. 随后他们进一步提出 LACE2,可以同时对多个数据集进行隐私保护.

除此之外,开源项目和商业项目的开发特征存在很大的差异性[9],例如开源项目一般采用分布式的开发方式,而商业项目一般采用集中式的开发方式. 因此基于开源项目构建的缺陷预测模型,是否适用于工业界的商业项目是一个值得关注的研究问题. 但由于目前共享的商业项目数据集很少,因此对该问题的研究工作较少. 若能共享更多的商业项目数据集,则有助于研究人员对该问题进行更为深入的研究.

(2)针对数据集预处理维度的研究展望

对数据集进行必要的预处理(例如度量元取值

变换等)，有助于提高随后构建的 CPDP 模型的性能.我们可以尝试从如下两个方面进行探索.

一方面可以分析缺陷预测数据集中的噪音对 CPDP 的影响.在缺陷预测数据集的搜集过程中，程序模块进行类型标记和软件度量时均有可能引入噪音[78-79].针对上述问题，首先需要使用已有的方法[37,80]，来尝试尽可能多的识别并移除源项目和目标项目数据集中的噪音.其次像传统软件缺陷预测研究一样[81-82]，需要分析噪音的存在对已有 CPDP 方法性能的影响.最后针对数据集内的噪音难以避免的问题，有必要设计出具有一定噪音抗干扰能力的跨项目缺陷预测方法.

另一方面可以针对 CPDP 问题设计新颖特征选择方法.在传统软件缺陷预测的研究中，特征选择方法尝试识别并移除数据集中的冗余特征和无关特征，因此可以提高缺陷预测模型的性能[83-85].针对 CPDP 问题，在特征选择的时候，则需要同时考虑源项目和目标项目间的数据分布差异并提出新颖的解决方案.

(3)针对模型构建和评估维度的研究展望

除了进一步引入迁移学习领域内的最新研究成果，在模型构建上还可以通过对通用缺陷预测模型的研究来缓解 CPDP 问题.对通用缺陷预测模型的深入研究，有利于更好地分析度量元与程序模块缺陷间的相关性，同时可以为新项目训练数据的搜集提供指南.Zhang 等人[73,86]在这个方面进行了尝试，他们选择了 32 种度量元，其中 5 种与软件开发过程相关，21 种与软件代码相关，剩余 6 种与项目的上下文相关.但不同项目的度量元取值分布存在较大的差异性，因此他们提出了一种新颖的数据预处理方法.具体来说：首先，根据项目上下文相关度量元取值，将所有的项目进行划分；其次，基于度量元取值分布的相似性，将这些分组进行聚类分析；最后，对于每个聚类，他们推导出对应的转化函数，借助该函数可以使得转化后的度量元取值处于同一量纲中.

除此之外，我们可以吸收另一个相似的研究领域(即软件工作量估算)的最新研究成果.软件工作量估算(Software Effort Estimation，SEE)是软件工程数据挖掘领域中的另一个重要研究问题.SEE 主要用于估计软件项目开发中所需的工作量，其也存在与软件缺陷预测相同的问题，即一个项目或企业不存在历史数据，甚至同一项目，不同时间的数据分布也不一样.针对 SEE 问题的研究要早于 CPDP 问题，因此可以积极吸收 SEE 领域的已有研究成果[87-90]到 CPDP 问题中，同时也可以将 CPDP 的研究成果应用于 SEE 问题中.

虽然针对 CPDP 的研究，产生了很多研究成果.但一些研究工作的结果缺乏收敛，甚至有的研究工作的实证研究结论存在相互矛盾.其产生的根源在于不同研究工作的 CPDP 方法的实现方式(例如有的基于 Weka 软件包、有的基于 R 语言软件包、有的基于 Python 语言的 scikit-learn 库)、采用的评测程序、使用的评测指标或模型性能的评估方法[91]都不完全一样.为了缓解该问题，一方面需要研究人员对论文使用的评测数据集和相关代码进行共享，以利于实证研究结果的重现.另一方面也有必要开发统一的实验平台，以方便不同方法之间的互相比较.Herbold[92]在这方面进行了尝试，并开发出了开源工具 CrossPare①.

(4)针对模型应用维度的研究展望

Lewis 等人通过与 Google 工程师的交流，他们发现一方面搜集缺陷预测数据集的代价比较高昂，另一方面企业也缺乏必要的资源和技术专家来完成数据集的搜集[93]，因此针对 CPDP 问题的深入研究，无疑可以充分利用一些已经共享的高质量缺陷预测数据集，从而进一步促进软件缺陷预测的研究成果应用到工业界的实践中，并为学术界提出更多新的研究挑战.

若能对上述研究挑战提出有效的解决方案，将使得软件缺陷预测可以更早地用于项目开发的早期阶段，并有助于更为精准地预测出被测项目内的缺陷程序模块，从而有效提高测试资源的分配效率，最终提高软件产品的质量.

致　谢　论文作者衷心感谢匿名审稿专家提出的宝贵审稿意见，可以有效提高这篇综述论文的质量!

参　考　文　献

[1] Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 2012, 38(6): 1276-1304

[2] Wang Qing, Wu Shu-Jian, Li Ming-Shu. Software defect prediction. Journal of Software, 2008, 19(7): 1565-1580(in Chinese)

① https://crosspare.informatik.uni-goettingen.de/

(王青,伍书剑,李明树. 软件缺陷预测技术. 软件学报,2008,19(7):1565-1580)

[3] Chen Xiang, Gu Qing, Liu Wang-Shu, et al. Survey of static software defect prediction. Journal of Software, 2016, 27(1): 1-25(in Chinese)
(陈翔,顾庆,刘望舒等. 静态软件缺陷预测方法研究. 软件学报,2016,27(1):1-25)

[4] Yu Shu-Si, Zhou Shui-Geng, Guan Ji-Hong. Software engineering data mining: A survey. Journal of Frontiers of Computer Science and Technology, 2012, 6(1): 1-31(in Chinese)
(郁抒思,周水庚,关佶红. 软件工程数据挖掘研究进展. 计算机科学与探索,2012,6(1):1-31)

[5] Radjenovic D, Hericko M, Torkar R, Zivkovic A. Software fault prediction metrics: A systematic literature review. Information and Software Technology, 2013, 55(8): 1397-1418

[6] Pan S J, Yang Q. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(10): 1345-1359

[7] Zhuang Fu-Zhen, Luo Ping, He Qing, Shi Zhong-Zhi. Survey on transfer learning research. Journal of Software, 2015, 26(1): 26-39(in Chinese)
(庄福振,罗平,何清,史忠植. 迁移学习研究进展. 软件学报,2015,26(1):26-39)

[8] Briand L, Melo W, Wust J. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Transactions on Software Engineering, 2002, 28(7): 706-720

[9] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process//Proceedings of the Joint Meeting of the European Software Engineering Conference and the International Symposium on the Foundations of Software Engineering. Amsterdam, Netherland, 2009: 91-100

[10] He Z, Shu F, Yang Y, et al. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2012, 19(2): 167-199

[11] Jureczko M, Madeyski L. Cross-project defect prediction with respect to code ownership model: An empirical study. e-Informatica Software Engineering Journal, 2015, 9(1): 21-35

[12] Rahman F, Posnett D, Devanbu P. Recalling the "Imprecision" of cross-project defect prediction//Proceedings of the International Symposium on the Foundations of Software Engineering. North Carolina, USA, 2012: 61:1-61:11

[13] Fukushima T, Kamei Y, McIntosh S, et al. An empirical study of just-in-time defect prediction using cross-project models//Proceedings of the Working Conference on Mining Software Repositories. Hyderabad, India, 2014: 172-181

[14] Kamei Y, Fukushima T, McIntosh S, et al. Studying just-in-time defect prediction using cross-project models. Empirical Software Engineering, 2016, 21(5): 2072-2106

[15] Kamei Y, Shihab E, Adams B, et al. A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering, 2013, 39(6): 757-773

[16] Turhan B. On the dataset shift problem in software engineering prediction models. Empirical Software Engineering, 2012, 17(1): 62-74

[17] Nam J, Pan S J, Kim S. Transfer defect learning//Proceedings of the International Conference on Software Engineering. San Francisco, USA, 2013: 382-391

[18] Panichella A, Oliveto R, De Lucia A. Cross-project defect prediction models: L'Union fait la force//Proceedings of the Conference on Software Maintenance, Reengineering and Reverse Engineering. Victoria, Canada, 2014: 164-173

[19] He P, Li B, Liu X, Chen J, Ma Y T. An empirical study on software defect prediction with a simplified metric set. Information and Software Technology, 2015, 59: 170-190

[20] Cruz A, Ochimizu K. Towards logistic regression models for predicting fault-prone code across software projects//Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Florida, USA, 2009: 460-463

[21] Zhang F. Towards Generalizing Defect Prediction Models [Ph. D. dissertation]. Queen's University, Kingston, USA, 2016

[22] Watanabe S, Kaiya H, Kaijiri K. Adapting a fault prediction model to allow inter languagereuse//Proceedings of the International Workshop on Predictor Models in Software Engineering. Leipzig, Germany, 2008: 19-24

[23] Turhan B, Menzies T, Bener A B, Stefano J D. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5): 540-578

[24] Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction//Proceedings of the International Conference on Predictive Models in Software Engineering. Timisoara, Romania, 2010: 9:1-9:10

[25] Menzies T, Butcher A, Marcus A, et al. Local vs. global models for effort estimation and defect prediction//Proceedings of the International Conference on Automated Software Engineering. Lawrence, USA, 2011: 343-351

[26] Menzies T, Butcher A, Cok D, et al. Local versus global lessons for defect prediction and effort estimation. IEEE Transactions on Software Engineering, 2013, 39(6): 822-834

[27] Bettenburg N, Nagappan M, Hassan A E. Think locally, act globally: Improving defect and effort prediction models//Proceedings of the Working Conference on Mining Software Repositories. Zurich, Switzerland, 2012: 60-69

[28] Bettenburg N, Nagappan M, Hassan A E. Towards improving statistical modeling of software engineering data: Think locally, act globally! Empirical SoftwareEngineering, 2015, 20(2): 294-335

[29] Herbold S. Training data selection for cross-project defect prediction//Proceedings of the International Conference on

Predictive Models in Software Engineering. Baltimore, USA, 2013: 6:1-6:10

[30] Peters F, Menzies T, Marcus A. Better cross company defect prediction//Proceedings of the Working Conference on Mining Software Repositories. San Francisco, USA, 2013: 409-418

[31] He Z M, Peters F, Menzies T, Yang Y. Learning from open-source projects: An empirical study on defect prediction// Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Baltimore, USA, 2013: 45-54

[32] He P, Li Bing, Zhang D G, Ma Y T. Simplification of training data for cross-project defect prediction. arXiv:1405.0773, ArXiv, 2014: 1-17

[33] Li Yong, Huang Zhi-Qiu, Wang Yong, Fang Bing-Wu. Novel approach of multi-source data driven cross-project defects prediction. Journal of Jilin University (Engineering and Technology Edition), 2016, 46(6): 2034-2041(in Chinese)
(李勇，黄志球，王勇，房丙午. 基于多源数据的跨项目软件缺陷预测方法. 吉林大学学报(工学版), 2016, 46(6): 2034-2041)

[34] Ma Y, Luo G, Zeng X, Chen A. Transfer learning for cross-company software defect prediction. Information and Software Technology, 2012, 54(3): 248-256

[35] Cheng Ming, Wu Guo-Qing, Yuan Meng-Ting. Transfer learning for software defect prediction. Acta Electronica Sinica, 2016, 44(1): 115-122(in Chinese)
(程铭，毋国庆，袁梦霆. 基于迁移学习的软件缺陷预测. 电子学报, 2016, 44(1): 115-122)

[36] Pan S J, Tsang I W, Kwok J T, Yang Q. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 2011, 22(2): 199-210

[37] Wu R, Zhang H, Kim S, Cheung S C. ReLink: Recovering links between bugs and changes//Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering. Szeged, Hungary, 2011: 15-25

[38] D'Ambros M, Lanza M, Robbes R. An extensive comparison of bug prediction approaches//Proceedings of the Working Conference on Mining Software Repositories. Cape Town, South Africa, 2010: 31-41

[39] Amasaki S, Kawata K, Yokogawa T. Improving cross-project defect prediction methods with data simplification// Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications. Madeira, Portugal, 2015: 96-103

[40] Liu Y, Khoshgoftaar T M, Seliya N. Evolutionary optimization of software quality modeling with multiple repositories. IEEE Transactions on Software Engineering, 2010, 36(6): 852-864

[41] Zhang Y, Lo D, Xia X, Sun J L. An empirical study of classifier combination for cross-project defect prediction// Proceedings of the Annual International Computers, Software and Applications Conference. Taichung, China, 2015: 264-269

[42] Dai Xiang, Mao Yu-Gang. Research on cross-company software defect prediction based on integrated sampling and ensemble learning. Journal of Chinese Computer Systems, 2015, 36(8): 1700-1705(in Chinese)
(戴翔，毛宇光. 跨机构的软件缺陷集成采样预测研究. 小型微型计算机系统, 2015, 36(8): 1700-1705)

[43] Chawla N V, Bowyer K W, Hall L O, Kegelmeyer W P. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 2002, 16: 321-357

[44] He H B, Garcia E A. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(9): 1263-1284

[45] Ryu D, Choi O, Baik J. Value-cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2014, 21(1): 43-71

[46] Ryu D, Choi O, Baik J. Improving prediction robustness of VAB-SVM for cross-project defect prediction//Proceedings of the International Conference on Computational Science and Engineering. Chengdu, China, 2014: 994-999

[47] Ryu D, Jang J I, Baik J. A hybrid instance selection using nearest-neighbor for cross-project defect prediction. Journal of Computer Science and Technology, 2015, 30(5): 969-980

[48] Wang S, Liu T, Tan L. Automatically learning semantic features for defect prediction//Proceedings of the International Conference on Software Engineering. Austin, USA, 2016: 297-308

[49] Canfora G, De Lucia A, Di Penta M, et al. Multi-objective cross-project defect prediction//Proceedings of the International Conference on Software Testing, Verification and Validation. Luxembourg, 2013: 252-261

[50] Canfora G, Lucia A D, Penta M D, et al. Defect prediction as a multiobjective optimization problem. Software Testing, Verification and Reliability, 2015, 25(4): 426-459

[51] Jing X, Wu F, Dong X, et al. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning//Proceedings of the Joint Meeting of the European Software Engineering Conference and the International Symposium on the Foundations of Software Engineering. Bergamo, Italy, 2015: 496-507

[52] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictor. IEEE Transactions on Software Engineering, 2007, 33(1): 2-13

[53] He P, Li Bing, Ma Y T. Towards cross-project defect prediction with imbalanced feature sets. arXiv:1411.4228, ArXiv, 2014: 1-10

[54] Nam J, Kim S. Heterogeneous defect prediction//Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering. Bergamo, Italy, 2015: 508-519

[55] Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 2008, 34(4): 485-496

[56] Ghotra B, McIntosh S, Hassan A E. Revisiting the impact of classification techniques on the performance of defect prediction models//Proceedings of the International Conference on Software Engineering. Florence, Italy, 2015: 789-800

[57] Shepperd M, Song Q B, Sun Z B, Mair C. Data quality: Some comments on the NASA software defect datasets. IEEE Transactions on Software Engineering, 2013, 39(9): 1208-1215

[58] Tantithamthavorn C, McIntosh S, Hassan A E, Matsumoto K. Automated parameter optimization of classification techniques for defect prediction models//Proceedings of the International Conference on Software Engineering. Austin, USA, 2016: 321-332

[59] Zhong S, Khoshgoftaar T, Seliya N. Unsupervised learning for expert-based software quality estimation//Proceedings of the International Symposium on High Assurance Systems Engineering. Tampa, USA, 2004: 149-155

[60] Catal C, Sevim U, Diri B. Clustering and metrics thresholds based software fault prediction of unlabeled program modules //Proceedings of the International Conference on Information Technology: New Generations. Las Vegas, USA, 2009: 199-204

[61] Bishnu P S, Bhattacherjee V. Software fault prediction using quad tree-based *k*-means clustering algorithm. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(6): 1146-1150

[62] Abaei G, Rezaei Z, Selamat A. Fault prediction by utilizing self-organizing map and threshold //Proceedings of the International Conference on Control System, Computing and Engineering. Penang, Malaysia, 2013: 465-470

[63] Yang B, Yin Q, Xu S, Guo P. Software quality prediction using affinity propagation algorithm//Proceedings of the International Joint Conference on Neural Networks. Hong Kong, China, 2008: 1891-1896

[64] Nam J, Kim S. CLAMI: Defect prediction on unlabeled datasets //Proceedings of the International Conference on Automated Software Engineering. Lincoln, USA, 2015: 452-463

[65] Zhang F, Zheng Q, Zou Y, Hassan A E. Cross-project defect prediction using a connectivity-based unsupervised classifier//Proceedings of the International Conference on Software Engineering. Austin, USA, 2016: 309-320

[66] Turhan B, Misirli A T, Bener A. Empirical evaluation of the effects of mixed project data on learning defect predictors. Information and Software Technology, 2013, 55(6): 1101-1118

[67] Ryu D, Jang J, Baik J. A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Quality Journal, 2017, 25(1): 235-272

[68] Xia X, Lo D, Pan S J, et al. HYDRA: Massively compositional model for cross-project defect prediction. IEEE Transactions on Software Engineering, 2016, 42(10): 977-998

[69] Chen L, Fang B, Shang Z W, Tang Y Y. Negative samples reduction in cross-company software defects prediction. Information and Software Technology, 2015, 62: 67-77

[70] Mao Fa-Gui, Li Bi-Wen, Shen Bei-Jun. Cross-project software defect prediction based on instance transfer. Journal of Frontiers of Computer Science and Technology, 2016, 10(1): 43-55(in Chinese)
(毛发贵, 李碧雯, 沈备军. 基于实例迁移的跨项目软件缺陷预测. 计算机科学与探索, 2016, 10(1): 43-55)

[71] Dai W Y, Yang Q, Xue G R, Yu Y. Boosting for transfer learning//Proceedings of the International Conference on Machine Learning. Corvalis, Oregon, USA, 2007: 193-200

[72] Arisholm E, Briand L C, Fuglerud M. Data mining techniques for building fault-proness models in telecom java software// Proceedings of the International Symposium on Software Reliability. Trollhättan, Sweden, 2007: 215-224

[73] Zhang F, Mockus A, Keivanloo I, Zou Y. Towards building a universal defect prediction model//Proceedings of the Working Conference on Mining Software Repositories. Hyderabad, India, 2014: 182-191

[74] Chidamber S R, Kemerer C F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493

[75] Peters F, Menzies T, Gong L, Zhang H Y. Balancing privacy and utility in cross-company defect prediction. IEEE Transactions on Software Engineering, 2013, 39(8): 1054-1068

[76] Peters F, Menzies T. Privacy and utility for defect prediction: Experiments with MORPH//Proceedings of the International Conference on Software Engineering. Zurich, Switzerland, 2012: 189-199

[77] Peters F, Menzies T, Layman L. LACE2: Better privacy-preserving data sharing for cross project defect prediction// Proceedings of the International Conference on Software Engineering. Firenze, Italy, 2015:801-811

[78] Bachemann A, Bird C, Rahman F, et al. The missing links: Bugs and bug-fix commits//Proceedings of the International Symposium on Foundations of Software Engineering. Santa Fe, USA, 2010: 97-106

[79] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: How misclassification impacts bug prediction//Proceedings of the Joint Meeting of the European Software Engineering Conference and the International Symposium on the Foundations of Software Engineering. San Francisco, USA, 2013: 392-401

[80] Nguyen A T, Nguyen T T, Nguyen H A, Nguyen T N. Multi-layered approach for recovering links between bug reports and fixes//Proceedings of the International Symposium on Foundations of Software Engineering. North Carolina, USA, 2012: 63:1-63:11

[81] Kim S, Zhang H Y, Wu R X, Gong L. Dealing with noise in defect prediction//Proceedings of the International Conference on Software Engineering. Cary, USA, 2011: 481-490

[82] Tantithamthavorn C, McIntosh S, Hassan A E, et al. The impact of mislabeling on the performance and interpretation of defect prediction models//Proceedings of the International Conference on Software Engineering. Florence, Italy, 2015: 812-823

[83] Liu S L, Chen X, Liu W S, et al. FECAR: A feature selection framework for software defect prediction//Proceedings of the Annual Computer Software and Applications Conference. Västerås, Sweden, 2014: 426-435

[84] Liu Wang-Shu, Chen Xiang, Gu Qing, et al. A noise tolerable feature selection framework for software defect prediction. Chinese Journal of Computers, to appear(in Chinese)
(刘望舒，陈翔，顾庆等. 一种面向软件缺陷预测的可容忍噪声的特征选择框架. 计算机学报，已录用)

[85] Liu W S, Liu S L, Gu Q, et al. Empirical studies of a two-stage data preprocessing approach for software fault prediction. IEEE Transactions on Reliability, 2016, 65(1): 38-53

[86] Zhang F, Mockus A, Keivanloo I, Zou Y. Towards building a universal defect prediction model with rank transformed predictors. Empirical Software Engineering, 2016, 21(5): 2107-2145

[87] Minku L L, Yao X. How to make best use of cross-company data in software effort estimation?//Proceedings of the International Conference on Software Engineering. Hyderabad, India, 2014: 446-456

[88] Kocaguneli E, Menzies T, Mendes E. Transfer learning in effort estimation. Empirical Software Engineering, 2015, 20(3): 813-843

[89] Kocaguneli E, Menzies T. How to find relevant data for effort estimation//Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Banff, Canada, 2011: 255-264

[90] Kitchenham B A, Mendes E, Travassos G H. Cross versus within-company cost estimation studies: A systematic review. IEEE Transactions on Software Engineering, 2007, 33(5): 316-329

[91] Tantithamthavorn C, McIntosh S, Hassan A E, Matsumoto K. An empirical comparison of model validation techniques for defect prediction models. IEEE Transactions on Software Engineering, 2017, 43(1): 1-18

[92] Herbold S. CrossPare: A tool for benchmarking cross-project defect predictions//Proceedings of the International Conference on Automated Software Engineering Workshops. Lincoln, USA, 2015: 90-96

[93] Lewis C, Lin Z P, Sadowski C, et al. Does bug prediction support human developers? Findings from a Google case study//Proceedings of the International Conference on Software Engineering. San Francisco, USA, 2013: 372-381

CHEN Xiang, born in 1980, Ph. D., associate professor. His research interests include software defect prediction, regression testing, software fault localization, and combinatorial testing.

WANG Li-Ping, born in 1992, M. S. Her research interest is software defect prediction.

GU Qing, born in 1972, Ph. D., professor, Ph. D. supervisor. His research interest is software quality assurance.

WANG Zan, born in 1979, Ph. D, associate professor. His research interests include softwarefault localization, automatic program repair.

NI Chao, born in 1990, Ph. D. candidate. His research interest is software defect prediction.

LIU Wang-Shu, born in 1987, Ph. D. His research interest is software defect prediction.

WANG Qiu-Ping, born in 1993, M. S. candidate. Her research interests is software defect prediction.

Background

In real software development process,a project, which is needed to predict software defects in advance, maybe a new project or a project having less training data. A simple solution is directly using training data from other projects to construct the prediction model. However the characteristic of different projects maybe not the same and thus cause the poor performance of defect prediction. Researchers named this problem as cross-project defect prediction (CPDP). We conduct a comprehensivesurvey onthis hot topic in software defect prediction domain and classify existing methods into three categories: supervised learning based methods, unsupervised learning based methods, and semi-supervised learning based methods. We summarize the motivation and achievements of existing research work in each category. Then we analyze the commonly used performance metrics and benchmarks in empirical studies in CPDP to support better empirical studies in future research. Finally we conclude this paper and discuss some potentially future research work. This survey can be a good support for domestic researchers to further solve the CPDP issue.

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61202006, 61202030, 61373012, 61402244, 61602267), the Open Project of State Key Laboratory for Novel Software Technology at Nanjing University (KFKT2016B18), and the University Natural Science Research Project of Jiangsu Province (Grant Nos. 15KJB520030, 16KJB520038).