ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 31,No. 1,Jan. 2016,pp. 56－64
DOI:10.16337/j.1004-9037.2016.01.005

http://sjcj.nuaa.edu.cn
E-mail:sjcj@nuaa.edu.cn
Tel/Fax:＋86-025-84892742

半监督软件缺陷挖掘研究综述

黎 铭 霍 轩

(南京大学计算机软件新技术国家重点实验室,南京,210023)

摘 要:软件质量是计算机系统安全可靠运行的保障,而软件缺陷是导致软件质量低下的重要诱因。软件缺陷挖掘技术凭借其能够通过对软件代码及其相关数据进行分析建模,发现软件系统潜在的缺陷,已得到了软件质量保障领域的广泛关注。要准确发现软件模块中潜在的缺陷,需要利用大量带有缺陷情况标注的模块进行学习。然而,缺陷情况标注往往需要通过详细测试或人工代码检查获取,要消耗大量测试和人工资源,在实际应用中难以满足,这严重制约了软件缺陷挖掘的性能。针对这一问题,半监督学习技术被引入软件缺陷挖掘,通过对大量缺少标注的模块进行利用,辅助提升软件缺陷挖掘的性能。本文对半监督缺陷挖掘技术的研究现状进行综述。首先综述了软件缺陷挖掘研究现状,然后简要介绍了半监督学习的4种学习范式;最后系统梳理了基于半监督学习进行软件缺陷挖掘的多种方法与技术。

关键词:软件挖掘;机器学习;半监督学习;软件缺陷挖掘
中图分类号:TP181;TP311.5　　**文献标志码**:A

Software Defect Mining Based on Semi-supervised Learning

Li Ming, Huo Xuan

(National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China)

Abstract: Software quality ensures the reliable running of the software system, and software defects reduce the quality of the software system. Software defects can be identified effectively by mining the codes as well as other related data, so the software defect mining technology has drawn significant attention in software quality assurance. To effectively identify potential software defects from the software modules, a large number of modules labeled as defective or non-defective information need to be collected for model construction. However, the labels of modules are usually obtained by extensive testing or manual code inspection, which consumes a huge amount of manpower and time. In practice, only a small number of labels can be collected, which seriously constrains the performance of defect identification. To solve this problem, the semi-supervised learning is introduced into software defect mining, thus the mining performance is improved by exploiting the large number of unlabeled modules. Here, the advances and the research status of semi-supervised software defect mining are reviewed and discussed extensively. Firstly, the existing studies on software defect mining is briefly review, and then the four major paradigms of semi-supervised learning are introduced. Finally, various methods and techniques on semi-supervised defect mining are systematically summarized and reviewed.

Key words: software mining; machine learning; semi-supervised learning; software defect mining

---

基金项目:国家自然科学基金(61422304,61272217)资助项目;江苏省自然科学基金(BK20131278)资助项目。
收稿日期:2016-01-04;**修订日期**:2016-01-15

引　言

软件是计算机系统的灵魂,软件的质量是计算机系统安全可靠运行的保障。伴随着软件运行环境逐渐从封闭、静态的单机环境向开放、动态多变的网络环境过渡,系统功能变得越发复杂,代码规模也不断增加,如何有效保证庞大而复杂的软件系统的质量,使其能够正确、可靠地运行,已成为软件研究者和软件开发人员所面临的巨大挑战。软件质量主要体现为软件的可靠性、可用性、有效性和可维护性[1],其中软件的可靠性尤为重要。导致软件系统不可靠的主要诱因之一是软件缺陷[2]。软件缺陷是软件中存在的某种破坏正常运行的问题、错误或隐藏的功能缺陷。包含缺陷的软件系统不仅会影响用户的正常使用,还可能会造成严重后果和巨大损失。例如:1996 年欧洲阿丽亚娜-5 型火箭因软件缺陷引发的飞行姿态错误导致升空数秒后爆炸,经济损失达 5 亿美元;2000 年美国国家癌症研究中心放疗机因软件缺陷错误计算放射量,造成 8 人死亡 20 人重伤。因此,要提高软件质量必须有效发现潜在的缺陷。

软件缺陷挖掘是一种能够有效发现软件中潜在缺陷的技术,它通过对软件模块源代码以及相关数据进行分析建模,挖掘蕴含于软件模块中的软件缺陷模式,据此来自动发现软件模块中包含的缺陷。软件缺陷挖掘通常需要对带有缺陷情况标注的软件模块进行学习,建立预测模型用于判断当前模块中是否包含软件缺陷。为获得良好的预测性能,除需要根据挖掘任务自身选取合适的学习算法以外,还需要提前收集大量带有缺陷情况标注程序模块作为训练数据。然而,一个程序模块的缺陷情况标注,即该模块"有缺陷"(Defective)或"无缺陷"(Non-defective),需要通过对该程序模块进行详细测试获得。为获取可靠的缺陷情况标注,对程序模块的详细测试往往应该覆盖程序不同的运行路径和运行状态,这势必带来大量测试资源的消耗。在实际软件开发过程中,测试资源通常十分有限,仅能负担对少量模块进行详细测试。这导致获得的带有缺陷情况标注的模块数量严重不足,极大影响了对软件缺陷模式的有效学习。要使软件缺陷挖掘技术能够在实际问题中发挥作用,必须寻找有效克服有缺陷情况标注的训练模块不足对学习建模造成的影响。

半监督学习是一种重要的利用未标注数据学习的技术。它通过让学习器自动地利用大量未标注数据辅助对少量有标注数据的学习,以期获得泛化性能的提升。凭借其无需大量有标注数据即可获得良好学习性能的特点,半监督学习被引入了软件缺陷挖掘任务,以期突破因有缺陷情况标注的模块不足对缺陷的学习建模过程造成的制约。本文对半监督软件缺陷挖掘技术的研究现状进行综述和讨论。

1　软件缺陷挖掘

软件缺陷挖掘是一种能够有效发现软件中潜在缺陷的技术,它将对软件模块源代码以及相关代码注释、设计文档等资源视为一种特殊的数据,基于数据挖掘技术对其进行分析建模,从中挖掘蕴含于软件模块中的软件缺陷模式,据此来自动发现软件模块中包含的缺陷。根据采用的挖掘技术不同,软件缺陷挖掘技术大致可分为两大类:描述型软件缺陷挖掘和预测型软件缺陷挖掘。

1.1　描述型软件缺陷挖掘

描述型软件缺陷挖掘的初衷是通过描述软件模块中存在的缺陷模式,以达到有效识别模块中软件缺陷的目的。然而,软件缺陷种类繁多且新缺陷不断出现,不同缺陷形成的原因也各异,难以对所有可能出现的缺陷进行有效描述。相比之下,由于程序编写过程需要遵循程序设计语言确立的各种编程规范和约定,无缺陷的正常程序应与此"正常模式"相符。基于上述观察,大多数描述型软件缺陷挖掘方法[3-5]并非直接对缺陷模式进行建模,而是设法对正常程序需要符合与遵循的模式进行建模,据此发现与此"正常模式"相违背的潜在软件缺陷。

获得程序正常模式描述的最直接方法是基于程序设计的领域知识定义出一套模板,然后根据所定

义的模板实现软件缺陷的检测[6]。然而,该方法对软件缺陷的甄别效果很大程度上依赖于所定义的模板正确性与完备性,需要经验丰富且对当前程序设计语言构造理解很深的专家才能定义出正确且相对完备的模板,难以在实际中大量应用。为此,Li 和 Zhou[3]提出了自动模板生成。他们假设程序中缺陷因不慎引入,缺陷自身的数量远不及正常模式多。因此,通过频繁模式进行挖掘,即可有效发现与描述程序中的正常模式。他们利用编码规则将函数作为项进行关联规则挖掘,获得了函数之间使用的耦合关系,据此发现不满足关系的潜在缺陷。正常程序代码中,许多函数经常成对出现(如:函数 malloc(·)与 free(·)),Livshitz 和 Zimmermann[4]通过关联规则挖掘方法,在软件的修改记录中挖掘匹配这样的成对模式,据此发现因未成对使用某些函数而造成的缺陷。当程序仅使用了一两次某些不常用的 API 函数时,其使用模式因不频繁而难以被关联规则挖掘所发现。Xie 等[5]借助 Google 代码搜索获得大量对该 API 使用历史记录,并通过对此进行关联规则挖掘获得 API 的使用规范。

利用关联规则挖掘可以发现程序中频繁出现的模式,但并非所有频繁模式都代表了正常模式。代码克隆是程序员在编程过程中将部分代码反复在不同地方复制使用而造成的,而这种代码克隆常因忘记修改相应变量或参数而导致错误。因此,在挖掘正常模式前,需要首先过滤该类过于频繁的模块。Li 等[7]提出 CloSpan 方法,通过哈希技术将代码片段转换成哈希编码,实现了克隆代码的检测。

1.2 预测型软件缺陷挖掘

预测型软件缺陷挖掘通过对大量带有缺陷情况标注的软件模块进行学习,建立对缺陷情况的预测模型,据此判断当前模块是否包含缺陷。预测型软件缺陷挖掘一般流程如图 1 所示。首先对软件源代码进行详细软件测试,获得每个模块的缺陷情况标注。然后利用软件度量[8]技术提取多种与软件模块缺陷相关的代码级统计指标(如:代码长度、代码分支数和代码的圈复杂度等)作为描述软件模块的特征,从而将程序代码转化为一组特征向量。基于特征向量表示的模块及其缺陷情况标注,通过监督学习得到预测模型,对新模块的缺陷情况进行挖掘和预测。

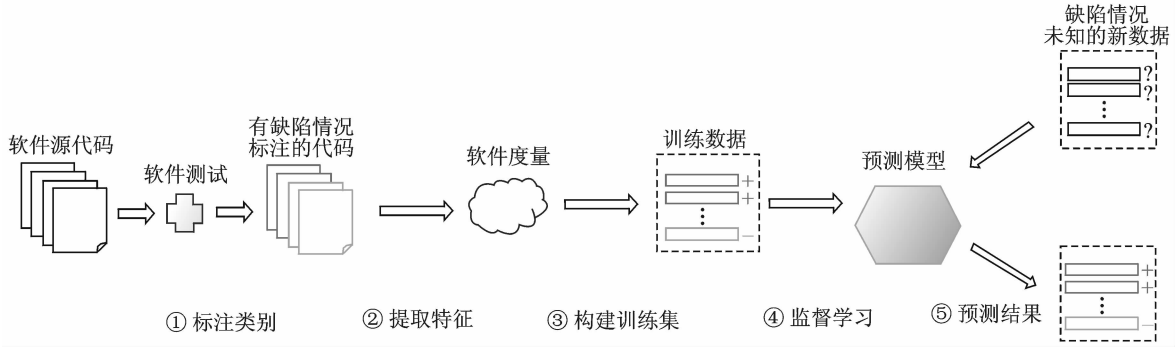

图 1 预测型软件缺陷挖掘

Fig. 1 Predictive software defect mining

预测型软件缺陷挖掘的建模过程有两个关键步骤:(1)提取适合于预测模块缺陷情况的特征;(2)选取合适的学习方法。在特征提取方面,Menzies 等[9]提出利用代码的静态特征(包括 LOC(代码长度)、McCabe 复杂度和 Halstead 复杂度等)来描述软件模块用于缺陷挖掘。在 Eclipse 开源项目上进行交叉验证的结果表明,基于代码静态特征,缺陷模块的召回率可达到 71%,验证了静态特征在软件缺陷挖掘方面的有效性。在上述特征基础上,Zimmermann 和 Nagappan[10]提出了图结构特征,用于软件缺陷挖掘。Kim 等[11]提出了软件时空局部性特征,并基于此预测软件模块的缺陷情况。除设计和使用新的软件度量作为特征以外,还可以基于现有特征学习更有利于进行软件缺陷挖掘的特征[12-13]。在学习算法

的选取方面,早些年学者们对不同的学习算法在软件缺陷挖掘任务上的表现进行了研究,使用的学习算法包括:朴素贝叶斯分类器[14]、逻辑回归[15]和神经网络[16]等。Lesserman 等[17]在 11 个 NASA 软件缺陷挖掘基准数据集上,通过大量实验对比了 19 个预测模型方法(SVM、逻辑回归、决策树以及朴素贝叶斯等),发现没有任何算法在所有数据上都一致优于其他算法。

近些年来,如何解决软件缺陷挖掘技术在实际任务应用中所面临的种种难题、提升软件缺陷挖掘在实际软件开发中的可用性,逐渐成为了研究者所关注的热点。Jiang 等[18]指出不同开发人员的编码习惯和风格会导致缺陷模式各异,因此需要针对不同开发人员构建预测模型,从而获得更优的缺陷预测性能。Tantithamthavorn 等[19]通过实验讨论了软件数据所包含噪声对缺陷预测结果的影响。Zimmermann 等[20]发现不同项目的代码特征分布相差大,进行跨软件项目的缺陷挖掘性能很差;Nam 等[21]基于迁移学习缓解跨项目缺陷挖掘性能不佳的问题,在一定程度上获得了性能的改善。Turhan 等[22]对跨公司软件缺陷挖掘问题进行了研究,为缓解挖掘性能不佳的困难,他们提出通过最近邻方法选择其他公司特征分布类似的缺陷情况标注已知的模块,然后再利用这些挑选出的模块进行跨公司缺陷挖掘。Peters 等[23]研究了跨项目挖掘中的隐私保护问题,提出了 LACE2 模型减少数据分享带来的隐私泄露隐患。Tan 等[24]提出了软件缺陷挖掘的评估和建模应该考虑时序关系。

除上述问题之外,实际软件开发中制约软件缺陷挖掘发挥作用的难题是难以获得大量带有缺陷情况标注的训练模块以供学习,导致得到的预测模型性能低下。半监督学习正是解决这一问题的有效途径,吸引了研究者广泛的关注。

2　半监督学习

半监督学习是与直推学习(Transductive learning)以及主动学习(Active learning)并列的三大利用未标注数据学习主流技术之一。与直推学习仅关注在未标注数据上的预测性能以及主动学习依赖于人工干预不同,半监督学习可以自动地对未标注数据加以利用,学习在整个数据分布上具有强泛化能力的模型。整个学习过程无需人工干预,完全基于学习系统自身实现对未标注数据的利用。半监督学习凭借其自身特点以及广阔的应用前景,已经成为机器学习界一个重要的热点研究方向。国际机器学习大会"十年最佳论文奖"自设立 6 年以来,已 3 次(2008 年,2009 年及 2013 年)授予半监督学习方面的研究工作。

半监督学习的核心问题是如何有效利用无标注数据辅助学习。根据对无标注数据的利用方式不同,半监督学习大致分为 4 大类,即:基于生成式模型的半监督学习、基于低密度划分的半监督学习、基于图的半监督学习方法以及基于分歧(Disagreement-based)的半监督学习。

基于生成式模型的半监督学习方法通常是把无标注样本属于每个类别的概率看成一组缺失参数,然后采用 EM(Expectation-maximization)算法对生成式模型的参数进行极大似然估计。不同方法的区别在于选择了不同的生成式模型作为基分类器,例如:混合高斯[25]、混合专家[26]以及朴素贝叶斯[27]。

基于低密度划分的半监督学习方法要求决策边界尽量通过数据较为稀疏区域,以免把聚类中稠密的数据点分到决策边界两侧。该类方法中的经典代表为 Joachims[28]提出的 TSVM 算法。TSVM 利用通过迭代式的交换分类边界两侧未标注样本的虚拟标注,实现在所有数据上的间隔最大化;在尽量正确分类有标注数据的同时,将决策边界"推"向数据分布相对稀疏的区域。由于 TSVM 的非凸损失函数会导致学习陷入局部极小点,且优化过程计算开销大,多种方法[29-31]被提出用于对其进行改进。

基于图的半监督学习方法利用有标注和无标注数据构建数据图,并且基于图上的邻接关系将标记从有标注的数据点向无标注数据点传播。根据标记传播方式可将基于图的半监督学习方法分为通过定义满足某种性质的标记传播方式来实现显式标记传播[32-33]和通过定义在图上的正则化项实现隐式标记传播[34]。相比标记的传播方式,数据图的构建方式对最终学习性能影响更大。虽然图的构建往往要依

赖大量领域知识,但仍可利用数据内在的结构信息,获得更鲁棒的数据图[35-37]。

基于分歧的半监督学习方法需同时协同多个有差异性的学习器来给未标注数据提供标注,并通过多学习器在未标注数据上预测的分歧选择置信度高的样本进行利用。典型方法包括:协同训练方法[38-39]、协同三分类器的半监督学习方法 tri-training[40]、多分类器集成半监督学习方法 Co-Forest[41]以及基于一致性置信度估计的半监督回归方法 COREG[42]等。更详细的介绍参见文献[43,44]。

3 基于半监督学习的软件缺陷挖掘

为缓解实际软件缺陷挖掘任务中难以获得大量模块的缺陷情况标注这一难题,半监督学习被引入软件缺陷挖掘,以利用大量缺少软件缺陷情况标注的模块,提升在少量带有缺陷情况标注的模块上学习建模的性能。半监督软件缺陷挖掘的基本流程如图 2 所示。首先选择部分模块送交测试,获得每个模块的缺陷情况标注,然后基于软件度量技术提取描述每个模块的特征,最后基于所形成的有标注模块和无标注模块进行半监督学习,用于预测新模块的缺陷情况。

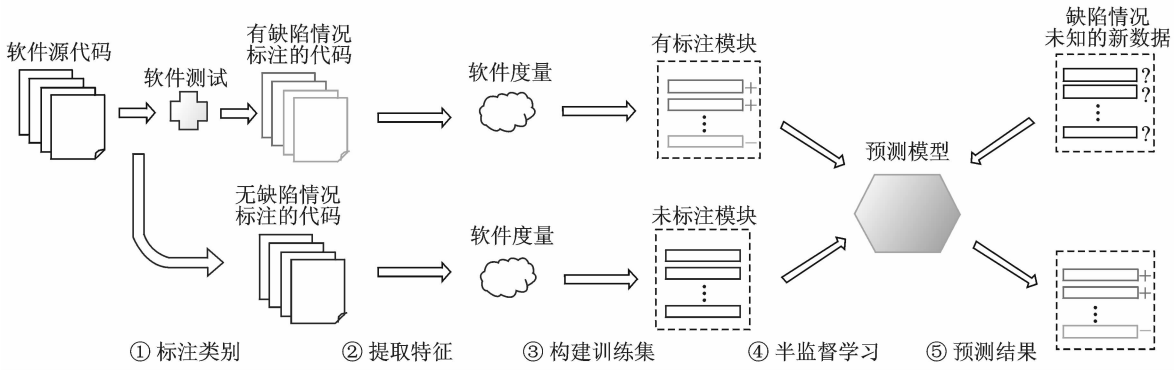

图 2 半监督软件缺陷挖掘

Fig. 2 Semi-supervised software defect mining

3.1 基于半监督聚类的软件缺陷挖掘

Seliya 等[45-46]最早将半监督学习技术引入软件缺陷挖掘。他们使用基于约束的 k-均值算法来对当前的软件模块数据进行半监督聚类。基于聚类结果,若一个簇(Cluster)中包含缺陷情况标注已知的模块,则根据该模块标注预测其余无标注模块的缺陷标注;若簇中不包含任何缺陷情况标注已知的模块,则提交软件工程师为该簇提供一个统一的标注。在 NASA 缺陷预测基准数据集上的结果表明,通过半监督学习能够性能提升约 14%。

Seliya 等[47]随后将上述工作进行了改进,利用半监督 EM 算法代替基于约束的 k-均值算法进行半监督学习。半监督 EM 算法可视为在有标注模块附近进行聚类[27],允许未标注模块可以概率形式隶属于不同簇,提升了对模块缺陷情况标注的估计性能。

Abaei 等[48]提出了一种结合了监督和非监督学习的神经网络模型 HySOM,用于进行半监督软件缺陷预测。该模型分为两个阶段,第 1 个阶段首先基于 SOM (Self-organization maps)神经网络对软件模块数据进行聚类,并根据聚类情况调整权值表示;第 2 个阶段训练一个前馈神经网络用于预测未标注模块的缺陷情况。

3.2 基于半监督分类的软件缺陷挖掘

Lu 等[49]使用了一种迭代式的半监督分类方法 FTF[50]进行缺陷预测。该方法是一种自我训练算法,在每轮迭代中基学习器同时利用有标注模块和未标注模块进行学习,并不断改变未标注模块的标注

状态,直至停止条件满足。为进一步提升学习性能,Lu 等[51]对上述工作进行了扩展,在半监督学习前首先使用多尺度分析对软件模块数据进行降维,基于降维后的结果利用 FTF 算法进行学习,取得了显著的性能提升。随后,Lu 等[52]又将该思想应用于基于主动学习的软件缺陷预测。Catal 和 Diri[53]使用了半监督分类方法 YATSI[54]来进行半监督软件缺陷挖掘。在此基础上,他们还对 YATSI 方法进行了扩展,提出了基于人工免疫算法的 YATSI 方法。

3.3 软件缺陷挖掘自身特性

上述研究[1,45-47,49,51,53]表明,半监督学习可以通过利用大量缺乏缺陷情况标注的模块提升软件缺陷挖掘的性能,然而半监督学习并非总能带来软件缺陷挖掘性能的提升,在一些情况下,半监督学习甚至会导致缺陷挖掘的性能下降[49,53,55]。除现有大多数半监督学习算法难以保证利用无标注数据进行学习的安全性[56-57]外,另一个重要的因素是直接应用现有半监督学习方法而不考虑软件缺陷挖掘任务自身的特性。若要获得更优、更鲁棒的软件缺陷挖掘性能,需要针对软件缺陷挖掘任务自身特性设计新型半监督软件缺陷挖掘方法。

在软件缺陷挖掘中,缺陷模块数量往往远少于正常模块数量,软件缺陷/正常模块的分布极为不平衡。这种不平衡在仅有少量有标注数据的半监督学习场景下会严重影响软件缺陷挖掘的性能。Jiang 等[55]讨论了软件缺陷挖掘中不平衡的数据分布对学习性能造成的影响,并提出了一种能够应对数据分布不平衡的半监督学习方法 ROCUS。该方法在基于分歧的半监督学习框架中嵌入了探索式下采样技术,在每轮迭代过程中,通过对自动标注的未标注模块进行多次探索式随机下采样,使得在对自动标注模块利用的过程中,既缓解了学习器更新过程中分布不平衡的自动标注模块对模型更新的影响,又增加了个体学习器之间的差异,提升了半监督学习的鲁棒性。在 NASA 软件缺陷挖掘基准数据集上的结果表明,该方法能够有效改善直接进行半监督学习导致的性能低下,还能够利用未标注模块获得比监督学习更优的软件缺陷挖掘性能。ROCUS 相对直接,进行监督学习性能提升超过 70%[55]。

在软件缺陷挖掘中,测试资源通常十分有限。若随意选取少数几个软件模块,通过详细测试的方式获得其缺陷情况标注,则获得标注的软件模块未必有助于缺陷模式的学习,从而导致学习性能低下。针对这一问题,Li 等[58]提出了主动半监督软件缺陷挖掘方法 ACoForest。该方法在基于分歧的半监督学习中引入主动学习机制,通过多学习器对未标注模块缺陷预测的分歧程度、估计学习器对该类缺陷模式的建模质量以及据此主动挑选出的最困难的模块,对其进行测试获得缺陷情况标注以供下一轮学习,从而在有限测试资源的情况下获得尽可能大的性能提升。在 Eclipse 等开源软件上软件缺陷挖掘结果表明,若固定获得缺陷情况标注的数量,引入主动学习后,ACoForest 方法较直接使用半监督学习方法以及监督学习方法有显著的性能提升;若达到相似的缺陷预测性能,ACoForest 较直接使用半监督学习、监督学习方法需要更少的缺陷情况标注,能够有效节约有限的测试资源。

在某些重要的软件开发任务中(例如载人航天器飞控系统),遗漏一个软件缺陷将会导致严重的后果。在现有半监督软件缺陷挖掘研究中,将漏报缺陷或误报缺陷等同对待,难以体现其不同预测错误所导致的不同代价。针对这一问题,Li 等[59]提出一种代价敏感半监督学习方法 CSCForest。该方法根据预设错误代价,对自动标注的模块进行重采样,通过样本分布的改变使得半监督学习过程能够隐含考虑不同代价的模块在学习中的权重,实现代价敏感学习。在 NASA 软件缺陷挖掘基准数据集上的挖掘结果表明,该方法能够获得更低的总代价。

4 结束语

软件缺陷挖掘能够通过对软件代码及其相关数据进行分析建模,自动挖掘隐藏在软件中的缺陷模式,从而辅助软件工程师更有效地发现软件缺陷,提升软件质量。为克服软件缺陷挖掘技术在实际应用

中面临的难以收集大量模块的缺陷情况标注这一困难,半监督学习技术被引入软件缺陷挖掘,通过对大量缺乏缺陷情况标注的模块进行自动利用,可有效提升软件缺陷挖掘的性能。本文对软件缺陷挖掘领域的概况及半监督软件缺陷挖掘技术的研究现状进行了介绍。半监督软件缺陷挖掘目前尚处于起步阶段,有不少重要问题有待进一步的研究。例如:半监督软件缺陷挖掘任务中,至少需要测试多少模块才能对隐藏的缺陷进行有效建模;如何确保半监督缺陷挖掘不会因利用缺乏缺陷情况标注的软件模块而导致性能下降。虽然"安全"的半监督学习方法在低密度划分假设下已有一些结果[57],但如何能够结合软件缺陷挖掘任务自身特性,设计出安全有效的半监督软件缺陷挖掘方法,尚需进一步探索。

参考文献:

[1] Sommerville I. Software engineering[M]. 9th Ed. Boston, MA, USA: Addison Wesley, 2010.

[2] Huizinga D, Kolawa A. Automated defect prevention: Best practices in software management[M]. New York, USA: John Wiley & Sons, 2007.

[3] Li Z, Zhou Y Y. PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code[C]//Proceedings of the 10th European Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Lisbon, Portugal: ACM, 2005:306-315.

[4] Livshits B, Zimmermann T. DynaMine: Finding common error patterns by mining software revision histories[C]//Proceedings of the 10th European Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Lisbon, Portugal: ACM, 2005:296-305.

[5] Tao X, Acharya M, Thummalapenta S, et al. Improving software reliability and productivity via mining program source code [C]//Proceedings ot the 22nd IEEE International Symposium on Parallel and Distributed Processing. Miami, USA: IEEE, 2008:1-5.

[6] Engler D, Chen D Y, Hallem S, et al. Bugs as inconsistent behavior: A general approach to inferring errors in systems code [C]//Proceedings of the 18th ACM Symposium on Operating System Principles. Banff, Alberta, Canada: [s. n.], 2001,35 (3):57-72.

[7] Li Z, Lu S, Myagmar S, et al. CP-Miner: A tool for finding copy-paste and related bugs in operating system code[C]//Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. San Francisco, USA: USENIX Association, 2004:289-302.

[8] Fenton N, Bieman J. Software metrics: A rigorous and practical approach[M]. Boca, Raton, USA: CRC Press, 2014.

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

[10] Zimmermann T, Nagappan N. Predicting defects using network analysis on dependency graphs[C]//Proceedings of the 30th International Conference on Software Engineering. Leipzig, Germany: ACM, 2008:531-540.

[11] Kim S, Zimmermann T, Whitehead E, et al. Predicting faults from cached history[C]//Proceedings of the 29th International Conference on Software Engineering. Minneapolis, MN, USA: IEEE, 2007:489-498.

[12] Nagappan N, Ball T, Zeller A. Mining metrics to predict component failures[C]//Proceedings of the 28th International Conference on Software Engineering. Shanghai, China: ACM, 2006:452-461.

[13] Jing X Y, Ying S, Zhang Z W, et al. Dictionary learning based software defect prediction[C]//Proceedings of the 36th International Conference on Software Engineering. Hyderabad, India: ACM, 2014:414-423.

[14] Pérez-Miñana E, Gras J J. Improving fault prediction using Bayesian networks for the development of embedded software applications[J]. Software Testing, Verification and Reliability, 2006,16(3):157-174.

[15] Gyimothy T, Ferenc R, Siket I. Empirical validation of object-oriented metrics on open software for fault prediction [J]. IEEE Transactions on Software Engineering, 2005,31(10):897-910.

[16] Dai Y S, Xie M, Long Q, et al. Uncertainty analysis in software reliability modeling by Bayesian approach with maximum-entropy principle[J]. IEEE Transactions on Software Engineering, 2007,33(11):781-795.

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

[18] Jiang T, Tan L, Kim S. Personalized defect prediction[C]//Proceedings of the 28th International Conference on Automated Software Engineering. Silicon, CA, USA: IEEE, 2013:279-289.

[19] Tantithamthavorn C, McIntosh S, Hassan A E, et al. The impact of mislabeling on the performance and interpretation of defect prediction models[C]//Proceedings of the 37th International Conference on Software Engineering. Firenze, Italy:

IEEE, 2015:812-823.
[20] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process[C]//Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering. Amsterdam, The Netherlands: ACM, 2009:91-100.
[21] Nam J, Pan SJ, Kim S. Transfer defect learning[C]//Proceedings of the 35th International Conference on Software Engineering. San Francisco, CA, USA: IEEE, 2013:382-391.
[22] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and withincompany data for defect prediction [J]. Empirical Software Engineering, 2009,14:540-578.
[23] Peters F, Menzies T, Layman L. LACE2: Better privacy-preserving data sharing for cross project defect prediction[C]//Proceedings of the 37th International Conference on Software Engineering. Firenze, Italy: IEEE, 2015:801-811.
[24] Tan M, Tan L, Dara S, et al. Online defect prediction for imbalanced data[C]//Proceedings of the 37th International Conference on Software Engineering. Firenze, Italy: IEEE, 2015:99-108.
[25] Shahshahani B, Landgrebe D. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon[J]. IEEE Transactions on Geoscience and Remote Sensing, 1994,32(5):1087-1095.
[26] Miller DJ, Uyar HS. A mixture of experts classifier with learning based on both labelled and unlabeled data[C]//Advances in Neural Information Processing Systems. Denver, CO, USA: MIT Press, 1997:571-577.
[27] Nigam K, McCallum A K, Thrun S, et al. Text classification from labeled and unlabeled documents using EM[J]. Machine Learning, 2000, 39(2/3):103-134.
[28] Joachims T. Transductive inference for text classification using support vector machines[C]//Proceedings of the 16th International Conference on Machine Learning. Bled, Slovenia: Morgan Kaufmann, 1999:200-209.
[29] Sindhwani V, Keerthi SS, Chapelle O. Deterministic annealing for semi-supervised kernel machines[C]//Proceedings of the 23rd International Conference on Machine Learning. Pittsburgh, USA: ACM, 2006:123-130.
[30] Collobert R, Sinz F, Weston J, et al. Large scale transductive SVMs [J]. Journal of Machine Learning Research, 2006,7(8):1687-1712.
[31] Li Y F, Kwok J T, Zhou Z H. Semi-supervised learning using label mean[C]//Proceedings of the 26th International Conference on Machine Learning. Montreal, Canada: ACM, 2009:633-640.
[32] Zhu X J, Ghahramani Z, Lafferty J. Semi-supervised learning using Gaussian fields and harmonic functions[C] //Proceedings of the 20th International Conference on Machine Learning. Washington, USA: AAAI, 2003:912-919.
[33] Zhou D Y, Bousquet O, Lal T N, et al. Learning with local and global consistency[C]//Advances in Neural Information Processing Systems 16. Cambridge, MA: MIT Press, 2004:321-328.
[34] Belkin M, Niyogi P, Sindwani V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples[J]. Journal of Machine Learning Research, 2006,7(11):2399-2434.
[35] Zemel R S, Carreira-Perpiñán M Á. Proximity graphs for clustering and manifold learning[C]//Advances in Neural Information Processing Systems. Vancouver, Canada: MIT Press, 2004:225-232.
[36] Zhang Y, Zhou Z H. Non-metric label propagation[C]//Proceedings of the 21st International Conference on Artificial Intelligence. Pasadena, USA: ExaCt, 2009:1357-1362.
[37] Wang F, Zhang C S. Label propagation through linear neighborhoods[J]. IEEE Transactions on Knowledge and Data Engineering, 2008,20(1):55-67.
[38] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training[C]//Proceedings of the 11th Annual Conference on Computational Learning Theory. Madison, USA: ACM, 1998:92-100.
[39] Goldman S, Zhou Y. Enhancing supervised learning with unlabeled data[C]//Proceedings of the 17th International Conference on Machine Learning. Stanford, CA, USA: Morgan Kaufmann, 2000:327-334.
[40] Zhou Z H, Li M. Tri-training: Exploiting unlabeled data using three classifiers [J]. IEEE Transactions on Knowledge and Data Engineering, 2005,17(11):1529-1541.
[41] Li M, Zhou Z H. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2007,37(6):1088-1098.
[42] Zhou Z H, Li M. Semi-supervised regression with co-training style algorithms[J]. IEEE Transactions on Knowledge and Data Engineering, 2007,19(11):1479-1493.
[43] Zhou Z H, Li M. Semi-supervised learning by disagreement [J]. Knowledge and Information Systems, 2010,24(3):415-439.
[44] 周志华.基于分歧的半监督学习[J].自动化学报,2013,39(11):1871-1878.

Zhou Zhihua. Disagreement-based semi-supervised leaning[J]. Journal of Automatica Sinica, 2013,39(11):1871-1878.

[45] Seliya N, Khoshgoftaar T M. Software quality analysis of unlabeled program modules with semisupervised clustering[J]. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2007,37(2):201-211.

[46] Seliya N, Khoshgoftaar T M, Zhong S. Analyzing software quality with limited fault-proneness defect data[C]//Proceedings of the 9th International Symposium on High-Assurance Systems Engineering. Heidelberg, Germany: IEEE, 2005:89-98.

[47] Seliya N, Khoshgoftaar T M. Software quality estimation with limited fault data: A semi-supervised learning perspective[J]. Software Quality Journal, 2007,15(3):327-344.

[48] Abaei G, Selamat A, Fujita H. An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction[J]. Knowledge-Based Systems, 2015,74:28-39.

[49] Lu H, Cukic B, Culp M. An iterative semi-supervised approach to software fault prediction[C]//Proceedings of the 7th International Conference on Predictive Models in Software Engineering. Banff, Canada: ACM, 2011:1-10.

[50] Culp M, Michailidis G. An iterative algorithm for extending learners to a semi-supervised setting[J]. Journal of Computational and Graphical Statistics, 2008,17(3):545-571.

[51] Lu H, Cukic B, Culp M. Software defect prediction using semi-supervised learning with dimension reduction[C]//Proceedings of the 27th International Conference on Automated Software Engineering. Essen, Germany: IEEE, 2012:314-317.

[52] Lu H, Kocaguneli E, Cukic B. Defect prediction between software versions with active learning and dimensionality reduction [C]//Proceedings of the 25th International Symposium on Software Reliability Engineering. Naples, Italy: IEEE, 2014:312-322.

[53] Catal C, Diri B. Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction[J]. Expert Systems, 2009,26(5):458-471.

[54] Driessens K, Reutemann P, Pfahringer B, et al. Using weighted nearest neighbor to benefit from unlabeled data[C]//Advances in Knowledge Discovery and Data Mining. Berlin, German: Springer, 2006:60-69.

[55] Jiang Y, Li M, Zhou Z H. Software defect detection with ROCUS [J]. Journal of Computer Science and Technology, 2011, 26(2):328-342.

[56] Cozman F G, Cohen I, Cirelo M. Unlabeled data can degrade classification performance of generative classifiers[C]//Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference. Florida, USA: AAAI, 2002: 327-331.

[57] Li Y F, Zhou Z H. Towards making unlabeled data never hurt [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015,37(1):175-188.

[58] Li M, Zhang H Y, Wu R X, et al. Sample-based software defect prediction with active and semi-supervised learning[J]. Automated Software Engineering, 2012,19(2):201-230.

[59] Li M, Tu W W, Fan C Y, et al. Cost sensitive semi-supervised defect prediction[C]//Working Notes of the ACML 2012 Workshop on Learning with Weak Supervision. Singapore: JMLR, 2012:45-59.

作者简介：

黎铭(1980-)，男，副教授，研究方向：软件挖掘、机器学习，E-mail：lim@lamda.nju.edu.cn。

霍轩(1990-)，男，博士研究生，研究方向：软件挖掘、机器学习，E-mail：huox@lamda.nju.edu.cn。