计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications [doi: 10.15888/j.cnki.csa.009558]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于动态分布对齐和伪标签学习的跨项目缺陷预测①

高芹芹, 凌松松, 于　婕, 于　旭

(青岛科技大学 数据科学学院, 青岛 266061)
通信作者: 于　旭, E-mail: yuxu0532@163.com

摘　要: 跨项目缺陷预测 (cross-project defect prediction, CPDP) 已经成为软件工程和数据挖掘领域的一个重要研究方向, 利用其他数据丰富项目的缺陷代码来建立预测模型, 解决了模型构建过程中的数据不足问题. 然而源项目和目标项目的代码文件之间存在的分布差异, 导致跨项目预测效果不佳. 大多数研究采用域适应方法来解决这一问题, 但是现有的方法一方面只考虑了条件分布或边缘分布对缺陷预测的影响, 忽视了其动态性; 另一方面没有选择合适的伪标签. 基于上述两个方面, 本文提出了一种基于动态分布对齐和伪标签学习的跨项目缺陷预测方法 (DPLD). 具体来说, 我们通过对抗域适应方法分别在域对齐和类别对齐模块中减小项目间的边缘分布差异和条件分布差异, 并借助动态分布因子动态、定量地描述了两种分布的相对重要性. 此外, 本文也提出了一种伪标签学习方法, 通过数据间的几何相似性来增强伪标签作为真实标签的准确性. 本文在 PROMISE 数据集上进行了实验, *F*-measure 和 AUC 的值分别提升了 22.98%、15.21%, 表明了本文方法在减小项目间分布差异、提升跨项目缺陷预测性能上的有效性.

关键词: 领域自适应; 跨项目缺陷预测; 条件分布; 边缘分布; 伪标签学习

引用格式: 高芹芹,凌松松,于婕,于旭.基于动态分布对齐和伪标签学习的跨项目缺陷预测.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/9558.html

Cross-project Defect Prediction Based on Dynamic Distribution Alignment and Pseudo-label Learning

GAO Qin-Qin, LING Song-Song, YU Jie, YU Xu

(School of Data Science, Qingdao University of Science and Technology, Qingdao 266061, China)

Abstract: Cross-project defect prediction (CPDP) has emerged as a crucial research area in software engineering and data mining. Using defective code from other data-rich projects to build prediction models solves the problem of insufficient data during model construction. However, the distribution difference between the code files of source and target projects results in poor cross-project prediction. Most studies adopt the domain adaptation methods to solve this problem, but the existing methods only focus on the influence of conditional or marginal distribution on domain adaptation, ignoring its dynamics. On the other hand, they fail to choose appropriate pseudo-labels. Based on the above two aspects, this study proposes a cross-project defect prediction method based on dynamic distribution alignment and pseudo-label learning (DPLD). Specifically, the proposed method reduces the marginal and conditional distribution differences between projects in the domain alignment and category alignment modules, respectively, by means of the adversarial domain adaptation method. Additionally, it dynamically and quantitatively characterizes the relative importance of the two distributions using dynamic distribution factors. Furthermore, this study proposes a pseudo-label learning method to enhance the

① 基金项目: 国家自然科学基金 (62172249); 中央高校基本科研业务费专项资金 (93K172022K01)
收稿时间: 2024-01-14; 修改时间: 2024-02-07; 采用时间: 2024-02-26; csa 在线出版时间: 2024-07-03

accuracy of pseudo-labels as real labels through the geometric similarity between data. Experiments conducted on the PROMISE dataset show that DPLD achieves average improvements of 22.98% and 15.21% in terms of *F*-measure and AUC, respectively. These results demonstrate the effectiveness of the DPLD method in reducing distribution differences between projects and improving the performance of cross-project defect prediction.

Key words: domain adaption; cross-project defect prediction; conditional distribution; marginal distribution; pseudo-label learning

软件缺陷 (software defect) 是指软件中存在的错误、漏洞或不完善之处, 可能会导致软件在运行或使用过程中出现异常行为、崩溃、不符合设计预期或不满足用户需求等问题. 由于软件开发过程中的复杂性、时间压力、变更管理等诸多因素, 软件缺陷的产生在很大程度上是不可避免的. 这些缺陷可能导致软件性能下降, 若未及时发现与修复, 不仅影响用户体验, 也可能威胁软件系统安全, 甚至造成巨大的经济损失. 因此, 研究者们提出了软件缺陷预测技术 (software defect prediction, SDP)[1,2], 旨在提前识别和修复潜在的软件缺陷, 从而提高软件质量、降低开发成本和减少用户风险, 以保障软件系统的稳定性和可靠性.

传统的 SDP 方法依赖于项目内足够多的已标记的历史数据来训练稳定的分类模型, 并使用这个分类模型预测同一个项目内新样本的缺陷标签, 这种方法称为项目内缺陷预测 (within-project defect prediction, WPDP)[3−5]. 然而, 获取足够多的历史缺陷数据并不现实, 特别是对于新项目来说, 历史数据可能是非常稀少的, 而且软件规模的扩大和版本的更迭也十分迅速, 这限制了项目内软件缺陷预测的发展. 据此, 研究人员提出了跨项目软件缺陷预测 (CPDP)[6−9], 可以有效解决项目内缺陷预测面临的数据稀疏问题.

然而, 由于开发人员、应用领域及编程语言的不同, CPDP 中源项目和目标项目之间的数据分布存在显著差异, 这不符合机器学习中训练集和测试集独立同分布的假设, 会造成模型预测效果较差, 难以达到预期的效果. 为了解决这个问题, 许多学者采用域适应的方法进行了相关研究, 特别是通过分布对齐[10−13]来减少域之间的分布差异. 所处理的两种分布包括边缘分布和条件分布. 边缘分布对齐旨在使不同项目中特征的边缘分布尽可能相似. 这有助于确保特征的全局分布统计特性 (如均值、方差等) 在不同项目之间保持一致, 从而提高了模型的稳定性和泛化性能. Pan 等人[10]提出了迁移成分分析方法 (transfer component analysis, TCA), 通过提取源项目和目标项目间的公共迁移成分, 使得不同项目在映射后的子空间中分布差异减小, 从而对齐项目间的边缘分布. Nam 等人[11]提出了 TCA+模型, 在 TCA 方法的基础上将数据归一化用于缺陷预测. 条件分布对齐确保不同项目中特征在给定条件下 (例如不同类别或不同标签) 的分布相似. Satpal 等人[12]基于输入和标签上共同定义的特征来训练一个条件概率模型, 结果表明这种领域适应方法显著降低了误差. 大多数研究仅考虑了项目间的边缘分布对齐, 近年来, 许多学者进一步拓展预测模型, 将边缘分布和条件分布对齐结合应用于缺陷预测. Zou 等人[13]利用两个自编码器同时学习项目间的全局特征表示和两个类别之间的局部特征表示, 并在每个堆栈层中实现渐进式分布匹配, 从而实现项目间边缘和条件分布对齐.

现有的 CPDP 方法要么集中在仅减少项目间的边缘分布差异或条件分布差异, 要么集中在同时减少以上两种分布的差异, 但是认为两种分布的重要性是相等的. 而当项目间的分布差异很大时, 边缘分布的重要性通常高于条件分布; 当项目间的分布相似时, 条件分布的重要性比边缘分布更大. 因此, 仅考虑一种分布或者平等地考虑两种分布的重要性可能会使得模型的预测性能不佳. 此外, 对于目标项目中的无标记数据, 伪标签的选择是不合理的. 现有的大多数方法直接采用共享分类器预测的伪标签, 受分类器性能的限制, 伪标签和真实标签之间可能存在着严重的偏差, 这种偏差的不断累积会使得预测结果与正确结果的误差越来越大.

据此, 本文提出了一种基于动态分布对齐和伪标签学习的跨项目软件缺陷预测方法. 首先通过域对齐和类别对齐分别减小项目间的边缘分布差异和条件分布差异, 同时借助动态分布因子动态、定量地描述了两种分布的相对重要性. 其次, 设计了一种伪标签学习方法, 借助原始数据间的几何相似性, 提升伪标签在充

当真实标签时的准确性. 本文的主要贡献如下.

(1) 我们使用域适应的方法, 实现了源项目到目标项目的知识迁移, 解决了目标项目有标注数据过少, 模型难以训练的问题.

(2) 我们设计了域对齐和类别对齐两个模块来减小项目间的分布差异, 并且动态地描述边缘分布和条件分布的相对重要性, 有效解决了 CPDP 过程中项目间的数据分布差异问题.

(3) 在类别对齐部分, 我们设计了一种伪标签学习方法, 以便为那些在特征空间中与被监督数据的类别质心具有几何相似度的实例分配准确的伪标签, 从而最大程度地利用标签信息.

(4) 在常用的缺陷公开数据集 PROMISE 上进行了充分实验, 结果显示本文方法明显优于现有的 CPDP 方法.

本文第 2 节是对相关工作的回顾, 包括跨项目缺陷预测和域适应方法的研究介绍. 第 3 节介绍本文的主要方法, 给出问题描述、总体框架、特征映射、域对齐、类别对齐、动态分布因子和模型目标函数的详细描述. 第 4 节主要展示实验的结果并进行详细分析. 第 5 节对全文进行总结和展望.

1 相关工作

1.1 跨项目缺陷预测

跨项目缺陷预测的目标是借助样本及标签信息丰富的源项目数据来训练分类器, 实现目标项目样本有无缺陷的倾向性预测. 按照迁移内容的不同, CPDP 方法可以分为基于实例迁移的方法和基于特征迁移的方法.

基于实例迁移的方法过滤出与目标项目相关的实例或者对实例进行加权以削弱来自不相关项目的不利影响. 这些筛选出的实例有助于减少项目间的分布差异. Turhan 等人[14]提出最近邻过滤 (NNFilter) 的方法, 目的是只从源数据中选择相似的样本与目标数据进行匹配. 通过不使用所有源数据, 只使用每个测试数据的最近邻居来形成训练集, 该训练集与目标数据具有相似的特征. He 等人[15]提出了一种利用分布特征 (如均值、中位数、最大值、最小值等) 来计算训练数据和测试数据的相似度, 自动选择合适的训练数据的方法. Ma 等人[16]提出了迁移朴素贝叶斯 (TNB) 方法, 该方法只使用源数据集和目标数据集之间的共享特征. 为减少跨公司与公司内部数据的分布差异, Chen 等人[17]提出了双传输增强算法 (DTB), 通过减少跨公司数据中的负样本来减少分布差异. Ryu 等人[18]提出了 VCB-SVM 方法, 根据从分布特征中获得的相似性权重计算误分类成本, 从而解决类别不平衡问题. 基于实例迁移的 CPDP 方法会导致可用实例很少或浪费源项目中的一些数据信息, 使得预测性能不佳.

基于特征迁移的方法通过从源项目中选择和目标项目特征相关的度量元或者构建一个公共的特征空间来实现域间的特征映射, 以消除项目间特征的差异性. Nam 等人[11]提出了 TCA 方法的扩展 (TCA+), 该方法在学习再生核希尔伯特空间 (RKHS) 中跨域的特定传输组件之前, 使用规则找到最佳策略来规范化数据. 在 RKHS 学习中, 使用最大均值差异 (MMD) 来测量距离. 结果表明, 与原始 TCA 方法相比, 该方法性能有所提升. Long 等人[19]提出了联合分布适应 (JDA), 该方法借助降维过程共同匹配边缘分布和条件分布, 并构建了有效且稳健的新特征表示. Wu 等人[20]提出 CKSDL 方法, 借助少量的有标签数据和大量的无标签数据来消除分布差异. Xu 等人[21]提出了平衡分布适应 (BDA) 来解决分布之间的较大差异, 通过调整项目间分布的重要性有效地处理其分布差异. Zou 等人[13]提出了基于联合特征表示的跨项目缺陷预测方法 (DMDA_JFR), 该方法首先利用编码器学习局部和全局特征表示, 其次在每个堆栈层学习后进行特征空间的渐次对齐. 基于特征迁移的方法只考虑了项目间的公共特征表示, 许多度量元中的信息没有得到充分挖掘直接被丢弃; 另一方面, 平等地考虑边缘分布和条件分布的重要性来减少项目间的差异, 使得预测模型的提升效果有限. 本文属于基于特征迁移的方法, 我们不仅将边缘分布和条件分布对齐用于 CPDP 中, 并利用动态分布因子动态、定量地描述两种分布的相对重要性, 有效提升了跨项目缺陷预测性能.

1.2 域适应

为了有效解决因不同领域的分布差异导致跨领域预测精度表现不佳的问题, 有关学者提出了“域适应”的概念. 这种方法是迁移学习的一种实现方式, 旨在通过训练学习源域和目标域之间的共性或对概率分布进行适配, 从而获得性能优良的学习模型, 以提高模型在目标域上的泛化能力. 根据采用技术的不同, 可以将域适应方法分为基于度量学习的域适应和基于对抗学习

的域适应.

基于度量学习的域适应方法通过最小化不同领域之间的距离来减小数据分布差异. Zhuang 等人[22]提出了一种名为 TLDA 的监督表示学习方法. 该方法的核心思想是在嵌入层中, 通过最小化 KL 散度来缩小源域和目标域之间嵌入实例的分布距离. 这一策略极大地提升了迁移学习的性能, 为处理跨域数据提供了新的可能性. Li 等人[23]提出了 LPJT 的联合适配方法, 该方法通过训练一个特征变换矩阵将两个项目的样本映射到一个低维的特征空间中, 然后借助最大均值差异同时实现边缘分布和条件分布的最小化. Chen 等人[24]设计了一种基于双层域混合的新框架, 通过提取区域级和样本级的域不变表示, 缓解不同领域间数据分布不匹配的问题. 为了缓解传统特征对齐中过度适应的问题, Gao 等人[25]提出了一种非对称适应范式 (AsyFOD), 借助统一的差异估计函数来识别源域中的实例, 这些实例与目标域相似. 此外, 与目标域不相似的实例和增强的目标实例之间采用异步对齐的方式, 进一步缓解数据不平衡的问题.

基于对抗学习的域适应方法引入了对抗训练, 试图通过训练一个生成器和一个判别器来减小领域之间的分布差异. Ganin 等人[26]提出的 DANN 模型通过对抗训练的方式, 利用神经网络在源域和目标域之间学习域不变的特征表示, 从而有效地减小了领域间的数据分布差异问题, 在目标域上取得了显著的改善效果. 在 DANN 模型的基础上, Yu 等人[27]进一步考虑了条件分布对域适应的贡献, 并引入动态分布自适应机制, 使模型能够根据不同目标领域的数据分布动态调整适应性, 从而提高了迁移性能. Zhou 等人[28]提出了多源域适应双工对抗网络 (DAN_MA), 通过对抗网络和域分类器生成域间的不变特征表示, 同时使目标样本接近每个源域的决策边界以消除不同源之间的域特征. 为了提高模型对目标数据的预测置信度, Sun 等人[29]研究了末层激活的对抗训练, 并系统分析了归一化对对抗训练的影响. 本文将对抗学习的思想应用于跨项目缺陷预测中, 以解决源项目和目标项目间的分布差异问题.

2 本文方法

对于有标签的源域数据和仅有少量标签的目标域数据, 我们可以将问题重新定义为基于半监督域适应的跨项目软件缺陷预测. 假设 $D_S=\{X_S,Y_S\}=\{(x_1^S,y_1^S),(x_2^S,y_2^S),\cdots,(x_m^S,y_m^S)\}$表示源项目的训练样本, 其中$x_i^S\in R^{ds}$ 表示数据集中的第 i 个样本, d_S 表示样本中度量元的个数, $y_i^S\in\{0,1\}$表示第 i 个样本对应的标签, 其中 1 和 0 分别代表样本有无缺陷, m 表示样本的个数. 类似地, 我们假设目标项目数据集为$D_T=\{D_L,D_U\}$, 其中$D_L=\{X_L,Y_L\}=\{(x_1^L,y_1^L),(x_2^L,y_2^L),\cdots,(x_n^L,y_n^L)\}$为目标项目中少量有标签的样本数据集, $D_U=\{X_U\}=\{x_1^U,x_2^U,\cdots,x_l^U\}$为无标签的样本数据集. 源项目和目标项目之间的边缘分布和条件分布均存在差异, 即$P_S(X_S)\neq P_T(X_T)$, $P_S(Y_S\mid X_S)\neq P_T(Y_T\mid X_T)$. 本文的目标是根据标签信息丰富的源项目数据训练一个分类器用于目标项目中实例有无缺陷的预测.

为了动态、定量地衡量边缘分布和条件分布的贡献并充分挖掘和利用伪标签, 我们提出了基于动态分布对齐和伪标签学习的跨项目软件缺陷预测方法 (DPLD). 首先将源项目和目标项目数据映射到一个公共的特征子空间, 然后分别利用域对齐和类别对齐减小项目间的边缘分布和条件分布差异, 同时根据特征的几何质心更新伪标签, 提高伪标签的准确性, 最后不断更新动态分布因子来调整边缘分布和条件分布的相对重要性. 本文所提方法的总体架构如图 1 所示.

2.1 特征映射

类别不平衡问题[30–32]一直是软件缺陷预测中普遍存在的问题. 研究指出, 过采样方法是处理类别不平衡问题的有效策略, 其效果远胜于欠采样方法及代价敏感方法[33]. 本文采用了以 SMOTE 作为基础的过采样方法来处理数据集.

获取平衡数据集之后, 需要将缺陷样本转化为特征向量, 以便模型能够进行进一步的计算. 缺陷样本由度量元组成, 度量元是与缺陷外在表现相关的内在属性 (如复杂度、耦合性、继承性等) 的统计信息. 因此, 如何对度量元进行充分挖掘和应用非常关键. 本文选取了具有较强的非线性拟合能力和并行性的多层感知机 (MLP)[34]模型作为样本度量元的映射模型. 以目标项目为例, 样本数据集$D_T=\{x_1^T,x_2^T,\cdots,x_{n+l}^T\}$经 MLP 映射后, 得到样本的特征向量集合$E_T=\{e_1^T,e_2^T,\cdots,e_{n+l}^T\}$. 同理可得源项目样本的特征向量集合$E_S=\{e_1^S,e_2^S,\cdots,e_m^S\}$, 向量的特征维度均为$d_{\text{input}}$. 映射函数如下:

$$e^T=Sigmoid(W^Tx+b^T) \tag{1}$$

$$e^S = Sigmoid(W^S x + b^S) \tag{2}$$

其中, *Sigmoid* 为激活函数, $\theta_f = \{W^T, W^S, b^T, b^S\}$是特征映射的权重矩阵和偏置参数.

本方法的最终目标在于建立一个分类器 C, 以能够准确预测目标项目中未标记样本的标签. 通常情况下, 分类器的训练是通过最小化源数据的经验误差来实现的, 忽视了目标项目中少量的标签信息. 本文中我们充分利用了源项目和目标项目中的标签信息进行监督学习, 将交叉熵损失作为其分类损失函数, 公式如下:

$$L_y = -\frac{1}{m+n}\sum_{e_i\in(E_S\cup E_L)} y_i \log(C(e_i)) + (1-y_i)\log(1-(C(e_i))) \tag{3}$$

其中, E_S 和E_L分别表示源项目和目标项目映射后的特征向量集合, m 和 n 分别表示源项目和目标项目中的样本个数, y_i表示样本 i 对应的标签.

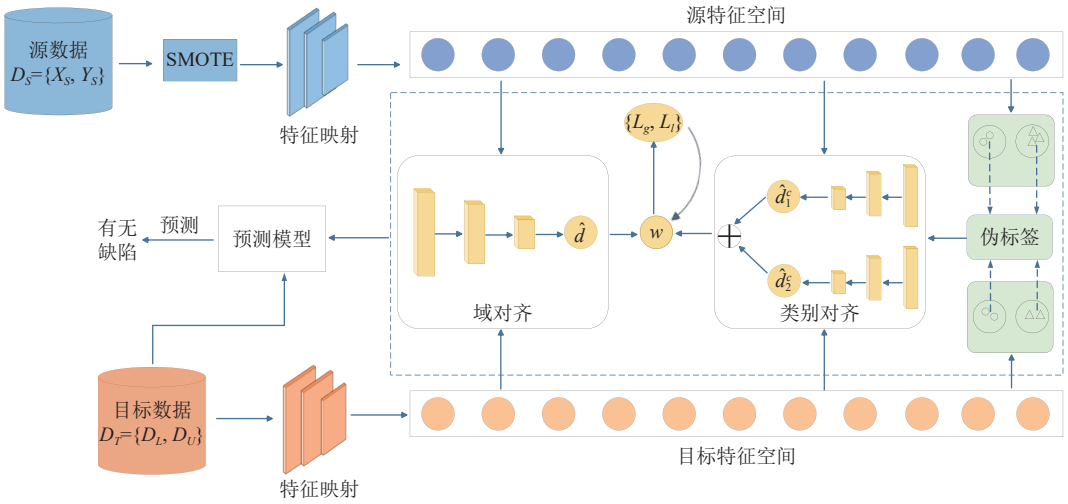

图 1 DPLD 方法总体架构图

2.2 域对齐

为了减小源项目和目标项目之间的边缘分布差异, 我们借鉴 GAN 网络[35]的思想设计了一个域对齐模块. 具体来说, 包括一个全局域鉴别器D_g, 可以区分样本所属的域类别. 全局域鉴别器通过 3 层的全连接层构建, 与特征映射网络在训练过程中进行对抗学习. 具体步骤如下: 将特征映射后的向量输入域鉴别器, 计算其损失函数, 首先通过最小化域鉴别器的损失函数更新域鉴别器的参数, 提高域鉴别器的判别能力; 然后通过最大化域鉴别器的损失函数学习并更新特征映射网络的参数, 从而调整特征映射网络输出的特征向量, 循环往复, 直到域鉴别器无法区分样本所属的领域. 全局域鉴别器的损失计算方法如下.

$$L_g = -\frac{1}{m+n}\sum_{e_i\in(E_S\cup E_T)} d_i \log(D_g(e_i)) + (1-d_i)\log(1-D_g(e_i)) \tag{4}$$

其中, D_g为全局域鉴别器, d_i为输入样本x_i的域标签.

2.3 类别对齐

为了对齐源项目和目标项目之间的条件分布, 本节设计了一个类别对齐模块, 主要包括两个局部域鉴别器 (D_l^1和D_l^2). 与全局域鉴别器相比, 局部域鉴别器可以实现与样本类别 (即有缺陷或无缺陷) 相关联的源项目和目标项目数据分布的匹配, 从而在更细粒度上减小项目间的分布差异. 局部域鉴别器的损失函数可计算为:

$$L_l = -\frac{1}{m+n}\sum_{e_i\in(E_S\cup E_T)} L_{ce}(D_l^1(\hat{y}_i^1 e_i), d_i) + L_{ce}(D_l^2(\hat{y}_i^2 e_i), d_i) \tag{5}$$

其中, L_{ce}表示交叉熵函数, $\hat{y}_i^1$和$\hat{y}_i^2$分别表示在有缺陷和无缺陷类别上的预测概率分布, d_i是输入样本x_i的域标签.

在跨项目软件缺陷预测中, 条件分布对齐可以确保不同项目中特征在不同缺陷标签下的分布相似. 这确保了模型能够在不同缺陷的情况下具有一致的预测性能. 在条件分布对齐的过程中, 我们需要用到样本的

标签信息, 但是目标项目中存在大量没有标签的样本, 无法满足条件分布的要求. 为了解决这一问题, 我们设计了一种伪标签学习方法, 以便为那些在特征空间中与被监督数据的类别质心具有几何相似度的实例分配准确的伪标签.

首先, 我们分别计算源项目和有标签的目标项目中类别 c (c=2, 表示有缺陷或无缺陷) 的质心, 这些质心是每个类别中映射后特征的均值向量. 计算公式如下:

$$\mu_c = \frac{1}{|E_S^c \cup E_L^c|}\left(\sum_{e_i^S \in E_S^c} e_i^S + \sum_{e_j^T \in E_L^c} e_j^T\right) \tag{6}$$

由此, 我们可以得到两个类别的质心$\{\mu_1, \mu_2\}$.

其次, 我们借助余弦相似度和获得的质心为目标项目中未标记的样本分配标签.

$$\hat{y}_i^U = \arg\max_c GR(Sigmoid(W^T x_i^U + b), \mu_c) \tag{7}$$

其中, $GR(\cdot)$表示特征空间中两个向量之间的几何关系, 本文选用余弦相似度.

2.4 动态分布因子

A-distance 是一种用于度量不同数据分布之间差异的距离度量方法, 可以衡量两个数据分布之间的相似程度. 训练过程中, A-distance 需要建立一个线性分类器, 本文中将全局和局部域鉴别器看作线性分类器, 它可以直接利用域鉴别器的损失自动学习并更新动态分布因子.

全局域鉴别器的 A-distance 计算公式如下:

$$A_g(D_S, D_T) = 2(1 - 2(L_g)) \tag{8}$$

其中, D_S, D_T 分别表示源项目和目标项目的样本, L_g 表示全局域鉴别器损失. A_g的值越小, 表明源项目和目标项目间的边缘分布差异越小, 其数据分布越相似, 边缘分布在域适应中的贡献更大.

同理可得, 局部域鉴别器的 A-distance 为:

$$A_l(D_S^c, D_T^c) = 2(1 - 2(L_l^c)) \tag{9}$$

其中, D_S^c, D_T^c 分别表示来自源项目和目标项目中类别 c 的样本, L_l^c表示类别 c 对应的局部域鉴别器损失. 由此, 可得动态分布因子的计算公式如下:

$$\hat{\omega} = \frac{A_g(D_S, D_T)}{A_g(D_S, D_T) + \frac{1}{C}\sum_{c=1}^{C} A_l(D_S^c, D_T^c)} \tag{10}$$

初始化阶段, 我们设定ω为 1. 之后, 在每一次迭代过程中, 根据全局和局部域鉴别器的损失函数更新ω的值, 直至模型收敛.

2.5 算法描述

综合所有组成部分, 本文的总体优化目标为:

$$L(\theta_f, \theta_y, \theta_d, \theta_d^c|_{c=1}^C) = L_y - \lambda((1-\omega)L_g + \omega L_l) \tag{11}$$

其中, λ是一个权衡参数. 当$\omega \to 0$时, 表示边缘分布对齐更重要, 当$\omega \to 1$时, 表示两个域之间的全局分布相对较小, 条件分布对齐占主导地位. 因此, 通过学习动态分布因子ω, 可以应用于不同的领域适应场景.

结合领域对抗损失, 领域分类器与其余组件之间具有对抗关系, 模型优化目标如下:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} L(\theta_f, \theta_y, \theta_d, \theta_d^c) \tag{12}$$

$$(\hat{\theta}_d, \hat{\theta}_d^c) = \arg\max_{\theta_d, \theta_d^c} L(\theta_f, \theta_y, \theta_d, \theta_d^c) \tag{13}$$

其中, $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d, \hat{\theta}_d^c$分别表示特征映射、监督分类损失、全局域鉴别器、局部域鉴别器的参数. 以上优化目标无法直接通过梯度下降优化, 所以, 我们在域鉴别器之前加入了梯度反转层 GRL[26]. 加入梯度反转层后的优化目标更改为:

$$(\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d, \hat{\theta}_d^c) = \arg\min_{\theta_f, \theta_y, \theta_d, \theta_d^c} L(\theta_f, \theta_y, \theta_d, \theta_d^c) \tag{14}$$

结合以上优化目标, 本文选取 Adam[36]对模型进行参数优化, 其为一个基于随机梯度下降的优化器, 可以使模型在较大的学习率下快速收敛. 本文的算法描述如算法 1.

算法 1. DPLD 算法.

输入: 带有标签的源项目数据集, 目标项目数据集.
输出: 无标签的目标项目数据集对应的预测标签.

1) 初始化模型参数
2) **repeat**
3) **for** $(x_s, y_s) \in (X_S, Y_S)$, $(x_t, y_t) \in (X_T, Y_T)$ **do**
4) 根据式 (1)、式 (2) 得到项目映射后的特征向量
5) 根据特征向量训练得到监督分类损失L_y, 建立预测模型
6) 计算全局域鉴别器损失L_g
7) 计算局部域鉴别器损失L_l
8) 模型的总体优化目标: $L=L_y-\lambda((1-\omega)L_g+\omega L_l)$
9) 反向传播, *Adam* 更新参数
9) until converage
10) 目标数据集的预测标签$\leftarrow$分类模型

3 实验

为验证所提方法 DPLD 的有效性, 我们在 PROMISE 数据集上进行实验, 并给出了相应的实验结果和分析.

实验环境配置为 2.70 GHz Intel(R) Core(TM) i5-11400H CPU, 16 GB RAM 和 Windows 11. 相关算法是在 Anaconda 的 PyTorch 环境下用 Python 3.7 实现的. 我们主要研究以下问题.

RQ1: 所提的 DPLD 方法是否优于其他项目内或跨项目软件缺陷预测方法?

RQ2: 本文采用的动态分布对齐方法是否优于其他分布匹配模型?

RQ3: 本文采用的伪标签学习方法是否提升了模型的预测性能?

3.1 数据集

实验中采用由 Jureczko 等人收集的公开数据集 PROMISE[37], 选取了数据集中 12 个基于 Java 语言的软件项目, 每个项目均以类为预测粒度. 每个类中包含 20 种度量元, 这些度量元主要关注代码复杂度、面向对象程序中固有的封装、继承、多态等特性. 实验所用数据集的详细信息如表 1 所示.

表 1 实验所用的缺陷数据集信息

项目	度量元个数	样本总数	缺陷个数	不平衡率 (%)
Ant	20	745	166	3.49
Camel	20	965	188	4.13
Ivy	20	352	40	7.80
Jedit	20	306	79	2.95
Log4j	20	135	34	2.97
Lucene	20	340	198	0.69
Poi	20	442	281	0.57
Synapse	20	256	86	1.98
Tomcat	20	858	77	10.14
Velocity	20	229	78	1.94
Xalan	20	723	110	5.57
Xerces	20	453	69	5.57

3.2 对比方法

LR: 我们进行了项目内缺陷预测的比较实验, 对比传统的 LR 方法和本文提出的 DPLD 方法. 传统的 LR 方法直接使用 LR 分类器对缺陷样本进行分类预测.

NN-filter[14]: 该方法根据项目间实例特征的相似性为目标项目构建合适的训练集, 然后进行目标项目模型的训练并将训练好的模型用于目标项目新样本的缺陷预测.

DMDA_JFR[13]: 该方法首先利用编码器学习局部和全局特征表示, 其次在每个堆栈层学习后进行特征空间的渐次对齐.

TCA+[11]: 基于特征的迁移学习方法, 结合数据归一化和迁移成分分析方法用于缺陷预测.

JDA[19]: 该方法借助降维过程共同匹配边缘分布和条件分布, 并构建了有效且稳健的新特征表示.

BDA[21]: 该方法基于分布匹配, 能够自适应地为两个领域之间的分布差异分配权重, 以使源数据和目标数据变得相似.

3.3 评价指标

在软件缺陷预测领域, 一般会利用混淆矩阵来计算性能评价指标, 如表 2 所示. 其中 TP 表示正确预测为有缺陷的样本数, FP 表示错误预测为有缺陷的样本数, FN 表示错误预测为无缺陷的样本数, TN 表示正确预测为无缺陷的样本数.

表 2 混淆矩阵

类别	预测有缺陷	预测无缺陷
真实有缺陷	TP	FN
真实无缺陷	FP	TN

为了有效评估所提 DPLD 算法的预测性能, 本文选用了 F-measure 和 AUC 两个性能指标. F-measure 是一个综合评价指标, 用于平衡召回率和精确度. F-measure 的取值范围在 0–1 之间, 数值越高表示模型的分类效果越好. AUC 是 ROC 曲线下方的面积, 是判断二分类预测模型性能高低的重要指标. ROC 曲线的横坐标表示假阳率 (pf), 纵坐标表示真阳率 (pd). 当前者数值越大且后者数值越小时, AUC 的值越大, 表明模型的预测性能越佳. 公式定义如下:

$$Precision = \frac{TP}{TP+FP} \tag{15}$$

$$Recall = pd = \frac{TP}{TP+FN} \tag{16}$$

$$F\text{-measure} = \frac{2\times Recall\times Precision}{Recall+Precision} = \frac{2\times TP}{2\times TP+FP+FN} \tag{17}$$

3.4 实验设置

本文使用公开的数据集 PROMISE 来验证提出的算法. PROMISE 数据集中包含 12 个项目, 每次实验中, 我们选择一个项目作为目标项目, 其余的项目依次作为源项目参与训练, 最后以本次实验中所有结果的平均值作为目标项目的最终预测结果. 每种预测组合随机重复实验 50 次. 此外, 由于本方法针对少量有标签的目标项目样本, 因此我们从目标项目中提取有标签的样本, 并将其纳入模型训练的流程中. 实验中随机选取 20% 的目标项目作为有标签的样本集, 剩下的 80%

作为无标签样本集.

3.5 实验结果与分析

RQ1: 所提的 DPLD 方法是否优于其他项目内或跨项目软件缺陷预测方法?

本文 DPLD 方法与其他项目内和跨项目预测方法 (LR、NN-filter、DMDA_JFR) 在 *F*-measure 和 AUC 上的性能表现如表 3、表 4 所示.

表 3 本文方法与项目内和跨项目缺陷预测方法在 *F*-measure 上的结果

目标项目	LR	NN-filter	DMDA_JFR	DPLD
Ant	0.453	0.438	0.608	**0.611**
Camel	0.412	0.313	0.516	**0.528**
Ivy	**0.587**	0.314	0.425	0.533
Jedit	0.584	0.364	0.534	**0.613**
Log4j	0.565	0.485	0.467	**0.623**
Lucene	0.382	0.510	**0.604**	0.521
Poi	0.374	0.336	0.484	**0.686**
Synapse	0.457	0.408	0.546	**0.593**
Tomcat	0.604	0.250	0.494	**0.654**
Velocity	0.454	0.418	**0.623**	0.552
Xalan	0.514	0.331	0.501	**0.543**
Xerces	0.506	0.307	0.526	**0.574**
Average	0.491	0.373	0.527	0.586

表 4 本文方法与项目内和跨项目缺陷预测方法在 AUC 上的结果

目标项目	LR	NN-filter	DMDA_JFR	DPLD
Ant	0.566	0.649	0.652	**0.712**
Camel	0.432	0.583	0.521	**0.628**
Ivy	0.549	**0.698**	0.429	0.646
Jedit	0.570	0.545	0.568	**0.615**
Log4j	0.610	0.592	0.546	**0.735**
Lucene	0.481	0.619	0.632	**0.673**
Poi	0.437	0.550	0.512	**0.631**
Synapse	0.563	0.535	0.567	**0.618**
Tomcat	**0.718**	0.645	0.637	0.648
Velocity	0.512	0.533	**0.630**	0.625
Xalan	0.645	0.617	0.534	**0.762**
Xerces	0.547	0.573	0.536	**0.633**
Average	0.553	0.595	0.564	0.661

从表 3、表 4 中可以看出, 在大多数情况下, 本文方法表现出比其他基线方法更高的性能. 与项目内缺陷预测方法 LR 相比, *F*-measure、AUC 分别提升了 19.35%, 19.53%. 这是因为大部分缺陷数据集中实例的个数都是比较少的, 实例个数过少使得模型无法有效地学习具有高度泛化性的预测模型, 甚至面临类别不平衡和过拟合问题的严峻挑战. 因此, 通过深入地学习并构建不同项目间的分布映射关系, 尝试推断并拟合其他项目的缺陷分布, 有时可以获得更好的泛化效果. 此外, 与跨项目方法 NN-filter、DMDA_JFR 相比, *F*-measure 分别提升了 57.1%, 11.2%, AUC 分别提升了 11.09%, 17.2%. 这是因为 NN-filter 和 DMDA_JFR 方法在构建模型时, 只使用了源项目和目标项目之间的公共特征集, 忽略了其他特征中包含的有效信息, 而本文方法不仅借助边缘分布和条件分布同时考虑项目间的全局特征和局部特征, 而且动态地衡量了两种分布的贡献, 更有效地挖掘了项目中的特征信息, 提升了目标项目缺陷预测的性能.

RQ2: 本文采用的动态分布对齐方法是否优于其他分布匹配模型?

表 5、表 6 分别表示 DPLD 方法与其他分布匹配模型 (TCA+、JDA、BDA) 在 *F*-measure 和 AUC 上的结果.

表 5 本文方法与分布匹配算法在 *F*-measure 上的结果

目标项目	TCA+	JDA	BDA	DPLD
Ant	0.572	0.602	**0.648**	0.611
Camel	0.378	0.501	**0.547**	0.528
Ivy	0.315	0.369	0.392	**0.533**
Jedit	0.553	0.458	0.455	**0.613**
Log4j	0.460	0.519	0.585	**0.623**
Lucene	0.534	0.521	**0.612**	0.521
Poi	0.391	0.430	0.500	**0.686**
Synapse	0.337	0.512	0.561	**0.593**
Tomcat	0.401	0.595	0.615	**0.654**
Velocity	0.519	0.590	**0.652**	0.552
Xalan	0.462	0.495	0.497	**0.543**
Xerces	0.508	0.527	0.572	**0.574**
Average	0.453	0.510	0.553	0.586

表 6 本文方法与分布匹配算法在 AUC 上的结果

目标项目	TCA+	JDA	BDA	DPLD
Ant	0.624	0.641	0.673	**0.712**
Camel	0.550	0.579	0.615	**0.628**
Ivy	0.537	0.531	0.554	**0.646**
Jedit	**0.665**	0.587	0.536	0.615
Log4j	0.581	0.537	0.569	**0.735**
Lucene	0.539	0.565	0.628	**0.673**
Poi	0.495	0.617	**0.648**	0.631
Synapse	0.372	0.605	0.576	**0.618**
Tomcat	0.515	0.630	0.613	**0.648**
Velocity	0.572	0.559	**0.671**	0.625
Xalan	0.524	0.568	0.535	**0.762**
Xerces	0.603	0.586	0.614	**0.633**
Average	0.548	0.584	0.603	0.661

从表 5、表 6 可见, 大部分情况下, 本文 DPLD 方法表现出比其他经典的分布匹配模型更好的预测性能. 在所有项目上, 本文方法相比于 TCA+、JDA、BDA 模型在 *F*-measure 上分别提升了 29.36%、14.9%、5.97%, 在 AUC 上分别提升了 20.62%、13.18%、

9.62%. 这是因为 TCA+模型只考虑了项目间的边缘分布对齐, 忽视了条件分布的影响; JDA 模型虽然实现了边缘分布和条件分布的对齐, 但是没有考虑分布之间的相对重要性; 相对于 BDA 模型, 本文 DPLD 方法充分挖掘了目标项目的伪标签信息, 有效提升了预测性能. 本文 DPLD 方法与分布匹配模型的对比实验充分说明了本文同时考虑项目间的边缘分布和条件分布并动态调整分布的权重对提高模型性能具有关键作用, 考虑分布的动态对齐是非常有必要且有意义的.

RQ3: 本文采用的伪标签学习方法是否提升了模型的预测性能?

为了探究本文采用的伪标签学习方法对模型性能的提升效果, 我们进行了两种不同情况的验证: 一种是几何伪标签学习方法, 另一种是直接使用共享分类器进行伪标签的预测. 然后, 我们比较了两种情况下模型在评价指标 *F*-measure 和 AUC 上的预测结果, 结果如图 2 所示.

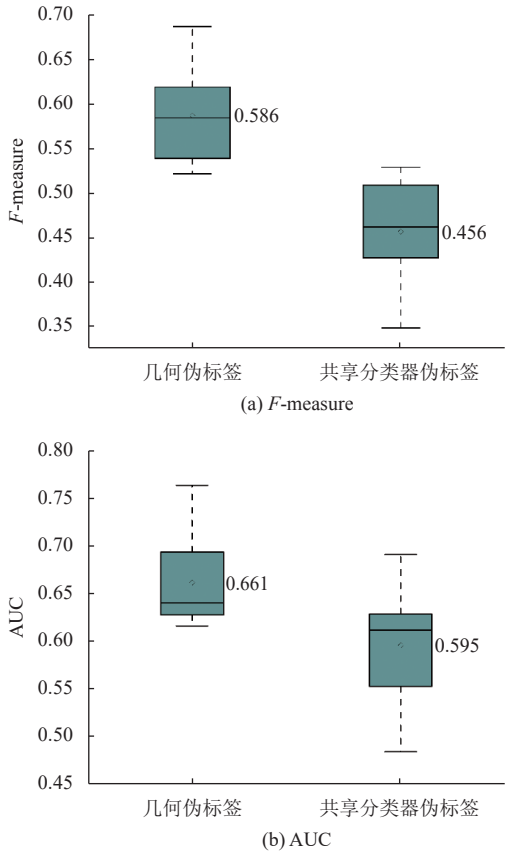

图 2 不同的伪标签学习方法对模型性能的影响

从图 2 可以明显看出, 使用共享分类器预测的伪标签所得的结果比使用几何相似度学习的伪标签所得的结果要差得多. 从均值上看, *F*-measure 平均低了 28.51%, 在 AUC 上平均低了 11.09%. 这充分说明了本文采用的几何伪标签学习方法可以更好地提高目标项目伪标签的准确性, 提升预测模型的性能.

4 结论

软件缺陷预测一直是软件工程和数据挖掘领域的热门议题, 因为它对确保软件质量至关重要. 针对现有研究中忽视了分布差异的动态性和伪标签选择不合适问题, 本文提出了一种基于动态分布对齐和伪标签学习的跨项目缺陷预测方法, 主要通过域鉴别器分别构成域对齐和类别对齐模块来学习域不变的特征表示, 消除项目间的分布差异. 同时借助类别间的几何相似性更新伪标签, 提高伪标签的准确性. 在公开缺陷预测数据集 PROMISE 上的实验表明, 我们的方法可以有效减少项目间的分布差异, 并且与基线方法相比具有更好的预测性能. 本文方法目前仅适用于源项目数据和目标数据的一对一缺陷预测, 我们计划在后续的工作中重点探讨多对一的跨项目缺陷预测.

参考文献

1 纪兴哲, 邵培南. 面向软件缺陷预测的过采样方法. 计算机系统应用, 2022, 31(1): 242–248. [doi: 10.15888/j.cnki.csa.008284]

2 陈凯, 邵培南. 基于深度学习的软件缺陷预测模型. 计算机系统应用, 2021, 30(1): 29–37. [doi: 10.15888/j.cnki.csa.007726]

3 Laradji IH, Alshayeb M, Ghouti L. Software defect prediction using ensemble learning on selected features. Information and Software Technology, 2015, 58: 388–402. [doi: 10.1016/j.infsof.2014.07.005]

4 Azzeh M, Elsheikh Y, Nassif AB, *et al*. Examining the performance of kernel methods for software defect prediction based on support vector machine. Science of Computer Programming, 2023, 226: 102916. [doi: 10.1016/j.scico.2022.102916]

5 Wang S, Liu TY, Tan L. Automatically learning semantic features for defect prediction. Proceedings of the 38th IEEE/ACM International Conference on Software Engineering. Austin: IEEE, 2016. 297–308. [doi: 10.1145/2884781.2884804]

6 Hosseini S, Turhan B, Gunarathna D. A systematic literature review and meta-analysis on cross project defect prediction.

IEEE Transactions on Software Engineering, 2019, 45(2): 111–147. [doi: 10.1109/TSE.2017.2770124]

7 Jin C. Cross-project software defect prediction based on domain adaptation learning and optimization. Expert Systems with Applications, 2021, 171: 114637. [doi: 10.1016/j.eswa.2021.114637]

8 陈曙, 叶俊民, 刘童. 一种基于领域适配的跨项目软件缺陷预测方法. 软件学报, 2020, 31(2): 266–281. [doi: 10.13328/j.cnki.jos.005632]

9 李伟漳, 陈翔, 张恒伟, 等. 一种基于同步语义对齐的异构缺陷预测方法. 软件学报, 2023, 34(6): 2669–2689. [doi: 10.13328/j.cnki.jos.006495]

10 Pan SJ, Tsang IW, Kwok JT, *et al*. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 2011, 22(2): 199–210. [doi: 10.1109/TNN.2010.2091281]

11 Nam J, Pan SJ, Kim S. Transfer defect learning. Proceedings of the 35th International Conference on Software Engineering. San Francisco: IEEE, 2013. 382–391. [doi: 10.1109/ICSE.2013.6606584]

12 Satpal S, Sarawagi S. Domain adaptation of conditional probability models via feature subsetting. Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. Warsaw: Springer, 2007. 224–235. [doi: 10.1007/978-3-540-74976-9_23]

13 Zou QY, Lu L, Yang ZY, *et al*. Joint feature representation learning and progressive distribution matching for cross-project defect prediction. Information and Software Technology, 2021, 137: 106588. [doi: 10.1016/j.infsof.2021.106588]

14 Turhan B, Menzies T, Bener AB, *et al*. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5): 540–578. [doi: 10.1007/s10664-008-9103-7]

15 He ZM, Shu FD, Yang Y, *et al*. An investigation on the feasibility of cross-project defect prediction. Automated Software Engineering, 2012, 19(2): 167–199. [doi: 10.1007/s10515-011-0090-3]

16 Ma Y, Luo GC, Zeng X, *et al*. Transfer learning for cross-company software defect prediction. Information and Software Technology, 2012, 54(3): 248–256. [doi: 10.1016/j.infsof.2011.09.007]

17 Chen L, Fang B, Shang ZW, *et al*. Negative samples reduction in cross-company software defects prediction. Information and Software Technology, 2015, 62: 67–77. [doi: 10.1016/j.infsof.2015.01.014]

18 Ryu D, Choi O, Baik J. Value-cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2016, 21(1): 43–71. [doi: 10.1007/s10664-014-9346-4]

19 Long MS, Wang JM, Ding GG, *et al*. Transfer feature learning with joint distribution adaptation. Proceedings of the 2013 IEEE International Conference on Computer Vision. Sydney: IEEE, 2013. 2200–2207. [doi: 10.1109/ICCV.2013.274]

20 Wu F, Jing XY, Sun Y, *et al*. Cross-project and within-project semisupervised software defect prediction: A unified approach. IEEE Transactions on Reliability, 2018, 67(2): 581–597. [doi: 10.1109/TR.2018.2804922]

21 Xu Z, Pang S, Zhang T, *et al*. Cross project defect prediction via balanced distribution adaptation based transfer learning. Journal of Computer Science and Technology, 2019, 34(5): 1039–1062. [doi: 10.1007/s11390-019-1959-z]

22 Zhuang FZ, Cheng XH, Luo P, *et al*. Supervised representation learning: Transfer learning with deep autoencoders. Proceedings of the 24th International Joint Conference on Artificial Intelligence. Buenos Aires: IJCAI, 2015. 4119–4125.

23 Li JJ, Jing MM, Lu K, *et al*. Locality preserving joint transfer for domain adaptation. IEEE Transactions on Image Processing, 2019, 28(12): 6103–6115. [doi: 10.1109/TIP.2019.2924174]

24 Chen SJ, Jia X, He JZ, *et al*. Semi-supervised domain adaptation based on dual-level domain mixing for semantic segmentation. Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Nashville: IEEE, 2021. 11013–11022. [doi: 10.1109/CVPR46437.2021.01087]

25 Gao YP, Lin KY, Yan JK, *et al*. AsyFOD: An asymmetric adaptation paradigm for few-shot domain adaptive object detection. Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Vancouver: IEEE, 2023. 3261–3271. [doi: 10.1109/CVPR52729.2023.00318]

26 Ganin Y, Lempitsky V. Unsupervised domain adaptation by backpropagation. Proceedings of the 32nd International Conference on Machine Learning. Lille: PMLR, 2015. 1180–1189.

27 Yu CH, Wang JD, Chen YQ, *et al*. Transfer learning with dynamic adversarial adaptation network. Proceedings of the 2019 IEEE International Conference on Data Mining. Beijing: IEEE, 2019. 778–786. [doi: 10.1109/ICDM.2019.00088]

28 Zhou Q, Zhou WA, Wang SR, *et al*. Duplex adversarial networks for multiple-source domain adaptation. Knowledge-based Systems, 2021, 211: 106569. [doi: 10.1016/j.knosys.2020.106569]

29 Sun T, Lu C, Ling HB. Domain adaptation with adversarial training on penultimate activations. Proceedings of the 37th AAAI Conference on Artificial Intelligence. Washington: AAAI, 2023. 9935–9943.

30 Bennin KE, Keung J, Phannachitta P, *et al*. MAHAKIL: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Transactions on Software Engineering, 2018, 44(6): 534–550. [doi: 10.1109/TSE.2017.2731766]

31 Feng S, Keung J, Yu X, *et al*. COSTE: Complexity-based oversampling technique to alleviate the class imbalance problem in software defect prediction. Information and Software Technology, 2021, 129: 106432. [doi: 10.1016/j.infsof.2020.106432]

32 Limsettho N, Bennin KE, Keung JW, *et al*. Cross project defect prediction using class distribution estimation and oversampling. Information and Software Technology, 2018, 100: 87–102. [doi: 10.1016/j.infsof.2018.04.001]

33 Japkowicz N, Stephen S. The class imbalance problem: A systematic study. Intelligent Data Analysis, 2002, 6(5): 429–449. [doi: 10.3233/IDA-2002-6504]

34 Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 1958, 65(6): 386–408. [doi: 10.1037/h0042519]

35 Wang KF, Gou C, Duan YJ, *et al*. Generative adversarial networks: Introduction and outlook. IEEE/CAA Journal of Automatica Sinica, 2017, 4(4): 588–598. [doi: 10.1109/JAS.2017.7510583]

36 Huang JY, Smola AJ, Gretton A, *et al*. Correcting sample selection bias by unlabeled data. Proceedings of the 19th International Conference on Neural Information Processing Systems. Vancouver: MIT Press, 2006. 601–608.

37 Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction. Proceedings of the 6th International Conference on Predictive Models in Software Engineering. Timişoara: ACM, 2010. 9. [doi: 10.1145/1868328.1868342]

(校对责编: 张重毅)