软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(6):2097−2112 [doi: 10.13328/j.cnki.jos.006571]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种采用对抗学习的跨项目缺陷预测方法*

邢 颖[1], 钱晓萌[2], 管 宇[1], 章世豪[1], 赵梦赐[1], 林婉婷[1]

[1](北京邮电大学 人工智能学院, 北京 100876)

[2](北京邮电大学 现代邮政学院(自动化学院), 北京 100876)

通信作者: 邢颖, E-mail: xingying@bupt.edu.cn

摘 要: 跨项目缺陷预测(cross-project defect prediction, CPDP)已经成为软件工程数据挖掘领域的一个重要研究方向, 它利用其他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着数据分布的差异, 导致跨项目预测效果不佳. 基于生成式对抗网络(generative adversarial network, GAN)中的对抗学习思想, 在鉴别器的作用下, 通过改变目标项目特征的分布, 使其接近于源项目特征的分布, 从而提升跨项目缺陷预测的性能. 具体来说, 提出的抽象连续生成式对抗网络(abstract continuous generative adversarial network, AC-GAN)方法包括数据处理和模型构建两个阶段: (1) 首先将源项目和目标项目的代码转换为抽象语法树(abstract syntax tree, AST)的形式, 然后以深度优先方式遍历抽象语法树得出节点序列, 再使用连续词袋模型(continuous bag-of-words model, CBOW)生成词向量, 依据词向量表将节点序列转化为数值向量; (2) 处理后的数值向量被送入基于 GAN 网络结构的模型进行特征提取和数据迁移, 然后使用二分类器来判断目标项目代码文件是否有缺陷. AC-GAN 方法在 15 组源-目标项目对上进行了对比实验, 实验结果表明了该方法的有效性.

关键词: 跨项目缺陷预测; 生成式对抗网络; 连续词袋模型; 抽象语法树

中图法分类号: TP311

中文引用格式: 邢颖, 钱晓萌, 管宇, 章世豪, 赵梦赐, 林婉婷. 一种采用对抗学习的跨项目缺陷预测方法. 软件学报, 2022, 33(6): 2097−2112. http://www.jos.org.cn/1000-9825/6571.htm

英文引用格式: Xing Y, Qian XM, Guan Y, Zhang SH, Zhao MC, Lin WT. Cross-project Defect Prediction Method Using Adversarial Learning. Ruan Jian Xue Bao/Journal of Software, 2022, 33(6): 2097−2112 (in Chinese). http://www.jos.org.cn/1000-9825/6571.htm

Cross-project Defect Prediction Method Using Adversarial Learning

XING Ying[1], QIAN Xiao-Meng[2], GUAN Yu[1], ZHANG Shi-Hao[1], ZHAO Meng-Ci[1], LIN Wan-Ting[1]

[1](School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2](School of Modern Post (School of Automation), Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Cross-project defect prediction (CPDP) has become an important research direction in data mining of software engineering, which uses the defective codes of other projects to build prediction models and solves the problem of insufficient data in the process of model construction. Nevertheless, there is difference in data distribution between the code files of source and target projects, which leads to poor cross-project prediction results. Based on the adversarial learning idea of generative adversarial network (GAN), under the action of discriminator, the distribution of target project features can be changed to make it close to the distribution of source project features, so as to improve the performance of cross-project defect prediction. Specifically, the process of the proposed abstract continuous GAN (AC-GAN) method consists of two stages: Data processing and model construction. First, the source and target project codes are converted into the form of abstract syntax trees (ASTs), and then the ASTs are traversed in a depth-first manner to derive the token sequences. The continuous bag-of-words model (CBOW) is used to generate word vectors, and the token sequences are transformed into

* 基金项目: 国家自然科学基金(61702044); 国家重点研发计划课题(2017YFD0401001)
本文由“系统软件安全”专题特约编辑杨珉教授、张超副教授、宋富副教授、张源副教授推荐.
收稿时间: 2021-09-05; 修改时间: 2021-10-15; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

numeric vectors based on the word vector table. Second, the processed numeric vectors are fed into a GAN structure-based model for feature extraction and data migration. Finally, a binary classifier is used to determine whether the target project code files are defective or not. The AC-GAN method conducted comparison experiments on 15 sets of source-target project pairs, and the experimental results demonstrate the effectiveness of this method.

Key words: cross-project defect prediction; generative adversarial network (GAN); bag-of-words model; abstract syntax tree (AST)

现代软件在人们的日常生活中发挥着越来越重要的作用. 然而, 潜在的软件缺陷对软件的质量和可靠性构成了严重威胁. 潜在的软件缺陷越晚被发现, 修复缺陷的成本就越高, 严重时可能给企业带来巨大的经济损失, 甚至有时会造成人员伤亡[1]. 为了更好地保证软件质量, 研究人员提出了软件缺陷预测(software defect prediction, SDP)技术[2]. SDP 是软件工程中数据挖掘领域的一个研究热点, 它希望在项目开发的早期阶段就能提前发现并改进有缺陷的程序模块, 从而帮助开发人员更快地发现缺陷代码, 以节省开发时间, 提高测试效率.

传统的 SDP 研究用手工提取的静态度量来进行缺陷预测. 程序模块中的源代码作为一种形式语言, 包含丰富的结构信息和语义信息, 手工提取的静态度量主要集中在程序的统计特性上[3], 无法从源代码中捕获这些复杂的信息, 这就使得当有 bug 代码与无 bug 代码在结构上类似时, 手工提取的静态度量无法捕获这些缺陷, 这个局限性是影响缺陷预测准确性的重要因素. 为了克服这一局限性, 一些研究人员利用深度学习从源代码中学习有意义的语义和上下文特征[4−6]. 为了在整个训练和预测过程中不丢失代码的语义和上下文信息, 本文采用抽象语法树(abstract syntax tree, AST)[7]的形式来表示代码, 利用连续词袋模型(continuous bag-of-words model, CBOW)[8]来提取包含上下文信息的词向量, 并将深度学习技术应用于特征提取阶段.

实现基于深度学习的 SDP 研究的前提是获得足够的历史数据来训练预测模型, 但在实际开发过程中, 需要预测的软件项目可能是一个全新的项目, 也可能是一个历史数据较少的项目. 跨项目缺陷预测(cross-project defect prediction, CPDP)[9]因此而被提出, 通过借用其他成熟项目数据来弥补历史数据的不足, 其中, 将成熟项目视为源项目(训练集), 而将被预测项目视为目标项目(测试集).

在 CPDP 研究中, 不同项目的功能、开发人员和开发流程不尽相同, 这将导致源项目和目标项目之间存在数据分布差异, 与训练集和测试集独立相同分布的假设相悖. 因此, 跨项目缺陷预测的性能往往低于项目内缺陷预测. 为了解决这个问题, 很多学者采用机器学习的方法进行了相关研究[10−12], 包括迁移学习和特征选择等. 然而, 这些传统的机器学习方法往往无法学习复杂的特征, 而且损失函数的设计也很复杂. 有实验表明: 对抗学习中的生成式对抗神经网络(generative adversarial network, GAN)[13]作为消除域间差异工具, 能够很好地应用在自然语言处理和图像识别等领域[14−18]. 与传统方法相比, GAN 网络有以下优点: GAN 网络的训练过程采用两种对抗性神经网络作为训练准则, 可以采用反向传播的方法进行训练, 而且该训练不依赖低效的马尔可夫链方法, 也不依赖近似推理, 没有复杂的变分下界, 大大降低了训练难度, 提高了训练效率; GAN 网络训练方式采用的是对抗训练方式, 可以产生更加清晰、真实的样本, 且 GAN 网络利用鉴别器来实现数据迁移, 能够避免迁移学习中损失函数设计的困难. 为了能够更好地解决源项目和目标项目代码文件间的数据分布差异问题, 本文采用了 GAN 网络中的对抗学习思想来训练具有领域适应性的神经网络, 不断改变目标项目特征的分布, 直至其与源项目特征分布相似.

基于上述分析, 为了解决跨项目缺陷预测问题, 本文提出了一种采用对抗学习的抽象连续生成式对抗网络(abstract continuous generative adversarial network, AC-GAN)方法. 其过程如下.

- 首先, 源项目和目标项目中每个模块的代码被解析为抽象语法树, 按深度优先原则遍历出节点序列;
- 接着, 通过词嵌入技术 CBOW 算法生成词向量, 并根据词向量表将节点序列转换为数值向量;
- 然后, 将与源项目相对应的带标记的数值向量作为输入, 对源特征提取器和源分类器进行训练;
- 再以源特征作为真实数据, 目标特征提取器提取的目标特征作为假数据输入到判别器中进行博弈, 通过 GAN 网络的对抗训练, 源数据和目标数据之间的分布差异被缩小;
- 最后, 分布改变后的目标特征被输入到目标分类器中进行缺陷预测.

具体来说, 本文针对跨项目缺陷预测的主要贡献如下:

(1) 通过抽象语法树的形式表征代码、使用 CBOW 算法提取词向量并采用深度学习特征提取器提取特征的方式极大地保留了代码文件中的上下文和语义信息;

(2) 根据 GAN 网络特性, 提出了与 GAN 网络结构相似的 AC-GAN 模型, 通过对抗博弈训练进行数据迁移, 解决源数据和目标数据之间的分布差异问题;

(3) 在 15 对源-目标项目上进行了对比实验, 以评估所提模型的性能.

1 相关工作

本文选择 3 类相关性最强的研究进行讨论, 包括软件缺陷预测、跨项目缺陷预测和对抗学习.

1.1 软件缺陷预测(SDP)

随着人们对软件依赖性的增强, 软件健康的重要性日益凸显, 相关的软件缺陷预测技术也越来越受到重视. 早期的软件缺陷预测研究采用数学统计方法挖掘缺陷, Chang 等人[19]利用行为修正建议来构建低缺陷和高缺陷行为, 并使用负关联规则挖掘技术来构建预测算法. Pradeep 等人[20]提出了模糊规则, 根据所用特征的比例, 选择有用的特征进行预测. 随着人工智能产业的兴起, 很多机器学习方法被尝试用于解决软件缺陷预测问题. Issam 等人[21]将特征选择与集成学习相结合, 提出了平均概率集成的学习方法, 将多个分类器输出的缺陷概率的平均值作为最终结果. He 等人[22]提出了名为 extRF 的学习方法, 该方法通过自训练模型扩展了有监督的随机森林算法, 形成了一个精炼的预测模型. Yang 等人[23]提出了一种两层的集合学习方法, 该方法的内层基于决策树建立随机森林模型, 外层使用随机抽样来训练不同的随机森林进行缺陷预测. Wu 等人[24]利用半监督的结构化字典学习方法, 通过学习总字典和多个子字典来预测软件缺陷. 由于机器学习往往采用手工提取的特征进行预测, 精度受到一定限制, 一些研究人员开始将深度学习的方法引入软件缺陷预测研究中. Yang 等人[4]使用深度信念网络(deep belief network, DBN)[25]从程序源代码的 AST 中遍历 token 向量, 然后从 token 向量中提取特征, 建立软件缺陷预测模型. Wang 等人[5]进一步用唯一的整数标识符映射每个 token, 将 token 向量转化为数值向量, 然后输入 DBN 提取语义特征. Li 等人[6]提出一个基于卷积神经网络(convolutional neural networks, CNNs)[26]的软件缺陷预测框架, 在这个框架中, 他们通过词嵌入算法将标记向量编码为数值向量, 然后使用 CNN 自动地学习语义特征. 实验结果表明, 这些采用深度学习提取程序源代码特征的软件缺陷预测方法都表现出较好的预测性能. 为获得更好的缺陷预测性能, 本文也将深度学习应用于特征提取阶段.

1.2 跨项目缺陷预测(CPDP)

在实际的软件缺陷预测应用中, 由于缺乏历史数据, 很难为新项目建立一个准确的预测模型. 为了解决软件缺陷预测的这一局限性, 研究人员更加关注利用其他成熟项目的数据来进行预测的 CPDP 方法. 常用的 CPDP 方法一般可分为 3 类: 有监督学习、半监督学习和无监督学习. 基于有监督学习的 CPDP 方法是研究者们最热衷的方法, Pan 等人[27]利用迁移成分分析(transfer component analysis, TCA)在保留数据属性的前提下迁移训练数据, 使得源数据与目标数据具有相似的数据分布. Long 等人[28]提出了联合分布适应(joint distribution adaptation, JDA)方法, 该方法在一个有原则的降维过程中共同适应边际分布和条件分布, 并构造出消除了数据分布差异的新特征表示. Turhan 等人[29]提出采用近邻过滤器(nearest neighbor filtering, NNFilter)的方法, 通过将特征选择相似的实例集合在一起, 以构建一个与目标数据集相同的训练集进行训练. Zhou 等人[30]提出了一个同时考虑边缘分布和条件分布, 并自适应地赋予不同权重的平衡分布适应(balanced distribution adaptation, BDA)方法, 该方法基于迁移学习平衡数据分布. 对于基于半监督学习的 CPDP 方法, Xia 等人[31]基于多个候选源项目构建多个基分类器, 并借助遗传算法输出一个最优组合(genetic algorithm, GA)分类器, 最后, 基于 GA 分类器, 并借助 Adaboost 集成方法构建最终模型. Ryu 等人[32]提出了转移成本敏感提升(transfer cost-sensitive boosting, TCSBooST)的方法, 将源项目实例和目标项目中的少量标记实例组合成训练集, 迭代 M 次并构建 M 个基分类器, 最后对基分类器进行加权集成. Wu 等人[33]提出了一种代价敏感核化半监督字典学

习方法(cost-sensitive kernelized semisupervised dictionary learning, CKSDL)方法，该方法通过半监督字典(semisupervised dictionary learning, SDL)技术，利用少量带标签数据和大量无标签数据将源和目标项目置于同一个子空间中，以消除分布差异. 对于基于无监督学习的 CPDP 方法, Zhong 等人[34]借助聚类的方法选择典型模块，再利用辅助信息邀请专家进行标注. Zhang 等人[35]通过频谱聚类的方法将所有程序模块分为两类，然后通过测量每类中度量元的数值之和来确定是否有缺陷. 本文拟采用基于有监督学习的 CPDP 方法，以期更客观地评价所提方法的性能.

1.3 对抗学习

目前，GAN 网络中的对抗学习思想被广泛应用于需要消除数据分布差异的场景中. Yaroslav 等人[14]提出了基于对抗学习思想的域对抗性神经网络(domain-adversarial neural network, DANN)，该网络通过使用标签预测损失和域判别损失，实现了不同域之间的特征选择. Zhu 等人[15]提出了周期一致性对抗网络(cycle-consistent adversarial network, CycleGAN)，该网络通过引入周期一致性损失来强制从一个域 X 到另一个域 Y 的映射大致相同，从而解决了未配对训练数据无法用于图像到图像转换的问题. Samaneh 等人[16]构建了一个多内容生成对抗网络(multi-content generative adversarial network, MC-GAN)模型，该模型由用于预测粗糙字形的 Glyphnet 和用于预测最终字形颜色和纹理的 Ornanet 组成，通过使用 Glyphne 来预测字形掩码，并用 Ornanet 来微调字形颜色和装饰，实现了字体风格的迁移. Yunjey 等人[17]将生成式对抗网络设计成星形，并增加了域控制信息，以实现图像中多域之间的风格转换. Jesse 等人[18]提出了对抗性神经音频合成(adversarial neural audio synthesis, GANsynth)模型，通过使用一系列频谱训练 GAN 网络，成功生成了高保真的音频. 本文计划把 GAN 网络思想和结构应用于 CPDP 领域，以解决源数据和目标数据之间的差异问题.

2 研究方法

图 1 展示了本文提出的 AC-GAN 方法的整体框架. 具体而言，它包含两个阶段：数据处理和模型构建.

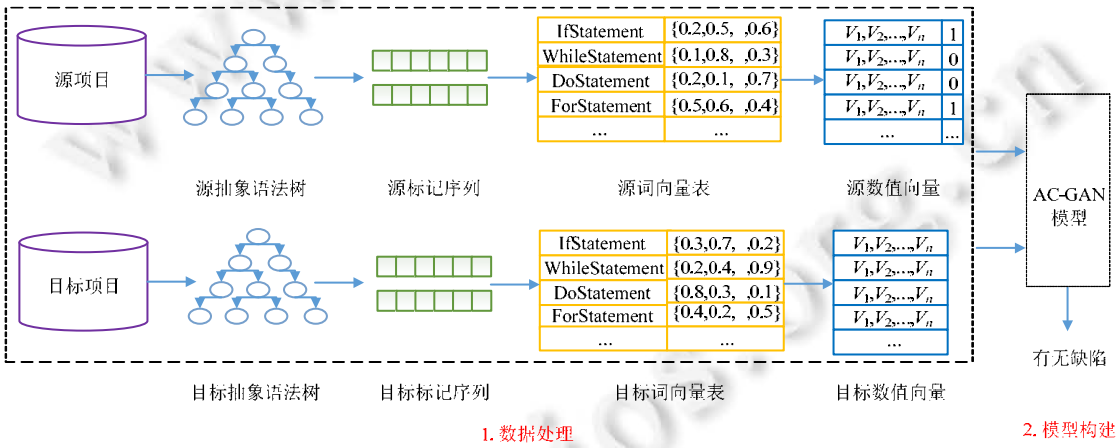


图 1 AC-GAN 方法的框架

具体来说，数据处理阶段包括 3 个步骤.

(1) 代码解析和序列生成(图 1 中，项目到标记序列过程);

(2) 词向量提取(图 1 中，标记序列到词向量表过程);

(3) 数据维度规整化(图 1 中，词向量表到数值向量过程).

而在模型构建阶段，则主要完成预测模型的搭建以及对目标项目的预测. 该阶段可以完成对目标项目代码文件的分类，即，将该代码文件预测为有缺陷倾向性(defect-proneness, DP)或无缺陷倾向性(non-defect-proneness, NDP).

2.1 缺陷数据处理

源项目和目标项目的代码文件是不能直接输入到特征提取器里提取特征的, 需要先对代码文件进行处理, 直至转换为可用于计算的具有统一长度的数值向量.

2.1.1 代码解析和序列生成

抽象语法树是代码程序的树状表示法. 这种树状结构由众多具有语义的节点组成, 在特定的抽象提取规则下, 最大限度地保留了原始数据句法规则的特征, 从而满足了本文对 CPDP 任务解决方案的基本要求. 因此, 本文首先需要将源项目和目标项目的代码文件转换为抽象语法树的形式, 以便后续提取语义和上下文特征. 在这个过程中, 本文使用一个开源的 Python 依赖包 Javalang[36]作为抽象语法树提取工具. 图 2 是 Java 代码片段以及相应的抽象语法树的例子.

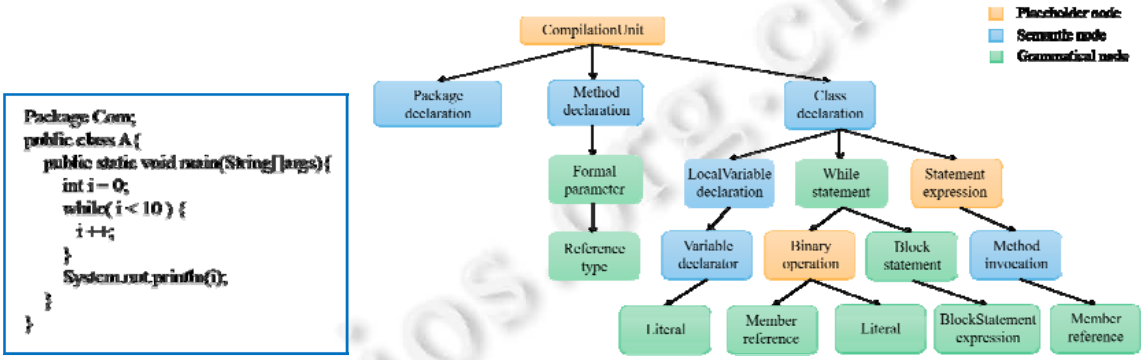


图 2 Java 代码片段及对应的抽象语法树

如图 2 所示, 抽象语法树由 3 种类型的节点组成: 占位符节点(placeholder node)、语义节点(semantic node)和语法节点(grammatical node). 占位符节点不是函数的实际组成部分, 但它们将函数的组件连接起来形成树. 所有的 AST 都有占位符节点, 比如 CompilationUnit, 它代表每棵树的根节点. 语义节点指的是函数调用节点和声明类型的所有节点, 例如: MethodInvocation 表示函数调用, ClassDeclaration 表示类声明. 语法节点是包含控制流元素和运算符的语法元素. 控制流元素来自于 while/do while 语句、if/else 语句等. 如果一个函数包含一个 if/else 语句, 那么它的 AST 中就有 IfStatement 节点.

在获得源项目和目标项目的抽象语法树后, 本文以深度优先的方式遍历抽象语法树的所有节点, 来获得每个代码文件对应的节点序列. 为了在保留语义信息的同时更简洁地表示代码文件, 本文只选择语法节点中表示控制流的节点和所有语义节点作为表征节点[37].

2.1.2 词向量生成

从抽象语法树中遍历出的节点序列是字符串序列, 无法直接输入到编码器中提取特征, 因此必须先要把节点序列转换成数值向量[38]. 在自然语言处理中, 通常使用词向量来表示单词, 在软件缺陷预测时, 可以将抽象语法树的节点视为自然语言中的一个单词, 生成相应的词向量来转化节点序列. 本文采用 CBOW 模型作为词向量生成算法.

CBOW 的思想是使用单词的上下文环境来预测单词, 以这种方式来训练神经网络. 连续词袋模型的模型结构如图 3 所示, 共由 3 层组成.

(1) 输入层(input): 假设一个词为代码文件中第 n 个单词, 输入层输入的是该目标单词上下文的单词. 上下文单词均用独热码的形式来表示, 即在构造字典以后, 单词都被表示为一个与字典长度一样的向量, 上文单词在字典中对应的序号为 n–1, 那么上文单词的独热码序列第 n–1 位为 1, 其余为 0; 同理, 下文单词在字典中对应的序号为 n+1, 则下文单词的独热码序列第 n+1 位为 1, 其余为 0;

(2) 投影层(projection): 上下文单词都乘以一个权值矩阵, 输出一个隐层向量, 将这些上下文单词输出的向量相加取平均, 得到投影层输出的向量;

(3) 输出层(output): 投影层输出的向量乘以一个权值矩阵, 得到输出层的输出向量; 然后对向量进行 *softmax* 归一化处理, 得到目标单词的分布式概率; 将概率最大的数字对应的序号作为目标单词在字典中的标号; 通过输出层 *softmax* 归一化之后的向量与目标单词的独热码进行比较、计算误差, 以反向传播的方式不断更新 CBOW 中的两个权值矩阵.

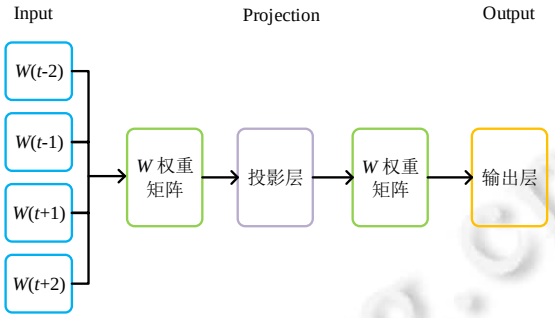


图 3 CBOW 的结构

将源项目和目标项目对应的节点序列分别输入到CBOW模型中进行训练, 最终得到的输入层与隐层间的权值矩阵便是词嵌入矩阵, 即训练想要的结果. 因为第 i 个单词独热码只有第 i 位为 1, 其他位为 0, 当它和权重矩阵相乘时, 输出的结果即为权值矩阵的第 i 行, 即这个单词可以使用权重矩阵的第 i 行表示. 这样一来, 独热码表示的词向量长度就是字典长度. CBOW 词向量表可以通过将目标单词的独热码与词嵌入矩阵相乘而得到.

2.1.3 数据维度规整化

在得到源项目和目标项目各自的词向量表之后, 可以把节点序列中的单词全部按照词向量表转化成词向量矩阵, 即后边要输入到特征提取器中的数值向量. 但是由于每个代码文件节点序列长度不同, 数值向量长度也有差异, 无法直接输入到特征提取器中提取特征, 所以需要对数值向量进行规整化. 填充和截断是处理各种长度的向量并提供输入向量的统一长度的标准实践方式. 为了平衡数值向量的长度, 本文统一规定数值向量为 32 维, 超过的进行截断, 不足的补零.

2.2 缺陷模型构建

在完成数据处理之后, 就可以进入构建缺陷预测模型的阶段了, 本节分两块来介绍本文的模型: 原始 GAN 网络和 AC-GAN 模型.

2.2.1 原始 GAN 网络

GAN 网络的思想是一种二人博弈思想(two-player game), 博弈双方的利益之和是一个常数, 一方利益多, 则另一方利益相对就少. 有两个这样的博弈者: 一个人名字是生成模型, 另一个人名字是判别模型. 如图 4 所示: 生成模型可看作是一个样本生成器(G), 输入一个噪声序列, 然后把它包装成一个逼真的假样本, 也就是输出; 判别模型则是一个二分类鉴别器(D), 以此来判断输入的样本是真是假. 两种模型都是神经网络模型, 生成网络的目的是尽可能地使得生成样本逼近真实样本, 混淆判别网络, 使判别网络无法识别假样本. 而判别网络的目的则是甄别虚假样本, 使其无法混入真实样本中. 两者进行博弈, 直至假样本分布与真实样本基本相同.

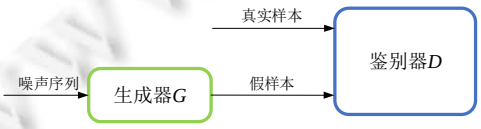


图 4 原始 GAN 架构

原始 GAN 网络的公式如下:

$$V(D,G)=E_{x\sim P_{data}}[\log D(x)]+E_{y\sim P_g}[\log(1-D(y))] \tag{1}$$

其中,

- x 表示真实样本;
- y 表示生成的假样本;
- P_{data} 表示真实样本的分布;
- P_g 表示生成器 G 生成的假样本的分布;
- $D(x)$表示鉴别器判断真实数据是否真实的概率;
- $D(y)$则表示鉴别器判断假样本是否真实的概率.

从公式(1)可以看出: 当生成器 G 固定不变时, $V(G,D)$就表示 P_{data} 和 P_g 之间的差异. 为了找到更接近的分布, 需要在鉴别器 D 性能最强的前提下选取最优的生成器 G^*. 也就是说, 即使鉴别器的鉴别能力达到了最强, 也无法区分生成器生成的假样本与真实样本. 此时两个分布之间的差异最小, 如公式(2)所示:

$$G^*=\arg\min_G\max_D V(G,D) \tag{1}$$

其中, arg 表示定义域的子集, 该子集中任一元素都可使 $V(G,D)$在鉴别器最强情况下取得最小值. Goodfellow 等人[39]证明: 在 $P_{data}=P_g$ 时, 即鉴别器输出等于 0.5 时, GAN 具有全局最优性, 生成器 G 性能最优. 此时, 生成器生成的假样本最大概率地接近真实样本, 使得鉴别器 D 不能区分真实样本和假样本的分布. 此时, 判别网络输出为 0.5, 判断全是真实样本.

2.2.2 AC-GAN 模型

在数据处理阶段, 源项目和目标项目的代码文件已经被转化成了统一长度的数值向量的形式, 可以被输入到特征提取器中提取特征. 本文依据跨项目缺陷预测的实际要求对原始 GAN 网络的架构进行了转化, 提出了 AC-GAN 模型. 其原理就是把训练好的源特征提取器输出的源特征当作真实样本, 而把目标特征提取器当作生成模型, 输出的目标特征当作生成的虚假样本, 再把源特征和目标特征输入到鉴别器, 即判别模型中进行博弈. 当鉴别器再也无法区分源特征和目标特征时, 证明两种特征的分布已经基本相同, 则消除了源数据与目标数据之间的分布差异.

循环神经网络(recurrent neural network, RNN)在自然语言处理领域取得了很好的效果[40]. 长短期记忆神经网络(long short-term memory, LSTM)[41]是 RNN 网络的优化, 解决了 RNN 网络在长序列中梯度消失的问题, 可以学习长期依赖信息. 由于代码是具有逻辑和语义的结构, 具有紧密耦合的性质, 因此产生缺陷的代码片段通常与其上下文都相关, 选用的特征提取器应该能够对输入序列进行正向和反向处理. 因此, 本文采用了双向长短期记忆网络(bidirectional long short-term memory, BLSTM)[42], 即两层 LSTM 网络作为源特征提取器和目标特征提取器, 以获得更深层的表示, 达到更强的鲁棒性. 逻辑回归模型(LR)是最常使用的二分类器[43], 为了方便后续与其他传统 CPDP 方法进行对比, 本文选取 LR 作为源分类器和目标分类器.

如图 5 所示, AC-GAN 模型的构建分为以下几个步骤.

(1) 训练源特征提取器和源分类器: 在进行博弈之前, AC-GAN 模型首先对源特征提取器和源分类器进行有监督的训练, 将源项目数据以 8:2 的比例分成训练集和测试集, 使其在项目内缺陷预测取得良好的效果;

(2) 训练鉴别器和目标特征提取器: 按照 GAN 网络的架构对鉴别器和目标特征提取器进行对抗博弈训练. 由于目标特征最后需要与源特征相似, 目标特征提取器的参数一开始被设为与源特征提取器相同. 在鉴别器判断真假之后, 如果输出不为 0.5, 就按照梯度上升法更新参数, 然后再按照梯度下降法更新目标特征提取器参数, 调整输出的目标特征, 再次鉴别. 如此循环往复, 直到鉴别器输出为 0.5, 无法判断真假为止. 当鉴别器输出达到 0.5 时, 目标特征提取器参数就不再变化, 训练停止;

(3) 预测: 将训练好的目标特征提取器提取的特征输入到目标分类器中进行有无缺陷的预测, 由于源分类器已经训练好, 目标分类器的参数应与训练好的源分类器相同.

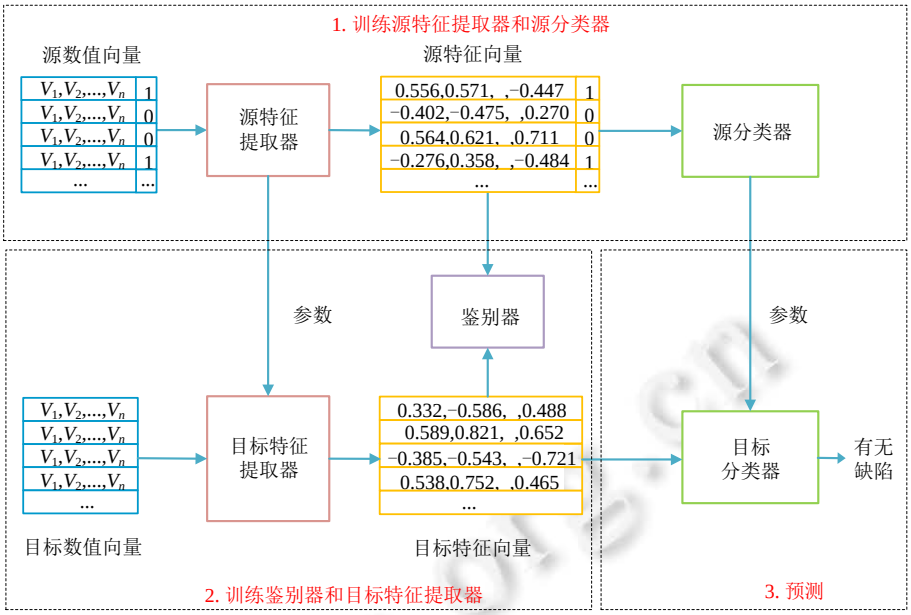


图 5 AC-GAN 模型架构

3 实验基础

实验的设计除了需要满足功能的完备性以外，还应充分测试本文工作的特异性和有效性. 因此，这一节将根据本文提出的方法特性设计对比实验，并描述实验各方面的设置.

3.1 研究问题

本文提出的 AC-GAN 方法采用 GAN 网络结构，通过对抗博弈达到消除源项目和目标项目数据分布差异的目的，且运用 BLSTM 网络作为特征提取器来进行特征提取. 为了验证 AC-GAN 方法的有效性，本文提出了以下两个研究问题.

- RQ1: 基于深度学习方法性能是否优于传统的 SDP 方法?
- RQ2: 相比于经典 CPDP 方法，本文提出的 AC-GAN 方法效果是否更好?

3.2 数据集

本文从开源数据库 PROMISE[44]中选取了 6 个 CPDP 领域常用的项目作为实验数据集. 如表 1 所示: 这 6 个项目的规模和缺陷率都不一样，因而保证了实验结果的广泛性. 其中，Log4j 项目的缺陷率太高，不能作为训练集使用.

因此，本文选择其他 5 个项目作为训练集，然后随机选择训练集本身以外的 3 个项目作为测试集.

表 1 本文选用项目的具体信息

项目名称	版本	文件数	缺陷率(%)
Camel	1.6	935	20.1
Poi	3.0	438	64.1
Xerces	1.4	508	76.8
Lucene	2.4	330	61.5
Xalan	2.6	875	53.1
Log4j	1.2	194	95.9

另外，数据集中的每个项目不仅含有源代码，还统计了源代码的静态代码度量和缺陷注释. 静态代码度量的具体指标见表 2. 这些指标可用于其他软件缺陷预测方法，与 AC-GAN 方法进行比较.

表 2 20 个静态代码度量的缩写和名字

缩写名	全名
LOC	Lines of code
DIT	Depth of inheritance tree
NOC	Number of children
RFC	Response for a class
CBO	Coupling between object classes
LCOM	Lack of cohesion in methods
LCOM3	Lack of cohesion in methods
NPM	Number of public methods
DAM	Data access metric
MOA	Measure of aggregation
MFA	Measure of function abstraction
IC	Inheritance coupling
CAM	Cohesion among methods of class
CBM	Coupling between methods
AMC	Average method complexity
Ca	Afferent couplings
Ce	Efferent couplings
Avg (CC)	Average McCabe
Max (CC)	Maximum McCabe
WMC	Weighted methods per class

3.3 实验设计

针对 RQ1, 本文选取了传统的 LR 方法与提出的采用深度学习特征提取器进行特征提取的 AC-GAN 方法进行项目内缺陷预测的对比实验. LR 方法是传统的直接采用 LR 分类器对数据集静态度量进行缺陷分类的方法.

针对 RQ2, 本文选取了几种传统的 CPDP 方法与基于 GAN 网络结构的 AC-GAN 方法进行跨项目缺陷预测的对比实验. 几种传统方法均采用 Matlab R2018b 软件编译, 简介如下.

- 迁移成分分析方法(TCA)[27]: 这是基于特征的迁移学习方法, 通过最小化源和目标的数据距离来达到源数据与目标数据分布相似的目的. 本该 CPDP 方法代码总行数为 973;
- 联合分布适应方法(JDA)[28]: 这是一种既考虑两个域之间的边缘分布(marginal distribution)又考虑条件分布(conditional distribution)的经典迁移学习方法. 本 CPDP 方法代码总行数为 320;
- 近邻过滤器方法(NNFilter)[29]: 该方法将类似的实例集合在一起, 以构建一个与目标数据集相同的训练集. 本 CPDP 方法代码总行数为 298;
- 平衡分布适应方法(BDA)[30]: 该方法自适应地为两个域间的边缘分布差异和条件分布差异分配不同的权重, 通过数据迁移使得源和目标数据分布趋于相似. 本 CPDP 方法代码总行数为 280;
- 代价敏感核化半监督字典学习方法(CKSDL)[33]: 这是一种可以充分利用内核空间中有限的标记缺陷数据和大量的未标记数据的半监督预测方法, 该方法还考虑了字典学习过程中的误分类代价. 本 CPDP 方法代码总行数为 1 592.

3.4 评价指标

根据实验关注的重点以及相关的实验设计, 本文选取了 *F*1-measure 和 AUC 两种性能指标对不同实验场景进行模型评估.

首先, 本文需要明确分类任务中常见的 4 种统计结果以及相关指标, 下面以二分类为例进行概念解释和说明.

- TP: 表示真实正类预测的数量, 即样本真实类别为正、预测类别也为正的实例个数;
- TN: 表示真实负类预测的数量, 即样本真实类别为负、预测类别也为负的实例个数;
- FP: 表示错误正类预测的数量, 即样本真实类别为负、预测类别为正的实例个数;
- FN: 表示错误负类预测的数量, 即样本真实类别为正、预测类别为负的实例个数.

以上 4 种统计结果构成了混淆矩阵. 根据混淆矩阵, 本文可以按照如下公式定义精确率(Precision, 即在

所有预测为正的样本中, 实际为正样本的概率)和召回率(Recall, 即在实际为正的样本中, 被预测为正样本的概率):

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

*F*1-measure 综合考虑了精确率和召回率, 是精确率和召回率的加权调和平均, 其定义如公式(5)所示:

$$F1\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

*F*1-measure 的值在 0 到 1 之间, 值越高, 表示模型分类效果越好.

而虚假阳性率(FPR)和真实阳性率(TPR)可按公式(6)和公式(7)计算:

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

ROC (receiver operating characteristic curve)曲线的横坐标是 FPR, 纵坐标是 TPR. AUC 就是 ROC 曲线下方的面积, 是判断二分类预测模型优劣的标准, 能够很好地描述模型整体性能的高低. TPR 越高, FPR 越低, 即当 AUC 的值越大时, 模型的预测性能更佳.

3.5 统计学分析方法

统计学分析方法可以帮助理解不同结果之间是否有统计学上的显著差异. Wilcoxon 符号秩检验(Wilcoxon signed-rank test)方法可以在不需要假设两个独立样本空间都为正态分布的情况下检测两者之间是否有显著性差异, 它可以应用于数据对. 在 RQ1 中, 本文采用了 Wilcoxon 符号秩检验方法来检验本文提出的使用深度学习提取的语义特征进行预测的方法是否优于传统的使用静态度量预测的方法. 在 95%的置信水平上, 该检验方法中的 p 值小于 0.05, 意味着两种方法之间的差异具有统计学意义; 而 p 值为 0.05 或更大, 则说明差异不具有统计学意义.

在 RQ2 中, 除了对每两种方法进行 Wilcoxon 符号秩检验外, 本文使用 Scott-Knott 测试(Scott-Knott test)方法从 *F*1-measure 和 AUC 评价指标的角度对 AC-GAN 方法与选取的传统 CPDP 方法进行比较和排名. Scott-Knott 测试以递归的方式执行方法分组过程. 它首先根据 *F*1-measure 或 AUC 将所有方法分为两类, 如果类别内部依然存在明显差异, 则递归地进一步划分等级. 当排名不能再继续细分时, Scott-Knott 测试就会终止.

3.6 实验设置

本文对 AC-GAN 方法的实验设置如下.

(1) 提取词向量: 抽象语法树中遍历出的节点序列被设置为 32 维, 词向量本身设置为 30 维, 即 CBOW 模型的输出为 32×30 维的数值向量;

(2) 训练源特征提取器及源分类器: 为了与数值向量匹配, 本文将源特征提取器的输入维度也设置为 32×30. 在本训练过程中, 训练数据按 8:2 进行分割, 分别进行训练和验证. 源特征提取器中的 BLSTM 网络可通过 Keras 库中的 Bidirectional 函数和 LSTM 函数实现, 训练的 epoch 数量设置 20. 该网络提取的特征向量需要先被全连接层映射为 30×1 维之后再输入到源分类器判断有无缺陷, 预测时以 *F*1-measure、AUC 作为评估指标. 训练好之后, 输出源特征;

(3) 训练鉴别器及目标特征提取器: 得到训练好的源特征提取器和源分类器后, 根据源特征提取器构造目标特征提取器, 保持目标特征提取器的初始参数与源特征提取器一致. 然后向目标特征提取器输入目标数值向量, 并输出 30×1 维的目标特征. 接着, 将目标特征和之前提取的源特征进行组合, 输入到由一维卷积、池化层和全连接层组成的鉴别器中, 输出鉴别结果. 根据鉴别结果, 不断地对目

标特征提取器和鉴别器进行训练, 直至无法对目标特征和源特征进行区分. 训练好之后, 根据源分类器构造目标分类器, 保持参数一致, 再对目标特征进行有无缺陷的分类.

本文使用 Python 环境和深度学习框架 TensorFlow 来实现本文提出的模型. 所有的实验都在装有 NVIDIA RTX 2080 的 Linux 服务器上运行. 本文提出的 AC-GAN 方法代码行数为 1 910 行.

4 实验结果和讨论

本节根据研究问题的顺序展示实验结果, 并进行实验讨论和分析.

4.1 针对RQ1的实验结果和讨论

一般的软件缺陷预测研究会采用手工提取的 20 个静态度量作为特征向量, 输入分类器中进行有无缺陷的预测. 然而静态度量往往只能反映代码文件的结构特征, 并不包含语义特征. 倘若代码存在细节上的 bug, 而没有在结构上体现出来, 那么传统的软件缺陷预测方法就无法识别此处的缺陷. 因此, 本文采用抽象语法树, 在保留语义信息的同时表示了代码结构, 利用 CBOW 算法来生成反映上下文信息的词向量, 并采用基于 BLSTM 神经网络的特征提取器来进行语义和上下文特征的提取. 表 3 为两种方法针对 6 个实验数据集在评估标准 *F*1-measure 上的对比实验结果.

表 3 AC-GAN 方法和 LR 方法的 *F*1-measure

项目名	AC-GAN 方法	LR 方法
Camel	**0.997**	0.325
Poi	**0.998**	0.726
Xerces	**0.998**	0.910
Lucene	**0.995**	0.581
Xalan	**0.999**	0.571
Log4j	**0.986**	0.960
Average	**0.996**	0.679
P-value	0.015 63	

如表 3 所示, AC-GAN 方法的 *F*1-measure 几乎接近于 1, 这说明本文的方法可以使软件缺陷预测的准确性大为提高. AC-GAN 方法的平均 *F*1-measure 值比 LR 方法的平均 *F*1-measure 值高 46.7%, 这说明使用抽象语法树、CBOW 算法和 BLSTM 特征提取器的 AC-GAN 方法是有效的, 且对预测性能有很大提高. 在 Wilcoxon 符号秩检验中, *p* 值仅为 0.015 6, 远小于 0.05, 表明两种方法之间存在着明显差异.

4.2 针对RQ2的实验结果和讨论

AC-GAN 方法利用 GAN 网络结构不断改变目标分布以匹配源分布, 直至数据分布差异问题被解决. 与传统的生成模型相比, GAN 网络的训练过程采用两种对抗性神经网络进行博弈, 可以采用反向传播的方法进行训练, 能够降低训练难度, 提高训练效率. 与传统的迁移学习相比, GAN 网络训练方式采用的是对抗训练方式, 可以产生更加清晰、真实的样本, 且 GAN 网络是利用鉴别器来实现数据迁移, 能够避免迁移学习中损失函数设计的困难. 表 4 和表 5 为 AC-GAN 方法与 5 种传统的 CPDP 方法的对比实验结果.

从表 4 和表 5 可以看出, 一共有 15 组源-目标项目对. 从整体上看, AC-GAN 方法的平均 *F*1-measure 和平均 AUC 都高于其他 5 种方法. 而在这 5 种传统方法中: 从 *F*1-measure 的结果看, BDA 方法的预测更加准确; 从 AUC 的结果看, CKSDL 方法的整体性能更强. 从单个实验的角度看: 在 *F*1-measure 的 15 组实验里, 有 86.7%的实验结果表示 AC-GAN 方法在 *F*1-measure 方面的表现优于其他 5 种方法; 同样, 在 AUC 的 15 组实验里, 有 80.0%的实验结果表示 AC-GAN 方法的 AUC 值都更好. AC-GAN 方法表现更优的原因如下.

(1) AC-GAN 方法在训练和预测过程中保留了代码程序的语义和上下文信息, 这些信息可发现静态度量无法识别的 bug, 使得预测更准确;

(2) AC-GAN 方法采用 GAN 网络结构做数据迁移, 迁移后的目标特征样本更加清晰、真实, 在与源特征分布相似的同时表征力更强, 提升了预测性能.

如表 4 和表 5 所示: 在 Wilcoxon 符号秩检验中, TCA、JDA、NNFilter、BDA 和 CKSDL 的 *F*1-measure 的 *p* 值分别为 6.10×10^{-4}、6.10×10^{-5}、1.22×10^{-4}、2.01×10^{-3} 和 1.16×10^{-3}, 均远远小于 0.05, 这意味着 AC-GAN 方法与其他方法之间存在着显著差异, 证明了 AC-GAN 方法的预测结果更加准确. 而 TCA、JDA、NNFilter、BDA 和 CKSDL 的 AUC 的 *p* 值也都小于 0.05, 说明 AC-GAN 方法整体性能优于其他传统方法.

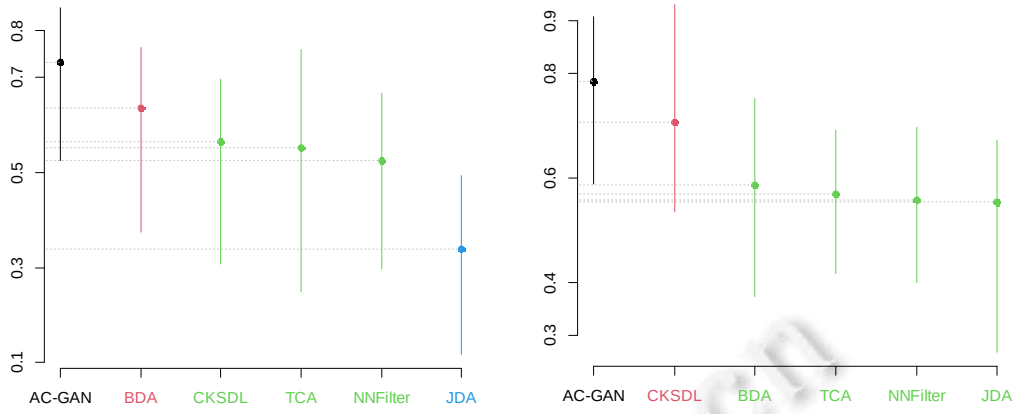
表 4 AC-GAN、TCA、JDA、NNFilter、BDA 和 CKSDL 的 *F*1-measure

源项目	目标项目	AC-GAN	TCA	JDA	NNFilter	BDA	CKSDL
Poi	Camel	**0.778**	0.500	0.457	0.455	0.374	0.307
Poi	Xerces	**0.813**	0.703	0.0623	0.660	0.757	0.594
Poi	Xalan	**0.801**	0.759	0.493	0.581	0.656	0.455
Lucene	Log4j	**0.782**	0.630	0.355	0.598	0.763	0.586
Lucene	Xerces	**0.766**	0.644	0.348	0.668	0.740	0.647
Lucene	Xalan	**0.847**	0.603	0.448	0.587	0.653	0.633
Xalan	Log4j	**0.690**	0.465	0.409	0.463	0.630	0.635
Xalan	Poi	**0.742**	0.523	0.474	0.501	0.660	0.676
Xalan	Lucene	0.525	0.528	0.315	0.530	0.604	**0.675**
Camel	Log4j	**0.801**	0.250	0.129	0.298	0.729	0.387
Camel	Xalan	0.552	0.384	0.118	0.380	**0.570**	0.548
Camel	Lucene	**0.810**	0.328	0.205	0.340	0.620	0.658
Xerces	Poi	**0.760**	0.618	0.349	0.598	0.676	0.696
Xerces	Xalan	**0.646**	0.697	0.426	0.566	0.496	0.356
Xerces	Lucene	**0.665**	0.639	0.427	0.647	0.623	0.627
Average		**0.732**	0.551	0.334	0.525	0.637	0.565
P-value		–	6.10×10^{-4}	6.10×10^{-5}	1.22×10^{-4}	2.01×10^{-3}	1.16×10^{-3}

表 5 AC-GAN、TCA、JDA、NNFilter、BDA 和 CKSDL 的 AUC

源项目	目标项目	AC-GAN	TCA	JDA	NNFilter	BDA	CKSDL
Poi	Camel	**0.886**	0.502	0.557	0.418	0.625	0.817
Poi	Xerces	**0.854**	0.621	0.45	0.55	0.673	0.593
Poi	Xalan	**0.769**	0.671	0.598	0.597	0.678	0.763
Lucene	Log4j	0.808	0.606	0.574	0.582	0.539	**0.877**
Lucene	Xerces	**0.908**	0.627	0.639	0.696	0.638	0.765
Lucene	Xalan	**0.844**	0.615	0.67	0.647	0.752	0.696
Xalan	Log4j	**0.801**	0.484	0.544	0.476	0.484	0.535
Xalan	Poi	**0.726**	0.572	0.666	0.568	0.581	0.543
Xalan	Lucene	**0.689**	0.589	0.663	0.615	0.575	0.618
Camel	Log4j	**0.853**	0.532	0.567	0.581	0.680	0.779
Camel	Xalan	0.733	0.690	0.532	0.683	0.515	**0.930**
Camel	Lucene	**0.855**	0.628	0.672	0.648	0.636	0.609
Xerces	Poi	0.588	0.417	0.464	0.426	0.603	**0.703**
Xerces	Xalan	**0.827**	0.509	0.268	0.400	0.374	0.754
Xerces	Lucene	**0.615**	0.471	0.437	0.494	0.456	0.613
Average		**0.784**	0.569	0.553	0.559	0.587	0.706
P-value		–	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}	3.53×10^{-2}

在 Scott-Knott 检验中, 6 种方法的 *F*1-measure 和 AUC 的排序如图 6 所示. 每一种颜色代表一个类别. 在图 6(a)中, 6 种方法的 *F*1-measure 被分为 4 类: 黑色代表 AC-GAN 方法, 排序最优; 红色代表 BDA 方法; 绿色代表 CKSDL、TCA 和 NNFilter 方法, 这 3 种方法被分为一类; 而蓝色代表 JDA 方法, 排序最末. 在图 6(b)中, 6 种方法的 AUC 被分为 3 类: 黑色代表 AC-GAN 方法, 排序依然最优; 红色代表 CKSDL 方法; 绿色代表剩余 4 种方法. 综合而言, AC-GAN 方法在两种排序中都是最高的类别, 具有最好的性能, 而 BDA 的 *F*1-measure 优于其他传统 CPDP 方法, CKSDL 方法的 AUC 仅次于 AC-GAN 方法.



(a) *F*1-measure 的排序 (b) AUC 的排序

图 6 AC-GAN、TCA、JDA、NNFilter、BDA 和 CKSDL 在 *F*1-measure 和 AUC 上的排序

5 风险分析

本节将从实验结果的可扩展性、实验评估的充分性和实验方法的健壮性这 3 个角度进行 AC-GAN 方法的风险分析.

5.1 实验结果的可扩展性

本文的实验数据集都来自开源的 PROMISE 资源库, 该资源库只包含 Java 数据集. 而在实际场景中, 编程语言多种多样, 因此本文的实验结果不能代表 AC-GAN 方法对其他项目的效果. 另外, 本文只采用了抽象语法树来表征程序代码, 这种表征方式只考虑了词法信息而并未考虑句法信息, 这些信息中可能会包含抽象语法树中未考虑到的特征, 所以本文方法存在一定的局限性.

5.2 实验评估的充分性

本文选择了 *F*1-measure 和 AUC 作为预测模型的评价指标. *F*1-measure 综合考虑了实验的准确率和召回率, 但其度量值取决于预定义的阈值(本文实验选取阈值为 0.3), 因此, 单独用 *F*1-measure 作为评价指标会使得实验评估不够充分. AUC 则考量了实验方法的整体性能, 其度量值基于 ROC 曲线. 同时考虑这两个评价指标可以较为客观地评估预测模型的性能. 然而, 即使这两个评价指标是 CPDP 研究中常用的, 也具有较好的代表性, 而实际上也还有许多其他指标可以用于二分类器的性能评估, 如描述实际分类与预测分类之间相关系数的 MCC (matthews correlation coefficient)和综合考虑召回率和特异性的评价系数 *G*-measure 等.

5.3 实验方法的健壮性

AC-GAN 方法针对的是一对一的源-目标项目对, 源项目的数据特征对目标项目预测结果有较大的影响, 而目标项目的数据特征也会影响源项目构建预测模型的性能表现. 例如在 RQ2 的实验结果中, 针对同一个目标项目 Xerces, 源项目 Poi 和 Lucene 预测的 *F*1-measure 分别是 0.813 和 0.766, 具有一定的差异; 而针对同一源项目 Poi, 目标项目 Camel、Xerces 和 Xalan 的 *F*1-measure 分别是 0.778、0.813 和 0.801, 差异较大.

6 结 论

CPDP 一直是软件工程数据挖掘领域的热门研究点, 对维护软件质量具有重要意义. 本文提出了一种新的基于深度学习的 CPDP 方法——AC-GAN 方法, 该方法利用 GAN 网络结构, 通过不断调整目标特征提取器和判别器参数, 达到使目标项目和源项目的数据分布相似的目的. 为了验证 AC-GAN 方法的有效性, 本文设计了两组对比实验, 使用 *F*1-measure 和 AUC 作为评价指标. 实验结果表明, 本文方法在数据处理阶段和模型

建立阶段都分别表现出良好的性能. AC-GAN 方法不仅可以提取语义、上下文信息, 还可以很好地消除不同项目之间的数据差异. 由于本文只使用了 LR 分类器, 并且只采用了抽象语法树一种表征方式, 所以在预测的准确性上有一定的限制. 在未来, 本文将使用集成方法或更合适的分类器, 采用数据流图、控制流图等表征方式, 以期获得更好的实验结果. 除此之外, 本文的方法仅限于一对一的源项目数据对的预测, 多对一的预测方式也是我们下一步探索的方向.

References:

[1] Gray J. Why do computers stop and what can be done about it? In: Proc. of the Symp. on Reliability in Distributed Software & Database Systems. 1986. [doi: 10.1039/9781847559319-FP007]

[2] Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. on Software Engineering, 2011, 38(6): 1276−1304. [doi: 10.1109/TSE.2011.103]

[3] Punitha K, Chitra S. Software defect prediction using software metrics—A survey. In: Proc. of the Int'l Conf. on Information Communication & Embedded Systems. IEEE, 2013. 555−558. [doi: 10.1109/ICICES.2013.6508369]

[4] Yang X, Lo D, Xia X, Yun Z, Sun J. Deep learning for just-in-time defect prediction. In: Proc. of the 2015 IEEE Int'l Conf. on Software Quality, Reliability and Security. IEEE, 2015. 17−26. [doi: 10.1109/QRS.2015.14]

[5] Wang S, Liu T, Tan L. Automatically learning semantic features for defect prediction. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). IEEE, 2016. 297−308. [doi: 10.1145/2884781.2884804]

[6] Qiao L, Li G, Yu D, Liu H. Deep feature learning to quantitative prediction of software defects. In: Proc. of the 45th IEEE Annual Computers, Software, and Applications Conf. (COMPSAC). 2021. 1401−1402. [doi: 10.1109/COMPSAC51774.2021.00204]

[7] Jones J. Abstract syntax tree implementation idioms. 2003. https://www.researchgate.net/publication/245153015_Abstract_Syntax_Tree_Implementation_Idioms

[8] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013.

[9] Gray D, Bowes D, Davey N, Yi S, Christianson B. Software defect prediction using static code metrics underestimates defect-proneness. In: Proc. of the 2010 Int'l Joint Conf. on Neural Networks (IJCNN). IEEE, 2010. 1−7. [doi: 10.1109/IJCNN.2010.5596650]

[10] Hosseini S, Turhan B, Gunarathna D. A systematic literature review and meta-analysis on cross project defect prediction. IEEE Trans. on Software Engineering, 2017, 45(2): 111−147. [doi: 10.1109/TSE.2017.2770124]

[11] Jin C. Cross-project software defect prediction based on domain adaptation learning and optimization. Expert Systems with Applications, 2021, 171(1). [doi: 10.1016/j.eswa.2021.114637]

[12] Chen S, Ye JM, Liu T. A cross project software defect prediction method based on domain adaptation. Ruan Jian Xue Bao/Journal of Software, 2020, 31(2): 266−281 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5632.htm [doi: 10.13328/j.cnki.jos.005632]

[13] Wang K, Gou C, Duan Y, Lin Y, Zheng X, Wang F. Generative adversarial networks: Introduction and outlook. IEEE/CAA Journal of Automatica Sinica, 2017, 4(4): 588−598. [doi: 10.1109/JAS.2017.7510583]

[14] Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V. Domain-adversarial training of neural networks. The Journal of Machine Learning Research, 2016, 17(1): 2096−2030. [doi: 10.1007/978-3-319-58347-1_10]

[15] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proc. of the IEEE Int'l Conf. on Computer Vision. IEEE, 2017. 2223−2232. [doi: 10.1109/ICCV.2017.244]

[16] Azadi S, Fisher M, Kim VG, Wang Z, Shechtman E, Darrell T. Multi-content gan for few-shot font style transfer. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. IEEE, 2018. 7564−7573. [doi: 10.1109/CVPR.2018.00789]

[17] Choi Y, Choi M, Kim M, Ha JW, Kim S, Choo J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. IEEE, 2018. 8789−8797. [doi: 10.1109/CVPR.2018.00916]

[18] Engel J, Agrawal KK, Chen S, Gulrajani I, Donahue C, Roberts A. Gansynth: Adversarial neural audio synthesis. arXiv preprint arXiv: 1902.08710, 2019.

[19] Chang CP. Integrating action-based defect prediction to provide recommendations for defect action correction. Int'l Journal of Software Engineering and Knowledge Engineering, 2013, 23(2): 147−172. [doi: 10.1142/S0218194013500022]

[20] Singh P, Pal NR, Verma S, Vyas OP. Fuzzy rule-based approach for software fault prediction. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 2016, 47(5): 826−837. [doi: 10.1109/TSMC.2016.2521840]

[21] 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]

[22] He Q, Shen B, Chen Y. Software defect prediction using semi-supervised learning with change burst information. In: Proc. of the 40th IEEE Annual Computer Software and Applications Conf. (COMPSAC), Vol.1. IEEE, 2016. 113−122. [doi: 10.1109/COMPSAC.2016.193]

[23] Yang X, Lo D, Xia X, Sun J. TLEL: A two-layer ensemble learning approach for just-in-time defect prediction. Information and Software Technology, 2017, 87: 206−220. [doi: 10.1016/j.infsof.2017.03.007]

[24] Wu F, Jing XY, Dong X, Cao J, Xu B. Cross-project and within-project semi-supervised software defect prediction problems study using a unified solution. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Software Engineering Companion (ICSE-C). IEEE, 2017. 195−197. [doi: 10.1109/ICSE-C.2017.72]

[25] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7): 1527−1554. [doi: 10.1162/NECO.2006.18.7.1527]

[26] Li J, He P, Zhu J, Lyu MR. Software defect prediction via convolutional neural network. In: Proc. of the 2017 IEEE Int'l Conf. on Software Quality, Reliability and Security (QRS). IEEE, 2017. 318−328. [doi: 10.1109/QRS.2017.42]

[27] Nam J, Pan SJ, Kim S. Transfer defect learning. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). IEEE, 2013. 382−391. [doi: 10.1109/ICSE.2013.6606584]

[28] Long M, Wang J, Ding G, Sun J, Yu PS. Transfer feature learning with joint distribution adaptation. In: Proc. of the IEEE Int'l Conf. on Computer Vision. 2013. 2200−2207. [doi: 10.1109/ICCV.2013.274]

[29] Turhan B, Menzies T, Bener AB, Stefano JD. 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]

[30] Xu Z, Pang S, Zhang T, Luo XP, Liu J, Tang YT, Yu X, Xue L. 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]

[31] Xia X, Lo D, Pan SJ, Nagappan N, Wang X. Hydra: Massively compositional model for cross-project defect prediction. IEEE Trans. on Software Engineering, 2016, 42(10): 977−998. [doi: 10.1109/TSE.2016.2543218]

[32] Ryu D, Jang JI, Baik J. A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Quality Journal, 2017, 25(1): 235−272. [doi: 10.1007/s11219-015-9287-1]

[33] Wu F, Jing, XY, Ying S, Jing S, Sun Y. Cross-project and within-project semi-supervised software defect prediction: A unified approach. IEEE Trans. on Reliability, 2018, 67(2): 581−597. [doi: 10.1109/TR.2018.2804922]

[34] Zhong S, Khoshgoftaar TM, Seliya N. Unsupervised learning for expert-based software quality estimation. In: Proc. of the HASE. 2004. 149−155. [doi: 10.1109/HASE.2004.1281739]

[35] Zhang F, Zheng Q, Zou Y, Hassan AE. Cross-project defect prediction using a connectivity-based unsupervised classifier. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). IEEE, 2016. 309−320. [doi: 10.1145/2884781.2884839]

[36] Javalang. 2020. https://github.com/c2nes/javalang

[37] Li Y, Huang CL, Wang ZF, Yuan L, Wang XH. Overview of software vulnerability mining methods based on machine learning. Ruan Jian Xue Bao/Journal of Software, 2020, 31(7): 2040−2061 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6055.htm [doi: 10.13328/j.cnki.jos.006055]

[38] Zhang X, Ben KR, Zeng J. Slice size defect prediction method based on code naturalness. Ruan Jian Xue Bao/Journal of Software, 2021, 32(7): 2219−2241 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6261.htm [doi: 10.13328/j.cnki.jos.006261]

[39] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. arXiv preprint arXiv: 1406.2661, 2014.

[40] Saon G, Tüske Z, Bolanos D, Kingsbury B. Advancing RNN transducer technology for speech recognition. In: Proc. of the 2021 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP 2021). IEEE, 2021. 5654−5658.

[41] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735−1780. [doi: 10.1162/neco.1997.9.8.1735]

[42] Ray A, Rajeswar S, Chaudhury S. Text recognition using deep BLSTM networks. In: Proc. of the 8th Int'l Conf. on Advances in Pattern Recognition (ICAPR). IEEE, 2015. 1−6. [doi: 10.1109/ICAPR.2015.7050699]

[43] Cramer JS. The Origins of Logistic Regression. Social Science Electronic Publishing, 2003. [doi: 10.2139/ssrn.360300]

[44] PROMISE repository. 2017. https://github.com/opensciences/opensciences.github.io

附中文参考文献:

[12] 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. http://www.jos.org.cn/1000-9825/5632.htm [doi: 10.13328/j.cnki.jos.005632]

[37] 李韵, 黄辰林, 王中锋, 袁露, 王晓川. 基于机器学习的软件漏洞挖掘方法综述. 软件学报, 2020, 31(7): 2040−2061. http://www.jos.org.cn/1000-9825/6055.htm [doi: 10.13328/j.cnki.jos.006055]

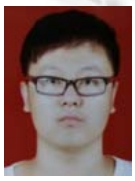
[38] 张献, 贲可荣, 曾杰. 基于代码自然性的切片粒度缺陷预测方法. 软件学报, 2021, 32(7): 2219−2241. http://www.jos.org.cn/1000-9825/6261.htm [doi: 10.13328/j.cnki.jos.006261]

邢颖(1978－), 女, 博士, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为软件测试, 人工智能的应用.

章世豪(1999－), 男, 硕士生, CCF 学生会员, 主要研究领域为深度学习及其应用.

钱晓萌(1997－), 女, 硕士生, CCF 学生会员, 主要研究领域为软件缺陷预测, 深度学习.

赵梦赐(1999－), 男, 硕士生, 主要研究领域为深度学习及其应用.



管宇(1998－), 男, 硕士生, 主要研究领域为深度学习, 软件测试.

林婉婷(1997－), 女, 硕士生, CCF 学生会员, 主要研究领域为机器学习, 软件缺陷预测.