软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(2):266−281 [doi: 10.13328/j.cnki.jos.005632]

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

一种基于领域适配的跨项目软件缺陷预测方法*

陈 曙, 叶俊民, 刘 童

(华中师范大学 计算机学院,湖北 武汉 430079)

通讯作者: 陈曙, E-mail: chenshu@mail.ccnu.edu.cn

摘 要: 软件缺陷预测旨在帮助软件开发人员在早期发现和定位软件部件可能存在的潜在缺陷,以达到优化测试资源分配和提高软件产品质量的目的.跨项目缺陷预测在已有项目的缺陷数据集上训练模型,去预测新的项目中的缺陷,但其效果往往不理想,其主要原因在于,采样自不同项目的样本数据集,其概率分布特性存在较大差异,由此对预测精度造成较大影响.针对此问题,提出一种监督型领域适配(domain adaptation)的跨项目软件缺陷预测方法.将实例加权的领域适配与机器学习的预测模型训练过程相结合,通过构造目标项目样本相关的权重,将其施加于充足的源项目样本中,以实例权重去影响预测模型的参数学习过程,将来自目标项目中缺陷数据集的分布特性适配到训练数据集中,从而实现缺陷数据样本的复用和跨项目软件缺陷预测.在 10 个大型开源软件项目上对该方法进行实证,从数据集、数据预处理、实验结果多个角度针对不同的实验设定策略进行分析;从数据、预测模型以及模型适配层面分析预测模型的过拟合问题.实验结果表明,该方法性能优于同类方法,显著优于基准性能,且能够接近和达到项目内缺陷预测的性能.

关键词: 软件缺陷预测;软件缺陷度量元;机器学习;迁移学习;领域适配

中图法分类号: TP311

中文引用格式: 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

英文引用格式: Chen S, Ye JM, Liu T. Domain adaptation approach for cross-project software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2020,31(2):266−281 (in Chinese). http://www.jos.org.cn/1000-9825/5632.htm

Domain Adaptation Approach for Cross-project Software Defect Prediction

CHEN Shu, YE Jun-Min, LIU Tong

(School of Computer, Central China Normal University, Wuhan 430079, China)

Abstract: Software defect prediction aims at the very early step of software quality control, helps software engineers focus their attention on defect-prone parts during verification process. Cross-project defect predictions are proposed in which prediction models are trained by using sufficient training data from already existed software projects and predict defect in some other projects, however, their performances are always poor. The main reason is that, the divergence of the data distribution among different software projects causes a dramatic impact on the prediction accuracy. This study proposed an approach of cross-project defect prediction by applying a supervised domain adaptation based on instance weighting. The sufficient instances drawn from some source project are weighted by assigning target-dependent weights to the loss function of the prediction model when minimizing the expected loss over the distribution of source data, so that the distribution properties of the data from target project can be matched to the source project. Experiments including dataset selection, data preprocessing and results are described over different experiment strategies on ten open-source software projects. Over fitting problems are also studied through different levels including dataset, prediction model and domain adaptation process. The results show that the proposed approach is close to the performance of within-project defect prediction, better than similar approach and significantly better that of the baseline.

Key words: software defect prediction; software defect metrics; machine learning; transfer learning; domain adaptation

* 基金项目: 国家科技支撑计划(2015BAK33B00)
Foundation item: National Key Technology Research and Development Program of China (2015BAK33B00)
收稿时间: 2017-12-17; 修改时间: 2018-01-29, 2018-03-26, 2018-04-12; 采用时间: 2018-08-09

基于数据挖掘和机器学习技术的软件缺陷预测(software defect prediction)是目前较为关注的研究热点,其主要思想是,通过分析软件代码或挖掘软件开发日志,设计出与软件缺陷相关的度量元,由此创建缺陷预测数据集,并通过机器学习算法在该数据集上训练预测模型,如分类器(classifier)或回归模型(regressor)等,然后预测不同粒度(如对象、模块、变更等)软件部件中是否可能存在缺陷[1−6].然而大多数相关研究均基于项目内缺陷预测(within-project defect prediction,简称 WPDP)[6−9],即构造预测模型的训练数据和进行缺陷预测的数据均采样自同一项目中.然而在实践中,给定一个新的项目,往往需要大量的人力和时间去采样和标记训练样本,这与软件缺陷预测“及早发现可能缺陷从而进行质量控制”的初衷是相悖的.

针对上述问题,提出了跨项目软件缺陷预测(cross-project defect prediction,简称 CPDP)[10−13],旨在复用现有项目中高质量数据集,构造预测模型对新项目进行缺陷预测.但此类方法往往效果较差,其主要原因在于,基于监督学习的机器学习模型,其要求训练数据和测试数据均来自于同分布的样本空间.而在 CPDP 设定下,由于不同软件项目的开发方式、开发人员、开发环境等均不相同,因此采样的数据其概率分布也不尽相同,导致难以直接将现有数据集(以下简称源项目缺陷数据集)中训练的缺陷预测模型泛化到新的项目数据(以下简称目标项目缺陷数据集)中.尽管如此,不同软件项目仍存在可能的共性,如采用相同的开发语言、相近的编码风格、相似的体系结构等,这些共性能够作为模式和规律传递的桥梁,建立不同概率分布样本集之间的联系.因此,若要提高 CPDP 模型预测性能,则需要将来自于不同项目的数据集进行概率特性适配,从而使不同的样本空间具有相似的分布特性.一个典型的基于领域适配方法的 CPDP 预测流程如图 1 所示.

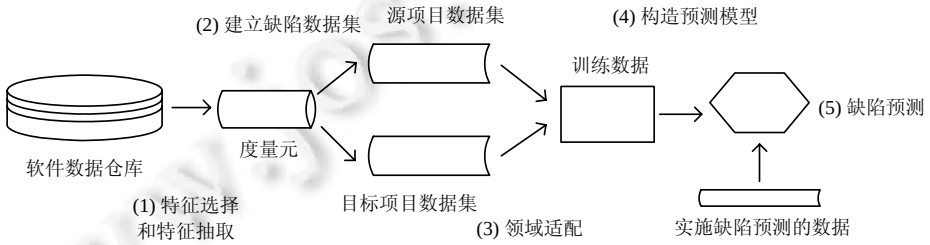

Fig.1 A typical process of CPDP with domain adaptation

图 1 一个典型的基于领域适配的 CPDP 预测流程

在机器学习领域,为了解决不同领域数据集所带来的分布差异而导致的跨领域预测精度低下的问题,研究者提出了“领域适配”的思想[14].领域适配是机器学习中迁移学习(transfer learning)[15]的一种实现方法,其主要思想是,训练学习源数据集(source domain)和目标数据集(target domain)之间的共性,或在两者之间进行概率分布适配,并在其基础上训练机器学习模型,使得该模型在目标数据集上具有较好的泛化能力.基于该思想,本文提出一种监督型领域适配的跨项目软件缺陷预测方法,利用目标数据集的概率分布特征对源数据集进行实例加权处理,使用实例权重去影响预测模型的参数学习过程,从而将目标数据集的分布特征适配到源数据集上,以实现软件缺陷数据的复用.此外,通过在预测模型中加入稀疏和正则约束,在缺陷样本的权重计算过程中加入度量元的先验知识,分别在预测模型和领域适配层面减轻模型的过拟合问题.

本文第 1 节对相关研究进行介绍.第 2 节、第 3 节详细介绍本文提出的方法.第 4 节给出实验及其相关分析讨论.第 5 节做出总结和展望.

1 相关研究

软件缺陷预测是软件工程领域中较受关注的研究课题,研究者提出了不同的缺陷分析度量元以及缺陷预测方法,而其中针对 CPDP 方法研究较少.Zimmermann 等人[11]针对 12 个开源软件项目组合研究了 CPDP 方法,得出结论为:直接将在源项目缺陷数据中训练的预测模型应用到目标项目中,其精度无法达到实践要求.Turhan 等人[12]研究了跨公司的软件缺陷预测(cross-company defect prediction)方法,通过使用 K 近邻聚类方式从某些软件公司采样的缺陷样本中选择缺陷实例,训练预测模型并应用到其他公司所开发的软件项目数据中,得出结

论为:由于不同公司开发软件所采用的技术、方法、环境的差异,使得跨越公司的软件缺陷预测较为困难.Nam等人[16]提出一种 CPDP 方法,主要思想是,使用迁移学习技术,通过最小化不同项目数据集概率分布之间的最大平均差异(maximum mean discrepancy)来抽取一个隐藏的特征空间,该特征空间代表了不同项目数据之间的共性,在该特征空间上训练预测模型,从而实现 CPDP 方法.Ma 等人[17]使用一种基于朴素贝叶斯模型的跨公司软件缺陷的迁移预测模型(transfer naive Bayes,简称 TNB).该方法通过源和目标项目度量元的极值、均值、中位数以及标准差等作为特征向量,在欧式空间中计算两者的相似性,结合引力公式构造相似度权重以训练贝叶斯预测模型,当源数据集和目标数据集特征差异较小时能够取得较好的效果.但由于该方法仅通过数据集的部分性质来衡量相似度,而未充分考虑样本所有度量元取值的分布,因此不能真实反映目标项目数据的所有缺陷分布特征.何吉元等人[18]提出一种基于搜索的半监督集成 CPDP 方法,该方法通过调整训练集中各类数据的分布比例以构建出多个朴素贝叶斯基分类器,通过具有全局搜索能力的遗传算法,基于少量已标记目标实例对基分类器进行集成,并构建出最终的缺陷预测模型.Rahman 等人[19]针对 12 个不同开源项目,从缺陷预测性能、预测稳定性、数据集特征等各方面进行分析,得出结论为:不同软件项目之间具备潜在的共性,利用这些共性以构造具有较高性能的CPDP模型是可行的.上述研究或仅从概念和可行性层面探讨CPDP问题,或仅考察了目标项目数据集的部分分布特征,或并未考虑到模型的复杂度以及训练数据样本的个体特征带来的过拟合问题,因此在特定条件下仍然存在效果不理想的情况.

领域适配技术是迁移学习技术的一种实现方式,其在机器学习、模式识别、数据挖掘等领域越来越受到关注,已经成功用于视觉分析、语音识别、自然语言处理等应用中[14,20].当目标数据不充足,或目标数据中已标记数据不充足时,领域适配技术可充分复用现有的充足的已标记源数据集,提高机器学习模型在目标数据集中的预测或分类泛化性能,从而提供一种灵活的有监督或无监督训练框架,实现不同领域数据之间的知识迁移和共享.从待预测的目标数据集角度考察,领域适配技术可适用于两种情况:(1) 目标数据样本中仍有少量的已标记数据样本[20,21];(2) 目标数据集中仅有未标记数据样本[22−24].本文主要考察第 1 种情况,并假定目标缺陷数据集中仅含有少量的可作为训练数据的已标记数据样本.使用目标缺陷数据集中少量的已标记训练样本的概率特征对源缺陷数据集实施实例加权,从而将目标数据集的分布特征适配到源数据集中.

文献[22,25]描述了基于实例加权的领域适配方法,该方法已成功实践于不同的应用中[22,25,26].实例加权的主要思想是,在优化基于机器学习的预测或分类模型的代价损失函数过程中,利用目标数据样本的分布特性,对其依赖的训练数据实例附加不同权重,从而影响模型参数的学习,使得预测模型在目标数据样本中具有较好的泛化性能.考察源数据集和目标数据集的分布特性,实例加权主要有两种实现方法以解决两种问题.

1) 标记类别不平衡(class imbalance)[27]:在该问题中,给定某一标记类别,其对应样本的条件分布在不同域(源数据集和目标数据集)中分布是相同的,但标记类本身的分布不同,即 $P^s(Y)\neq P^t(Y)$.
2) 协变量转移(covariate shift)[28]:在该问题中,给定同一样本,其标记类别的分布在不同域中相同,但其样本的边缘分布不同,即 $P^s(X)\neq P^t(X)$.

区别于上述两种情况,在本文的 CPDP 设定下,我们假设缺陷数据实例样本和其对应的标记类别的联合分布不同,即 $P^s(X,Y)\neq P^t(X,Y)$.

本文提出一种 CPDP 方法.它既充分考虑了缺陷样本所有度量元取值,从整体分布上权衡不同项目数据集之间的差异,又通过在逻辑回归预测模型中加入稀疏和正则约束,在缺陷样本的权重计算过程中加入度量元的先验知识,分别在预测模型和领域适配层面减轻因数据集的个体特征导致的模型过拟合问题.此外,与贝叶斯等生成式模型相比,逻辑回归的判别式特性更为宽松,能够加入正则惩罚项降低模型复杂度,无需严格的度量元独立性假设,且不需要计算样本的联合分布,当实践中缺陷样本容量较大时,其计算复杂度更低.

2 研究问题定义

本节针对所研究问题,给出如下定义.

定义 1(缺陷度量元). 缺陷度量元是指一种可量化的与软件缺陷具有可能相关性的软件部件的特征和性

质的基本属性的描述.度量元的量化值,可认为是该度量元的一个实例.

缺陷度量元可作为软件缺陷的特征(feature),用于构造缺陷数据样本.由于“特征”一词常用于机器学习领域,用于描述数据或样本的抽象性质,因此在本文中,“度量元”和“特征”可认为等价.

定义 2(领域). 在本文中,领域定义为一组特定的特征集合,其实例样本采样自同一分布中.

在 CPDP 设定中,采样自不同的软件项目(如 Eclipse,Microsoft explorer,CAD 等)中的缺陷数据样本认为是来自不同的领域.需要注意的是,不同领域中的特征集可相同也可不同.本文仅考察特征相同的情形.

定义 3(已标记样本和未标记样本). 给定样本 x_i,其对应的极性类别标记为 y_i.当 x_i 采样自可能存在缺陷的软件部件中,则认为样本 x_i 可能引发缺陷(buggy:y_i=+1);否则,认为 x_i 不会引发缺陷(clean:y_i=0).当样本对应的极性标记有确定取值,认为其是已标记样本{(x_i,y_i)};反之,则认为该样本是未标记样本.

定义 4(跨项目软件缺陷预测). 给定两个不同领域$\mathcal{D}^s$和$\mathcal{D}^t$,其中,$\mathcal{D}^s$代表现有的已标记样本充足的源项目缺陷数据集,$\mathcal{D}^t$代表需要预测的目标项目缺陷数据集.$\mathcal{D}^s$和$\mathcal{D}^t$具有相同的度量元,且样本和类别标记的联合概率分布不同:$P^s(X^s,Y^s)\neq P^t(X^t,Y^t)$.

令 $\mathcal{D}^s=\{(x_i^s,y_i^s)\}_{i=1}^{ns}$,其中,$ns$ 为$\mathcal{D}^s$ 中已标记样本的数量.令$\mathcal{D}^t=\mathcal{D}^{lt}\cup\mathcal{D}^{ut}$,其中,$\mathcal{D}^{lt}=\{(x_i^t,y_i^t)\}_{i=1}^{nt}$ 为$\mathcal{D}^t$ 中 nt 个已标记样本;$\mathcal{D}^{ut}=\{(x_j^t)\}_{j=1}^{nu}$ 表示$\mathcal{D}^t$ 中 nu 个未标记样本,且满足 $nu>nt$.x_i 为单一样本,定义为 $x_i=\{x_i^1,x_i^2,...,x_i^n\}$,表示 n 个不同的度量元的度量值.CPDP 的目标是,以充足的已标记源数据集$\mathcal{D}^s$ 和目标数据集中少数的已标记样本$\mathcal{D}^{lt}$作为训练集,即$\mathcal{D}^s\cup\mathcal{D}^{lt}$,训练预测模型,将该模型应用于$\mathcal{D}^t$ 中多数未标记的数据$\mathcal{D}^{ut}$ 上进行缺陷预测(测试)活动.其直观设定如图 2 所示.

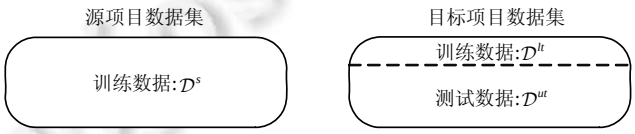

Fig.2 Basic CPDP setting in which source data and a small part of target data are available for training

图 2 CPDP 的基本设定,其中,源项目数据集和少部分目标项目数据集作为预测模型训练数据

3 基于领域适配的缺陷预测模型

3.1 预测模型定义

从机器学习角度看,CPDP 建模可看成是构造具有领域适配过程的二元分类器.本文使用软件缺陷预测中较为常见的逻辑回归(logistic regression)模型作为基础分类器[11,16],其模型定义如下:

$$f(x)=g(\theta^Tx+b)=1/(1+\exp(-\theta^Tx-b)) \tag{1}$$

其中,$g(z)$为 sigmoid 函数,定义为 $g(z)=1/(1+\exp(-z))$,θ和 b 分别为模型参数向量和偏移值.给定分类器参数集$\theta^*=\{\theta,b\}$和输入缺陷实例样本 x,则该样本包含缺陷的后验概率定义为

$$P(y=+1|x,\theta^*)=\exp(\theta^Tx+b)/(1+\exp(\theta^Tx+b)) \tag{2}$$

参数集θ^*通过在训练数据样本集上优化最小二乘损失函数(least square loss function)的方式来估计.为了降低模型复杂度从而避免多重共线性和过拟合现象[29],在其上加入 L_1(Lasso)和 L_2(Ridge)范式约束[29,30],以构造分类器的稀疏和正则约束惩罚项.其中,L_1(Lasso)范式约束通过惩罚分类器参数的绝对值大小来增强模型参数的稀疏性,以避免过拟合;L_2(Ridge)范式约束通过对模型参数向量的长度平方增加惩罚项来实现参数收缩,减轻参数的多重相关性.综合 L_1 和 L_2 范式约束构成 ElasticNet 惩罚约束(ElasticNet 逻辑回归)[30],因此从参数空间Θ中获取模型参数θ^*的目标函数定义为

$$\theta^*=\arg_{\theta\in\Theta}\min\sum\nolimits_{x\in X,y\in Y}^{n}P(x,y)L(x,y,\theta) \tag{3}$$

其中,L 为结合了 L_1 和 L_2 范式约束的最小二乘损失函数,定义为

$$L(x,y,\theta)=\|\theta^Tx+b-y\|_2^2/2n+\alpha\rho\|\theta\|_1+\alpha(1-\rho)\|\theta\|_2^2/2 \tag{4}$$

其中,α和ρ为超参数,从[0,1]取值.参数α控制模型参数的稀疏程度;,ρ为 ElasticNet 惩罚项混合参数,用于控制 L_1 和 L_2 范式约束之于损失函数的惩罚权重.

由于样本空间整体的分布函数 $P(x,y)$未知且较难获取,因此通过在训练集 $\{(x_i,y_i)\}_{i=1}^n$ 上计算经验分布来做参数的近似估计:

$$\theta^* \approx \arg_{\theta\in\Theta}\min\sum_{i=1}^{n} L(x_i,y_i,\theta) \tag{5}$$

其中,经验损失函数定义为

$$L(x_i,y_i,\theta)=\|\theta^T x_i+b-y_i\|_2^2/2n+\alpha\rho\|\theta\|_1+\alpha(1-\rho)\|\theta\|_2^2/2 \tag{6}$$

3.2 领域适配方法

由于目标函数(5)仅考虑了来自单一项目的缺陷数据样本,而在 CPDP 设定中需要同时在具有不同分布特征的目标缺陷数据集上进行目标函数的优化和参数估计,因此需通过对源项目缺陷数据样本构造实例加权.令 P^s 和 P^t 表示源项目缺陷数据集和目标项目数据集的概率分布函数,其领域适配过程推理如下:

$$\left.\begin{aligned}\theta^* &= \arg_{\theta\in\Theta}\min\sum_{x\in X,y\in Y}^{n} P^t(x,y)L(x,y,\theta)\\ &= \arg_{\theta\in\Theta}\min\sum_{x\in X,y\in Y}^{n} P^t(x,y)P^s(x,y)L(x,y,\theta)/P^s(x,y)\\ &\approx \arg_{\theta\in\Theta}\min\sum_{i=1}^{ns} p^t(x,y)L(x_i^s,y_i^s,\theta)/p^s(x,y)\end{aligned}\right\} \tag{7}$$

其中,$p^s(x,y)$和 $p^t(x,y)$分别表示源项目缺陷数据集和目标项目缺陷数据集样本的概率密度函数.由于我们仍有少量的已标记目标缺陷数据样本,因此加入其经验损失项后,修改目标函数(7)为

$$\left.\begin{aligned}\theta^* &\approx \arg_{\theta\in\Theta}\min\sum_{i=1}^{ns} p^t(x_i^s,y_i^s)L(x_i^s,y_i^s,\theta)/p^s(x_i^s,y_i^s)+\sum_{j=1}^{nt} L(x_j^t,y_j^t,\theta)\\ &= \arg_{\theta\in\Theta}\min\sum_{i=1}^{ns} w_i L(x_i^s,y_i^s,\theta)+\sum_{j=1}^{nt} L(x_j^t,y_j^t,\theta), w_i=p^t(x_i^s,y_i^s)/p^s(x_i^s,y_i^s)\end{aligned}\right\} \tag{8}$$

公式(8)中:w_i 为源项目缺陷数据样本的附加权重,该权重通过源数据样本在目标数据分布下的度量,结合其自体分布的正则化来衡量源数据样本和目标数据样本之间分布的相似性.权重 w_i 亦可看做是源数据样本 (x_i^s, y_i^s) 在目标数据分布下对目标函数优化进行模型参数估计的影响因子.图 3 给出了一个更为直观的描述,图 3(a)中的叉代表源项目数据样本,圈代表目标项目数据样本;图 3(b)中的概率密度曲线表示源数据样本和目标数据样本之间的分布差异;图 3(c)中的实例权重曲线表示当源数据样本落在高密度目标数据样本区域时,其附加的实例权重较大,其对于预测模型参数学习的贡献也越高.特别的,对于少量已标记的目标缺陷数据样本,其对应权重设定为常数 1.将上述领域适配方法融入 ElasticNet 逻辑回归中得到最终的目标函数定义如下:

$$\theta^* \approx \arg_{\theta\in\Theta}\min\sum_{i=1}^{ns} w_i\|\theta^T x_i^s+b-y_i^s\|_2^2/2ns+\sum_{j=1}^{nt} L\|(\theta^T x_j^t+b-y_j^t\|_2^2/2n+\alpha\rho\|\theta\|_1+\alpha(1-\rho)\|\theta\|_2^2/2 \tag{9}$$

目标函数(9)可通过传统的梯度下降等优化算法快速而有效地进行求解.

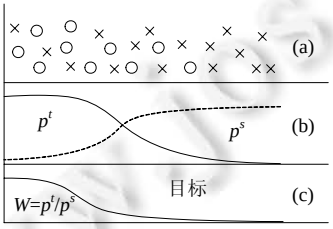

Fig.3 An illustration of instance weighting from target (circle) to source (cross)

图 3 目标(圈)到源(叉)的实例加权表示

3.3 适配权重计算

在该方法中,计算源数据样本实例权重 w_i 是关键.然而,需要估计的样本为多维随机变量,其具有较高的维

度,因此,估计精确的密度函数 $p(x,y)$需要指数级的样本数量,从实践角度看不可行.参考类似方法如朴素贝叶斯等,对多维随机变量加入度量元条件独立性假设: $p(x_i,y_i)=p(y_i)p(x_i^1)p(x_i^2)...p(x_i^K)$.下面主要考察两类度量元,即离散度量元和连续度量元.

1) 离散型度量元

大多数缺陷度量元为离散类型,如代码行数、类的方法数、类的属性个数等.对于缺陷标记和离散型度量元,使用概率质量函数(PMF)来计算度量元在每个离散点的概率:

$$g(z,\boldsymbol{p})=\begin{cases}p_z(\mathcal{D}_f=z), & z\in S\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

其中,集合 S 表示度量元 f 所能够取到的离散值空间;$\boldsymbol{p}=\{p_1,p_2,\ldots,p_K\}$为 PMF 的参数向量,表示其在所有可能离散点的概率.传统方法主要通过直方图或多项式分布等定义 PMF[25−28],但这些方法主要从训练样本集中估计参数,而如果数据样本本身分布不平衡,则会造成过拟合而降低模型的泛化能力.参考贝叶斯理论,使用多项式分布的先验共轭狄利克雷(Dirichlet)分布来取代传统的多项式分布.假设有 m 个训练样本,其中,第 i 个度量元能够取到 K 个不同的离散值,其对应的狄利克雷 PMF 定义如下:

$$Dir(\boldsymbol{p}\mid\boldsymbol{\alpha})=\Gamma\left(\sum_{i=1}^{K}\alpha_i\right)\prod_{i=1}^{K}p_i^{\alpha_i-1}\Big/\prod_{i=1}^{K}\Gamma(\alpha_i) \tag{11}$$

其中,Γ为 gamma 函数.超参数$\boldsymbol{\alpha}$是一个 K 维向量,表示软件开发/测试人员对于度量元 f 的先验知识.例如,对于样本缺陷标记y_i,可使用对称狄利克雷来构造其 PMF,其先验知识假定了所有缺陷样本包含潜在缺陷的概率是 0.5.此外,满足满足 $\sum p_i=1$.Mult 表示多项式分布函数,定义如下:

$$Mult(\boldsymbol{n};p)=\frac{m!}{n_1!n_2!...n_K!}p_1^{n_1}p_2^{n_2}...p_k^{n_K}=\Gamma\left(\sum_i n_i+1\right)\prod_{i=1}^{K}p_i^{n_i}\Big/\prod_i\Gamma(n_i+1) \tag{12}$$

其中,向量 $\boldsymbol{n}=\{p_1,p_2,\ldots,n_k\}$表示度量元 f 在训练样本中取到 K 个不同离散值的频次,且满足 $m=\sum_{i=1}^{K}n_i$.因此,加入先验知识后,度量元 f 最终的后验 PMF 定义如下:

$$p(\boldsymbol{p}\mid\mathcal{D}_f,\boldsymbol{\alpha})=\Gamma\left(\sum_{i=1}^{K}\alpha_i+n_i\right)\prod_{i=1}^{K}p_i^{\alpha_i+n_i-1}\Big/\prod_{i=1}^{K}\Gamma(\alpha_i+n_i) \tag{13}$$

其中,$Dir(\boldsymbol{p}|\boldsymbol{\alpha})$为 $p(\boldsymbol{p}|\mathcal{D}_f,\boldsymbol{\alpha})$的共轭先验.因此,当有新的样本数据加入时,可直接将当前的后验分布作为先验知识乘以新样本的似然函数,得到新的后验分布,而无需修改算法.参数向量 $\boldsymbol{p}$ 可通过最大后验概率估计(maximum a posteriori estimation,简称 MAP)算法[31]从训练样本中估计得到.

2) 连续型度量元

连续型度量元的分布信息往往是未知的,因此可通过非参数估计来进行度量.本文使用核密度估计算法来获取一个平滑的概率密度函数(PDF).给定连续型度量元 f,其 PDF 估计如下:

$$p(f)\approx p_{kde}(f)=\left(\sum_{i-1}^{n}K((f-f_i)/h)\right)\Big/nh \tag{14}$$

其中,K 为核函数(本文采用高斯核函数),h 为平滑参数.h 值越大,获得的 PDF 曲线越平滑;相反,获得的 PDF 则越尖锐.平滑参数 h 的取值可通过综合均方误差(integrated mean squared error)[30]方法,结合斯科特法则(Scott rule)[32]或西尔弗曼拇指法则(Silverman rule of thumb)[33]自动优化获取.

4 实验及分析

本节主要描述使用本方法进行 CPDP 实验中使用的数据集、数据预处理、模型实现、实验相关的设置、实验结果、相关研究问题的探讨以及方法的局限性分析.

4.1 数据集

本文主要使用了 D'Ambros 等研究人员[34]和 Kamei 等研究人员[35]发布并公开的缺陷数据集 AEEEM 和 Kamei.AEEEM 数据集包含了从 5 个大型开源软件项目中收集到的 61 种不同的软件缺陷度量元.缺陷度量元包括有代码级度量元(source code)、缺陷历史度量元(previous-defect)、代码变更熵度量元(changes entropy)、代

码熵度量元(source code entropy)以及代码流失度量元(churn of source code)等几个类别.Kamei 数据集则包含了从 6 个开源软件项目中获取的 14 种不同的缺陷度量元,缺陷度量元包括有代码流失度量元和代码熵度量元.本实验采用的数据集和度量元的基本信息见表 1 和表 2.

Table 1 Summary of defect datasets in our experiment

表 1 实验所使用的缺陷样本数据集总体信息

数据集名称	软件项目名称	时间段	总样本数	缺陷比率(%)
AEEEM	Eclipse (Eclipse JDT)	1/1/2005-17/6/2008	997	20.66
	Mylyn	1/1/2005-3/17/2009	1 862	13.16
	Lucene (Apache Lucene)	1/1/2005-10/8/2008	691	9.29
	Equinox	1/1/2005-25/6/2008	324	39.69
	PDE (Eclipse PDE UI)	1/1/2005-9/11/2008	1 492	14.01
Kamei	Bugzilla	1/8/1998-1/12/2006	4 620	37.20
	Columba	1/11/2002-1/7/2006	4 455	31.30
	Eclipse JDT	1/5/2001-1/12/2007	35 386	14.12
	Platform	1/5/2001-1/12/2007	64 250	15.31
	Mozilla	1/1/2000-1/12/2006	98 275	5.20
	PostgreSQL	1/7/1996-1/5/2010	20 431	25.11

Table 2 Metrics we selected in our experiment

表 2 实验所选取的缺陷度量元

数据集名称	类别	度量元标记	描述
AEEEM	Source code	ck_oo_cbo	对象之间的耦合度
		ck_oo_numberOfLinesOfCode	代码行数
	Churn of source code	churn_cbo	对象之间的耦合度(Churn)
		churn_dit	类继承树的深度(Churn)
		churn_fanIn	引用当前类的其他类数量(Churn)
		churn_fanOut	被当前类引用的其他类数量(Churn)
		churn_lcom	方法欠内聚程度(Churn)
		churn_noc	当前类的子类数(Churn)
		churn_numberOfAttributes	当前类的属性数(Churn)
		churn_numberOfAttributesInherited	当前类所继承的属性数(Churn)
		churn_numberOfMethods	当前类的方法数(Churn)
		churn_numberOfMethodsInherited	当前类所继承的方法数(Churn)
		churn_numberOfLinesOfCode	代码行数(Churn)
	Entropy of source code	ent_cbo	对象之间的耦合熵
		ent_ numberOfLinesOfCode	代码行数熵
	Entropy of changes	CvsEntropy	代码变更熵
		CvsLogEntropy	CvsEntropy 的对数衰减
	Defect history	numberOfBugsFoundUntil	历史缺陷发现数
		numberOfCriticalBugsFoundUntil	历史致命缺陷发现数
Kamei	Churn of source code	NS	子系统数目(Churn)
		ND	子文件夹数(Churn)
		NF	文件数(Churn)
		LA	新增的代码行数(Churn)
		LD	删除的代码行数(Churn)
		LT	修改前的原始代码行数(Churn)
		NDEV	参与修改文件的开发人员数(Churn)
		AGE	当前更改和最终更改的平均时间间隔(Churn)
		NUC	关于修改文件所做的唯一修改数(Churn)
	Entropy of changes	Entropy	代码变更熵
	Others	FIX	所做的修改是否更正了当前 bug
		EXP	开发人员的经验评分
		REXP	对文件作修改的开发人员经验评分
		SEXP	对子系统作修改的开发人员经验评分

实验中所采样的度量元均以对象作为基本单位.理论上,选取不同的度量元会对预测性能造成较大影响,且

相同的度量元集对于不同的软件项目,其预测效果亦不同[3,5,7].本文所选取的大部分度量元类型为与程序代码变更(change level/just in time)相关的度量元,因其被认为与软件缺陷的相关度较高,能够作为预测软件缺陷的有效依据[34−36],其主要原因是频繁修改和变更,且变更后传播面较广的代码块,更容易产生 bug.将此类与软件缺陷相关度较高的度量元作为缺陷预测特征集,能够使得实验效果更显著[37].其中,代码流失度量元主要考察软件开发人员在较短时间内反复修改代码所引起的相应变更,本实验所选取的代码流失度量元包含了大部分常规缺陷预测所采用的度量元.

变更熵的概念由 Hassan 等人[1]提出,用于衡量代码修改后其影响的扩散程度.将变更熵的概念推广到一般代码度量元,即为代码熵度量元[34].为了不失一般性,本实验同时也选取了少量其他缺陷预测研究中经常使用的常规缺陷度量元,如对象耦合度、代码行数、历史缺陷数以及开发人员的经验评价等[2,16,34,35].

4.2 数据预处理

由于各种度量元纲量不统一,且可能存在线性相关性,其会造成预测模型训练难以收敛以及预测精度下降等问题,因此需进行数据预处理.此外,由于实验中所选择的缺陷度量元大部分为代码变更相关的度量元,而相关研究表明,对此类度量元作标准化处理后具备更好的缺陷预测效果[37].不失一般性,在模型训练之前对数据进行标准化处理,统一纲量,并在数据层面消除其可能的相关性.实验中使用线性标准化方法(MIN-max normalization)统一度量元的纲量.此外,由于训练样本具有明显的类标记不平衡特性,因此通过对多数类样本实施随机下采样[38]方法,从而将其数量降低到与少数类样本相当的程度.再者,采用主成分分析方法(PCA)[39],选择累积方差超过 95%的主成分,对标准化后的数据进行主成分投影,并将投影后的数据分别作为训练数据和测试数据,以消除不同度量元之间可能存在的潜在线性相关性.

4.3 模型仿真实现

在 Python 3.5+Anaconda+Ubuntu14.04 下实现了仿真系统原型,其中,逻辑回归模型使用了 Liblinear 工具包[40]中的 python 接口,通过设置损失函数和惩罚正则项参数来构造基本的 ElasticNet 逻辑回归分类器.计算训练样本实例权重所需的 dirichlet 分布和核密度估计分别使用了 numpy 工具包中的 dirichlet 函数和 scipy 工具包中的 gaussian_kde 函数.

4.4 实验设计

在 CPDP 设定中,对于来自不同项目的数据集构造两两组合的实验方法.例如,对于组合(JDT,PDE),使用 JDT 数据集作为源缺陷数据集,PDE 作为目标缺陷数据集.基于相同的数据预处理,对每对组合(s,t)构造 3 种不同的实验设定如下.

(1) 不使用领域适配的 CPDP:使用源缺陷数据集训练预测模型,并直接将该模型用于目标缺陷数据进行缺陷预测活动,记为($s\to t$),记录其预测性能,作为预测性能基线(baseline).在该设定中,将所有源缺陷数据样本作为训练数据,随机选择 10%的目标缺陷数据样本作为测试数据.

(2) 基于领域适配的 CPDP:使用本文提出的方法构造预测模型,记为($\bar{\boldsymbol{s}}\to\boldsymbol{t}$ This).为了模拟实际情况,假设仅有 20%的目标样本为可用于训练的样本,并使用全部源缺陷样本作为训练数据,并实施实例加权.随机选择 10%的剩余目标缺陷样本作为未标记的测试数据.在相同设置下,同时考察 Ma 等人[17]提出的迁移贝叶斯模型的预测方法,记为($\bar{\boldsymbol{s}}\to\boldsymbol{t}$ TNB),同时保持该模型中的实例加权不变,将分类器置为本方法所使用的逻辑回归模型,记为($\bar{\boldsymbol{s}}\to\boldsymbol{t}$ TLR),与上述两种方法进行横向比较.

(3) WPDP:同时提供基于目标缺陷数据集的 WPDP 作为 CPDP 的对比参考,记为 $\boldsymbol{t}\to\boldsymbol{t}$.在该设定中,从目标缺陷数据集中随机选择 20%和 10%的样本分别作为训练和测试数据,以模拟训练数据不足的场景;同时,随机选择 50%和 10%的样本分别作为训练和测试数据,以模拟训练数据充足的场景.

对于每对数据集组合,使用精确率(precision)、召回率(recall)和 $F1$ 值[1,6,16,35]来衡量上述 3 种实验设定中的预测效果.由于精确率和召回率之间存在相互权衡的问题,使用其中任意一个均无法衡量模型的真实效果,因此,使用能够同时考察精确率和召回率的 $F1$ 值,从而能够更客观地分析模型预测效果.

由于实验设定中存在随机性,因此对每对数据集组合,重复运行上述 3 种实验设定共 50 次,记录其平均 $F1$ 值作为最终预测效果.

4.5 实验结果及分析

通过上述 3 种不同的实验设定策略来验证本方法的有效性,并进行如下分析讨论.

1) 基于领域适配的跨项目缺陷预测的有效性分析

通过将实验设置(1)设定为基线,将实验设置(3)设定为参考,并与 TNB 方法进行横向比较.可以看出,相对于性能基准,基于领域适配的 CPDP 预测方法在性能上有明显提高,且优于 TNB 方法,能够接近和达到 WPDP 的性能,其结果见表 3~表 10.

Table 3 $F1$ score on combinations of eclipse JDT over other projects in AEEEM

表 3 JDT 与 AEEEM 中其他项目组合的预测 $F1$ 值

数据集组合	$s \to t$	$\bar{s} \to t$ (TLR)	$\bar{s} \to t$ (TNB)	$\bar{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(JDT,Mylyn)	0.331 6	0.621 5	0.600 9	0.512 7	0.401 4	0.460 3
(JDT,Luce)	0.412 5	0.510 3	0.514 2	0.531 3	0.521 8	0.523 1
(JDT,Equ)	0.282 1	0.491 2	0.521 6	0.551 3	0.512 7	0.615 2
(JDT,PDE)	0.321 9	0.412 7	0.402 1	0.613 7	0.479 1	0.501 2
Average	0.337 0	0.508 9	0.509 7	0.552 3	0.478 8	0.525 0

Table 4 $F1$ score on combinations of Mylyn over other projects in AEEEM

表 4 Mylyn 与 AEEEM 中其他项目组合的预测 $F1$ 值

数据集组合	$s \to t$	$\bar{s} \to t$ (TLR)	$\bar{s} \to t$ (TNB)	$\bar{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Mylyn,JDT)	0.473 1	0.512 3	0.502 9	0.552 4	0.522 8	0.531 3
(Mylyn,Luce)	0.302 7	0.513 5	0.525 4	0.512 9	0.437 9	0.541 1
(Mylyn,Equ)	0.382 2	0.562 2	0.552 7	0.551 3	0.463 2	0.562 1
(Mylyn,PDE)	0.359 1	0.313 6	0.418 3	0.422 7	0.405 2	0.412 6
Average	0.379 3	0.475 4	0.499 8	0.509 8	0.457 3	0.511 8

Table 5 $F1$ score on combinations of Lucene over other projects in AEEEM

表 5 Lucene 与 AEEEM 中其他项目组合的预测 $F1$ 值

数据集组合	$s \to t$	$\bar{s} \to t$ (TLR)	$\bar{s} \to t$ (TNB)	$\bar{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Lucene,JDT)	0.373 1	0.380 4	0.351 1	0.230 1	0.414 1	0.541 3
(Luce,Mylyn)	0.223 1	0.360 3	0.440 1	0.431 6	0.425 2	0.524 5
(Lucene,Equ)	0.450 2	0.413 4	0.371 3	0.379 4	0.245 9	0.473 2
(Lucene,PDE)	0.321 4	0.431 2	0.403 0	0.578 2	0.421 6	0.515 1
Average	0.342 0	0.396 3	0.391 4	0.404 8	0.376 7	0.513 5

Table 6 $F1$ score on combinations of Equinox over other projects in AEEEM

表 6 Equinox 与 AEEEM 中其他项目组合的预测 $F1$ 值

数据集组合	$s \to t$	$\bar{s} \to t$ (TLR)	$\bar{s} \to t$ (TNB)	$\bar{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Equ,JDT)	0.314 2	0.417 4	0.526 1	0.521 5	0.512 5	0.574 5
(Equ,Mylyn)	0.353 1	0.507 2	0.442 1	0.502 4	0.395 1	0.464 1
(Equ,Luce)	0.294 2	0.552 1	0.522 5	0.461 5	0.412 1	0.521 5
(Equ,PDE)	0.351 7	0.425 1	0.448 1	0.543 5	0.452 2	0.382 0
Average	0.328 3	0.475 5	0.484 7	0.507 2	0.443 0	0.485 5

Table 7 $F1$ score on combinations of PDE over other projects in AEEEM

表 7 PDE 与 AEEEM 中其他项目组合的预测 $F1$ 值

数据集组合	$s \to t$	$\bar{s} \to t$ (TLR)	$\bar{s} \to t$ (TNB)	$\bar{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(PDE,JDT)	0.461 2	0.524 1	0.538 4	0.682 6	0.502 8	0.631 2
(PDE,Mylyn)	0.362 1	0.351 8	0.370 2	0.321 9	0.363 3	0.442 2
(PDE,Luce)	0.342 1	0.462 1	0.462 1	0.496 2	0.424 2	0.475 1
(PDE,Equ)	0.302 2	0.609 2	0.559 2	0.582 1	0.482 2	0.502 1
Average	0.366 9	0.486 8	0.482 5	0.520 7	0.443 1	0.512 7

Table 8 *F*1 score on combinations of Bugzillaover other projects in Kamei

表 8 Bugzilla 与 Kamei 中其他项目组合的预测 *F*1 值

数据集组合	$s \to t$	$\overline{s} \to t$ (TLR)	$\overline{s} \to t$ (TNB)	$\overline{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Bugz,Colum)	0.282 1	0.392 1	0.324 1	0.422 2	0.401 1	0.461 1
(Bugz,Platf)	0.302 7	0.341 9	0.360 1	0.312 6	0.257 1	0.324 1
(Bugz,Moz)	0.270 5	0.251 7	0.277 1	0.289 1	0.213 1	0.261 2
(Bugz,PSQL)	0.346 1	0.539 1	0.519 5	0.532 7	0.422 1	0.522 1
Average	0.300 4	0.381 2	0.370 2	0.389 2	0.323 4	0.392 1

Table 9 *F*1 score on combinations of Columba over other projects in Kamei

表 9 Columba 与 Kamei 中其他项目组合的预测 *F*1 值

数据集组合	$s \to t$	$\overline{s} \to t$ (TLR)	$\overline{s} \to t$ (TNB)	$\overline{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Colum,Bugz)	0.492 5	0.592 1	0.671 1	0.532 1	0.552 1	0.541 3
(Colum,Platf)	0.294 1	0.251 8	0.220 1	0.251 1	0.224 1	0.264 3
(Colum,Moz)	0.212 6	0.217 6	0.267 1	0.286 1	0.224 2	0.325 1
(Colum,PSQL)	0.406 1	0.409 1	0.409 5	0.473 1	0.433 1	0.442 5
Average	0.351 3	0.367 7	0.392 0	0.385 6	0.358 4	0.393 3

Table 10 *F*1 score on combinations of Platform over other projects in Kamei

表 10 Platform 与 Kamei 中其他项目组合的预测 *F*1 值

数据集组合	$s \to t$	$\overline{s} \to t$ (TLR)	$\overline{s} \to t$ (TNB)	$\overline{s} \to t$ (This)	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
(Platf,JDT)	0.612 1	0.644 1	0.612 3	0.622 6	0.631 6	0.691 5
(Platf,Colum)	0.405 1	0.420 1	0.413 6	0.421 6	0.421 5	0.414 2
(Platf,Moz)	0.152 1	0.302 3	0.271 6	0.227 6	0.175 1	0.295 1
(Platf,PSQL)	0.222 6	0.316 5	0.356 1	0.522 1	0.512 5	0.502 7
Average	0.348 0	0.420 8	0.413 4	0.448 5	0.435 2	0.475 9

表 3~表 10 给出了数据集 AEEEM 和 Kamei 中的实验结果,可以看出,基于本方法的 CPDP 模型性能明显优于基准性能,且在大部分项目组合中优于 TNB 方法和 WPDP 方法(20%:10%).例如,针对数据集 AEEEM,表 3 中使用本方法从项目 JDT 适配到其他项目的平均 *F*1 是 0.552 3,高于基准性能的 0.337 0、TLR 方法的 0.508 9、TNB 方法的 0.509 7 和 WPDP(20%:10%)的 0.478 8;针对数据集 Kamei,表 10 中使用本方法从项目 Platform 适配到其他项目的平均 *F*1 是 0.448 5,高于基准性能的 0.348 0、TLR 方法的 0.420 8、TNB 方法的 0.413 4 和 WPDP(20%:10%)预测性能的 0.435 2.其主要原因是,基准模型未考虑不同项目缺陷样本之间的分布差异;而 TLR 和 TNB 方法中仅通过极值、均值、标准差等有限的特征来衡量不同项目之间的差异性,并未从整体上进行考察,因此并不能完全反映度量元在数据集中所有取值的分布特征;而本方法通过度量元在数据集中的整体分布密度考察不同项目的分布差异性,其衡量方式更完备和客观.这意味着结合本文提出的领域适配技术,能够更好地利用当前充足的数据知识,实现缺陷数据的复用,以提高 CPDP 的性能.

此外,实验中也包含了 50%:10%的 WPDP 设定(50%的训练数据和 10%的测试数据),用于模拟训练数据充足的场景.从实验结果来看,基于本领域适配的预测方法在某些项目组合下同样占优.例如表 3 中,本方法的平均 *F*1 为 0.552 3,高于 WPDP(50%:10%)的 0.525 0.而在某些项目组合中,WPDP(50%:10%)性能占优,例如表 8 中,其平均 *F*1 值为 0.392 1,高于本方法的 0.389 2;表 9 中,其平均 *F*1 值为 0.393 3,高于本方法的 0.385 6.但可见差距并不大,这意味着结合本领域适配技术,其 CPDP 性能能够接近或达到训练数据充足情形下的 WPDP 预测性能.虽然在表 5 中 WPDP(50%:10%)性能优势明显,其主要原因是,源项目数据严重不平衡(9.29%)且样本容量较低,导致仅有少量样本可用于领域适配.

表 11 给出了 3 种实验设定下 50 轮实验的威尔科克森符号秩配对检验结果[41],用于比较 3 种实验设定性能的统计差异化(*p* 值小于 0.05,则认为差异化明显).由表 11 可知,结合了领域适配方法的缺陷预测性能与基准和 WPDP(20%:10%)相比,其 *p* 值分别为 0.000 025 8 和 0.000 112 7,差异化明显,即性能有显著提高;而与 WPDP(50%:10%)相比,其 *p* 值为 0.500 844 3,差异化并不明显,即可认为二者性能具备可比性.

Table 11 *p*-values of our approach compared with the baseline and WPDP settings in terms of *F*1 score

表 11 *F*1 值下本方法与基准方法以及 WPDP 方法的威尔科克森配对检验 *p* 值

Approaches	$s \to t$	$t \to t$ (20%:10%)	$t \to t$ (50%:10%)
$\bar{s} \to t$	0.000 025 8	0.000 112 7	0.500 844 3

2) 基于权重的训练集筛选

由于源项目缺陷数据集上附加的实例权重决定了其对于模型参数训练的影响因子,权重越大,则影响越大.因此,当数据集数量能够得以保证时,一种更好的策略是从源缺陷数据集样本中筛选出权重高于某个设定阈值的子集,可认为该子集对于模型学习具有更好的适配效果.但若阈值设定过低,则其效果将不明显;若阈值设定过高,则筛选后的训练样本数量将得不到保证,容易造成过拟合.图 4 显示了(PDE,JDT)组合下保持训练数据集不变,不同阈值对于 *F*1 值的影响.由图可见,当阈值设定为 0.6 时,其 *F*1 值达到最大,然后开始衰减.其主要原因是,过高的阈值将会极大地影响可利用的训练样本数量,从而降低预测性能.

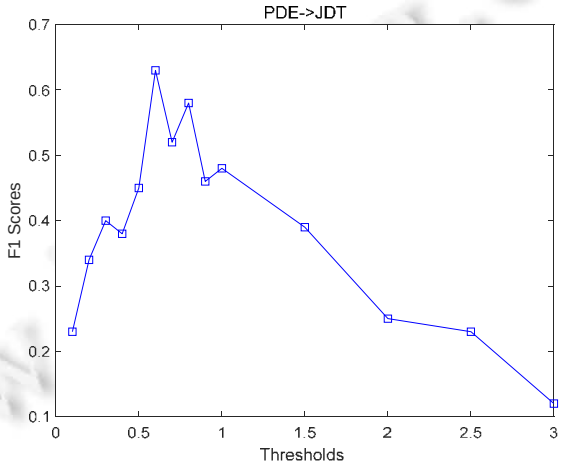

Fig.4 *F*1 scores of the pair (PDE,JDT) under different threshold settings

图 4 (PDE,JDT)配对中基于不同阈值设定的预测 *F*1 值

3) 实验原理分析

由于来自不同项目的缺陷数据,其分布性质上的差异是造成 CPDP 性能低下的主要原因,而基于本文领域适配的预测方法,其主要思想是通过对源项目数据样本附加权重,以降低分布差异.在预测模型参数的训练过程中,当某个源项目缺陷样本与目标项目存在较大的分布差异时,其模型目标函数优化算法(如传统梯度下降算法等)会通过样本附加的权重自动调整模型参数的变化大小以及变化方向,以适应分布差异所带来的影响.此外,实验中选用的数据集对应的软件项目具有相似的体系结构(面向对象),这也意味着不同数据集缺陷样本之间存有潜在的共性.从表 3 中可知,使用本领域适配预测方法的数据组合(JDT,PDE),其性能相比 AEEEM 数据集中其他 3 种设定分别提高了 0.291 8,0.134 6 和 0.112 5,其提高程度比起其他数据组合要显著;表 7 中的数据组合(PDE,JDT)分别提高了 0.221 4,0.179 8 和 0.051 4,其提高程度同样显著.我们认为,JDT 和 PDE 与其他组合相比具有更多的分布共性,因此适配效果更好.

图 5 显示了(PDE,JDT)组合和(Mylyn,PDE)组合领域适配权重密度直方图.可以看出,两者均只有少部分样本其权重超过 1.0.而对于(PDE,JDT)组合,其具有较高权重的样本明显多于(Mylyn,PDE)组合,因此对于(PDE,JDT)组合,将有更多的具有高影响因子的样本能够用于模型训练,从而获得更好的预测效果.

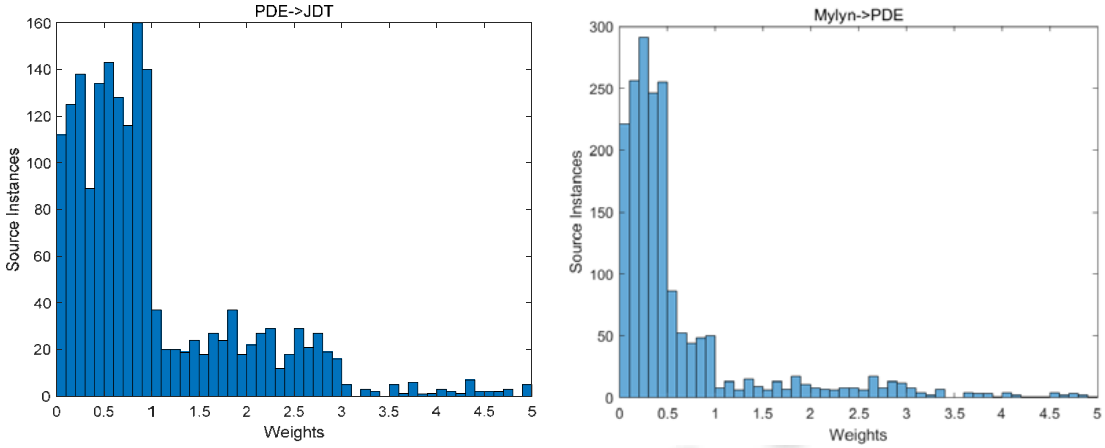

Fig.5 Histogram of the density of the source project instances on different weights for both (PED,JDT) and (Mylyn,PDE)

图 5 基于(PED,JDT)和(Mylyn,PDE)的源项目实例权重密度直方图

4) 数据类不平衡问题

由于缺陷数据集具有不平衡的特点(标记为缺陷的样本大于标记为无缺陷的样本),直接在其基础上训练预测模型会使得模型参数严重偏向多数类样本,从而造成过拟合问题,使得预测效果降低.本文实验中采用了随机下采样操作,将多数类样本削减至与少数类样本相当的程度.Kamei 等人[35,42]研究了缺陷预测中下采样方法对预测效果的影响,结论为:严格的 50%缺陷样本比例并非最优的选择,因过量的下采样会丢弃掉重要的样本.针对 Kamei 所使用的数据集,使用逻辑回归的理想缺陷样本比例为 20%~33%[42].使用本实验数据集和本方法在实验设置(2)下进行 10 轮测试并取均值,发现较为理想的缺陷样本比例在 30%~50%间浮动,且平衡性越差的数据集,其理想的缺陷样本比例越小.图 6 显示了(Equinox,JDT)和(Mylyn,PDE)两种组合基于不同缺陷样本比例设定的预测效果曲线,其中,数据集 Equinox 和 JDT 的原缺陷比例分别为 39.69%和 20.66%,其理想缺陷比例为 47%左右;而 Mylyn 和 PDE 的原缺陷比例分别为 13.16%和 14.01%,其理想缺陷比例为 38%左右.

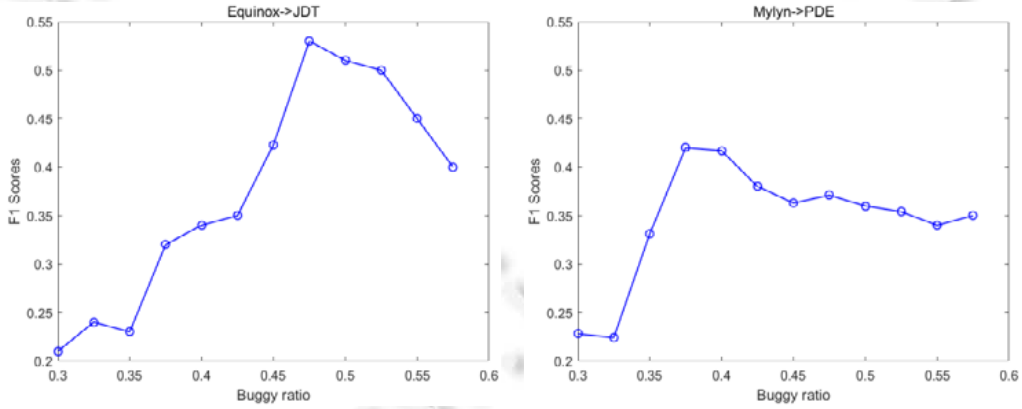

Fig.6 *F*1 scores regards to different ratio settings of buggy samples

图 6 基于不同缺陷样本比率设定的 *F*1 值

5) 概率密度及先验估计

实例加权的领域适配方法中,其权重主要通过估计源项目和目标项目缺陷数据样本的概率密度函数计算得到.对于离散型度量元,由于其对应的离散取值空间往往较大,例如度量元 *numberOfLinesOfCode* 在数据集

PDE 中其取值为[3,2169]中的整数.较大的取值空间会导致大部分点在训练样本中出现的频次为 0(或仅为 1 次、2 次),从而造成密度稀疏且特征不显著,难以刻画数据样本的概率分布特性.此外,由于度量元对应的离散取值空间在不同的域中也可能不同,例如,*numberOfLinesOfCode* 在 Lucene 数据集中其取值为[1,1869]中的整数,从而导致在使用公式(8)计算权重时出现错误.因此,将原始的离散度量元取值域按照对应度量元的语义进行等价类划分,重构为定序空间(ordinal data space).例如,对于度量元 *numberOfLinesOfCode*,可将其离散空间[1,2169]划分为 50 个不同等价类,分别表示 50 个不同的源代码规模等级,即原取值域[1,2169]重构为只含有 50 个元素的定序[1,50],由此获得具有更合理粒度的新取值空间.

基于新的定序取值空间,计算度量元取值频次以估计概率质量函数,并采用狄利克雷分布以融入软件开发人员对于需要估计度量元的经验性认知,以避免由于训练数据集本身的个体差异和离群点而导致的过拟合.先验知识由超参数向量$\boldsymbol{\alpha}$来决定.基于狄利克雷先验共轭的特性(见第 3.3 节),向量$\boldsymbol{\alpha}$可直观认为是某度量元取其可能取到的所有离散数值在一个先验超数据集(hyper-dataset)中的伪计数(pseudo counts).$\boldsymbol{\alpha}$中元素取值越大,其先验对于模型的影响越显著.当向量$\boldsymbol{\alpha}$中元素取值相等时,退化为对称狄利克雷分布,即认为开发人员对于需要估计密度的度量元缺乏经验认识,且有$\boldsymbol{\alpha}=\alpha\times\boldsymbol{u}$,其中,$\boldsymbol{u}$ 为元素为全 1 的向量,α为浓度系数.α取值越小(在 1.0 附近),先验知识的影响力越小,则度量元的估计密度越集中;反之,则越分散.

构造度量元的定序取值空间和进行先验估计,能进一步减小因样本空间的个体差异带来的过拟合问题.

4.6 方法的局限性分析

针对本方法在两个数据集中的 10 个大型开源软件项目中进行了实验,并取得了较好的效果,但并不排除偶然性.未来将针对更多的开源或商业软件项目数据集进行实践,进一步证明该方法的有效性.

过拟合是基于领域适配的缺陷预测中不可避免的问题,虽然本方法在数据和模型层面对过拟合问题进行了处理,但并不能将其完全消除,尤其是在训练数据样本容量较小且不平衡的情况,因为预测模型所学习的参数无法覆盖数据集中尚未出现的模式.此外,本实验针对数据不平衡问题采用了下采样方法,而不同的数据集所使用的理想下采样强度不同,其也会造成预测效果的不同.再者,由于在密度估计中使用了用户对度量元的先验知识以减轻密度估计的过拟合,因此具备不同先验知识的用户使用本方法可能产生不同的迁移效果.

由于本方法基于领域适配技术,根据迁移学习理论[15],其要求源数据集和目标数据集之间具备一定的分布相似性才能获得较好的效果,因此,使用不同的项目数据集以及不同的度量元集会产生不同的预测效果.此外,本方法实验中仅从与缺陷的相关性角度出发选择度量元,并未涉及到专门针对领域适配算法的特征选择策略.虽然在机器学习领域中已有不少特征选择方法[43],但其大多基于训练样本充足的前提.而在 CPDP 设定中,仅有少量可用于训练的目标样本,因此,寻求高效普适的 CPDP 特征选择算法仍是值得研究的方向.

5 总结与展望

本文通过使用不同项目的缺陷样本数据的概率密度计算实例权重,通过实例权重去影响预测模型的参数学习过程,以实现跨项目的软件缺陷预测.此外,描述了在模型中加入稀疏和正则约束,在缺陷样本概率密度估计过程中加入先验知识,从而分别在预测模型和领域适配层面减轻过拟合问题.在 10 个大型开源软件项目上对本方法进行实证,从数据集、数据预处理、实验结果等多个角度针对 3 种不同的实验设定策略进行分析,证明了基于实例加权的领域适配方法,其性能优于同类方法,显著优于基准性能,且与 WPDP 预测性能相当.根据本方法理论及其实验分析,对 CPDP 预测进行如下总结.

(1) 仅依赖缺陷样本数量以提高 CPDP 模型的泛化能力是不够的,因为来自不同项目的缺陷数据样本,其分布存在较大差异,从而导致泛化和预测性能低下.

(2) 在 CPDP 设定下,提高源项目缺陷样本数量将有助于提高模型的预测性能,因其能够覆盖更多的缺陷产生模式.

(3) 预测模型的过拟合是不可避免的,通过稀疏和正则惩罚方法降低预测模型复杂度,在领域适配过程中加入先验知识能够有效地减轻过拟合,从而让预测模型具有更好的泛化能力.此外,不平衡的数据集

也会造成过拟合问题,对多数类样本采取下采样等方法能够有效减轻不平衡样本造成的过拟合,但过量的下采样是不可取的.另一种更好的做法是使用领域适配技术从其他项目数据集中收集样本,对当前数据集的少数类样本进行“上采样”,在今后的工作中将对该思想作进一步实证.

(4) 需要利用少量的已标记目标项目缺陷样本以调整预测模型的参数训练过程,从而实现预测模型的适配.获取 15%~25%比率的已标记目标项目缺陷样本用于监督预测模型的领域适配训练,在具体实践中是较为合理和有效的.

除了针对本方法的局限性做进一步深入研究外,将其他迁移学习以及领域适配方法,如基于核的迁移学习方法、基于深度学习的领域适配方法用于 CPDP,是未来研究的方向之一.此外,将本方法用于软件工程中的其他传统任务,如错误报告定位(bug report localization)、story point 预测、自动化测试用例生成以及自动化需求跟踪(software requirement traceability)等,也将列入未来的研究中.

References:

[1] Hassan AE. Predicting faults using the complexity of code changes. In: Proc. of the Int’l Conf. on Software Engineering. 2009. 78−88. [doi: 10.1109/ICSE.2009.5070510]

[2] Kim S, Whitehead EJ, Jr Zhang Y. Classifying software changes: Clean or buggy. IEEE Trans. on Software Engineering, 2008, 34(2):181−196. [doi: 10.1109/TSE.2007.70773]

[3] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans. on Software Engineering, 2007,33(1):2−13. [doi: 10.1109/TSE.2007.256941]

[4] Zimmermann T, Nagappan N. Predicting defects using network analysis on dependency graphs. In: Proc. of the Int’l Conf. on Software Engineering. 2008. 531−540. [doi: 10.1145/1368088.1368161]

[5] Nagappan N, Ball T, Zeller A. Mining metrics to predict component failure. In: Proc. of the Int’l Conf. on Software Engineering. 2006. 452−461. [doi: 10.1145/1134285.1134349]

[6] Chen X, Gu Q, Liu WS, Liu SL, Ni C. Software defect prediction. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):1−25 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

[7] Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proc. of the ICSE. 2008. 181−190. [doi: 10.1145/1368088.1368114]

[8] Wu R, Zhang H, Kim S, Cheung S. Relink: Recovering links between bugs and changes. In: Proc. of the Joint Meeting of the European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering. 2011. 15−25. [doi: 10.1145/2025113.2025120]

[9] Lee T, Nam J, Han D, Kim S, Hoh IP. Micro interaction metrics for defect prediction. In: Proc. of the Joint Meeting of the European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering. 2011. 311−321. [doi: 10.1145/2025113.2025156]

[10] Rahman F, Posnett D, Devanbu P. Recalling the imprecision of cross-project defect prediction. In: Proc. of the Int’l Symp. on the Foundations of Software Engineering. 2012. 1−11. [doi: 10.1145/2393596.2393669]

[11] Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B. Cross-project defect prediction: A large scale experiment on data vs domain vs process. In: Proc. of the Joint Meeting of the European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering. 2009. 91−100. [doi: 10.1145/1595696.1595713]

[12] Turhan B, Menzies T, Bener AB, Di Stefano J. On the relativevalue 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]

[13] Chen X, Wang LP, Gu Q, Wang Z, Ni C, Liu WS, Wang QP. A survey on cross-project software defect prediction methods. Journal of Chinese Computers, 2018,41(1):254−274 (in Chinese with English abstract).

[14] Daume III H, Marcu D. Domain adaptation for statistical classifiers. Journal off Artificial Intelligence Research, 2006,26(1): 101−126.

[15] Zhuang FZ, Luo P, He Q, Shi ZZ. Survey on transfer learning research. Ruan Jian Xue Bao/Journal of Software, 2015,26(1):26−39 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

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

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

[18] He JY, Meng ZP, Chen X, Wang Z, Fan XY. Semi-supervised ensemble learning approach for cross-project defect prediction. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1455−1473 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5228.htm [doi: 10.13328/j.cnki.jos.005228]

[19] Rahman F, Devanbu P. How, and why, process metrics are better. In: Proc. of the Int'l Conf. on Software Engineering. 2013. 432−441. [doi: 10.1109/ICSE.2013.6606589]

[20] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans. on Knowledge and Data Engineering, 2010,22(10):1345−1359. [doi: 10.1109/TKDE.2009.191]

[21] Dai W, Xue GR, Yang Q, Yu Y. Transferring naive Bayes classifiers for text classification. In: Proc. of the Association for the Advance of Artificial Intelligence, Vol.1. 2007. 540−545.

[22] Jiang J, Zhai C. Instance weighting for domain adaptation in NLP. In: Proc. of the Annual Meeting on the Association for Computational Linguistics. 2007. 264−271.

[23] Pan SJ, Kwok JT, Yang Q. Transfer learning via dimensionality reduction. In: Proc. of the Association for the Advance of Artificial Intelligence. 2008. 677−682.

[24] Dai W, Yang Q, Xue GR, Yu Y. Boosting for transfer learning. In: Proc. of the Int'l Conf. on Machine Learning. 2007. 193−200. [doi: 10.1145/1273496.1273521]

[25] Bickel S, Brückner M, Scheffer T. Discriminative learning for differing training and test distributions. In: Proc. of the Int'l Conf. on Machine Learning. 2007. 81−88. [doi: 10.1145/1273496.1273507]

[26] Wachinger C, Reute M. Domain adaptation for Alzheimer's disease diagnostics. Journal of NeuroImage, 2016,139:470−479. [doi: 10.1016/ j.neuroimage.2016.05.053]

[27] Japkowicz N, Stephen S. The class imbalance problem: A systematic study. Intelligent Data Analysis, 2002,6(5):429−449.

[28] Shimodaira H. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference, 2000,90(2):227−244. [doi: 10.1016/S0378-3758(00)00115-4]

[29] Goeman JJ. L_1 penalized estimation in the Cox proportional hazards model. Biometrical Journal, 2010,52(1):70−84. [doi: 10.1002/bimj.200900028]

[30] Zou H, Hastie T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, 2005,67(2):301−320. [doi: 10.1111/j.1467-9868.2005.00503.x]

[31] Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. Journal of Machine Learning Research, 2003,3(1):993−1022.

[32] Scott DW. Multivariate density estimation and visualization. In: Handbook of Computational Statistics. 2011. 549−569. [doi: 10.1007/978-3-642-21551-3_19]

[33] Silverman BW. Density estimation for statistics and data analysis. In: Monographs on Statistics and Applied Probability. London: Chapman & Hall, 1986.

[34] D'Ambros M, Lanza M, Robbes R. An extensive comparison of bug prediction approaches. In: Proc. of the Working Conf. on Mining Software Repositories. 2010. 31−41. [doi: 10.1109/MSR.2010.5463279]

[35] Kamei Y, Shihab E, Adams B, Hassan AE, Mockus A, Sinha A, Ubayashi N. A large-scale empirical study of just-in-time quality assurance. IEEE Trans. on Software Engineering, 2013,39(6):757−773. [doi: 10.1109/TSE.2012.70]

[36] Schulte L, Sajnani H. Active files as a measure of software maintainability. In: Proc. of the Int'l Conf. on Software Engineering. 2014. 4−43. [doi: 10.1145/2591062.2591176]

[37] Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proc. of the Int'l Conf. on Software Engineering. 2005. 15−21. [doi: 10.1109/ICSE.2005.1553571]

[38] He H, Garcia EA. Learning from imbalanced data. IEEE Trans. on Knowledge and Data Engineering, 2009,21(9):1263−1284. [doi: 10.1109/TKDE.2008.239]

[39] Schneeweiss H, Mathes H. Factor analysis and principal components. Journal of Multivariate Analysis, 1995,55(1):105−124. [doi: 10.1006/jmva.1995.1069]

[40] Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 2008,9(9):1871−1874. http://www.csie.ntu.edu.tw/~cjlin/liblinear

[41] Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bulletin, 1945,1(6):80−83. [doi: 10.1007/978-1-4612-4380-9_16]

[42] Kamei Y, Monden A, Matsumoto S, Kakimoto T, Matsumoto K. The effects of over and under sampling on fault-prone module detection. In: Proc. of the Int'l Symp. on Empirical Software Engineering and Measurement. 2007. 20−21. [doi: 10.1109/ESEM.2007.28]

[43] Guyon I, Elisseeff A. An introduction to variable and feature selection. The Journal of Machine Learning Research, 2003,3(1): 1157−1182.

附中文参考文献:

[6] 陈翔,顾庆,刘望舒,刘树龙,倪超.静态软件缺陷预测方法研究.软件学报,2016,27(1):1−25. http://www.jos.org.cn/1000-9825/4923.htm [doi: 10.13328/j.cnki.jos.004923]

[13] 陈翔,王莉萍,顾庆,王赞,倪超,刘望舒,王秋萍.跨项目软件缺陷预测方法研究综述.计算机学报,2018,41(1):254−274.

[15] 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. http://www.jos.org.cn/1000-9825/4631.htm [doi: 10.13328/j.cnki.jos.004631]

[18] 何吉元,孟昭鹏,陈翔,王赞,樊向宇.一种半监督集成跨项目软件缺陷预测方法.软件学报,2017,28(6):1455−1473. http://www.jos.org.cn/1000-9825/5228.htm [doi: 10.13328/j.cnki.jos.005228]

陈曙(1981－),男,湖北武汉人,博士,讲师,CCF 专业会员,主要研究领域为软件工程,需求工程,软件缺陷预测.

刘童(1991－),女,硕士,主要研究领域为软件工程,软件缺陷预测.

叶俊民(1965－),男,博士,教授,CCF 高级会员,主要研究领域为软件工程,可信软件工程,软件缺陷定位.