文章编号：2095-2163(2019)05-0354-04 中图分类号：TP311.5 文献标志码：A

基于迁移学习的软件缺陷预测算法研究

何金虎，吴翔虎，曲明成
（哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001）

摘 要： 软件缺陷预测技术可以用于预测软件缺陷是否存在以及其可能存在的数目，以决定软件是否可以交付，对于软件性能的提升和质量的保证有着重要的意义。迁移学习则可以利用不同软件项目中的数据，进行跨项目的软件缺陷预测工作，以应对传统缺陷预测算法中数据不足的问题。本文首先阐述了缺陷预测和迁移学习的相关理论研究现状及其分类，然后对现有的 TrAdaboost 算法进行优化，修改了迭代分类器的评估指标，并结合实验证明了其合理性和优越性。

关键词： 迁移学习；缺陷预测；不平衡数据

Research on software defect prediction algorithm based on transfer learning

HE Jinhu, WU Xianghu, QU Mingcheng
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 Software defect prediction technology can be used to predict the existence of software defects and the number of possible defects to determine whether software can be delivered. It is of great significance for software performance improvement and quality assurance. Transfer learning can use the data in different software projects to perform software defect prediction work across projects to solve the problem of insufficient data in traditional defect prediction algorithms. This paper first expounds the current theoretical research and classification of defect prediction and transfer learning, then optimizes the existing TrAdaboost algorithm, modifies the evaluation index of the iterative classifier, and proves its rationality and superiority by combining experiments.

【Key words】 transfer learning; defect prediction; unbalanced data

0 引 言

软件缺陷，通常指的是软件实现未能满足客户需求。通过对软件缺陷的预测和发现，可以更好地提高软件产品质量。现有的软件缺陷预测方法多为通过对大量已标记的软件缺陷样本的训练，构建一个软件缺陷预测模型，再对一个新的软件项目进行预测，以判定其内部各个模块中是否包含缺陷。已有的模型构建算法通常是应用传统的机器学习算法，并结合 bagging 和 Boosting 的思想，对原始数据集进行训练。但在实际项目中，一个新的问题往往会因为缺少原始数据集样本而无法进行预测，这也正是迁移学习产生的原因。

迁移学习可以利用已经学习到的知识对新的问题进行预测，具体到软件缺陷预测问题中，就可以利用已有的其它项目或公司的已标记软件缺陷数据，选用合适的数据预处理和模型训练算法，对一个新的项目的缺陷进行模型构建和预测。现有的跨项目缺陷预测技术，大多集中在数据预处理阶段，如通过度量辅助数据集和目标数据集之间样本的相关性，筛选合适的辅助样本数据，然后使用传统的机器学习方法进行模型构建[1]；或者通过降维的方式将 2 个数据集的数据映射到同一个维度空间等。

同上文中提出的研究方法不同，本文拟结合已有的 TrAdaboost[2] 迁移学习算法，并针对软件缺陷数据集中存在的数据不平衡的问题进行特殊处理，对迭代过程中每个阶段的弱分类器的评估指标进行了修改，同时通过一系列的实验数据，验证了其相对于现有算法的优越性和合理性。本文对此将展开研究论述如下。

1 相关理论和技术

1.1 迁移学习定义

文中，将讨论迁移学习中所涉及到的域和任务的定义。域通常包含 2 部分内容，即特征空间 X 与特征空间的边缘分布函数 $P(x)$，研究中就将域表示

基金项目： 国家自然科学基金（61402131）。

作者简介： 何金虎（1990-），男，硕士研究生，主要研究方向：软件工程应用、机器学习；吴翔虎（1968-），男，博士，教授，CCF 高级会员，主要研究方向：嵌入式计算、操作系统、高可靠软件工程；曲明成（1980-），男，博士，硕士生导师，主要研究方向：自动化软件工程、嵌入式计算。

收稿日期： 2019-06-11

为 $D=\{X,P(x)\}$。当域不同时，则往往拥有不同的边缘分布函数或特征空间，就软件缺陷预测问题而言，2 个域的特征空间是一致的。任务是同域相结合的，同样包含 2 部分，即标签空间 Y 与目标预测函数 $f(x)$，该目标预测函数又可表示为 $P(Y|X)$，因此任务往往可以表示为 $T=\{Y,P(X|Y)\}$。

目前，迁移学习算法处理的通常是只有一个辅助领域和一个目标领域的问题，其中辅助领域中存在大量含标签的样本，目标领域中仅包含少量含目标样本，或不含样本。两者所面临的任务相同或存在一定关联。

结合上述定义的符号，迁移学习定义[3]如下：已知辅助领域 Ds 和辅助任务 Ts、目标领域 Dt 和目标任务 Tt，且辅助领域和目标领域之间、辅助任务和目标任务之间至少存在一个不同点，此时，迁移学习即为使用辅助领域 Ds 和 Ts 中的知识提升或优化目标领域 Dt 中目标预测函数 $ft(x)$ 的学习效果。

只有当域和任务两者之一存在差异时，才会用到迁移学习，否则直接应用传统的机器学习方法即可。同时，研究中还要求目标领域存在较少、甚至不存在已标记数据，否则同样不需要使用迁移学习方法。相对于传统的机器学习方法而言，迁移学习构建的模型就性能和准确性而言还是有一定差距的，当传统的机器学习方法条件可以得到满足时，优先使用机器学习方法来构建模型和获得任务中的预测函数 $f(x)$。

1.2　软件缺陷预测技术概述

软件作为一款虚拟的产品，虽然不会因为硬件的损耗而产生错误或者性能下降，但却有可能因为本身固有的问题，在运行的过程中失效或者产生错误的结果，造成经济损失。

为了尽可能地避免这种损失，预测尚未被发现的缺陷，软件缺陷预测技术则应运而生。该项技术研发时，可以将预测结果应用于软件测试的指导工作，根据待预测模块中包含缺陷的概率，以及预测的缺陷数目，对包含缺陷概率较大或者数目较多的模块进行优先和重点测试，从而在测试资源和时间有限的情况下尽可能地发现更多的缺陷，保证软件产品的质量。

研究表明，随着软件开发过程成熟度的提高，软件缺陷的数目也在随之下降；此外，开发人员经验是否丰富，软件结构是否合理，软件规模是否庞大等一系列因素也都在影响着软件缺陷的存在数目，如果可以通过建模，发现并量化各种因素同缺陷之间的关系，就可以利用统计学的方法来进行软件缺陷预测的工作。软件度量指标和软件质量以及软件缺陷预测模型之间的关系如图 1 所示，其中软件度量元包括原始数据以及相关的域和任务，在构建模型过程中，需要对这些数据进行数据预处理，包括降噪、特征选择等，以提高软件缺陷预测模型的正确性。

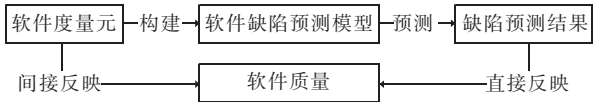

图 1　度量元、预测结果同软件质量关系图

Fig. 1　Metric element, prediction result and software quality relationship diagram

软件缺陷预测技术主要包括静态预测技术和动态预测技术两类[4]。其中，静态预测技术出现得较早，也是当前的研究重点所在，设计原理就是利用软件度量元的统计对缺陷的数目和分布做出预测，常见的分类包括基于软件度量元的缺陷预测技术、软件缺陷分布预测技术、缺陷预测模型技术；动态缺陷预测技术则是针对软件缺陷随着软件工程过程或者软件运行生命周期的变化而变化的趋势进行预测，得到其分布规律。

由于缺陷预测数据集为不平衡数据集，包含有缺陷的样本所占比例通常不足 10%，使用常用的迁移学习算法很难对其做到正确分类，因此需要针对该问题对现有算法加以优化，并使用 AUC[5] 值和缺陷检出率 PD 值对其性能进行评估。

2　问题定义和算法设计

2.1　问题定义

为了更好地理解算法伪代码中的符号含义，现在对研究中可能出现的一些符号进行定义，详见表 1。

表 1　基本符号定义表

Tab. 1　Basic symbol definition table

符号	含义
X_a	辅助领域数据集
X_b	目标领域数据集
Y	类标签空间，可取值为 $\{-1,1\}$
T_a	辅助数据，$T_a \subseteq \{X_a \times Y\}$
T_b	目标数据，$T_b \subseteq \{X_b \times Y\}$
T	训练数据集定义：$T \subseteq \{(X = X_a \cup X_b) \times Y\}$ $T = T_a \cup T_b$
g	$X \rightarrow Y$ 真实预测函数
h	$X \rightarrow Y$ 假设函数，由算法和训练数据通过训练生成
S	测试数据集，不含标签列

其中，训练数据集中辅助领域数据集和目标领域数据集的长度分别为 n 和 m，一般情况下，$n>>m$，

且 m 值较小,因为如果 m 值较大,则说明在目标领域中含有大量的可以用于模型构建和类别预测的训练数据,并可以独立获取预测函数,此时应用传统的机器学习方法即可。就数据分布而言,T_b 和 S 来源于同一个数据域,具有相同的边缘概率分布,而 T_a 则有着与 S 不同的分布。通常意义上,T_a 和 T_b 可以用于预测类似的或者相关的任务,否则 2 个不同分布的数据很难用于同一个问题的训练工作。结合前述内容,研究中可以做出如下问题定义:

给定大量辅助领域含标签数据集 T_a 和少量目标领域含标签数据 T_b,以及待预测未标注的测试数据集 S,T_b 和 S 同分布。无论是 T_a、还是 T_b 数据集,均满足正类数目远远小于负类数目,存在类别不平衡问题。研究中需要利用 T_a 和 T_b 结合算法来构建预测模型,使其可以用于 S 中的标签预测和标注。

2.2 TrAdaboost 优化算法设计与分析

在软件缺陷预测模型构建过程中,选用的数据集包括辅助数据集和目标数据集在内均存在数据不平衡问题,数据集中含有缺陷的模块仅占很小的一部分,这也导致在训练模型过程中,正例和反例不能被同等重视,如果仅仅依据预测正确率,则无法正确反映模型性能。就软件缺陷预测问题而言,研究需要预测出更多的包含缺陷的模块,因此使用缺陷检出率 *PD* 来评估模型能够更好地契合预定目标,同时 *PD* 也是 ROC 曲线坐标轴的纵轴,*FPR* 为横轴。

该算法的优化思想为:在每次执行 TrAdaboost 算法迭代时,不再通过错误率来评估单次迭代分类器的重要性,而是通过简易的 *AUC* 值来度量。这样可以在每次迭代的过程中,有倾向性地提高真正例率,从而得到更优的预测结果。优化后 TrAdaboost 算法的设计代码见如下。

输入:辅助领域数据集 T_a 和目标领域数据集 T_b,及其并集 T, 未标记目标领域测试集 S

弱分类算法 WL(意为 WeakLearn)

迭代次数 N

Step 1 初始化

初始化权重向量 $w^t = (\omega_1^1, \ldots, \omega_{n+m}^1)$,权重可以根据具体情况进行初始化,默认为:

$$\omega_i^1 = \begin{cases} 1/n & \text{当 } i = 1, \ldots, n \\ 1/m & \text{当 } i = n+1, \ldots, n+m \end{cases}$$

Step 2 For $t = 1, \cdots, N$

(1)权值重构, $p^t = \dfrac{w^t}{\sum_{i=1}^{n+m} \omega_i^t}$

(2)使用分布为 p^t 的样本数据集 T 对 WL 进行训练,返回假设 h_t

(3)计算 h_t 在 T_b 上 *AUC* 值, p_i 为预测结果, y_i 为实际标签

$$TP = \sum_i w_i; p_i = 1, y_i = 1$$

$$FP = \sum_i w_i; p_i = 1, y_i = 0$$

$$TN = \sum_i w_i; p_i = 0, y_i = 0$$

$$FN = \sum_i w_i; p_i = 0, y_i = 1$$

$$TPR = TP/(TP + FN)$$

$$FPR = FP/(FP + TN)$$

$$AUC = 1/2 * (1 - FPR + TPR)$$

(4)设置 $\beta_t = (1 - AUC)/AUC$, *AUC* 必须大于 0.51,否则将其置为 0.51

(5)更新权重:

$$\omega_i^{t+1} = \begin{cases} \omega_i^t \beta_t^{|h_t(xi) - c(xi)|}, & \text{当 } i = 1, \ldots, n \\ \omega_i^t \beta_t^{-|h_t(xi) - c(xi)|}, & \text{当 } i = n+1, \ldots, n+m \end{cases}$$

输出:最终分类器为: $h_f(xi) =$

$$\begin{cases} 1, \sum_{t=[\frac{N}{2}]}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(xi) \geqslant \frac{1}{2} \sum_{t=[\frac{N}{2}]}^{T} \left(\log \frac{1}{\beta_t}\right); \\ 0, \text{其它}. \end{cases}$$

本次优化的部分为算法 for 循环内的步骤(3),其中计算的 *AUC* 值并不准确,而是将阈值固定后直接依据分类器的预测结果和实际标签来结合权重进行计算,得到 *TPR* 和 *FPR* 两个值,在此基础上计算点(0,0),(*TPR*,*FPR*),(1,1)以及(1,0)围成的面积,由于阈值不变,因此该面积仅为近似面积,小于实际的值。关于 *AUC* 值的下限设置为 0.51,这是因为若设置 0.5,权值将不会发生变化,则下次迭代的结果会完全一致。

3 实验结果

本实验中,采用的数据集来自于 ESEM2016[6]。ESEM2016 数据集是作者使用增项的 SZZ 算法和 JHawk 工具生成的。目前有研究指出,软件缺陷预测问题常用的数据集还包括 NASA 的数据集,但却存在不开源、前后数据不一致等问题,因此本实验未采用该数据集。

ESEM2016 数据集由 2 部分组成,也就是:ClassLevel、即类级别和 MethodLevel、即方法级别,每一个数据集中则包含有 42 种度量元对应的统计数据。本次实验主要使用 ClassLevel 级别的数据,着重测试不同项目之间的迁移效果。

本文通过对比 TrAdaboost 与优化后的

TrAdaboost 算法在不同数据集上的预测效果，验证了对权值调整后的算法在面对数据集不平衡问题时的有效性和优越性。该实验选择的评估数据包括正确率、*AUC* 值以及 *PD* 值三种。其中，正确率为统计预测结果正确的样本数目所占的比例，*AUC* 值则是通过计算 ROC 曲线下方区域的面积得到；*PD* 又称为错误检出率，$PD = TP/(TP+FN)$。正如前文研究得到的，正确率无法正确反映预测结果的性能，*AUC* 值和 *PD* 值均可反映正例被正确预测的比例，但 *AUC* 值同等对待 2 个类别的预测，而在软件缺陷预测问题中，研究是致力于能够更多地检测出存在的缺陷，同时允许一定数量的误判，即将无缺陷样本预测为有缺陷，因此较好的 *PD* 值度量更符合本次研发的实验要求。

实验选取了 11 个不同的数据集，并按照文件名排序后使相邻的 2 个数据集两两结合，前者作为辅助数据集，后者作为目标数据集，再用其来进行模型的构建和评估，实验结果见表 2。

表 2　TrAdaboost 与优化后 TrAdaboost 实验结果对比

Tab. 2　Comparison of TrAdaboost and optimized TrAdaboost experimental results

辅助数据集名称	目标数据集名称	TrAdaboost			优化后 TrAdaboost		
		错误率/%	*AUC*	*PD*	错误率/%	*AUC*	*PD*
autoplot	drjava	15.6	0.68	0.39	16.3	0.66	0.44
drjava	genoviz	27.7	0.73	0.64	27.8	0.72	0.72
genoviz	htmlunit	17.5	0.79	0.64	27.9	0.75	0.74
htmlunit	jitterbit	11.6	0.70	0.38	10.8	0.67	0.42
jitterbit	jmol	15.7	0.62	0.18	12.8	0.65	0.38
jmol	jmri	11.3	0.66	0.37	9.1	0.67	0.51
jmri	jppf	19.3	0.68	0.42	19.1	0.66	0.47
jppf	jump	17.4	0.65	0.37	15.7	0.68	0.48
jump	omegat	5.2	0.62	0.19	6.9	0.63	0.31
omegat	saros	15.2	0.56	0.22	19.9	0.59	0.29
autoplot	tango	6.9	0.70	0.28	8.3	0.61	0.31

实验结果表明，在数据集不平衡的软件缺陷预测问题中，优化后的算法在大部分情况下 *PD* 值是优于原有的 TrAdaboost 算法的，而 *AUC* 值方面两者则比较接近，较大的 *PD* 值也表示可以有更多的缺陷被预测出来。与此同时还发现，相对于传统的机器学习方法而言，迁移学习的预测效果相对较差，因此在拥有足够的训练数据时，将优先使用传统的机器学习算法。

4　结束语

软件缺陷预测可以对软件的测试和交付提供指导意见，对软件质量的保证具有实际意义。本文参考了 TrAdaboost 算法，结合软件缺陷预测问题的特征，对 TrAdaboost 算法进行了优化，可以在一定程度上应对不平衡数据集带来的问题。

但需指出，该算法还存在一定的局限性。一个缺陷预测模型的构建，除了需要优化训练算法，对于数据的预处理也同样重要。在未来的工作中，需要在辅助数据的筛选和度量元的选择方法上继续深入研究，以进一步提升预测模型的性能。

参考文献

[1] TURHAN B, MENZIES T, BENER A B, et al. On the relative value of cross-company and within-company data for defect prediction[J]. Empirical Software Engineering, 2009, 14(5): 540-578.

[2] DAI Wenyuan, YANG Qiang, XUE Guirong, et al. Boosting for transfer learning[C]// Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007). Corvallis, Oregon, USA: ACM, 2007: 193-200.

[3] PAN S J, YANG Qiang. A survey on transfer learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(10): 1345-1359.

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

[5] FAWCETT T. An introduction to ROC analysis[J]. Pattern Recognition Letters, 2006, 27(8): 861-874.

[6] SHIPPEY T, HALL T, COUNSELL S, et al. So you need more method level datasets for your software defect prediction?: Voila![C]// the 10th ACM/IEEE International Symposium. Ciudad Real, Spain: ACM, 2016: 12.

[7] FREUND Y, SCHAPIRE R E. A decision-theoretic generalization of on-line learning and an application to Boosting[M]//VITÁNYI P. Computational learning theory. EuroCOLT 1995. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence). Berlin/Heidelberg: Springer. 1995: 23-37.

[8] JOSHI M V, KUMAR V, AGARWAL R C. Evaluating boosting algorithms to classify rare cases: Comparison and improvements[C]// Proceedings of the 1st IEEE International Conference on Data Mining. Washington DC: IEEE, 2001: 257-264.