基于 Boosting 的代价敏感软件缺陷预测方法

杨 杰 燕雪峰 张德平
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 Boosting 重抽样是常用的扩充小样本数据集的方法,首先针对抽样过程中存在的维数灾难现象,提出随机属性子集选择方法以进行降维处理;进而针对软件缺陷预测对于漏报与误报的惩罚因子不同的特点,在属性选择过程中添加代价敏感算法。以多个基本 k-NN 预测器为弱学习器,以代价最小为属性删除原则,得到当前抽样集的 k 值与属性子集的预测器集合,采用代价敏感的权重更新机制对抽样过程中的不同数据实例赋予相应权值,由所有预测器集合构成自适应的集成 k-NN 强学习器并建立软件缺陷预测模型。基于 NASA 数据集的实验结果表明,在小样本情况下,基于 Boosting 的代价敏感软件缺陷预测方法预测的漏报率有较大程度降低,误报率有一定程度增加,整体性能优于原来的 Boosting 集成预测方法。

关键词 软件缺陷预测,Boosting,代价敏感,随机属性选择,集成 k-NN
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.08.031

Cost-sensitive Software Defect Prediction Method Based on Boosting

YANG Jie YAN Xue-feng ZHANG De-ping
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract Boosting resampling is a common method to expand data sets for small samples. Firstly, aiming at dimension disaster phenomenon during resampling process, a randomly feature selection method is used to reduce the dimensions. In addition, considering the characteristic that software defect prediction's penalties for missing of true positives and the wrongly reported of negatives are different, cost-sensitive algorithm is added in feature selection process. On the basis of multi-normal k-NN weak learning, taking minimum costs as the principle, preditor which consists of k value and attributes subset of the current sampling set is get, cost-sensitive theory is imported to update weight vector during Boosting resampling process, and different instances are given corresponding weights. An adaptive ensemble k-NN learning is constructed using all the predictors, and a software defect prediction model is established. The results using NASA's data sets show that under the condition of small samples, with this model, missing of true positive rate reduces largely and the wrongly reported of negative rate increases to some extent. On the whole, compared with the origen boosting-based learning, the method of cost-sensitive software defect prediction based on boosting greatly improves the prediction effect.

Keywords Software defect prediction, Boosting, Cost-sensitive, Randomly feature selection, Ensemble k-NN

1 引言

随着软件技术的发展与软件产业的应用,软件的可信性受到普遍关注,软件缺陷预测技术是可信性研究领域中的热点问题之一[1]。合理预测软件缺陷可以有效地帮助测试者快速定位并弥补软件缺陷,从而达到显著减少软件开发成本和提高软件可信性的效果。

一般而言,软件缺陷预测技术可分为静态和动态两种类型[2],静态预测主要是指根据缺陷相关的度量数据对缺陷的数量或分布进行预测;而动态预测则是基于缺陷或者失效产生的时间对系统缺陷随时间的分布进行预测[3]。机器学习是静态预测技术领域中常用的缺陷预测方法,经典的学习方法包括朴素贝叶斯(Naive Bayesian, NB)[4]、支持向量机(Support Vector Machine, SVM)[5]、决策树、BP 神经网络、随机森林等;近年来,随着软件工程在诸多领域中的应用,建立在上述学习算法之上的跨项目软件缺陷预测成为了研究热点,基于迁移学习的软件缺陷预测方法[6]也因此被提出。由于大部分经典学习算法对于被测数据有严格的限制,将其应用于软件缺陷预测取得的效果并不理想,因此国内外学者对其进行了一系列的改进,如文献[7]利用条件概率查找出属性间的依

到稿日期:2016-07-27 返修日期:2016-10-21 本文受十三五重点基础科研项目(JCKY2016206B001),江苏省六大人才高峰项目(XXRJ-004),软件新技术与产业化协同创新中心资助。

杨 杰(1990—),男,硕士生,主要研究方向为系统建模与仿真,E-mail:604957608@qq.com;燕雪峰(1975—),男,博士,教授,主要研究方向为软件工程方法论、系统建模与仿真等;张德平(1973—),男,博士,讲师,主要研究方向为软件测试与软件可靠性建模。

赖关系，进而在朴素贝叶斯算法中添加描述属性间关联关系的边作为参数，提出了无需假设各属性之间满足独立性条件的改进算法，弥补了原有算法要求属性之间必须满足独立性条件的不足；文献[8]提出先对训练集进行修剪，再通过判断每个样本与其最近邻训练样本所属类别是否相同来决定对其取舍，改善了 SVM 的泛化能力；文献[9]通过判断增加或减少训练样本对系数和支持向量的影响程度提出了 SVM 增量训练算法，解决了原有 SVM 算法只能求解出近似解的不足。文献[7-9]所用方法在一定程度上提高了预测的准确程度，但由于并未综合考虑到小样本、类不平衡性以及代价不等的因素，因此其与实际需求并不完全相符。文献[10-11]分别使用了基于单源与多源的迁移学习方法，其将以上经典或改进后的学习方法作为弱学习器，并选取一定的度量指标，通过将预测结果从一个或多个源领域应用于另一目标领域的方式来实现；文献较好地考虑到了代价敏感以及类不平衡问题，但并未对参与训练的属性集进行筛选，一方面，造成了较高的时间复杂度，从而影响了效率；另一方面，因存在冗余或无效属性，导致预测效果与真实结果之间仍存在一定差距。

针对软件模块的类不平衡的特点，常用的处理方法可归结为过抽样与欠抽样[12]；在小样本预测方面，Boosting 重抽样表现出较好的适应性[13]，此外 Boosting 对于平衡数据集也具有一定贡献；针对预测过程中所犯的漏报与误报代价不等的问题，代价敏感算法[14-15]可通过调整错误代价比例来满足测试者对于误报率和漏报率的特定要求。文献[16]以决策树作为弱学习器，提出了基于代价敏感分类的集成软件缺陷预测模型构建方法，并给出了代价因子选择方法，但其判断待预测模块是否有缺陷时未能充分利用训练过程中产生的大量分类值，且不具有自适应特性。何亮等人[17]通过 Boosting 迭代抽样生成多个不同 k-NN 基本预测器，并分别对软件模块进行预测，最终将所有预测结果进行融合得出精确度较高的分类值的集成预测模型（Boosting k-NN，Bk-NN），但在属性选择时由于一次性删除属性过多而导致部分有效属性被删除，影响了最终的预测效果。

本文采用随机逐一删除属性的子集选择方法，以防止删除有价值的属性。同时，考虑到软件缺陷预测对于漏报与误报错误代价的敏感程度不同，将代价敏感算法引入到属性选择与 Boosting 权值更新的过程中，设定漏报的代价高于误报代价，继续进行子集选择的依据是删除已选定的属性后，总代价不会增大。更新权值时提高预测代价大的实例权重，以保证对其进行多次学习。综合以上思路，提出基于 Boosting 的代价敏感软件缺陷预测方法（Cost-Sensitive Boosting k-NN，CSBk-NN）。

2　基于 Boosting 的代价敏感软件缺陷预测方法

在软件缺陷预测领域，存在对于误报与漏报的惩罚因子不等的情况。实际上，较高的误报率会导致软件公司在原本没有错误的软件模块中投入过多的人力、物力、财力以及时间等，这会导致有限资源的浪费；而较高的漏报率则意味着软件中隐藏着较多的缺陷尚未被解决便已进入下一步开发工作或者已经交付至用户手中[18]。由于软件开发过程中对于缺陷的修复代价呈指数级增长，在交付至用户后甚至会对人身安全构成威胁，因此软件预测过程中对漏报的惩罚应该大于对误报的惩罚。

2.1　基于代价敏感的逐一删除属性方法

Bootstrap 抽样的过程中，在对属性子集进行选择时，若误将有效属性删除，将致使属性选择过程过早停止，进而影响预测的准确率。对于含有 m 个属性的实例，其子属性的组合方式共有 2^m 种，即子集数目关于属性数目呈指数级增长，从所有的属性子集中寻找最优的属性子集会带来维数灾难。通过大量计算各属性子集对于最终预测结果的效果并从中选取最优组合的方式并不可行，因此提出逐一删除属性的子集选择方法。判断是否继续进行属性选择的依据是随机删除某个属性后，其平均误差代价是否会减小或保持不变。若令漏报代价为 C_L，误报代价为 C_E，实例数目为 n，记平均误差代价为 $costMean$，即：

$$costMean=average\{cost(1),cost(2),\cdots,cost(i),\cdots,cost(n)\} \tag{1}$$

其中，$cost(i)$ 表示第 i 个实例的预测误差代价，其计算方式为：

$$cost(i)=\begin{cases}\mathrm{abs}(y_i-\hat{y}_i)\times C_L, & y_i-\hat{y}_i>0\\ \mathrm{abs}(y_i-\hat{y}_i)\times C_E, & y_i-\hat{y}_i<0\\ \mathrm{abs}(y_i-\hat{y}_i)\times 1, & y_i-\hat{y}_i=0\end{cases} \tag{2}$$

其中，$i=1,2,3,\cdots,n$。

记采用当前的属性集 $uParent$ 进行预测时得出的平均误差代价为 $costParient$，进而从 $uParent$ 中随机删除一个属性，若用剩余属性 $uChild$ 进行预测得到的平均误差代价 $costChild$ 没有增大，即 $costChild\leqslant costParient$，则将剩余属性用作属性全集并继续进行属性子集的选择和误差代价的计算，直至平均误差代价增大。此时跳出属性选择过程，返回当前的属性集。使用逐一删除属性的方式，其每次的抽样过程中属性选择部分的时间复杂度为 $O(M)$。

Boosting 被认为是一种提升任一给定弱学习器算法精度的通用方法，其独立于特定的弱学习器，经典 Boosting 算法中的权重更新机制依据预测误差的大小，误差大的权值较大，误差小的权值较小，这种更新机制可以保证对较难学习的数据集进行多次训练从而提高预测精度，但是却并未考虑到软件缺陷预测领域漏报与误报代价不同的特点，本文采用在权重更新时以预测代价为依据的权重更新机制。

2.2　基于代价敏感的权重更新方法

引入代价敏感后，在权重更新时可以依据如下步骤进行。

计算当前预测器 h_t 来预测 D 中各实例的最大误差代价。

$$costMax=\max\{cost(1),cost(2),\cdots,cost(i),\cdots,cost(n)\} \tag{3}$$

其中，$cost(i)$ 的计算方式与式(2)相同。

式(2)根据预测误差选择乘以不同系数来计算误差代价，当预测值小于真实值时，即存在将有缺陷模块预测为无缺陷模块的风险时，乘以代价 C_L，反之则意味着有可能将无缺陷模块预测为有缺陷模块，需乘以 C_E；当预测值与真实值相等

时，乘以系数 1 即可。另外，对于判断条件的选择，还可以根据实际需求进行调整，通过此方式放宽代价敏感的条件。如式(4)所示：

$$cost(i)=\begin{cases}\text{abs}(y_i-\hat{y}_i)\times C_L, & y_i-\hat{y}_i>0.05\\ \text{abs}(y_i-\hat{y}_i)\times C_E, & \hat{y}_i-y_i>0.05\\ \text{abs}(y_i-\hat{y}_i)\times 1, & \text{else}\end{cases} \tag{4}$$

其中，$i=1,2,3,\cdots,n$。

计算$(x,y)\in D$的损失，将其映射至$[0,1]$区间：

$$L(i)=\frac{\text{cost}(i)}{\text{cost}Max},i=1,2,3,\cdots,n \tag{5}$$

进而依据原有的第 i 个实例的权重$w(i)$和代价损失$L(i)$求得平均加权损失$\bar{L}$：

$$\bar{L}=\sum_{i=1}^{n}w(i)L(i) \tag{6}$$

求出当前 h_t 的置信度β：

$$\beta=\frac{\bar{L}}{1-\bar{L}} \tag{7}$$

其中，β越小，则平均加权误差越小，依据 β 在进行 D 的实例权重的更新过程中引入代价敏感思想，则更新过程为：

$$v(i)=\begin{cases}w(i)\beta^{1-L(i)}\times C_L, & y_i-\hat{y}_i>0\\ w(i)\beta^{1-L(i)}\times C_E, & y_i-\hat{y}_i<0\\ w(i)\beta^{1-L(i)}\times 1, & y_i-\hat{y}_i=0\end{cases} \tag{8}$$

进行归一化处理得到更新后的权重向量 $w(i)$：

$$w(i)=\frac{v(i)}{\sum_{i=1}^{n}v(i)} \tag{9}$$

根据新的权重向量 $w(i)$抽样生成下一次的数据集 D_{t+1}。

2.3 阈值选取标准与 CSBk-NN 预测模型评价准则

集成预测的另一关键问题是阈值的选择，阈值的选择是为了使该预测模型达到自适应的学习功能而采用的处理方式。该方法从训练集各实例的集成预测结果中选出一个合适的值作为区分待预测模块是否为缺陷模块的阈值，集成预测结果小于该阈值的模块被标记为正常模块，否则被标记为缺陷模块。首先给出软件缺陷预测评价准则的召回率 pd、误报率 pf、平衡值 bal，定义二值分类的混淆矩阵如表 1 所列。

表 1 二值分类的混淆矩阵

	预测	预测
真值	TP	FN
真值	FP	TN

$$pd=\frac{TP}{TP+FN} \tag{10}$$

$$pf=\frac{FP}{FP+TN} \tag{11}$$

$$bal=1-\frac{\sqrt{(0-pf)^2+(1-pd)^2}}{\sqrt{2}} \tag{12}$$

由混淆矩阵表格可知，较高的 pd 一般会导致较高的 pf，因此需要选取 bal 作为二者之间的平衡值。阈值的选择以最大化 bal 为准则，对于模型的预测效果，使用 pd，pf 与 bal 作为评价准则以验证本文思路的可行性。

2.4 CSBk-NN 模型的构建

本文所提模型可分解为 3 部分：重抽样模块、阈值选择模块和待预测软件的缺陷预测模块。

重抽样模块首先在原始数据集 $D(|D|=n)$上使用 Bootstrap 方法等比例地抽取出数据集 $D_1(|D_1|=n)$，在 D_1 上使用 1-NN 预测器，采用 2.1 节的属性子集选择方法进行训练并求出使得预测误差代价 $costParient(1)$最小的属性子集，同理使用 2-NN 预测器进行训练，得出此时的预测误差代价 $costParient(2)$，直至计算得到 $costParient(kMax)$，其中 $kMax$ 是预设定的最大 k 值。选取 $costParient(1)$至 $costParient(kMax)$中的最小值所对应的 k 值和 $uParent(k)$作为基本预测器 h_1 的参数 k_1 和 u_1，即得到预测模型 $h_1(k_1,u_1)$。使用预测模型 h_1 对原始数据集进行验证，按照 2.2 节的思路对每一数据集权重进行更新并依据权重进行下一次的 Bootstrap 抽样，得出新的预测模型 $h_2(k_2,u_2)$，直至达到规定的抽样次数 T 为止，最终得到 T 个基本 k-NN 预测器集合 $h_1,h_2,\cdots,h_T$。以上步骤中对某一实例进行近邻选择时，由于 Bootstrap 为有放回的重复抽样，每次抽样过程中可能存在重复的数据实例$(x_i,y_i),i\in(1,n),y\in\{0,1\}$，此时选用 k-NN 得出的预测效果不具有泛化能力，因此近邻选择时搜索的范围应为除去当前数据行外的所有其他原始数据，即 $D-(x_i,y_i)$。

阈值选择模块中，确定阈值的过程即依次以 h_1 至 h_T 的所有预测器逐个对原始数据集实例进行预测，以 T 个预测器预测结果的中位数作为当前实例的集成预测结果。预测结束后得到 n 个集成预测结果。再依次以其中一个作为分类值，其余的预测结果中小于该分类值的被识别为正常模块，否则为缺陷模块，并通过对比真实值计算当前分类值的 pd，pf，bal。从 n 个结果中选取 bal 最大的分类值作为最终的阈值 $threshold$。

对于待预测的软件模块 x，使用预测器集合 $h_1,h_2\cdots h_T$，得出 T 个预测结果，以 T 个结果的中位数作为 x 的预测结果 $yPredict$，将 $yPredict$ 与 $threshold$ 进行对比，若 $yPredict<threshold$，则判定 x 为正常模块；若 $yPredict=threshold$，则 x 与阈值对应的模块的缺陷归类相同；否则判定 x 为缺陷模块。

针对本文使用的模型，给出基于代价敏感的随机属性选择方法求解 h_t 参数的算法，具体过程如算法 1 所示。

算法 1 基于代价敏感的随机属性选择方法求解 h_t 参数的算法

1. 输入：抽样集 D(t)，原始数据集 D，属性全集 uParent(k)，最大 k 值 kMax
2. 输出：h_t 的参数(代价最小时的 k 值 kValue 和属性子集 uValue)
3. for k=1:kMax
4. costParient(k)=KnnFunction(D(t),D,uParent(k),k);//使用 k-NN 基本预测器在抽样集 D(t)上求出属性全集所对应的平均最小误差代价
5. uChild(k)=DeleteOneAttributeRandomly(uParent(k));//使用随机删除某一个属性的子集选择方式对属性子集进行选择
6. costChild(k)=KnnFunction(D(t),D,uChild(k),k);//使用 k-NN基本预测器，在抽样集 D(t)上求出当前属性子集所对应的

平均最小误差代价

7. while costChild(k)<=costParient(k)//若使用属性子集求得的平均误差代价没有增大
8. costParient(k)=costChild(k);//用属性子集的平均误差代价替代属性全集代价
9. uParent(k)=uChild(k);//用属性子集替代属性全集
10. uChild(k)=DeleteOneAttributeRandomly(uParent(k));
11. costChild(k)=KnnFunction(D(t),D,uChild(k),k);
12. end while//当前 k 值下的属性子集选择过程结束
13. end for//当前 D(t)下的所有 k 值的属性子集选择过程结束
14. kValue=1,uValue=uParent(1),costV=costParient(1);//开始求解所有 k 值下的最小误差代价，初始化为 1-NN 的属性子集对应的代价最小
15. for k=2:kMax//从 2-NN 开始对比
16. if costParient(k)<costV//若当前 k 值下的预测代价比已有的代价小
17. kValue=k,uValue=uParent(k);//将最终输出的 k 值替换为当前值，属性集为当前 k 值下对应的属性子集
18. costV=costParient(k);//以当前 k 值下的代价更新已有代价
19. end if
20. end for//所有 k 值下的代价对比结束
21. return kValue and uValue for ht(kValue,uValue) //返回 D(t)抽样集最小代价误差对应的 k 值与属性子集，将其作为 h_t 的参数

在 Bootstrap 抽样过程中，添加代价敏感的权重更新机制后，其权重更新部分的算法如算法 2 所示。

算法 2　Bootstrap 抽样过程中的权重更新算法

1. 输入：原始数据集 D，h_t 的参数 kValue 与 uValue
2. 输出：更新后的权重矩阵 w
3. cost=KnnFunction(D,uValue,kValue);//使用 k-NN 基本预测器，以算法 1 所求得的 uValue 和 kValue 为参数，计算在原始数据集 D 上的误差代价矩阵
4. costMax=max(cost);//计算代价矩阵中的最大值
5. lost(i:n)=cost(i:n)/costMax; //将数据集 D 中的所有实例对应的代价归一化至[0,1]区间
6. meanLost=0;//初始化平均加权误差代价为 0
7. for row=1:n
8. meanLost=w(i)*lost(i)+meanLost;//根据所有实例本次的抽样权重以及归一化之后的代价更新平均加权误差代价
9. end for
10. believeLevel=meanLost/(1-meanLost);//根据平均加权误差计算置信度
11. varyW(i:n)=w(i:n)*(believeLevel^(1-lost(i:n)));//根据置信度与归一化之后的代价计算所有原始数据集 D 中所有实例对应的误差代价损失
12. varySumW=sum(varyW);//将所有误差代价损失求和
13. w(i:n)=varyW(i:n)/ varySumW;//将所有误差代价损失归一化，并使用归一化后的结果更新权重矩阵，用于下一次的抽样依据
14. return new weight vector w(1:n)//返回更新后的权重矩阵

3　实验仿真与结果分析

3.1　实验设置

本文实验部分所采用的数据为 NASA 公开的 cm1，pc1，kc1，pc3 数据集。以模块作为实例，表格中的“有效数”是指将原始数据集中存在的重复模块、不一致模块删除后剩余的模块数。实验过程中的参数根据经验值分别做如下设置：集成 k-NN 预测器的最大 k 值 k_{Max} 为 5，Bootstrap 抽样次数 T 为 10，漏报代价 C_L 为 4、误报代 C_E 价为 2。分别从以上 4 个数据集中随机抽取 25%，50%，75%，100%进行实验，以 pd，pf 和 bal 作为评价指标，采用 5 次五折交叉验证结果的平均值对比分析 CSBk-NN 方法与 Bk-NN 方法的预测效果。为增强对本文思想可行性的说服力，对比实验中同时添加了在软件缺陷预测领域预测效果较好的朴素贝叶斯方法(NB)。相关数据集的主要信息如表 2 所列。

表 2　实验过程中使用的 NASA 数据集

数据集	模块数	属性数	有效数	缺陷数	缺陷比/%
cm1	344	37	327	42	12.84
pc1	759	37	705	61	8.65
kc1	1183	22	1183	314	26.54
pc3	1125	37	1077	134	12.44

3.2　实验数据记录

按照 3.1 节所设置的方式进行实验，实验结果如表 3 所列。

表 3　抽取不同比例数据集的实验结果

数据集	度量	25%			50%			75%			100%		
		CSBk-NN	Bk-NN	NB	CSBk-NN	Bk-NN	NB	CSBk-NN	Bk-NN	NB	CSBk-NN	Bk-NN	NB
cm1	pd	0.520	0.327	0.770	0.621	0.520	0.473	0.680	0.467	0.355	0.667	0.494	0.386
	pf	0.325	0.224	0.701	0.233	0.206	0.163	0.336	0.194	0.116	0.320	0.227	0.117
	bal	0.590	0.454	0.425	0.642	0.630	0.582	0.672	0.599	0.536	0.674	0.609	0.559
pc1	pd	0.424	0.380	0.336	0.579	0.374	0.329	0.623	0.425	0.270	0.657	0.558	0.369
	pf	0.176	0.219	0.094	0.229	0.063	0.093	0.197	0.084	0.076	0.172	0.073	0.068
	bal	0.570	0.509	0.514	0.654	0.553	0.519	0.676	0.575	0.480	0.703	0.680	0.551
kc1	pd	0.602	0.488	0.322	0.715	0.541	0.313	0.691	0.575	0.302	0.653	0.564	0.320
	pf	0.486	0.337	0.153	0.559	0.336	0.098	0.452	0.310	0.127	0.401	0.322	0.129
	bal	0.556	0.559	0.508	0.556	0.592	0.509	0.612	0.623	0.499	0.625	0.612	0.510
pc3	pd	0.612	0.472	0.491	0.640	0.477	0.879	0.681	0.500	0.631	0.635	0.518	0.913
	pf	0.240	0.120	0.198	0.271	0.143	0.630	0.276	0.150	0.242	0.269	0.137	0.702
	bal	0.657	0.612	0.590	0.673	0.612	0.533	0.700	0.624	0.673	0.669	0.644	0.496

如表 3 所列，所有数据都为 5 次五折交叉验证结果的平均值。根据每一数据集按照 3 种预测方式得出的 4 次抽样结果分别绘制 pd 与 bal 图，如图 1－图 8 所示。

综合图 1－图 8 可得出结论：整体上，使用 CSBk-NN 方法的 bal 值与原有 Bk-NN 方法相比略有提高或基本持平，并且 CSBk-NN 方法有更高的 pd 值，这两者的性能都优于 NB 方法。

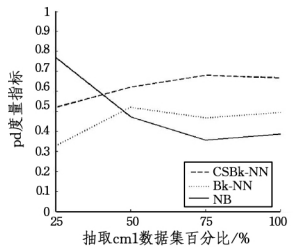

图 1 cm1 数据集的 pd 度量值

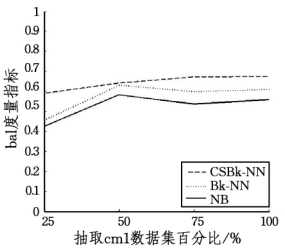

图 2 cm1 数据集的 bal 度量值

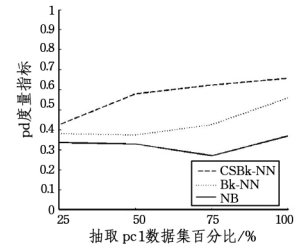

图 3 pc1 数据集的 pd 度量值

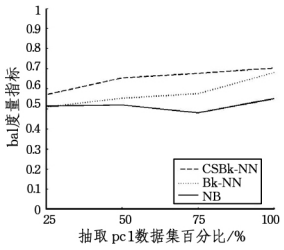

图 4 pc1 数据集的 bal 度量值

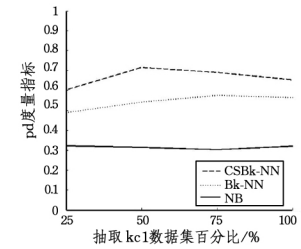

图 5 kc1 数据集的 pd 度量值

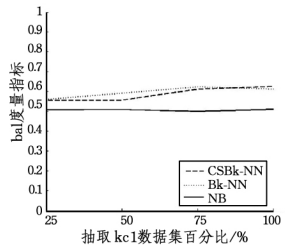

图 6 kc1 数据集的 bal 度量值

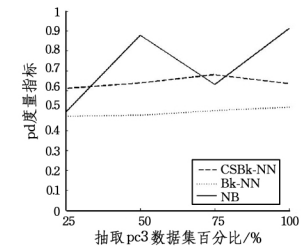

图 7 pc3 数据集的 pd 度量值

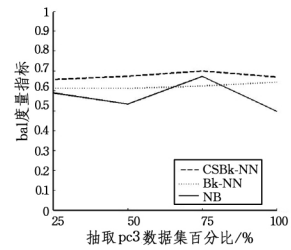

图 8 pc3 数据集的 bal 度量值

另一方面，使用 NB 方法预测得到的结果的稳定性最差，即其易受数据集与抽样集的影响。具体分析后发现使用 CSBk-NN 方法的预测结果比原有的 Bk-NN 方法在 pd 上平均有 0.145 的提高，在抽取 25%，50%，75%样本的情况下，分别平均提高了 0.124，0.161，0.177；在抽取 25%～75%数据的过程中，CSBk-NN 方法的增幅也最为明显。由于 CSBk-NN 方法与 Bk-NN 相比在误报率 pf 上也有一定幅度增加，因此最终其在 bal 值上的预测结果的平均值基本保持不变或略有增加。使用 NB 方法抽取 25%的数据时，在 cm1 数据集上得出的 pd 值为0.770，pf 为 0.701，bal 仅为 0.425；抽样 50%和 100%的数据时，在 pc3 数据集上的 pd 分别为 0.879 和 0.913，然而 bal 却仅为 0.533 和 0.496，这也是由于两者的 pf 过高而导致。从该实验结果可知，基于以上 4 个数据集的预测方法中 CSBk-NN 方法最优，Bk-NN 次之，NB 算法最差，可见 CSBk-NN 方法比 Bk-NN 和 NB 方法更适合于小样本的软件缺陷预测。

结束语 本文以 k-NN 分类器为弱学习器，在应用 Boosting 技术进行重抽样扩充数据集的过程中，使用了基于代价敏感的随机删除一个属性的子集选择方法，同时引入了代价敏感的权重更新方法，通过增加代价较大的数据样本权重保证其在下次抽样时进行再学习。通过 CSBk-NN，Bk-NN 与 NB 方法的结果对比可以印证本文思想的可行性。

本文也有所不足：实验过程中对于漏报代价与误报代价选取定值，未考虑到灵活性问题；最大 k 值使用固定值 5，Boosting 抽样次数根据经验数据选取 $T=10$，未考虑其他值的选取；所选用的数据集以及数据集的预处理方式也会对预测效果产生一定影响，由于只进行了简单的预处理操作，导致整体预测效果略差，如何保证所选取的数据集的质量也是后期改进时需要考虑的问题。最后需要指出，使用 CSB-kNN 方式的 pf 稍高，因此会对预测资源造成浪费，不适合于对某些资源有限、漏报代价又不高的软件进行缺陷预测。

参 考 文 献

[1] LIU H，HAO K G. Cause Analysis Method of Software Defect [J]. Computer Science，2009，36(1)：242-243. (in Chinese)
刘海，郝克刚. 软件缺陷原因分析方法[J]. 计算机科学，2009，36(1)：242-243.

[2] WANG Q，WU S J，LI M S. Software Prediction[J]. Journal of Software，2008，19(7)：1565-1580. (in Chinese)
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.

[3] QIAO H. Research on Software Defect Prediction Techniques [D]. Zhengzhou：The PLA Information Engineering University，2013. (in Chinese)
乔辉. 软件缺陷预测技术研究[D]. 郑州：解放军信息工程大学，2013.

[4] JAMBET C，MOULY C. The Indifferent Naive Bayes Classifier [C]// Sixteenth International Florida Artificial Intelligence Research Society Conference，2003. St. Augustine，Florida，USA，2003：341-345.

[5] VAPNIK V，GOLOWICH S E，SMOLA A. Support Vector Method for Function Approximation，Regression Estimation，and Signal Processing[J]. Advances in Neural Information Processing Systems，1970，9：281-287.

[6] ZHUANG F Z，LUO P，HE Q，et al. Survey on Transfer Learning Research[J]. Journal of Software，2015，26(1)：26-39. (in Chinese)
庄福振，罗平，何清，等. 迁移学习研究进展[J]. 软件学报，2015，26(1)：26-39.

[7] TAHERI S，MAMMADOV M，BAGIROV A M. Improving Naive Bayes classifier using conditional probabilities[C]// Australasian Data Mining Conference. 2011：63-68.

[8] LI H L，WANG C H，YUAN B Z. An Improved SVM：NN-SVM [J]. Chinese Journal of Computers，2003，26(8)：1015-1020. (in Chinese)
李红莲，王春花，袁保宗. 一种改进的支持向量机 NN-SVM[J]. 计算机学报，2003，26(8)：1015-1020.

(下转第 206 页)

tion,1999,1(5):473-479.

[8] MELCHIOR N A,SIMMONS R. Particle RRT for Path Planning with Uncertainty[C]// IEEE International Conference on Robotics & Automation. 2007:1617-1624.

[9] LINDEMANN S R,LAVALLE S M. Incrementally reducing dispersion by increasing voronoi bias in rrts[J]. IEEE International Conference on Robotics & Automation,2004,4(4):3251-3257.

[10] KARAMAN S,FRAZZOLI E. Incremental sampling-based algorithms for optimal motion planning[J]. International Journal of Robotics Research,2010,30(7):2011.

[11] QURESHI A H,MUMTAZ S,IQBAL K F,et al. Triangular geometry based optimal motion planning using RRT*-motion planner[C]// IEEE International Workshop on Advanced Motion Control. 2014:380-385.

[12] KUFFNER J J,LAVALLE S M. RRT-connect:An efficient approach to single-query path planning[C]// IEEE International Conference on Robotics and Automation. 2000:995-1001.

[13] JORDAN M,PEREZ A. Optimal Bidirectional Rapidly-Exploring Random Trees:MIT-CSAIL-TR-2013-021[R]. 2013.

[14] LAVALLE S M,KUFFNER J J. Rapidly-Exploring Random Trees:Progress and Prospects[J]. Algorithmic & Computational Robotics New Directions,2000:293-308.

[15] KARAMAN S,FRAZZOLI E. Sampling-based algorithms for optimal motion planning[J]. International Journal of Robotics Research,2011,30(7):846-894.

[16] QURESHI A H. AYAZ Y. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments[J]. Robotics & Autonomous Systems,2015,68:1-11.

[17] CARSTEN J,FERGUSON D,STENTZ A. 3D field D*:Improved path planning and replanning in three dimensions[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems. 2006:3381-3386.

[18] RAJA R,DUTTA A,VENKATESH K S. New potential field method for rough terrain path planning using genetic algorithm-for a 6-wheel rover[J]. Robotics and Autonomous Systems,2015,72(C):295-306.

[19] KARAMAN S,FRAZZOLI E. Sampling-based optimal motion planning for non-holonomic dynamical systems[C]// IEEE International Conference on Robotics and Automation. IEEE,2013:5041-5047.

[20] ARYA S,MOUNT D M,SILVERMAN R,et al. An optimal algorithm for approximate nearest neighbor search in fixeddimensions[J]. Journal of the ACM,1999,45(6):891-923.

(上接第 180 页)

[9] CAUWENBERGHS G,POGGIO T. Incremental and Decremental Support Vector Machine Learning[M]// Advances in Neural Information Processing Systems 13. 2010:409-415.

[10] WU F J. Understanding Knowledge Sharing Activities in Software Fault-prone Prediction:a Transfer Learning Study[J]. Journal of Chinese Computer Systems,2014,35(11):2416-2421. (in Chinese)

吴方君. 软件缺陷预测经验共享:一种迁移学习方法[J]. 小型微型计算机系统,2014,35(11):2416-2421.

[11] ZHANG Q,LI M,WANG X S,et al. Instance-based Transfer Learning for Multi-source Domains[J]. Acta Automatica Sinica,2014,40(6):1176-1183. (in Chinese)

张倩,李明,王雪松,等. 一种面向多源领域的实例迁移学习[J]. 自动化学报,2014,40(6):1176-1183.

[12] CHAWLA N V,BOWYER K W,HALL L O,et al. SMOTE:synthetic minority over-sampling technique[J]. Journal of Artificial Intelligence Research,2011,16(1):321-357.

[13] RICHELLI A,COMENSOLI S,KOVACS-VAJNA Z M. A DC/DC Boosting Technique and Power Management for Ultralow-Voltage Energy Harvesting Applications[J]. IEEE Transactions on Industrial Electronics,2012,59(6):2701-2708.

[14] ZHENG J. Cost-sensitive boosting neural networks for software defect prediction[J]. Expert Systems with Applications,2010,37(6):4537-4543.

[15] LI Y,HUANG Z Q,FANG B W,et al. Using Cost-Sensitive Classification for Software Defects Prediction[J]. Journal of Frontiers of Computer Science and Technology,2014,8(12):1442-1451. (in Chinese)

李勇,黄志球,房丙午,等. 代价敏感分类的软件缺陷预测方法[J]. 计算机科学与探索,2014,8(12):1442-1451.

[16] MIAO L S. Software Defect Prediction Based on Cost-Sensitive Neural Networks[J]. Electronic Science and Technology,2012,25(6):75-78. (in Chinese)

缪林松. 基于代价敏感神经网络算法的软件缺陷预测[J]. 电子科技,2012,25(6):75-78.

[17] HE L,SONG Q B,SHEN J Y. Boosting-Based k-NN Learning for Software Defect Prediction[J]. Pattern Recognition and Artificial Intelligence,2012,25(5):792-802. (in Chinese)

何亮,宋擒豹,沈钧毅. 基于 Boosting 的集成 k-NN 软件缺陷预测方法[J]. 模式识别与人工智能,2012,25(5):792-802.

[18] CHEN X,GU Q,LIU W S,et al. Survey of Static Software Defect Prediction[J]. Journal of Software,2016,27(1):1-25. (in Chinese)

陈翔,顾庆,刘望舒,等. 静态软件缺陷预测方法研究[J]. 软件学报,2016,27(1):1-25.