基于多核集成学习的跨项目软件缺陷预测

黄 琳,荆晓远,董西伟
(南京邮电大学 自动化学院,江苏 南京 210003)

摘 要:软件缺陷预测的目的是通过历史缺陷数据预测新软件模块的缺陷倾向性,从而提高软件系统的质量。软件的缺陷模块存在结构复杂和类别分布不平衡的问题,并且历史数据是有限的。针对这些问题,提出了一种多核集成学习的跨项目软件缺陷预测方法。跨项目软件缺陷预测是解决项目初期缺陷预测缺乏数据集的有效途径。多核学习方法能够将不同特性的核函数进行组合,使数据在新的特征空间中得到更好的表达,提高预测精度。集成学习方法能够解决类别分布不平衡问题。考虑到在软件缺陷预测中将有缺陷模块预测为无缺陷模块的风险远远大于将无缺陷模块预测为有缺陷模块,在计算误差时引入了代价敏感矩阵。使用 NASA 和 AEEEM 这两个数据库来评估所有比较方法的性能,实验结果表明,提出的算法能够达到很好的效果。

关键词:跨项目缺陷预测;多核学习;集成学习;代价敏感学习;有监督学习
中图分类号:TP181 **文献标识码**:A **文章编号**:1673-629X(2019)06-0027-05
doi:10.3969/j.issn.1673-629X.2019.06.006

Cross-project Software Defect Prediction Based on Multiple Kernel Ensemble Learning

HUANG Lin,JING Xiao-yuan,DONG Xi-wei
(School of Automation,Nanjing University of Posts and Telecommunications,Nanjing 210003,China)

Abstract:Software defect prediction aims to predict the defect proneness of new software modules with the historical defect data so as to improve the quality of software system. The defect modules of software have complex structure and unbalanced category distribution with limited historical data. In order to solve these problems,we propose a cross-project software defect prediction method based on multiple kernel ensemble learning. Cross-project software defect prediction is an effective way to solve the lack of datasets in the initial project defect prediction. Multiple kernel learning can combine kernel functions with different characteristics to make the data better expressed in the new feature space and improve the prediction accuracy. Ensemble learning can solve the problem of category distribution imbalance. Considering that the risk of predicting a defective module as a defect-free module in software defect prediction is far greater than predicting a defect-free module as a defective module,a cost-sensitive matrix is introduced in the calculation of the error. The NASA and AEEEM datasets as test data are used to evaluate the performance of all comparison methods. The experiment shows that the proposed algorithm is efficient.

Key words: cross-project software defect prediction; multiple kernel learning; ensemble learning; cost-sensitive learning; supervised learning

0 引 言

软件缺陷的产生往往是由于开发人员需求理解不正确或是经验不足。而含有缺陷的软件在运行时可能会产生意料之外的结果,严重的时候会给企业造成巨大的经济损失,甚至会威胁到人们的生命安全。软件缺陷预测(software defect prediction,SDP)是软件工程中最重要的研究课题之一[1],吸引了学术界和工业界的广泛关注。基于度量的静态软件缺陷预测技术借助于从已有的软件模块中获得历史数据,对新的软件模块进行缺陷预测,来判断它们是否存在缺陷,从而为软

收稿日期:2018-07-24 **修回日期**:2018-11-27 **网络出版时间**:2019-03-06
基金项目:国家自然科学基金(61702280);国家自然科学基金重点项目通用技术基础研究联合基金(U1736211);江苏省自然科学基金(BK20170900);江苏省高等学校自然研究项目(17KJB520025);南京邮电大学引进人才科研启动基金(NY217009)
作者简介:黄 琳(1993-),女,硕士研究生,研究方向为模式识别、软件缺陷预测等;荆晓远,教授,研究方向为机器学习、软件缺陷预测、深度学习等。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20190306.0952.076.html

件项目提供决策支持[2-4]。近年来,支持向量机(support vector machine,SVM)[5]、决策树[6]、朴素贝叶斯[7-8]、代价敏感学习[9]和集成学习[10-11]这些机器学习技术已经被广泛地用于软件缺陷预测领域。最近,字典学习[12]、稀疏表示[13]等方法也被引入了软件缺陷预测中。软件缺陷预测的历史数据具有样本不足、结构复杂和类别分布不平衡的特点。为了解决这些问题,文中提出一种基于多核集成学习的跨项目软件缺陷预测算法(cross-project software defect prediction based on multiple kernel ensemble learning,CMKEL)。

1 相关工作

1.1 跨项目缺陷预测

软件缺陷预测是当前大数据技术在软件工程领域的重点研究方向之一[14]。在早期大多数的研究都是使用项目内已经标记的程序模块,一部分样本来构建预测模型,另一部分样本作为预测。然而在实际应用中项目内的历史数据是有限的。由于历史数据的缺少,研究者们开始关注跨项目软件缺陷预测的问题,跨项目就是使用其他项目的训练数据来构建预测模型,并对一个全新项目进行缺陷预测。

1.2 多核学习技术

多核学习就是将不同特性的核函数进行组合,获得多类核函数的优点,来达到更优的映射性能。和传统的单核方法相比,多核学习不仅能够组合利用各基本的单核函数的特征映射的能力,而且还能将不同特性的核函数进行组合,来获得多类单核数的优点,从而得到更优的映射能力,提高预测的精度。

1.3 集成学习技术

集成学习就是用多个弱分类器结合为一个强分类器,从而提升分类方法的效果。集成学习能够得到比基分类器更好的分类效果和泛化能力,不容易出现过拟合情况,因此适用于跨项目缺陷预测问题。

2 基于多核集成学习的跨项目软件缺陷预测

对于静态软件缺陷模块存在的结构复杂、类别不平衡、历史数据缺少的问题,文中提出一种基于多核集成学习的跨项目软件缺陷预测方法。

假设训练的软件模块 $D=\{(x_i,y_i),i=1,2,\cdots,N\}$,其中 x_i 是模块属性的向量,$y_i\in\{0,1\}$ 是模块的标签,N 是项目数。$K=\{k_j:X\times X\to R,j=1,2,\cdots,M\}$ 是 M 个核函数的集合,其中 X 是输入空间。CMKEL 主要是学习基于多核的一个分类器 $f(x)$,这个分类器是通过有标记的历史数据进行多核训练得到的,通过基于多核的分类器 $f(x)$ 来预测新的软件模块,基于多核分类器表示为:

$$f(x)=\sum_{t=1}^{T}\alpha_t f_t(x) \tag{1}$$

其中,T 为整个 boosting 过程的次数;$f_t(x)$ 为第 t 次 $(1\leqslant t\leqslant T)$ boosting 过程得到的弱分类器;α_t 为相应的权重值。

该算法的关键点是每次在 M 个基于核函数的分类器中进行 boosting 过程后,学习得到一个错误分类误差最小的 $f_t(x)$ 分类器及其组合权重 α_t。经过 T 次 boosting 过程之后,将这些分类器按照权重值集成,得到最终的分类器。

2.1 多核学习分类器

多核学习就是将不同特性的核函数进行组合,获得多类核函数的优点,以得到更优的映射性能。多核学习中的合成核是多个核函数的凸组合,可以表示为多个核函数的加权和,其形如 $K=\sum_{j=1}^{M}\beta_j k_j,\beta_j\geqslant 0$,$\sum_{j=1}^{M}\beta_j=1$,其中 k_j 是一个基本核函数,M 是基本核的总个数,β_j 是权系数。在多核框架下,将样本在特征空间中的表示问题转化为基本核与权系数的选择的问题,可以表示为式 2 所示的优化问题:

$$\min_{\boldsymbol{\theta}\in\Delta}\min_{f\in H_{k(\theta)}}\frac{1}{2}\|f\|^2_{H_{k(\theta)}}+C\sum_{i=1}^{N}l(f(x_i),y_i) \tag{2}$$

其中,$\boldsymbol{\theta}=(\theta_1,\theta_2,\cdots,\theta_M)$,$\Delta=\{\boldsymbol{\theta}\in R_+^M\mid \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{e}_M=1\}$,$K(\theta)=\sum_{j=1}^{M}\theta_j k_j$,$l(f(x_i),y_i)=\max(0,1-y_i f(x_i))$,$\boldsymbol{e}_M$ 是 M 维向量,它的元素全部是 1。

多核学习的目标是通过最优化方法来求取合成核的参数,将上式转化为如下的最优化形式:

$$\min_{\boldsymbol{\theta}\in\Delta}\max_{\boldsymbol{\alpha}\in\Xi}\{\boldsymbol{\alpha}_T\boldsymbol{e}_M-\frac{1}{2}(\boldsymbol{\alpha}\circ\boldsymbol{y})^T(\sum_{j=1}^{N}\theta_j,K^j)(\boldsymbol{\alpha}\circ\boldsymbol{y})^{\mathrm{T}}\} \tag{3}$$

其中,$K^j\in R^{N\times N}$,$K^j_{p,q}=k_j(x_p,x_q)$,$\Xi=\{\alpha\mid\alpha\in[0,C]^{\mathrm{T}}\}$,“∘”表示为两个向量的内积。从公式可以看到此方程的优化计算很复杂,为了避免这个问题可以采用 boosting 过程来计算多核学习的参数。

2.2 多核集成学习分类器

为了学习一个多核集成的分类器,文中采用一个典型且有效的 boosting 算法。具体而言,在初始化训练集之后通过一系列 boosting 过程,重复学习一些基于核的分类器,然后根据它们的组合权重进行集合,从而得到最终的 CMKEL 分类器。在 boosting 过程之前,需要先对训练集进行初始化。文中在整个训练集上直接执行随机抽样策略,然后将这些选择的样本作为 CMKEL 初始训练集。

在训练集初始化完成之后,需要有对应的 D_t 向量,D_t 表示每个样本在整个 boosting 过程的不同权重。最初,这些权重都是相同的。为了更加关注那些错误分类的样本,在每一次 boosting 过程增加错误分类的权重或是减少正确分类的权重,来调整权重向量 D_t。一旦得到训练集和权重向量 D_t,就可以进行 boosting 过程。每次 boosting 过程的关键是要从 M 个基于核函数的分类器中学习得到一个基于核假设的分类器 $f_t(x)$。

在已知核函数 k_t^j 的情况下要得到 $f_t(x)$,首先使用常规分类方法学习单核分类器 $f_t^j(x)$,文中用到的常规分类方法是 SVM。其次,求出该单核分类器 $f_t^j(x)$ 在整个训练数据中的错误分类误差。然而当目标被错分成其他类的代价有比较大的差异的时候,区分这些代价是有必要的,即使是被错分类,也是希望被错分到代价较小的类上,此处引入代价敏感分类的软件缺陷预测模型。

样本中无缺陷模块的标签为 0,有缺陷模块的标签为 1。软件缺陷预测模型对有缺陷的模块被预测为无缺陷时称为Ⅰ类错误,代价表示为 cost(1,0);反之Ⅱ类错误表示无缺陷模块被预测为有缺陷的,代价表示为 cost(0,1),显然Ⅰ类错误的代价敏感要远大于Ⅱ错误代价敏感。文中定义的代价矩阵如表 1 所示。

表 1　代价矩阵

True	Predictive	
	1	0
1	cost(1,1) = 0	cost(1,0) = μ
0	cost(0,1) = 1	cost(0,0) = 0

在表 1 中,μ 表示Ⅰ类错误的代价敏感系数。

考虑到代价矩阵后的每个基于核分类器的错误分类误差的计算为:

$$\varepsilon_t = \varepsilon(f_t^j(x)) = \sum_{i=1}^{N} \text{cost}(l,g) D_t(i) (f_t^j(x_i) \neq y_i) \tag{4}$$

其中,cost(l,g) 为代价矩阵,cost(l,g) 表示 l 类被错分成 g 类的代价。

最好的分类器是得到最小的错误分类误差,在第 t 次 boosting 过程之后得到的弱分类器为:

$$f_t(x) = \underset{f_t^j, j \in \{1,2,\cdots,M\}}{\text{argmin}} \varepsilon(f_t^j(x)) \tag{5}$$

在第 t 次的 boosting 过程中,$f_t(x)$ 分类器的权重 α_t 和错误分类误差的关系表达式为:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{6}$$

在得到权重 α_t 之后,第 t 次的 boosting 过程完成。在下一次 boosting 开始前,要根据第 t 次 boosting 过程的结果更新下一次 boosting 过程中的权重向量 D_{t+1}。权重向量的更新原则是根据分类结果进行更新,通过提高错分样本的权重,降低正确分类样本的权重。目的是在下一次 boosting 过程中将重点放在错误分类的样本上,训练样本集的权重向量更新表达式为:

$$D_{t+1,i} = \frac{D_{t,i}}{Z_t} \exp(-\alpha_t y_i f_t(x_i)), i = 1,2,\cdots,N \tag{7}$$

其中,Z_t 是规范化因子。

根据分类的结果,权重向量的更新策略详细表示为:

$$D_{t+1,i} = \begin{cases} \dfrac{D_{t,i}}{Z_i} exp(-\alpha_t), f_t(x_i) = y_i \\ \dfrac{D_{t,i}}{Z_i} exp(\alpha_t), f_t(x_i) \neq y_i \end{cases}, i = 1,2,\cdots,N \tag{8}$$

其中,$f_t(x_i) = y_i$ 表示被正确分类;$f_t(x_i) \neq y_i$ 表示被错误分类。

经过 T 次 boosting 过程后,最终的分类器为:

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right) \tag{9}$$

综上所述,CMKEL 算法的描述如下:

输入:训练集、核函数 k_j、初始权重分布 $D_1 = 1/N$。

(1)开始第 $t = 1 \sim T$ 次的 boosting 过程,权重向量表示每个训练样本 D_t 的权重分布。

(2)对于 $j = 1 \sim M$ 来训练对应 k_j 核函数的弱分类器,根据式 4 计算在 D_t 下的错误分类误差。

(3)训练完所有的基于核的分类器之后,选择最小错误分类的分类器,由式 5 得到 $f_t(x)$。

(4)由式 6 计算 α_t。

(5)根据式 8 权重更新的策略,更新下一次 boosting 过程的权重向量分布。

(6) T 次 boosting 过程之后,得到 T 组分类器和与之对应的权重。

输出:最终分类器 $G(x)$。

3　实　验

为验证 CMKEL 算法的预测效果,将该算法与加权朴素贝叶斯(weighted Naïve Bayes,NB)算法[7]、压缩的 C4.5 决策树(compressed C4.5 decision tree,CC4.5)算法[15]和代价敏感的 Boosting 神经网络(cost-sensitive boosting neural network,CBNN)算法[16]在 NASA、AEEEM 数据库中进行对比验证。

3.1　实验数据库

NASA、AEEEM 数据库的工程数都是五个,它们各个工程的静态代码度量和缺陷占比如表 2 所示。

3.2　实验指标

预测模型的评估指标有召回率(pd)、误报率

(pf)、查准率(precision)和精确度(acc)。

表 2 数据集的详细信息

数据库	项目	特征数	样本总数	缺陷样本数	缺陷样本占比/%
NASA	CM1	37	327	42	12.84
	MW1	37	253	27	10.67
	PC1	37	705	61	8.65
	PC3	37	1 077	134	12.44
	PC4	37	1 458	178	12.21
AEEEM	EQ	61	324	129	39.81
	JDT	61	997	206	20.66
	LC	61	691	64	9.26
	ML	61	1 842	245	13.16
	PDE	61	1 497	209	13.96

文中主要采用两个综合性能指标:F-measure,就是将 pd 与 precision 结合起来评价;AUC 值(area under curve)被定义为 ROC 曲线下的面积,使用 AUC 值可以评估二分类问题分类效果的优劣。F-measure 和 AUC 的数值越大,表示软件缺陷预测模型的预测性能越好。

3.3 实验设置

在多核学习的训练中,选择具有 20 个不同宽度(2^{-10},2^{-9},…,2^{9})的高斯核函数,然后使用这 20 个基核分别将 NASA 和 AEEEM 里面的每一个工程映射到一个新的特征空间。对于弱分类器 SVM,结合每个核函数训练得到一个基于核的分类器,整个过程采用流行的 LISVM 工具箱作为 SVM 求解器。为了验证代价敏感系数对模型是否有影响,设置 μ = 1,5,10,15,20,观察代价敏感系数不同时对实验的影响。Boosting 过程的初始设计如下:在同一个数据库下,随机选择一个工程作为训练样本,另一个工程作为测试样本,每组算法迭代 20 次,最后将 20 次跑出来的数据取平均。Boosting 的过程和文献[17]中一致,训练次数 T 为 100。

3.4 实验结果及分析

为了验证在 CMKEL 算法中代价敏感系数的大小对实验结果的影响,设置了不同的 μ 值,在 AEEEM 数据库上的实验结果如表 3 所示。其中 μ = 1 表示没有引入代价敏感系数。

表 3 不同代价敏感系数下的 AUC 值

dataset	source→target	μ = 1	5	10	15	20
AEEEM	ML→PDE	0.62	0.67	0.70	0.76	0.55
	JDT→LC	0.65	0.62	0.67	0.73	0.61
	PDE→JDT	0.59	0.67	0.67	0.71	0.56
	LC→EQ	0.65	0.67	0.69	0.64	0.56
	JDT→ML	0.67	0.66	0.66	0.70	0.48
	average	0.64	0.66	0.68	0.71	0.55

从表 3 中的实验结果可以看出:当 μ > 1 时,AUC 值要比 μ = 1 高,说明引入代价敏感系数提高了预测的效果;随着 μ 值的增大,AUC 的值也在增长,但是当 μ > 15 时,AUC 值开始下降,说明代价敏感系数并不是越大越好,当 μ = 15 时,CMKEL 算法能达到最好的效果。

为了验证文中算法是否有更好的性能,分别在 NASA 和 AEEEM 两个数据库与其他算法进行对比,结果如表 4 所示。(实验结果中将 F-measure 值表示成 F 值)

表 4 算法对比结果

source→target	指标	NB	CC4.5	CBBN	CMKEL
PC3→PC1	F	0.22	0.32	0.43	0.44
	AUC	0.62	0.65	0.68	0.79
PC4→PC3	F	0.30	0.26	0.38	0.40
	AUC	0.56	0.65	0.64	0.78
PC1→PC4	F	0.32	0.26	0.46	0.49
	AUC	0.67	0.58	0.68	0.79
CM1→MW1	F	0.32	0.29	0.38	0.54
	AUC	0.65	0.60	0.67	0.77
MW1→CM1	F	0.37	0.32	0.43	0.36
	AUC	0.70	0.61	0.70	0.74
ML→PDE	F	0.32	0.34	0.53	0.32
	AUC	0.64	0.62	0.69	0.64
JDT→LC	F	0.52	0.38	0.46	0.36
	AUC	0.63	0.65	0.66	0.78
PDE→JDT	F	0.57	0.37	0.56	0.48
	AUC	0.65	0.64	0.70	0.81
LC→EQ	F	0.42	0.29	0.48	0.37
	AUC	0.57	0.59	0.67	0.72
JDT→ML	F	0.36	0.37	0.42	0.46
	AUC	0.65	0.65	0.72	0.76

通过以上实验可以看出:NB,CC4.5,CBBN 算法在某些项目上面能够有比较好的 F-measure 值,但是 CMKEL 在大部分项目上都同时有很好的 F-measure 值、AUC 值,效果要比前三种算法好,表明了 CMKEL 算法的优越性。

4　结束语

为了解决软件缺陷预测中样本数据结构复杂与类别分布不平衡的问题,提出一种基于多核集成学习的跨项目软件缺陷预测算法。利用多核学习将数据映射到新的特征空间,更好地表示数据,提高预测的精确度。集成学习能够解决类别分布不平衡的问题。在 NASA 和 AEEEM 这两个数据库上的实验结果证明了该算法的有效性。

参考文献:

[1] 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.

[2] 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.

[3] 李　勇,黄志球,王　勇,等. 数据驱动的软件缺陷预测研究综述[J]. 电子学报,2017,45(4):982-988.

[4] 李　乔,郑　啸. 云计算研究现状综述[J]. 计算机科学,2011,38(4):32-37.

[5] 王　涛,李伟华,刘　尊,等. 基于支持向量机的软件缺陷预测模型[J]. 西北工业大学学报,2011,29(6):864-870.

[6] LI Biwen, SHEN Beijun, WANG Jun, et al. A scenario-based approach to predicting software defects using compressed C4.5 model[C]//IEEE 38th annual computer software and applications conference. Vasteras, Sweden: IEEE, 2014:406-415.

[7] WANG Tao, LI Weihua. Naive bayes software defect prediction model[C]//International conference on computational intelligence and software engineering. Wuhan, China: IEEE, 2010:1-4.

[8] OKUTAN A, YILDIZ O T. Software defect prediction using Bayesian networks[J]. Empirical Software Engineering, 2014,19(1):154-181.

[9] 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.

[10] 熊　婧,高　岩,王雅瑜. 基于 Adaboost 算法的软件缺陷预测模型[J]. 计算机科学,2016,43(7):186-190.

[11] SUN Zhongbin, SONG Qinbao, ZHU Xiaoyan. Using coding-based ensemble learning to improve software defect prediction[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6):1806-1817.

[12] JING Xiaoyuan, YING Shi, ZHANG Zhiwu, et al. Dictionary learning based software defect prediction[C]//Proceedings of the 36th international conference on software engineering. Hyderabad, India: ACM, 2014:414-423.

[13] ZHANG Zhiwu, JING Xiaoyuan, WANG Tiejian. Label propagation based semi-supervised learning for software defect prediction[J]. Automated Software Engineering, 2017, 24(1):47-69.

[14] JING Xiaoyuan, ZHANG Zhiwu, YING Shi, et al. Software defect prediction based on collaborative representation classification[C]//Companion proceedings of the 36th international conference on software engineering. Hyderabad, India: ACM, 2014:632-633.

[15] WANG Jun, SHEN Beijun, CHEN Yuting. Compressed C4.5 models for software defect prediction[C]//12th international conference on quality software. Xi'an, Shaanxi, China: IEEE, 2012:13-16.

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

[17] XIA Hao, HOI S C H. MKBoost: a framework of multiple kernel boosting[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(7):1574-1586.