CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 40 卷第 10 期 2018 年 10 月
Vol. 40, No. 10, Oct. 2018

文章编号:1007-130X(2018)10-1787-09

基于代价敏感支持向量机的软件缺陷预测研究*

任胜兵,廖湘荡
(中南大学软件学院,湖南 长沙 410075)

摘　要:软件缺陷预测是典型的非平衡学习问题。基于 CS-SVM 和聚类算法改进代价敏感支持向量机(SVM)算法,提出了 CCS-SVM 软件缺陷预测模型。在 CCS-SVM 预测模型中,将 SVM 与类别误分代价结合起来,以非平衡数据评价指标作为目标函数,优化错分代价因子,提升少数类样本的识别率。通过聚类找到每类样本的中心点,根据样本到其中心点的距离定义每个样本的类别置信度,给每个样本分配不同的误分代价系数,并把样本的置信度引入到代价敏感 SVM 优化问题中,提高算法鲁棒性,提升 SVM 分类性能。此外,为了提高模型的泛化能力,使用遗传算法优化特征选择和模型参数。通过美国航空航天局 NASA MDP 数据集实验表明,本文方法的 *G-mean* 和 *F-measure* 模型评价值有明显的提升。

关键词:软件缺陷预测;代价敏感;支持向量机;非平衡数据分类;参数选择;遗传算法
中图分类号:TP311.5　　**文献标志码**:A
doi:10.3969/j.issn.1007-130X.2018.10.010

Software defect prediction based on cost-sensitive support vector machine

REN Sheng-bing, LIAO Xiang-dang
(School of Software, Central South University, Changsha 410075, China)

Abstract: Software defect prediction is a typical unbalanced learning problem. We propose a CCS-SVM software defect prediction model based on cost sensitive SVM algorithm improved by the CS-SVM and clustering algorithm. In the CCS-SVM prediction model, we combine SVM and the cost of class misclassification, take unbalanced data evaluation index as the objective function, and optimize the misclassification cost factor so as to enhance the recognition rate of the minority class samples. We find the center point of each sample through clustering, define the class confidence for each sample according to the distance of the sample to its center point, assign different misclassification cost factors to different samples, and introduce the class confidence of each sample to the optimization problem of cost sensitive SVM, and improve the robustness of the algorithm and classification performance of SVM. To enhance the generalization ability of the model, we use the genetic algorithm to optimize feature selection and model parameters. Experimental results of the NASA Metric Data Program (MDP) dataset show that our method is significantly improved in the *G-mean* and *F-measure* value for model evaluation.

Key words: software defect prediction; cost sensitivity; support vector machine; unbanlanced data classification; parameter selection; genetic algorithm

* 收稿日期:2017-11-08;修回日期:2018-01-24
通信地址:410075 湖南省长沙市韶山南路 22 号中南大学铁道校区软件学院
Address: School of Software, Tiedao Campus, Central South University, 22 Shaoshan South Rd, Changsha 410075, Hunan, P. R. China

1 引言

软件缺陷是计算机软件或程序中存在的某种破坏正常运行能力的问题、错误,或者隐藏的功能缺陷。缺陷的存在会导致软件产品在某种程度上不能满足用户的需要,被认为是影响软件质量最主要的原因。在程序发布后修正软件开发阶段的所有缺陷,维护成本可能非常高。因此,在程序发布之前尽早检测出所有可能的错误非常重要[1,2]。

目前,有大量研究根据程序静态属性用机器学习来建立软件缺陷预测模型。为了得到高性能预测模型,主要研究工作集中在如何选择有效的机器学习算法。Menzies 等[3]用朴素贝叶斯 NB(Naive Bayesian)构造分类器预测软件缺陷,并利用信息增益对特征选择进行研究。Shuai 等[4]提出了 GA-CSSVM(Genetic Algorithm Cost Sensesitive Support Vector Machine)预测模型,使用遗传算法调整 CS-SVM(Cost-Sensitive SVM)的惩罚参数,实验表明该方法基于 AUC 评价函数取得了比较好的结果。Seliya 等[5]提出了"RBBag"预测模型,用朴素贝叶斯和 C4.5 结合数据抽样构建 bagging 预测模型,实验结果表明该方法比不建立 bagging 和数据抽样的模型性能更好。然而,这些研究在处理数据不平衡问题方面都存在一些问题。软件缺陷数据是典型的非平衡数据,因为大部分缺陷都集中在少量的模块。比如 NASA 的 PC1 项目包含 705 个模块,其中 61 个模块有缺陷,占比所有模块的 8.65%[6]。所以,虽然他们的实验结果表明这些算法能够得到不错的预测性能,但由于他们的实验主要是基于准确率来评价模型的,这在软件缺陷预测问题上是不合理的。假设有缺陷模块占比 10%,把所有的有缺陷模块都预测为无缺陷模块,准确率仍然是 90%!

Wang 等[7]比较了已有的软件缺陷预测方法和非平衡数据分类方法,包含欠采样和过采样技术、阈值移动、集成学习等算法,结果表明 AdaBoost.NC 优于朴素贝叶斯和随机森林 RF(Random Forest)等方法。最近,支持向量机在软件缺陷预测领域和非平衡学习上都表现出了不错的预测性能[8-11]。Elish 等[12]比较了 SVM 和八种传统的统计和学习算法,比如决策树 DT(Decision Tree)、朴素贝叶斯、AdaBoost 等,实验结果显示 SVM 在多数数据集上优于其它算法。通过调研分析发现,SVM 在软件缺陷预测上表现出了不错的预测效果,而且代价敏感是处理非平衡数据常用的方法。在本文中,我们结合 SVM 和代价敏感来建立软件缺陷预测模型。

SVM 是平衡数据下非常优秀的二分类器,但标准的 SVM 并不能很好地处理非平衡数据[13,14]。为了使 SVM 能处理非平衡问题,常见的做法主要有两种:一是数据预处理[15,16],虽然这种方法被广泛用作简单的解决不平衡问题的解决方案,但大多数数据预处理方法由于随机抽样而具有共同的限制。预处理后可能得到完全不同于原来的数据集,从而导致严重偏差的数据分布。另一种方法是算法层面的改进,代价敏感学习是最常见的解决方法[4,17,18],已有的研究工作都是通过调整对负类和正类的惩罚因子,使分类面向少数类偏移。但是,因为 SVM 训练过程中对所有的训练样本是平等对待的,这会使 SVM 分类器对噪音和孤立点数据样本极为敏感,进而导致了过拟合的情况发生,已有的研究方法并不能很好地解决这个问题。我们在代价敏感 SVM 的基础上构建了 CCS-SVM(Cluster Cost Censitive-SVM)软件缺陷预测模型。首先通过聚类找到每类样本的中心点,根据样本到其中心点的距离定义每个样本的类别置信度,给每个样本不同的误分代价系数,通过给每个样本分配不同的误分代价来建立 SVM 模型,提升分类性能。然后用遗传算法优化模型参数和特征选择。

本文的实验是基于 NASA MDP(Metrics Data Program)[19]数据集,这是软件缺陷预测研究使用最广泛的数据集[3-5,7,12,20]。本文将 CCS-SVM 与 AdaBoost、AdaBoost.NC、朴素贝叶斯(NB)、随机森林(RF)、k 近邻(k-NN)、决策树(DT)进行了比较,评价指标包括 *F-measure* 值和 *G-mean* 值。实验结果表明,本文提出的算法在 *F-measure* 值和 *G-mean* 值上都有明显的提升,在 *G-mean* 值上比其余算法中的最优算法平均高出 9 个点,在 *F-measure* 值上平均高出 5 个点。

2 基于聚类的代价敏感支持向量机(CCS-SVM)

2.1 支持向量机(SVM)

支持向量机的核心思想是寻找两类样本之间的最优分类面。最优分类面必须保证最高的分类准确度和最大的分类间隔。

假设样本集 $z=\{(\boldsymbol{x}_i,y_i),i=1,\cdots,m\}$,其中,

$\boldsymbol{x}_i$ 为样本的特征向量,$\boldsymbol{x}_i=(x_i^{(1)},\cdots,x_i^{(n)})\in\mathbf{R}^n$,代表样本不同的指标;$y_i$ 代表不同的样本类别。对于二分类问题,所有样本分为两类:A 和 B,以 $y_i\in\{-1,+1\}$,$i=1,\cdots,m$ 来表示,每一个样本 $\boldsymbol{x}_i$ 只对应一个 y_i,若 $y_i=+1$,则将 $\boldsymbol{x}_i$ 划分为 A 类;若 $y_i=-1$,则将 $\boldsymbol{x}_i$ 划分为 B 类。

假设最优分类面方程为 $\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b=0$,其中,$\boldsymbol{\omega}^{\mathrm{T}}$ 是一个可调参数向量,b 是一个常数,则对于 $y_i=+1$,$\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b\geqslant 0$;对于 $y_i=-1$,$\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b\leqslant 0$。因此,可以得到式(1)。

$$y_i[\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b]\geqslant 0,i=1,\cdots,m \tag{1}$$

其中,$(\boldsymbol{x}_i):\mathbf{R}^n\rightarrow\mathbf{R}^d$ 表示映射函数,将 n 维向量映射到 d 维,将低维不可分的样本集非线性地映射到高维空间,从而能够线性可分。式(1)中,使等号成立的样本点 $\boldsymbol{x}_i$ 被称为支持向量。

由最优分类面 $\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b=0$ 可以计算出两类样本的分类间隔为 $2/\|\boldsymbol{\omega}^2\|$。因此,要保证分类间隔最大,就等价于使 $\|\boldsymbol{\omega}\|^2/2$ 最小。另外,考虑到一些噪音样本点会导致这些样本即使在映射后的高维空间也会被判别错误,因此支持向量机模型在优化约束问题中增加一个松弛变量 $\xi,\xi>0$。最终,支持向量机模型变为式(2)[21]。

$$\begin{cases}\min\limits_{\boldsymbol{\omega},b,\xi}\left(\dfrac{1}{2}\|\tilde{\boldsymbol{\omega}}\|^2+C\sum\limits_1^l\xi_i\right),\\ \text{s.t.}\quad y_i\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b\geqslant 1-\xi_i,\\ \qquad \xi_i\geqslant 0,i=1,2,\cdots,m\end{cases} \tag{2}$$

其中,$C>0$,代表样本被错误分类时的惩罚系数,C 数值越大,表明对错误的惩罚越严重。

根据式(2)求得最优解 $\boldsymbol{\omega}^*$、b^* 和 ξ^*,构造划分超平面,进而求得决策函数 $f(\boldsymbol{x})=\mathrm{sgn}((\boldsymbol{\omega}^*\ \boldsymbol{x}_i)+b^*)$。由式(2)确定的优化目标包含两部分:最大化分类间隔和最小化训练错误。两个优化目标是相互矛盾的,而惩罚参数 C 则起着平衡这两个目标的作用。

2.2 CCS-SVM

代价敏感 SVM 算法会使分类器对噪音和孤立点数据样本非常敏感,容易导致过拟合。为了提高 SVM 算法在缺陷预测上的性能,本文提出一种基于聚类计算每个样本误分代价的代价敏感 SVM,在代价敏感 SVM 的基础上,根据每个样本到聚类中心的距离计算其置信度,置信度乘上该样本类别的惩罚因子就是该样本的误分类代价。

标准的 SVM 对所有样本的误分代价相等,如果对正类和负类使用不同的惩罚参数 C_+ 和 C_- 来代替原来的参数 C,从而得到代价敏感支持向量机 CS-SVM,使得分类时加大对少数类的误分代价,从而提高对少数类样本的分类能力,如图 1 所示[22]。

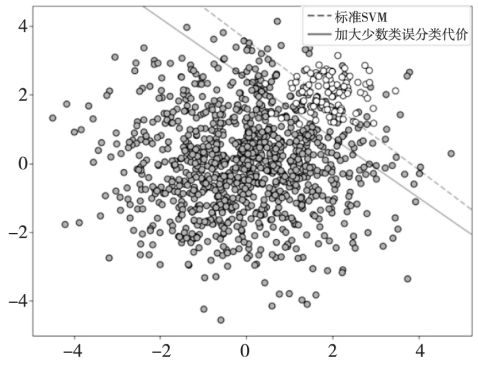


Figure 1 Performance comparison between SS-SVM and standard SVM on unbalanced data

图 1 SS-SVM 和标准 SVM 在非平衡数据上性能对比

标准 SVM 的目标函数优化为式(3)的目标函数:

$$\begin{aligned}&\min_{\boldsymbol{\omega},b,\xi}\left(\frac{1}{2}\|\tilde{\boldsymbol{\omega}}\|^2+C_+\sum_1^l\xi_{i;y_i=+1}+C_-\sum_1^l\xi_{i;y_i=-1}\right),\\ &\text{s.t.}\quad y_i\boldsymbol{\omega}^{\mathrm{T}}\ (\boldsymbol{x}_i)+b\geqslant 1-\xi_i,\\ &\qquad \xi_i\geqslant 0,i=1,2,\cdots,m\end{aligned} \tag{3}$$

其中,C_+ 为正类较高的误分代价,C_- 为负类较低的误分代价,其余参数的意义与式(2)中的一样。

CS-SVM 对同一类样本的误分代价是相同的,这样会导致模型对噪音和异常样本过分敏感,为了解决这个问题,我们提出了一种基于 CS-SVM 和聚类算法改进的代价敏感 SVM 算法(CCS-SVM)。我们主要的改进是给每个样本赋予类别置信度,用聚类算法找到每类样本的中心点,同一类别中,距离中心点越近的样本类别置信度越高,反之则越小,根据样本的类别置信度赋予不同的误分代价,从而减少或忽略噪音或者孤立点对预测模型的影响,得到性能更好的预测模型。

Frey 等[23]提出了近邻传播聚类算法 AP(Affinity Propagation),该算法对聚类中心的选取做了改进,无需在最开始就选择聚类中心,而是把所有样本点都作为潜在的聚类中心,最后通过样本点之间的消息传递得到最优的聚类中心,这是其他传统的聚类算法(如 K 中心聚类算法)所不能及的。我们使用 AP 聚类算法来计算各个类别的中心点,具体算法描述如下:

算法 1 样本中心点计算

输入:数据集 $DataSet$,类别 $Label$。

输出:类别 $Label$ 的中心点。

1 得到类别为 $Label$ 的样本;
2 使用 AP 算法聚类成 K 个簇;
3 for $t=1,\cdots,K$ do
4 得到第 t 个簇的中心点 $centers[t]$;
5 用第 t 个簇的数量除以该类别样本总数得到权重 $weights[t]$;
6 end for
7 根据每个簇的中心点乘上对应权重累加后计算出类别 $Label$ 的中心点;

算法 1 中第 7 行中心点的计算公式如式(4)所示:

$$\boldsymbol{Cen}^{Label}=\sum_{k=1}^{K}weights[k]\times\boldsymbol{centers}[k] \quad (4)$$

其中,$Label$ 为类标记,K 为聚类后簇的数量,$weights[i]$为第 i 个簇的权重,$\boldsymbol{centers}[i]$为第 i 个簇的中心点。

通过该算法得到的中心点明显会偏向样本数量较多的簇,计算出每个类别的中心点以后,接下来根据每个样本到中心点的距离计算每个样本的置信度:

先定义 $Dis_{\max}$为样本到中心点的最大距离:

$$Dis_{\max}=\max(\|\boldsymbol{x}_i-\boldsymbol{Cent}^{Label}\|)$$

样本 $\boldsymbol{x}_i$ 的置信度为:

$$Confi(\boldsymbol{x}_i)=1-\frac{\|\boldsymbol{x}_i-\boldsymbol{Cent}^{Label}\|^2}{Dis_{\max}^2}$$

计算出每个样本的置信度后,在标准代价敏感支持向量机目标函数中加入了样本置信度 $Confi(\boldsymbol{x}_i)$,重新定义 SVM 的目标函数为式(5):

$$\begin{aligned}\min_{\boldsymbol{\omega},b,\xi}\Big(&\frac{1}{2}\|\tilde{\boldsymbol{\omega}}\|^2+C_+\sum_{1}^{l}\xi_{i:y_i=+1}*Confi(\boldsymbol{x}_i)+\\&C_-\sum_{1}^{l}\xi_{i:y_i=-1}*Confi(\boldsymbol{x}_i)\Big),\\\text{s.t.}\quad&y_i\boldsymbol{\omega}^{\mathrm{T}}(\boldsymbol{x}_i)+b\geqslant 1-\xi_i,\\&\xi_i\geqslant 0,i=1,2,\cdots,m\end{aligned} \quad (5)$$

2.3 基于 CCS-SVM 的软件缺陷预测方法

2.3.1 基本步骤

按照传统的预测方法,模型的建立过程主要包含数据预处理、特征处理和模型训练三个部分,三个环节中任意一个环节没有处理好都会影响模型性能。为了提高模型性能,除了在算法上做优化,对数据处理和特征处理也做了相应的改进,本文提出的缺陷预测模型建立过程具体步骤如下所示:

步骤 1 获取软件复杂度度量数据集;

步骤 2 数据标准化处理,对特征去除均值和方差缩放;

步骤 3 用训练样本基于 CCS-SVM 建立软件缺陷预测模型,选择径向基函数 RBF(Radial Basis Function)核函数;

步骤 4 利用遗传算法进行模型参数 C_+(正类的惩罚参数)、C_-(负类的惩罚参数)、σ(RBF 核函数的参数)和特征选择的优化;

在步骤 4 的特征选择上面,没有直接采用传统的 Filter、Wrapper 和 Embedded 等[24]方法,而是直接使用遗传算法做选择特征,用一个二进制编码的参数表示特征的选择状态,1 表示选中该特征,0 表示不选中该特征。Huang 等[25]验证了该方案的可行性。

2.3.2 基于遗传算法的参数优化与特征选择

遗传算法是借鉴生物界自然选择和遗传机制的一种启发式搜索算法,它遵循"适者生存,优胜劣汰"的原则[26]。遗传算法是通过交叉、选择及变异等处理机制对一种群的进化过程进行模拟,在每次迭代进化过程中都保留一组已有的候选个体,经过若干进化周期后,种群在理想情况下,它的适应度能达到近似最优的状态。遗传算法多用于提高求解复杂系统问题的效率,自被提出以来,在函数优化、神经网络、自适应控制、生产调度、模式识别、机器学习、人工生命以及图像处理等领域得到了广泛的应用。目前,遗传算法不仅在特征选择方面有很多成功的实践,在 SVM 参数选择方面也已经有很多成功的实践[4,27-29]。本文利用遗传算法对模型参数 C_+(正类的惩罚参数)、C_-(负类的惩罚参数)、σ(RBF 核函数的参数)和特征的选择状态(f)进行优化。

本实验种群中的个体由 C_+、C_-、σ 和 f 4 个参数组成。使用评价函数 $G\text{-}mean$ 作为目标函数,种群大小设置为 30,迭代次数为 100。具体如算法 2 所示。

算法 2 模型参数优化和特征选择算法

输入:训练数据和 SVM 模型,参数范围和目标函数阈值,种群规模,繁殖次数 T,复制概率,交叉概率,突变概率。

输出:缺陷预测模型。

1 设置遗传算法参数;
2 生成个体种群;
3 for $t=1,\cdots,T$ do
4 用训练数据和 CCS-SVM 模型对每个个体进行 k 折交叉验证,计算 $G\text{-}mean$ 值;
5 计算个体适应值;
6 选出最好的个体和最好的函数值;
7 保存每次繁殖后最好的结果;
8 采用轮盘赌算法进行自然选择,淘汰一部分适应性低的个体;
9 交叉繁殖;

10 基因突变;

11 end for;

对于极度不平衡或者样本数量比较少的数据集,采用 5 折交叉验证,否则采用 10 折交叉验证。

在本文中,算法首先在参数范围内随机产生第一代种群,目标函数设置为 *G-mean* 值。随着种群的不断进化,目标函数值不断优化并趋于稳定,当目标函数值达到预先设定的阈值或者连续 5 代没有提升时,提前结束算法。

3 实验与结果分析

3.1 实验环境

实验环境为 4 核 Intel Corei5-4210M CPU,主频为 2.60 GHz,12 GB 内存,Ubuntu 16.04 操作系统。算法实现使用 python 编程语言,在 Sklearn 开源机器学习库上进行改进。

3.2 数据描述

为了进一步验证 CCS-SVM 缺陷预测模型的有效性,在 NASA MDP[6] 数据库中使用 12 组数据集进行实验。表 1 中描述了 12 组数据集,表 2 显示了用于表示实验中使用的软件特征的度量元,主要包括 McCabe 度量元(Mc-Cabe)[30]、line of code 度量元(LineCount)、Halstead 基本度量元(BHalstead)[19]及其扩展量元(DHalstead)。从表 1 中可以看出,有缺陷模块占比显著低于无缺陷模块,有缺陷模块平均占比为 16.01,CM1、PC2 两个模块中,有缺陷模块占比仅为 2.31 和 2.51,绝大部分软件缺陷预测数据属于非平衡数据。

Table 1 Software defect date sets

表 1 缺陷数据集

Name	Samples	Attributes	Defect Class	Non-Defect Class	%Defect
KC1	1 183	21	314	869	26.54
KC3	194	39	36	158	18.56
PC4	1287	37	177	1 110	13.75
MW1	253	37	27	226	10.67
MC2	125	39	44	81	35.20
CM1	1 988	38	46	1 942	2.31
PC2	745	36	16	729	2.15
PC5	1 711	38	471	1 240	27.53
PC3	1 077	37	134	943	12.44
MC1	327	37	42	285	12.84
PC1	705	37	61	644	8.65
JM1	7 782	21	1 672	6 110	21.49

Table 2 Software metric element

表 2 软件度量元

metrics	type	metrics	type
V(g)	McCabe	E	DHalstead
IV(g)	McCabe	UniqOpnd	DHalstead
EV(g)	McCabe	TotalOp	DHalstead
LOC	McCabe	UniqOp	DHalstead
L	DHalstead	LOCcode	LineCount
V	DHalstead	LOCComment	LineCount
N	DHalstead	TotalOpnd	DHalstead
I	DHalstead	LOCBlank	LineCount
D	DHalstead	LOCCodeAndcomment	LineCount
T	DHalstead	UniqOp	DHalstead
B	DHalstead	…	…

3.3 模型评价

软件缺陷预测领域对模型评价指标主要使用 *F-measure* 和 *G-mean* 值,将有缺陷模块视为正类,无缺陷模块视为负类,得到预测模型后,测试样本可以在混淆矩阵中分为四种情况,如表 3 所示。

Table 3 Confusion matrix

表 3 混淆矩阵

实际值	预测值	
	正类	负类
正类	TP	FN
负类	FP	TN

根据混淆矩阵中的 4 个值可以计算出真实的正类率 TP_{rate}、真实的负类率 TN_{rate}、查准率 PP_{rate}、*F-measure*[31]、和 *G-mean* 值,具体计算方式如式(6)所示:

$$\begin{aligned} TP_{\text{rate}} &= TP/(TP+FN),\\ TN_{\text{rate}} &= TN/(TN+FP),\\ PP_{\text{value}} &= TP/(TP+FP) \end{aligned} \tag{6}$$

如果只考虑有缺陷模块的性能,真实正类率 TP_{rate} 和正类预测值 PP_{value} 是重要的度量。在信息检索领域,将真实正类率 TP_{rate} 定义为查全率 *recall*,表示检索到的相关对象占实际正类的比例。将正类预测值 PP_{value} 定义为查准率 *precision*,表示相关对象占检索出的所有对象的比例。*F-measure* 是查全率和查准率的调和均值,其取值接近二者的较小值,因此较大的 *F-measure* 表示 *recall* 和 *precision* 都比较大[31]:

$$F\text{-}measure = \frac{(1+\beta^2)\times recall \times precision}{\beta^2 \times recall + precision}$$

其中,β 用于调节 *precision* 和 *recall* 的相对重要

度,通常取为1。如果同时关注模型在两个类别上的预测性能,即希望TP_{rate}和TN_{rate}都取较大值,可以使用*G-mean*度量学习算法在两个类上的平均性能:

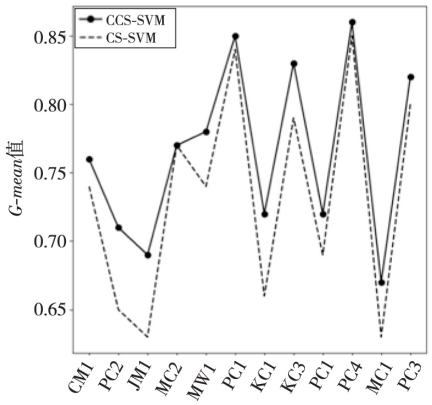
$$G\text{-}mean = \sqrt{TP_{rate} \times TN_{rate}}$$

本文使用*F-measure*和*G-mean*作为评价度量。

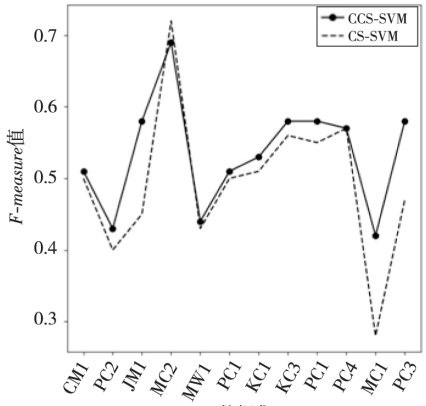
3.4 实验结果

3.4.1 CCS-SVM和CS-SVM对比

同样的使用遗传算法做选择特征和优化模型参数,用本文方法(CCS-SVM)和代价敏感SVM(CS-SVM)方法做对比实验,比较两种方法的模型性能,实验结果如图2所示。从图2可以看出,本文方法通过对每个样本赋予不同的误分系数,相对于CS-SVM对所有样本赋予相同的误分系数,本文方法在*G-mean*值和*F-measure*值上都有明显的提升。



a CCS-SVM和CS-SVM的G-mean比较



b CCS-SVM和CS-SVM的*F-measure*比较

Figure 2 Comparison between CCS-SVM and CS-SVM

图2 CCS-SVM和CS-SVM对比

3.4.2 CCS-SVM和相关算法对比

本文方法与AdaBoost、AdaBoost.NC、朴素贝叶斯(NB)、随机森林(RF)、k近邻(k-NN)、决策树(DT)进行对比。实验结果如表4和表5所示。

Table 4 Comparison of model performance *F-measure* value

表4 模型性能*F-measure*值比较

Name	Prediction model						
	AdaBoost	AdaBoost.NC	NB	RF	k-NN	DT	CCS-SVM
KC1	0.38	**0.55**	0.38	0.37	0.35	0.39	0.53
KC3	0.38	0.56	0.34	0.17	0.15	0.34	**0.58**
PC4	0.54	**0.59**	0.31	0.45	0.27	0.50	0.57
MW1	0.24	0.28	0.36	0.19	0.08	0.26	**0.44**
MC2	0.46	0.55	0.43	0.47	0.48	0.48	**0.69**
CM1	0.35	**0.51**	0.24	0.11	0.22	0.32	**0.51**
PC2	0.07	0.12	0.05	0.00	0.00	0.06	**0.43**
PC5	0.48	**0.59**	0.24	0.48	0.41	0.49	0.58
PC3	0.23	0.42	0.23	0.14	0.18	0.24	**0.58**
MC1	0.37	**0.51**	0.07	0.28	0.23	0.40	0.42
PC1	0.30	0.43	0.32	0.22	0.11	0.35	**0.51**
JM1	0.33	**0.52**	0.23	0.27	0.27	0.32	0.45
Avg	0.34	0.47	0.27	0.26	0.23	0.35	0.52

Table 5 Comparison of model performance *G-mean* value

表5 模型性能*G-mean*值比较

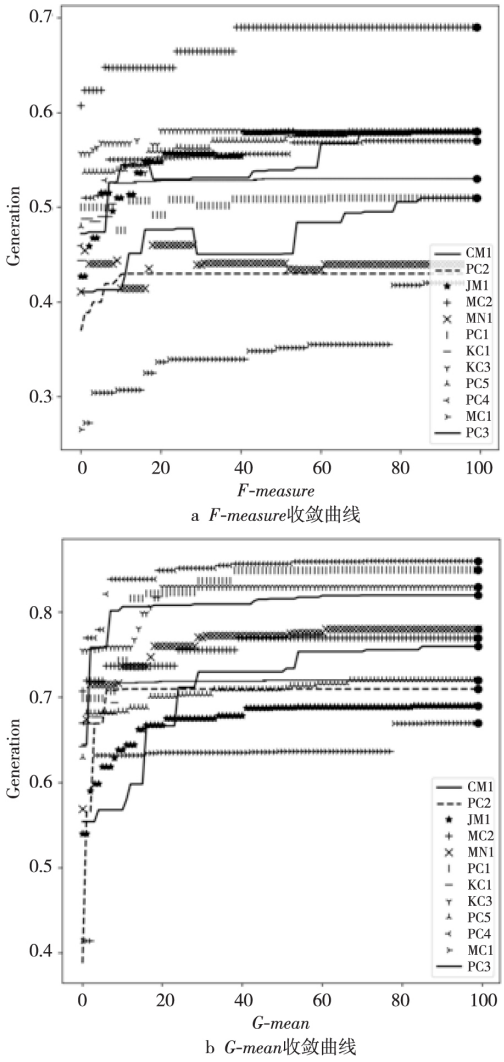
Name	Prediction model						
	AdaBoost	AdaBoost.NC	NB	RF	k-NN	DT	CCS-SVM
KC1	0.54	**0.72**	0.53	0.51	0.52	0.55	**0.72**
KC3	0.55	0.78	0.53	0.29	0.27	0.51	**0.83**
PC4	0.71	**0.86**	0.47	0.57	0.48	0.68	**0.86**
MW1	0.48	0.63	0.66	0.29	0.16	0.47	**0.78**
MC2	0.57	0.72	0.53	0.57	0.59	0.58	**0.77**
CM1	0.54	0.70	0.44	0.16	0.43	0.51	**0.76**
PC2	0.10	0.10	0.26	0.00	0.00	0.11	**0.71**
PC5	0.62	**0.73**	0.38	0.59	0.57	0.63	0.72
PC3	0.44	0.69	0.29	0.27	0.38	0.45	**0.82**
MC1	0.61	**0.72**	0.31	0.43	0.44	0.63	0.67
PC1	0.54	0.78	0.53	0.34	0.27	0.59	**0.85**
JM1	0.53	**0.72**	0.38	0.42	0.48	0.52	0.69
avg	0.52	0.68	0.44	0.37	0.38	0.52	0.77

从表4和表5可以看出,本文方法在大部分数据集上优于已有的几种学习算法,在*G-mean*值上比其余算法中的最优算法平均高出9个点,在*F-measure*值上平均高出5个点。在PC2和CM1这两个极度不平衡的数据集上面,AdaBoost.NC等几种算法在*G-mean*值上有明显的下降,本文方法也表现得很稳定。从图2可以看出,CS-SVM的*G-mean*值大部分超过了0.65、*F-measure*值大部

分超过了 0.4,已有研究[3-5,7,12,20]和我们的实验结果都显示,CS-SVM 的模型性能明显优于几种传统的算法,我们在 CS-SVM 的基础上给每个样本加入误分代价建立的 CCS-SVM 模型,进一步提升了预测性能,有效提升了算法的鲁棒性和泛化能力。

3.4.3 *G-mean* 和 *F-measure* 值的收敛曲线

G-mean 和 *F-measure* 值的收敛曲线如图 3 所示。



a *F-measure*收敛曲线

b *G-mean*收敛曲线

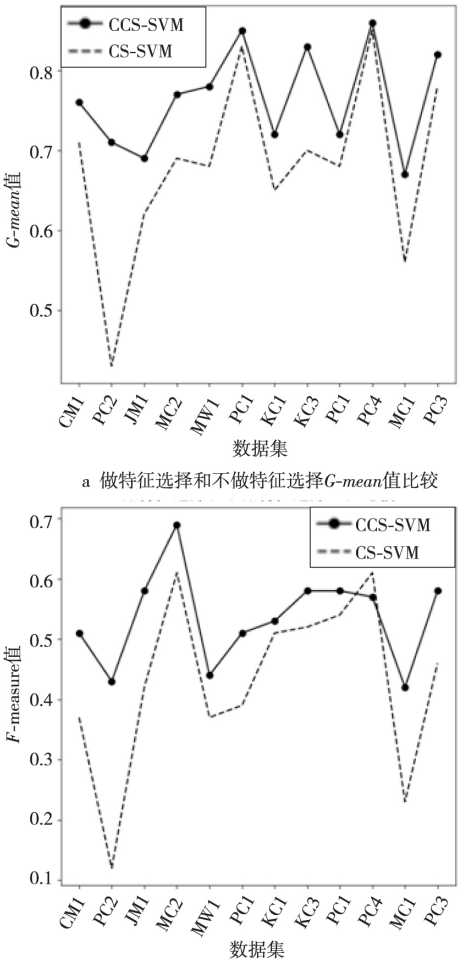
Figure 3 Convergence curves of *F-measure* and *G-mean* values

图 3 *F-measure* 值和 *G-mean* 值的收敛曲线

从图 3 可以看出,算法在迭代过程中是收敛的。前期迭代过程中最优解有较大的提升,迭代到 80 代左右达到全局最优解。遗传算法迭代过程中是以 *G-mean* 值作为目标函数,但从图 3 可以看出,*F-measure* 值也是收敛的,因为 *F-measure* 值和 *G-mean* 值的目的都是为了在评估模型的时候能够兼顾到查全率和查准率。

3.4.4 特征选择实验

用本文方法做特征选择和不做特征选择的对比,从而验证特征选择是否对模型性能有影响。为了做对比实验,本文在第二种方法中使用全部特征建立模型,不再优化特征选择,其余处理过程完全一样,实验结果如图 4 所示。从图 4 可以看出,特征选择对模型性能影响比较明显,在 PC2、MC1 和 KC3 这三个数据集上表现得非常明显,数据集 PC2、MC1 和 KC3 是 12 个数据集中不平衡度最大的几个数据集,进行特征选择后,显著提高了分类性能。



a 做特征选择和不做特征选择*G-mean*值比较

b 做特征选择和不做特征选择*F-measure*值比较

Figure 4 Comparison between feature selection and non feature selection

图 4 做特征选择和不做特征选择对比

4 结束语

本文的贡献是针对软件缺陷预测问题提出了一种基于代价敏感 SVM 的软件缺陷预测方法,基本思想是通过聚类算法找到样本的中心点,根据样

本到中心点的距离对每个样本分配不同的误分代价,弱化噪音和孤立点数据样本对模型的影响,并用遗传算法优化模型参数和做特征选择。实验表明,该模型比已有方法具有更好的预测性能,但对于样本数量过大的数据集,该方法训练模型的时间较长。如何进一步减少模型训练时间、提高模型性能是今后的课题。

参考文献:

[1] Wang Qing, Wu Shu-jian, Li Ming-shu. Software defect prediction[J]. Journal of Software, 2008, 19(7): 1565-1580. (in Chinese)

[2] Shi Hai-feng. Analysis of software defect management scheme[J]. Computer & Telecommunication, 2014 (10): 71-73. (in Chinese)

[3] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors[J]. IEEE Transactions on Software Engineering, 2007, 33(9): 637-640.

[4] Shuai B, Li H F, Li M J, et al. Software defect prediction using dynamic support vector machine[C]// Proc of the 2013 9th International Conference on Computational Intelligence and Security (CIS), 2013: 260-263.

[5] Seliya N, Khoshgoftaar T M, Van Hulse J. Predicting faults in high assurance software[C]// Proc of 2010 IEEE 12th International Symposium on High-Assurance Systems Engineering (HASE), 2010: 26-34.

[6] Nasa I V V. Metrics data program[EB/OL]. [2004-12-03]. http://mdp.ivv.nasa.gov/.

[7] Wang S, Yao X. Using class imbalance learning for software defect prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434-443.

[8] Lee W, Jun C H, Lee J S. Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification[J]. Information Sciences, 2017, 381: 92-103.

[9] Choeikiwong T, Vateekul P. Software defect prediction in imbalanced data sets using unbiased support vector machine[M]// Information Science and Application, Berlin: Springer Berlin Heidelb, 2015: 923-931.

[10] Jian C, Gao J, Ao Y. A new sampling method for classifying imbalanced data based on support vector machine ensemble [J]. Neurocomputing, 2016, 193(C): 115-122.

[11] Jiang Hui-yan, Zong Mao, Liu Xiang-ying. Research of software defect prediction model based on ACO SVM[J]. Chinese Journal of Computers, 2011, 34(6): 1148-1154. (in Chinese)

[12] Elish K O, Elish M O. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems and Software, 2008, 81(5): 649-660.

[13] Cao Peng, Zhao Da-zhe, Zaiane O. An optimized cost sensitive SVM for imbalanced data learning [C]// Proc of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2013: 280-292.

[14] López V, Fernández A, García S, et al. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics[J]. Information Sciences, 2013, 250(11): 113-141.

[15] Rivera W A, Xanthopoulos P. A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets[J]. Expert Systems with Applications, 2016, 66: 124-135.

[16] Vluymans S, Tarragó D S, Saeys Y, et al. Fuzzy rough classifiers for class imbalanced multi-instance data[J]. Pattern Recognition, 2016, 53(C): 36-45.

[17] Lin Y, Lee Y, Wahba G. Support vector machines for classification in nonstandard situations[J]. Machine Learning, 2002, 46(1): 191-202.

[18] Khan S H, Hayat M, Bennamoun M, et al. Cost-sensitive learning of deep feature representations from imbalanced data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(8): 3573-3587.

[19] Halstead M H. Elements of software science [M]. New York: Elsevier, 1977.

[20] Gray D, Bowes D, Davey N, et al. Software defect prediction using static code metrics underestimates defect-proneness [C]// Proc of the 2010 International Joint Conference on Neural Networks (IJCNN), 2010: 1-7.

[21] Cai Yan-yan, Song Xiao-dong. New fuzzy SVM model used in imbalanced datasets[J]. Journal of Xidian University, 2015, 42(5): 120-124. (in Chinese)

[22] Zhao Yong-bin, Chen Shuo, Liu Ming, et al. Imbalanced data learning for support vector machine based on confidence cost sensitivity[J]. Computer Engineering, 2015, 41(10): 177-180. (in Chinesse)

[23] Frey B J, Dueck D. Clustering by passing messages between data points[J]. Science, 2007, 315(5814): 972-976.

[24] Chandrashekar G, Sahin F. A survey on feature selection methods[J]. Computers & Electrical Engineering, 2014, 40(1): 16-28.

[25] Huang C L, Wang C J. A GA-based feature selection and parameters optimization for support vector machines[J]. Expert Systems with Applications, 2006, 31(2): 231-240.

[26] Mitchell M. An introduction to genetic algorithms [M]. Cambridge: MIT Press, 1998.

[27] Wu C H, Tzeng G H, Goo Y J, et al. A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy[J]. Expert Systems with Applications, 2007, 32(2): 397-408.

[28] Lee Y, Lin Y, Wahba G. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data[J]. Journal of the American Statistical Association, 2004, 99(465): 67-81.

[29] Sajan K S, Kumar V, Tyagi B. Genetic algorithm based support vector machine for on-line voltage stability monitoring [J]. International Journal of Electrical Power & Energy Systems, 2015, 73: 200-208.

[30] McCabe T J. A complexity measure[J]. IEEE Transactions on Software Engineering, 1976, 4(SE-2): 308-320.

[31] Dong Yuan-fang, Li Xiong-fei, Li Jun. Gradually learning algorithm for imbalanced data [J]. Computer Engineering, 2010, 36(24): 161-163. (in Chinese)

附中文参考文献:

[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] 史海峰. 软件缺陷管理方案分析[J]. 电脑与电信, 2014 (10): 71-73.

[11] 姜慧研,宗茂,刘相莹. 基于 ACO-SVM 的软件缺陷预测模型的研究[J]. 计算机学报, 2011, 34(6): 1148-1154.

[21] 蔡艳艳,宋晓东. 针对非平衡数据分类的新型模糊 SVM 模型[J]. 西安电子科技大学学报(自然科学版), 2015, 42(5): 120-124.

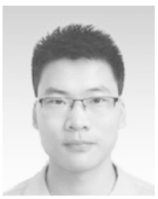
[22] 赵永彬,陈硕,刘明,等. 基于置信度代价敏感的支持向量机不均衡数据学习[J]. 计算机工程, 2015, 41(10): 177-180.

[31] 董元方,李雄飞,李军. 一种不平衡数据渐进学习算法[J]. 计算机工程, 2010, 36(24): 161-163.

作者简介:

任胜兵(1969—),男,湖南岳阳人,博士,副教授,CCF 会员(10663S),研究方向为可信软件、图像处理和模式识别。**E-mail**: rsb@csu.edu.cn

REN Sheng-bing, born in 1969, PhD, associate professor, CCF member(10663S), his research interests include dependable software, image processing, and pattern recognition.



廖湘荡(1993—),男,湖南涟源人,硕士生,研究方向为可信软件。**E-mail**: lxhao580@163.com

LIAO Xiang-dang, born in 1993, MS candidate, his research interest includes dependable software.