软件缺陷集成预测模型研究*

刘小花，王　涛，吴振强
（陕西师范大学 计算机科学学院，西安 710062）

摘　要：利用单一分类器构造的缺陷预测模型已经遇到了性能瓶颈，而集成分类器相比单一分类器往往具有显著的性能优势。以构造高效的集成缺陷预测模型为出发点，比较了七种不同类型集成分类器的算法和特点。在 14 个基准数据集上的实验显示，部分集成预测模型的性能优于基于朴素贝叶斯的单一预测模型。其中，基于投票的集成分类框架具有最优的预测性能以及统计学意义上的性能优势显著性，随机森林算法次之。Stacking 集成框架也具有较强的泛化能力。

关键词：软件缺陷预测；集成分类；投票；随机森林
中图分类号：TP311.5　　**文献标志码**：A　　**文章编号**：1001-3695(2013)06-1734-05
doi:10.3969/j.issn.1001-3695.2013.06.035

Software defect prediction based on classifiers ensemble

LIU Xiao-hua, WANG Tao, WU Zhen-qiang
(*School of Computer Science, Shaanxi Normal University, Xi'an* 710062, *China*)

Abstract: Software defect prediction using classification algorithms was advocated by many researchers. However, several new literatures show the performance bottleneck by applying a single classifier recent years. On the other hand, classifiers ensemble can effectively improve classification performance than a single classifier. This paper conducted a comparative study of various ensemble methods with perspective of taxonomy. A series of benchmarking experiments on public-domain datasets MDP show that applying classifiers ensemble methods to predict defect could achieve better performance than using a single classifier. Specially, in all seven ensemble methods evolved by this experiments, voting and random forest have obvious performance superiority than others, and Stacking also has better generalization ability.

Key words: software defect prediction; classifiers ensemble; vote; random forest

0　引言

软件缺陷预测的目的是在软件生命周期的各个阶段为缺陷检测提供指导，将有限的测试资源和时间进行合理的分配。因此，研究构造有效的缺陷预测模型具有极为重要的意义。应用包括遗传算法、回归方法、神经网络、关联规则、决策树等数据挖掘或机器学习方法进行缺陷预测取得了不错的效果。然而，研究者已经发现了缺陷预测模型的性能瓶颈[1~4]。这些结果显示单纯依靠更强的数据挖掘技术已不足以得到更好的分类效果。基于软件度量的缺陷预测研究似乎已经遇到了瓶颈。

许多理论和实验研究证明，将多个分类器集成在一起，然后用于分类或预测将会使决策更为准确[5]。然而，将集成分类器(classifiers ensemble)技术应用于缺陷预测的研究却鲜有报道。Tosun 等人[6]提出了集成朴素贝叶斯、人工神经网络以及特征间隔投票(voting feature intervals)三种不同的算法，应用于缺陷预测后相比朴素贝叶斯算法显著地提高了预测能力，其不足之处在于实验仅在少数的数据集上得以验证。Zheng[7]针对缺陷预测研究了基于神经网络的三种代价敏感的 Boosting 算法，但他的工作仅仅集中在 Boosting 这一种集成模型上。

目前利用分类器集成技术构造缺陷预测模型仍处于研究初级阶段。因此，本文主要研究两个问题：a)如何在缺陷预测中应用集成分类器；b)可应用于缺陷预测的分类器集成算法哪个性能更出色。本文分析了七种集成分类器构造缺陷预测模型的方法，基于构造方法的层次将这些集成方法进行了分类；在标准的缺陷预测数据集上，对七种集成方法进行了实验分析，并将其与单一分类器模型 Naïve Bayes 进行比较，从准确性指示方面给出了最优的集成缺陷预测模型；针对准确性实验结果开展了假设检验分析，从统计学角度分析了这些集成算法性能差异的显著性。

1　集成分类器

集成分类器是一种模仿人类第二天性的策略——人类总是在考虑了多方面因素后才作出某个关键决策[5]。集成分类器技术是综合多个分类器来构造分类模型的方法，通过有效地利用各种类型分类器的多样性，在不增加偏差的同时降低方差，从而可以有效地提高缺陷预测性能[5~7]。图 1 显示了集成分类器与单个分类器相比较的优势。A、B、C 三条直线分别代表三个分类器，标记为“+”的样本为实际正类样本，标记为“-”的样本为实际负类样本。容易发现，无论哪个分类器都

收稿日期：2012-10-10；**修回日期**：2012-11-24　**基金项目**：国家自然科学基金面上项目(61173190)；陕西省自然科学基础研究计划项目(2009JM8002)；中央高校基本科研业务费专项资金资助项目(GK201302055)

作者简介：刘小花(1980-)，女，甘肃永昌人，博士研究生，主要研究方向为数据挖掘、软件工程(floweret@snnu.edu.cn)；王涛(1980-)，男，讲师，博士，主要研究方向为数据挖掘、软件工程等；吴振强(1968-)，男，教授，博导，主要研究方向为信息安全、计算机网络等.

无法单独地彻底将所有样本分开。例如,*A* 将其右侧的样本分类为正类,将其左侧的样本分类为负类。因此,处于 3 号区域的样本 *A* 就无法对其正确分类。实际上,对于图 1 中描述的样本不可能有直线将其完美正确分类。然而,通过组合 *A*、*B*、*C* 三个分类器却有办法可以实现这一点。可以观察到,对于 1 号区域的样本,三个分类器都可分类为负类,而剩余的 2、3、4 三个区域中的样本,三个分类器的分类无法实现统一,但是通过一个简单的集成策略就可实现对样本的正确划分。显然,这个策略可以是多数投票,即 *A*、*B*、*C* 三个分类器对样本分类出现争议时采用持多数意见者获胜的策略。因此,2、3、4 三个区域中的样本将被正确分类。可以看出,分类器集成的结果就是将原有的三个直线分类器转换成一个多段的曲线分类器(图 1 中的粗体线段构成的曲线)。

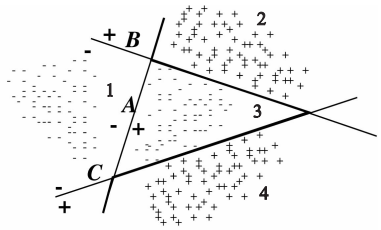

图1　集成分类器的优势

2　基于集成分类器的缺陷预测模型

本章将分析主流的七种集成技术如何构造缺陷预测模型。首先,假定预测对象即模块 *m* 由一组代码属性描述,*m* :{ a_1, a_2,…, a_n}。分类器的任务是决策 *m* 分类为 c_d 或者 c_{nd},其中 c_d 为缺陷类标记,而 c_{nd} 为非缺陷类标记。

Bagging[8](bootstrap aggregating)是一个著名的集成分类器算法。其核心思想是从训练集抽取出不同的部分(有交集)来训练集成分类器成员。预测阶段,采取平均或投票方式集成分类器成员。如算法 1 所示,Bagging 算法每次有放回地、均匀地采样 *N* 个样本,也就是说每个分类器由这次采样得到的数据集进行训练,并且该数据集和原始数据集的大小一致。因此,通常认为利用 Bagging 算法得到的集成分类器,其性能往往优于在原始数据集上训练出来的单个模型。

算法 1　Bagging 算法描述

Input: the number of ensemble members M; Training set S = {(m_1, c_1), (m_2,c_2),…, (m_N,c_N); c_1, c_2,…, c_N ∈ {c_d,c_{nd}}};Testing set T

Training phase:

for i = 1 to M do

Draw (with replacement) a bootstrap sample set S_i(N examples) of the data from S;

Train a classifier C_i from S_i and add it to the ensemble;

end for

Testing phase:

for each t in T do

Try all classifiers C_i;

Predict the class that receives the highest number of votes;

end for

Boosting 也是一个非常著名的集成算法,其中 AdaBoost[9] 是 Boosting 算法家族中最著名的一个实现。如算法 2 所示,AdaBoost 通过多个回合的迭代构造集成分类器,每个回合利用不同的样本加权来训练一个新的模型,整个迭代过程串行地完成。每个回合不正确分类样本的权重将增大,使得在下次迭代回合中,这些不正确分类的样本对分类器的训练贡献更大。该算法产生了一系列的分类器,并且后续的分类器是之前分类器的补充,最终 AdaBoost 以投票方式将这些分类器组合在一起。

算法 2　AdaBoost 算法描述

Input: the number of ensemble members M; Training set S = {(m_1, c_1), (m_2,c_2),…, (m_N,c_N); c_1, c_2,…, c_N ∈ {c_d,c_{nd}}}

Initialize: each training example weight $w_i = 1/N$ (i = 1,…,N)

Training phase:

for x = 1 to M do

Tran a classifier C_x using the current example weights;

compute a weighted error estimate: $err_x = \sum$ (w_i of all incorrectly classfied m_i)/$\sum_{i=1}^{N} w_i$

compute a classifier weight: $\alpha_x = \log((1 - err_x)/ err_x)/2$;

for all correctly classified examples m_i: $w_i \leftarrow w_i\, e^{-ax}$;

for all incorrectly classified examples m_i: $w_i \leftarrow w_i\, e^{ax}$

normalize the weights w_i so that they sum to 1;

end for

Testing phase:

for each t in T do

Try all classifiers C_x;

Predict the class that receives the highest sum of weights α_x;

end for

Dietterich[10] 提出了一种名叫随机 C4.5 的算法。这个算法更流行的名称为随机树(random tree)。随机树的主要思想是随机化算法的内部决策。具体讲,随机树实际上是 C4.5 的一个修改版本,通过对决策树的内部节点随机地从属性集合中选择来分裂。决策树中的每个节点都要测试得到前 20 个最优属性,并从中随机选择一个属性进行分裂。对于连续属性,同一个属性可能在多次测试中均处于前 20,因此被选中的概率也就更大。

随机森林算法(random forest)[11] 在构造每个 CART 决策树时也采用随机选取属性的方法。每个树的构造相对简单,而且都会完整成长,不会被剪枝。若令 *N* 为训练样本的个数,*M* 为属性的个数,则选定 $m < M$ 来确定在一个节点上作决定时,会使用到多少个属性。从 *N* 个训练样本中,以可重复采样的方式采样 *N* 次,形成一组训练集(即 bootstrap 采样)。对于每一个节点,随机选择 *m* 个基于此点的属性,并根据这 *m* 个属性计算其最佳的分割方式。对于样本的分类采取多数投票机制来集成分类器。随机森林算法的优点在于其速度快,且对于包含大量属性的样本很有效。

与随机森林不同的是,随机子空间算法(random subspace)利用在构造每个树时随机选取属性的方法来构造一个决策森林[12]。均匀地并随机地选取一个属性子集分配给任意一个学习算法,从而增加随机森林集成的多个成员之间的多样性。Ho[12] 的实验研究显示,在构造每棵决策树时,均匀地、随机地选取 50% 的属性可以获得较好的集成分类器。随机子空间算法同样对于包含大量属性的样本很有效。

Stacking 也是一个非常著名的集成学习技术[13]。Stacking 的框架包含两层分类器,level-0 为基分类器,level-1 为元分类器。基分类器由训练集以 bootstrap 采样的方式来构造,然后将基分类器的输出结果作为元分类器的输入,即训练集。元分类

器的任务就是合理组合输出集,纠正基分类器的分类错误,以便正确分类目标。

投票(vote)是一种分类器结合策略[14]。多数投票和加权多数投票是最常用的投票策略。多数投票机制是在分类时,每个分类器对样本的类投票,最终样本被分类为投票数多的那一类。加权多数投票采取加权求和的投票策略,权值依赖于分类器对预测的信任度及错误估计。

基于构造方法的不同层次,可以将集成分类器技术分为以下几类:数据级,指集成的焦点在于使用什么样的数据集或哪部分数据集来构造每个分类器;属性级,指利用不同的属性组合来构造单个分类器;分类器级,指利用不同的基分类器来构成集成分类器;组合级,定义不同的组合方式来集成单个分类器的结果。基于上述分类方法容易看出,Bagging 和 Boosting 属于数据级的集成技术;随机树和随机子空间属于属性级的集成技术;Stacking 属于分类器级的集成技术;投票属于组合级的集成技术;比较特殊的是随机森林,通常认为它是 Bagging 算法和随机子空间的混合体。

3 实验设计及结果分析

3.1 实验设计

本文采用的实验集是表 1 中描述的 NASA MDP[15] 的全部 14 个数据集。所有实验执行 10 折交叉验证[16],即对每个实验集分类算法随机采样分为 10 份,轮流将 9 份作为训练集训练生成分类器,而剩余 1 份作为本分类器的测试集,从而使得一个分类算法生成了 10 个分类器并测试得到 10 个结果。执行 10 次 10 折交叉实验,最终得到 100 个正确率结果,平均值即为该分类算法在该数据集上的平均准确率。

为了比较各种集成分类器技术在软件缺陷预测中的性能差异,本文利用知名的数据挖掘平台 Weka[17] 对 Bagging、AdaBoostM1、RandomForest、RandomTree、RandomSubSpace、Stacking 及 Vote 七种集成技术进行比较,并与简单高效的单一分类器 Naïve Bayes 进行对比。其中 Bagging、AdaBoostM1 及 RandomSubSpace 均为元分类器,本文均选取 Naïve Bayes 作为其基分类器,从而更好地体现对比效果。Vote 是组合多分类器投票算法,本文选取在缺陷预测中最常见的几种分类算法作为其基分类器,包括 Naïve Bayes、Logistic、LibSVM 以及 J48。组合策略为求概率平均值。Stacking 的 0 层分类器为 Naïve Bayes,而基分类器组合为 Naïve Bayes、Logistic、LibSVM 以及 J48。实验过程如算法 3 所描述。

算法 3 本文实验算法过程

Input: the MDP datasets, D = {CM1, JM1, KC1, KC2, KC3, KC4, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5}; all algorithms, A = {Bagging, AdaBoostingM1, NaïveBayes, Stacking, Vote, RandomForest, RandomTree, RandomSubspace}

Dataset preprocess:

a) Replacing missing attribute values; Apply ReplaceMissingValues of Weka to D;

b) Discretizing attribute values; Apply Discretize of Weka to D;

10-fold cross validation:

for each dataset in D do

for each algorithm in A do

Perform 10-fold cross validation;

end for

Perform Paired Two-tailed T-test;

end for

Output:

a) Accuracy and standard deviation of 8 classifiers on 14 datasets;

b) AUC and standard deviation of 8 classifiers on 14 datasets;

c) The results of Paired Two-tailed T-test.

3.2 评价方法

假定实际为正类,预测也为正类的样本数量称为正确正类(true positive,TP);实际为正类,预测为反类的称为错误反类(false negative,FN);实际为反类,预测为正类的称为错误正类(false positive,FP);实际为反类,预测也为反类的称为正确反类(true negative,TN)。分类准确率为 accuracy(AC) = (TP + TN)/(TP + FP + TN + FN)。本文采用的另一个评价指标为 AUC。AUC 是指 ROC 曲线下面包括的面积,即 ROC 曲线的积分。AUC 能以定量的方式表示该 ROC 曲线对应的分类器的泛化能力[16]。本文还对 10 折交叉验证生成的 AC 值和 AUC 值在多个数据集之间进行配对双尾 T 检验,以期在统计学意义上比较七种集成模型的性能差异显著性。

3.3 实验集预处理

为了本文实验的要求,对数据集进行了下列预处理:

a)替换缺失属性值。利用 Weka 的非监督过滤器 ReplaceMissingValues 将所有数据集中所有的缺失值(空值)进行替换。替换的规则是,对于字符型属性替换成模数;而对于数值型属性替换成均值。对于 MDP 而言,所有 14 个数据集的属性均为数值型,因此 ReplaceMissingValues 将所有数据集中所有的缺失值替换成其他所有样本该属性的均值。

b)属性值离散化。利用 Weka 的非监督 10-bin 离散化过滤器 Discretize 将所有数值属性的值离散化。

3.4 实验结果及其分析

实验得到的各分类算法在各数据集上的平均分类准确率结果如表 2 所示。由表 2 的数据至少可以得到如下结论:

a)对比 Bagging、AdaBoostM1 以及 Naïve Bayes 可以看出,采用 Naïve Bayes 作为基分类器的 Bagging 和 AdaBoostM1 对分类准确率的提高几乎没有贡献,甚至应用了 Bagging 后的 Naïve Bayes 在除了 KC4、PC3 以及 PC5 以外的 11 个数据集上分类准确率反而降低了,而应用了 AdaBoostM1 后的 Naïve Bayes 在除了 MC1、MW1、PC1、PC2 以及 PC3 以外的 9 个数据集上分类准确率均没有提高。虽然 Bagging 和 AdaBoostM1 在 14 个数据集上的平均分类准确率均高于 Naïve Bayes,但提高却非常有限。

b)以 Naïve Bayes 作为基分类器的 RandomSubSpace 在除了 MC2 和 JM1 两个数据集以外的其他 12 个数据集上的分类准确率较 Naïve Bayes 均有提高,平均分类准确率提高了 1.12%。

c)对于所有数据集上的平均分类准确率而言,RandomForest、RandomTree、Stacking 及 Vote 明显优于 Naïve Bayes 及其他集成算法。Vote 具有最高 88.48% 的分类准确率,其次是 RandomForest 的 87.90%,相比其他算法,这两者的分类准确性具

有明显的优势。

d)Vote 在 14 个数据集中的 7 个数据集上均取得了最优分类准确率,表现最为突出;其次,RandomForest 在 4 个数据集上取得了最优分类准确率。

e)实验证明了部分集成分类器应用于软件缺陷预测能够显著提高分类准确率。

本文还对每个分类算法在每个数据集上得到的 100 个分类准确率值进行了两两配对的双尾 T 检验,结果如表 3 所示。其中进行 T 检验的显著性水平为 0.05,在统计学意义上,表中的每一项 $w/t/l$ 表示该行的算法在 w 个数据集上的 AC 均值比该列的算法高;在 t 个数据集上行列算法没有显著差异;在 l 个数据集上该行对应算法的 AC 均值比该列的算法低。

表 1　NASA MDP 14 个数据集的基本信息

	CM1	JM1	KC1	KC2	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
language	C	C	C++	C++	Java	Perl	C++	C	C	C	C	C	C	C++
LOC	20k	315k	43k	18k	18k	25k	63k	6k	8k	40k	26k	40k	36k	164k
modules	505	10878	2107	522	458	125	9466	161	403	1107	5589	1563	1458	17186
defective	48	2102	325	105	43	75	68	52	31	76	23	160	178	516

表 2　在 14 个数据集上各个算法的 AC 和标准差比较　/%

dataset	Bagging	AdaBoostM1	Naïve Bayes	RandomForest	RandomTree	RandomSubSpace	Stacking	Vote
CM1	84.15 ±4.89	84.32 ±5.69	84.58 ±4.90	89.11 ±2.19	85.01 ±4.28	85.58 ±4.94	83.49 ±5.56	89.64 ±2.30
JM1	80.40 ±0.91	80.45 ±0.88	80.45 ±0.88	79.66 ±0.98	75.30 ±1.26	80.41 ±0.84	79.19 ±1.11	81.44 ±0.56
KC1	82.44 ±2.22	82.50 ±2.23	82.50 ±2.23	85.47 ±1.81	82.85 ±2.44	82.77 ±2.06	81.41 ±2.43	85.62 ±1.64
KC2	83.53 ±3.47	83.60 ±3.41	83.62 ±3.42	82.62 ±4.22	79.86 ±4.86	83.82 ±3.37	81.52 ±4.28	82.91 ±3.38
KC3	84.74 ±5.06	84.70 ±4.98	84.78 ±5.00	89.72 ±2.40	87.39 ±4.43	85.41 ±5.17	85.24 ±5.35	89.98 ±3.20
KC4	65.81 ±11.05	64.48 ±10.47	64.48 ±10.47	72.81 ±11.70	70.14 ±12.29	64.88 ±10.92	76.58 ±10.68	75.38 ±11.43
MC1	93.73 ±1.12	93.86 ±1.38	93.80 ±1.02	99.51 ±0.13	99.43 ±0.21	94.15 ±0.99	97.20 ±1.12	99.42 ±0.13
MC2	73.00 ±8.51	73.09 ±9.39	73.62 ±8.04	70.39 ±10.10	64.05 ±12.19	73.31 ±8.03	72.12 ±9.44	72.57 ±7.14
MW1	83.12 ±5.36	87.43 ±4.73	83.35 ±5.25	90.58 ±2.75	87.65 ±4.48	84.54 ±5.13	88.63 ±4.03	91.67 ±3.07
PC1	89.04 ±2.59	89.90 ±2.70	89.12 ±2.59	93.63 ±1.53	91.64 ±2.11	89.49 ±2.32	88.64 ±2.95	93.73 ±1.45
PC2	96.73 ±1.03	97.13 ±0.94	97.11 ±0.92	99.56 ±0.10	99.29 ±0.23	97.44 ±0.80	97.0 ±00.88	99.53 ±0.13
PC3	51.00 ±11.43	51.45 ±13.84	48.30 ± 10.00	89.83 ±1.35	86.01 ±2.59	59.22 ±10.94	87.01 ±2.68	89.12 ±1.77
PC4	86.99 ±2.86	86.78 ±3.27	87.11 ±2.57	90.13 ±2.05	87.74 ±2.83	87.30 ±2.48	87.67 ±2.66	90.28 ±1.75
PC5	96.51 ±0.36	96.44 ±0.38	96.44 ±0.38	97.54 ±0.26	97.08 ±0.37	96.58 ±0.39	96.01 ±0.48	97.46 ±0.23
MEAN	82.23 ±2.92	82.58 ±3.16	82.09 ±2.69	87.90 ±1.54	85.25 ±2.47	83.21 ±2.74	85.84 ±3.12	88.48 ±2.01

表 3　14 个数据集上 8 个分类算法的 AC 值两两配对双尾 T 检验的结果

algorithm	Bagging	AdaBoostM1	NaïveBayes	RandomForest	RandomTree	RandomSubSpace	Stacking
AdaBoostM1	2/12/0						
NaïveBayes	1/13/0	0/13/1					
RandomForest	10/3/1	9/4/1	10/3/1				
RandomTree	6/6/2	4/7/3	6/6/2	0/6/8			
RandomSubSpace	4/10/0	2/12/0	3/11/0	1/3/10	2/7/5		
Stacking	4/7/3	3/8/3	4/7/3	0/5/9	1/9/4	4/6/4	
Vote	12/2/0	12/2/0	12/2/0	1/12/1	9/5/0	12/2/0	11/3/0

表 4　在 14 个数据集上各个算法的 AUC 和标准差比较

dataset	Bagging	AdaBoostM1	NaïveBayes	RandomForest	RandomTree	RandomSubSpace	Stacking	Vote
CM1	0.77 ±0.11	0.72 ±0.10	0.77 ±0.10	0.72 ±0.13	0.57 ±0.10	0.76 ±0.11	0.79 ±0.11	0.80 ±0.11
JM1	0.69 ±0.02	0.58 ±0.03	0.69 ±0.02	0.72 ±0.02	0.59 ±0.02	0.69 ±0.02	0.72 ±0.02	0.73 ±0.02
KC1	0.79 ±0.04	0.79 ±0.04	0.79 ±0.04	0.80 ±0.05	0.62 ±0.06	0.79 ±0.03	0.81 ±0.04	0.81 ±0.04
KC2	0.85 ±0.06	0.68 ±0.06	0.84 ±0.06	0.80 ±0.07	0.62 ±0.11	0.85 ±0.06	0.84 ±0.06	0.83 ±0.06
KC3	0.82 ±0.08	0.75 ±0.10	0.83 ±0.08	0.80 ±0.11	0.61 ±0.11	0.82 ±0.08	0.80 ±0.11	0.80 ±0.11
KC4	0.76 ±0.14	0.71 ±0.14	0.78 ±0.13	0.80 ±0.12	0.70 ±0.13	0.77 ±0.13	0.83 ±0.11	0.83 ±0.12
MC1	0.92 ±0.04	0.88 ±0.05	0.90 ±0.05	0.88 ±0.09	0.78 ±0.10	0.92 ±0.04	0.92 ±0.08	0.95 ±0.04
MC2	0.72 ±0.13	0.73 ±0.14	0.72 ±0.12	0.70 ±0.15	0.58 ±0.13	0.72 ±0.13	0.76 ±0.14	0.76 ±0.14
MW1	0.77 ±0.14	0.75 ±0.14	0.76 ±0.14	0.72 ±0.15	0.58 ±0.13	0.77 ±0.14	0.77 ±0.13	0.76 ±0.14
PC1	0.76 ±0.08	0.72 ±0.07	0.75 ±0.08	0.81 ±0.09	0.67 ±0.08	0.74 ±0.08	0.82 ±0.10	0.85 ±0.07
PC2	0.84 ±0.14	0.73 ±0.17	0.82 ±0.15	0.64 ±0.15	0.51 ±0.07	0.87 ±0.11	0.78 ±0.20	0.87 ±0.12
PC3	0.77 ±0.07	0.76 ±0.07	0.76 ±0.07	0.81 ±0.06	0.64 ±0.06	0.76 ±0.07	0.83 ±0.05	0.82 ±0.06
PC4	0.84 ±0.05	0.80 ±0.05	0.84 ±0.05	0.92 ±0.03	0.71 ±0.07	0.83 ±0.05	0.93 ±0.03	0.92 ±0.03
PC5	0.84 ±0.03	0.83 ±0.03	0.83 ±0.03	0.95 ±0.02	0.73 ±0.04	0.94 ±0.01	0.96 ±0.01	0.96 ±0.01
MEAN	0.80 ±0.08	0.75 ±0.09	0.79 ±0.08	0.79 ±0.09	0.64 ±0.09	0.80 ±0.08	0.83 ±0.09	0.84 ±0.08

表5 14个数据集上8个分类算法的AUC值两两配对双尾T检验的结果

algorithm	Bagging	AdaBoostM1	NaïveBayes	RandomForest	RandomTree	RandomSubSpace	Stacking
AdaBoostM1	0/8/6						
NaïveBayes	0/12/2	4/10/0					
RandomForest	3/9/2	5/9/0	3/10/1				
RandomTree	0/1/13	0/4/10	0/1/13	0/0/14			
RandomSubSpace	1/11/2	6/8/0	1/12/1	1/10/3	12/2/0		
Stacking	5/9/0	9/5/0	6/8/0	1/13/0	14/0/0	6/8/0	
Vote	6/8/0	11/3/0	7/7/0	4/10/0	14/0/0	7/7/0	2/12/0

结果显示：

a)Vote相比于单一分类算法Naïve Bayes而言12胜2平，从统计学意义上来讲Vote具有压倒性的优势。而相比RandomForest而言Vote得到了1胜12平1负的结果，从统计学意义上来讲Vote与RandomForest没有显著差异。而对比其他的集成技术，Vote却具有显著的优势，对于Vote性能的分析结论与通过表2得到的结论相吻合。

b)RandomForest对比除Vote的其他各算法的检验结果显示，其在大多数数据集上的AC均值具有显著优势，这个结论与表2得到的关于RandomForest的结论相吻合。

c)Bagging相较于Naïve Bayes不但没有任何胜出，而且在一个数据集上的AC均值还显著低于Naïve Bayes。AdaBoostM1相较于Naïve Bayes仅在一个数据集上的AC均值显著高于Naïve Bayes，除此之外没有显著性差异。

AUC是本文采取的另外一个准确性评价标准，在14个数据集上各个算法的AUC和标准差比较由表4显示。可以发现，Vote在9个数据集上以及平均AUC值上都取得了最高值，因此Vote性能表现最好的事实在AUC指标上同样得到了验证。但是不难看出，在AC值及其假设检验中表现很好的RandomForest，在AUC指标上却没有脱颖而出，基本与除Stacking和Vote以外的算法相近，仅高于RandomTree。而从AUC指标上本文发现了另外一个表现出较好性能的算法——Stacking，其在7个数据集上取得了最高值，而且平均AUC值也与Vote最为接近，这证明了Stacking相对于除Vote外其他算法具有更好的泛化能力。

表5给出了每个分类算法在每个数据集上得到的100个AUC值进行两两配对的双尾T检验结果。可以看出，Vote依然具有显著性优势，而Stacking的检验结果相比于除Vote的其他算法也具有明显的优势，其次表现最好的算法是RandomForest。这些结论同样与表4的相关结论相吻合。

4 结束语

本文在单一分类模型进行缺陷预测出现性能瓶颈的背景下，对七种常见的集成分类算法进行了分析，并给出了它们构建缺陷预测模型时的基本策略或算法。重点对前述七种集成算法和单一分类算法Naïve Bayes进行比较实验；从准确性指标、泛化能力以及统计学意义的性能差异显著性三个方面进行了深入分析。三个方面的实验结果都显示：投票(Vote)不但具有远远超过Naïve Bayes的性能，而且在全部七种分类器中也具有明显的性能优势和泛化能力；随机森林的预测性能仅次于Vote，但其泛化能力不及Stacking。在利用集成分类算法构造缺陷预测模型时，本文的实验证明了Vote应该是一个很好的选择。

集成分类算法的特点在于组合不同的单一分类器，并保证分类器之间的多样性。下一步的工作主要包括，在利用Vote构造集成模型时，尝试组合除本文采用的四种模型之外的算法，并分析其预测性能；进一步研究如何保证基分类器多样性等问题。

参考文献：

[1] MENZIES T, MILTON Z, TURHAN B, *et al*. Defect prediction from static code features: current results, limitations, new approaches[J]. Automated Software Engineering, 2010, 17(5): 375-407.

[2] JIANG Y, CUKIC B, MENZIES T, *et al*. Comparing design and code metrics for software quality prediction[C]//Proc of the 4th International Workshop on Predictor Models in Software Engineering. New York: ACM Press, 2008: 11-18.

[3] MENZIES T, TURHAN B, BENER A, *et al*. Implications of ceiling effects in defect predictors[C]// Proc of ACM International Conference on Predictive Models in Software Engineering. 2008: 47-54.

[4] ZHANG H, NELSON A, MENZIES T. On the value of learning from defect dense components for software defect prediction[C]// Proc of ACM International Conference on Predictive Models in Software Engineering. 2010: 1-9.

[5] STEFANO C D, FONTANELLA F, FOLINO G, *et al*. A Bayesian approach for combining ensembles of GP classifiers[C]// Proc of the 10th International Workshop on Multiple Classifier Systems. 2011: 26-35.

[6] TOSUN A, TURHAN B, BERNER A B. Ensemble of software defect predictors: a case study[C]//Proc of the 2nd International Symposium on Empirical Software Engineering and Measurement. 2008: 318-320.

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

[8] BREIMAN L. Bagging predictors[J]. Machine Learning, 1996, 24(2): 123-140.

[9] FREUND Y, SCHAPIRE R. Experiments with a new boosting algorithm[C]// Proc of the 13th International Conference on Machine Learning. 1996: 148-156.

[10] DIETTERICH T. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization[J]. Machine Learning, 2000, 40(2): 139-157.

[11] BREIMAN L. Random forests[J]. Machine Learning, 2001, 45(1): 5-32.

[12] HO T K. The random subspace method for constructing decision forests[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1998, 20(8): 832-844.

[13] WOLPERT D H. Stacked generalization[J]. Neural Networks, 1992, 5(2): 241-259.

[14] KUNCHEVA L I. Combining pattern classifiers: methods and algorithms[M]. 1st ed. New Jersey: Wiley, 2004.

[15] CHAPMAN M, CALLIS P, JACKSON W. Metrics data program [EB/OL]. http://mdp.ivv.nasa.gov/.

[16] WITRREN H, FRANK E. Data mining practical machine learning tools and techniques[M]. 2nd ed. San Francisco: Morgan Kaufmann Publisher, 2005.

[17] HALL M, FRANK E, HOLMES G, *et al*. The WEKA data mining software: an update[J]. ACM SIGKDD Explorations Newsletter, 2009, 11(1): 10-18.