一种面向软件缺陷预测的可容忍噪声的特征选择框架

刘望舒[1),2)] 陈 翔[1),3)] 顾 庆[1),2)] 刘树龙[1),2)] 陈道蓄[1),2)]

[1)](南京大学计算机软件新技术国家重点实验室 南京 210023)

[2)](南京大学计算机科学与技术系 南京 210023)

[3)](南通大学计算机科学与技术学院 江苏 南通 226019)

摘 要 软件缺陷预测通过挖掘软件历史仓库,构建缺陷预测模型来预测出被测项目内的潜在缺陷程序模块.但在挖掘过程中,对程序模块进行类型标记或软件度量时均可能产生噪声.虽然研究人员对已有特征选择方法的噪声容忍能力进行了分析,但据我们所知,很少有研究人员在软件缺陷预测研究中,针对性的设计出可容忍噪声的新颖特征选择方法.为了解决此问题,我们提出一种可容忍噪声的特征选择框架FECS.具体来说,首先借助聚类分析,将原始特征集划分到指定数目的簇中,随后设计出3种不同的启发式特征选择策略,依次从每一个簇中选出最为典型的特征.在实证研究中,以Eclipse和NASA等实际项目为评测对象.首先借助一系列数据预处理方法来提升数据集质量,随后同时注入类标噪声和特征噪声来模拟噪声数据集.通过与典型的特征选择方法进行比较,验证了FECS框架的有效性,除此之外,通过深入分析噪声注入率、特征选择比例及噪声类型对缺陷预测性能的影响,为更有效的使用FECS提供了指导.

关键词 软件质量保证;软件缺陷预测;特征选择;噪声容忍能力;聚类分析

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2018.00506

A Noise Tolerable Feature Selection Framework for Software Defect Prediction

LIU Wang-Shu[1),2)] CHEN Xiang[1),3)] GU Qing[1),2)] LIU Shu-Long[1),2)] CHEN Dao-Xu[1),2)]

[1)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

[2)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023)

[3)](*School of Computer Science and Technology, Nantong University, Nantong, Jiangsu* 226019)

Abstract Software defect prediction constructs a software defect prediction model based on the mining of software historical repositories. Then it uses the trained model to predict potential defect-proneness program modules. However noises are inevitable when labeling or measuring the software entities. Although some researchers have investigated the noise tolerance of existing feature selection methods, few studies focus on proposing new feature selection methods with a certain noise tolerance. To solve this issue, we propose a novel framework FECS (FEature Clustering with Selection strategies). In particular, FECS first cluster original features into specified number of clusters based on cluster analysis. Then it selects a most typical feature from each cluster based on our proposed three heuristic feature selection strategies. During empirical studies, we choose real-world software projects, such as Eclipse and NASA. We first perform a set of data preprocessing steps to improve the quality of these datasets. We then inject class level and feature level noises simultaneously to imitate noisy datasets. After using classical feature

收稿日期:2015-07-30;在线出版日期:2016-02-17.本课题得到国家自然科学基金(61373012,61202006,61321491,91218302)、欧盟项目(612212)、南京大学计算机软件新技术国家重点实验室开放课题(KFKT2016B18)、南京大学优秀博士研究生创新能力提升计划和软件新技术与产业化协同创新中心部分资助.**刘望舒**,男,1987年生,博士研究生,主要研究方向为软件缺陷预测.E-mail:liuws0707@gmail.com.**陈 翔**,男,1980年生,博士,副教授,硕士生导师,主要研究方向为软件缺陷预测、回归测试、缺陷定位和组合测试.**顾 庆**(通信作者),男,1972年生,博士,教授,博士生导师,主要研究领域为软件质量保障和分布式计算.**刘树龙**,男,1990年生,硕士,主要研究方向为软件缺陷预测.**陈道蓄**,男,1942年生,教授,博士生导师,主要研究领域为分布式计算和并行处理.

selection methods as the baseline, we confirm the effectiveness of FECS and provide a guideline of using FECS after analyzing the effects of varying either percentage of selected features or the noise injection rates, and different noise types.

Keywords　software quality assurance; software defect prediction; feature selection; noise tolerable ability; cluster analysis

1　引　言

软件缺陷预测(software defect prediction)[1-3]是当前软件工程数据挖掘领域[4]中的一个研究热点.首先通过分析软件代码或开发过程,设计出与软件缺陷相关的度量元(metric)[5],随后通过挖掘软件历史仓库来创建缺陷预测数据集.最后基于上述搜集的缺陷预测数据集,构建缺陷预测模型,并用于预测出被测项目内的潜在缺陷程序模块.

但在挖掘软件历史仓库时,在对程序模块进行类型标记或软件度量时均有可能产生噪声,这些噪声的存在会影响到缺陷预测模型的构建.虽然一些研究人员对已有特征选择方法的噪声容忍能力进行了分析[6-7],但据我们所知,很少有研究人员去有针对性地设计出具有一定噪声容忍能力的特征选择方法.

在我们前期研究工作中,提出了一种新颖的基于聚类分析的特征选择框架 FECAR[8].FECAR 首先根据特征间的特征相关度将特征进行聚类分析,这样可以将具有冗余关系的特征集中于同一聚类中.随后对于每一个聚类,根据特征与类间的类相关度从高到低进行排序,并从中选出指定数量的特征,该阶段可以有效避免选出无关特征.本文对该框架进行了拓展,并设计出了一种具有噪声容忍能力的特征选择框架 FECS(FEature Clustering with Selection strategies).FECS 重点对特征选择阶段进行了扩展.具体来说,我们设计出了 3 种不同的启发式特征选择策略,这些策略基于类相关度(FC-Relevance)和特征相关度(FF-Correlation)来从每一个聚类中选出一个最为重要的特征.在实证研究中,为评估 FECS 方法的噪声容忍能力,我们选择了 Eclipse 和 NASA 实际项目作为评测对象,首先通过一系列数据预处理方法来确保数据集的质量.随后我们同时注入类标噪声(C-Noise)和特征噪声(F-Noise)来模拟含有噪声的数据集.最后我们通过将 FECS 与一些经典特征选择方法(IG、CFS 和 Consist 等)进行比较,验证了 FECS 的有效性.

本文的主要贡献可总结如下:(1)针对软件缺陷预测问题,提出了一种具有噪声容忍能力的特征选择框架 FECS.其中在第 1 个阶段,我们提出了一种基于 k-medoids 的特征聚类方法.随后在第 2 个阶段,我们设计出了 3 种不同的特征选择策略 FR、MR 和 LE;(2)通过来自实际项目的实证研究,验证了 FECS 框架的有效性并为有效的使用 FECS 提供了相应指导.

本文第 2 节介绍研究背景和相关工作;第 3 节介绍 FECS 的实现细节;第 4 节依次介绍需要回答的实验问题、数据集特征、噪声注入方式、评测指标和实验细节;第 5 节对实证研究的结果进行分析和讨论,并对影响实证研究有效性的影响因素进行分析;最后总结全文并对未来研究工作进行展望.

2　研究背景和相关工作

2.1　软件缺陷预测

软件缺陷产生于开发人员的编码过程.含有缺陷的软件在运行时可能会产生意料之外的结果或行为,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.在软件项目的开发生命周期中,检测出内在缺陷的时间越晚,修复该缺陷的代价也越高.因此项目主管借助软件测试或代码审查等软件质量保障手段,希望能够在软件部署前尽可能多地检测出内在缺陷.但是若关注所有的程序模块会消耗大量的人力物力,因此项目主管希望能够预先识别出需要重点关注的程序模块,并对其分配足够的测试资源.软件缺陷预测[1-3]是一种可行方法,具体来说,通过分析软件代码或开发过程,设计出与软件缺陷相关的度量元[4],随后通过挖掘软件历史仓库来创建缺陷预测数据集.目前可以挖掘与分析的软件历史仓库包括项目所处的版本控制系统(例如 CVS、SVN 或 Git)或缺陷跟踪系统(例如 Bugzilla、Mantis、Jira 或 Trac)等.最后基于上述搜集的缺陷预测数据集,构建缺陷预测模型,并用于预测出项目内的潜在缺陷程序模块.

软件缺陷预测可以将程序模块的缺陷倾向性、缺陷密度或缺陷数设置为预测目标.以预测模块的缺陷倾向性为例,其典型的模型构造和预测过程如图 1 所示.

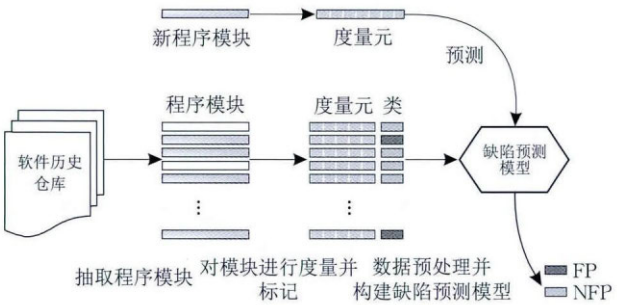


图 1 以缺陷倾向性为预测目标的软件缺陷预测过程

该过程主要包括 4 个阶段:(1)挖掘软件历史仓库,从中抽取出程序模块.程序模块的粒度根据实际应用的场景,可设置为文件、包、类或函数等.随后将该程序模块标记为有缺陷模块或无缺陷模块,在图 1 中,我们将有缺陷模块用红色进行标记,无缺陷模块用绿色进行标记;(2)通过分析软件代码或开发过程设计出与软件缺陷存在相关性的度量元,通过这些度量元对程序模块进行软件度量并构建出缺陷预测数据集;(3)对缺陷预测数据集进行必要的数据预处理后,借助特定的建模方法构建出缺陷预测模型;(4)当面对新程序模块时,在完成对该模块的软件度量后,基于缺陷预测模型和度量元取值,可以完成对该模块的分类,即将该模块预测为有缺陷倾向性(Defect-Proneness,简称 FP)模块或无缺陷倾向性(Non Defect-Proneness,简称 NFP)模块.

2.2 特征选择方法在软件缺陷预测中的应用

缺陷预测数据集中若含有大量特征(度量元),会使得数据集存在维数灾难问题.因此需要提出有效方法来重点识别出数据集中的冗余特征和无关特征.其中冗余特征大量或完全重复了其他单个或多个特征中含有的信息.而无关特征则对采用的数据挖掘算法不能提供任何帮助.研究表明无关特征和冗余特征的存在会增加缺陷预测模型的构建时间,并会降低模型的预测性能.

特征选择是解决维数灾难问题的一种有效方法,通过识别并移除数据集中的冗余特征和无关特征,来确保选出最为有用的特征子集来构建缺陷预测模型.目前已有的特征选择方法可简单分为两类:包装(wrapper)法和过滤(filter)法.其中包装法借助预先指定的学习算法的预测精度来决定选择出的特征子集.因此虽然精度较高,但计算开销较大.过滤法则通过分析数据集来完成特征的选择,因此与选择的学习算法无关,具有更好的通用性且计算开销较小,但精确度较低.

Menzies 等人[9]和 Song 等人[10]提出的通用缺陷预测框架中均包含了基于包装法的特征选择方法.Gao 等人[11]针对一个大规模遗留电信软件系统,考虑了一种混合特征选择方法,在该方法中考虑了不同的特征排序评估方法和特征子集评估方法.我们也基于过滤法,提出了一种新颖的基于聚类分析的特征选择框架 FECAR[8].FECAR 首先根据特征间的特征相关度将特征进行聚类分析,这样可以将具有冗余关系的特征集中于同一聚类中.随后对于每一个聚类,根据特征与类间的类相关度从高到低进行排序,并从中选出指定数量的特征,该阶段可以有效避免选出无关特征.

2.3 缺陷预测数据集中的噪声问题

在挖掘软件历史存档时,在对程序模块进行类型标记和软件度量时均可能产生噪声,这些噪声的存在会影响到缺陷预测模型的构建,并会对已有实证研究结论的有效性产生严重影响.

以程序模块类型标记为例,首先,将版本控制系统中的修改日志与缺陷跟踪系统中的缺陷报告建立链接关系时容易产生噪声.例如通过在修改日志中搜索特定关键词(例如 fixed 或 bug)和缺陷 ID 号(例如 #53322)可以去建立这种链接关系,随后基于该链接关系将代码修改涉及的程序模块标记为有缺陷模块.但在实际的软件开发过程中,开发人员在修复完缺陷并提交代码修改时,经常会忘记在修改日志中输入这次修复的缺陷 ID 号,从而造成一些链接关系的遗漏,这种遗漏会造成一些有缺陷模块被误标记为无缺陷模块.我们称这种噪声为假阴性(false negative)噪声.Bird 和 Bachmann 等人[12-13]通过分析一些开源项目,发现大约 54% 的已修复缺陷并未与对应的修改日志建立链接关系.Nguyen 等人[14]同样在一个商业项目中也发现了上述问题.随后 Wu 等人[15]和 Nguyen 等人[16]分别提出了 ReLink 方法和 MLink 方法来尝试重新发现这些遗漏的链接关系.Rahman 等人[17]则从另一个角度出发,他们深入分析了缺陷预测数据集的假阴性噪声和规模对缺陷预测模型性能的影响.他们认为可以通过提高数据集的规模来缓解噪声问题的影响.

其次在缺陷跟踪系统中,如果问题报告(issue

report)存在错误分类也会产生噪声. Herzig 等人[18]手工检查了来自五个开源项目的 7000 多份问题报告,他们发现大部分的问题报告均被分类为缺陷报告(为了修复缺陷). 但深入分析之后,他们发现其中 33.8%的问题报告存在误分类问题. 即其提交的问题报告不是为了修复缺陷,而是为程序模块添加新的特征、修改相应文档或重构内部代码等. 这些被误分类的问题报告会造成一些无缺陷模块被误标记为有缺陷模块. 我们称这种噪声为假阳性(false positive)噪声. 他们提出了一套分类指南,可以有效提高对这些问题报告的分类准确率.

通过上述分析,不难看出缺陷数据集中的噪声问题难以避免. Kim 等人[6]采用随机方式,手工注入类标噪声,并对已有缺陷预测方法的噪声抗干扰能力进行了分析. 他们在实证研究中发现:(1)当数据集中的缺陷模块足够多时,增加噪声并不会显著降低已有方法的预测性能;(2)已有方法对假阴性噪声的抗干扰能力更强一些;(3)当数据集中包含的假阳性和假阴性噪声的总比例不超过 20%～35%时,已有方法的性能不会显著下降. Tantithamthavorn 等人[7]认为 Kim 等人采用随机方式来为数据集注入噪声并不合理,他们基于 Herzig 等人[18]的研究工作,发现因问题报告误分类产生的噪声并不是随机出现的. 同时不同的噪声注入方式对最终的实证研究结果存在显著的影响.

与上述研究工作不同,我们对前期研究工作[8]进行了扩展,其扩展之处包括:(1)在特性选择阶段,设计出了 3 种不同的启发式策略,旨在具有一定的噪声容忍能力;(2)设计出噪声注入机制,通过同时注入类标噪声 C-Noise 和特征噪声 F-Noise 来模拟噪声数据集. 具体来说,通过随机修改实例的标记来注入 C-Noise,借助 Gaussian 噪声来注入 F-Noise. 基于上述噪声数据集,系统的验证了论文所提方法的噪声容忍能力;(3)分别从性能评测指标 AUC 和稳定性评测指标 RLA 来验证论文所提方法的有效性,同时深入分析了特征选择比例和噪声注入率对结果的影响.

3　特征选择框架 FECS

本节将给出论文所提的特征选择框架 FECS 的实现细节. 首先,对 FECS 框架中使用的类相关度和特征相关度进行定义. 随后描述方法的整体框架结构以及特征选择阶段中的 3 种不同的启发式特征选择策略.

3.1　特征关联性分析

论文依次对类相关度和特征相关度做如下定义.

定义 1.　类相关度(FC-Relevance). 对于给定的特征 f_i 和类标,用 FC-Relevance 定义它们之间的类相关度,并借助类相关度度量公式计算出特征与类标间的关联性强度.

目前存在多种用于计算类相关度的度量公式[19-20]. 常用的度量公式可以分为 3 类:第 1 类基于统计理论,包括卡方值(Chi-Square)等;第 2 类基于信息熵,包括信息增益(information gain)、对称不确定性(symmetric uncertainty)等;还有一类基于实例,包括 Relief 和 ReliefF 等. 已有研究工作[8,21]表明:在软件缺陷预测领域中,基于信息增益的度量公式在数据集内含有噪声的情况下,仍然可以取得较好的预测性能. 因此,论文采用信息增益作为 FECS 框架的类相关度度量公式. 其具体公式为

$$IG(X|Y)=H(X)+\sum_{y\in Y}p(y)\sum_{x\in X}p(x|y)\log_2 p(x|y) \quad (1)$$

在式(1)中,$H(X)$用于计算随机变量 X 的信息熵,$p(y)$表示变量 Y 取值为 y 的先验概率,$p(x|y)$表示当变量 Y 取值为 y,变量 X 取值为 x 的后验概率. 其中信息熵 $H(X)$的计算公式为

$$H(X)=-\sum_{x\in X}p(x)\log_2 p(x) \quad (2)$$

定义 2.　特征相关度(FF-Correlation). 对于任意两个特征 f_i 和 f_j($i\neq j$),用 FF-Correlation 定义它们之间的相关度,并借助特征相关度度量公式计算出两个特征间的关联性强度.

目前同样存在多种特征相关度度量公式,可以计算出两个特征间的相关度[19-21]. Yu 和 Liu 发现对于软件度量特征来说,非线性的相关度度量方法比线性方法更为实际有效. 同样,我们的前期研究工作也表明:非线性的对称不确定性(Symmetric Uncertainty,简称 SU)在软件缺陷预测中能取得足够好的预测性能. 因此论文采用 SU 作为 FECS 框架的特征相关度度量公式,其计算公式为

$$SU(X,Y)=2\times\frac{IG(X|Y)}{H(X)+H(Y)} \quad (3)$$

其中,$IG(X|Y)$和 $H(X)$可通过式(1)和式(2)来计算.

3.2　FECS 框架的总体结构

本文提出的 FECS 框架如图 2 所示,具体来说,在按照 3.1 节对特征集进行关联性分析后,FECS

分为 2 个阶段. 在第 1 阶段,借助聚类分析,基于 FF-Correlation 将原始特征集划分到指定数目的簇中. 在第 2 阶段,基于类相关度 FC-Relevance 和特征相关度 FF-Correlation 设计出 3 种不同的启发式特征选择策略,依次从每一个簇中选中最有价值的特征.

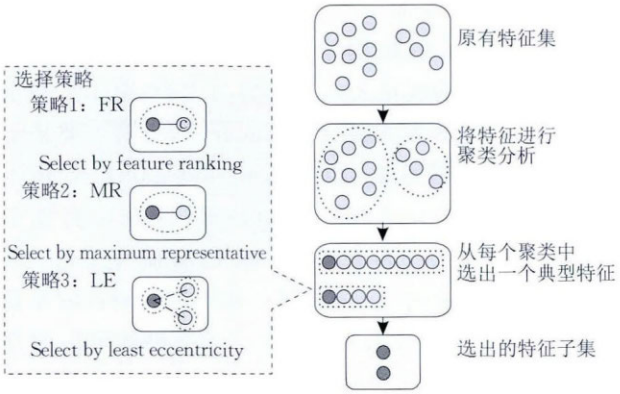


图 2 FECS 框架的总体结构

3.2.1 特征聚类阶段

在该阶段,借助 k-medoids 聚类方法. 该方法首先基于 FC-Relevance 对特征进行排序,并选取前 k 个特征作为初始中心点. 接着,基于 FF-Correlation 将剩余的特征划分到最近的中心点所在的簇中. 之后,计算簇中每个特征与其他特征之间的特征相关度之和,选取最大的相关度之和来更新簇中心. 重复上述过程,直至方法收敛为止. 具体的算法细节可以参考我们的前期研究工作[8]. 相对于最常用的 k-means 聚类方法,k-medoids 选取的中心点都是簇内的实际数据点,而不是簇中数据均值,因此尽管存在噪声也不会对中心点的选取产生严重的影响. 另一方面,在算法第一次计算出所有特征相关度后,每次更新簇中心都不需要再次计算各个特征之间的特征相关度,而是可以利用第一次的计算结果进行簇中心的更新运算,相比于 k-means 可以大幅度减少计算开销. 因此论文在特征聚类阶段选取了更适用于构建鲁棒特征选择算法的 k-medoids 聚类方法.

3.2.2 特征选择阶段

在该阶段,基于第 1 阶段的聚类分析结果,依次从每一个簇中选出一个最有代表性的特征来构成最终选出的特征子集. 在前期研究工作[8]中,我们可能会从每个簇中选出不止一个特征. 但是,由于数据集中可能含有 C-Noise 和 F-Noise,这种策略可能无法适用于存在噪声的数据集中. 除此之外,我们发现若基于 FF-Correlation,有时候也有助于选出一些具有噪声容忍能力的特征. 我们将在 3.3 节中具体说明框架内考虑的 3 种不同的启发式特征选择策略.

3.3 3 种启发式特征选择策略

完成聚类分析后,我们在特征选择阶段设计了 3 种不同的启发式策略,可以依次从每个簇中选出有助于容忍噪声的最典型特征. 3 种策略的描述如下:

策略 1(FR). 选择类相关度最大的. 该策略在每个簇中,计算各个特征与类标的类相关度 FC-Relevance,选择该值最大的一个作为最终选择的特征.

策略 2(MR). 选择簇中最具代表的. 该策略在每个簇中,计算各个特征与其他所有特征的特征相关度 FF-Correlation 之和,选择该值最大的一个(簇中心)作为最终选择的特征.

策略 3(LE). 选择与其他簇关联性最低的. 该策略在每个簇中,计算各个特征与其他簇中心的特征相关度 FF-Correlation 之和,选择该值最小的一个作为最终选择的特征.

除了从各个簇中仅选出一个特征,策略 1 和我们前期工作[8]类似. 该策略也被其他研究人员[22-23]用于特征选择. 但是,由于数据集中可能会存在 C-Noise,因此该策略基于 FC-Relevance 所选出的特征容易受到 C-Noise 的影响.

而在设计策略 2 和策略 3 时,因没有考虑 FC-Relevance,所以能有效缓解 C-Noise 带来的影响. 尤其在策略 2 中,该策略选出的是每个簇中最具代表的簇中心特征. 文献[24]表明存在噪声的软件度量特征与正常特征之间的相关度很低. 所以,簇中心特征受到 F-Noise 的影响也会较小. 与策略 2 不同,在不含有噪声的数据集中,策略 3 可能表现较好,主要因为该策略会选出特征之间冗余度最小的特征. 不过在存在 F-Noise 的数据集上,策略 3 只考虑特征与其他簇中心之间的特征相关度,因此簇内则可能选取与簇内其他特征相关度较低的噪声特征,所以容易受到 F-Noise 的影响.

3.4 算法描述及分析

本节对 FECS 的算法进行总结. FECS 的输入是原有特征集、指定的 FC-Relevance 和 FF-Correlation 度量方法、启发式特征选择策略以及聚类分析时需要生成的簇的数量. 输出是选出的特征子集. 算法使用 SU 来度量两个特征之间的特征相关度,使用 IG 来度量特征与类标之间的类相关度. 具体算法如算法 1 所示.

算法 1.　FECS 框架.

输入：原有特征集 F，FC-Relevance 的度量方法 IG，FF-Correlation 的度量方法 SU，簇数量 k，特征选择策略 s

输出：选出的特征子集 *Subset*

1. 特征集关联性分析

for each pair of *features* (f_i, f_j) in F do

　　借助 SU 计算特征 f_i 与特征 f_j 之间的特征相关度

endfor

构造出矩阵 $\boldsymbol{M}$，其中 $M_{i,j}$ 表示 $SU(f_i, f_j)$

for $i=1$ to m do

　　借助 IG 计算特征与类标之间的类相关度，并构造向量 $\boldsymbol{V}$，其中 V_i 表示 $IG(f_i, label)$

endfor

2. 特征聚类阶段

根据矩阵 $\boldsymbol{M}$ 和向量 $\boldsymbol{V}$，将原有的特征集划分为 k 个簇

3. 特征选择阶段

Subset $\leftarrow \varnothing$

for $i=1$ to k do

　　根据策略 s 从簇 C_i 中选出一个典型特征，并添加到 *Subset*

endfor

return *Subset*

随后分析该算法的时间复杂度. FECS 的时间消耗主要包括三个部分，即特征集关联性分析和特征聚类、簇内特征选择两个阶段. 假设数据集中的实例数量是 n、数据集的维度是 m、FECS 聚类后得到 k 个簇. 通常来说，实例数量 n 要远大于特征数量 m，而 m 又要大于聚类后的簇数量 k. 相对于 n，可以将 m 和 k 视做常数.

首先给出计算特征集关联性分析的时间复杂度. 该部分包括(1)类相关度计算：对于信息增益，计算一个特征的类相关度是线性的，所以在含有 n 个实例的数据集上计算一个特征的类相关度的时间复杂度是 $O(n)$. 总共有 m 个特征，计算它们类相关度的时间复杂度是 $O(nm)\approx O(n)$；(2)特征相关度计算：由 SU 的计算公式可知，计算两个变量 SU 的时间复杂度也是线性的，即时间复杂度是 $O(n)$. 总共有 m 个特征，那么计算特征相关度的时间复杂度是 $O(nm^2)\approx O(n)$.

接着给出 FECS 两阶段的时间复杂度分析：

第 1 阶段(特征聚类)：3.2.1 节已经说明特征聚类主要包括计算特征与各个簇的中心点距离和更新每个簇的中心点两个步骤. 对每一个特征，计算步骤 1 的过程就是从 k 个簇中心特征中找出与其最近邻的中心点，因此时间复杂度是 $O(k)$，又因为总共有 m 个特征，所以这一步总的时间复杂度是 $O(mk)$. 步骤 2 在更新簇内的簇中心时，需要计算簇内每个特征与簇内其他特征的特征相关度之和，而每个簇的期望大小是 m/k，所以更新簇中心特征的时间复杂度是 $O(k(m/k)^2)=O(m^2k)$.

第 2 阶段(特征选择)：可以使用快速排序算法对簇内的特征按照不同策略根据类相关度或者特征相关度进行排序选择，该时间复杂度是 $O(k(m/k)\log_2(m/k))=O(m\log_2(m/k))$.

因此 FECS 最终的时间复杂度是 $[O(n)+O(n)]+[O(mk)+O(m^2k)]+[O(m\log_2(m/k))]<O(n\log_2 n)$.

4　实验设计

本节对 FECS 的有效性进行实证研究. 下面是实证研究中需要回答的 3 个实验问题：

RQ1. 相对于经典的特征选择方法，FECS 是否在无噪声的数据集上能够提高缺陷预测模型的性能？

RQ2. 当数据集同时存在类标噪声 C-Noise 和特征噪声 F-Noise 的情况下，相比于其他经典特征选择方法，其预测模型的性能损失是增加还是减少？

RQ3. 对于 FECS 而言，噪声注入率、特征选择比例以及噪声类型是如何对预测模型性能产生影响的？

前两个问题的设计主要在没有噪声和存在噪声的数据集上，探讨 FECS 的预测性能及稳定性. 而第三个问题的设计，则是期望为更有效地使用 FECS 提供相应指导.

4.1　数据集

本文选择了来自实际软件项目的数据集来进行实验研究，包括来自 Eclipse 项目的 3 个数据集和来自 NASA 项目的 10 个数据集(如表 1 所示). 其中

表 1　测评数据集具体信息

数据集	程序模块粒度	特征数	实例数	缺陷实例数(所占比例)
Eclipse2.0	类	155	6729	975(14.49%)
Eclipse2.1	类	155	7888	854(10.83%)
Eclipse3.0	类	155	10593	1568(14.80%)
cm1	函数	37	505	48(9.50%)
kc1	类	86	145	60(41.37%)
kc3	方法	39	458	43(9.39%)
kc4	方法	43	125	61(48.80%)
mc2	函数	39	161	52(32.30%)
mw1	函数	37	403	31(7.69%)
pc1	函数	37	1107	76(6.87%)
pc3	函数	37	1563	160(10.24%)
pc4	函数	37	1458	178(12.21%)
pc5	函数	39	17186	516(3.00%)

Eclipse 项目的数据集可以从 Promise 库中下载获得,而 NASA 项目的数据集可以从 MDP 数据库中获取.所有这些数据集都已公开,并且被广泛用于软件缺陷预测研究领域[6,8-10,12].

为了尽可能的减少数据集内的噪声,论文对下载的数据集执行了如下的数据预处理:(1)移除所有非数值型及取值完全相同的特征;(2)原有数据集的类标表示的是软件缺陷数,而本文处理的是一个二元分类问题,因此我们将缺陷数大于 0 的模块标记为有缺陷模块,其他模块则标记为无缺陷模块;(3)最后依据 Hulse 和 Khoshgoftaar 推荐的方法[25]来识别并移除存在异常特征取值的模块实例.通过上述数据预处理,我们假设所得到的数据集不存在噪声问题.

4.2 噪声注入

本文主要考虑两种类型的数据噪声,分别是类标噪声 C-Noise 和特征噪声 F-Noise,下面依次定义如下.

定义 3. 类标噪声(C-Noise).当数据集中的实例(程序模块)存在类标错误时,则称该类噪声为类标噪声.

定义 4. 特征噪声(F-Noise).当数据集中的实例的某些特征值出现丢失或者度量有误时,则称该类噪声为特征噪声.

在缺陷预测数据集搜集过程中,这两种噪声通常会同时出现.因此,本文在噪声注入时同时考虑了 C-Noise 和 F-Noise,并通过变量 ω 来表示注入的噪声率.下面是论文设计的噪声注入过程:

(1)对于 C-Noise 的注入,先随机选取 $\omega\times\#instances$ 个实例,$\#instances$ 表示数据集中含有的实例数.然后翻转所选实例的类标(将没有缺陷的标记修改为有缺陷的标记,或者相反).

(2)对于特征噪声 F-Noise 的注入,使用了文献[26-27]推荐的高斯噪声(Gaussian noise).具体来说,每次随机选取一个实例,然后再随机选取该实例的一个特征.将服从高斯分布 $N(\mu,\sigma^2)$ 的随机噪声赋值给该特征,其中,$\mu=0$,$\sigma^2=(max-min)/6$,max 和 min 分别表示所有实例在该特征上取值的最大值和最小值.此过程会重复 $\omega\times\#instances\times\#features$ 次.

在噪声率 ω 的取值设置上,本文参考了 Herzig 等人[18]的结论:在缺陷预测数据集中 33.8%的数据会遭到噪声污染.因此,在本文实验中,对 C-Noise 和 F-Noise 的噪声率 ω 均设置为 33.8%.例如,数据集 cm1 包含 505 个实例和 37 个特征.在噪声率设置为 33.8%的情况下,170 个实例的类标会被翻转,6315 个特征值将会被注入高斯噪声.在 RQ3 中,我们将进一步分析不同的噪声率 ω 对 FECS 在缺陷预测模型性能上的影响.

最后需要注意的是,本文仅对训练集上的数据注入噪声,并假设测试集上的数据不存在噪声.这种约定可以直观地比较出注入噪声前后各种特征方法的性能优劣,从而更公平地去评价 FECS 是否能提高预测模型的容错能力.

4.3 评测指标

本文采用两种评测指标 AUC[28] 和 RLA[26] 来评价特征选择方法的性能.其中 AUC 指标用来度量模型的预测性能,而 RLA 指标则度量在噪声数据集中,模型预测性能的稳定度.

AUC(Area Under ROC Curve)[28] 基于 ROC 曲线来评估不同缺陷预测模型的预测性能.其中 ROC 曲线[29] 在评估分类器的时候,综合考虑了不同的分类阈值.在 ROC 曲线中,横坐标表示 tpr(true positive rate)值,纵坐标表示 fpr(false positive rate)值,对每一个分类阈值,分类器都有对应的 tpr 值和 fpr 值(对应坐标系上的一个坐标点).将所有坐标点连接起来就是该分类器对应的 ROC 曲线.而 AUC 指标则对应的是 ROC 曲线下的面积,其取值范围是[0,1],越接近于 1,则代表对应的分类器性能越好.AUC 指标还特别适用于具有类不平衡问题的数据集[30],目前 AUC 指标已经被很多研究人员用于测评缺陷预测模型的性能[9,28].

RLA(Relative Loss of Accuracy)[26] 在噪声数据集上,可评估不同缺陷预测模型的预测性能的损失情况.与文献[31]不同,论文中 RLA 主要考虑 AUC 值的损失情况,而不是准确率(accuracy)的损失情况.因此,RLA 指标的计算公式是

$$RLA_{x\%}=\frac{AUC_0-AUC_{x\%}}{AUC_0} \tag{4}$$

在式(4)中,$AUC_{x\%}$ 表示 $x\%$ 的噪声(既包括 C-Noise 又包括 F-Noise)被注入后的 AUC 值,而 AUC_0 表示在数据集没有注入噪声时的 AUC 值.

4.4 实验细节

为了评估 FECS 框架的有效性,本文基于 3.3 节的 3 种策略,实现了 3 种不同的特征选择方法.其中 FECS_FR 方法在特性选择阶段使用策略 1,FECS_MR 方法使用策略 2,而 FECS_LE 方法使用策略 3.并且选取 3 种经典的特征选择算法(IG、

CFS 和 Consist)作为对比方法.除此之外,还选择了 ALL 方法作为对比方法,该方法并不使用特征选择方法(保留所有特征).在分类器选择方面,选取了朴素贝叶斯(NB)和决策树(C4.5)作为缺陷预测模型的构建方法.图 3 是本文实验配置的整体框架图.

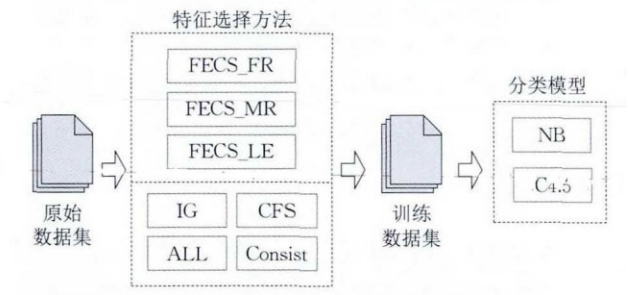


图 3　实验配置的框架图

为评估缺陷预测模型的性能,我们采用 10×10 折交叉验证方式[6].具体来说:10 折交叉验证(10-fold cross validation)是评估分类方法性能的一种常用方法.即将数据集划分为 10 份,轮流将其中的 9 份作为训练数据,剩余 1 份作为测试数据.上述过程重复 10 次(确保每个实例都被预测过一次),并最终取这 10 次运行结果的平均值.为了避免数据集中实例次序对结果的影响,在实验中我们进一步重复 10 折交叉验证 10 次,每次执行前将数据集中的实例随机打乱.论文将上述验证方法称为 10×10 折交叉验证方式.

4.4.1　经典特征选择方法介绍

论文考虑了 4 种对比方法:IG、CFS、Consist 和 ALL.具体来说:IG 方法[8,21]是一种有效并且广泛应用于特征选择的排序算法;CFS 特征选择方法[32]同时考虑了特征与类标间的类相关度以及特征彼此间的特征相关度,用 Best-First 搜索策略来寻找高类相关度低特征相关度的特征子集;Consist 方法[33]的目的是寻找一个最小特征子集,使用该子集的分类效果和使用全集的分类效果一致;ALL 方法不做特征选择,而使用原始特征集来构建缺陷预测模型.加上 3 种 FECS 特征选择方法,本次实验总共实现了 7 种方法.

在方法的参数取值设定上,我们根据宋擒豹等人[34]的建议,将从原始特征集中选出的特征数设置为$\lfloor \log n/2 \rfloor$,其中 n 表示原始特征数.这种参数取值适用于 IG 方法和本文提出的 FECS 方法.而 CFS 和 Consist 方法因为不能精确控制所选出的特征数[32-33],最终可能会超出实验设定的特征数量.在 RQ3 中,本文还会进一步分析 FECS 所选出的特征比例对模型预测性能的影响.

4.4.2　分类器

本文考虑了两类经典的分类方法来构建缺陷预测模型:贝叶斯方法中的朴素贝叶斯(Naive Bayes,NB)[9,23,28]和决策树方法中的 C4.5[35].与其他贝叶斯方法不同,朴素贝叶斯算法假设特征之间条件独立.即在类标确定的情况下,特征和特征之间相互独立.C4.5 在构建决策树的过程中根据特征的信息增益率选择最大的特征节点进行分裂,同时使用剪枝方法来减缓过拟合问题.NB 分类器和 C4.5 分类器是缺陷预测领域中应用最广泛的分类算法[9].本文基于 Weka 软件包实现了 NB 和 C4.5 分类器,并采用默认的参数取值.

5　结果的分析和讨论

为了回答上一节中提出的 3 个实验问题,本节首先在没有注入噪声的数据集上对 FECS 和其他特征选择方法进行比较.接着,对数据集注入噪声,观察 FECS 在噪声数据集上的性能.最后,进一步研究噪声注入率和特征选择比例对 FECS 性能的影响.

为了进一步分析两个方法间的性能差异,我们统计了两个方法间的 Win/Draw/Loss 信息,具体来说:"方法 A vs 方法 B"的 Win/Draw/Loss 信息包括 3 个值:Win、Draw 和 Loss.这 3 个值分别表示方法 A 好于、等于和差于方法 B 的数据集数量.在实验中,ALL 方法被固定为"方法 B",以此作为评测基准来比较其他特征选择方法.

本文为了分析结果在统计上的显著性,还使用了 Friedman 检验[28,36-37].Friedman 检验服从自由度为 $k-1$ 的卡方分布,其原假设是多个方法间的效果并不存在显著性差异.具体来说,如果发现检验结果的 p 值足够小(小于 0.05),则认为原假设不成立,即各个方法之间存在显著性差异.如果发现各个方法之间存在显著性差异,则可以使用后续检验方法来进一步找出哪些方法和其他方法不同.

5.1　RQ1:在注入噪声前的 FECS 性能分析

依据 4.4 节,本次实验对所有方法都做了 10×10 折交叉验证方式.表 2 给出了基于两种分类器的不同特征选择方法在 13 个数据集上的 *AUC* 值的均值,其中左边是基于 NB 分类器,右边是基于 C4.5 分类器.在每个数据集上,将最优结果加粗.表中最后两行,第 1 行是平均值,第 2 行是以 ALL 方法为评测基准的 Win/Draw/Loss 统计信息.

表 2 在注入噪声前,基于两种分类器的不同特征选择方法的 *AUC* 值

数据集	基于朴素贝叶斯分类器(NB)							基于决策树分类器(C4.5)						
	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE
Eclipse2.0	0.791	0.788	0.799	**0.800**	0.798	0.795	0.741	0.671	0.745	0.692	0.701	**0.758**	0.748	0.736
Eclipse2.1	0.745	0.758	0.745	0.760	**0.768**	0.764	0.755	0.590	0.700	0.712	0.628	**0.738**	0.736	0.723
Eclipse3.0	0.762	0.762	0.755	0.769	**0.783**	0.770	0.774	0.637	0.732	0.704	0.671	**0.748**	0.728	0.635
cm1	0.750	0.764	0.757	0.500	**0.769**	0.726	0.768	0.528	0.540	0.548	0.500	**0.558**	0.502	0.488
kc1	0.788	0.792	0.791	0.790	**0.793**	0.792	0.790	0.660	0.725	0.697	0.692	0.707	0.718	**0.729**
kc3	0.808	0.766	0.797	0.795	0.821	**0.822**	0.813	0.567	0.586	0.606	0.611	**0.752**	0.750	0.660
kc4	0.757	0.764	0.751	0.744	**0.770**	0.744	0.759	0.775	0.726	0.754	0.765	0.726	0.744	**0.800**
mc2	**0.709**	0.647	0.646	0.654	0.656	0.648	0.654	**0.636**	0.577	0.578	0.589	0.606	0.559	0.624
mw1	0.744	0.777	0.780	0.772	0.784	0.781	**0.815**	0.556	0.558	0.570	0.602	**0.605**	0.562	0.560
pc1	0.746	0.739	0.741	0.743	**0.780**	0.772	0.768	0.650	0.649	0.677	0.672	0.674	**0.697**	0.653
pc3	0.773	0.791	0.788	0.729	**0.796**	0.787	0.778	**0.635**	0.536	0.584	0.566	0.558	0.634	0.602
pc4	**0.836**	0.828	0.814	0.832	0.834	0.825	0.827	0.760	0.854	0.852	0.771	**0.870**	0.773	0.789
pc5	0.928	0.943	0.871	0.935	**0.944**	0.930	0.932	0.733	0.890	0.850	0.821	0.852	**0.901**	0.874
平均值	0.780	0.778	0.772	0.756	**0.792**	0.781	0.783	0.646	0.678	0.679	0.661	**0.704**	0.696	0.683
W/D/L	—	7/1/5	5/1/7	6/0/7	11/0/2	9/0/4	10/0/3	—	9/0/4	10/0/3	9/0/4	10/0/3	9/0/4	9/0/4

当选择 NB 作为分类器时,从 *AUC* 平均值角度看,3 种 FECS 方法占据了所有方法的前 3 名.其中,FECS_FR 方法的表现最好.不过,我们也发现特征选择方法并不一定对所有的数据集都有效.例如,在数据集 mc2 和 pc4 上,ALL 方法的性能要优于其他特征选择方法.从 Win/Draw/Loss 角度看,3 种 FECS 方法在 13 个数据集中赢得了 10 次,其中 FECS_FR 赢得 8 次.在与评测基准 ALL 方法的对比中,FECS 方法也都赢多输少.执行 Friedman 检验后的 p 值为 0.0288,进一步说明各个方法之间存在显著性差异.

当选择 C4.5 作为分类器时,从 *AUC* 平均值角度看,3 种 FECS 方法仍旧占据了前 3 名.其中,仍是 FECS_FR 方法的表现最好.而 FECS_MR 和 FECS_LE 表现类似.ALL 方法只在数据集 mc2 和 pc3 上表现较好,但平均而言是所有方法中最差的.从 Win/Draw/Loss 角度看,3 种 FECS 方法在 13 个数据集中赢得了 11 次,其中 FECS_FR 赢得 7 次.在与评测基准 ALL 方法的对比中,所有的特征选择方法,也包括 FECS 方法,都是赢多输少.最后,执行 Friedman 检验后的 p 值为 0.0435,再一次说明各个方法之间存在显著性差异.

通过上述分析,论文得出如下结论:(1)FECS 方法在注入噪声前的数据集上,性能优于其他特征选择算法,对于不同分类器,这种性能优越性保持稳定;(2)在 3 种 FECS 方法中,FECS_FR 方法的平均性能最好.

5.2 RQ2:在注入噪声后的 FECS 性能分析

为了回答 RQ2,本文按照 4.2 节的方法对数据集注入 F-Noise 和 C-Noise 这两类噪声.根据 Herzig 等人的[18]的建议,将噪声率 ω 控制为 33.8%.对于实验结果,这里使用 *AUC* 和 *RLA* 来度量方法的有效性.

5.2.1 *AUC* 值分析

表 3 显示了在噪声注入后,预测模型的 *AUC* 值的平均值.与表2类似,表中左边基于NB分类器,

表 3 在注入噪声后,基于两种分类器的不同特征选择方法的 *AUC* 值(ω=33.8%)

数据集	基于朴素贝叶斯分类器(NB)							基于决策树分类器(C4.5)						
	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE
Eclipse2.0	0.749	0.739	**0.794**	0.778	0.744	0.735	0.742	0.604	0.587	0.592	0.626	**0.657**	0.621	0.630
Eclipse2.1	0.732	0.714	0.722	0.700	**0.742**	0.727	0.720	0.571	0.556	0.554	0.553	**0.609**	0.578	0.571
Eclipse3.0	0.731	0.697	**0.755**	0.747	0.720	0.715	0.742	0.598	0.591	0.625	**0.628**	0.624	0.616	0.599
cm1	0.632	0.588	0.500	0.493	**0.633**	0.607	0.612	0.499	0.504	0.502	0.505	**0.544**	0.500	0.461
kc1	**0.743**	0.712	0.632	0.600	0.683	0.660	0.681	0.557	0.573	0.554	0.563	0.587	0.610	**0.623**
kc3	0.707	0.718	0.555	0.521	0.700	**0.724**	0.690	0.546	0.502	0.509	0.500	0.567	**0.585**	0.540
kc4	0.664	0.648	0.558	0.518	**0.736**	0.716	0.684	0.549	0.564	0.530	0.538	**0.636**	0.568	0.599
mc2	**0.613**	0.582	0.500	0.518	0.576	0.597	0.553	0.526	0.517	0.513	0.514	**0.581**	0.547	0.579
mw1	0.679	0.692	0.523	0.531	**0.742**	0.728	0.738	0.500	0.525	0.509	0.498	**0.565**	0.560	0.540
pc1	0.572	0.609	0.525	0.516	0.635	0.594	**0.643**	0.504	0.530	0.500	0.503	0.540	0.544	**0.552**
pc3	0.661	0.706	0.542	0.521	**0.747**	0.712	0.721	0.508	0.515	0.511	0.516	0.544	0.553	**0.558**
pc4	0.726	0.753	0.633	0.531	0.753	**0.754**	0.750	0.561	0.569	0.537	0.547	**0.630**	0.604	0.600
pc5	0.915	0.923	0.837	0.927	**0.932**	0.926	0.929	0.637	0.605	0.587	0.629	**0.652**	0.623	0.607
平均值	0.702	0.699	0.621	0.608	**0.719**	0.707	0.708	0.551	0.549	0.540	0.548	**0.608**	0.578	0.583
W/D/L	—	6/0/7	2/0/11	3/0/10	8/0/5	7/0/6	7/0/6	—	7/0/6	4/0/9	5/0/8	13/0/0	12/0/1	9/1/3

右边基于 C4.5 分类器.在每个数据集上,将最优结果进行加粗.表中最后两行,第 1 行是平均值,第 2 行是以 ALL 方法为评测基准的 Win/Draw/Loss 统计信息.

当选择 NB 作为分类器时,从 *AUC* 平均值角度看,3 种 FECS 方法还是占据了前 3 名.在 3 种方法中,FECS_FR 仍然表现最好.数据表明在不论数据集是否注入噪声的情况下,FECS 方法都表现较好,并且策略 1 也要优于其他两种策略.ALL 方法紧随其后,并且相比其他特征选择方法有较为明显的优势.特别是在数据集 kc1 和 mc2 上,ALL 表现最优.这说明其他经典方法在噪声数据集上也存在问题,因而提出具有抗噪能力的特征选择方法是有必要的.从 Win/Draw/Loss 角度看,在与评测基准 ALL 方法的对比上,3 种 FECS 特征选择方法都是赢多输少,而另外 3 种经典特征选择方法并不完全如此.执行 Friedman 检验的 p 值为 0.0011(远小于 0.05),说明各个方法之间确实具有显著性差异.

当选择 C4.5 作为分类器时,从 *AUC* 平均值角度看,3 种 FECS 方法依然与前面的结论保持一致,FECS_FR 仍然表现最好.除此之外,相对于其他 3 种经典特征选择方法,ALL 方法在 *AUC* 和 Win/Draw/Loss 上都表现更优.在与评测基准 ALL 方法的对比上,Win/Draw/Loss 结果证实了 FECS 具有显著优势,FECS 在全部 13 个数据集上均优于 ALL 方法.执行 Friedman 检验的 p 值为 1.55-E06(远小于 0.05),再次说明方法间存在显著性差异.

5.2.2　*RLA* 值分析

这里我们度量 *AUC* 的相对损失(即 *RLA*)作为衡量特征选择方法在预测性能上的稳定性,*RLA* 值越小说明方法在注入噪声前后的 *AUC* 损失越小(稳定性越好).表 4 给出了预测模型稳定性 *RLA* 的平均值.其中左边基于 NB 分类器,右边基于 C4.5 分类器.同样,在每个数据集上,将最优结果进行加粗.表中最后两行,第 1 行是平均值,第 2 行是以 ALL 为评测基准的 Win/Draw/Loss 信息.

表 4　在注入噪声后,基于两种分类器的不同特征选择方法的 *RLA* 值(ω=33.8%)

数据集	基于朴素贝叶斯分类器(NB)							基于决策树分类器(C4.5)						
	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE	ALL	IG	CFS	Consist	FECS_FR	FECS_MR	FECS_LE
Eclipse2.0	0.053	0.062	0.006	0.028	0.068	0.076	***−0.001***	**0.099**	0.212	0.144	0.107	0.133	0.169	0.143
Eclipse2.1	**0.017**	0.058	0.031	0.079	0.034	0.048	0.046	**0.032**	0.205	0.222	0.119	0.174	0.215	0.211
Eclipse3.0	0.041	0.085	**0.001**	0.029	0.080	0.071	0.042	0.061	0.193	0.113	0.063	0.166	0.154	**0.056**
cm1	0.157	0.230	0.340	**0.015**	0.177	0.164	0.203	0.056	0.066	0.083	***−0.010***	0.024	0.004	0.055
kc1	**0.057**	0.101	0.201	0.240	0.138	0.166	0.137	0.155	0.210	0.205	0.186	0.169	0.150	**0.146**
kc3	0.125	**0.063**	0.304	0.345	0.148	0.119	0.152	**0.038**	0.143	0.161	0.182	0.246	0.220	0.182
kc4	0.123	0.152	0.257	0.304	0.044	**0.038**	0.099	0.292	0.223	0.297	0.296	**0.125**	0.236	0.251
mc2	0.135	0.100	0.226	0.209	0.122	**0.078**	0.154	0.173	0.104	0.112	0.127	0.041	**0.022**	0.072
mw1	0.088	0.110	0.330	0.312	**0.054**	0.067	0.095	0.101	0.060	0.108	0.173	0.065	**0.004**	0.036
pc1	0.233	0.176	0.292	0.306	0.186	0.231	**0.163**	0.224	0.183	0.261	0.251	0.199	0.220	**0.155**
pc3	0.144	0.107	0.312	0.285	**0.061**	0.096	0.074	0.200	0.038	0.124	0.088	**0.024**	0.128	0.073
pc4	0.132	0.090	0.222	0.362	0.097	**0.086**	0.093	0.262	0.334	0.370	0.291	0.275	**0.218**	0.239
pc5	0.014	0.021	0.039	0.008	0.013	0.004	**0.003**	**0.131**	0.320	0.309	0.234	0.235	0.308	0.306
平均值	0.101	0.104	0.197	0.194	**0.094**	0.096	0.097	**0.140**	0.176	0.193	0.162	0.144	0.158	0.148
W/D/L	—	5/0/8	2/0/11	4/0/9	7/0/6	8/0/5	6/0/7	—	5/0/8	2/0/11	3/0/10	6/0/7	8/0/5	9/0/4

当选择 NB 作为分类器时,根据 *RLA* 平均值数据来看,3 种 FECS 方法都具有良好的稳定性.在 3 种方法中,稳定性最好的是 FECS_FR.除此之外,ALL 方法比其他特征选择方法具有更好的稳定性,并且在数据集 Eclipse2.1 和 kc1 上表现最好.通过 Win/Draw/Loss 分析,3 种 FECS 特征选择方法并不比 ALL 方法具有明显优势.均值最好的方法 FECS_FR 在 13 个数据集上,仅赢得 7 次,同时有 6 次不及 ALL 方法.在执行 Friedman 检验后,得到 p 值为 0.0153,说明结果仍旧具有显著性差异.

当选择 C4.5 作为分类器时,根据 *RLA* 平均值数据来看,ALL 表现最好,FECS 方法则比其他 3 种经典特征选择方法要稳定.从各个数据集上看,ALL 也同样是最为稳定的方法,至少在 4 个噪声数据集上超过其他特征选择方法.这说明 FECS 还需要进一步的改进来提高在各种分类器上的稳定性.但总体来说,在本文的实验中,C4.5 分类器的效果不如 NB 分类器.从 Win/Draw/Loss 角度看,FECS_MR 和 FECS_LE 相对于 FECS_FR 都是赢多输少.最后,在进行 Friedman 检验后,得到 p 值为 0.027,该取值可以确保实验结果的可信度.另外,实验结果存在一个有趣的负值现象,例如在数据集 cm1 上,Consist 方法的在注入噪声后的预测效果比注入之前的效果有微弱的提高.同样的现象还发生在将

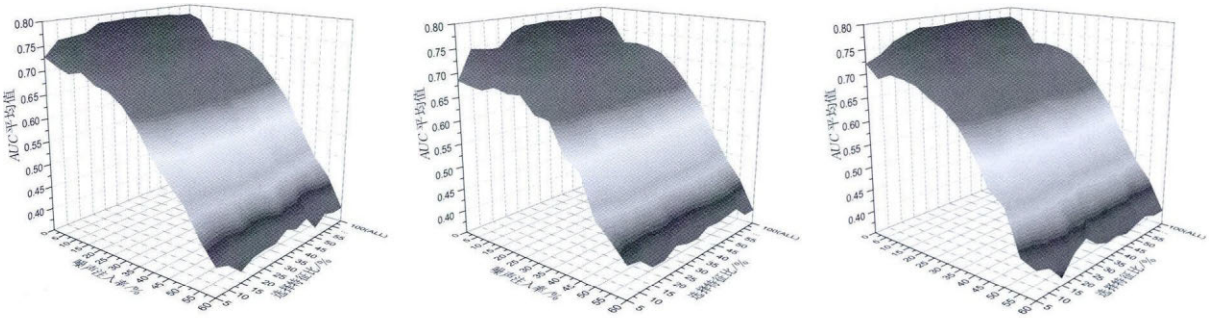
基于 NB 分类器的 FECS_LE 方法应用于数据集 Eclipse2.0 上.这说明特征选择方法对 C-Noise 和 F-Noise 这两类噪声具有相对好的容错能力.

通过上述分析,论文得出如下结论:(1) FECS 方法在注入噪声后的数据集上,预测的性能和稳定性都要优于其他特征选择算法,尤其是 FECS_FR 方法;(2) 另一方面,当采用 C4.5 分类器时,ALL 方法相对于 FECS 方法更加稳定.这说明 FECS 需要进一步的改进,以使得其在不同分类器上都可以取得更加稳定的表现.

5.3 RQ3:噪声注入率、特征选择比例及噪声类型对 FECS 的影响

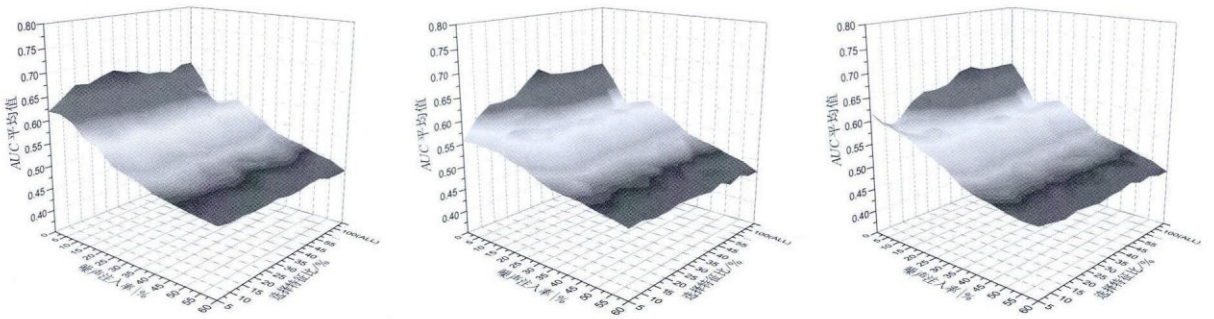
该小节先后调节噪声注入率 ω 和选择比例(记作 $|F_{\%}|$),分析取值变化对 FECS 性能会产生何种影响.与前面的实验保持一致,这里仍是同时注入 C-Noise 和 F-Noise 这两种噪声,使用平均 13 个数据集上得到的 *AUC* 值作为预测结果的评价指标.

在实验中,每一次都是固定一个噪声注入率 ω 和特征选择比例 $|F_{\%}|$,经过 10×10 折交叉验证后,在 13 个数据集上得到的 *AUC* 平均值.噪声率 ω 的设置范围从 0 到 60%,每次步长为 5%,而特征选择比例的设置范围从 10% 到 55%,每次步长也是 5%.本文还考虑 ALL 方法($|F_{\%}|=100\%$),作为不同噪声率下 FECS 的评测基准.图 4 和图 5 是实验结果的 3D 图,分别记录采用 NB 分类器和 C4.5 分类器的预测性能结果.所有 6 幅子图都使用统一的坐标尺度,以便于比较.



(a) FECS_FR (b) FECS_MR (c) FECS_LE

图 4 基于朴素贝叶斯分类器(NB),噪声注入率和选择特征比例对 *AUC* 值的影响



(a) FECS_FR (b) FECS_MR (c) FECS_LE

图 5 基于决策树分类器(C4.5),噪声注入率和选择特征比例对 *AUC* 值的影响

5.3.1 噪声注入率的影响

图 4 选择 NB 作为分类器,其中每张子图都对应一种策略.随着噪声率 ω 的不断增加,3 种 FECS 方法都会受到严重的干扰.尽管 FECS_FR 是 3 种策略中最好的,但随着噪声的不断注入,其优势相对于另外两种策略并不明显.不过通过图 4,可以看到 3 种 FECS 方法都存在一个可容忍的噪声注入率范围,在这个范围内噪声对方法的预测性能影响是可接受的,即图中左侧深色区域.这个区域对于 3 个方法来说差别并不很明显,其上界接近 40%的噪声率.

图 5 选择 C4.5 作为分类器,实验结果类似.随着噪声率 ω 的上升,3 种 FECS 方法的预测性能都会明显下降.同时 FECS_FR 也没有比其他两种策略具有明显优势,但和噪声率的影响相比,该方法提升的优势并不明显.随后,通过分析噪声可接受的左上侧深色区域后,我们发现:当噪声率小于 30% 的时候,FECS 都具有良好的抗噪性能.除此之外,当噪声率高于 50% 的时候,不管基于何种策略的 FECS,基于 NB 分类器的性能要差于在 C4.5 分类器的性能.这表明在含有噪声的数据集上,即使在无

噪声时预测效果较好的分类器性能，也可能大幅下降，甚至不如无噪声时预测效果一般的其他分类器.

5.3.2　特征选择比例的影响

对于特征选择比例来说，它和噪声注入率变化的情况完全不同. 仅选择很少一部分的特征（例如 $|F_{\%}|=15\%$），3 种 FECS 都能获得比选择全部特征的 ALL 方法更好的性能结果，而不用考虑噪声率和分类器选择的影响. 事实上，无论基于哪种特征选择策略，FECS 仅仅选用少量特征就能提高最后的预测性能. 这种趋势在两个分类器上都很明显，也与最新的一些相关研究工作结论保持一致[23]. 但对于 FECS 的 3 种方法，都很难找出其最优的选择特征比例 $|F_{\%}|$. 固定某个噪声注入率，观察选择特征比维度的截面，可以看到预测性能起伏变化还是比较大的，这可能由于在不同数据集上，方法取得最优性能所需选择的特征比例一般是不尽相同的. 但总体来说，FECS_FR 要优于另外两种 FECS 方法.

5.3.3　噪声类型的影响

为了更好的研究不同类型的噪声 C-Noise 和 F-Noise 对 FECS 性能的影响. 下面的实验将分别单独注入两种噪声类型，进而分析 C-Noise 和 F-Noise 在不同注入比率上对 FECS 性能的影响. 和前面一样，我们仍然选取 ALL 方法作为对比方法. 同时，为了便于比较，我们将选中的特征数与 RQ1 中的实验设置保持一致，即均为 $\lfloor \log n/2 \rfloor$，所有的结果都取在 13 个数据集上的 *AUC* 值的均值.

图 6 和图 7 的柱状图分别显示了随着 C-Noise 和 F-Noise 注入率从 10% 上升到 50%，各个方法在使用 NB 分类器做缺陷预测的效果对比图. 图中 x 轴表示噪声的注入率，y 轴表示 *AUC* 值. 3 种矩形分别表示 FECS 的 3 种策略. 而黑色直线则表示对比方法 ALL 的预测结果.

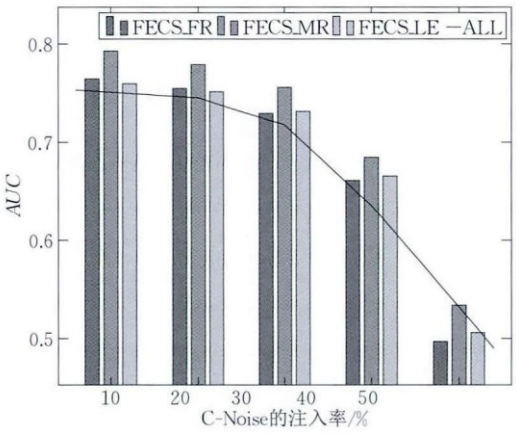


图 6　基于朴素贝叶斯（NB），C-Noise 的注入率对 *AUC* 值的影响

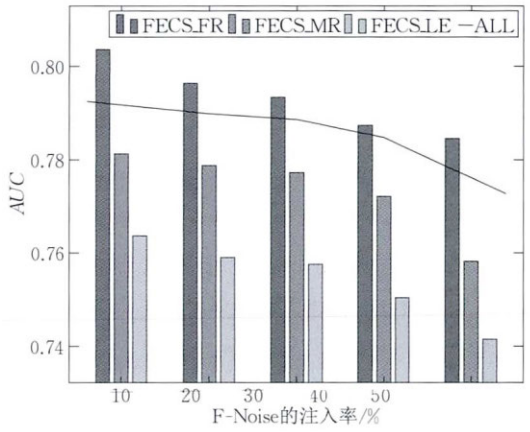


图 7　基于朴素贝叶斯（NB），F-Noise 的注入率对 *AUC* 值的影响

首先我们分析 C-Noise 注入率对 *AUC* 值的影响. 基于图 6，我们认为 FECS_MR 在 C-Noise 噪声下相对其他 3 种方法具有更好的预测性能，而 FECS_FR 和 FECS_LE 则和对比方法 ALL 效果类似. 前面 3.3 节已对 FECS_MR 和 FECS_LE 两种策略进行过分析，实验表明这两种策略通过特征相关度 FF-Correlation 的计算来取代类相关度 FC-Relevance 的计算，确实能有效缓解 C-Noise 带来的影响. 除此之外，图 6 中当噪声注入率在提高到 30% 之前，FECS_LE 策略并没有优于 FECS_FR，但是随着噪声注入率的不断提高，FECS_LE 的预测性能反超了 FECS_FR. 总体而言，4 种方法在噪声率小于 40% 时都具有稳定的性能表现. 但是当噪声注入率提高到 40% 以上，预测性能由于误分类会出现显著下降. 该发现也验证了 Kim 等人[6]的研究发现，他们在实证研究中发现：若数据集中存在 20%～35% 的 C-Noise，分类器的预测性能会显著下降. 我们还计算了 10×10 折交叉检验在 13 个数据集的 *AUC* 方差均值，FECS_MR 的方差范围在 0.01～0.05 之间，因此相较于其他方法表现更加稳定.

其次我们分析 F-Noise 注入率对 *AUC* 值的影响. 从图 7 可以看到，任意单个方法随着噪声注入率不断提高，与 C-Noise 相比预测性能的下降程度相对平缓很多. 这主要是因为特征和特征之间本身存在某种依赖关系，而我们选取的特征很可能包含了那些噪声特征的部分有效信息. 因此尽管训练集存在特征噪声 F-Noise，也不会严重影响到预测性能. 特别是 FECS_FR 策略，在噪声注入率提高到 40% 至 50% 时也依然能表现平稳，显示出该策略在抵抗 F-Noise 上具有显著优势. 一个可能的解释是

FECS_FR 更关注类相关度，而减小了特征相关度，特别是噪声特征相关度的影响，因此在存在 F-Noise 时，要明显优于其他几种方法. 除此之外，我们还计算了 *AUC* 的均值方差，结果显示 FECS_FR 方法甚至好过 FECS_MR 和 FECS_LE 的最好情况（*AUC* 均值加上视作误差值的 *AUC* 均值方差）.

通过上述分析，我们基于噪声类型给出了使用 FECS 这 3 种策略的指导建议. 当数据集含有较多 C-Noise 噪声时，FECS_MR 和 FECS_LE 都是一种好的选择，其中 FECS_MR 更为稳定. 而当数据集中含有较多 F-Noise 时，FECS_FR 则是最好的选择. 同样地，基于 C4.5 决策树分类器，我们也得到了相似的结论，限于篇幅，在论文中不再赘述.

基于 5.1 节到 5.3 节的分析，论文得出如下结论：(1) 噪声注入率会对 FECS 的预测产生负面影响. 而 3 种 FECS 的抗噪范围比较相似，该范围会由于选择的分类器不同而变化；(2) 另一方面，仅选择少量特征就能获得不错的预测性能，而不用考虑噪声率和分类器选择的影响. 和另外两种策略相比，在特征选择比例变化的情况下，FECS_FR 拥有更稳定的预测性能；(3) FECS_FR 在含有混合噪声或大量 F-Noise 的数据集上预测效果明显，但当已知数据集中只包含大量 C-Noise 时，FECS_MR 方法则更为有效.

5.4 有效性影响因素分析

本节主要讨论可能影响实验研究有效性的一些影响因素. 外部有效性主要涉及到实验研究得到的结论是否具有一般性. 本文选用了软件缺陷预测研究中常采用的 NASA 项目数据集和 Eclipse 项目数据集[8-9,28]，因此可以保证研究结论具有一定的代表性. 此外，本文还依据文献[26-27]设计了噪声注入方法来模拟真实环境下存在的噪声模式. 内部有效性主要涉及到可能影响到实验结果正确性的内部因素. 本文编写的代码主要基于 Weka 软件包，因此可以最大程度的保证分类器和特征选择方法实现的正确性，除此之外，我们还通过一些简单实例对算法实现的正确性进行了验证. 结论有效性主要涉及到使用的评测指标是否合理. 因为论文中采用的数据集具有类不平衡问题，因此我们考虑了 *AUC* 这一重要评测指标. 除此之外，我们还提出了一种 *RLA* 指标来评估特征选择方法的性能稳定性. 同时为了避免实例次序对结果的影响，实验结果均采用 10×10 折交叉验证方式.

6 结论与展望

论文针对软件缺陷预测数据集中的噪声问题，提出一种新颖的可容忍噪声的特征选择框架 FECS，即首先基于特征之间的关联性，将已有特征进行聚类分析，随后提出 3 种启发式特征选择策略，从每个簇中选出一个最为典型的特征. 基于来自实际项目的数据集上，对该方法的有效性进行了验证.

我们认为该方法仍有一些后续工作值得扩展，具体来说：(1) 我们将尝试继续设计有效的特征选择策略来提高 FECS 的有效性；(2) 考虑更多来自实际项目的数据集来验证论文结论的一般性；(3) 尝试分析 FECS 方法在其他分类器上的效果.

参 考 文 献

[1] Wang Qing, Wu Shu-Jian, Li Ming-Shu. Software defect prediction. 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. 软件学报，2008，19(7)：1565-1580)

[2] Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 2012, 38(6): 1276-1304

[3] Chen Xiang, Gu Qing, Liu Wang-Shu, et al. Software defect prediction. 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. 软件学报，2016，27(1)：1-25)

[4] Yu Shu-Si, Zhou Shui-Geng, Guan Ji-Hong. Software engineering data mining: A survey. Journal of Frontiers of Computer Science and Technology, 2012, 6(1): 1-31(in Chinese)
(郁抒思，周水庚，关佶红. 软件工程数据挖掘研究进展. 计算机科学与探索，2012，6(1)：1-31)

[5] Radjenovic D, Hericko M, Torkar R, Zivkovic A. Software fault prediction metrics: A systematic literature review. Information and Software Technology, 2013, 55(8): 1397-1418

[6] Kim S, Zhang H Y, Wu R X, Gong L. Dealing with noise in defect prediction//Proceedings of the International Conference on Software Engineering. Honolulu, USA, 2011: 481-490

[7] Tantithamthavorn C, McIntosh S, Hassan A E, et al. The impact of mislabeling on the performance and interpretation of defect prediction models//Proceedings of the International Conference on Software Engineering. Firenze, Italy, 2015: 812-823

[8] Liu S L, Chen X, Liu W S, et al. FECAR: A feature selection framework for software defect prediction//Proceedings of the Annual Computer Software and Applications Conference. Västerås, Sweden, 2014: 426-435

[9] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 2007, 32(11): 1-12

[10] Song Q B, Jia Z H, Shepperd M, et al. A general software defect-proneness prediction framework. IEEE Transactions on Software Engineering, 2011, 37(3): 356-370

[11] Gao K H, Khoshgoftaar T M, Wang H J, Seliya N. Choosing software metrics for defect prediction: An investigation on feature selection techniques. Software-Practice and Experience, 2011, 41(5): 579-606

[12] Bird C, Bachmann A, Aune E, et al. Fair and balanced? bias in bug-fix datasets//Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on Foundations of Software Engineering. Amsterdam, Netherlands, 2009: 121-130

[13] Bachmann A, Bird C, Rahman F, et al. The missing links: Bugs and bug-fix commits//Proceedings of the International Symposium on Foundations of Software Engineering. Santa Fe, USA, 2010: 97-106

[14] Nguyen T H, Adams B, Hassan A E. A case study of bias in bug-fix datasets//Proceedings of the Working Conference on Reverse Engineering. Boston, USA, 2010: 259-268

[15] Wu R X, Zhang H Y, Kim S, Cheung S C. Relink: Recovering links between bugs and changes//Proceedings of the European Conference on Foundations of Software Engineering. Szeged, Hungary, 2011: 15-25

[16] Nguyen A T, Nguyen T T, Nguyen H A, Nguyen T N. Multi-layered approach for recovering links between bug reports and fixes//Proceedings of the International Symposium on Foundations of Software Engineering. Cary, USA, 2012: 1-11

[17] Rahman F, Posnett D, Herraiz I, Devanbu P. Sample size vs. bias in defect prediction//Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on Foundations of Software Engineering. Saint Petersburg, Russia, 2013: 147-157

[18] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: How misclassification impacts bug prediction//Proceedings of the International Conference on Software Engineering. San Francisco, USA, 2013: 392-401

[19] Witten I H, Frank E. Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, USA: Morgan Kaufmann Publishers, 2005

[20] Kononenko I. Estimating attributes: Analysis and extensions of relief//Proceedings of the European Conference on Machine Learning. Catania, Italy, 1994: 171-182

[21] Yu L, Liu H. Efficient feature selection via analysis of relevance and redundancy. The Journal of Machine Learning Research, 2004, 5: 1205-1224

[22] Hall M A, Holmes G. Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(6): 1437-1447

[23] Shivaji S, Whitehead E J, Akella J R, Kim S. Reducing features to improve code change-based bug prediction. IEEE Transactions on Software Engineering, 2013, 39(4): 552-569

[24] Wald R, Khoshgoftaar T M, Shanab A A. Comparison of two frameworks for measuring the stability of gene-selection techniques on noisy class-imbalanced data//Proceedings of the International Conference on Tools with Artificial Intelligence. Washington, USA, 2013: 881-888

[25] Hulse J V, Khoshgoftaar T M. Knowledge discovery from imbalanced and noisy data. Data & Knowledge Engineering, 2009, 68(12): 1513-1542

[26] Saez J A, Galar M, Luengo J, Herrera F. Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness. Information Sciences, 2013, 247: 1-20

[27] Zhu X, Wu X. Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review, 2004, 22(3): 177-210

[28] Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 2008, 34(4): 485-496

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

[30] Provost F, Fawcett T. Robust classification for imprecise environments. Machine Learning, 2001, 42(3): 203-231

[31] Ling C X, Huang J, Zhang H. AUC: A statistically consistent and more discriminating measure than accuracy//Proceedings of the International joint Conference on Artificial Intelligence. Acapulco, Mexico, 2003: 519-524

[32] Hall M A. Correlation-based feature selection for discrete and numeric class machine learning//Proceedings of the International Conference on Machine Learning. Stanford, USA, 2000: 359-366

[33] Dash M, Liu H, Motoda H. Consistency based feature selection//Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications. Kyoto, Japan, 2000: 98-109

[34] Song Q, Ni J, Wang G. A fast clustering-based feature subset selection algorithm for high-dimensional data. IEEE Transactions on Knowledge and Data Engineering, 2003, 25(1): 1-14

[35] Quinlan J R. C4.5: Programs for Machine Learning. San Francisco, USA: Morgan Kaufmann Publishers, 1993

[36] Friedman M. A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics, 1940, 11(1): 86-92

[37] Demsar J. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 2006, 7: 1-30

LIU Wang-Shu, born in 1987, Ph. D. candidate. His research interest is software defect prediction.

CHEN Xiang, born in 1980, Ph. D., associate professor. His research interests include software defect prediction, regression testing, fault localization, and combinatorial testing.

GU Qing, born in 1972, Ph. D., professor, Ph. D. supervisor. His research interests include software quality assurance and distributed computing.

LIU Shu-Long, born in 1990, M. S. His research interest is software defect prediction.

CHEN Dao-Xu, born in 1942, professor, Ph. D. supervisor. His research interests include distributed computing and parallel processing.

Background

Currently software defect prediction (SDP) is a hot research topic in software engineering. It can identify potential defect-proneness software modules based on a defect prediction model. Based on the prediction results, we can allocate the test resources in a cost-effective manner. However, to construct an effective defect prediction model, it depends on the high quality datasets by mining software archives, such as software configuration management and bug tracking systems. Although researchers have designed many useful code or process metrics (i. e., features) to measure these modules, a large amount of features and noisy data are still the two major issues in this research field. The former is the high dimensionality problem caused by too many unnecessary features, and the latter is the noisy data problem caused by mis-recorded features or labels. Hence, in this paper, we propose a robust feature selection method to filter the irrelevant and redundant feature, and to resist the inevitable noises in software datasets simultaneously.

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61373012, 61202006, 61321491, 91218302), the EU FP7 IRSES MobileCloud Project Grant (No. 612212), the Open Project of State Key Laboratory for Novel Software Technology at Nanjing University (KFKT2016B18), the Program for Outstanding Ph. D. Candidate of Nanjing University and the Collaborative Innovation Center of Novel Software Technology and Industrialization.