中国科学: 信息科学 2016 年 第 46 卷 第 9 期: 1298–1320
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS

论 文

软件缺陷预测中基于聚类分析的特征选择方法

刘望舒[①②], 陈翔[①③], 顾庆[①②*], 刘树龙[①②], 陈道蓄[①②]

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, 南通 226019
* 通信作者. E-mail: guq@nju.edu.cn

收稿日期: 2016–04–25; 接受日期: 2016–05–31; 网络出版日期: 2016–09–18
国家自然科学基金 (批准号: 61373012, 61321491, 91218302, 61202006)、国家重点基础研究发展计划 (973 计划)(批准号: 2009CB320705)、江苏省高校自然科学研究项目 (批准号: 12KJB520014)、南京大学计算机软件新技术国家重点实验室开放课题 (批准号: KFKT2016B18) 和南京大学软件新技术与产业化协同创新中心资助项目

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. 但有时候搜集到的缺陷预测数据集中含有的冗余特征和无关特征会影响到缺陷预测模型的性能. 提出一种基于聚类分析的特征选择方法 FECAR. 具体来说, 首先基于特征之间的关联性 (即 FFC), 将已有特征进行聚类分析. 随后基于特征与类标间的相关性 (即 FCR), 对每个簇中的特征从高到低进行排序并选出指定数量的特征. 在实证研究中, 借助对称不确定性 (symmetric uncertainty) 来计算 FFC, 借助信息增益 (information gain)、卡方值 (chi-square) 或 ReliefF 来计算 FCR. 以 Eclipse 和 NASA 数据集等实际项目为评测对象, 重点分析了应用 FECAR 方法后的缺陷预测模型的性能, FECAR 方法选出的特征子集冗余率和比例. 结果验证了 FECAR 方法的有效性.

关键词 软件质量保障 缺陷预测 数据挖掘 特征选择 聚类分析

1 引言

软件缺陷产生于开发人员的编码过程. 需求理解不正确, 软件开发过程不合理或开发人员的经验不足, 均有可能产生软件缺陷. 而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. 软件缺陷预测 (software defect prediction, 简称 SDP) [1,2] 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 (metrics), 随后通过挖掘软件历史仓库 (software historical repositories) 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.

但在构建缺陷预测数据集时, 若考虑了大量度量元 (即特征), 会造成数据集中的部分特征是冗余特征 (redundant feature) 或无关特征 (irrelevant feature). 这类特征的存在会提高缺陷预测模型的构建时间, 甚至有时会降低模型的预测性能. 因此如何设计出新颖有效的特征选择方法来移除这类特征具有重要的研究意义.

引用格式: 刘望舒, 陈翔, 顾庆, 等. 软件缺陷预测中基于聚类分析的特征选择方法. 中国科学: 信息科学, 2016, 46: 1298–1320, doi: 10.1360/N112015-00276

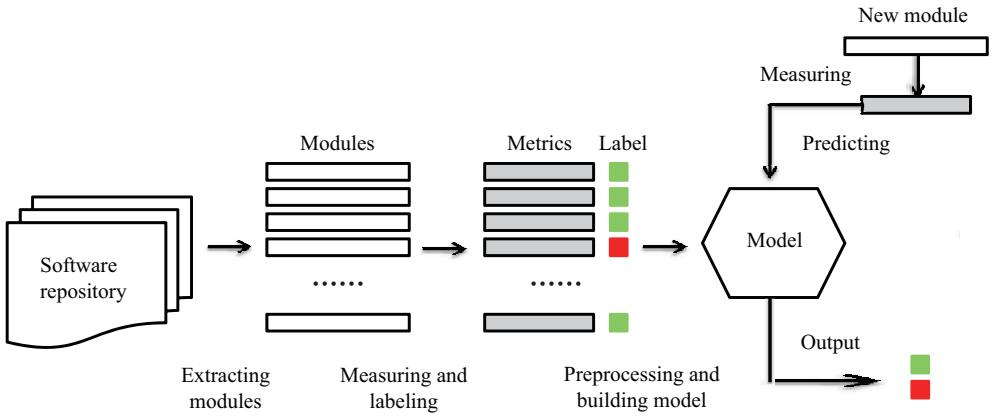

图 1 (网络版彩图) 软件缺陷预测过程

Figure 1 (Color online) A process of software defect prediction

通过分析特征之间的关联性以及特征与类标之间的相关性, 论文提出一种新颖的特征选择方法 FECAR (feature clustering and feature ranking), 尝试识别并移除缺陷预测数据集的无关特征和冗余特征. 该方法首先通过分析特征之间的关联性进行聚类分析, 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, 根据特征与类标间的相关性从高到低进行排序, 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. 实证研究中, FECAR 方法借助对称不确定性 (symmetric uncertainty, 简称 SU) 来计算特征之间的关联性, 借助信息增益 (information gain, 简称 IG), 卡方值 (chi-square, 简称 CS) 或 ReliefF (简称 RF) 来计算特征与类标间的相关性. 以 Eclipse 和 NASA 等实际项目为评测对象, 结果表明: (1) 不论选择哪种特征与类标间的相关性度量方法 (即 IG, RF 或 CS), 与仅基于排序的特征选择方法相比, FECAR 通过考虑特征聚类可以有效提高缺陷预测性能. (2) 当使用 IG 或 CS 作为特征相关性度量方法时, 特征聚类可以有效地降低所选出的特征子集的冗余率; 但使用 RF 这一种可以有效处理冗余特征的度量方法时, 特征聚类反而会提高所选出的特征子集的冗余率. (3) 若 FECAR 方法使用 IG 作为特征相关性度量方法, 则可以获得最好的缺陷预测性能, 同时也要显著优于其他经典的特征选择方法.

论文剩余内容结构安排如下: 第 2 节首先介绍软件缺陷预测的研究背景和已有研究工作, 随后重点介绍论文关注的特征选择方法及其在软件缺陷预测中的应用. 第 3 节介绍 FECAR 方法及其具体实现细节. 第 4 节介绍论文的实证研究, 包括评测对象、评测指标、实验设计和结果分析等. 最后总结全文并对下一步研究工作进行展望.

2 研究背景与相关工作

2.1 软件缺陷预测

软件缺陷预测 [1~4] 是当前软件工程数据挖掘领域的一个研究热点 [5~13]. 缺陷预测模型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.

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. 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 等), 缺陷跟踪系统 (例如 Bugzilla, Mantis, Jira 或 Trac 等) 或相关开发人员的电子邮件等. 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深色进行标记, 无缺陷模块用浅色进行标记.

(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 (例如噪声移除、特征选择、数据归一化等) 后, 借助特定的建模方法构建出缺陷预测模型. 目前大部分建模方法都基于机器学习方法.

(4) 当面对被测项目内新程序模块时, 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) 模块或无缺陷倾向性 (non defect-proneness) 模块.

针对软件缺陷预测的已有研究工作可以大致分为两类: (1) 一部分研究工作尝试设置与软件缺陷存在强相关性的度量元 [14]. 早期的研究工作主要集中于分析代码模块的规模和复杂度, 重点关注基于代码的软件度量, 例如: McCabe 环路复杂度 [15], Halstead 科学度量 [16] 或 CK 度量元 [17] 等. 近些年来, 更多的研究工作集中于挖掘软件历史存档, 重点关注基于软件开发过程的软件度量, 包括从代码修改角度 [18~20], 开发人员经验角度 [21~23], 程序模块间的依赖性角度 [24, 25] 或项目团队组织构架角度 [26~28] 等出发来设计度量元. (2) 另一部分研究工作重点关注如何提高缺陷预测数据集的质量. 已有研究表明数据集质量对缺陷预测模型的构建具有重要影响. 以研究人员经常使用的 NASA 数据集为例, Shepperd 等 [29] 分析了该数据集的两个不同版本 (分别来自 Promise 库和 NASA 网站), 发现这些数据集中存在诸多质量问题, 例如部分数据取值遗失或不一致, 以及一些实例存在重复等. 除此之外, 他们还发现不同版本的数据集对实证研究结论存在显著的影响, 因此他们认为研究人员在选用 NASA 数据集时, 需要明确使用的数据集版本并详细介绍数据集预处理步骤以确保实证研究可以重现. 目前研究人员重点关注数据集中噪声产生的原因 [30~32]、维数灾难问题 [33~38] 以及类不平衡问题 [39, 40] 等. 论文重点针对部分缺陷预测数据集中的维数灾难问题展开深入的研究.

2.2 特征选择方法及在软件缺陷预测中的应用

程序模块在度量时, 若考虑了大量与代码或开发过程相关的度量元 (即特征), 会使得一些数据集存在维数灾难问题. Hall [41] 发现: 如果一个特征子集里的特征和类标之间强相关, 同时子集中特征之间的关联性很低, 那么该特征子集会取得比较好的预测性能. 因此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. 其中无关特征是指和类标 (即模块是否含有缺陷) 相关性低的特征. 这些特征包含的信息量对采用的数据挖掘算法不能提供任何帮助, 甚至有时会降低预测性能. 例如模块编号这一特征就属于无关特征. 如果一个特征重复了其他单个或多个特征含有的信息, 则称该特征为冗余特征. 例如操作符的数量和操作数的数量这两个特征大致呈正比例关系, 因此这两个特征之间存在一定冗余关系. 冗余特征的存在会违背一些分类器对特征独立性 (例如 Naive Bayes 分类器假设特征之间条件独立) 的假设, 从而降低模型的预测性能.

目前已有的特征选择方法可以简单分为两类: 基于排序的特征选择方法和基于子集搜索的特征选择方法. 其中基于排序的特征选择方法将特征按照与类标之间的相关性从高到低进行排序, 并从中选出最为相关的特征来构建出缺陷预测模型. 这类方法通过分析数据集特征来估算出一个特征与类标间的相关性, 其相关性越高, 表明该特征区分不同类别实例的能力越强. 其计算开销与特征数呈线

性关系, 因此即使在特征数较多的数据集上, 其运行效率也较高. 但这类方法选出的特征子集内通常会含有冗余特征. 例如操作符的数量和操作数的数量这两个特征, 虽然和类标的相关性很高, 但在使用基于排序的特征选择方法时, 这两个特征都会被选择出来, 但这两个特征之间却具有冗余性, 因此这类方法无法达到 Hall [41] 需要的具有低冗余率的特征子集. 而基于子集搜索的特征选择方法 (例如 CFS [41] 和 FCBF [42]) 虽然能够同时考虑特征与类标之间的相关性以及特征之间的冗余性. 但这类方法需要搜索的特征子集空间较大, 尽管存在很多启发式搜索策略 (例如贪心算法或遗传算法等), 但在特征数较多的数据集上, 这类方法的计算开销过大.

目前很多研究人员将特征选择方法应用到软件缺陷预测问题. 在 Menzies 等 [34] 和 Song 等 [35] 提出的通用缺陷预测方法中均包含了基于子集搜索的特征选择方法, 他们在特征子集搜索时分别考虑了两种不同的基于贪心法的搜索策略, 即前向搜索策略和后向搜索策略. Gao 等 [33] 针对一个大规模电信遗留软件系统, 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. 最终结果表明移除 85% 的特征并不会大幅度降低缺陷预测性能, 甚至有时会提高预测性能. Wang 等 [37] 尝试借助集成学习方法将多种特征选择方法进行融合, 他们考虑了 18 种不同的特征选择方法, 最终发现集成少数特征选择方法, 其预测性能甚至要优于集成所有特征选择方法. Khoshgoftaar 等 [38] 将特征选择方法与类不平衡学习方法相结合, 他们发现: 在特征选择的基础上, 若通过采样方法来缓解类不平衡问题, 则可以进一步提高缺陷预测性能. Kim 等 [43] 通过分析源代码、代码修改和修改日志, 抽取出大量与代码修改相关的特征来尝试构建基于代码修改的缺陷预测模型, 但 Shivaji 等 [36] 发现: 上述方法构造出的训练集含有过多的特征 (实证研究对象包含的特征数介于 6127 个到 41942 个之间), 因此存在明显的维数灾难问题. 他们考察了多种不同的特征选择方法. 结果表明大部分时候最多只需使用其中 10% 的特征.

与上述研究工作不同, 为了有效地识别并移除特征集中的冗余特征和无关特征, 论文首次提出了一种基于聚类分析的特征选择方法 FECAR. 该方法包括特征聚类阶段和特征排序阶段. 其中通过分析特征之间的关联性进行聚类分析, 可以将具有冗余关系的特征集中于同一聚类中, 从而可以有效避免传统的基于排序的特征选择方法可能选出的特征之间具有冗余关系的问题.

3 基于聚类分析的特征选择方法 FECAR

3.1 FECAR 方法描述

为了进行特征冗余性分析和相关性分析, 该节首先对特征关联性和特征相关性进行定义, 具体的度量方法将依次在 3.2 和 3.3 小节中进行介绍.

定义 1 (特征关联性 FFC (feature-feature correlation)) 对于任意两个特征 f_i 和 f_j ($i \neq j$), 用 FFC 定义它们之间的关联性, 并借助函数 $C(f_i, f_j)$ 计算出两个特征间的关联性强度.

函数 $C(f_i, f_j)$ 的取值范围是 $[0, 1]$, 其取值越大, 表示特征 f_i 和 f_j 间的特征关联性越高. 若取值为 1, 表示特征 f_i 和 f_j 完全相关, 即两者含有的信息量完全相同. 若取值为 0, 则表示特征 f_i 和 f_j 之间相互独立.

定义 2 (特征相关性 FCR (feature-class relevance)) 用 FCR 定义特征 f_i 与类标之间的相关性, 并借助函数 $R(f_i)$ 计算出特征与类标间的相关性强度.

函数 $R(f_i)$ 的取值范围也是 $[0, 1]$, 其取值越大, 表示特征 f_i 与类标间的相关性越高.

论文提出的特征选择方法 FECAR 包括两个阶段: 特征聚类阶段和特征排序阶段. 其中第一个阶

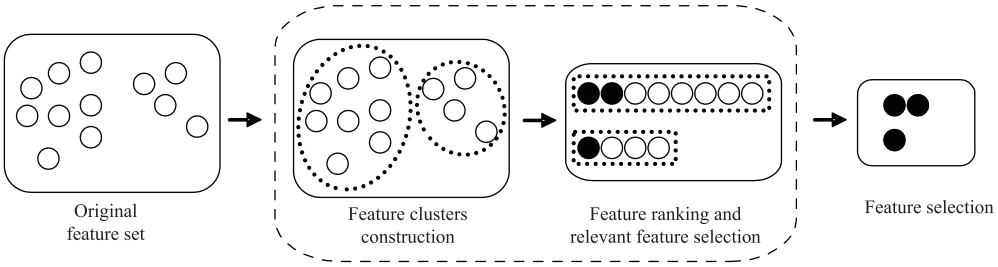

图 2 FECAR 方法的流程图

Figure 2 Flow chart of FECAR framework

段通过分析特征之间的关联性进行聚类分析, 这样可以将具有冗余关系的特征集中于同一聚类中. 随后在第 2 阶段, 对于每一个聚类, 根据特征与类标间的相关性, 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. 图 2 通过一个简单实例, 对 FECAR 方法的执行流程进行解释. 图中空心圆圈代表原始特征 (假设共有 12 个特征), 首先执行聚类分析并生成两个聚类, 随后进行特征选择, 假设从聚类 1 中选出两个特征, 从聚类 2 中选出一个特征, 选出的特征用实心圆圈表示. 最终 FECAR 方法共选出 3 个特征.

算法 1 FECAR 方法

Input:

原有特征集 F, FCR 度量方法 Rel, FFC 度量方法 SU, 选出的特征子集规模 m, 簇的数量 k

Output:

选出的特征子集 S

/* 特征聚类阶段 */

for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    借助 SU 计算特征 f_i 与特征 f_j 之间的关联性.
  end for
end for
构造出矩阵 M, 其中 $M_{i,j}$ 表示 $C(f_i, f_j)$.
for $i = 1$ to n **do**
  借助 Rel 计算特征与类标之间的相关性, 并构造向量 V, 其中 V_i 表示 $R(f_i)$.
end for
根据矩阵 M 和向量 V, 将原有的特征集划分成 k 个簇.

/* 特征排序阶段 */

for $i = 1$ to k **do**
  将簇 C_i 中的特征按照 $R(f_i)$ 从高到低进行排序.
end for
$S \leftarrow \emptyset$
for $i = 1$ to k **do**
  将簇 C_i 中前 $\left\lceil \frac{|C_i| \times m}{n} \right\rceil$ 个特征添加到 S.
end for
return S

因此 FECAR 方法的输入是原有特征集, 指定的 FCR 和 FFC 度量方法, 选出的特征子集规模, 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. 我们使用 SU 来度量两个特征之间的关联性, 使用 IG, CS 或 RF 来度量特征与类标之间的相关性. 具体如算法 1 所示. 接下来, 将对 FECAR

方法中两个阶段的实现细节进行介绍.

3.1.1 特征聚类阶段

聚类分析是根据数据对象间的相似性将其分成簇的过程. 聚类分析认为: 若簇内元素的相似性越高, 簇间元素的相似性越小, 则聚类效果越好. 在 FECAR 中, 聚类分析将特征视为数据对象, 借助 FFC 来计算数据元素间的相似性. FECAR 将原有特征集分成不同的簇, 每个簇里的特征之间关联性较高, 而不同簇间的特征之间关联性则较低. FECAR 采用了 K-Medoids 聚类算法来进行聚类分析. 具体来说: K-Medoids 聚类算法的输入是原始特征集和每对特征之间的关联性, 以及需要生成的簇的数量 K, 输出是 K 个聚成簇的特征子集. 该算法首先为每个簇选择一个代表特征 (即质心); 然后对其余的每个特征, 根据其与每个质心的关联性, 将其分配给关联性最高的质心所代表的簇; 随后不断更新簇的质心, 直至簇内的代表特征保持不变. 具体如算法 2 所示.

算法 2 K-Medoids 聚类算法

1: 选择 K 个特征作为簇的初始代表特征.
2: **repeat**
3: 将每个特征指派到关联性最高的代表特征, 形成 K 个簇.
4: 更新每个簇的代表特征.
5: **until**
6: 每个簇的代表特征不发生变化.

在算法 2 的步骤 1 中, 有多种方法可以选择初始的代表特征. 如果初始特征选择不当, 容易出现算法收敛速度慢或簇的大小不均等问题. 通常情况下, 特征集里的大部分特征和类标之间具有一定的相关性, 所以每个簇的代表特征和类标之间的相关性较高. 因此 FECAR 启发式地选择和类标最相关 (即 FCR 取值最高) 的前 K 个特征作为簇的代表特征, 以加快算法的收敛速度.

在算法 2 的步骤 4 中, 通常做法是计算簇内对象的平均值, 然后选择离平均值最近的数据对象作为新的代表数据. 然而, FECAR 以特征之间的关联性作为聚类的标准, 目的是最大化簇内特征的关联性, 因此不存在数据平均值的问题, 所以我们选择和簇内其他特征关联性最大的特征作为新的代表特征. 假设簇 $C = \{f_{\mathrm{c},1}, f_{\mathrm{c},2}, \ldots, f_{\mathrm{c},n}\}$, 其中 $f_{\mathrm{c},i}$ 表示簇 C 中的第 i 个特征, 令 $F_{\mathrm{c},i}$ 计算出特征 $f_{\mathrm{c},i}$ 和簇内其他特征的相关性之和, 其计算公式表示如下:

$$F_{\mathrm{c},i} = \sum_{1 \leqslant j \leqslant n, j \neq i} C(f_{\mathrm{c},i}, f_{\mathrm{c},j}). \tag{1}$$

最后选出具有最大 $F_{\mathrm{c},i}$ 取值的对应特征作为簇的代表特征.

3.1.2 特征排序阶段

完成特征聚类阶段后, FECAR 将关联性很高的特征分配到同一个簇中, 而不同簇之间的特征关联性则较小. 随后针对每个聚类进行特征选择, 选择时不仅考虑每个簇的规模, 而且还考虑特征和类标之间的相关性. 具体来说, 对于簇 $C = \{f_{\mathrm{c},1}, f_{\mathrm{c},2}, \ldots, f_{\mathrm{c},n}\}$, 我们先根据特征相关性将特征从高到低进行排序, 然后从该簇中选出前 $[\frac{|C| \times m}{M}]$ 个特征. 其中, $|C|$ 表示簇 C 的规模, m 表示最终要选出的特征子集规模, M 表示原始的特征数. 对每个簇进行上述的特征排序和选择操作后, 用各个簇中选择的特征构建出 FECAR 最终选出的特征子集.

通过上述两个阶段, FECAR 最终可以从原始特征集中选出一个相互之间关联性小, 而和类标相关性大的特征子集. 下一节将依次介绍 FECAR 中使用的特征关联性度量方法和特征相关性度量方法.

3.2 特征之间关联性的度量方法

由于缺陷预测数据集中, 特征之间并不具有线性相关性, 因此论文采用非线性的对称不确定性 (简称 SU) 来度量特征之间的关联性.

SU 借助信息论中的熵, 通过衡量两个变量 X 和 Y 分布的差异性来度量这两个变量的关联性, 用 X 和 Y 的熵对它们之间的互信息进行归一化就可以计算出 SU. 将特征视为一个变量, 不同实例上特征的值就是该变量的取值, 因此可以用 SU 来度量两个特征之间的关联性, 其具体计算公式为

$$\mathrm{SU}(X,Y)=2\times\frac{\mathrm{IG}(X|Y)}{H(X)+H(Y)}. \tag{2}$$

SU 的取值范围是 $[0,1]$, 取值越大, 则说明两个特征之间的关联性越强. 其中 $H(X)$ 表示变量 X (即特征) 的熵. 假设 $p(x)$ 表示 X 取值为 x 的先验概率, 则 $H(X)$ 的计算公式为

$$H(X)=-\sum_{x\in X}[p(x)\times\log_2(p(x))]. \tag{3}$$

$\mathrm{IG}(X|Y)$ 称为信息增益, 它表示在给定变量 Y 的情况下, 变量 X 的熵的减少量. 因此在通常情况下, X 不确定性减少的越多, 则 X 和 Y 的依赖性越高, 即它们之间的关联性越强, 其计算公式为

$$\mathrm{IG}(X|Y)=H(X)-H(X|Y), \tag{4}$$

其中 $H(X|Y)$ 衡量的是在给定变量 Y 的情况下, 变量 X 的熵. 假设 $p(y)$ 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, 则 $H(X|Y)$ 的计算公式为

$$H(X|Y)=-\sum_{y\in Y}p(y)\sum_{x\in X}[p(x|y)\log_2(p(x|y))]. \tag{5}$$

3.3 特征与类标间的相关性度量方法

特征相关性 (FCR) 可用于度量特征和类标之间的相关性. 常用的特征相关性度量方法可以简单分为 3 类, 第 1 类是基于信息熵的技术, 包括信息增益 (information gain)、信息增益率 (gain ratio)、对称不确定性 (symmetric uncertainty) 等; 第 2 类是基于实例的技术, 包括 Relief, ReliefF 等; 第 3 类是基于统计信息的技术, 如卡方值 (chi-square) 等. 论文分别从这 3 类方法中选出一种典型的度量方法, 最终选出的方法包括: 信息增益、卡方值和 ReliefF. 下面依次介绍这 3 种特征相关性度量方法.

信息增益衡量的是特征 f 提供的关于类标 c 的信息量, 论文用 $\mathrm{IG}(f)$ 表示特征 f 的信息增益, 取值越大, 表示特征 f 与类标的相关性越高.

卡方值是一种无参数的统计值, 使用卡方检验来验证类标分布与特征 f 的取值是否相关. 该检验的空假设是两者不相关, 随后通过计算观察到的值和空假设成立时的期望值之间的距离, 来衡量空假设成立的可能性. 其距离越大, 表示空假设成立的可行性越小. 具体计算公式表示如下:

$$\chi^2=\sum_{i=1}^{r}\sum_{j=1}^{n_c}\frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}, \tag{6}$$

其中 r 表示一个特征的不同取值数, n_c 表示类别的数量 (在缺陷预测中取值为 2), $O_{i,j}$ 和 $E_{i,j}$ 分别是在类别为 j 的实例中, 特征 f 取值为 i 的实例数和期望数. 卡方值越大, 表示特征值分布和类标分布相关的可能性越大.

ReliefF [44] 是一种基于实例的特征相关性计算方法. 不同于前两种度量方法, RelieF 不是直接计算特征和类标间的相关性, 而是赋予特征一个重要性系数, 用来衡量一个特征区分实例的能力, 然后根据实例的特征值和类标来更新该系数.

通过上述分析, 不难看出: 这 3 种不同类型的相关性度量方法从不同角度出发, 计算出特征与类别之间的相关性. 在 FECAR 中, 可以借助这 3 种不同的度量方法, 来对每个簇中的特征按照其与类标之间的相关性进行排序, 并从每个簇中选出最相关的特征来构建出需要的特征子集. 在随后的实证研究中, 我们将分析不同的特征相关性度量方法对缺陷预测性能的影响.

3.4 FECAR 方法的时间复杂度分析

本节主要对 FECAR 的时间复杂度进行分析. FECAR 的时间消耗主要包括特征关联性的计算, 特征相关性的计算, 特征聚类和簇内特征排序 4 个部分. 假设 N 表示数据集中的实例数量, M 表示数据集上的维度, k 表示 FECAR 聚类簇的数量, 依次分析这 4 个部分的时间复杂度如下:

(1) 计算特征关联性. 由 SU 的计算公式可知, 计算两个变量 SU 的时间复杂度是线性的. 所以在含有 N 个实例的数据集上计算两个特征之间 SU 的时间复杂度是 $O(N)$. 总共有 M 个特征, 因为需要计算两两特征之间的关联性, 总共要计算 $M(M-1)/2$ 次, 所以计算特征关联性的时间复杂度是 $O(N \times M^2)$.

(2) 计算特征相关性. 对于信息增益和卡方值, 计算一个特征的相关性是线性的, 即 $O(N)$, 总共有 M 个特征, 计算它们相关性的时间复杂度是 $O(N \times M)$.

对于 ReliefF, 首先要线性遍历寻找一个实例的最近邻, 时间复杂度是 $O(N)$, 然后更新每个特征的重要性系数, 此时时间复杂度变成了 $O(N \times M)$. 因为 ReliefF 需要多次执行上述过程, 而执行的次数通常和实例的数量成正比, 因此整体的时间复杂度是 $O(M \times N^2)$. 此外, 在寻找最近邻的过程中 ReliefF 还需要知道两个实例间的距离, 可以事先对任意两个实例间的距离进行计算并存储, 在 ReliefF 执行时, 直接查找两对实例间的距离即可. 上述计算距离的时间复杂度是 $O(M \times N^2)$. 因此, ReliefF 整体的时间复杂度是 $O(M \times N^2)$.

(3) 特征聚类. 上文已经分析, 特征聚类主要包括指定特征到关联性最高的质心和更新每个簇的质心两个步骤. 对每一个特征, 指定它到关联性最高的质心的过程就是从 k 个代表特征中找它的最近邻, 因此时间复杂度是 $O(k)$; 因为总共有 M 个特征, 所以这一步的时间复杂度是 $O(k \times M)$. 假设某个簇含有 x 个元素, 在更新簇内的质心时, 需要计算簇内每个特征到簇内其他特征的关联性之和, 时间复杂度是 $O(x^2)$. 总共 k 个簇, 每个簇的期望大小是 M/k, 所以更新代表特征这一步的时间复杂度是 $O(k \times (M/k)^2) = O(M^2/k)$. 实验发现, 在 FECAR 中, K-Medoids 算法迭代常数次可以收敛. 因此聚类的时间复杂度是 $O(M^2/k + k \times m)$.

(4) 簇内特征排序. 可以使用快速排序算法对簇内的特征按照它们与类标的相关性进行排序, 时间复杂度是 $O(k \times (M/k)\log(M/k)) = O(M \times \log(M/k))$.

通常来看, 实例数量 N 要远大于特征数量 M, 而 M 的数量要大于聚类的数量 k. 相对于 N, 可以将 M 和 k 看作常数, 则 FECAR 关于实例数量的时间复杂度是 $O(N \times \log N)$(以信息增益和卡方值度量相关性) 和 $O(N^2)$(以 ReliefF 度量相关性).

4 实证研究

本节对 FECAR 方法的有效性进行实证研究. 下面是实证研究需要回答的 3 个实验问题: (1) 相对于仅作特征排序的特征选择方法, FECAR 是否能够提高缺陷预测模型的性能? (2) 相对于仅作特征排序的特征选择方法, FECAR 是否能够降低选出的特征子集的冗余度? (3) 特征与类标间的相关性度量方法的选择对 FECAR 是否存在影响? 和其他特征选择方法比较, FECAR 方法是否能够具有更好的性能?

4.1 评测对象

论文选择了来自实际软件项目的数据集来进行实证研究. 选择的项目包括 Eclipse 项目和 NASA 项目. 其中 Eclipse 项目的数据集可以从 Promise 库中下载, NASA 项目中的数据集 kc1 同样可以从 Promise 库中获取, 而 NASA 项目的其他数据集则从 MDP 数据集中获取. 其中 Eclipse 数据集包含 3 个版本 (分别是 Eclipse 2.0, Eclipse 2.1 和 Eclipse 3.0), 以软件发布后的缺陷数为预测目标, 包含的特征包括代码行数、环路复杂度和类数量等代码复杂度特征, 以及基于语法树的特征等. 为提高数据集质量, 我们对 Eclipse 数据集进行了一些预处理, 包括: (1) 移除所有非数值型的特征; (2) 移除取值完全相同的特征, 因为这些特征不能提供任何分类信息; (3) 原有数据集的类标表示的是软件发布后的缺陷数, 针对该问题, 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. NASA 项目数据集主要考虑的特征包括代码行数、McCabe 环路复杂度 [15]、Halstead 科学度量 [16] 等. 目前 NASA 项目总共有 13 个数据集, 其中 mc1 和 pc2 这两个数据集上存在严重的类不平衡问题, 而 jm1 数据集含有的特征数很少. 因此最终选择了 10 个数据集. 因为 FECAR 在计算特征关联性时, 需要对特征进行离散化, 因此我们使用了 MDL 方法 [45] 对 Eclipse 和 NASA 项目中的连续型属性进行了离散化处理. 表 1 中总结了论文实证研究选用的所有数据集的统计信息.

4.2 评测指标

论文借助 AUC (area under ROC curve) 值来评估不同缺陷预测模型的预测性能. 其中 ROC 曲线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 (即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 值则对应的是 ROC 曲线下的面积, 其取值越接近于 1, 则代表对应的分类器性能越好. 目前 AUC 值被很多研究人员用于评测缺陷预测模型的性能 [33, 35, 38, 39, 46].

为了比较不同特征选择方法选出的特征子集内含有的冗余信息, 论文提出了一种新的度量指标: 冗余率 (redundancy rate). 给定一个特征集 S, 借助 $\mathrm{RR}(S)$ 计算该特征集的冗余率, 其计算公式为

$$\mathrm{RR}(S) = \frac{2}{1 + e^{-\lambda \times \mathrm{sum}C(S)}} - 1, \tag{7}$$

其中 $\mathrm{sum}C(S)$ 函数计算特征集 S 中特征之间的关联性之和, 特征之间的关联性越大, $\mathrm{sum}C(S)$ 越大. 考虑到随机因素可能造成特征之间的弱关联, 而弱关联性不会造成特征之间的信息冗余, $\mathrm{sum}C(S)$ 只考虑大于某个阈值 α 的关联性. 由于特征关联性的取值范围是 $[0, 1]$, 我们将 α 设为 0.5. $\mathrm{sum}C(S)$ 的计算公式如下:

$$\mathrm{sum}C(S) = \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} (w_{i,j} \times C(f_i, f_j)), \tag{8}$$

表 1 数据集统计信息

Table 1 Statistical information of datasets

Dataset	Level	#Features	#Entities	#Defective entities
Eclipse 2.0	File	155	6729	975 (14.49%)
Eclipse 2.1	File	155	7888	854 (10.83%)
Eclipse 3.0	File	155	10593	1568 (14.80%)
cm1	Method	37	505	48 (9.50%)
kc1	Class	86	145	60 (41.37%)
kc3	Method	39	458	43 (9.39%)
kc4	Method	14	125	61 (48.80%)
mc2	Method	39	161	52 (32.30%)
mw1	Method	37	403	31 (7.69%)
pc1	Method	37	1107	76 (6.87%)
pc3	Method	37	1563	160 (10.24%)
pc4	Method	37	1458	178 (12.21%)
pc5	Method	39	17186	516 (3.00%)

其中 $w_{i,j}$ 的计算公式如下:

$$w_{i,j}=\begin{cases}1,\ C(f_i,f_j)\geqslant\alpha\wedge i\neq j,\\ 0,\ \text{otherwise.}\end{cases}\tag{9}$$

由于 sum$C(S)$ 的取值范围从 0 到正无穷, 我们使用修改后的 sigmoid 函数

$$y=\frac{2}{1+e^{-x}}-1\tag{10}$$

将其归一化到 0 到 1 之间, 以方便比较不同特征子集的冗余度. 修改后的 sigmoid 函数对应的曲线形状如图 3 所示, 根据图 3 可知, 该函数首先有个明显的上升段, 随后进入平稳段. 在平稳段, y 值 (即 RR(S)) 变化较小, 因此并不利于冗余度的对比. 为了方便对各个方法的冗余率进行比较, 我们尽量将 RR(S) 值控制在公式 (10) 的上升段, 为此我们使用了一个参数 λ 来控制 x 值 (即 sum$C(S)$) 的范围. 除了特征之间关联性的强弱, sum$C(S)$ 的取值还跟特征数有关, 为了使不同特征选择方法生成的特征子集的冗余度具有一定可比性, 我们将 λ 值设置为原有特征集 S_0 规模的倒数 (即 $1/|S_0|$). 因此最终特征选择方法选出的特征子集的冗余率的计算式如式 (7) 所示.

4.3 显著性检验

根据 Lessmann 等 [46] 的建议, 论文采用 Friedman 检验来比较不同特征选择方法的效果差异, 该检验是一种无参数的假设检验方法, 它的空假设是各种方法的效果相同, 通过比较各个数据集上不同算法的排名, 可以判断各种方法之间是否存在显著性差异.

假设要在 N 个数据集上比较 k 个方法的实验效果. 对每个数据集, Friedman 检验按照方法的效果对方法进行排名, 表现最好的方法排名第一, 然后是第二, 依次类推. 如果两个方法效果相同, 则它们的名次取两者的平均值. Friedman 检验值的计算公式如下:

$$\gamma_F^2=\frac{12N}{k(k+1)}\left[\sum_{i=1}^{k}R_i^2-\frac{k(k+1)^2}{4}\right],\tag{11}$$

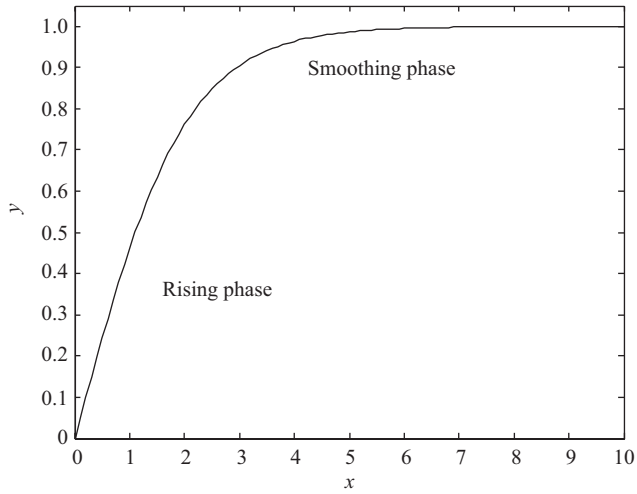

图 3　式 (10) 对应的曲线形状

Figure 3　The curve of formula (10)

其中 $R=\frac{1}{N}\sum_i r_i^j$ 表示方法 j 在所有数据集上的平均排名, 其中 r_i^j 表示方法 j 在数据集 i 上的排名.

Friedman 检验服从自由度为 $k-1$ 的卡方分布, 检验值越大, 空假设成立的可能性越小. 如果空假设成立, 则各个方法的效果没有差异. 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. 论文使用 Nemenyi 检验来比较两个方法的差异. 如果两个方法排名的差异超过临界值 CD, 则 Nemenyi 检验认为空假设不成立, 即这两个方法具有显著性差异. 临界值 CD 的计算公式为

$$\mathrm{CD}_\alpha = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{12}$$

其中 q_α 由 t 分布的值除以 $\sqrt{2}$ 所得.

4.4　实验设计

4.4.1　特征选择方法

FECAR 在设计时综合考虑了特征之间的关联性和特征与类标间的相关性, 旨在有效识别并移除数据集中的冗余特征和无关特征, 为了回答上述实验问题, 论文选取了 3 类代表性特征选择方法, 作为 FECAR 的比较对象: (1) FullSet, (2) IG, CS 和 RF, (3) CFS, FCBF 和 Consist.

(1) FullSet 方法不做特征选择, 即使用原始特征集来构建缺陷预测模型. 与该方法进行对比, 可以研究使用 FECAR 做特征选择是否可以提高缺陷预测性能.

(2) IG, CS 和 RF 这 3 种方法均属于基于排序的特征选择方法, 在排序时分别考虑了信息增益、卡方值和 ReliefF 等相关性度量方法. 而 FECAR 先对特征进行聚类, 然后再使用信息增益, 卡方值和 ReliefF 对每个簇中的特征进行排序, 我们将使用 3 种不同特征相关性度量方法的 FECAR 分别称为 CIG, CCS 和 CRF. 通过将 CIG 和 IG, CCS 和 CS, CRF 和 RF 进行对比, 我们可以研究特征聚类对所选特征子集的冗余度的影响, 以及对缺陷预测模型性能的影响.

(3) CFS [41], FCBF [42] 和 Consist [47] 分别是 3 种经典的特征选择方法. 其中 CFS 特征选择方法同时考虑了特征与类标间的相关性以及特征彼此间的关联性, CFS 使用 Best-First 搜索策略来寻找高相关性低关联性的特征子集. FCBF 也同时考虑了特征与类标的相关性和特征间的关联性, 该方法每

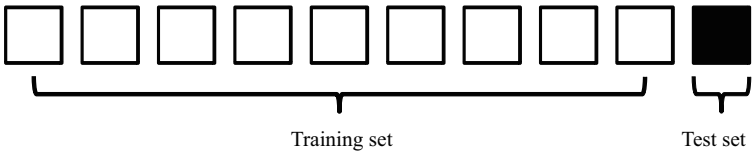

图 4 十折交叉验证

Figure 4 10-fold cross validation

次评估一个特征, 因此不需要成对地进行相关性分析. 根据 Yu 等 [42] 的建议, 将 FCBF 中的相关性阈值设置为 $\lfloor M/\log M \rfloor$, 其中 M 是原始的特征数. Consist 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. Consist 也使用 Best-First 搜索策略进行搜索. 我们将上述方法与 FECAR 进行比较, 研究它们在构建缺陷预测模型后的性能差异.

4.4.2 不同特征选择方法的参数设定

FECAR 的参数主要包括聚类的簇数量和最终选择的特征子集规模. 根据 Gao 等 [33] 的建议, 最终从原始特征集中选择 $\lceil \log_2 M \rceil$ 个特征; 同时在聚类分析时, 我们将簇的数量启发式地设定为 $\lceil \log_2 M/2 \rceil$. 这里 M 表示原始特征数. 对于 IG, CS 或 RF 等基于排序的特征选择方法, 为了使其与 CIG, CCS 或 CRF 具有对比性, 我们选出同等数量的特征.

4.4.3 分类模型

论文采用两种分类方法来构建缺陷预测模型: Bayes 方法中的 Naive Bayes (NB) 和决策树方法中的 C4.5. 与其他 Bayes 方法不同的是, NB 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分裂节点特征, 同时使用剪枝方法来防止过拟合问题的出现. NB 分类器和 C4.5 分类器是缺陷预测领域应用最广泛的分类算法 [34,35,39,46]. 在实证研究中, 我们使用 Weka 软件包实现 NB 和 C4.5 分类器, 并采用默认的参数设置.

4.4.4 实验流程

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 (如图 4 所示). 上述过程重复 10 次 (即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.

在每一轮中, 首先不同的特征选择方法根据训练集确定要选择的特征子集, 然后根据该特征子集同时对训练数据和测试数据进行降维处理 (即只保留该特征子集中的特征), 其次根据降维后的训练数据构建缺陷预测模型, 并将该模型应用到测试数据上取得性能结果. 具体过程如图 5 所示.

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.

4.5 结果分析

本节将整理并分析实证研究中的结果以回答实验设计问题. 对每一组实验结果, 我们首先执行 Friedman 检验来验证它们之间是否存在显著性差异, 然后采用 Nemenyi 检验来比较不同方法间的效

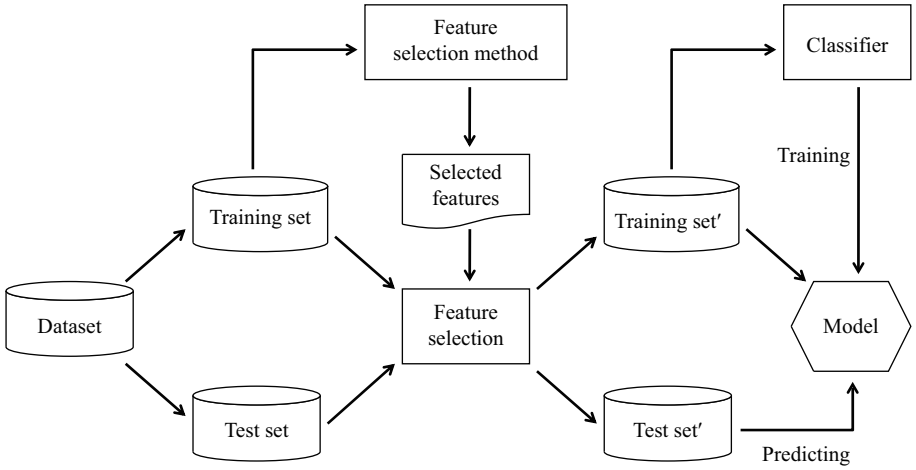

图 5　每一轮的执行过程

Figure 5　Execution process of each fold

果差异. 为了进一步发现两个方法间的差异, 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, Loss. 分别表示方法 A 好于、等于和差于方法 B 的数据集的数量. 下面将依次对各个研究问题进行深入分析.

4.5.1　缺陷预测模型的性能

相对于仅作特征排序的特征选择方法, FECAR 是否能够提高缺陷预测模型的性能? 表 2 和 3 分别总结了基于 NB 和 C4.5 分类器, 使用不同特征选择方法后的预测性能. 对 NB 和 C4.5 分类器的预测结果进行了 Friedman 检验, 得到的 p 值分别是 1.76E−07 和 4.72E−6, 这表明各种方法的效果具有统计性差异.

为了进一步对各种方法进行两两比较, 我们进行了 Nemenyi 检验 (将显著性水平设置为 0.05). 由于共有 10 种特征选择方法和 13 个数据集, 按照式 (12) 可以计算出 Nemenyi 检验的临界值是 3.757. 因此如果两个方法的排名差异超过 3.757, 则这两个方法有显著性差异. 对表 2 和 3 的结果进行 Nemenyi 检验后, 检验结果如图 6 所示. 其中横坐标表示方法的名次, 每个三角点表示一个方法的平均排名. 纵坐标标记的是相应的方法名称, 各方法按照排名依次纵向排开. 三角点后面的线段表示 Nemenyi 检验的临界值, 这里其长度是 3.757; 如果方法 A 的排名在方法 B 临界值线段的右侧, 则说明方法 A 显著差于方法 B. 因为论文中, 我们主要对 FECAR 特征选择方法和其他方法的效果进行对比, 因此在 CIG, CCS 和 CRF 方法的临界值线段处做垂直于 x 轴的虚线, 以方便将它们与其他方法进行区别.

研究表 2 和 3 的第 2∼7 列的数据, 可以发现:

(1) 若选择 NB 作为分类器, 相对于只做特征排序的特征选择方法, 特征聚类可以有效提高软件缺陷预测性能. 即相对于 IG, RF 和 CS, 进行特征聚类后, CIG, CRF 和 CCS 的预测性能分别提高了 1.78%, 2.56% 和 0.77%. 表 4 中的 Win/Draw/Loss 分析表明, 在大部分数据集上, FECAR 方法优于仅作特征排序的方法. 但图 6(a) 中的 Nemenyi 检验却表明 CIG 和 IG, CRF 和 RF 以及 CCS 和 CS 的预测性能不存在显著性差异.

(2) 若选择 C4.5 作为分类器, 相对于只做特征排序的特征选择方法, 特征聚类同样可以有效提高软件缺陷预测性能. 即 CIG, CRF 和 CCS 相对于 IG, RF 和 CS 的预测性能分别提高了 6.67%, 5.37%

表 2 基于 Naive Bayes 分类器, 不同特征选择方法的 AUC 值

Table 2 The AUC of different feature selection methods based on Naive Bayes classifier

Dataset	AUC									
	FullSet	IG	RF	CS	CIG	CRF	CCS	CFS	FCBF	Consist
Eclipse 2.0	0.795	0.788	0.716	0.774	0.808	0.750	0.801	0.810	0.708	0.803
Eclipse 2.1	0.746	0.758	0.660	0.760	0.770	0.726	0.769	0.745	0.723	0.762
Eclipse 3.0	0.762	0.762	0.698	0.756	0.775	0.717	0.769	0.756	0.758	0.769
cm1	0.756	0.772	0.750	0.776	0.779	0.764	0.759	0.763	0.705	0.598
kc1	0.796	0.808	0.813	0.793	0.816	0.825	0.808	0.820	0.776	0.813
kc3	0.808	0.800	0.799	0.813	0.822	0.802	0.805	0.797	0.798	0.795
kc4	0.757	0.764	0.746	0.765	0.781	0.765	0.775	0.751	0.724	0.770
mc2	0.719	0.661	0.663	0.682	0.713	0.697	0.710	0.657	0.605	0.679
mw1	0.744	0.785	0.733	0.774	0.794	0.770	0.791	0.780	0.765	0.789
pc1	0.757	0.768	0.724	0.749	0.784	0.717	0.759	0.790	0.754	0.743
pc3	0.773	0.791	0.751	0.787	0.799	0.772	0.795	0.788	0.746	0.743
pc4	0.842	0.835	0.807	0.837	0.826	0.807	0.797	0.816	0.780	0.839
pc5	0.939	0.943	0.930	0.925	0.948	0.924	0.936	0.883	0.779	0.935
Average	0.784	0.787	0.753	0.784	0.801	0.772	0.790	0.781	0.740	0.772

表 3 基于 C4.5 分类器, 不同特征选择方法的 AUC 值

Table 3 The AUC of different feature selection methods based on C4.5 classifier

Dataset	AUC									
	FullSet	IG	RF	CS	CIG	CRF	CCS	CFS	FCBF	Consist
Eclipse 2.0	0.671	0.745	0.773	0.740	0.769	0.771	0.770	0.707	0.715	0.723
Eclipse 2.1	0.590	0.700	0.566	0.705	0.738	0.736	0.733	0.718	0.736	0.668
Eclipse 3.0	0.637	0.732	0.679	0.722	0.762	0.705	0.756	0.704	0.750	0.705
cm1	0.528	0.540	0.524	0.538	0.589	0.584	0.578	0.548	0.531	0.525
kc1	0.702	0.727	0.711	0.704	0.750	0.724	0.722	0.715	0.742	0.701
kc3	0.578	0.599	0.627	0.614	0.699	0.625	0.652	0.606	0.624	0.611
kc4	0.775	0.733	0.761	0.729	0.802	0.776	0.829	0.754	0.740	0.765
mc2	0.636	0.594	0.583	0.593	0.621	0.564	0.608	0.582	0.581	0.589
mw1	0.569	0.614	0.577	0.593	0.655	0.575	0.569	0.570	0.530	0.602
pc1	0.650	0.725	0.513	0.687	0.752	0.585	0.674	0.677	0.592	0.672
pc3	0.635	0.536	0.535	0.544	0.620	0.586	0.599	0.590	0.517	0.615
pc4	0.760	0.854	0.564	0.857	0.873	0.618	0.860	0.866	0.859	0.771
pc5	0.774	0.890	0.867	0.894	0.907	0.870	0.894	0.853	0.856	0.821
Average	0.654	0.692	0.637	0.686	0.734	0.671	0.711	0.684	0.675	0.675

和 3.64%. 表 4 中的 Win/Draw/Loss 分析表明: 在 13 个数据集中的大部分数据集上, FECAR 方法优于仅作特征排序的方法. 图 6(b) 中的 Nemenyi 检验表明: 除了 CIG 和 IG, CRF 和 RF 以及 CCS 和 CS 的预测性能不存在显著性差异.

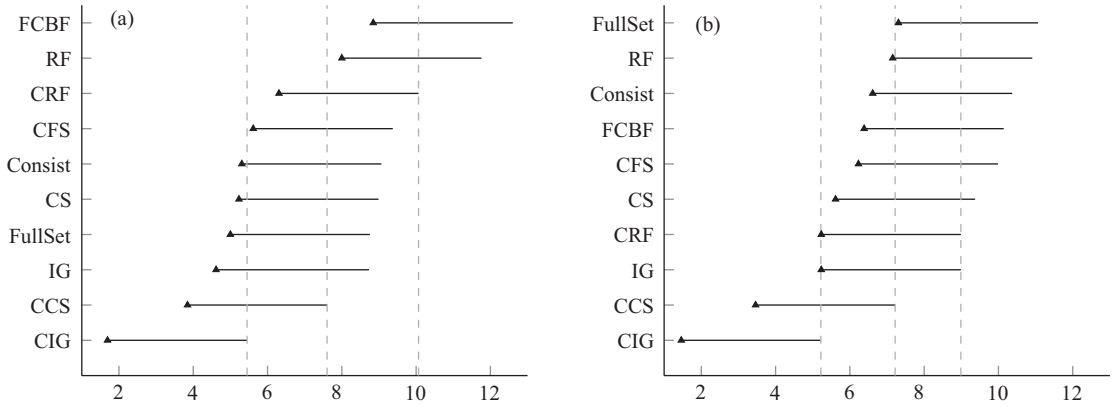

图 6　针对各个特征选择方法的 Nemenyi 检验结果

Figure 6　Nemenyi test for different feature subset selection methods based on (a) NB classifier, and (b) C4.5 classifier

表 4　FECAR 和基于排序的特征选择方法的 Win/Draw/Loss 分析

Table 4　Win/Draw/Loss analysis between FECAR and ranking-based feature selection methods

Classifier	Win/Draw/Loss record of		
	CIG vs IG	CRF vs RF	CCS vs CS
NB	12/0/1	10/1/2	10/0/3
C4.5	13/0/0	9/0/4	11/0/2

表 5 分析了论文实证研究中选择的 9 种 (不包括 FullSet) 特征选择方法在每个数据集上最终选出的特征数占原有特征数的比例. 因为 IG, RF 和 CS 3 种方法选择的特征数量相同, 所以这里将这 3 种方法的选择结果放于同一列中, 即 IG 列. 基于同一原因, 也将 CIG, CRF 和 CCS 这 3 种方法的选择结果放于同一列中, 即 CIG 列.

通过分析表 5, 可以看出: 基于排序的特征选择方法 IG, RF 和 CS 选出的特征比例均值是 13.8%. 为了使实验具有可比性, 我们设定 FECAR 保留的特征数量和基于排序的特征选择方法保留的特征数量相同. 但如第 3 节所述, FECAR 从每个簇中选择的特征数量和该簇的大小有关. 为了防止一个簇因为太小而导致没有特征被选中, 实验中我们约定 FECAR 从每个簇中至少需要选出一个特征. 因此, FECAR 最终选出的特征比例略高于基于排序的特征选择方法 (即 13.85%), 但两者之间非常接近.

基于上述分析, 我们认为在选择的特征数几乎相等的情况下, 不论选择哪种特征与类标间的相关性度量方法 (即 IG, RF 或 CS), 与基于排序的特征选择方法相比, FECAR 通过考虑特征聚类可以有效提高缺陷预测的性能.

4.5.2　特征子集的冗余度

相对于只做特征排序的特征选择方法, FECAR 是否可以降低所选特征子集的冗余度? 为了识别出更多的冗余特征, 相对于只做特征排序的方法, FECAR 在做特征选择时多了一个特征聚类阶段. 因此, 这里将比较 FECAR 和相应的基于排序的特征选择方法所选特征子集的冗余率. 即, 比较 IG 和 CIG, RF 和 CRF 以及 CS 和 CCS 所选特征子集的冗余率. 因为我们期望通过降低所选特征子集的冗余信息来提高缺陷预测模型的预测性能, 所以我们还将分析特征子集的冗余度和该特征子集上建立的缺陷预测模型的预测性能之间的关系.

表 5 不同特征选择方法选出的特征比例

Table 5 Proportion of selected features by different feature selection methods

Dataset	Proportion of selected features				
	IG (RF, CS)	CIG (CRF, CCS)	CFS	FCBF	Consist
Eclipse 2.0	5.16	5.16	22.00	2.26	10.19
Eclipse 2.1	5.16	4.52	9.61	2.45	16.26
Eclipse 3.0	5.16	5.35	17.94	3.23	17.10
cm1	16.22	16.49	20.54	3.78	15.68
kc1	8.14	8.14	9.65	1.16	11.05
kc3	15.38	15.38	14.62	3.85	32.82
kc4	28.57	28.57	30.71	7.86	31.43
mc2	15.38	15.38	20.77	5.38	22.05
mw1	16.22	16.49	18.11	3.51	20.00
pc1	16.22	16.22	15.68	4.86	42.16
pc3	16.22	16.22	21.62	2.70	46.22
pc4	16.22	16.76	6.22	5.41	53.78
pc5	15.38	15.38	8.46	2.56	34.62
Average	13.80	13.85	16.61	3.77	27.18

表 6 不同特征选择方法所选择的特征子集的冗余率

Table 6 Redundancy rate of feature subsets by different feature selection methods

Dataset	Redundancy rate					
	IG	RF	CS	CIG	CRF	CCS
Eclipse 2.0	0.581	0.122	0.533	0.526	0.116	0.360
Eclipse 2.1	0.457	0.008	0.416	0.354	0.108	0.364
Eclipse 3.0	0.565	0.117	0.557	0.467	0.086	0.373
cm1	0.109	0	0.135	0.094	0.011	0.115
kc1	0.549	0.093	0.621	0.610	0.187	0.564
kc3	0.410	0.022	0.144	0.396	0.054	0.218
kc4	0.212	0.045	0.225	0.210	0.166	0.227
mc2	0.181	0	0.179	0.167	0.001	0.118
mw1	0.149	0	0.162	0.180	0	0.132
pc1	0.049	0.006	0.019	0.024	0.009	0.055
pc3	0.041	0	0.020	0.086	0.001	0.050
pc4	0.099	0	0.222	0.067	0.020	0.096
pc5	0.274	0.022	0.369	0.241	0.044	0.264
Average	0.283	0.034	0.277	0.263	0.062	0.226

表 6 分析了 FECAR 和仅基于排序的特征选择方法在 13 个数据集上所选出的特征子集的冗余率. 对上述结果做 Friedman 检验得到的 p 值是 2.49E−09, 这表明各个方法之间存在显著性差异. 表 7 进一步对不同方法所选出的特征子集的冗余率进行 Win/Draw/Loss 分析.

表 7　对不同方法选出的特征子集的冗余度的 Win/Draw/Loss 分析

Table 7　Win/Draw/Loss analysis on redundancy rate of feature subsets of different methods

IG vs CIG	RF vs CRF	CS vs CCS
3/0/10	11/0/2	4/0/9

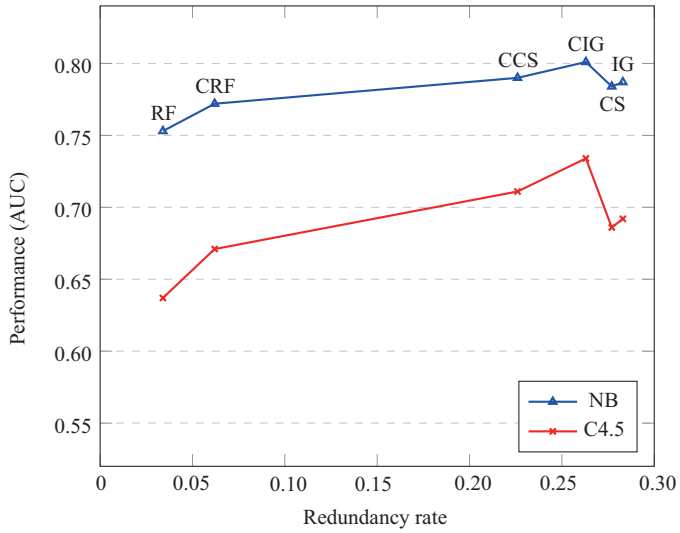

图 7　(网络版彩图) 特征子集冗余率和预测模型的 AUC 值的关系

Figure 7　(Color online) Relationship between redundancy rate of feature subset and AUC of model performance

从表 7 可以看出, CIG 方法选出的特征子集的冗余率在多数情况下要低于 IG 方法选择的特征子集. 通过分析表 6 中这两种方法的冗余率均值, CIG 方法同样低于 IG 方法. 对比 CCS 方法和 CS 方法, 可以得到同样的结论. 因此当采用 IG 或 CS 作为特征相关性度量方法时, FECAR 选出的特征子集的冗余率要低于仅基于排序的特征选择方法, 即进一步考虑特征聚类可以有效降低选出的特征子集的冗余率. 其原因可总结如下: IG 和 CS 这两种度量方法均根据特征数据分布与类标分布的相似性来衡量特征与类标间的相关性, 其计算方式与特征之间关联性的度量方法 SU 相似. 因此, 如果特征 1 和 2 与类标的 IG 值或 CS 值较大, 则特征 1 与 2 之间的关联性 (即 SU 值) 也会较高. 因此若通过 IG 或 CS 选择和类标最为相关的特征, 通常在选出的特征子集中会含有较多的冗余特征.

但与 IG 和 CS 相反, 使用 RF 作为特征相关性度量方法时, 冗余率不但没有因为特征聚类而降低, 反而有所提升. 通过分析表 6, 我们发现 RF 选出的特征子集的冗余率非常低, 平均值远低于 IG 和 CS 所选出的特征子集的冗余率. 具体来说, 在 13 个数据集上, RF 在其中 5 个数据集上选出的特征子集的冗余率是 0. 上述发现与 Kononenko [48] 的研究结论一致, 他们认为 RF 可以有效地处理相互独立和相互依赖 (即冗余度高) 的特征. 因此, RF 选出的特征子集有较低的冗余率. 而 FECAR 将关联性很高的特征聚到了同一个簇中, 且需要从每个簇中选出至少一个特征, 因此最终导致了选出的特征子集的冗余率提高.

基于上述分析, 我们认为当使用 IG 或 CS 作为特征相关性度量方法时, 特征聚类可以有效地降低所选出的特征子集的冗余率; 而使用 RF 这一种可以有效处理冗余特征的度量方法时, 特征聚类反而会提高所选出的特征子集的冗余率.

图 7 显示了 IG, RF, CS 和 CIG, CRF, CCS 这 6 种特征选择方法在 13 个数据集上所选出的特

表 8 不同 FECAR 方法间的 Win/Draw/Loss 分析

Table 8 Win/Draw/Loss analysis among different FECAR methods

Classifier	Win/Draw/Loss record of		
	CIG vs CRF	CIG vs CCS	CRF vs CCS
NB	12/0/1	13/0/0	3/0/10
C4.5	12/0/1	11/0/2	5/0/8

征子集的平均冗余率与基于这些特征子集构建的预测模型的效果间的关系. 从图 7 中, 我们可以发现: 随着冗余率的提高, 缺陷预测模型的预测性能开始提升; 当冗余率达到一定程度后, 缺陷预测性能开始下降. 因此, 冗余率过高或过低都会影响缺陷预测性能.

但在第 1 个研究问题中, 我们发现: 不论是选出高冗余率特征子集的 IG 和 CS 方法, 还是选出低冗余率特征子集的 RF 方法, 相对于仅作特征排序的方法, 通过特征聚类均可以提高缺陷预测模型的预测性能. 其原因是: (1) 仅基于排序的特征选择方法, 若选择 IG 和 CS, 则选出的特征子集的冗余率较高, 因此会存在一些冗余信息, 而借助特征聚类, 则可以使得相互之间关联性低的特征被选出, 从而增加了选出的特征子集的信息量, 最终提高了缺陷预测性能. (2) 仅基于排序的特征选择方法, 若选择 RF, 虽然可以选出冗余率低的特征子集, 但过低的冗余率会造成与类标相关性低的特征被选出. 因此会造成特征子集中可用于缺陷预测的信息量太少, 造成缺陷预测性能较差. 而特征聚类则可以提高 RF 所选出的特征子集的冗余率, 在一定程度上缓解了上述问题, 从而可以有效提高模型的预测性能.

4.5.3 特征选择方法的性能比较

特征与类标间的相关性度量方法的选择对 FECAR 是否存在影响? 和其他特征选择方法比较, FECAR 方法是否能够具有更好的性能? 表 8 借助 Win/Draw/Loss 对 CIG, CRF 和 CCS 这 3 种方法的缺陷预测性能进行了两两分析. 具体来说: (1) 基于 NB 分类器, CIG 在 13 个数据集上均要优于 CCS, 在 12 个数据集上要优于 CRF. 其中 CIG 平均比 CRF 和 CCS 分别高 3.76% 和 1.38%. 图 6(a) 中的 Nemenyi 检验发现 CIG 要显著优于 CRF, 而 CIG 和 CCS 的排名差异是 3, 所以两者间的差异不具有显著性差异. CRF 在 FECAR 的 3 种方法中效果最差. RF 相关性度量方法属于基于实例的特征选择方法, 通过寻找最近邻来调整每个特征的重要性系数. 因为无关特征的存在会影响到实例间距离的计算, 并随后影响近邻的寻找, 从而使得 RF 作为相关性度量方法的 FECAR 效果较差. (2) 基于 C4.5 分类器, CIG 同样是效果最好的方法, 其均值比 CRF 和 CCS 分别高 9.39% 和 3.23%. 基于上述分析, 可以看出使用不同的特征相关性度量方法, FECAR 的效果并不相同. 其中使用 IG 作为相关性度量方法可以使得 FECAR (即 CIG) 取得最好的效果. 下面我们将基于 CIG, 进一步比较 FECAR 和常用的特征选择方法对缺陷预测性能的影响.

表 9 是 CIG 和其他经典特征选择方法在缺陷预测性能上的 Win/Draw/Loss 分析. 比较经典的方法包括: FullSet, CFS, FCBF 和 Consist.

若选择 NB 作为分类器: (1) 从表 2 中发现, CIG 平均比 Fullset 高 2.17%. 通过 Win/Draw/Loss 分析 (见表 9) 表明, CIG 在 11 个数据集上效果要优于 FullSet. 这表明使用 CIG 做特征选择可以提高 NB 分类器的缺陷预测性能. (2) 从表 2 中可以发现, 与 CFS, FCBF 和 Consist 相比, CIG 平均可以提高 2.56%, 8.24% 和 3.76%. 表 9 中的 Win/Draw/Loss 分析也表明 CIG 在大多数情况下要优于这 3 种方法. 图 6(a) 中的 Nemenyi 检验同样发现 CIG 对 CFS, FCBF 和 Consist 的优势具有显著性.

若选择 C4.5 作为分类器: (1) 从表 3 中发现, CIG 平均比 FullSet 高 12.23%. 通过 Win/Draw/Loss

表 9 CIG 和其他经典特征子集选取方法的 Win/Draw/Loss 分析

Table 9 Win/Draw/Loss analysis between CIG and other classical feature selection methods

Classifier	Win/Draw/Loss record of			
	CIG vs FullSet	CIG vs CFS	CIG vs FCBF	CIG vs Consist
NB	11/0/2	10/0/3	13/0/0	12/0/1
C4.5	11/0/2	13/0/0	13/0/0	13/0/0

分析 (见表 9) 表明, CIG 在 11 个数据集上的效果要优于 FullSet. 除此之外, 通过图 6(b) 的 Nemenyi 检验发现, CIG 要显著优于 FullSet. 这表明使用 CIG 做特征选择可以有效提高 C4.5 分类器的缺陷预测性能. (2) 从表 3 中发现: 与 CFS, FCBF 和 Consist 相比, CIG 平均可以提高 7.30%, 8.74% 和 8.74%. 通过 Win/Draw/Loss 分析 (见表 9) 也显示 CIG 在大多数情况下优于这 3 种方法. 图 6(b) 中的 Nemenyi 检验同样发现 CIG 对 CFS, FCBF 和 Consist 的优势具有显著性.

随后我们分析不同方法选出的特征比例, 从表 5 中可以看出, CFS 和 Consist 选出的特征比例远高于 FECAR 选出的特征比例, 分别是 16.61% 和 27.18%. 尽管 FCBF 选出的特征比例仅为 3.77%, 要小于 FECAR 选出的特征比例, 但通过表 2 和 3, 可以看出 FCBF 的效果要远差于 FECAR.

基于上述分析, 我们认为使用信息增益作为特征相关性度量方法的 FECAR (即 CIG) 可以有效提高缺陷预测模型的预测性能. 而且, 使用 CIG 做特征选择后预测性能要好于其他经典的特征选择方法, 例如 CFS, FCBF 和 Consist.

4.6 有效性影响因素分析

本节主要讨论可能影响实证研究有效性的一些影响因素. 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. 论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. 外部有效性主要涉及到实证研究得到的结论是否具有一般性. 论文选用了软件缺陷预测研究中常采用的 Eclipse 项目数据集和 NASA 项目数据集 [34, 35, 37, 39, 40, 46, 49], 因此可以保证研究结论具有一定的代表性. 同时选择的分类器构造方法也是软件缺陷预测研究中经常使用的方法 [34, 35, 39, 46]. 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, 因此考虑了 AUC 这一重要评测指标. 除此之外, 我们还提出了一种新的指标 RR, 可以有效评估选出的特征子集的冗余度.

为了进一步验证所提出方法的有效性, 我们基于 F-measure 评价指标进行了性能分析, 其具体计算公式请参考文献 [50], 表 10 是 FECAR 和其他经典特征子集选取方法的 F-measure 实验结果. 表中第 2, 3 列和第 4, 5 列分别是选择 NB 作为分类器和选择 C4.5 作为分类器, 在 13 个数据集上的平均 F-measure 和对应的 Win/Draw/Loss 分析. 结果表明: 基于 F-measure 的实验结果与基于 AUC 的实验结果相似.

若选择 NB 作为分类器: (1) 从表 10 的第 2 和 3 列中可以发现, CIG 平均比 FullSet 高 3.80%. 通过 Win/Dr aw/Loss 分析, CIG 在 13 个数据集上的效果要优于 FullSet. 这表明使用 CIG 做特征选择可以提高 NB 分类器的缺陷预测性能. (2) 从表中还可以发现, 与 CFS, FCBF 和 Consist 相比, CIG 平均可以提高 0.90%, 0.90% 和 1.50%. 同时 Win/Draw/Loss 分析也表明 CIG 在大多数情况下要优于这 3 种方法. 除此之外, 我们对 NB 分类器的预测结果进行了 Friedman 检验, 得到的 p 值是 0.0234 (小于 0.05), 这表明各种方法的预测结果具有统计性差异.

表 10 FECAR 和其他经典特征子集选取方法的 F-measure 分析

Table 10 F-measure analysis between FECAR and other classical feature selection methods

	Naive Bayes		C4.5	
	F-measure	Win/Draw/Loss	F-measure	Win/Draw/Loss
CIG	0.835	–	0.852	–
CRF	0.830	8/0/5	0.848	5/0/8
CCS	0.834	7/0/6	0.849	9/0/4
FullSet	0.797	13/0/0	0.843	9/0/4
CFS	0.826	11/0/2	0.840	10/0/3
FCBF	0.826	7/0/6	0.833	13/0/0
Consist	0.820	11/0/2	0.841	9/0/4

若选择 C4.5 作为分类器: (1) 从表 10 的第 4 和 5 列中可以发现, CIG 平均比 FullSet 高 0.90%. 通过 Win/D raw/Loss 分析, CIG 在 9 个数据集上的效果要优于 FullSet. 这表明使用 CIG 做特征选择可以提高 C4.5 分类器的缺陷预测性能. (2) 从表中还可以发现, 和 CFS, FCBF 和 Consist 相比, CIG 平均可以提高 1.20%, 1.90% 和 1.10%. 同时 Win/Draw/Loss 分析也表明 CIG 在大多数情况下要优于这 3 种方法. 对 C4.5 分类器的预测结果同样进行了 Friedman 检验, 得到的 p 值是 0.0339 (小于 0.05), 这表明各种方法的预测结果也具有统计性差异.

5 总结与展望

论文针对软件缺陷预测数据集中的无关特征和冗余特征, 提出一种新颖的特征选择方法 FECAR. 即基于特征之间的关联性, 将已有特征进行聚类分析, 随后基于特征与类标之间的相关性, 将每个簇中的特征从高到低进行排序, 并选出指定数量的特征. 在来自实际项目的数据集上对该方法的有效性进行了验证. 但该方法仍有一些后续工作值得扩展, 包括: (1) 在特征聚类分析阶段, 虽然论文启发式地选择了和类标最为相关的特征作为初始中心, 但仍需与其他中心点初始方法进行比较, 来验证论文提出的策略是否能够加快收敛速度. (2) 论文在实证研究中, 根据 Gao 等 [33] 的建议, 将最终选出的特征数设置为 $\lceil \log_2 M/2 \rceil$. 虽然最终结果表明 FECAR 可以有效提高缺陷预测的性能, 但该特征数并不一定是最优值, 因此进一步分析选出的特征数对预测性能的影响具有一定的研究意义.

参考文献

1 Wang Q, Wu S J, Li M S. Software defect prediction. J Softw, 2008, 19: 1565–1580 [王青, 伍书剑, 李明树. 软件缺陷预测技术. 软件学报, 2008, 19: 1565–1580]

2 Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng, 2012, 38: 1276–1304

3 Yu S S, Zhou S G, Guan J H. Software engineering data mining: a survey. J Front Comput Sci Tech, 2012, 6: 1–31 [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: 1–31]

4 Chen X, Gu Q, Liu W S, et al. Survey of static software defect prediction. J Softw, 2016, 1: 1–25 [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, 1: 1–25]

5 Ghotra B, McIntosh S, Hassan A E. Revisiting the impact of classification techniques on the performance of defect prediction models. In: Proceedings of the International Conference on Software Engineering, Firenze, 2015. 789–800

6 Peters F, Menzies T, Layman L. LACE2: better privacy-preserving data sharing for cross project defect prediction. In: Proceedings of the International Conference on Software Engineering, Firenze, 2015. 801–811
7 Tantithamthavorn C, McIntosh S, Hassan A E, et al. The impact of mislabelling on the performance and interpretation of defect prediction models. In: Proceedings of the International Conference on Software Engineering, Firenze, 2015. 812–823
8 Jing X Y, Wu F, Dong X W, et al. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. In: Proceedings of the International Symposium on Foundations of Software Engineering, Bergamo, 2015. 496–507
9 Nam J, Kim S. Heterogeneous defect prediction. In: Proceedings of the International Symposium on Foundations of Software Engineering, Bergamo, 2015. 508–519
10 Kim M, Nam J, Yeon J, et al. REMI: defect prediction for efficient API testing. In: Proceedings of the International Symposium on Foundations of Software Engineering, Bergamo, 2015. 990–993
11 Nam J, Kim S. CLAMI: defect prediction on unlabeled datasets. In: Proceedings of the International Conference on Automated Software Engineering, Lincoln, 2015. 452–463
12 Rahman F, Khatri S, Barr E T, et al. Comparing static bug finders and statistical prediction. In: Proceedings of the International Conference on Software Engineering, Hyderabad, 2014. 424–434
13 Shepperd M, Bowes D, Hall T. Researcher bias: the use of machine learning in software defect prediction. IEEE Trans Softw Eng, 2014, 40: 603–616
14 Radjenovic D, Hericko M, Torkar R, et al. Software fault prediction metrics: a systematic literature review. Inf Softw Tech, 2013, 55: 1397–1418
15 McCabe T J. A complexity measure. IEEE Trans Softw Eng, 1976, 2: 308–320
16 Halstead M H. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevier Science Inc., 1977
17 Chidamber S R, Kemerer C F. A metrics suite for object oriented design. IEEE Trans Softw Eng, 1994, 20: 476–493
18 Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proceedings of the International Conference on Software Engineering, St. Louis, 2005. 284–292
19 Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proceedings of the International Conference on Software Engineering, Leipzig, 2008. 181–190
20 Hassan A E. Predicting faults using the complexity of code changes. In: Proceedings of the International Conference on Software Engineering, Vancouver, 2009. 78–88
21 Pinzger M, Nagappan N, Murphy B. Can developer-module networks predict failures? In: Proceedings of the International Symposium on Foundations of Software Engineering, Atlanta, 2008. 2–12
22 Meneely A, Williams L, Snipes W, et al. Predicting failures with developer networks and social network analysis. In: Proceedings of the International Symposium on Foundations of Software Engineering, Atlanta, 2008. 13–23
23 Jiang T, Tan L, Kim S. Personalized defect prediction. In: Proceedings of International Conference on Automated Software Engineering, Silicon Valley, 2013. 279–289
24 Zimmermann T, Nagappan N. Predicting defects using network analysis on dependency graphs. In: Proceedings of the International Conference on Software Engineering, Leipzig, 2008. 531–540
25 Bird C, Nagappan N, Gall H, et al. Putting it all together: using socio-technical networks to predict failures. In: Proceedings of the International Symposium on Software Reliability Engineering, Mysuru, 2009. 109–119
26 Nagappan N, Murphy B, Basili V R. The influence of organizational structure on software quality: an empirical case study. In: Proceedings of the International Conference on Software Engineering, Leipzig, 2008. 521–530
27 Mockus A. Organizational volatility and its effects on software defects. In: Proceedings of the International Symposium on Foundations of Software Engineering, Santa Fe, 2010. 117–126
28 Bird C, Nagappan N, Devanbu P, et al. Does distributed development affect software quality? An empirical case study of Windows Vista. In: Proceedings of International Conference on Software Engineering, Vancouver, 2009. 518–528
29 Shepperd M, Song Q B, Sun Z B, et al. Data quality: some comments on the NASA software defect datasets. IEEE Trans Softw Eng, 2013, 39: 1208–1215
30 Bird C, Bachmann A, Aune E, et al. Fair and balanced? Bias in bug-fix datasets. In: Proceedings of the the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software

Engineering, Amsterdam, 2009. 121–130
31 Bachmann A, Bird C, Rahman F, et al. The missing links: bugs and bug-fix commits. In: Proceedings of International Symposium on Foundations of Software Engineering, Santa Fe, 2010. 97–106
32 Nguyen T H, Adams B, Hassan A E. A case study of bias in bug-fix datasets. In: Proceedings of the Working Conference on Reverse Engineering, Beverly, 2010. 259–268
33 Gao K H, Khoshgoftaar T M, Wang H J, et al. Choosing software metrics for defect prediction: an investigation on feature selection techniques. Softw Pract Exper, 2011, 41: 579–606
34 Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng, 2007, 32: 1–12
35 Song Q B, Jia Z H, Shepperd M, et al. A general software defect-proneness prediction framework. IEEE Trans Softw Eng, 2011, 37: 356–370
36 Shivaji S, Whitehead Jr E J, Akella R, et al. Reducing features to improve code change-based bug prediction. IEEE Trans Softw Eng, 2013, 39: 552–569
37 Wang H J, Khoshgoftaar T M, Napolitano A. A comparative study of ensemble feature selection techniques for software defect prediction. In: Proceedings of the International Conference on Machine Learning and Applications, Washington, 2010. 135–140
38 Khoshgoftaar T M, Gao K H, Seliya N. Attribute selection and imbalanced data: problems in software defect prediction. In: Proceedings of the International Conference on Tools With Artificial Intelligence, Arras, 2010. 137–144
39 Wang S, Yao X. Using class imbalance learning for software defect prediction. IEEE Trans Reliab, 2013, 62: 434–443
40 Jing X Y, Ying S, Zhang Z W, et al. Dictionary learning based software defect prediction. In: Proceedings of the International Conference on Software Engineering, Hyderabad, 2014. 414–423
41 Hall M A. Correlation-based Feature selection for discrete and numeric class machine learning. In: Proceedings of the International Conference on Machine Learning, Stanford, 2000. 359–366
42 Yu L, Liu H. Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the International Conference on Machine Learning, Washington, 2003. 856–863
43 Kim S, Whitehead Jr E J, Zhang Y. Classifying software changes: clean or buggy? IEEE Trans Softw Eng, 2008, 34: 181–196
44 Kira K, Rendell L A. A practical approach to feature selection. In: Proceedings of the International Workshop on Machine Learning, Aberdeen, 1992. 249–256
45 Fayyad U M, Irani K B. Multi-Interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the International Joint Conference on Artificial Intelligence, Chambery, 1993. 1022–1029
46 Lessmann S, Baesens B, Mues C, et al. Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng, 2008, 34: 485–496
47 Dash M, Liu H. Consistency-based search in feature selection. Artif Intell, 2003, 151: 155–176
48 Kononenko I. Estimating attributes: analysis and extensions of RELIEF. In: Proceedings of the European Conference on Machine Learning, Catania, 1994. 171–182
49 Zimmermann T, Premraj R, Zeller A. Predicting defects for eclipse. In: Proceedings of the International Workshop on Predictor Models in Software Engineering, Washington, 2007. 1–7
50 Witten I H, Frank E, Hall M A. Data Mining: Practical Machine Learning Tools and Techniques. 3rd ed. San Francisco: Morgan Kaufmann Publishers Inc., 2011

A cluster-analysis-based feature-selection method for software defect prediction

Wangshu LIU[1,2], Xiang CHEN[1,3], Qing GU[1,2*], Shulong LIU[1,2] & Daoxu CHEN[1,2]

1 *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China*;
2 *Department of Computer Science and Technology, Nanjing University, Nanjing* 210023*, China*;
3 *School of Computer Science and Technology, Nantong University, Nantong* 226019*, China*
*E-mail: guq@nju.edu.cn

Abstract By mining historical software repositories, software defect prediction can construct defect-prediction models to predict potentially faulty modules in projects under testing. However, redundant and irrelevant features in the gathered datasets may influence the effectiveness of existing methods. A novel cluster-analysis-based feature-selection method (FECAR) is proposed. In particular, the original features are first clustered, based on a specific feature correlation (i.e., FFC) measure. Then, for each cluster, features are ranked based on a specific feature and class relevance (i.e., FCR) measure and a given number of features are chosen. In empirical studies, we chose symmetric uncertainty as the FFC measure, and information gain, chi-square, or ReliefF as the FCR measures. Based on some real-world projects, such as Eclipse and NASA, we focus on the prediction performance after using FECAR, and analyze the redundancy rate and selection proportion of the selected feature subset. The final results show the effectiveness of FECAR.

Keywords software quality assurance, defect prediction, data mining, feature selection, cluster analysis

Wangshu LIU was born in 1987. He received his M.S. degree in Computer Science from Nanjing University of Science and Technology, Nanjing, China, in 2013. Currently, he is a Ph.D. candidate in the Department of Computer Science and Technology, Nanjing University, Nanjing. His research interests include software defect prediction, software data preprocessing, and data mining.

Xiang CHEN was born in 1980. He received his Ph.D. degree in Computer Science from Nanjing University, Nanjing, China, in 2011. Currently, he is an associate professor in the Department of Computer Science and Technology at Nantong University. His research interests include software defect prediction, software fault localization, regression testing, and combinatorial testing.

Qing GU was born in 1972. He received his Ph.D. degree in Computer Science from Nanjing University. He is a professor of the State Key Laboratory of Novel Software Technology, and the Department of Computer Science and Technology, Nanjing University. His research interests include software testing, quality and process improvement, software maintenance and evolution, and complex networks. He is a senior member of the CCF.

Shulong LIU was born in 1990. He received his M.S. degree in Computer Science from Nanjing University, Nanjing, China, in 2015. Currently, he is a software engineer in Qihoo 360 Technology Company, Beijing. His research interests include software defect prediction, software data preprocessing, and data mining.