基于半监督聚类方法的测试用例选择技术

程雪梅 杨秋辉 翟宇鹏 陈 伟
(四川大学计算机学院(软件学院) 成都 610065)

摘 要 回归测试的目的是保证软件修改后没有引入新的错误。但是随着软件的演化,回归测试用例集不断增大,为了控制成本,回归测试用例选择技术应运而生。近年来,聚类分析技术被运用到回归测试用例选择问题中。将半监督学习引入到聚类技术中,提出了判别型半监督K-means聚类方法(Discriminative Semi-supervised K-means clustering Method,DSKM)。该方法从回归测试的历史执行记录中挖掘出隐藏的成对约束信息,同时利用大量的无标签样本和少量的有标签样本进行学习,优化聚类的结果,并进一步优化测试用例选择的结果。实验表明,相对于Constrained-Kmeans方法和SSKM方法,DSKM方法能够更好地提高约简率并保持覆盖率。

关键词 回归测试,测试用例选择,K-means算法,成对约束,线性判别分析,半监督聚类
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.01.044

Test Case Selection Technique Based on Semi-supervised Clustering Method

CHENG Xue-mei YANG Qiu-hui ZHAI Yu-peng CHEN Wei
(College of Computer Science(Software Engineering College),Sichuan University,Chengdu 610065,China)

Abstract The purpose of regression testing is to ensure that no new faults are introduced into software after modifications. The case of regression test is increasing with the evolution of software,the software of so test selection techniques are used to control costs. In recent years,cluster analysis techniques are applied to test selection problem. We proposed a novel method called discriminative semi-supervised K-means clustering method (DSKM),which introduces semi-supervised learning clustering technology. Through DSKM,hidden pairwise constraints information is mined from the test execution history. By taking advantage of a large number of unlabeled samples and a small amount of labeled samples,DSKM can optimize the results of the cluster,and further optimize test case selection results. Experiment shows that compared with Constrained-Kmeans algorithm and SSKM method,DSKM is more effective.

Keywords Regression testing,Test case selection,K-means algorithm,Pairwise constraints,Linear discriminant analysis,Semi-supervised clustering

1 引言

在软件开发过程中,软件系统及其环境在不断地变化。增强功能、纠正错误、新增或者删除功能都需要修改代码并触发软件的演化。为了确保演化后的软件能够正确运行且不引入新的错误,必须对软件进行回归测试[1]。统计数据表明,回归测试的开销一般占整个软件测试预算的80%以上,占整个软件维护预算的50%以上,因此有效的回归测试非常重要[2]。然而,随着软件的不断演化,测试用例集不断增大,由于资源有限,几乎不可能运行所有的测试用例。为了提高回归测试的运行效率,许多策略被提出,如失效测试用例的识别和修复技术、测试用例选择技术(Regression Test case Selection,RTS)、测试用例集约简技术、测试用例优先级排序技术和测试用例集扩充技术。其中最主要的一种技术是回归测试用例选择技术[3]。

近年来,数据挖掘中的聚类技术被用于解决回归测试用例选择问题。其基本思想是:根据测试用例的历史执行剖面进行聚类,将具有相似函数覆盖、能够发现相同故障的测试用例聚为一个簇。同一簇中的测试用例具有相似的行为,而不同簇中的测试用例的行为差异较大。若一个测试用例能检测某一故障,则属于同一簇的其他测试用例通常也能检测这一故障[1]。在回归测试时,只需从每一个簇中选取一定比例的测试用例组成新的测试用例集,新测试用例集的精度和覆盖率在很大程度上依赖于聚类结果,而聚类结果依赖于聚类算法的选取。传统的K-means聚类算法是无监督的,对选取的K值和初始的簇中心都非常敏感,可能会产生局部最优的聚类结果,从而导致选择出的测试用例无法满足较高的准确率和覆盖率。为了优化聚类结果,提出一种判别型半监督K-means聚类方法(Discriminative Semi-supervised K-means clustering Method,DSKM)。将半监督聚类运用到回归测试用例选择问题中,结果表明,其在保证覆盖率的同时减少了测试用例,降低了回归测试的成本,提高了回归测试的效率。

到稿日期:2016-11-21 返修日期:2017-04-12 本文受四川省应用基础研究项目(2014JY0112)资助。
程雪梅(1991-),女,硕士,主要研究方向为软件工程、软件测试、数据挖掘,E-mail:chengxuemei1991@163.com(通信作者);杨秋辉(1970-),女,博士,副教授,主要研究方向为软件测试、经验软件工程,E-mail:yangqiuhui@scu.edu.cn。

文章第 2 节总结了目前该领域的相关研究；第 3 节介绍了提出的方法；第 4 节通过设计和执行实验来验证所提方法；最后总结全文，提出未来工作。

2 相关工作

关于 RTS 问题的研究，众多研究人员提出了很多解决方案。从源代码和模型角度可以将 RTS 技术分为线性规划法、数据流分析法、防火墙法、程序切片法和图遍历法。Fischer 等人首次针对 Fortran 编程语言对 RTS 问题进行研究，提出了一种使用可达矩阵和测试用例依赖矩阵的线性规划法。随后，Rothermel 和 Harrolad 总结并提出了一种统一评估框架[3-4]，提出了一种基于控制流图的 RTS 技术并开发了 Deja-Vu 工具[5]。Beydeda 和 Gruhn 基于 Rothermel 等人的研究工作，通过将黑盒测试中的数据流信息添加到类控制流图来对面向对象的程序进行测试[6]。Leung 等人实现了防火墙方法，在受代码修改影响且需要重新测试的模块周围构造防火墙来帮助解决 RTS 问题[7]。Gupta 等人基于数据流分析法，结合程序切片技术来提高执行效率[8]。Larprattanakul 等人则使用对象依赖图来减少测试用例[9]。Gligoric 等人提出了一种新的轻量级的 RTS 技术（即 Ekstazi），它可以很好地与测试框架集成，他们实现了 Ekstazi 与 JUnit 的集成，并通过实验证明了尽管 Ekstazi 选择出了更多的测试用例，但是它具有更低的端对端时间[10]。广义上又可以将 RTS 技术分为静态和动态两类，Legunsen 和 Hariri 首先针对静态 RTS 技术的性能优势和安全性进行了扩展研究，实现了类级别和方法级别的两类静态 RTS 技术，并将这两类技术和 Ekstazi 都进行了比较分析[11]。Shi 和 Yung 等人则首先对测试用例集约简和回归测试用例选择技术进行了比较研究，并且对两种方法的结合进行了评估[12]。

近年来，随着计算机软硬件性能的不断提高，不少学者将聚类技术用于 RTS 问题。Dickinson 等人[13]提出基于执行剖面聚类的测试用例选择，用于挖掘测试用例之间隐藏的联系。Zhang 等人[1]在 Rothermel 的基础上，通过对遍历测试修改的执行剖面进行聚类，增强了安全选择技术。Shin 等人[14]提出了基于聚类的测试用例优先级技术，用来减少成对比较的数量。SONGYU 等人[15]率先将半监督聚类运用到回归测试用例选择的问题上，提出了 SSKM 方法，利用测试用例的函数覆盖信息对测试用例进行半监督聚类，从而减少测试用例。

使用聚类技术解决 RTS 问题，可以在保证测试用例的错误检测能力和覆盖率的前提下，提高测试用例的约简率[1,16]。但目前已有的方法大都是非监督的，它们利用无标签数据进行训练和组织数据，聚类的结果依赖于目标函数的设定和参数的输入，而这些参数往往需要人工设置。由于软件系统的复杂性和多样性，很难进行参数设置，因此获得的聚类效果也难以得到保证。尹学松等人提出了基于成对约束的判别型半监督聚类分析方法[17]。文章基于 SONGYU 和尹学松等人的方法，结合半监督降维（Semi-Supervised Dimensionality Reduction，SSDR）和线性判别分析（Linear Discriminant Analysis，LDA）来解决回归测试用例选择问题，优化回归测试用例选择的结果。

3 基于 DSKM 的测试用例选择方法的整体流程

基于 DSKM 的测试用例选择方法主要包括数据提取、约束推导、数据降维、数据聚类、用例取样 5 个部分，如图 1 所示。

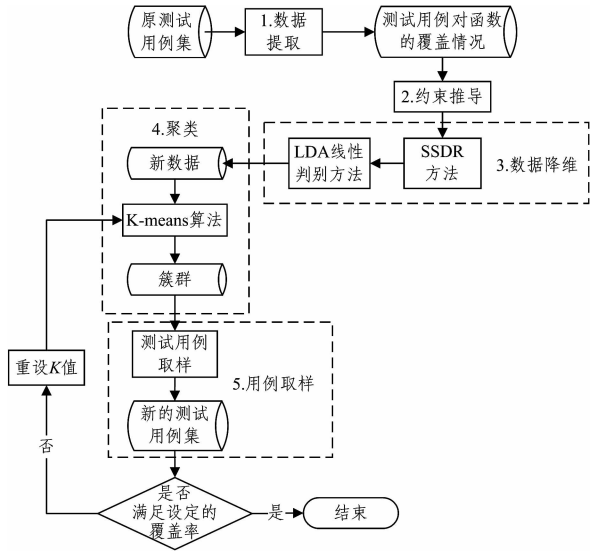


图 1 DSKM 方法的整体流程图

Fig. 1 The overall flowchart of DSKM

DSKM 方法首先通过分析测试用例对函数的覆盖情况来得到原始数据集，即测试用例与其覆盖函数的二进制向量组成的二维表；然后将从测试用例执行历史记录中挖掘出的成对约束作为输入，使用 SSDR 算法[18]得到投影矩阵，在投影空间内对原始数据聚类得到聚类标号；接着再将聚类标号作为输入，通过线性判别分析选择子空间，在子空间上对数据集进行投影，得到新的数据集，即降维过后的二维表；最后使用 K-means 算法对新的数据集进行聚类，将测试用例聚为 K 簇，使用自适应的取样策略从每一个簇中选出一定比例的测试用例组成新的测试用例集。该方法结合 SSDR 方法和 LDA 方法对聚类结果进行优化。下面具体描述各步骤。

3.1 数据提取

本阶段进行原始数据集的收集。在测试用例执行过程中，其执行结果会被记录。使用代码覆盖率分析工具对每个测试用例的代码覆盖情况进行分析。使用一个二进制向量表示测试用例的函数覆盖，每一位都记录对应函数在测试用例执行过程中是否被覆盖，如果某个函数被覆盖，则该位被置为 1，否则置为 0。最终得到测试用例与其覆盖的函数组成的二维表，即原始数据集。得到的原始数据集表示为 $X=\{x_1,x_2,x_3,\cdots,x_n\}$，形式如表 1 所列，其中 x_i 表示测试用例 i 对应的函数覆盖的二进制向量，每个 x_i 代表一个数据对象，f_j 表示第 j 个函数，n 是测试用例数量，m 是函数数量。

表 1 原始数据集 X

Table 1 Original data set X

	f_1	f_2	f_3	…	f_m
x_1	0	0	0	…	0
x_2	1	1	1	…	1
x_3	1	1	1	…	1
…	…	…	…	…	…
x_n	1	0	0	…	0

3.2 约束推导

得到原始数据集 X 后，通过使用测试用例执行历史记录来推导成对约束信息。在半监督聚类中，通过数据标签或者数据间的约束形式使用有限的监督。文中使用了两种类型的成对约束(pairwise constraints)来表示测试用例之间的约束关系[18]。

(1)Must-link：两个测试用例必须在同一簇中。

(2)Cannot-link：两个测试用例必须在不同的簇中。

经过约束推导得到约束集 M 和 C。M 代表 Must-link 约束集，C 代表 Cannot-link 约束集。例如在表 1 中，若只考虑测试用例 x_1，x_2 和 x_3 以及函数 f_1，f_2 和 f_3，由表中数据可知 x_2 和 x_3 都覆盖了函数 f_1，f_2 和 f_3，因此 x_2 和 x_3 应该属于约束集 M。反之，x_1 和 x_2 覆盖的函数都不相同，因此 x_1 和 x_2 应该属于约束集 C。

3.3 数据降维

在对测试用例进行聚类之前，结合 SSDR 方法和 LDA 方法对提取的数据集进行降维处理。首先，利用 SSDR 方法求出投影矩阵 W，在投影空间内对数据集聚类得到聚类标号。然后利用 LDA 方法选择子空间。在子空间上对数据集进行投影，得到新的数据集，即降维后的二维表。

(1)使用 SSDR 算法求解投影矩阵

将原始数据集 X、约束集 M 和 C 作为输入，利用 SSDR 算法[18]生成变换矩阵 $W=\{w_1,w_2,w_3,\cdots,w_d\}$，$d$ 代表 W 矩阵的行数，然后使用 W 矩阵将原始数据集 X 转换为低维度数据集 $Y=\{y_1,y_2,y_3,\cdots,y_n\}$，其中 $y_i=W^{\mathrm{T}}x_i$。SSDR 算法通过式(1)求解 W 矩阵，所求 W 矩阵为使目标函数 $J(w)$ 最大化的 W 的值，其中 $W^{\mathrm{T}}W=1$。

$$J(w)=\frac{1}{2n^2}\sum_{i,j}(w^{\mathrm{T}}x_i-w^{\mathrm{T}}x_j)^2+\frac{\alpha}{2n_C}\sum_{(x_i,x_j)\in C}(w^{\mathrm{T}}x_i-w^{\mathrm{T}}x_j)^2+\frac{\beta}{2n_M}\sum_{(x_i,x_j)\in M}(w^{\mathrm{T}}x_i-w^{\mathrm{T}}x_j)^2 \tag{1}$$

其中，n 表示数据集 X 中所有数据对象的数量，n_C 表示属于 Cannot-link 约束集的数据对象的数量，n_M 表示属于 Must-link 约束集中的数据对象的数量。第一项表示所有数据对象的平均平方距离，第二项和第三项表示包含在成对约束集中的所有数据对象的平均平方距离。通过式(1)，将 Cannot-link 约束集中数据对象间的距离增大，同时将 Must-link 约束集中的数据对象间的距离减小，从而保证属于 Must-link 约束集中的测试用例被聚到同一簇中，属于 Cannot-link 约束集中的测试用例被聚到不同簇中。上式使用了两个参数 α 和 β 来平衡约束的权重，α 与 β 的比值会影响聚类的结果。

采用 SSDR 方法求出投影矩阵 W 后，利用 W 矩阵将原始数据集 X 投影到低维空间，得到数据集 Y，并使用传统的 K-means 聚类算法对数据集进行聚类，得到每个测试用例的聚类标号组成的向量 *Labels*，以便于下一步使用 LDA 选择子空间。

(2)使用 LDA 方法选择子空间

LDA 的目的是最大化类间距离，最小化类内距离。将使用 SSDR 方法得到的聚类标号向量 *Labels* 和数据集 Y 作为 LDA 方法的输入，进行有监督的维数约简处理，得到降维后的新数据集 X'，用于下一步 K-means 聚类算法的输入。

LDA 是监督维数约简方法，它寻找一个最优的投影方向，使得在投影空间中的不同类数据对象之间的距离远，而同类数据对象之间的距离近。LDA 的目标函数如下：

$$W_{opt}=\arg\max\frac{W^{\mathrm{T}}S_BW}{W^{\mathrm{T}}S_WW} \tag{2}$$

式(2)中类间散布矩阵 S_B 和类内散布矩阵 S_W 可以分别表示如下：

$$S_B=\sum_{i=1}^{L}n_i(m_i-m)(m_i-m)^{\mathrm{T}} \tag{3}$$

$$S_w=\sum_{j=1}^{L}\sum_{i=1}^{n_j}(x_i-m_j)(x_i-m_j)^{\mathrm{T}} \tag{4}$$

式(3)和式(4)中，L 是类数，m 是全部样本的均值，m_i 是第 i 类样本均值，n_i 是第 i 类样本数，m_j 是第 j 类中全部样本的均值，n_j 是第 j 类样本数，x_i 是第 j 类中的一个样本。

3.4 聚类

对原始数据集 X 进行降维处理得到新数据集 X' 后，使用 K-means 算法对 X' 进行聚类，通过聚类将具有相似函数覆盖、能够发现相同错误的测试用例聚到同一簇中，得到测试用例的 K 个聚类。

3.5 用例取样

经过 K-means 算法聚类以后，原测试用例集被聚到 K 个簇中。最后从每一个簇中选出测试用例，建立新的回归测试用例集。取样策略有很多种，常见的有自适应取样策略[1,13]、动态取样策略[19]、随机取样策略[20]等。本文选择自适应取样策略，首先按照一定比例(如 10%)从每一个簇中随机选取少量测试用例(至少选出一个测试用例)，如果某个测试用例覆盖了修改的函数，则直接将其放到新的测试用例集中。如果选出的测试用例为可以发现故障的测试用例，则与其在同一簇中的全部测试用例都将被选出，用于组成新的测试用例集。

4 实验

为了验证 DSKM 方法的可行性和有效性，文中进行了大量的实验探究，并将其对 DSKM 方法、Constrained-Kmeans 方法、SSKM 方法进行了对比。

4.1 实验对象

论文选用 Xml-security 项目和 Junit 项目作为实验对象。Xml-security 项目和 Junit 项目都是 GitHub 上的 Java 开源项目。表 2 列出了实验对象对应的版本数量、类数量、函数数量、测试用例的数量和原测试用例集在基础版本上的代码覆盖率。对于实验对象，下载的最新版本为正确的基础版本。依次将实验对象的历史版本中的 bug 添加到基础版本中，每次添加一个 bug，形成多个修改版。将测试用例在各个修改版本上运行并记录运行结果。执行此操作的原因在于不能直接从下载的实验对象中得到测试用例的执行结果记录，在实际应用中则可以直接使用测试用例的执行结果记录。

表 2　实验对象

Table 2　Experimental subjects

实验对象	版本数量	类数量	函数数量	测试用例数量	代码覆盖率/%
Xml-security	16	333	2430	117	56.7
Junit	20	235	1450	159	85.1

4.2 实验步骤

实验主要分为 5 个基本步骤。

4.2.1 原始数据提取

利用 eclipse 工具在实验对象的各修改版本上执行测试用例集，并使用 Eclemma 工具收集各测试用例的覆盖信息。需要记录 3 类信息：

(1)测试用例集在实验对象的各修改版本上的总的函数覆盖率。

(2)测试用例集中每个测试用例的函数覆盖信息，形成函数覆盖的二进制向量。

(3)测试用例集中每个测试用例在实验对象的每个修改版本上的执行结果，如果与基础版本的执行结果相同，则表示通过；如果不同，则表示失败。记录失败测试用例的序号和对应的版本号，将其用于在下一步中提取成对约束。

4.2.2 约束推导

一个执行失败的测试用例至少可以检测到一个 bug，某些执行失败的测试用例在多个修改版本中都能检测到 bug。建立测试用例和版本之间的映射关系。$V(x)$代表测试用例 x 能够检测到 bug 的版本集合，用重合度 CD 表示两个测试用例 x_1 和 x_2 发现相同 bug 的比率，定义 CD 如下[15]：

$$CD=\frac{|V(x_1)\cap V(x_2)|}{\max\{|V(x_1)|,|V(x_2)|\}}\times 100\% \tag{5}$$

其中，$V(x_1)$和 $V(x_2)$不能同时为 0，即不计算两个已通过的测试用例的重合度。根据 CD 的值，建立 Must-link 和 Cannot-link 约束集。

(1)Must-link：如果 $CD(x_1,x_2)\geqslant TM$，则 x_1 和 x_2 属于 Must-link 约束集。TM 为 Must-link 约束的阈值，为了探究阈值定义严格与否对实验结果的影响，在实验中使用了两个值：$TM=100\%$和 $TM=50\%$，100%是极端情况，50%是中间值情况。

(2)Cannot-link：如果 $CD(x_1,x_2)=0$，则 x_1 和 x_2 属于 Cannot-link 约束集，即 x_1 和 x_2 没有检测到相同的 bug。

不难看出，不同的约束定义会推导出不同的成对约束，也会影响聚类的结果，进而影响测试用例选择的结果。

4.2.3 数据降维

推导出成对约束后，使用 SSDR 方法得到权值矩阵 W，然后利用权值矩阵 W 将原始数据集 X 映射为 Y，$y_i=W^{\mathrm{T}}x_i$。在 X 中每个元素的值为 1 或者 0，代表函数是否被覆盖。经过数据转换之后，每个元素的值都是一个浮点数，向量的大小比以前更小。使用 MATLAB 工具编程实现 SSDR 方法，得到经过 SSDR 方法处理过后的数据集 Y。

然后，使用 K-means 算法对经过 SSDR 算法处理后的数据集 Y 进行聚类，得到每个测试用例对应的类标号，用 *Labels* 向量表示。将类标号向量 *Labels* 和数据集 Y 作为 LDA 方法的输入，完成对数据的有监督的维数约简。使用 MATLAB 工具实现 LDA 方法，得到最终用于聚类的新数据集 X'。

4.2.4 聚类

使用 SSDR 方法和 LDA 方法对数据集进行降维之后，再使用 K-means 算法对上面形成的新数据集 X'进行聚类，得到聚类的结果。该步使用 Rapidminer 工具的 K-means 算法。

4.2.5 用例取样

对测试用例聚类结束以后，使用自适应的取样策略选取测试用例。从每个簇中选取一定比例的测试用例组成新的测试用例集。对选出的新的测试用例集计算覆盖率，如果覆盖率接近或者等于原始测试用例集在历史版本上的覆盖率，则停止，返回新的测试用例集；否则，返回聚类步骤，调整 K 值。

4.3 评估度量

为了评估所提方法的有效性，采用的评测指标包括召回率、准确率、约简率以及代码覆盖率。设 T 为原测试用例集，T_F 为原测试用例集中能够发现故障的用例集合，T'为回归测试选出的测试用例集，T_F'为选出的测试用例集中能够发现故障的用例集合。

(1)召回率

Recall 是测试用例选择的完整性度量，定义为 $Recall=|T_F'|/|T_F|$。*Recall* 值越大，代表了更强的错误检测能力。

(2)准确率

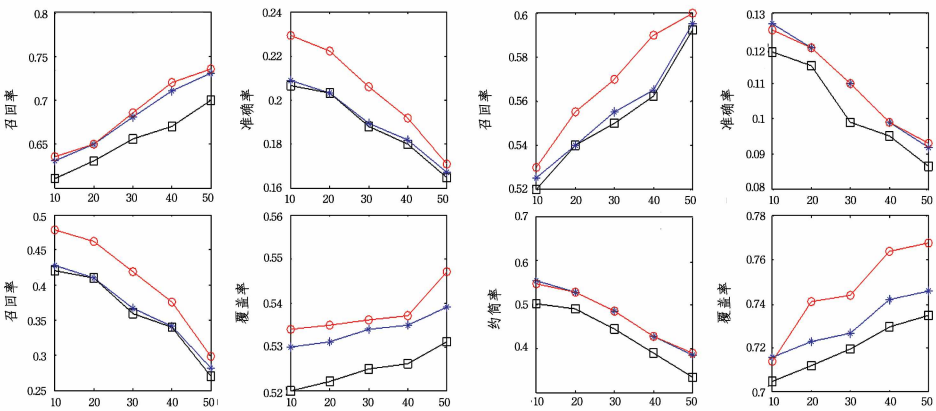
Precision 是测试用例选择的准确性程度，定义为 $Precision=|T_F'|/|T'|$。*Precision* 值越大，代表更少的测试资源被浪费。

(3)约简率

Reduction 用于衡量测试用例集约简的程度，定义为 $Reduction=|T-T'|/|T|$。*Reduction* 值越大，代表约简程度越大。

4.4 结果及分析

为了验证 DSKM 方法的有效性，将其与 Constrained-Kmeans 方法和 SSKM 方法进行对比，对比结果如图 2 和图 3 所示。图 2 中 P 为最后取样策略中的比率值，如果某簇中仅含一个测试用例，则直接选出该测试用例。



注：—□—表示 Constrained-Kmeans 方法；—✳—表示 SSKM 方法；—○—表示 DSKM 方法

图 2 不同 P 值下 3 种方法的选择结果

Fig. 2 The selected results of three methods under different P

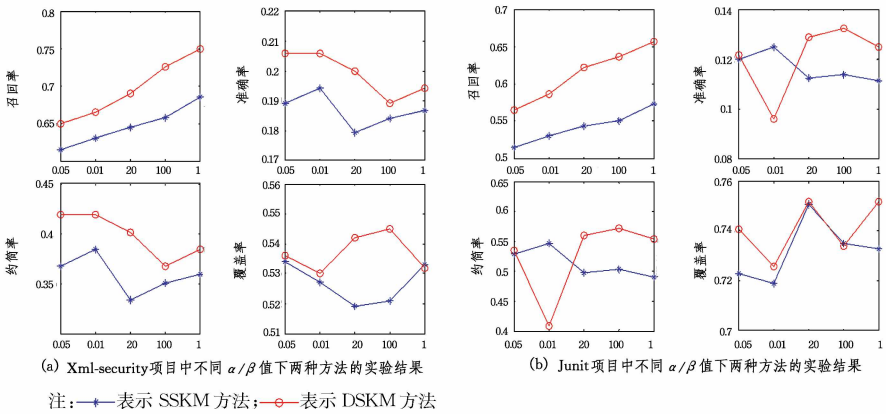


图 3　不同的 α 和 β 的比值下两种方法的选择结果

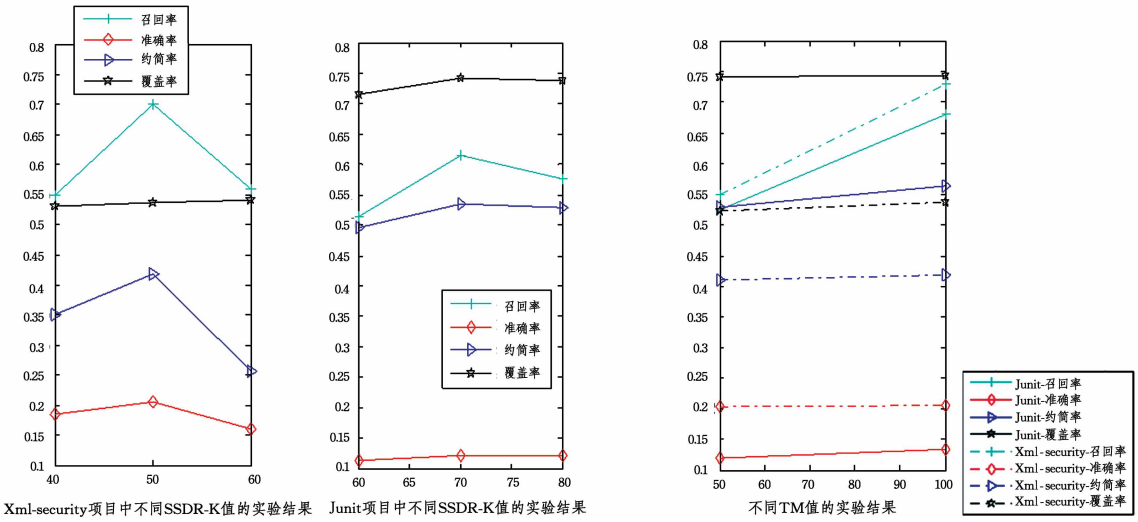
Fig. 3　The selected results of two methods under different rate of α and β

图 2 给出了选出的测试用例的比例值 P 对选择结果的召回率、准确率、约简率和覆盖率的影响。

召回率和覆盖率的值随着 P 值的增大而增大，准确率和约简率的值随着 P 值的增大而减小。从结果中可以看出，相对于 Constrained-Kmeans 方法[21]，DSKM 方法在保持甚至提高覆盖率的前提下，明显地提高了召回率、准确率和约简率。相对于 SSKM 方法[12]，DSKM 方法在同等召回率、约简率和准确率的前提下，能够更好地保持覆盖率。

图 3 给出了 α 和 β 的比值对选择结果的召回率、准确率、约简率和覆盖率的影响。从图 3 中可以看出，DSKM 方法在保持召回率和覆盖率方面都明显优于 SSKM 方法。针对准确率和约简率的比较，对于 Xml-security 项目，DSKM 方法明显优于 SSKM 方法。对于项目 Junit，虽然当 α 和 β 的比值较小时 SSKM 方法优于 DSKM，但是从整体上看，特别是当 α 和 β 的比值较大时，DSKM 方法在准确率和约简率上都优于 SSKM。

除了将 DSKM 方法与 Constrained-Kmeans 方法和 SSKM 方法进行对比以外，还对 *SSDR-K* 和 *TM* 的值对选择结果的影响进行了分析。结果如图 4 所示，图中 *TM* 为 Must-link 约束的阈值，定义为两个值：*TM*＝100％和 *TM*＝50％。*SSDR-K* 为使用 SSDR 处理以后对数据进行聚类的类数。



(a) *SSDR-K* 值对实验结果的影响　　(b) *TM* 值对实验结果的影响

图 4　*SSDR-K* 和 *TM* 的值对选择结果的影响

Fig. 4　The influence of *SSDR-K* and *TM* on the selected results

图 4 给出了 DSKM 方法，*SSDR-K* 和 *TM* 的值对召回率、准确率、约简率、覆盖率的值的影响。整体上，在一定阈值内，若 *SSDR-K* 的值越大，则召回率、准确率、约简率和覆盖率越高，但是超过阈值后会随着 *SSDR-K* 值的增大而下降。*TM* 的值越大，即对于 Must-link 约束集的定义越严格，在保证覆盖率的前提下，召回率、准确率、约简率更高。

从实验结果的分析中可以得出以下结论：1）通过对图 2 和图 3 的分析，在同等条件下，相对于 Constrained-Kmeans 方法和 SSKM 方法，DSKM 方法在保持覆盖率的前提下，能够更好地提高召回率、准确率和约简率。2）通过对图 4 的分析，在一定阈值内，当 *SSDR-K* 值越大时，召回率、准确率、约简率和覆盖率更高，但是超过阈值以后会随着 *SSDR-K* 值的增大而下降。3）约束的定义会对聚类的结果产生影响，从而影响测试用例选择的结果。通常 *TM* 的值越大，即对于 Must-link 约束集的定义越严格，在保证覆盖率的前提下，召回率、准确率、约简率更高。

结束语 文章将判别型半监督 K-means 聚类方法(DSKM)应用到回归测试用例选择中。DSKM 方法首先利用回归测试历史执行信息找出隐藏的成对约束信息。然后利用 SSDR 方法和 LDA 方法对原始数据进行降维,优化后的数据在距离度量上更加适合聚类分析,从而提高了 K-means 算法的精度。通过实验说明了 DSKM 方法在大多数情况下可以优化回归测试用例选择的结果。相对于 Constrained-Kmeans 方法和 SSKM 方法,DSKM 方法在保持较高的召回率和代码覆盖率的前提下,明显地提高了准确率和约简率。由于基于半监督聚类技术的回归测试用例选择问题较为新颖,还有很多后续的工作可以进行扩展,例如如何构造高质量的约束,降低"噪音"对聚类结果的影响;如何进一步提高测试用例选择的准确率和约简率。这些都是下一步的工作。

参 考 文 献

[1] CHEN Z, ZHENYU C, ZHIHONG Z, et al. An Improved Regression Test Selection Technique by Clustering Execution Profiles[C]// Tenth International Conference on Quality Software. New York: IEEE, 2010: 171-179.

[2] LEUNG H K N, WHITE L. Insights into regression testing [C]// Conference on Software Maintenance. New York: IEEE, 2002: 60-69.

[3] ROTHERMEL G, HARROLD M J. Analyzing regression test selection techniques[J]. IEEE Transactions on Software Engineering, 1996, 22(8): 529-551.

[4] ROTHERMEL G, HARROLD M J. A framework for evaluating regression test selection techniques [C] // 16th International Conference on Software Engineering. New York: IEEE, 2002: 201-210.

[5] ROTHERMEL G, HARROLD M J. A safe, efficient regression test selection technique[J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(2): 173-210.

[6] BEYDEDA S, GRUHN V. Integrating white-and black-box techniques for class-level regression testing[C]// 25th Annual International Computer Software and Applications Conference. New York: IEEE, 2002: 357-362.

[7] WHITE L J, LEUNG H K N. A firewall concept for both control-flow and data-flow in regression integration testing[C]// Conference on Software Maintenance. New York: IEEE, 2002: 262-271.

[8] GUPTA R, HARROLD M J, SOFFA M L. An approach to regression testing using slicing [C] // Conference on Software Maintenance. New York: IEEE, 2002: 299-308.

[9] LARPRATTANAKUL A, SUWANNASART T. An Approach for Regression Test Case Selection Using Object Dependency Graph[C] // 2013 5th International Conference on Intelligent Networking and Collaborative Systems. New York: IEEE, 2013: 617-621.

[10] GLIGORIC M, ELOUSSI L, MARINOV D. Practical regression test selection with dynamic file dependencies[C]// Proceedings of the 2015 International Symposium on Software Testing and Analysis. New York: ACM, 2015: 211-222.

[11] LEGUNSEN O, HARIRI F, SHI A, et al. An extensive study of static regression test selection in modern software evolution[C]// Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York: ACM, 2016: 583-594.

[12] SHI A, YUNG T, GYORI A, et al. Comparing and combining test-suite reduction and regression test selection[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2015: 237-247.

[13] DICKINSON W, LEON D, FODGURSKI A. Finding failures by cluster analysis of execution profiles [M]. Washington, DC, USA: IEEE Computer Society, 2001: 339-348.

[14] YOO S, HARMAN M, TONELLA P, et al. Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge [C] // Eighteenth international symposium on Software testing and analysis. New York: ACM, 2009: 201-212.

[15] SONGYU C, ZHENYU C, ZHIHONG Z, et al. Using semi-supervised clustering to improve regression test selection techniques [C] // 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. New York: IEEE, 2011: 1-10.

[16] PARSA S, KHALILIAN A, FAZLALIZADEH Y. A new algorithm to Test Suite Reduction based on cluster analysis[C]// 2nd IEEE International Conference on Computer Science and Information Technology. New York: IEEE 2009: 189-193.

[17] YIN X S, HU S L, CHEN S C. Discriminative semi-supervised clustering analysis with pairwise constraints [J]. Journal of Software, 2008, 19(11): 2791-2802. (in Chinese)
尹学松,胡思良,陈松灿. 基于成对约束的判别型半监督聚类分析[J]. 软件学报, 2008, 19(11): 2791-2802.

[18] ZHANG D, ZHOU Z H, CHEN S. Semi- Supervised Dimensionality Reduction[C] // Proceedings of the 2007 SIAM International Conference on Data Mining. Philadelphia: SIAM, 2007: 629-634.

[19] SHALI Y, ZHENYU C, ZHIHONG Z, et al. A Dynamic Test Cluster Sampling Strategy by Leveraging Execution Spectra Information[C] // 2010 Third International Conference on Software Testing, Verification and Validation. New York: IEEE, 2010: 147-154.

[20] MUTHYALA K, NAIDU R. A novel approach to test suite reduction using data mining[J]. Indian Journal of Computer Science and Engineering, 2011, 2(3): 500-505.

[21] WAGSTAFF K, CARDIE C, ROGERS S, et al. Constrained K-means Clustering with Background Knowledge[C]// Eighteenth International Conference on Machine Learning. San Francisco: ACM, 2001: 577-584.