

doi: 10.11830/ISSN.1000-5013.201606024



一种模糊 K -means 算法在测试用例集约简中的应用

余国清^{1,2}, 周兰蓉², 罗可¹

(1. 长沙理工大学 计算机与通信工程学院, 湖南 长沙 410114;
2. 湖南信息职业技术学院 计算机工程学院, 湖南 长沙 410200)

摘要: 为了提高软件测试用例集约简的成效, 提出一种基于模糊 K -means 的软件测试用例集约简算法, 引入模糊划分思想, 结合测试需求集, 从各个簇中抽取测试用例, 尽可能地发现相似的用例. 实验结果表明: 算法能够最小化约简用例集, 用例集覆盖范围最广泛, 错误率检测较高.

关键词: 用例约简; 模糊 K -means 算法; 复杂度; 软件测试

中图分类号: TP 311 **文献标志码:** A **文章编号:** 1000-5013(2016)06-0778-04

Fuzzy K -Means Algorithm of Software Testing Using Case Reduction

YU Guoqing^{1,2}, ZHOU Lanrong², LUO Ke¹

(1. Deptment of Computer and Communication Engineering,
Changsha University of Science and Technology, Changsha 410114, China;
2. School of Computer Engineering, Hunan College of Information Technology, Changsha 410200, China)

Abstract: To improve the effectiveness of the software test set reduction, a software test set case reduction algorithm based on fuzzy K means is proposed. The fuzzy partition idea is introduced. The test suite is extracted from each cluster, finded similar cases. Experimental results showed that the algorithm can minimize the reduction case set, covers the most extensive and highly detect the error rate.

Keywords: case reduction; fuzzy K -means software testing; complexity; software testing

随着互联网、云计算等技术的快速发展, 软件功能及规模也迅速增大, 程序开发潜在的错误和漏洞也逐渐增多^[1-3]. 软件测试可以在软件开发完成之后, 使用控制数据、加工数据等用例进行测试, 发现软件存在的错误, 保证软件的完整性、可靠性和准确性. 但是, 大规模软件需要设计较多的测试用例, 造成人工测试模式定位难度大、耗费时间较长, 不能够快速定位和修改软件错误^[4-5]. 软件测试用例约简已经成为降低软件测试用例数量和提高测试成效的关键技术, 许多学者对其进行了研究^[6-11], 引入了集合约简、关联规则、专家数据库等技术, 在保持较高测试准确度的条件下, 大大减少了用例数量. 软件测试用例约简仍存在用例集规模较大、用例重复出现、错误定位精确度小等问题. 本文引入隶属度函数改进 K -means 算法, 旨在利用模糊数学提高软件测试用例划分的准确度.

收稿日期: 2016-10-20

通信作者: 余国清(1971-), 男, 副教授, 主要从事人工智能、智能控制、数据挖掘的研究. E-mail: yuguoqing@mail.hnui.cn.

基金项目: 湖南省科学技术计划项目(2011FJ3086)

1 背景理论

1.1 软件测试用例约简

测试用例需要全覆盖软件的每一个步骤和环节, 针对每一行代码进行测试. 软件测试用例约简是指在用例集 T 中, 寻找一个最简化的子集 T' , 该子集可以满足所有的测试任务.

1.2 模糊 K-means 算法

K-means 算法采用硬聚类划分思想, 划分过程较为武断, 不能够充分地考虑测试用例的多个属性, 导致聚类结果与实际情况存在偏差. 因此, 为了能够提高 K-means 算法的准确度, 假设软件测试用例共有 n 个, 每一个测试用例使用 m 个特征进行刻画, 则引入了隶属度函数, 使用隶属度刻画测试用例归属, 提高数据对象划分准确性, 易于解释和描述. 软件测试用例为

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}. \quad (1)$$

式(1)中: $x_{i,j}$ 表示样本 j 的特征 i , $j=1,2,\dots,n$, $i=1,2,\dots,m$. 为了能够更好的保证 K-means 算法正确执行, 算法运行之前需要对其检修归一化处理, 即

$$r_{i,j} = \frac{x_{i,j} - x_{i,\min}}{x_{i,\max} - x_{i,\min}}. \quad (2)$$

式(2)中: $x_{i,\max}$ 表示第 i 个指标特征的最大取值; $x_{i,\min}$ 表示第 i 个指标特征的最小取值; $r_{i,j}$ 表示归一化后 $x_{i,j}$ 的取值. 软件测试用例集矩阵 \mathbf{X} 归一化之后为

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ r_{m,1} & r_{m,2} & \cdots & r_{m,n} \end{bmatrix} = (r_{i,j})_{m \times n}. \quad (3)$$

软件测试用例集 \mathbf{X} 假设存在 C 个类别, 则数据集 \mathbf{X} 可以使用模糊识别矩阵进行描述, 即

$$\mathbf{U} = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{m,1} & u_{m,2} & \cdots & u_{m,n} \end{bmatrix} = (u_{i,j})_{c \times n}. \quad (4)$$

式(4)中: $u_{i,j}$ 为一个隶属度, 代表软件测试样本 j 归属于某个类别 h 的概率, $h=1,2,\dots,C$, 并且 $u_{h,j}$ 需要满足约束条件 $0 \leq u_{h,j} \leq 1$, $\sum_{h=1}^C u_{h,j} = 1$, $\sum_{j=1}^n u_{h,j} > 0$.

假设软件测试样本划分类别 h 的 m 个特征向量值作为 K-means 算法的中心数据, 则一个具有 \mathbf{X} 个类别的软件测试用例集拥有 C 个模糊聚类中心, 即

$$\mathbf{S} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{bmatrix} = (s_{i,h})_{m \times c}. \quad (5)$$

式(5)中: $s_{i,h}$ 为类别 h 指标 i 的归一化特征值, $0 \leq s_{i,h} \leq 1$.

为了满足不同的聚类用户处理需求, 在模糊聚类执行过程中, 可以针对不同的划分设置不同的特征权值, 以便能够突出某些特征属性的贡献度, 即

$$\mathbf{W} = \{\omega_1, \omega_2, \dots, \omega_m\}, \quad \sum_{i=1}^m \omega_i = 1. \quad (6)$$

模糊 K-means 算法的目标函数形式为

$$\sum_{j=1}^n \sum_{h=1}^c \{u_{h,j} [\sum_{i=1}^m (\omega_i (r_{i,j} - s_{i,h}))^p]^{1/p}\}^2. \quad (7)$$

2 在测试用例机约简中的应用

在模糊 K-means 算法中,假设软件测试用例集为 $T=\{t_1,t_2,\cdots,t_n\}$,数据集拥有的类别数目为 K 个, m_i 为第 i 个簇的中心, $i=1,2,\cdots,k$. 模糊 K-means 的目标函数为

$$F(u_j(t_i)) = \sum_{i=1}^k \sum_{m=1}^n [u_j(t_i)]^b \|t - m_i\|^2. \tag{8}$$

式(8)中: b 为一个常数,可以控制模糊 K-means 隶属度. 通过对模糊 K-means 的隶属度函数求导数,可以获取模糊 K-means 算法的最优解,最优解的求解过程分别为

$$m_j = \frac{\sum_{i=1}^n [u_j(t_i)]^b t_i}{\sum_{i=1}^n [u_j(t_i)]^b}, \quad j = 1, 2, 3, \cdots, k; \tag{9}$$

$$u_j(t_i) = \frac{(1/\|t - m_j\|^2)^{1/(b-1)}}{\sum_{i=1}^k (1/\|t - m_i\|^2)^{1/(b-1)}}, \quad i = 1, 2, 3, \cdots, n; \quad j = 1, 2, 3, \cdots, k. \tag{10}$$

模糊 K-means 算法使用程序,迭代执行最优化求解式(9),(10),可以针对软件测试用例集进行划分. 具体的算法伪代码流程如下. 算法输入:软件测试用例约简类簇数目 K ,隶属度控制常数参数 b , N 个数据对象的软件测试用例集. 算法输出:软件测试用例约简后的 K 个簇. 算法有以下 4 个步骤.

- 步骤 1 随机化的将 N 个软件测试用例分配到 K 个簇中,分派每一个簇的中心特征向量为 m_i .
- 步骤 2 使用式(10)准确计算获取软件测试用例的隶属度函数.
- 步骤 3 使用式(9)重新计算软件测试用例簇的中心值 m_i .
- 步骤 4 计算每一个数据对象的隶属度,直到隶属度不再变化,终止算法运行;否则,返回步骤 2.

由于在 K-means 算法基础上,引入模糊控制 b ,可以根据模糊控制因子的取值经验. 当 $b=0.75$ 时,能够取得较好的约简效果. 基于此,可以分析测试用例的属性相近程度.

3 实验结果分析

3.1 数据集

采用 Siemens Corporate Research 提供的软件测试公共数据集,数据集 1,2,3 包含的用例数为 30;数据集 4,5,6,7 包含的用例数为 32.

3.2 评价标准

约简算法的评价标准有算法约简率(SR)、软件错误检测率(FDE)、软件错误检测丢失率(FL)3 个方法,分别表示为

$$SR = \frac{|T| - |T'|}{|T|} \times 100; \quad FDE = \frac{|F'|}{|F|} \times 100; \quad FL = \frac{|F| - |F'|}{|F|} \times 100.$$

其中: $|T|$ 为软件测试用例集的元素数; $|T'|$ 为约简算法约简后的软件测试用例集中的元素数; $|F|$ 为未约简的用例集测试软件功能时检测出来的错误数; $|F'|$ 为约简后的用例集检测出来的软件功能错误数.

3.3 结果分析

1) 约简率比较. 3 种算法约简率,如图 1 所示. 模糊 K-means 算法引入了模糊控制因子,可以动态地调整隶属度特征权重,将概率值较为接近的软件测试用例划分到一起,在保证全覆盖的条件下,可以有效提高 K-means 算法的约简率,降低软件测试的复杂度和工作量.

2) 软件错误检测率比较. 3 种算法检错率,如图 2 所示. 模糊 K 均值算法约简后的测试用例能够检测出更多软件错误数,表明文中算法约简后的依然可以保持全覆盖能力. 另外两种算法约简后,把一些

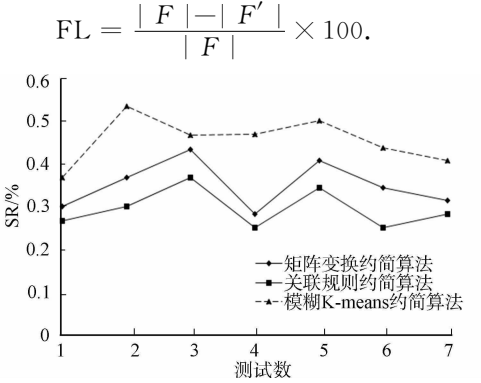


图 1 3 种算法约简率

Fig. 1 Three algorithms reduction rate

测试功能不同的用例划分到一个簇中, 这样造成约简后的用例无法全覆盖软件功能, 错误检测率较低.

3) 错误检测丢失率比较. 3 种算法的错误检测丢失率, 如图 3 所示. 由图 3 可知: 矩阵行列变换算法、关联模式约简算法非常容易将不同覆盖类型的用例约简掉, 因此, 漏检率较低, 算法约简效果不好.

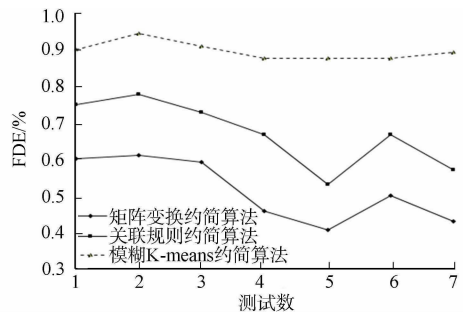


图 2 3 种算法检错率
Fig. 2 Three algorithms error rate

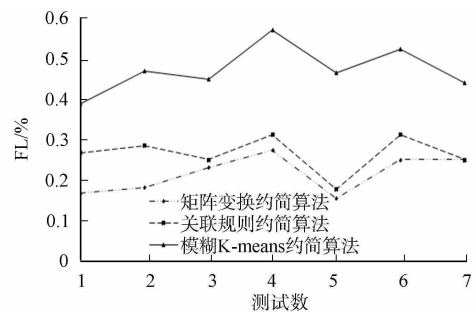


图 3 3 种算法错误检测丢失率
Fig. 3 Three algorithms error detection loss rate

4 结束语

在 K-means 算法中引入了隶属度函数, 采用模糊数学思想提高 K-means 算法划分软件测试用例集, 可以提高用例约简数量, 并且保持用例集检测软件错误的可靠性. 未来研究工作包括两个关键内容: 引入模拟退火理论, 实现模糊控制因子的自动化确定; 针对软件测试用例集进行泛化, 提高算法的稳定性和鲁棒性.

参考文献:

[1] KUMAR G, BHATIA P K. Software testing optimization through test suite reduction using fuzzy clustering[J]. Csi Transactions on Ict, 2013, 1(3): 253-260.

[2] PAKINAM N B, NAGWA L B, MOHAMED H, et al. Test case generation and test data extraction techniques[J]. International Journal of Electrical and Computer Sciences, 2011, 24(5): 112-119.

[3] SUN F, TONG X H, XUE S F. A study of relative redundancy in test-suite reduction while retaining or improving fault-localization effectiveness[C] // Proceedings of the 2010 ACM Symposium on Applied Computing. [S. l.]: ACM, 2010: 2229-2236.

[4] HAO D, XIE T, ZHANG L, et al. Test input reduction for result inspection to facilitate fault localization[J]. Automated Software Engineering, 2010, 17(1): 5-31.

[5] GONG Hongfang, LI Junyi. Generating test cases of cluster-level based on classes dependencies reduction[J]. Journal of Central South University: Science and Technology, 2010, 41(1): 238-244.

[6] GU Qing, TANG Bao, CHEN Daoxu. A test suite reduction technique for partial coverage of test requirements[J]. Chinese Journal of Computers, 2011, 34(5): 879-888.

[7] CHEN Jing, YANG Meihong, WANG Lu, et al. Regression test case reduction model based on association mode[J]. Computer Engineering, 2011, 37(2): 63-65.

[8] ZHOU Chongbo, LOU Jungang, CHENG Long. Test suites reduction based on matrix transformation[J]. Application Research of Computer, 2013, 30(3): 779-782.

[9] CHEN Yangmei, DING Xiaoming. Test suite reduction methods based on K-medoids[J]. Computer Science, 2012, 39(6): 422-424.

[10] SU Xiaohong, GONG Dandan, WANG Tiantian, et al. Automatic fault localization approach combining test case reduction and joint dependency probabilistic model[J]. Journal of Software, 2014, 25(7): 1492-1504.

[11] 刘竹松, 陈洁. 考虑数据不确定性的非均匀挖掘算法[J]. 华侨大学学报(自然科学版), 2016, 37(3): 308-311.

(责任编辑: 陈志贤 英文审校: 吴逢铁)