文章编号:1671-4598(2021)06-0046-05 DOI:10.16526/j.cnki.11—4762/tp.2021.06.010 中图分类号:TP 311 文献标识码:A

基于自适应高斯混合模型的软件测试用例集约简算法研究

杨永国
(中国人民解放军91550部队,辽宁 大连 116023)

摘要:为了最大化的找出软件测试用例集中的相似用例,实现对用例的最优精简,提出了一种自适应的高斯混合模型;提出的模型使用K—means初始化EM,自适应地确定聚类簇数目,在此过程中能够评判聚类结果,同时给出式高斯混合模型的所有参数,这些参数作为各个聚类簇进行新一轮迭代计算的参数,最终得到的结果更趋于最优解;实验结果表明,相对现有的高斯混合模型和模糊K—Means聚类模型等算法,文章提出的自适应高斯混合模型算法能够最小化软件测试用例集,约简后的用例所覆盖的范围相对更广,测试出的软件错误率较高,对软件测试用例集多变的适应性好。

关键词:软件测试;用例约简;高斯混合模型;自适应

ResearchonSoftware Testing Case Reduction Algorithm Based on AdaptiveGaussian Mixture Model

Yang Yongguo
(91550 Troops of the Chinese People's Liberation Army, Dalian 116023, China)

Abstract: In order to find out the similar test cases in the software test case set and realize the optimal simplification of test cases, an adaptive Gaussian mixture model is proposed. The proposed model uses K—means to initialize EM, adaptively determines the number of clusters. In this process, the clustering results can be evaluated. At the same time, all the parameters of the Gaussian mixture model are given. These parameters are used as the parameters of each cluster for a new round of iterative calculation, and the final results tend to be the optimal solution. The experimental results show that, compared with the existing Gaussian mixture model and fuzzy K—means clustering model, the adaptive Gaussian mixture model algorithm proposed in this paper can minimize the software test case set, and the reduced cases cover a wider range. The software error rate is higher, and it has good adaptability to the variety of software test case set.

Keywords: software testing; case reduction; Gaussian mixture model; adaptive

0 引言

软件规模越大,其逻辑复杂性越高,对这些软件的可靠性要求也就越高。软件的隐藏错误能够造成其自身运行的失败。也就是说软件的隐藏错误是影响软件可靠性的最关键因素之一[1]。对于小规模的软件,工程实践中大多采用人工设置断点的方法来进行测试,以查找出错误。不过人工方法判断错误位置较为复杂,且难度较大,不适用于大规模的软件测试。因此,为了更加精准且高效地查找和消除软件错误,专家学者们开展了广泛的研究,提出了一些解决方法,并研发出对软件进行单元、静态和动态测试的工具软件。这些方法和测试软件,都要求测试用例集尽可能覆盖全面,只有这样才能精准地定位错误,提高测试的有效性[2-4]。但测试用例集包含的用例数量应该尽量约简,以降低测试成本。

刘锋等人提出基于向量相似度的测试用例集约简方法[5]。张蕊等人提出一种基于搜索树的用例约简方法,求得约简问题的全局最优解[6]。谢经纬在测试用例集约简中引入需求分析,提出了两阶段用例集优化方法,分为需求切片和用例集优化两个步骤[7]。张晨光等人提出一种在线测试用例集约简方法,将测试集约简嵌入测试生成流程内,测试生成过程为测试集约简提供了测试序列与测试目标之间的满足关系[8]。杨羊等人通过定义和合并基于ASM模型测试生成的等价迁移和等价状态,减少了无效访问状态和无效迁移路径的数量,实现了测试用例集空间的约简[9]。苏小红等人采用面向错误定位需求的测试用例约简方法,降低错误定位的复杂度,提高错误定位的精度[10]。

这些用来约简测试用例集的方法还存在一些主要问题如下:1)如何建模测试用例集之间的关系,以提高查找软件错误的准确度;2)如何实现参数设置的简化,或者实现自适应的参数设置。为了解决这些问题,本文提出了一种基于自适应高斯混合模型的用例约简算法。该算法引入高

收稿日期:2021-03-28; 修回日期:2021-04-07。
作者简介:杨永国(1976-),男,内蒙古通辽人,大学本科,高级工程师,主要从事指挥控制系统、实时测控方向的研究。
引用格式:杨永国.基于自适应高斯混合模型的软件测试用例集约简算法研究[J].计算机测量与控制,2021,29(6):46-50.

斯混合模型，寻找测试用例间的关系，抽取满足测试需求的测试用例，实现用例集的约简；同时通过自适应策略，简化参数设置。

1　高斯混合模型

软件测试用例集数据的概率分布通常很复杂，因此引入高斯混合模型来模拟逼近和约简软件测试用例集，以简化问题[11-14]。高斯混合模型（GMM，gaussian mixture model）是 M 个高斯模型（聚类簇）的加权和，其对数据的 M 类概率密度分布进行高斯估计。每个高斯概率密度函数都与一个类对应。训练时采用了期望最大（EM，expectation maximization）算法。

设每个样本都对应于一个类，也就是对应于一个高斯概率密度函数，而整个样本集对应 M 个高斯概率密度函数。但具体每个样本 x_i 对应于哪个高斯概率密度函数不确定，在 GMM 中，每个高斯概率密度函数所占的权重 φ_j 也不确定。式（1）所示为 GMM：

$$p(x_i)=\sum_{j=1}^{M}\varphi_j f_j(x_j/\mu_j,\sum\nolimits_j) \tag{1}$$

其中：$\varphi_j\geqslant 0$，$\sum_{j=1}^{M}\varphi_j=1$；$f_j$ 为第 j 个高斯概率密度函数，均值为 μ_j，方差为 $\sum_j$。

每一个高斯概率密度函数，也就是单高斯模型都有 3 个参数 φ_j、μ_j、$\sum_j$。进行高斯混合模型建模时，就要确定 $3N$ 个参数。

需要通过样本集 $X=\{x_1,\ x_i,\ \cdots,\ x_N\}$ 来估计 GMM 的所有参数[15] $\Phi=(\varphi_1,\ \varphi_j,\ \cdots,\ \varphi_M)^T$，样本 x_j 的概率如式（2）所示：

$$p(X\mid\Phi)\prod_{i=1}^{N}\sum_{j=1}^{M}\varphi_j N_j(x_i\mid\mu_j,\boldsymbol{C}_j) \tag{2}$$

其中：$\boldsymbol{C}_j$ 为协方差矩阵。

对 GMM 参数进行估计，目前最多采用的是 EM 算法，其具体流程如下[15-19]：

1）初始化。用单位矩阵作初始化 $\boldsymbol{C}_j$，用 $\frac{1}{M}$ 初始化 φ_j，用随机数初始化 $\mu_0=\frac{1}{N}\sum_{i=1}^{N}x_i$。

2）后验概率估计步骤。用式（3）估计 φ_j 的后验概率：

$$\beta_{ij}=\frac{\varphi_j N_j(x;\Phi)}{\sum_{p=1}^{M}\varphi_p N_p(x_i;\Phi)} \tag{3}$$

3）最大化步骤。根据式（4）对权值 φ_j 进行更新：

$$\varphi_j=\sum\nolimits_{i=1}^{N}\beta_{ij}/N \tag{4}$$

根据式（5）对均值 μ_j 进行更新：

$$\mu_j=\sum\nolimits_{i=1}^{N}x_i\beta_{ij}/\sum\nolimits_{i=1}^{N}\beta_{ij} \tag{5}$$

根据式（6）对方差矩阵 $\boldsymbol{C}_j$ 进行更新：

$$\boldsymbol{C}_j=\frac{\sum_{i=1}^{N}\beta_{ij}(x_i-\mu_j^T)(x_i-\mu_j^T)^T}{\sum_{i=1}^{N}\beta_{ij}} \tag{6}$$

4）收敛条件。迭代计算步骤 2）和步骤 3），权值 φ_j、均值 μ_j 和方差矩阵 $\boldsymbol{C}_j$ 会不断地更新，当 $p(X\mid\Phi)-p(X\mid\Phi)'<\varepsilon$ 时停止进行迭代。$p(X\mid\Phi)'$ 为更新权值 φ_j、μ_j 均值和方差矩阵 $\boldsymbol{C}_j$ 后根据式（3）计算的值，ε 为阈值，一般 $\varepsilon=10^{-5}$。

2　基于自适应高斯混合模型

但采用的高斯混合模型聚类簇数目通常是经验值，很容易造成聚类准确度的降低。具体到软件测试用例集约简的场景，经验聚类簇数目对用例集多变的适应性较差，无法做到自适应。因此本文对高斯混合模型进行改进，使其可以自适应的确定聚类簇数目。

改进的思想为：使用 K－means 初始化 EM，自适应地确定聚类簇数目，在此过程中能够评判聚类结果，同时给出式高斯混合模型的所有参数，这些参数作为各个聚类簇进行新一轮迭代计算的参数，最终得到的结果更趋于最优解[15]。

针对软件测试用例集约简研究，为解决现有高斯混合模型聚类算法只能使用经验聚类簇数目值的缺陷，根据上述改进思想，本文使用了自适应高斯混合模型算法，算法的具体实现步骤如下所述，实现流程如图 1 所述。

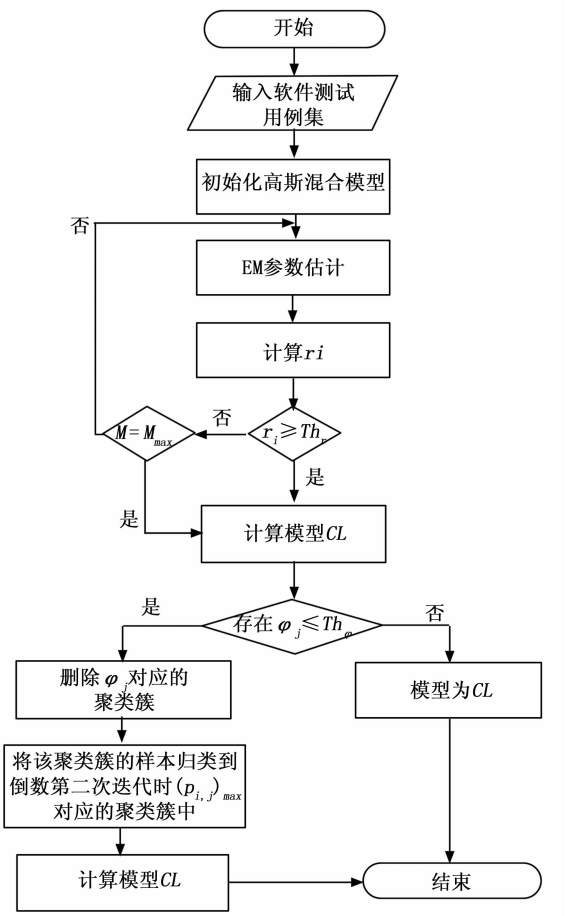

图 1　自适应高斯混合模型的实现流程

1）先按经验设置 M_0 个聚类簇，按照第 1 章节中的步骤（1）进行初始化协方差矩阵 $\boldsymbol{C}_j$、每个高斯概率密度函数

所占的权重 φ_j 和均值 μ_j，并用 EM 算法对模型进行优化，计算出初始化聚类结果。

2）在后续迭代计算的过程中，计算出第 i 个样本 x_i 属于第 j 个聚类簇的概率 $p_{i,j}$。根据式（7）计算最大的 $(p_{i,j})_{\max}$ 与排第二的（$p_{i,j\max-1}$）的比值 r_i。第 i 个样本 x_i 聚类到某个聚类簇，要求 $(p_{i,j})_{\max}$ 相对于 $(p_{i,j})_{\max-1}$ 要有明显的差别，即 $(p_{i,j})_{\max}$ 应当比 $(p_{i,j})_{\max-1}$ 大很多。也就是可以用 r_i 来衡量聚类结果的优劣。r_i 越小，聚类结果越好；r_i 越大，聚类结果越差。

3）设定阈值 Th_r。通过给定 Th 的值，能够衡量类聚类结果是否满足要求。当 $r_i \leqslant Th_r$ 时，第 i 个样本 x_i 可以归类为 $(p_{i,j})_{\max}$ 对应的聚类簇；当 $r_i > Th_r$ 时，第 i 个样本 x_i 不适合归类为目前的所有聚类簇，需要增加聚类簇数目。聚类簇数目不能无限的增加下去，否则就失去了聚类的意义，需要设定一个最大聚类簇数目值 $M_{\max}$。当聚类簇数目数目达到 $M_{\max}$ 时，则第 i 个样本可以归类为 $(p_{i,j})_{\max}$ 对应的聚类簇，而不再新增聚类簇的数目。

4）当聚类簇数目增加后，需要对新模型的参数进行初始化调整。迭代的重新执行步骤 1）～3），当聚类簇数目不再增加，或者达到最大时，完成模型训练，得到模型参数。

5）当步骤 4）得到的模型中，存在 φ_j 很小的聚类簇，即 $\varphi_j \leqslant Th_\varphi$ 时，可将该聚类簇删除，并将该聚类簇对应的样本 x_j 归类到倒数第二次迭代时 $(p_{i,j})_{\max}$ 对应的聚类簇中。因为聚类簇的 $\varphi_j \leqslant Th_\varphi$ 时，可以认为该聚类簇的代表性不足。然后对模型进行计算，得到最终的模型 CL，及模型参数。

3 实验结果与分析

3.1 实验数据集

本文采用使用广泛并且便于比较的西门子软件测试用例集[1,20]。该由 Siemens Corporate Research 的专家开发，含有 7 类程序代码 Print _ tokens1、Print _ tokens2、Schedule1、Schedule2、Replace、tacs 和 tot _ info，每类程序代码有 1 个无错误版本和一些含有错误的版本。软件测试用例集中每类程序代码的行数、初始测试用例数、程序中错误数、测试用例数如表 1 所示[1]。针对每类程序代码，该软件测试用例集使用了 5 种类型编程语言进行的代码编写，满足常用编程语言需求的代码需求。

表 1 软件测试用例约简用数据集

数据集	初始用例数	错误数	行数	测试用例数
Print_tokens1	30	20	565	4 130
Print_tokens2	30	18	510	4 115
Schedule1	32	24	412	2 650
Schedule2	32	32	307	2 710
Replace	30	22	563	5 542
tacs	32	24	173	1 608
tot_info	32	28	406	1 502

3.2 评价标准

实验中，采用约简率、错误检测率和错误检测丢失率这三个指标来评价实验结果，以验证算法的有效性、正确性和稳定性。

3.2.1 约简率

约简率计算如式（8）所示，为现有软件测试用例集中的测试用例数目与约简后软件测试用例集中的测试用例数目的差值，比上现有软件测试用例集中的测试用例数目，得到的比值[1]。

$$RR = \frac{USN - USN'}{USN} \times 100\% \tag{8}$$

其中：RR 为约简率，$|USN|$ 表示现有软件测试用例集中的测试用例数目；$|USN'|$ 表示约简后软件测试用例集中的测试用例数目。

3.2.2 错误检测率

错误检测率计算如式（9）所示，为约简学习初始用例后，从测试用例中检测到的程序代码错误个数与测试用例中实际程序代码错误个数的比值[1]。

$$EDR = \frac{EN'}{EN} \times 100\% \tag{9}$$

其中：EDR 为错误检测率，EN 表示现有软件测试用例集的测试用例包含的程序代码错误数；EN' 表示使用约简算法学习初始用例后，从软件测试用例集的测试用例中检测出的程序代码错误数。

3.2.3 错误检测丢失率

错误检测丢失率计算如式（10）所示，为约简学习初始用例后，从测试用例中检测到的程序代码错误个数与测试用例中实际程序代码错误个数的差值，比上测试用例中实际程序代码错误个数，得到的比值[1]。

$$ELR = \frac{|EN| - |EN'|}{|EN|} \times 100\% \tag{10}$$

其中：ELR 为错误检测丢失率。

由式（8）、式（9）和式（10）可以比较高斯混合模型、模糊 K－Means 聚类模型和本文提出算法的约简率、错误检测率和误差检测丢失率。

3.3 实验结果分析

为了验证本文提出的基于自适应高斯混合模型的软件测试用例集简约算法的性能，对比分析了本文提出算法、基于高斯混合模型的软件测试用例集简约算法和基于模糊 K－Means 聚类模型的软件测试用例集简约算法。

3.3.1 约简率分析

使用 3 种算法约简后，得到的软件测试用例集中的测试用例数量如表 2 所示[1]。

3 种算法约简率如表 3 所示[1]。通过分析，基于模糊 K－Means 聚类模型的软件测试用例集简约算法的约简率优于基于高斯混合模型的软件测试用例集简约算法。本文提出算法的约简率高于上述两种算法，其总约简率相对基于模糊 K－Means 聚类模型的软件测试用例集简约算法高

11.46%，相对基于高斯混合模型的软件测试用例集简约算法高 6.42%。约简率的提升，可以在一定程度上降低软件测试的复杂度，提高软件测试的效率。

表 2　约简后的初始用例数

数据集	初始用例数	本文算法	高斯混合	模糊 K－Means
Print_tokens1	30	17	23	19
Print_tokens2	30	13	17	14
Schedule1	32	14	20	17
Schedule2	32	12	15	16
Replace	30	15	14	16
tacs	32	18	21	18
tot_info	32	16	20	19
Total	218	105	130	119

表 3　3 种算法的约简率

数据集	约简率/%		
	本文算法	高斯混合	模糊 K－Means[1]
Print_tokens1	43.33	23.33	36.67
Print_tokens2	56.67	43.33	53.33
Schedule1	56.25	37.50	46.88
Schedule2	62.5	53.13	50.00
Replace	50.00	53.33	46.66
tacs	43.75	34.38	43.75
tot_info	50.00	37.50	40.63
Total	51.83	40.37	45.41

3.3.2　错误检测率分析

使用 3 种算法约简后，从测试用例集中检测出来的程序代码错误数量如表 4 所示[1]。

表 4　约简后的检测出的错误数

数据集	错误数	本文算法	高斯混合	模糊 K－Means[1]
Print_tokens1	20	19	14	18
Print_tokens2	18	16	13	17
Schedule1	24	23	13	21
Schedule2	32	31	15	28
Replace	22	20	14	20
tacs	24	23	15	21
tot_info	28	26	16	25
Total	168	158	100	150

3 种算法的错误检测率如表 5 所示。从表中数据可知，基于模糊 K－Means 聚类模型的软件测试用例集简约算法的错误检测率优于基于高斯混合模型的软件测试用例集简约算法。本文提出算法的错误检测率高于上述两种算法，其总错误检测率相对基于模糊 K－Means 聚类模型的软件测试用例集简约算法高 34.53%，相对基于高斯混合模型的软件测试用例集简约算法高 4.76%。使用本文提出算法能够检测出更多的错误数，可以提高软件测试的可靠性，进而提高被测软件的正确性。

表 5　3 种算法的错误检测率

数据集	错误检测率/%		
	本文算法	高斯混合	模糊 K－Means[1]
Print_tokens1	95.00	70.00	90.00
Print_tokens2	88.88	72.22	94.44
Schedule1	95.83	54.17	87.50
Schedule2	96.86	46.66	87.50
Replace	90.91	77.27	90.91
tacs	96.83	62.50	87.50
tot_info	92.86	57.14	89.29
Total	94.05	59.52	89.29

3.3.3　错误检测丢失率分析

3 种算法的错误检测丢失率如表 6 所示。

表 6　3 种算法的错误检测丢失率

数据集	错误检测丢失率/%		
	本文算法	高斯混合	模糊 K－Means[1]
Print_tokens1	5.00	30.00	10.00
Print_tokens2	11.12	27.78	5.56
Schedule1	4.17	45.83	12.50
Schedule2	3.14	53.34	12.50
Replace	9.09	22.73	9.09
tacs	3.17	37.5	12.50
tot_info	7.14	42.86	10.71
Total	5.95	40.48	10.71

从表中数据可知，基于模糊 K－Means 聚类模型的软件测试用例集简约算法的错误检测丢失率优于基于高斯混合模型的软件测试用例集简约算法。本文提出算法的错误检测丢失率低于上述两种算法。错误检测丢失率低，代表漏检的少错误数量，错误检测丢失率越低越好。相对其它两种算法，本文提出算法的错误检测丢失率最低，这体现了提出算法约简得到的测试用例集对程序代码错误的覆盖度较高。

4　结束语

软件测试用例集聚类和约简是一个时研时新且充满挑战的研究方向。当前软件测试用例集聚类、约简方法存在需要根据经验给出聚类簇数目，导致自适应性不强的问题。本文提出了一种改进的高斯混合模型算法应用于软件测试用例集约简，以自适应的确定聚类簇的数目，并评估聚类效果的优劣。

提出的自适应高斯混合模型算法的优点是：算法在初始化和迭代过程中无需固定聚类簇数目，可以根据不同软件测试用例集数据的特点自适应确定聚类簇数目，提升聚类的自适应性、准确性和泛化性，实现用例集最优约简化。实验结果表明，相对于高斯混合模型算法和模糊 K－means 聚类算法，提出算法的约简率和错误检测率更高。约简后虽然软件测试用例集规模精简，但覆盖率高。

参考文献：

[1] 高丑光，林 都，鲜 浩. 基于 K 均值的软件测试集用例约简算法研究 [J]. 微电子学与计算机，2016，33 (5)：133 - 136，141.

[2] 左万娟，虞砺琨，王小丽，等. 航天嵌入式软件测试用例典型设计缺陷研究 [J]. 计算机测量与控制，2019，27 (10)：36 - 40.

[3] 陈 强，陈 双，吴立金，等. 分布式复杂系统软件测试建模方法与应用研究 [J]. 计算机测量与控制，2019，27 (2)：129 - 134.

[4] 张倩宜，李 妍. 基于改进 PSO 算法的路径软件测试用例生成方法 [J]. 计算机测量与控制，2018，26 (3)：50 - 53.

[5] 刘 锋，李 朋，朱二周. 基于向量相似度的测试用例集约简方法 [J]. 计算机与网络，2017，34 (3)：35 - 39.

[6] 张 蕊，薛 黎. 搜索树在测试用例约简中的应用 [J]. 计算机与网络，2016，42 (23)：48.

[7] 谢经纬. 基于需求分析的两阶段测试用例集优化研究 [D]. 长沙：湖南大学，2011.

[8] 张晨光，徐 珞，李 宁. 一种在线测试集约简方法 [J]. 计算机与现代化，2018，280 (12)：40 - 44，50.

[9] 杨 羊，何柳柳，尚 颖，等. 面向测试生成的 ASM 模型约简研究 [J]. 计算机工程与科学，2018，40 (6)：1084 - 1091.

[10] 苏小红，龚丹丹，王甜甜. 结合用例约简与联合依赖概率建模的错误定位 [J]. 软件学报，2014，25 (7)：1492 - 1504.

[11] 张万星，王 炜，白立辉. 基于高斯混合模型的逆变器故障诊断方法研究 [J]. 计算机测量与控制，2020，28 (3)：14 - 18.

[12] 唐 榜，吴 珏，杨福军，等. 基于高斯混合模型的 Web 代理服务器缓存替换策略 [J]. 计算机测量与控制，2021，29 (2)：166 - 170.

[13] 张婧懿，隋思逸. 基于高斯混合模型联合 CamShift 的运动图像检测跟踪方法 [J]. 计算机测量与控制，2018，26 (5)：59 - 61，65.

[14] 唐洪良，黄 颖，黄 淮，等. 改进的自适应高斯混合模型运动目标检测算法 [J]. 现代电子技术，2017，40 (11)：65 - 67.

[15] 葛丽阁，孙 伟. 自适应高斯混合模型海上移动对象浮标轨迹聚类研究 [J]. 现代计算机，2018，23：3 - 8.

[16] 柯文俊，王泊涵，杜泽峰，等. 一种无监督的软件复杂度度量与评估模型 [J]. 高技术通讯，2020，30 (4)：333 - 341.

[17] 张美霞，李 丽，杨 秀，等. 基于高斯混合模型聚类和多维尺度分析的负荷分类方法 [J]. 电网技术，2020，44 (11)：262 - 275.

[18] 陈 刚，王丽娟. 基于高斯混合模型的非平衡数据对称翻转算法 [J]. 信息与控制，2020 (2)：203 - 209.

[19] 梁盛楠. 基于 EM 算法的高斯混合模型参数估计 [J]. 黔南民族师范学院学报，2020 (4)：5 - 8.

[20] 林晶昱，赵逢禹. 一种计算缺陷贡献率的代码缺陷定位规则 [J]. 计算机应用研究，2015，32 (9)：2702 - 2707.

(上接第 45 页)

提出的算法进行了稳定性和收敛性的论证推导和仿真分析，仿真结果表明，所提出的三阶相关峭度反卷积算法对恢复信号的信噪比、收敛速度、稳定性都有不错的效果，通过少数几次迭代就能获得相关峭度最大值，不需要对信号进行任何的预处理，因为该算法针对的是非平稳信号，所以其适用范围是比较广的，能够解决很多实际中产生的问题，应用于薄膜生产监测系统时能够根据峭度监测出故障点所在位置。但是该算法使用的是三阶累积量，其计算量较大，复杂度较高，在程序仿真时，由于算法的自适应能力问题，需要多次进行实验仿真，需要控制噪声大小，因此在以后希望能对该算法进行优化。

参考文献：

[1] 陈 明，马 洁. ALIF 和 MCKD 相结合的滚动轴承早期故障诊断 [J/OL]. 机械科学与技术，2020：1 - 8[2020 - 10 - 17]. https://doi.org/10.13433/j.cnki.1003－8728.20200182.

[2] 张守成，李宏伟，刘永凯. 一单元 ICA－R 快速算法 [J]. 计算机工程与应用，2009，45 (2)：158 - 161.

[3] 张洪梅，邹金慧. 自适应 MCKD 和 CEEMDAN 的滚动轴承微弱故障特征提取 [J]. 电子测量与仪器学报，2019，33 (4)：79 - 86.

[4] Cichocki A，Rutkowski T，Siwek K. Blind signal extraction of signals with specified frequency band [A]. Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing [C]. Switzerland. USA：IEEE，2002：515 - 524.

[5] 季 策，王艳茹，王晓宇. 一种基于标准峭度的新型复数盲分离算法 [J]. 东北大学学报 (自然科学版)，2015，36 (5)：614 - 617.

[6] 陈寿齐，沈越泓，马 明，等. 基于峭度的稳健特定信号盲提取 [J]. 系统仿真学报，2008 (22)：6093 - 6096.

[7] 肖健华. 基于支持对象的野点检测方法 [J]. 计算机工程，2003 (11)：43 - 45.

[8] 杨光永，胡国清，陈 乐，等. 用于信号盲提取的最大三阶相关峭度反卷积算法 [J]. 华南理工大学学报 (自然科学版)，2014，42 (1)：3 - 6.

[9] 杨光永. 基于激光位移测量的光学压力传感器研究 [D]. 广州：华南理工大学，2014.

[10] 邹某谈. 反卷积与信号复原 [M]. 北京：国防工业出版社，2001.

[11] 任子良，秦 勇. 一种噪声未知条件下的盲信号提取方法 [J]. 电子科技大学学报，2018，47 (5)：646 - 653.

[12] 彭 聪，刘 彬，周 乾. 基于机器视觉和盲源分离的机械故障检测 [J]. 上海交通大学报，2020，54 (9)：953 - 960.

[13] Palade V，Bocaniala C D. Computational intelligence in fault diagnosis [M]. Springer London，2006.