文章编号 1671-2730(2010)03-0162-05

机器学习在软件测试用例集优化生成中的应用

胡 静，赵 莹

(上海电机学院 电子信息学院，上海 200240)

摘 要：软件测试用例集生成是软件测试中的重要环节。如何优化软件测试用例集，有效提高软件测试用例集的质量，已成为提高软件测试效率的主要手段。机器学习是解决这类问题的有效方法，目前也取得了许多重大的成果。本文对机器学习在软件测试用例集优化生成中的国内外研究现状及存在问题进行了分析，并在此基础上对机器学习与软件测试用例优化结合的发展趋势提出了看法。

关键词：软件测试用例集；机器学习；遗传算法；流形学习；测试用例优化生成

中图分类号：TP 18；TP 311.56 **文献标识码**：A

Application of Machine Learning in Optimized Generation of Software Test Suite

HU Jing， *ZHAO Ying*

(School of Electrics and Information，Shanghai Dianji University，Shanghai 200240，China)

Abstract：Software test suite generation is important in software testing. Improving quality of test suite is a key in software testing. Machine learning is an effective way in solving the problem，and have a lot of results. This paper introduces some current research，methods and problems in machine learning for optimized generation test suites，and provides some opinions.

Key words：software test suite；machine learning；genetic algorithms；manifold learning；optimized generation of test suite

软件测试是为了发现错误而执行程序的过程，其中，设计和生成有效的测试用例是决定软件测试质量的重要保证[1]。选取一批对错误敏感的测试用例可有效地提高发现软件错误的可能性，从而降低软件测试成本[2]。因此，如何发现对错误敏感的测试用例一直都是软件测试人员所研究的重要课题。

测试用例的生成可以被理解成一个数据抽样的过程，即根据相应的数据测试覆盖标准，采用一定的方法，在数据集中进行抽样，以获取一批高质

收稿日期：2010-02-26

基金项目：上海市教育委员会科研创新项目(09Y2479)

作者简介：胡 静(1964-)，女，副教授，博士，专业方向为机器学习与智能信息处理，E-mail：hujing@sdju.edu.cn

量的、对随后的测试过程贡献较大的测试用例[3]。这个过程可以看成是一个学习过程，通过对测试用例总集的不断学习，从中获取高质量的用例子集，并尽可能地降低测试用例的总数，节省测试成本。因而机器学习的方法自然而然地就成为了人们关注的焦点，各种机器学习的方法和成果被用来设计与生成软件测试用例，并取得了相应的研究成果，使得机器学习的方法在软件测试中发挥了越来越大的作用[4-5]。

1　国内外研究现状

软件测试方法是为了更好地实现软件测试的目的而提出的途径和优良做法，其内容非常丰富，依据不同的目的、不同的角度有许多不同的分类结果。其中以功能测试和结构测试最为经典，本文以这两种测试方法为主，介绍软件测试与机器学习二者的结合。

在功能测试领域中，主要是以检查程序是否正确执行规格说明书的手段来完成测试任务，测试用例的生成是根据规格说明来选择的。早期的测试由于缺乏形式化标准的规格说明，不能很好地完成测试要求。为了尽可能多地发现被测系统中的缺陷，最理想的情况是将系统中所有合法和不合法的输入情况全部执行一遍，故一般采用人工方式选择测试用例集。在利用机器学习方法优化测试用例集方面，Cohen 等[6-8]提出了一种数据启发方法，根据某种启发性原则来选取测试用例集，其中启发性原则的好坏直接影响测试的结果，同时开发了相应的测试数据自动生成系统 AETG (Automatic Efficient Test Generate)。Lei 等[9]利用贪心算法，提出一种依据参数顺序来组合测试数据生成的方法。该算法的主要思想是每次从测试用例集中挑选出能满足最多测试需求的测试用例，然后在测试需求中去掉被该测试用例满足的测试需求，直到所有测试需求都被满足，这时可以认为挑选出来的测试用例所组成的集合就是最优测试用例集。该方法也开发了相应的测试数据生成工具 PAIRTEST。此外，Cohen 等[10-11]将模拟退火法应用于测试数据生成；Schrocder 等[12]提出了一种测试数据约简方法，利用系统的输入-输出关系等附加信息，在不降低错误检测能力前提下，对测试数据集进行约简。

在结构测试领域中，主要根据程序的内部结构设计测试用例，检测程序的每条路径是否都按照预定的要求正确执行。结构测试要求对被测程序的结构特性做到一定程度的覆盖，如语句覆盖、分支覆盖、数据流覆盖、路径覆盖等。其测试数据生成是在输入域中，寻找满足测试准则的输入数据的过程。

结构测试用例的生成较多地受到符号执行等自动化技术的局限，不能自动产生达到满意覆盖程度的用例集，故测试人员需要手工生成数据以覆盖一些特定的目标。随机测试虽然可以提高自动化程度，但是它会产生大量的测试数据，查错率也不是很高[13]。

机器学习方法在结构测试领域的应用主要以遗传算法为主。遗传算法在机器学习领域是一个非常有前途的学习算法，它将测试用例的生成问题转化为一个利用遗传算法进行数值优化的问题。其中，被搜索的解空间就是测试数据空间，最优解就是满足特定测试目标测试数据。该搜索过程可以完全自动化，从而有助于提高测试效率。在遗传算法中，适应度函数根据测试数据对结构性元素的覆盖程度来计算测试数据的重要性。Poper 等[14]提出了一种基于程序流程图的适应度函数构造方法，适应于语句覆盖与分支覆盖。Weichselbaum 等[15]提出了一种新的进化测试方法，可用于语句、分支与条件覆盖。针对不同的测试目标，研究者们[16-17]提出了一系列的适应度值构造方法，可用于语句与分支的覆盖。此外，还有一些方法可用于条件覆盖[18]。

机器学习与软件测试技术结合在我国虽然起步较晚，但随着软件测试在我国许多部门越来越受到重视，关于机器学习与测试用例集优化研究也越来越被重视起来，并取得了一些成果。如文献[19]提出了基于贝叶斯统计推断理论的软件测试用例估计方法，以贝叶斯的统计理论为依据，在对随机测试过程进行形式化描述的基础上，确定测试用例分布的总体信息。文献[20]提出了应用模糊数学理论，给出选取测试用例的方法，并证明

此方法有效。

上述研究或从输入域的角度，探索利用机器学习的方法精简输入集，节省了测试时间，提高了测试效率；或从输入-输出对的角度，探索测试数据与测试目标之间的距离，试图缩短从输入空间到输出空间的有效路径，达到整体提高测试效率的目的。每个方法都取得了一定的成果，也存在着一些问题。

2 存在的若干问题

在结构测试领域，机器学习方法是精选少量的测试用例的科学有效的方法，也是研究者们研究的重点[21-22]，也取得了一些成果。但是，由于最优的测试用例生成是一个多项式复杂度的非确定性难题(NP)，所以大多数方法都只是提供了近似解，每种方法在特定情况下都存在着不足之处。

(1) 遗传算法、模拟退火法等优化方法最大的特点就是可以很快地生成满足测试目标的测试数据，且检错能力也大大提高。但是，大多数的研究仅针对控制流测试中的语句覆盖和分支覆盖，对于路径覆盖的研究较少。此外，目前的研究仅限于单元测试领域，较少深入到集成测试领域中。

(2) 遗传算法中，适应度函数设计是决定遗传算法执行结果好坏的关键。在面向结构性测试的应用中，适应度函数的设计一般遵循如下规则[6]：适应度函数根据数据对结构性元素的覆盖程度来计算测试数据的适应度值。此类方法先要构造测试程序流程图，然后利用测试数据执行轨迹来计算相应的适应值。其适应度函数 F 的构造为

$$F(\text{测试},\text{对象})=\frac{\text{已被测试执行覆盖的对象路径个数}}{\text{需要被测试执行覆盖的总对象路径个数}} \quad (1)$$

F 取值为$[0,1]$。适应度函数的设计不仅能考虑测试过程中非常重要的覆盖因素，而且还能考虑测试过程中的空间和时间因素，则所设计的适应度值将更具有通用性。若适应度函数的设计思路设计为

$$F(T_j,K_j)=\frac{\mathrm{cov}(T_j,K_j)}{\mathrm{cost}(T_j,K_j)} \quad (2)$$

式中，T_j 为时间参数，$j\in[0,1]$；K_j 为空间参数；$\mathrm{cost}(T_j,K_j)$的值与空间、时间都相关，即与内存的消耗量成反比，与程序运行的速度成正比；而$\mathrm{cov}(T_j,K_j)$的值取决于测试用例所覆盖到的路径的价值，被覆盖的路经越新，其分数就越高。但是目前在遗传算法与测试用例生成的结合中，按照这种思想来设计适应度函数的应用还较少。

贝叶斯理论可以简单地被描述成一条原则：为了预见未来，必须要看过去。某件事情发生的概率大致可由它过去发生的频率近似估计出来。贝叶斯方法的最大特点是方法简单，理论依据充分，模型假设易于理解等，但是其缺点也很明显，即它的先验分布函数估计建立在随机的先验分布函数基础上，受随机测试数据选取影响较大，虽然要求尽可能地选取均匀分布的随机测试数据，却给随机测试也带来了一定的难度。

启发式算法是一种根据某种启发性原则来选取测试用例的选择性测试方法，启发性原则的好坏直接影响测试的结果。对于一些测试数据，较难自动地构造出启发性原则，故如何构造合适的启发性选择原则，也是研究人员研究的重点所在。

3 分析与展望

一个成功的测试用例集首先应该满足充分性，即待测软件在这个有限的测试用例集上的行为，应该充分体现软件在整个输入空间的行为。无论理论研究还是实验结果都表明，对于相同数量的测试数据，结合机器学习方法所产生的测试数据，在发现错误的能力方面明显要高于随机性所产生的测试数据。但是到底能提高多少还是一个未知数，即对利用某种机器学习方法而挑选出来的测试用例集，其价值到底有多高？每个测试用例对测试的贡献有多大？还缺乏一个根本的度量。

机器学习与具体的领域知识相结合，以此来解决一些实际问题，其主要途径有：① 整体建模方式；② 局部建模方法。目前机器学习方法在软件测试用例生成方面的研究大多采用整体建模的方法，实际上就是先假设满足测试要求的所有用例集存在，然后在所有测试用例集上选择较小的

较优的用例集。此外，局部建模的方法在解决非线性、海量数据处理方面有着比整体建模方法更大的优势，如流形学习法、集群学习法等。从测试角度看，某些软件系统功能庞大，参数复杂，满足测试要求的测试用例具有数量大、维数高的特点。解决这类软件系统的测试用例精简问题一般难度都较大，如遗传算法中适应度函数的设计，启发式方法中启发性信息的确定等。

作为机器学习的重要研究领域——流形学习的主要目的在于发现并学习数据集里的内在规律与性质，完成或协助完成数据挖掘等各项任务[23]。流形学习和软件测试用例精简看起来是两个不同的概念，但两者之间存在一定的联系。

假设所有测试用例的集合记为 U_{all}，被测软件系统如果有 N 个测试参数，每个参数有 T 个取值，每个取值里又有 t 个元素，且任何一个测试用例都不是其他测试用例的线性组合，显然，$U_{all} \in \boldsymbol{R}^n$（$\boldsymbol{R}^n$ 为实数高维空间）具有高维非线性的特点。同时，又由于它们属于同一被测软件，因此，从高维空间角度看，它们具有连续分布的特性，即在 U_{all} 中，从一个测试用例 $U_i(1<i<N)$ 到另一个测试用例 $U_j(1<j<N)$ 必须要经历同一被测软件的其他一些测试用例，这两点恰好符合流形的概念，故可以将同一被测软件的测试用例集合 U_{all} 看成是高维空间的一个流形。

从感知的角度看，流形是认知的基础，任何一个高维流形一般存在于一个非线性低维流形上[24-25]，认知过程在很大程度上是通过这种非线性低维流形表示来识别事物的。也就是说，无论流形结构有多么复杂，人们都可以通过少数几个“本质”参数去迅速地认识它，同理，被测软件的测试用例集合也可通过这些“关键”参数来认识其内在规律和性质。

利用流形学习的方法进行软件测试用例集精简，其目标就是从在高维空间里具有“流形”分布的全体测试用例集中选择出“最具代表性”的测试用例集，以达到有效精简测试用例集的目的。对于中、大型软件系统，从输入空间角度看，因测试数据集满足非线性、海量数据集的特点，故如流形学习等基于局部建模的机器学习方法与软件测试领域结合将有着更好、更广泛的发展前景。

4　结　语

随着软件测试工作的重要性越来越高，以及机器学习方法与其结合越来越紧密，期望机器学习解决的测试问题层出不穷。各类机器学习方法在软件测试方面的应用由此派生。这无论对机器学习领域还是软件测试领域来说，都具有十分重要的研究价值。

参考文献：

[1] Myer G J, Sandler C, Badgett T, et al. The art of software testing[M]. 2nd ed. [S.l.]: John Wiley & Sons, 2004: 234-240.

[2] Patton R. Software Testing.[M]. 2nd ed. [S.l.]: SAMS & Education, 2006: 10-14.

[3] Clarke J, Dolado J J, Haman M, et al. Reformulating software engineering as a search problem[J]. IEE Proceedings Software, 2003,150(3): 161-175.

[4] Kuhn D R, Wallace D R, Gallo A M. Software fault interaction and implication for software testing [J]. IEEE Transcations on Software Engineering, 2004,30(6): 418-421.

[5] Wegener J. Overview of evolutionary testing [C]// IEEE Seminal Workshop. Toronto, Canada: IEEE, 2001: 46-54.

[6] Coher D M, Dalal S R, Fredman M, et al. The AETG system: an approach to testing based on combinatoral design [J]. IEEE Transactions on Software Engineering, 1997,23(7): 437-444.

[7] Coher D M, Dalal S R, Parelius J, et al. The combinatorial design approach to automatic test generation[J]. IEEE software, 1996,13(5): 83-88.

[8] Cohen D M, Fredman M L. New techniques for designing qualitatively independent system [J]. Journal of combinatorial design, 1998, 6(6): 411-416.

[9] Tai K C, Lei Y. A test generation strategy for pairwise testing [J]. IEEE Transactions on Software Engineer, 2002,28(1): 109-111.

[10] Poole J. A method to determine a basis set of paths to perform program testing[EB/OL]. [2009-12-28]. http://hissa.nist.gov/publications/nistir5737.

[11] Cohen M B, Gibbons P B, Mugridge W B, et al. Constructing test suites for interaction testing [C]// Processing of the 25th International Conference on Software Engineering. Washington, D C: IEEE Computer Society, 2003: 38-48.

[12] Schroeder P J, Korel B. Black-box test reduction using input-output analysis [J]. ACM SIGSOFT Software Engineering Notes, 2000: 173-177.

[13] Chen T Y, Tes T H, Zhou Zhiquan. Fault-based testing without the need of oracle [J]. Information and Software Technology, 2003, 45(1): 1-9.

[14] Roper M. Computer-aided software testing using genetic algorithms [C]//The 10th international software quality week (QW'97). San Francisco, USA: [s. n], 1997.

[15] Weichselbaum R. Software test automation by means of genetic algorithms [C]//Proceedings of The 6th International Conference on Software Testing, Analysis and Review. Munich, Germany: [s.n.], 1998.

[16] Sthamer H H. The automatic generation of software test data using genetic algorithms [D]. Pontyprid, U K: Thesis University of Glamorgan, 1996.

[17] Tracy N, Clark J, Mander K, et al. An automated framework for structural test-data generation [C]//Proceedings of the 13th IEEE Conference on Automated Software Engineering. Hawaii, USA: [s.n.], 1998: 285-288.

[18] Kaner C, Bach J, Pettichord B. Lessons learned in software testing[J]. Computing Reviews, 2004, 45 (12): 769-770.

[19] 张广梅.软件测试与可靠性评估[D].北京：中科院计算技术研究所，2006.

[20] 杨劲涛，郭荷清.一种精简测试用例方法的研究[J].计算机科学，2005，32(5)：236-239，F004.

[21] 荚 伟，谢军闯，奚红宇，等.遗传算法在软件测试数据生成中的应用[J].北京航天航空大学学报，1998，24(4)：434-437.

[22] 史 亮，徐宝文.测试数据自动生成技术研究[D].南京：东南大学，2006：5-8.

[23] 赵连伟，罗四维，赵艳敞，等.高维数据流形的低维嵌入及嵌入维数研究[J].软件学报，2005，16(8)：1423-1430.

[24] Camastra F. Data dimensionality estimation methods: a survey [J]. Pattern Recognition, 2003, 36 (12): 2945-2954.

[25] Gonzalez J, Rojas J, Pomares H, et al. A new clustering technique for function approximation [J]. IEEE Transactions on Neural Networks, 2002, 13(1): 132-142.

(上接第 150 页)

[7] 万国强，杜栓义，张任奇.基于软件无线电的发信机原理及实现[J].单片机与嵌入式系统应用，2006，(1)：35-38.

[8] 胡廉民，谢先明.软件无线电接收机中下变频技术研究[J].乐山师范学院学报，2008，23(5)：25-27.

[9] 张 雷，邓江平，王建宇.GPS 软件接收机的模块设计与信号处理[J].计算机工程，2009，35(1)：198-200.

[10] 张辰光.软件无线电在智能天线中的应用[J].现代电子技术，2007，30(1)：50-51，60.

[11] 王 雪.浅析软件无线电技术及其在下一代通信系统中的应用[J].天津工程师范学院学报，2006，16(3)：57-60.