AI技術によるソフトウェアテストケースの自動生成

若　者

郭　　秀景*

Automated Generation of Software Test Cases Using AI Technology

Key Words：Software testing, test case generation, AI test

はじめに

ソフトウェアテストは、エラーを見つけるためにプログラムまたはシステムを実行するプロセスで、信頼性の高いソフトウェアシステムを構築するための最も重要なアクティビティの1つである。ソフトウェアは私たちの生活や社会のあらゆるところで利用されるようになっている。見つかっていない欠陥は、ソフトウェア自体とそのアプリケーションシステムに深刻な情報セキュリティの脅威、または大きな経済的損失をもたらす可能性がある。例えば、2015年4月にスターバックスコーヒーはPOSシステムのソフトウェアシステムの不具合で取引ができなくなり、アメリカとカナダの約60%の店舗を一時的に閉鎖させられた。また、2016年2月に打ち上げられたX線天文衛星「ひとみ」は、プログラムの不備や入力ミスなどが重なって、早々に運用断念に追い込まれた。このようなソフトウェア事故は後を絶たない。したがって、ソフトウェアのセキュリティ事故を回避するために、ソフトウェア開発のプロセスにおいてソフトウェアの信頼性を重視することが必要である。

私は、大学に入学した後、ソフトウェア開発の勉強を始めたばかりの頃は、まだソフトウェアテストの重要性に気が付いていなかった。プログラムがほぼ完成し、実行してみると多くのバグがあることに気が付いた。一つのバグを修正するともっと多くのバグが出てきた場合もある。そこで初めてソフトウェアテストの重要性に気づき、本を通じてソフトウェアテストに関する知識を身につけた。また、ソフトウェアプロジェクト管理の研究を通じて、ソフトウェア開発にとって品質保証は非常に重要であることがわかった。

さらに、ソフトウェアの品質保証について学ぶことは、自分の興味を深めるだけでなく、情報技術分野の発展にとっても大きな意味があると感じている。そこで、大学卒業後来日し、広島大学、および、大阪大学で、AIベースのソフトウェアテストの研究を行っている。本稿では、現在研究している境界値テストケースの自動生成に関する研究[1]について紹介させていただく。

* Xiujing Guo
1994年6月生まれ
広島大学　大学院先進理工系科学研究科
情報科学プログラム博士後期課程修了
（2024年）
現在、大阪大学　大学院情報科学研究科
情報システム工学専攻　助教
博士（情報科学）
TEL：06-6879-4547
E-mail：guoxiujing@ist.osaka-u.ac.jp

機械学習を用いた境界値分析

テストカバレッジは、テストケース設計において、テストケース集合がソフトウェアの実行パスのどれだけをカバーしているかを測定する基準として重要な役割を果たす。例えば、図1のように、分岐カバレッジは、プログラム内の全ての可能な分岐のうち、テストケースによって実行された決定分岐の割合として定義される。分岐カバレッジ100%を達成することにより、if-else文、ループ、switch文などの制御構造から生じる各経路が少なくとも一度はテストされることが保証される。分岐カバレッジはプ

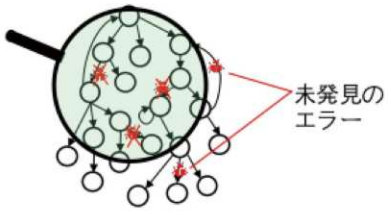

図1：分岐カバレッジ

ログラムの論理をより包括的に評価し、文カバレッジのような単純なカバレッジ基準では十分にテストされない可能性のある領域を特定するのに役立つ。

しかし、完全な分岐カバレッジを達成しても、すべてのエラー、特に境界に関連する問題が検出される保証はない。例えば、図2に示すような、値が有効かどうかをチェックする関数を考えた場合、テストケースが「value = -5」や「value = 5」といった値をカバーしていれば、if 文の「true」と「false」の両方の分岐をテストすることで分岐カバレッジは満たされる。しかし、もし「>=」ではなく「>」といった境界エラーが存在する場合、「value = 0」が誤って「false」を返す可能性がある。このエラーは、正確な境界値「value = 0」をテストしなければ検出されない。

```
bool is_valid(int value){
    if (value >= 0) {
        return true;
    } else {
        return false;
    }
}
```

図2：値を検証する関数

筆者らは、ソフトウェアの信頼性向上を目的とし、境界値分析と機械学習を融合した境界値テストケース生成手法を検討している。境界値分析は、入力空間を複数の部分領域に分割し、その分割同士の「境界」においてバグが生じやすいという経験則に基づき [2]、境界付近の入力を重点的に選ぶことでテスト効率を高める方法として知られている。しかし、境界値分析を手作業で行う場合、複雑なソフトウェアでは膨大な仕様書やソースコードの解析が必要になり、入力変数同士の依存関係を正確に把握したり、実際に境界値を見つけたりするのが難しいという問題がある [3]。これを克服するために、筆者らはプログラムの実行経路の違いに着目した機械学習モデルを用いて、自動的に境界を推定しテストケースを生成する手法を提案している。

本手法は、大きく二つのステップに分かれる。まず、テスト入力二つをペアとしたとき、それらの実行経路が同一であるか否かを判断するための機械学習ベースの識別モデルを訓練する。次に、この識別モデルの出力を基に、マルコフ連鎖モンテカルロ（MCMC）[4] を用いたテスト入力の生成を行う。

具体的にはまず、ある入力 x と x にごく小さな摂動を加えた x+h が同じ実行経路を通るかどうかを判定する識別器を機械学習によって訓練する。ここで「実行経路が同じ」とは、プログラム内の分岐条件が同じ通り方をしたかどうかを指す。もし両者の経路が異なれば、x は境界付近にある可能性が高いと考えられる。実際の学習手順としては、プログラムを多数の入力値で実行し、各分岐が実行・非実行・条件成立のいずれに該当したかを収集して「経路情報」を得る。そして、得られた2つの入力対 (x, x') に対して実行経路が同一ならラベル 0、異なるならラベル 1 を振り、これを大量に集めてニューラルネットワークなどのモデルを訓練することで、「2つの入力が同じ経路を通るか否か」を予測する識別器を構築する。このとき Gcov などのカバレッジ解析ツールを用いると、分岐ごとの実行回数やテスト状況を比較的容易に取得できるため、大規模な訓練データを自動的に用意できる。

次に、識別器を活用してテスト入力を生成する段階では、MCMC を用いる。これは確率分布からのサンプリングを効率よく行う手法の一つであり、提案手法では「境界付近ほどサンプリングされやすい」ような分布を設定している。すなわち、ある入力 x の近傍を少し動かしたときに「経路が変わる」確率が高いほど、その入力は境界に近いとみなし、そのような点を高い確率でテスト入力として採択するようにマルコフ連鎖を設計する。具体的には、現在のサンプル x から一様分布などで次の候補 x' を選び、その時点で識別器から得られる「x' と x'+h が異なる経路を通る可能性」を評価値として、従来の x に比べてその値が高ければ高確率で x' を採択し、低ければ採択しない。図3は、2次元空間における本研究のアプローチを示している。この図では、2つの入力 a と b および1つの境界が存在する。円は摂動の半径 R を表している。x' は x よりも境界に近いため、x' と x'+h が異なる等価クラスに属する確率は、x の場合よりも高い。こうして境界近傍の入力が徐々に採択されやすくなり、最終的に出力される入力群（テストケース集合）が境界値を広くカバーすると期待できる。

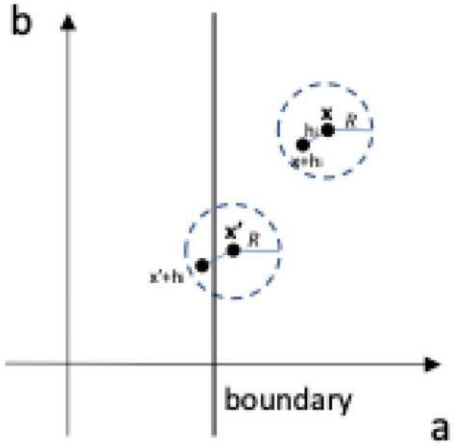

図3：2次元空間における例

この手法の利点としては、ソースコードや仕様の複雑な解析を必要とせず、分岐の情報だけを学習して境界を推定できる点が挙げられる。一方で、このアプローチには注意点もある。特に識別器の精度が十分でなければ、誤って境界でない箇所を境界とみなす、もしくは真の境界付近を見落とすリスクがある。また学習データを得るために多数の入力値でプログラムを実行し、分岐状況を記録する必要があるため、事前の準備に時間がかかる場合もある。加えて、境界を特徴付ける分岐の組み合わせが非常に多い巨大ソフトウェアを扱うには、さらに効率的に学習データを生成する工夫や、学習の最適化が求められる。たとえば、ランダムにデータを収集するのではなく、被覆率の高い経路を優先的に走らせるようなテスト戦略と組み合わせれば、少ないサンプル数でも効果的に境界情報を学習できる可能性がある。筆者らは今後、そうした手法の改良を通じて、より大規模かつ複雑なプログラムに対しても境界値近傍のテストケースを自動で高精度に生成できる仕組みを目指している。

おわりに

本稿では、ソフトウェアの境界値分析と機械学習を融合したテストケース自動生成手法について述べた。提案手法は、分岐情報を活用して境界近傍を推定する識別器を学習し、その識別結果に基づいてMCMCを用いてテスト入力を効率的にサンプリングする点に特徴がある。これにより、従来のランダムテストや手動解析に比べ、短時間で境界値付近のカバレッジを高められることが期待できる。一方で、大規模ソフトウェアへの適用を見据えた際の識別器の精度向上や学習データ収集の効率化など、今後解決すべき課題も依然として多い。今後は、高いカバレッジを達成するテスト戦略との連携や、モデルの汎化性能を強化する学習手法の検討などを進め、境界値テストのさらなる自動化と高品質化を実現していきたいと考えている。

参考文献

1) Guo, X, Okamura, H. , Dohi, T, Optimal test case generation for boundary value analysis, Software Quality Journal, February 2024. DOI 10.1007/s11219-023-09659-9.
2) Reid, S C, An empirical analysis of equivalence partitioning, boundary value analysis and random testing, Proceedings Fourth International Software Metrics Symposium. IEEE, 1997: 64-73.
3) Dobslaw F, de Oliveira Neto F G, Feldt R., Boundary Value Exploration for Software Analysis, 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2020: 346-353.
4) Brooks S., Markov chain Monte Carlo method and its application, Journal of the royal statistical society: series D (the Statistician), 1998, 47(1): 69-100.