B-003

確率的プログラムシステムのバグ検出ツールの開発

ヴォダイチン　　于海波
九州産業大学情報科学研究科

1. はじめに

確率的プログラミングは、確率的モデルが指定され、これらのモデルに基づく推論が自動的に実行されるプログラミングパラダイムである。近年、 AI の発展に伴い、確率的プログラミングは特に分析にベイジアン推論が必要な機械学習などの分野で非常に重要になってきた。しかし、現在、確率的プログラムの開発には、主に従来の伝統的プログラムの開発支援環境によって提供されている技法とツールを流用しており、確率的プログラムの特徴を考慮していない。本研究では、確率的プログラムの特徴を考慮した確率的プログラムの不具合を効率よく検出できるシステムの開発を目的としている。具体的には、まず先行研究のバグ分析より PyMC3 の開発環境から抽出された確率的プログラミング関連のバグパターンを自動的に検出する手法を提案し、提案された手法に基づいたバグ検出器を開発した。その後、今回開発されたバグ検出器を組み込んだ確率的プログラムのバグ検出 Web システムを構築した。本 Web システムでは、拡張機能を重視し、今後も新しいバグパターンを検出できるバグ検出器を組み込むことができるように設計した。

第 2 章では関連研究を紹介し、第 3 章では静的解析の AST 分析手法を利用したバグ検出手法及び関連バグパターン検出器の実現について述べる。第 4 章では確率的プログラムのバグ検出 Web システムの構築について述べる。最後に、第 5 章で本研究をまとめる。

2. 関連研究

伝統的なプログラムのバグ検出について、今まで多くの研究が行われ、バグ検出ツールも数多く開発された。本研究での確率的プログラムのバグ検出器の開発のため、伝統的なバグ検出ツールについて調査し、それらの実現方法を参考したい。バグ検出では、静的解析と動的解析による検出手法がある。静的解析はコードの構文や構造を静的に分析し、バグや問題を検出する。一方、動的解析はプログラムを実行しながらバグや問題を検出する。静的解析の利点は、早期のバグ検出、網羅的な検出、パフォーマンス改善が可能であることである。静的解析バグ検出ツールは FindBugs[1]、PMD、ESLint、SonarQube などが存在している。そのうち、FindBugs は、Java プログラムの静的解析を行い、豊富なルールセットを提供し、多くのバグパターンや問題を網羅的に検出できる。カスタマイズ性も高く、統合開発環境（IDE）との連携も可能であるため、本研究では、主に FindBugs を参考にした。

文献[2]では、確率的プログラミング言語のプログラミングシステム（PP システム）のバグ分析とテストに焦点を当てている。Edward、Pyro、Stan の 3 つのオープンソースの PP システムにおいて、過去に報告された 118 のバグを特徴付け、PP システムをテストするための拡張可能なシステムである ProbFuzz を提案した。この研究では、PP システムのバグを Algorithmic/accuracy bugs、Dimension/boundary-value bugs、General numerical bugs、Language/translation bugs の 4 つのカテゴリに分類した。

文献[3]では PyMC3 確率的プログラミングシステムから報告された 271 個のバグを調査し、20 個の確率的プログラム特有なバグを絞り込んだ。これらのバグを文献[2]のバグ分類手法に基づき分類した。また、これらのバグから、8 個のバグパターンを抽出した。本研究では、文献[3]で抽出された 8 個の確率的プログラム特有なバグパターンのうち 3 個のバグパターン検出手法を提案し、関連バグパターンの検出器を実現した。

3. 確率的プログラムバグパターン検出器の開発

本研究では、文献[3]で抽出された 8 個の確率的プログラム特有なバグパターンのうち 3 個のバグパターンに対する検出手法を提案し、関連バグパターンの検出器を実現した。基本的な手法としては、Python のソースコードを AST(抽象構文木)に変換し[4]、すべてのノードを走査して各バグパターンの条件を満たすかどうかを確認する。条件を満たす場合はバグとして報告し、次のバグパターンに対しても検査を行う。以下では、そのうちの 1 つのバグ検出器の実現例を挙げ、実現方法に関して述べる。

例として、図 1 で示す「log 計算時に 0 を代入するバグ」をあげる。このバグでは、PyMC3 の Exponentially modified Gaussian distribution を使用する際に、予期せぬバグが発生することが報告されている。具体的には、log 関数内の計算で 0 が代入される可能性があり、それによって結果が-inf となるケースが発生していた。

log 計算時に 0 を代入するバグ.py

```
lp = tt.switch(tt.gt(nu, 0.05 * sigma),
               -tt.log(nu) + (mu - value) / nu + 0.5 *
(sigma / nu)**2
               + logpow(std cdf((value - mu) / sigma -
sigma / nu), 1.),
               -tt.log(sigma * tt.sqrt(2 * np.pi))
               - 0.5 * ((value - mu) / sigma)**2)
```

図 1 log 計算時に 0 を代入するバグ

図 2 ではこのバグパターンを検出するアルゴリズムを示している。Python のソースコードを AST に変換し、与えられた AST ノードが log 関数または logpow 関数を呼び出しであるかどうかを判定し、引数に 0 が含まれているかどうかをチェックする関数である。もし 0 が含まれていれば、警告メッセージを表示する。さらに、ノードが logpow 関数の呼び出しである場合、引数の中に std_cdf 関数の呼び出しがあるかどうかもチェックする。もし引数に std_cdf 関数の呼び出しがある場合、特定の式（std_cdf((value-mu)/sigma-sigma/nu)）を使用している場合には、修正の提案を行う。

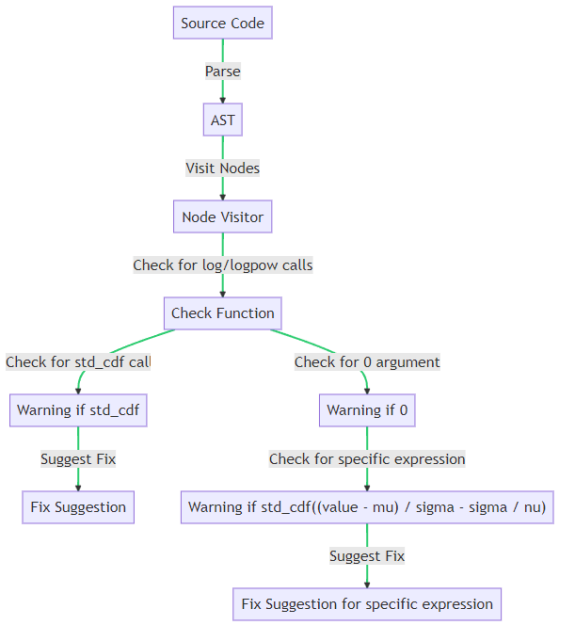

図 2 log 計算時に 0 を代入するバグ検出アルゴリズム

4. バグ検出 Web システムの構築

第 3 章では一部の確率的プログラムのバグパターンの検出器の開発について述べたが、これらの検出器をより多くの人が分かりやすく使えるため、または今後より多くのバグパターンの検出器を組み込まれるため、本研究では、確率的プログラムのバグ検出 Web システムを開発した。本システムの設計において、ユーザインターフェースの設計及び拡張機能を重視した。

本 Web システムでは、ユーザーがエディタにソースコードを貼り付けるかファイルのアップロードをすることよりソースコードを取り込む。取り込んだソースコードをサーバーに送信し、サポートされているバグパターンに対して解析を行い、潜在的なバグに対する警告を表示する。また、検出できるバグパターンのリストが表示され、デフォルトとして全てのバグパターンに対して検出するが、ユーザーがその中の一部のバグパターンのみ選択する場合、選択されたバグパターンのみに対して検出作業を実施する。

バグ検出システムの主な機能を以下にまとめる。

(1) バグパターンの選択: 必要に応じて、検出させたいバグパターンをユーザー自身が選択できる。
(2) ソースコードの切り替え: プログラムはテキストを貼り付けるか、ファイルからのアップロードか、異なる形のソースコードの受け取り方を簡単に切り替えられる。
(3) コードのインポート: コードをドラッグ＆ドロップまたは貼り付けでインポートすることができる。
(4) 解析の結果表示: ソースコードを解析し、バグの検出結果を表示する。
(5) バグの詳細表示: バグアイコンにマウスを合わせると、詳細情報が表示される。

本 Web システムの開発において、データの処理や API の提供など、バックエンドの部分は Python Flask を使用し、フロントエンドでは、より迅速なインターフェース開発ができるために React を利用した。

システムの構造は図 3 で示している。まず、フロントエンド側では、ユーザーが解析したいソースコードを受け取り、(SourceCodeHandler.py を介して)サーバーに送信する。次に、バックエンド側では、(BugFinder クラス bug_finder.py) ソースコードを AST 形式に変換し、それぞれのバグパターンに対応する検出器を実行し、ソースコードを解析する。すべての結果は(ResultArray 配列に)保存され、呼び出された API を介してフロントエンドにバグ検出結果を送信し、表示する。

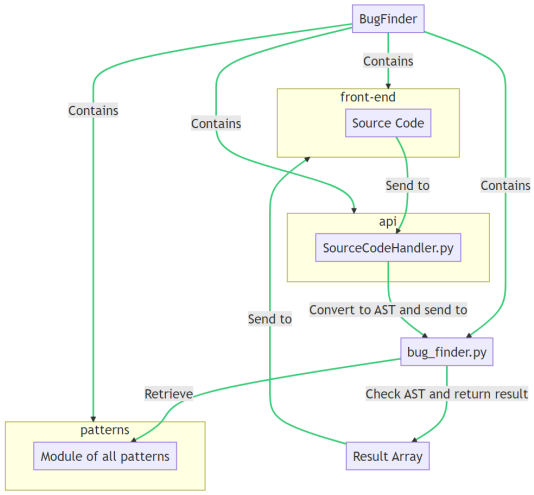

図 3 バグ検出 Web システムの構造

5. まとめ

本研究では、確率的プログラムの特徴を考慮した確率的プログラムのバグ検出 Web システムを開発した。また、PyMC3 確率的プログラミングシステムから抽出された確率的プログラム特有のバグパターンのうち 3 つのバグパターンの検出器を開発し、バグ検出 Web システムに組み込んだ。開発された Web システムは、ユーザーのソースコードのバグの検出を行い、検出されたバグは可視化され、詳細情報が表示される。これらの情報を活用してユーザーのプログラムの品質向上に貢献することができると考えている。今後、より多くの確率的プログラム特有なバグパターンを検出するバグ検出器を開発し、開発された確率的プログラムのバグ検出 Web システムに組み込みたい。また、本研究室のみならず、世界中のほかの研究者が開発された確率的プログラムのバグ検出器でも組み込めるため、Github 上に本 Web システムを公開し、検出器を組み込むためのインターフェースの定義を行う予定。

参考文献

[1] Findbugs: https://findbugs.sourceforge.net/
[2] Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic, Testing Probabilistic Programming Systems, In Proceedings of ESEC/FSE'18, 2018, pp. 574-586.
[3] Shoma Hamada, Haibo Yu, Vo Dai Trinh, Yuri Nishimura, Jianjun Zhao, Bug Patterns in Probabilistic Programming Systems, in Proceedings of International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2022, pp. 384-391.
[4] Python AST: https://docs.python.org/3/library/ast.html