**確率的プログラムにおけるバグ分析**

西村　優里　　于 海波

九州産業大学 情報科学研究科

# **はじめに**

　確率的プログラミング(PP)を用いることで、統計的モデルの取り扱い、機械学習などの研究開発がより円滑にできるため、近年AIの発展により、非常に重要になってきた。しかし、現在は既存の開発環境を利用しており、確率的プログラムの特徴に対応していないという問題がある。

　本研究では、GitHubから確率的プログラミング言語Pyroで開発されたシステムの実際のバグを調査分析することより、確率的プログラミング言語独特なバグパターンを抽出し、そのバグパターンを自動検出する手法の提案を行うことを目的としている。これにより確率的プログラムのバグの特徴をつかみ、本研究室で行われているバグ検出ツールの開発に貢献することが期待できる。本論文では、確率的プログラムのバグ分析の方法及び現時点での分析結果について述べる。

# **バグ分析方法**

## **調査対象確率的プログラミング言語**

　PPシステムでよく利用されている確率的プログラミング言語にはStan、Pyro、PyMC3などがあり、本研究では確率モデリングが容易に可能で、

ガウス過程やVAEなどが利用でき、モンテカルロ法、変分推論など多くの手法が実装されているPyroを調査対象とした。PyroはバックエンドとしてPyTorchによってサポートされているPythonで記述されたプログラミング言語である。

## **調査対象バグデータ源**

　GitHub上数多くのプロジェクトが存在しており、関連プロジェクトの開発と使用上の問題があった場合、Issuesとして登録し、問題解決まで管理されている。GitHub上登録されているIssuesの中に実際のbugが多数報告されている。そのため、本研究では、GitHub上登録されている実際のバグを調査対象とする。そのうち、Pyro、PyTorchが利用され、Python言語で開発されたプロジェクトのバグのうち、既にclosedされ、修正プログラムまで登録されたバグのみを最終的な分析対象とした。

## **バグ分類方法**

バグの分類は、Saikat Dutta氏らの論文[1]を参考に4つにバグを分類し、それらに当てはまらないバグ、確率的プログラミン特有のバグではない調査対象外のバグをその他と記録を残した。4つの分類[1][2]は以下のとおりである。

1. Algorithmic/accuracy bugs

推論アルゴリズムや確率分布、統計的手順の誤った実装に起因するバグで、誤った推論結果を示すものの、一見正しい数値となるために発見が難しい。

1. Dimension/boundary-value bugs

関数が入力データの次元(スカラー、ベクトル、行列など)、入力データの長さ、分布関数のパラメータの長さなどを適切に扱えないことに起因するバグで、これらはNaNやInfとして出力される。

1. General numerical bugs

通常の数学計算によって生じるバグで、これらはNaNやInf、誤った計算値を出力する。

1. Language/translation bugs

PPシステムに使用されているプログラミング言語の特徴を誤った使い方をして生じるバグで、これらはビルドの失敗やランタイムエラー、誤った出力結果としてあらわれる。

## **バグ分析方法**

　2.2節の条件で分析対象として絞り込んだリポジトリ内のIssuesを確認し、bugのタグが付いているが実際にはバグではない報告、修正プログラムがないなど情報が不完全なバグを除外した上で、バグの記述内容や修正プログラムなどを調査し、バグ内容によって、2.3節の分類カテゴリで分類する。本研究は、それらを手動で行った。

# **調査・分析結果**

当調査では、Pyroに関連したリポジトリ内の Issuesの中で、既に閉じたバグ(state:closed、bugのラベルがついている)総数155個のバグ報告を調査した。その結果、52個のPyroに関連するバグが見つかった。これらのバグを2.3節で示した分類ごとにカテゴライズした。カテゴライズできないバグのほとんどはドキュメントの問題やインポートエラー、互換性の問題などである。バグ分類ごとのバグ数を表3-1にまとめた。

表 3‑1　バグ分析結果

|  |  |
| --- | --- |
| バグ分類 | バグ数 |
| Algorithmic/accuracy bugs | 11 |
| Dimension/boundary-value bugs | 20 |
| General numerical bugs | 8 |
| Language/translation bugs | 13 |
| 総 計 | 52 |

　調査した確率的プログラミングに関連するバグの例について以下に説明する。

1. cholesky->lanczos閾値を超えると、特異行列エラーが発生するバグ

このバグ[3]は、特異行列に関連するバグで、x値間の共分散行列が800を超える観測値で、get\_fantasy\_model関数を試みると、発生する。つまり、cholesky分解によって共分散行列が特異になるため、lanczos法を用いた際に特徴値や特徴ベクトルを求めることができなくなるバグである。これは、特異な行列の逆行列が存在しないため、それらを使用する計算は失敗する。

|  |
| --- |
| n\_dims = 1  base\_n = 801  x = torch.ones((base\_n, n\_dims))  model = ExactGpModel(x, y, likelihood=likelihood, covar\_module=MaternKernel())  new\_x = torch.randn((77, n\_dims))  new\_y = torch.randn(77)  model.get\_fantasy\_model(new\_x, new\_y) |

上記はエラー発生時のコードである。観測値の数を減らすか、covar\_module=MaternKernel()の箇所にMaternKernelの部分を異なる共分散関数に変更することで問題を解消できる。

このバグをカテゴライズすると、General numerical bugsに分類される。

1. Mllで計算されたInf値

このバグ[4]は、損失がinfになり、逆伝播の後に共分散がnan値になることによって引き起こされる。これは、確率的プログラミングで行われる最適化に関するバグである。

|  |
| --- |
| def fit(self):  self.model.train()  self.likelihood.train()  optimizer = torch.optim.Adam([{'params': self.model.parameters()}, ], lr=self.learning\_rate)  mll = gpytorch.mlls.VariationalELBO(self.likelihood, self.model, num\_data=y.size(0))  for restart in range(0 + 1):  for i in range(self.training\_iters):  optimizer.zero\_grad()  output = self.model(self.X)  loss = -mll(output, self.y)  loss.backward()  optimizer.step() |

上記のコードのoutput = self.model(self.X)、loss = -mll(output, self.y)の部分で、mllの計算中にlossがInfになっていることが問題である。そのため、outputの値やself.yの値が正しいかを確認し、損失がinfから非infの値に変化することで、原因を解消できる。

　このバグをカテゴライズすると、 Dimension/boundary-value bugsに分類される。

# **まとめ**

　本研究では、確率的プログラミング言語Pyroで開発されたシステムのバグを調査分析し、バグの分類を行った。今後の課題は、さらにより多くのバグの収集・分類を行い、まとめたバグからバグパターンを抽出することである。また、今回調査対象としたPyro以外の確率的プログラミング言語に対しても同様の調査を行い、バグパターンを自動検出する手法の提案に結びつけることである。

**参考文献**

1. Saikat Dutta, Owolabi Legunsen, Zixin Huang, and Sasa Misailovic. Testing proba-bilistic programming systems. In Proceedings of the 2018 26th ACM Joint Meetingon European Software Engineering Conference and Symposium on the Foundationsof Software Engineering, pp. 574–586, 2018
2. 濱田翔馬,確率的プログラムにおけるバグパターンに関する調査研究,修士論文, 九州大学,2021
3. Singular matrix error when crossing cholesky->lanczos threshold・issues#1602・https://github.com/cornellius-gp/gpytorch/issues/1602
4. Inf value calculated by mll・issues#1579・https://github.com/cornellius-gp/gpytorch/issues/1579.