2211autoCorrect
2022.11　ブログ：『**ニューラルネットワークモデルのバグ限局・自動修正技術**』を読んで、
の詳細　（→ http://www.1968start.com/M/blog/index2.html#2211　）

『ニューラルネットワークモデルのバグ限局・自動修正技術』を読んで

中所武司

■この本の読書のきっかけ

情報処理学会誌の最新号に掲載されていた下記の解説に興味を持った。
特集：AI の品質保証：「**5．ニューラルネットワークモデルのバグ限局・自動修正技術**」
情報処理, 63(11), e28-e33 (2022-10-15)

■記事内容の要約とコメント（→★）

【機械学習を含むソフトウェアの品質保証】

従来のソフトウェア開発との違い

・自動運転や医療画像解析，音声アシスタントなどのソフトウェアに、機械学習モデルが組み込まれている．特に，層を深く重ねた**ディープニューラルネットワーク**（DNN）はその性能の高さからよく用いられている．

・機械学習モデルの開発は，モデルがデータからルールを学習する**帰納的開発**であり，
従来のプログラム開発は、人があらかじめルールを決める**演繹的開発**である．
この違いに伴い，新たな品質保証上の問題が生じている．

・機械学習ソフトの品質基準や達成目標を定めた機械学習品質マネジメントガイドラインは，
最終的な利用者に提供すべき品質（利用時品質）として，
「**安全性，有用性，公平性，プライバシー**」の 4 つを挙げているが、
帰納的に決定された振舞いの品質について、利用者の求める基準を保証するのは難しい．

品質保証の難しさ（自動運転の例）

・品質保証が難しい自動運転のソフトウェアでは，利用時品質の中でも**安全性**が特に重要．

・機械学習品質マネジメントガイドラインでは，**安全性**を次のように定義している：
「機械学習要素の望ましくない判断動作によって，その製品の運用者・利用者または第三者などに人的被害や経済損失・機会損失などの悪影響を及ぼすことを回避する品質特性」

・この**安全性**に関する事故が国内外で報告されている．

2020 年に自動運転モードのテスラ車が，横転中の白いトラックに激突した事故では，
白いトラックという障害物を正しく認識できていなかった可能性が指摘されている．
モデルが学習データで経験していないデータにうまく対応できないリスクがあるが、
想定されるデータをすべて網羅してモデルを学習させることは非現実的である．

→★自動運転ルール作成の難しさについては、次のブログで言及している：
2015.12 「自動車の自動運転（右折はまだ苦手）に一言」
http://www.1968start.com/M/blog/old.html#1512

品質保証に向けた研究

・機械学習ソフトウェアの品質保証や品質向上のため，モデルのバグを自動的に特定し，
修正する研究が行われている．バグとは誤動作（好ましくない振舞い）の原因を指し，
バグを自動的に特定することを**バグ限局**と呼ぶ．

・本稿では，ニューラルネットワークのモデル自体に対するバグ限局・修正の研究を紹介．

【ニューラルネットワークのバグ限局・修正の研究動向】

・これらの研究は，従来のソフトウェアに対するテストやバグ限局・修正手法を
DNN に対して応用することで発展してきた．年代順に紹介する：
＊DeepXplore（2017）：従来のソフトウェアテストの手法を DNN に適用
＊Deep-Fault（2019）：ニューロンに対してプログラムのバグ限局手法を適用
＊PAFL（2022）　：筆者らの研究で，ニューラルネットワークの一種である
リカレントニューラルネットワーク（RNN）に特化したバグ限局手法
＊Arachne（2019）：プログラムの自動バグ修正を応用して DNN の自動修正を行う手法
＊CARE（2022）　：因果分析を利用してバグ限局を行い，
モデルの満たすべき性質を満たすように自動修正を行う手法

DeepXplore（2017）

・ソフトウェアテスト手法のホワイトボックステストは，
一連のテスト入力がプログラムの振舞いを網羅することが望ましいという観点から、
命令や条件分岐の何割をテストで実行できたか（命令網羅率，分岐網羅率），などの
テストカバレッジ指標を用いる。

・本方式では，DNN に対するホワイトボックステストのテストカバレッジとして，
ニューロンカバレッジという指標を考案した．
DNN を構成する全ニューロンの何割が発火したかを表す指標である．

→★この論文については、次のブログでコメントしている。概念は似ているが、
網羅率向上のための手法が異なる。従来のソフトウェアテストの分野では、
未実行の命令や分岐を網羅するために新たなテストを追加して、品質向上を図るが、
本論文は、未発火のニューロンを特定して発火させるテストを追加していない。
＜関連ブログ＞
2019.1　「ＡＩシステム検証へのニューロンカバレッジの有用性について」
http://www.1968start.com/M/blog/index.html#1901

・さらに，ニューロンカバレッジを拡張し，テスト入力が各ニューロンの出力値を
広くカバーできたかを測定するカバレッジを提案している．

→★この拡張の効果は小さいのでは？
出力値を「広くカバー」の意味が明確でないが、網羅率という意味なら、
機械学習の場合、出力直前の内部ニューロンの網羅率と相関があるはず。

DeepFault（2019）

・DNN に対してバグ限局の手法（Spectrum-Based Fault Localization：SBFL）
を適用して，誤動作の原因となるニューロン（欠陥ニューロン）を特定する．

・SBFL では，プログラムの各行に対し、
「失敗テスト（望ましい出力と異なる出力をしたテスト入力）で実行された行は
バグである可能性が高い」という考えで、バグの度合いを表す疑惑値を計算する．

・DeepFault は「予測失敗時に発火したニューロンはバグの可能性が高い」という考えで，
各ニューロンの疑惑値を計算し，疑惑値の高いニューロンを欠陥ニューロンと特定する．

・また，特定した欠陥ニューロンを発火させるような新たなデータを生成し、
再学習することで，欠陥ニューロンの疑惑値を下げる手法も提案している．

→★この説明は分かりにくいが、再学習とは、予測失敗時に発火したニューロンに対して、
予測成功時に発火するような新たなデータを用いて学習させて、
欠陥ニューロンへ入力するニューロンとの結合係数を大きくすることにより、
予測失敗時に発火したときのニューロンに入力するニューロンとの結合係数が
相対的に小さくなり、予測失敗しないようにすることと思われる。
しかし、欠陥ニューロンに対して、ピンポイントで新たなデータを生成する方法が不明。

PAFL（2022）

・PAFL（Probabilistic Automaton-based Fault Localization）は，

本稿の筆者らの，自然言語処理などによく用いられる RNN に特化したバグ限局手法．
RNN は再帰的な構造を持ち，入力によって変化する内部状態を持つという特徴がある．

・RNN の振舞いを表現する重み付き確率オートマトンという状態遷移モデルを抽出し，
その上で従来のプログラムに対するバグ限局手法を適用する．以下はその手順：
⑴ RNN という複雑なモデルから解釈が容易な重み付き確率オートマトンを抽出
⑵ このオートマトン上で SBFL を適用し，RNN のバグとなるオートマトン上の状態の
n-gram（n 個の状態の列）を特定する．バグ限局の対象を複数の状態とすることで，
RNN の動的に変化する内部状態の振舞いを捉えることができる．
⑶ 特定した n-gram を利用して，RNN のバグを引き起こすデータを選択できる．
データごとに RNN の誤動作をどれくらい引き起こしそうかという怪しさを計算し，
その値の高いデータを選択する．
⑷ 選択したデータは，運用開始前の学習済み RNN の検証に利用することが期待できる．

・PAFL で重要な，抽出したモデルの状態の n-gram を特定するというアイディアは，
重み付き確率オートマトンに限らず、一般の状態遷移モデルに適用可能である．

Arachne（2019）

・先に紹介した 2 つの研究は，追加のデータを用意し，再学習を行うので、
時間的・計算資源的にコストがかかるが，これは DNN の重みを直接自動修正する．

・バグ限局とパッチ生成の 2 つのフェーズからなる．
⑴ バグ限局フェーズで，修正すべき重みを特定する．
⑵ パッチ生成フェーズで，特定した重みの修正後の値の候補（パッチ）を複数生成し，
目的関数の値が大きくなるようなパッチを適用する．
目的関数は，既存の誤分類を修正すると同時に，新たな誤分類を生まないような
パッチが高い値になるように設計される．

→★複数のパッチを用意し、その中から目的関数の値が大きいものを選ぶ方法では、
既存の誤分類の修正の確認テストは簡単だが、新たな誤分類を生まない確認テストは、
すべてのテストの再実行が必要なので、再学習に比べてどの程度のコスト削減か不明。
「特定した重みの修正後の値の候補」の決定方法が不明。少しずつ小さくする？

・再学習ではモデル全体の重みが修正されるが、Arachne では，修正対象となる重みは
バグ限局のフェーズで限定されるので，計算コストを小さくできる．

→★上記コメントで述べたように、最適なパッチを見つけるために、すべてのテストを
繰り返す作業は、大変に思える。誤分類のケースが複数あったらさらに大変！

・全体的に精度を上げたい場合は，データを追加して再学習をする方が適切であるが、
データの追加が難しい場合や全体的な精度は高いが特定の誤分類を修正したい場合は
Arachne の方法が適している．

→★「全体的な精度は高い」状態を保持しながら、「特定の誤分類を修正」するような
一般的方法はないので、ひたすら試行錯誤を繰り返すことになりそう。
特定の重みにパッチをあてる方法は、いわば、DNN の膨大な数のニューロン間の重みに
その値を調整するダイヤルをつけて値を微調整する方法と思われ、実用性が疑わしい。

→★従来のプログラミングによる開発においても、稼働後の不良の修正用のパッチをあてて、
当座の不良が解決したように見えて、他の不良を作りこむことがあるので、
修正用のパッチの正しさは、すべてのテストを再実行して確認する必要がある。

CARE（2022）

・これもバグ限局とパッチ生成のステップからなるが、以下の点が Arachne と異なる：
（1）DNN の各ニューロンと振舞いの因果関係の分析により，誤動作の原因の度合いを
計算し、その値の高いニューロンを欠陥ニューロンとして特定する．

（2）モデルの公平性や敵対的攻撃への耐性（ロバスト性）という評価基準により，
誤動作の原因となるニューロンは異なるので、それぞれに関して修正を行える．

【現状の課題と今後】

・ニューラルネットワークモデルに対するバグ限局や修正のために，プログラムに対する
テストや自動バグ修正の技術を適用してきたが，これらの手法は研究段階であり，
実際のモデルの開発現場で利用するためには 2 つの課題がある．

(1) ニューラルネットワークモデルへの修正手法適用で新たな問題が生じる可能性あり．
特定のクラスの誤分類の修正により、全体的な精度が犠牲になることがある．
また，ある観点（予測精度など）での修正が，別の観点（公平性など）での劣化を
引き起こす可能性もある．

→★これまでの関連部分のコメントで指摘済み

(2) 一般に，機械学習モデルの開発には，モデルの学習以外に、データの収集や前処理，
推論，後処理の段階があり、これらをつなげた機械学習パイプラインによる
一連の処理の自動化がモデル開発の効率化につながるが，モデルのバグ限局・修正手法
の研究では，機械学習パイプラインにどう組み込むべきか明確でいない．

以上