ソフトウェアコードレビューの AI 技術を応用した自動化の試み

A Case Study of Fault Prone Module Detection with AI

テクマトリックス（株）　R&D 室
TechMatrix Cooperation R&D
○　西田　啓一 1)
○　Keiichi Nishida

Abstract

The bug detection has been extensively studied actively in order to make the review and test more efficient. An anticipated model using complexity metrics, amount of change of Sorse code, number of past bugs, etc., has been constructed. In this paper, we have constructed a prediction model using machine learning from metrics and bug information, and report the results. At the same time, we will construct and report on suggestions for improvement of modules judged as having bugs.

1.　はじめに

本稿では、バグが含まれる可能性が高いモジュール（fault prone module）を機械学習で予想可能か述べる。バグ予想は、テストやレビューの人的リソースを効率よく集中させることができると期待され、多くの研究が行われている。バグの有無や密度を予想するモデルを構築するため、複雑度メトリクス、ソースコードの変更量、過去バグの数、等々が計測されてきた。しかし、これらのモデルが実際の開発現場に適用された事例は少ない。

すべてのバグの原因や対処方法が整理されていない現在、予想するバグの対象をすべてのバグまで広げることは得策ではない。やはり、特定の領域のバグに絞って予想モデルを構築することが現実的である。

特定の領域に絞ったバグ予想でも、レビューの対象や、テスト観点が絞りやすくなるために、開発の際には有用と考えた。

従来の予想モデルは、線形回帰モデルのように均一的な分散などを仮定した近似モデルが多い。

本稿では、特段の仮定をすることなく、予想モデルの構築を行うために、機械学習で予想モデルの構築を行った。学習モデルの構築には、ソースコードから直接モデルを構築するのではなく、ソースコードから計算された、メトリクスを入力として使用した。メトリクスは、オブジェクト指向型の構造の複雑度を求める CK メトリクスを使用した。

バグの存可能性のみを開発者に伝えても、対処方法に迷ってしまう。そこで、今回は、CK メトリクスから Fuzzy 推論を用いて、リファクタリングの対処方法を導出する試みも併せて行った。

テクマトリックス株式会社　R&D 室
Techmatrix　Corporation　R&D Department

東京都港区三田 3－11－24　Tel: 03-4405-7856　e-mail:nisida@techmatrix.co.jp
3-11-24, Mita, Minato-ku, Tokyo, Tokyo Japan
1) テクマトリックス株式会社　R&D 室
Techmatrix Corporation　R&D Department

【キーワード：】バグ予想、機械学習、メトリクス

2. CK メトリクス

機械学習でバグ予想モデルを構築するための入力データとして、CK メトリクスを使用したことは、「はじめに」で述べた、CK メトリクスは、Chidamber、Kemerer が 1994 年に提唱ししたオブジェクト指向プログラムの複雑度を計測するためのメトリクスである。CK メトリクスを選択した理由は、CK メトリクスを使用したバグ検出予想の先行研究が多数存在することと、これらの事例のなかで、以下の成果が得られている点である

- DIT(継承ツリーの深さ)、NOC(子クラスの数)、RFC(クラスの応答数)、CBO(オブジェクトクラス間の結合度)は、バグが存在するクラスと著しい相関を示す。
- オブジェクト指向型の尺度は、バグが存在するクラスを予想する上で、コード尺度（クラスのネストレベルの最大、メソッドの数、コール数など）より優れている.

上記の DIT、NOC、RFC、CBO を学習の特徴データとして使用すれば、精度の高い学習モデルが構築可能であると考えた。

CK メトリクスの詳細は[1][2][3]を参照。

3. 試行内容の詳細

3.1 システムの概要

今回作成したシステム構成を図 1 に示す。

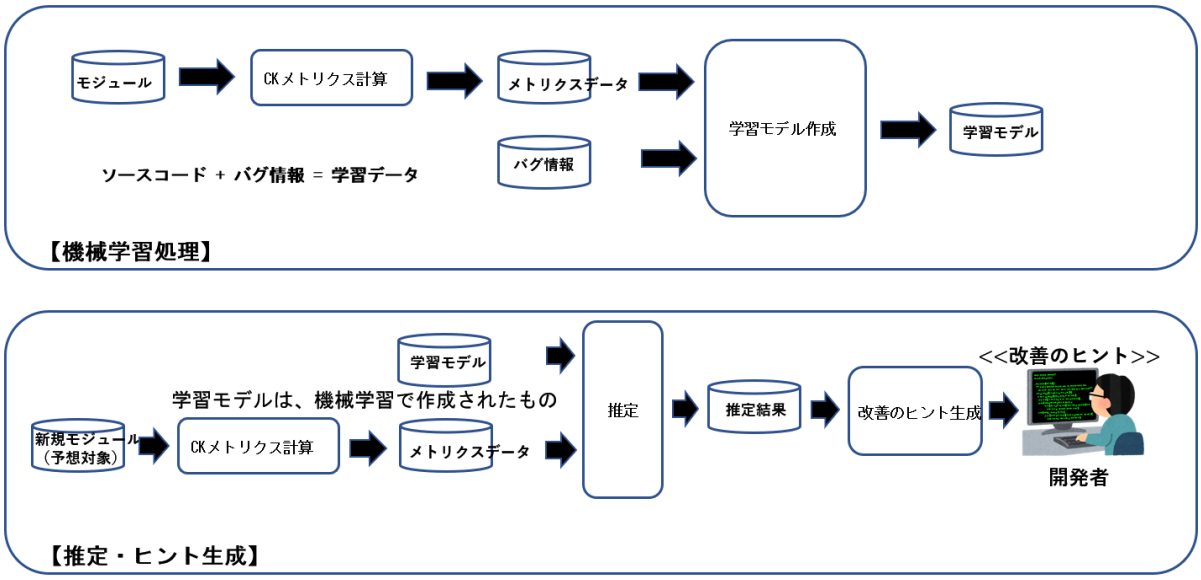

図 1

システムは、2 つの部分で構成されている。

- 「機械学習処理」は、モジュールから計算された CK メトリクスとモジュールに対応したバグ情報（Issue Tracking System から取得）から作成された学習データで機械学習を行いバグ予想の学習モデルを作成する。学習モデルは、新規モジュールのバグ予想（fault prone module であるかどうか）を行うために使用される学習モデルである。
- 「推定・ヒント生成」は、新規モジュール中のバグの有無の可能性（fault prone module であるかどうか）を推定し、併せて、問題点解決のヒントを生成する。推定を行うための学習モデルは、「機械学習」で作成した学習モデルである。

3.2 対象プログラム言語

バグ予想モデルを構築する際、キーポイントとなるのが、バグ情報である。そこで、今回対象とするプログラム言語（システムの開発言語）は、Java とした。Java で開発されているオープンソースのシステムは多数存在し、その中には、Issue Tracking System が非常に充実（バグの情報が適切に管理されている）しているものが多数存在し、かつ、リポジトリに容易にアクセス可能で学習に必要なデータの収集比較的容易である。

3.3 学習データの収集、作成

(1) 収集

入力データとしては、CK メトリクス、開発言語としては Java を作用することとした。これを基本にして、以下の基準で学習用データを収集した。

- 自社開発モジュールではなく、オープンソースとして公開されているモジュールを利用する。
- オープンソースとして公開されているシステムの中でも、Issue Tracking System でバグの管理が適切におこなわれているもの。
- モジュールに問題が発生していることの傍証をとるため、バグ情報と修正対象となったモジュールの対応が取れていること。

今回は、オーペンソースのデータとして、GitHub、Apache Software Foundation(ASF)のリポジトリを使用した。収集対象としたプロジェクト、バージョンの一覧を表 1 に示す。

表 1

システム名	バージョン
velocity	1.4 、1.5 、1.6
tomcat	6.0
poi	1.5 、2.0 、2.5 、3.0
log4j	1.0 、1.1 、1.2
jedit	3.0 、4.0 、4.1 、4.2 、4.3
ivy	1.0 、1.1 、2.0
ant	1.7

収集したデータ件数は 1283 モジュールとなった。

(2) 作成

学習データは、[ラベル（モジュール名から作成)、メトリクスデータ、バグ情報]の形式で CSV フォーマットで作成。収集されたデータから、学習データの作成に当たっては表 2 の点に留意して作成した。

表 2

メトリクスの計測対象	メトリクスは、バグ有りのメトリクスデータ、バグ無しのメトリクスデータの 2 種類のデータを作成する必要がある。 バグ有りのデータは、バグが発生した時点でのメトリクス値を計測する必要があるので、収集対象のリポジトリを参照し、バグが発生時の直前のモジュールで計測した。バグ無しのデータ一定期間モジュールの修正が無く、バグの発生も無いものを対象としてメトリクスを計測した。

正規化	CK メトリックスの各データは各々スケールが違うため、そのままでは学習を実行できない、今回は各メトリクスの不偏分散と自由度を利用して値が０～1.0 に収まるように正規化した。

CK メトリクスを計測するために使用したツールは[4]を参照。
表 3 には、今回収集したデータで、DIT、NOC、RFC、CBO とバグの有り無しとの相関係数を計算した結果を示す。

表 3

	DIT	NOC	RFC	CBO
バグ有りの相関係数	0.02	0.04	0.69	0.61
バグ無しの相関係数	0.04	0.01	0.25	0.11

3.4 機械学習

(1) モデルの概要

CK メトリクスを計測するツールは、モジュールごとに 20 個のデータを出力する。従って、入力データはすべて固定長（20 次元）であり、3.3－(2)で示した通りデータはすべて 0～1.0 の間で正規化してある。機械学習に当たっては、CK メトリクスの DIT、NOC、RFC、CBO の各値を特徴データとして生かしたので、モデルは畳み込みニューラルネットワークを使用した。コードは TensorFlow 上で実装した。
機械学習モデルの特徴は以下の通りである。

- 入力データは次元 20 の固定
- Convolution 、Pooling 層は２つ
- 2 層目の Pooling 層の後は、全結合層、Softmax 関数（分類器）
- 出力結果は 2 値（バグの有無）

概要図を図 2 に示す。

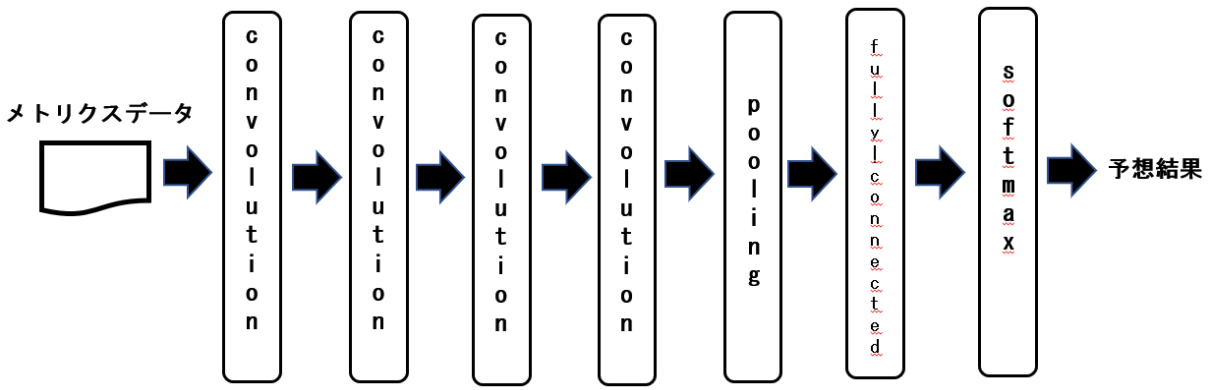

図 2

(2) 入力データの調整

バグ有りのデータの占める割合は、バグ無しのデータに比べて学習データ全体に占める割合は非常に小さい（通常 1/10 程度）。従って、学習を行う際には、バグ有りとバグ無しの比率が 1：1 になるように、入力データを調整する必要がある。これを行わないと、学習結果の評価を誤る[5]。

(3) 学習アルゴリズム、学習時パラメータ

学習アルゴリズムは、多数の手法が提唱されているが、今回は確率的勾配法の一つである AdamOptimizer を使用して、誤差関数が最小になるモデルを探索した。
学習時パラメータである、バッチサイズ、エポックサイズは複数に組み合わせで試してみたが、今回正答率が最もよかった組み合わせはバッチサイズが 100、エポックサイズが 300 の組み合わせであつた。ちなみに、正答率（accuracy）は 92%であつた。

3.4　改善点提示システム

学習モデルで、問題が発生する可能性が高いと判断されたモジュールを、開発者に対して改善案を提示する仕組を実装した（可能性のみを提示しただけでは、開発者自身で対処方法を見出すのは困難)。改善案の選択には Fuzzy 推論モデルを使用した。学習データとして準備したメトリクスデータとバグ情報から、メンバーシップ関数を決定。Fuzzy ルールの決定方法に関しては、[6]を参考に作成した。ただし、今回はバグと相関が強いと推測された、DIT、NOC、RFC、CBO の 4 種類のみを使用した。

4.　試行結果

4.1　学習モデルの評価

203 件の検証データ（バグ有りが 108 件、バグ無しが 96 件）に対して、正答率 87%となった。
正答率はおおむね良好な結果が得られた。
検証データの ROC 曲線を図 3 に示す。

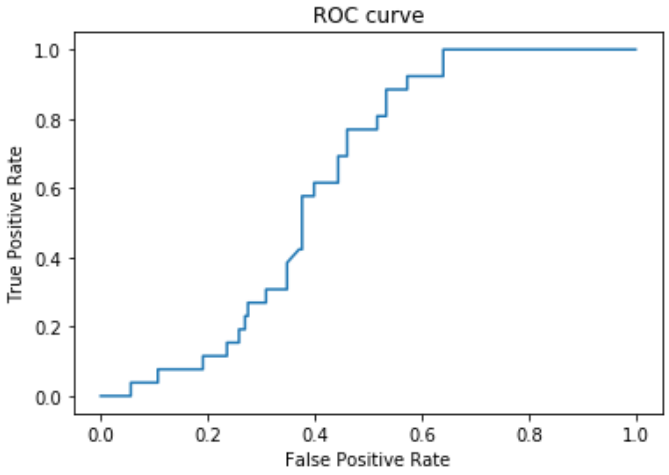

図 3

4.2 Fuzzy 推論モデル

log4j の Ver.1.1 -> Ver.1.2 への変更点の内、バグに関連した修正点をどの程度 Fuzzy 推論によってカバーできたかで評価を行った。しかし、実際のバグが CK メトリクスに関係しているかどうかは判然としないため、実際の効果は数量的に判然としなかった。

4.3 社内システムへの適用

実施結果に示したように、概ね、良好な確認結果が得られたことから、社内のシステム開発にて評価中である。導入に当たっては、下記に留意した

- CK メトリックスの値が高くなりがちなモジュール等が存在（setter、getter method のみで構成されているモジュール）する。これらのモジュールを推定対象から除外す

るなどの工夫が必要である。
- 推定結果のバグ有りと判断するべき確率（バグ有りと判断するための閾値）についても、プロジェクトのメンバーと十分打ち合わせを行い、決定する必要がある。

これらの調整を行わず、モジュール個々の（今回は約 3000 モジュール）推定確率の一覧を提示しても、結果を理解してもらうことは不可能。

これらの運用上の工夫の結果、バグ有りと推定されたモジュールが 4 つと限定されたため、レビュー、テストの集中化　に対して効果ありとの感想を得た。

5. 今後

今後は、学習データの収集範囲を広げると同時に、社内で開発されたシステムについても学習データとして加えた。また、今回は、CK メトリクスを使用しているが、他の言語にも適用範囲を拡大するために、他のメトリクスについて調査を行う必要がある。これにより推定精度の向上を図るだけではなく、適用範囲を拡大したい。

Fuzzy 推論での修正案は、まだ実際の開発者との乖離が大きいで、より現場の実感に近い改善版を提示できるようにメンバー関数等を改善して行きたい。

6. 参考文献

［1］http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/
［2］StephenH. Kan （著）, 古山 恒夫（翻訳）, 富野 寿（翻訳）,ソフトウェア品質工学の尺度とモデル, 共立出版　272 - 277, 2004
［3］倉下 亮,吉村 博昭,野中 誠,誉田 直美, ＣＫメトリクスの分布に基づくソフトウェア設計の質の定量的評価, SQiP シンポジウム　, 2011
［4］https://github.com/mjureczko/CKJM-extended
［5］Nitesh V. Chawla, Kevin W., Lawrence O. Hall, W. Philip Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 (2002), 321-357, 2002
［6］秦野克彦、乃村能成、谷口秀夫、牛島和夫、ソフトウェアメトリクスを利用したリファクタリングの自動化支援、情報処理学会論文誌、Vol. 44 No.6、1548-1558、2003