Vol.2016-SE-193 No.24
2016/7/15

Deep Learningのリポジトリマイニングへの適用に向けた初期研究

松本　卓大[1,a)]　山下　一寛[1,b)]　亀井　靖高[1,c)]　鵜林　尚靖[1,d)]

概要：近年，Deep Learning はパターン認識や強化学習，機械翻訳などの様々な研究分野で成功している（既存の研究手法よりも高い精度での分類や予測を実現している）．Deep Learning が従来の機械学習を用いた手法よりも高い精度を持つ要因の 1 つとして事前学習による特徴量の選択がある．一方，リポジトリマイニングの分野において分析で使用する特徴量の選択は重要な課題である．リポジトリマイニングの分野において，Deep Learning による特徴選択がうまく機能すれば有用である．そこで本研究では，リポジトリマイニング分野における Deep Learning の特徴選択の有用性を確かめることを目的とし，初期調査として，ノイズを含むデータセットを対象に Deep Learning の事前学習である Deep Belief Network（DBN）を適用しバグ予測の分析を行った．1）学習データにノイズを追加することにより回帰分析の精度にどの程度影響するのか，2）DBN による特徴選択によって回帰分析の精度にどの程度影響するのか，という 2 つの調査を行った．その結果，1）ノイズの有無で Accuracy，AUC，F-measure，Precision がそれぞれ 16%，18%，12%，41%の精度の低下がある，2）DBN による特徴選択により Accuracy，AUC の精度低下をそれぞれ 14%，5%抑えることができる，3）DBN による特徴選択を行った場合の Precision の値が，ノイズ無しと同等になる，ということがわかった．

キーワード：機械学習，Deep Learning，リポジトリマイニング，バグ予測

1. はじめに

近年，Deep Learning はパターン認識 [5][11][12] や強化学習 [8]，機械翻訳 [13] などの様々な分野で成功している（既存の研究手法よりも高い精度での分類や予測を実現している）．Deep Learning が従来の機械学習を用いた手法よりも高い精度を持つ要因の 1 つとして，事前学習による特徴量の抽出がある．従来の機械学習を用いた手法では，特徴量として何を用いるかは，研究者や実務者が決めており，機械学習・適用対象・データセット等に対する深い理解が必要不可欠であった．一方で，Deep Learning では多層の RBM（Restricted Boltzmann Machine）や AutoEncoder を用いて多数の特徴量（説明変数）から自動的に学習に必要なものを抽出する．

画像認識などと同様に，リポジトリマイニングの分野において，工数予測やバグ予測などの分析で使用する特徴量の選択は重要な課題である．リポジトリマイニングの分野において，Deep Learning による特徴選択がうまく機能すれば，分析に有効な特徴量が自動的に選択されるので有用である．

本研究では，リポジトリマイニング分野における Deep Learning の有用性を確かめることを目的とし，その初期調査として，バグ予測の分析に Deep Belief Network（DBN）を用いた特徴選択を適用した．従来のバグ予測に用いられている特徴量にノイズデータとして乱数を生成し新たな特徴量として追加し，Deep Learning の特徴選択を適用した際に，うまく特徴量が選択されるかを確かめる．ノイズデータをデータセットして用いる理由としては，多数のデータの中からバグ予測に有用な特徴を上手く抽出できるか確かめるためである．

バグ予測に用いられている特徴量としては，D'Ambros ら [1] によって公開されているデータセットのうち，Eclipse JDT Core の change metrics をベースの特徴量とする．Change metrics は，SVN や Git といった版管理システムのログから取得することのできるソフトウェアの変更に関するメトリクス（例えば，変更行数や変更回数など）であり，既存研究 [1], [9] でもバグ予測に用いられている基本的

[1] 九州大学
Kyushu University
[a)] 2IE15096T@s.kyushu-u.ac.jp
[b)] yamashita@posl.ait.kyushu-u.ac.jp
[c)] kamei@ait.kyushu-u.ac.jp
[d)] ubayashi@ait.kyushu-u.ac.jp

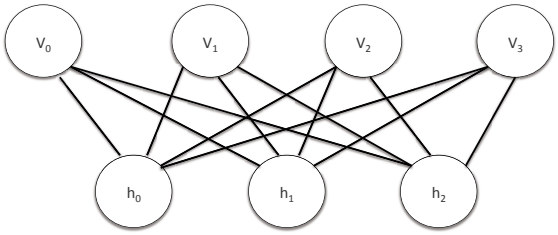

図 1 RBM の構造

なメトリクスである．このデータセットに対して，1）学習データにノイズを追加することにより回帰分析の精度にどの程度影響するのか，2）DBN による特徴選択によって回帰分析の精度にどの程度影響するのか，の 2 つの初期調査を行う．

以降，第 2 章では，本研究の背景について紹介する．第 3 章では，本研究で行った初期調査の調査方法と調査結果，結果に対する考察について述べる．第 4 章では，本研究の初期調査に関する議論を記述する．第 5 章では，関連研究について述べる．最後に第 6 章では，本研究のまとめと今後の課題について述べる．

2. 背景

2.1 Deep Learnig

Deep Learning とは多層構造のニューラルネットワーク [*1] による機械学習手法である [18]．Deep Learning のモデルにデータを入力すると，データが第 1 層から伝達されていき，各層で学習が繰り返される．この技術は主に画像認識や音声認識等の研究分野で活用されてきた [5][11][12]．

Deep Learning の特徴の 1 つとして特徴選択がある．従来の画像認識では特徴量の選択を手動で行ってきた一方で [23], Deep Learning では特徴量の選択を中間層で自動的に計算する．特徴選択に用いられる手法として，大きく分けて Restricted Boltmann Machine（RBM）と AutoEncoder の 2 つがある．両者の違いとしては，ネットワークの挙動を確率的に記述するか決定論的に記述するかであり，一般に RBM の方が AutoEncoder よりも良い性能を発揮するといわれている [19]．Deep Learning の特徴選択では，RBM を多数重ねて，下の層から順に RBM を 1 つずつ学習させるという，Deep Belief Network（DBN）が用いられる．

2.2 Deep Belief Network（DBN）

Deep Belief Network（DBN）は，Deep Learning における特徴選択法である．DBN では多数の RBM の層を含んでいる．RBM の構造について図 1 に示す．RBM は入力層と隠れ層という 2 つの層で構成される．入力層は可視変数のユニットにより構成される．隠れ層は隠れ変数のユニットにより構成される．各層のユニット同士は無向エッジにより結合されるが，同層のユニット間は結合されない．それぞれの層のユニット数は経験的に選択される．RBM の原理としてはネットワークのエネルギーを最小化することを目的として評価する．RBM のエネルギー関数は下記の様に表される．

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_j w_i j$$

上記に等式において，w は可視変数と隠れ変数間の結合の重み，a 及び b は可視変数及び隠れ変数のバイアスである．このエネルギー関数により，下記に示す様な RBM の 2 層の同時確立を獲得できる．

$$p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}e^{-E(\boldsymbol{v},\boldsymbol{h})}, where Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$

$\boldsymbol{v}$ に関する確立分布関数は下記の様に表される．

$$P(\boldsymbol{v}) = \sum_h P(\boldsymbol{v},\boldsymbol{h}) = \sum_h \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}$$

上記の式より，RBM のエネルギー関数を最小化するためには $P(\boldsymbol{v})$ を最大化する必要がある．これにより，RBM の最適な w，a，b を獲得する．

2.3 従来のバグ予測における特徴選択

バグ予測において、分析しようする特徴量の選択は重要な課題である．従来のバグ予測においては，プロダクトメトリクス，プロセスメトリクスの中から使用する特徴量を選択してきた．プロダクトメトリクとは，ソースコードに関するメトリクスである．代表例としては，「ソースコード行数（Lines of Code：LOC）」がある [1][7]．プロセスメトリクスは，ソフトウェア開発の過程に関するメトリクスである．代表例としては「ソースコードの変更回数」，「発生したバグの数」，「モジュールの存在期間」等がある [2][15]．この他にも組織メトリクス [22] といったメトリクス等もあり，多数のメトリクスの中から使用するメトリクスを選択することは予測モデルに与える影響が大きく，バグ予測においては重要な課題である．

2.4 本研究の位置づけ

本研究では，リポジトリマイニングにおける Deep Learning の有用性を確認することを目的としてバグ予測に Deep Learning を適用する．従来のバグ予測で課題となっている特徴量の選択に Deep Learnig の特徴選択である DBN を用いて，バグ予測に有用な特徴をうまく選択できるか確認する．

3. 初期調査

本研究では，リポジトリマイニング分野における Deep

[*1] ニューラルネットワークとは，脳神経系を模した数学モデルである，多数の人工ニューロンによりネットワークを形成し，シナプス結合強度を変化させて問題解決能力を獲得する [20]．

表 1 バグ予測に使用する特徴量

項目	説明	最小値	中央値	最大値
numberOfVersionsUntil	バージョン数	1	30	709
numberOfFixesUntil	バグ修正数	0	5	166
numberOfRefactoringsUntil	リファクタリング数	0	0	2
numberOfAuthorsUntil	Author の数	1	6	15
linesAddedUntil	追加行数	0	276	655,571
maxLinesAddedUntil	追加行数の最大値	0	82	7,452
avgLinesAddedUntil	追加行数の平均	0	10	222.2
linesRemovedUntil	削除行数	0	199	59,724
maxLinesRemovedUntil	削除行数の最大値	0	66	7,452
avgLinesRemovedUntil	削除行数の平均	0	6.7	206.2
codeChurnUntil	変更行数	-1,745	31	10,624
maxCodeChurnUntil	変更行数の最大値	0	23	2,768
avgCodeChurnUntil	変更行数の平均値	-39.9	1.4	121.4
ageWithRespectTo	年齢	7.857	329.1	367
weightedAgeWithRespectTo	重み付き年齢	0	63.2	227.6

Learninng の有用性を確かめる事を目的として, Deep Learning の特徴の 1 つである DBN による特徴選択を用いた初期調査を行った．具体的には，ノイズとして乱数を追加したデータセットに対して DBN による特徴選択を適用し，回帰分析を行った際に結果の精度がどのように変化するかについて調査した．

3.1 データセット

基本となるバグ予測のためのデータセットとして D'Ambros らが公開しているデータセット [1][*2] のうち，Eclipse JDT Core を用いる．データセットの中から，特徴量として change metrics を利用した．Change metrics は，CVS や SVN，Git などの版管理システムから得ることができるソフトウェアの変更に関する基本的な特徴量であり，様々なバグ予測の研究でも用いられている [1], [9]．本データセットに含まれる，具体的な特徴量とその値の分布を表 1 に示す．

このデータセットは，Eclipse JDT Core を構成する各クラスごとの各 change metrics と post release bug の数，また post release bug の分類（nonTrivialBugs，majorBugs，criticalBugs，highPriorityBugs）が含まれている．Eclipse JDT Core は 997 のクラスから構成され，うち 206 のクラス（21%）がバグを含む．本研究は簡単のため，バグの分類を利用せず，また，バグの数からバグの有無に変換して利用する．

3.2 調査 1．学習データにノイズを追加することにより回帰分析の精度にどの程度影響するのか

DBN による特徴選択の効果を確認するために，特徴選択を行わない場合の回帰分析の精度を確認する．関係の無いデータ（ノイズ）がデータセットに含まれていることで，回帰分析の精度にどの程度の影響があるのかを確認する．

3.2.1 アプローチ

Eclipse JDT Core の change metrics のデータセットに加え乱数を用いて生成したノイズを含むデータを用いて，ロジスティック回帰分析を用いたバグ予測を行う．ノイズが含まれていないデータセットをベースとし，ノイズが増えることによりどのように精度が変化するかを見る．ノイズの項目数は 0-500 の間で変化させていき [*3]，それぞれでロジスティック回帰分析を用いた 10-fold cross validation[*4] を行い，作成したモデルの精度を評価する．精度の評価には，10 回の試行での Precision，Recall，F-measure，AUC，Accuracy の平均値を計算することにより行う．

モデリング手法． ロジスティック回帰分析において，判別関数はロジスティック関数の形で表される．

各ケースに対して，p 個のメトリクスが観測されるとき，判別関数は以下の通りになる．

$$P(y|x_1,\cdots,x_p) = \frac{1}{1+e^{-(\alpha_1 x_1+\cdots+\alpha_p x_p)}}$$

ここで，$y \in 0,1$ は群を表す目的変数（本論文ではバグの有無），x_i は説明変数（各メトリクス），α_i は判別係数であり，$P(y|x_1,\cdots,x_p)$ は，説明変数の組 $x_1,\cdots,x_p$ に対して，y が 1 を取る確率である．本論文では既存研究 [4] を参考に，$P(y|x_1,\cdots,x_p)$ が 0.5 以上の値をとる場合，バグを含むクラスであると判定する．

評価指標． 本論文では，モデルの評価の指標として，Accuracy，Precision，Recall，F-measure，AUC を用いる．表 2 に判別結果の分類を示し，表内の記号を用いて各指標につ

[*2] http://bug.inf.usi.ch/download.php

[*3] 実際には（0，1，5，10，50，100，200，300，400，500）個とノイズの数を変化させた．

[*4] 10-fold cross validation では，標本群を 10 個に分割し，そのうちの 1 つをテストデータ，残りを訓練データとして学習を行う．10 個に分割された標本群をそれぞれテストデータとしていき 10 回検証を行う．得られた 10 回分の結果を平均して 1 つの推定を得る

表 2　判別結果の分類

		予測値	
		陽性（Positive）	陰性（Negative）
正解	陽性	真陽性（True Positive：TP）	偽陰性（False Negative：FN）
	陰性	偽陽性（False Positive：FP）	真陰性（True Negative：TN）

いて説明する．Accuracy は，予測が当たった割合を示し，以下のように表される．

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision は，バグ有りと予測し，実際にバグがある割合を表し，以下の式で表される．

$$Precision = \frac{TP}{TP + FP}$$

Recall は，全てのバグを含むクラスのうち，バグ有りと予測した割合を表し，以下で表される．

$$Recall = \frac{TP}{TP + FN}$$

Precision と Recall はトレードオフの関係にあるため，Precision と Recall のみでは精度評価を行うのが難しく，Precision と Recall の調和平均で与えられる F-measure が評価基準としてしばしば用いられる．F-measure は以下の式で表される．

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

また，Area Under the Curve（AUC）は，Receiver Operating Characteristic（ROC）曲線の下側の面積であり，AUC は予測モデルの性能の良さを表す．AUC は，[0, 1] の値をとり，0.5 以上であることは，ランダムに判断することよりも効果的であることを示す．ROC 曲線は，x 軸を false positive の割合（$\frac{TP}{TP+FN}$），y 軸を true positive の割合（$\frac{FP}{FP+TN}$）とし，分類において取りうる全ての閾値上で変化させ描画した曲線である．

また，それぞれのメトリクスの値は [0, 1] の値域に正規化を行った．あるメトリクスの値 v_i の正規化された値 $norm(v_i)$ は以下の式によって求められる．

$$norm(v_i) = \frac{v_i - max(\boldsymbol{v})}{max(\boldsymbol{v}) - min(\boldsymbol{v})}$$

ここで，$min(\boldsymbol{v})$ と $max(\boldsymbol{v})$ はそれぞれのメトリクスの最小値と最大値を表す．

3.2.2　調査結果と考察

ノイズの項目数による精度の変化について図 2 に示す．Recall を除く各指標ではノイズの増加とともに，その値が減少している．例えば，AUC の場合，ノイズを含まない際の AUC の値は 0.81 である一方，200 項目のノイズを含む際には 0.69 である．また，ノイズの数が 300 項目あたりまでは値が大きく減少し，それ以降はわずかに減少している．Accuracy, AUC, F-measure, Precision で，ノイズを含まない場合とノイズを 500 項目含む場合で，それぞれ 16%，18%，12%，41%の精度の低下がみられる．

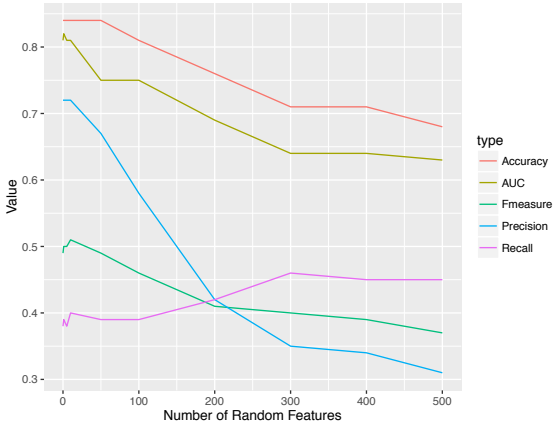

図 2　ノイズの項目数による精度の変化

一方で，Recall に関しては，他の指標と反対の傾向を示した．この理由としては，ノイズの項目数が増えることで，回帰分析がデータの傾向をうまくとらえることができず，ランダムにバグの有無を判別する傾向が強くなった結果であると考える．ランダムにバグの有無を判別する傾向が強くなった証拠として，AUC の値が 0.5 に近くなっている（ランダムに予測すると AUC は 0.5）．

これらの結果により，ノイズを含むデータセットでは回帰分析の性能が低下することが確認できた．

3.3　調査 2．DBN による特徴選択によって回帰分析の精度にどの程度影響するのか

ノイズが含まれたデータを用いた場合にも，DBN による特徴選択によって回帰分析の精度が向上するのか，また，ノイズ量に応じてどの程度回帰分析の精度に影響があるのか確認する．

3.3.1　アプローチ

Yang ら [17] の研究を参考に，本論文では 3 層の RBM を用いて，ノイズを含んだデータセットに対して特徴選択を行う．その後，調査 1 と同様に，選択された特徴量を用いてロジスティック回帰分析を行い，10-fold cross validation によって評価を行う．図 2 から，ノイズが 300 個以上ではさほど精度に変化が無かったため，調査 2 ではノイズの数を 300 個とする．

DBN は複数の RBM を含んだネットワークであり，我々は RBM の数と各隠れ層のユニット数を設定できる．RBM の数は，先にも述べたとおり Yang ら [17] を参考に 3 層の RBM を用いることにした．この 3 層の RBM は，Hinton ら [3] によって推奨された一般的な設定である．各層のユニット数の組み合わせによって選択される特徴量は変化す

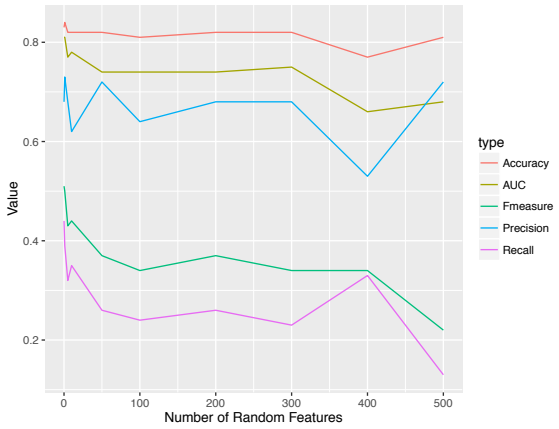

図 3 DBN 用いて特徴選択を行った場合のノイズの数による精度の変化

る．そのため，全ての組み合わせについて精度を比較することが好ましいと考えられるが，本論文では簡単のため各層が選択可能なユニット数を（2，3，4，5，10，15，20，25，30）とした．

RBM の数やユニット数に加えて，RBM に関するパラメータとして epoch 数や GibbsSampling の回数などを設定することが可能であるが，本論文では，実装に利用したパッケージの初期値を利用する [10]．

3.3.2 調査結果と考察

DBN を用いて特徴選択を行った場合のノイズの数による精度の変化について図 3 に示す．Precision，Accuracy，AUC の値は特徴選択を行った場合の方が，ノイズの増加による値の減少を抑えている．Accuracy，AUC のノイズを含めた場合の精度の低下が，特徴選択を行わない場合ではそれぞれ，16%，18%であったのに対して，特徴選択を行った場合にはそれぞれ，2%，13%となり，それぞれ 14%，5%の精度低下を抑えている．特に特徴選択を行った場合の Precision の値は，ノイズ無しの場合に 0.68，ノイズの数が 300 項目の場合に 0.68 と，ノイズの増加による精度の低下の影響を受けていない．これは特徴選択によりノイズを上手く除去できているためではないかと考えられる．

一方で，Recall，F-measure の値は特徴選択を行った方が精度が低下した．ノイズの数が 500 の場合において，特徴選択を行わない場合の Recall，F-measure の値がそれぞれ 0.45，0.37 であるのに対して，特徴選択を行った場合には，0.13，0.22 とそれぞれ 32%，15%精度が低下している．Recall が低下した原因については，調査することが今後の課題である．

4. 議論

4.1 最適なパラメータとコスト

DBN は複数の RBM を含んだネットワークであり，RBM の数や各隠れ層のユニット数の組み合わせにより選択される特徴量は変化する．RBM の数が増加するほど設定しなければならないユニット数は増え，各隠れ層のユニット数の組み合わせは指数的に増加する．今回は 3 層の RBM を用いて，各層が選択できるユニット数を（2，3，4，5，10，15，20，25，30）に制限し最も精度の高くなったユニット数を選択した．これらの組み合わせであれば，実行時間は数分であったが，全ての組み合わせを確認するための実行時間はあまり現実的ではない．特徴選択において分析に求める精度と分析にかけることができるコストは十分に検討する必要がある．

4.2 従来の特徴選択法との関連

本研究では，事前学習における特徴選択法として DBN を用いた．しかしながら，従来の特徴選択法として他にもいくつかの特徴選択法がある．一般に特徴選択法は，(1) Filter 手法：予測アルゴリズムとは独立に，変数間の関係にのみに着目して選択する変数を決定する手法，(2) Wrapper 手法：予測アルゴリズムを実績データのサンプルに繰り返し適用・評価することで，選択する変数を探索的に決定する手法，の 2 つに分かれる [21]．(1) の Filter 手法の例としてはー相関係数を用いた選択ーが挙げられており，DBN は Fliter 手法に属している．(2) の wrapper 手法としては，ー全探索法ーやー山登り法ーがある．同様のデータセットの場合であっても，適用する特徴選択法により結果の出力は異なる．今回用いたデータセットに対して他の特徴選択法を適用した場合にどのような変化があるか調査する必要がある．

4.3 出力結果の解釈の困難さ

DBN では多数の特徴量から多数の隠れ層で特徴選択し指定した項目数の結果を出力する．そのため，出力の値に対する意味付けが難しい．出力の値の意味付けが難しいため，分析の精度が低下した場合の原因の特定が難しい．画像認識の場合，特徴量の意味付けができていなかったとしても，分析結果が良ければ問題ない．しかしながら，工数予測やバグ予測といったリポジトリマイニングの分野においては結果に対する合理的な判断理由を求められる場合があり，DBN では理由を説明することができない．リポジトリマイニングにおける DBN の使用は分析に求める要求に応じて検討する必要があるのではないかと考えられる．

5. 関連研究

本論文と同様に，Deep Learning をリポジトリマイニングに適用した研究はいくつか存在する [6][14][16][17]．

Yang ら [17] は，Just-In-Time バグ予測（JIT バグ予測）の精度向上を目的に，本論文と同様に 3 層の RBM を含む，DBN を JIT バグ予測に適用した．Yang らは，ロジスティック回帰の 2 つの問題点，1）特徴量を組み合わせられ

ない点（例えば，2 つの特徴量 x と y があり，$x \times y$ がより重要な場合は，新たな特徴量として $x \times y$ を入力する必要がある），2）入力と出力が線形の関係であるときのみ性能がよくなる点，に対処するため DBN を用いた．その結果，ベースとなる研究よりも 33%ほど多くバグを正しく予測でき，F-measure に関しても既存研究よりも高い値であった．一方で，本論文では，特徴量の選択に Deep Learning の事前学習が利用できるかに着目し，ランダムな変数ではなく，従来バグ予測に用いられる特徴（少なくとも，ランダムな変数よりも有力でありそうな）を選択できるかを調査した点が異なる．

Wang ら [14] は，従来のバグ予測に用いられるメトリクスでは，十分にソースコードの意図を反映できていないと考え，ソースコードの意図をバグ予測に用いるため，Deep Learning を用いた．例えば，同じ行数，複雑度であっても，**if** 文と **for** 文では開発者の意図は異なる．そこで，これらのソースコードの意図を汲み取るため，Abstract Syntax Trees（ASTs）を入力とし DBN によって特徴量の選択を行い，その特徴量を用いてバグ予測を行った．その結果，同一プロジェクト内でのバグ予測では，既存研究よりも Precision で 14.7%，Recall で 11.5%，F-measure で 14.2%高い精度を得た．また，プロジェクトをまたいだバグ予測では，F-measure で 8.9%高い値を得た．

Lam ら [6] は，バグレポートを用いたバグの位置特定に Deep Learning を適用した．彼らは，情報検索（IR）の一手法である revised Vector Space Model（rVSM）と Deep Neural Network（DNN）を組み合わせた新しいモデル HyLoc を提案した．6 つの OSS プロジェクトを用いた HyLoc の評価実験では，IR を用いた手法や Naive Bayes などの機械学習手法と比較して高い Accuracy を得ている．

White ら [16] は，software language model に対して Recurrent Neural Network（RNN）を適用し，従来の手法である n-gram と有用性について比較を行った．Perplexity（PP）を用いた n-gram との比較において RNN を用いた提案手法は n-gram を用いた従来手法よりも良かった．また，software language model を用いたソフトウェア工学のタスクとしてコード推薦を取り上げ，ケーススタディとして RNN を用いた手法と n-gram を用いた手法の比較を行った．その結果，ケーススタディにおいても，RNN を用いた提案手法の方が従来手法よりもより高い精度であることを示した．

6. まとめと今後

本論文では，リポジトリマイニングにおける Deep Learning の特徴選択の有用性を確かめることを目的として，ノイズを含むデータセットを用いたバグ予測の分析に，DBN を用いた特徴選択を適用した．1）学習データにノイズを追加することにより回帰分析の精度にどの程度影響するのか，2）DBN による特徴選択によって回帰分析の精度にどの程度影響するのか，という 2 つの初期調査を行い，1）ノイズの有無で Accuracy，AUC，Fmeasure，Precison がそれぞれ，16%，18%，12%，41%の精度の低下がある 2）DBN による特徴選択により Accuracy，AUC の精度低下をそれぞれ 14%，5%抑えることができる，3）DBN による特徴選択を行った場合の Precision の値が，ノイズ無しの場合とノイズを 300 項目含めた場合とで等しい，ということが分かった．

また，4 章でも述べた通り，リポジトリマイニングに Deep Learning の特徴選択を適用する際の課題として，「分析結果に対して用いた入力データにより合理的判断を行うことが難しい」ということがわかった．

今後の課題として，1）他の代表的な特徴選択法を用いた場合の分析結果の精度比較，2）ノイズを含まないバグ予測に関する多数の特徴量をデータセットとした場合の分析結果の精度比較，等が挙げられる．

参考文献

[1] M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In *Proceedings of MSR 2010 (7th IEEE Working Conference on Mining Software Repositories)*, pages 31 – 41. IEEE CS Press, 2010.

[2] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy. Predicting fault incidence using software change history. *Software Engineering, IEEE Transactions on*, 26(7):653–661, 2000.

[3] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[4] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K. Matsumoto. The effects of over and under sampling on fault-prone module detection. In *Proc. Int'l Symposium on Empirical Softw. Eng. and Measurement (ESEM'07)*, pages 196–204, 2007.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen. Combining deep learning with information retrieval to localize buggy files for bug reports (n). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 476–481, Nov 2015.

[7] A. Meneely, L. Williams, W. Snipes, and J. Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23. ACM, 2008.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[9] R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of*

the 30th International Conference on Software Engineering, ICSE '08, pages 181–190, New York, NY, USA, 2008. ACM.

[10] X. Rong. *deepnet: deep learning toolkit in R*, 2014. R package version 0.2.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[12] F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440, 2011.

[13] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[14] S. Wang, T. Liu, and L. Tan. Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 297–308, New York, NY, USA, 2016. ACM.

[15] E. J. Weyuker, T. J. Ostrand, and R. M. Bell. Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models. *Empirical Software Engineering*, 13(5):539–559, 2008.

[16] M. White, C. Vendome, M. Linares-Vásquez, and D. Poshyvanyk. Toward deep learning software repositories. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 334–345, Piscataway, NJ, USA, 2015. IEEE Press.

[17] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun. Deep learning for just-in-time defect prediction. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 17–26. IEEE, 2015.

[18] 岡谷貴之. 深層学習 (機械学習プロフェッショナルシリーズ), 2015.

[19] 久保陽太郎 et al. ディープラーニングによるパターン認識. **情報処理**, 54(5):500–508, 2013.

[20] 船橋賢一. 特集 ニューラネット 階層型ニューラルネットワークの原理的機能. **計測と制御**, 30(4):280–284, 1991.

[21] 瀧進也, 戸田航史, 門田暁人, 柿元健, 角田雅照, 大杉直樹, 松本健一, et al. プロジェクト類似性に基づく工数見積りに適した変数選択法. **情報処理学会論文誌**, 49(7):2338–2348, 2008.

[22] 畑秀明, 水野修, and 菊野亨. 不具合予測に関するメトリクスについての研究論文の系統的レビュー. **コンピュータソフトウェア**, 29(1):106–117, 2012.

[23] 柳井啓司 et al. 一般物体認識の現状と今後, 2007.