2007－SE－155 (7)
2007／3／22

Fault-Proneモジュール判別モデルに対する外れ値除去法の適用効果

松本 真佑† 亀井 靖高† 門田 暁人† 松本 健一†

† 奈良先端科学技術大学院大学 情報科学研究科

要 旨

本稿ではfault-proneモジュール判別モデルに対する外れ値除去法の効果を実験的に明らかにする．外れ値除去法とは，標本の中から外れ値を検出・除去する方法であり，判別モデル構築の前処理に用いることができる．実験では3つの代表的な判別モデル（線形判別分析，ロジスティック回帰分析，分類木）に対して，2つの外れ値除去法（MOA，LOFM）を適用した場合の計6通りについて比較を行った．実験の結果，外れ値除去法を適用しない場合と比べて，MOAを適用した場合は，F1値が最小0.04，最大0.17，平均0.10向上し，全ての判別モデルにおいて判別精度が向上した．一方，LOFMを適用した場合は，最小-0.01，最大0.04，平均0.01変化し，判別モデルによってはF1値が向上したものの，その幅はMOAと比べて小さかった．

Comparison of Outlier Detection Methods in Fault-Prone Module Detection Models

Shinsuke Matsumoto,† Yasutaka Kamei,† Akito Monden† and Ken-ichi Matsumoto†

† Graduate School of Information Science, Nara Institute of Science and Technology

Abstract

In this paper, we experimentally evaluate outlier detection methods, which detect data points that are far away from others in a data set, in terms of improving the prediction performance of fault-prone module detection models. In the experiment, we compared two outlier detection methods (MOA, LOFM) each applied to three well-known fault-prone module detection models (LDA, LRA, CT). The result showed that MOA improved F1-values of all fault-proneness models (0.04 at minimum, 0.17 at maximum and 0.10 at mean) while improvements by LOFM were relatively small (-0.01 at minimum, 0.04 at maximum and 0.01 at mean).

1 はじめに

ソフトウェア開発において，限られた開発期間で信頼性を確保するためには，テスト工程の効率化が重要である[14]．その一つの手段は，欠陥（fault）を含んでいる可能性の高いモジュール（以降，fault-proneモジュールと呼ぶ）を特定し，テスト工数を重点的に割り当てることである[1][13]．そのために，従来，fault-proneモジュール判別モデルが数多く提案されている[6][15][17]．

Fault-proneモジュール判別モデルは，モジュールの特性値（ソースコード行数や分岐の数，サイクロマティック数など）を説明変数とし，モジュール

の fault の有無を目的変数とする数学的モデルであり，線形判別分析，ロジスティック回帰分析，分類木などが用いられる．判別モデルの構築（モデルのパラメータの推定）は，過去のソフトウェア開発における各モジュールの特性値と fault の有無を記したデータセット（以降，フィットデータセットと呼ぶ）を用いて行われる．

本稿では，判別モデルの構築における課題の一つである外れ値の問題に着目する．一般に，フィットデータセットには外れ値（分岐の数が多いにもかかわらず，fault を含んでいないモジュールなど）が存在しており，この外れ値が判別モデルの精度を低下させる原因となる [11]．このため，外れ値はあらかじめフィットデータセットから除去した上で，判別モデルを構築することが望ましい．外れ値を検出し，除去する手法（外れ値除去法）はこれまでに多数提案されている [3][12][18][19] が，fault-prone モジュール判別モデルに対する適用事例はほとんど報告されていない．

そこで本稿では，fault-prone モジュール判別に対して，いずれの外れ値除去法が最も効果があるかを明らかにすることを目的とする．そのために，3 つの代表的な fault-prone モジュール判別モデル（線形判別分析 [5]，ロジスティック回帰分析 [4]，分類木 [2]）に対して，2 つの外れ値除去法 Mahalanobis Outlier Analysis (MOA)，Local Outlier Factor Method (LOFM) を適用した場合のそれぞれの判別精度を比較する．実験では，NASA が公開しているにモジュールの特性値と fault のデータ [16] を用いた．

以降，2 章で実験に用いた外れ値除去法について説明し，3 章で実験の方法と手順について説明する．4 章で結果と結果に対する考察を述べ，5 章で関連研究を紹介し，6 章で本稿のまとめと今後の課題を述べる．

2 外れ値除去法

外れ値除去法とは，標本（本稿ではモジュール）の中から外れ値を検出し，母集団（本稿ではモジュールのデータセット）の中から除去する方法である．本稿では各標本の外れた度合いを算出し，ある閾値を超えるものを外れ値と見なす 2 つの手法について比較を行う．

以降，実験に用いた 2 つの外れ値除去法について説明する．

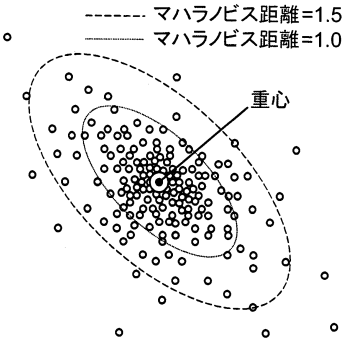


図 1: マハラノビス距離の例

2.1 Mahalanobis Outlier Analysis

Mahalanobis Outlier Analysis (MOA) とは，マハラノビス距離が閾値 θ_{MOA} を超える標本を外れ値と見なし，除去する距離に基づいた手法である [9].

マハラノビス距離とは，母集団の分散に基づいて算出される距離尺度であり，母集団の重心と標本との距離を表す．**図 1** は，ある母集団とマハラノビス距離の例を示す．図中の点は標本を表し，楕円はマハラノビス距離の等高線を表す．この図のように標本が楕円状に分布している場合，マハラノビス距離の等高線は楕円を描く．$\overline{x}$ を各変数の平均値のベクトル，x_i を標本 i の持つ変数のベクトル，n を標本の数，T を転置ベクトルとしたとき，共分散行列 S は

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T \qquad (1)$$

と表され，標本 i のマハラノビス距離 MD_i は

$$MD_i = \sqrt{(x_i - \overline{x})^T S^{-1}(x_i - \overline{x})} \qquad (2)$$

と表される．MD_i は平均が 1 になるように正規化され，標本 i が重心と同じ位置にある場合 $MD_i = 0$ となり，重心から離れた標本ほど大きな値をとる．

これらのことから，MOA とは母集団の重心から離れた標本を外れ値と見なす手法であるといえる．

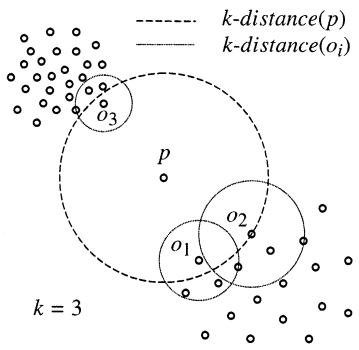


図 2: Local Outlier Factor の例

2.2 Local Outlier Factor Method

Local Outlier Factor Method (LOFM) とは，母集団の局所的な密度に基づいて算出される値LOF[3]が，閾値 θ_{LOFM} を超える標本を外れ値と見なし，除去する密度に基づいた手法である．

図 2 は $k = 3$ のときの LOF の例である．標本 p に対する k 最近傍を $o_i(i = 1, 2, ..., k)$ とする．$k\text{-}distance(p)$ は p と k 最近傍の距離の最大値を表す．ここで，標本 p の LOF の値 $LOF_k(p)$ は，$k\text{-}distance(p)$ と $k\text{-}distance(o_i)$ の比が大きいときに 1 を超える値をとるように算出される．**図 2** の p に関しては，$k\text{-}distance(p)$ が $k\text{-}distance(o_i)$ に比べ非常に大きく，$LOF_k(p) \gg 1$ となる．逆に，一定の密度でばらついている標本は，$k\text{-}distance(p)$ と $k\text{-}distance(o_i)$ の比が小さく，$LOF_k(p) \fallingdotseq 1$ となる．

つまり LOFM は $k\text{-}distance(p)$ と $k\text{-}distance(o_i)$ の比を密度と見なして，密度がばらついた領域にある標本を外れ値と見なす手法であるといえる．

3 実験

3.1 実験概要

実験の目的は，fault-prone モジュール判別モデルに対して，最も判別精度の向上につながる外れ値除去法を明らかにすることである．そのために，3 つの fault-prone モジュール判別モデル（線形判別分析，ロジスティック回帰分析，分類木）と，2 つの外れ値除去法（MOA，LOFM）の組み合わせ，合計 6 通りについて判別精度の比較を行った．

まず，外れ値除去法の閾値（θ_{MOA}，θ_{LOFM}）を決定するために事前実験を行った．次に，モデル構築用のフィットデータセットに対して，事前実験で決定した閾値を用いて外れ値除去法を適用した．最後に，外れ値除去済みのフィットデータセットを用いて判別モデルを構築し，モデル評価用のテストデータセットを用いてその判別精度を求めた．実験の妥当性を確保するために，事前実験と本実験をそれぞれ 10 回ずつ繰り返した．また，外れ値除去法は，fault-prone モジュールの集合と，それ以外のモジュール（以降，not fault-prone モジュールと呼ぶ）の集合のそれぞれに対して適用した．

分類木の構築アルゴリズムには CART (Classification And Regression Trees) を用い，分岐の基準には Gini 係数を用いた [2]．また，LOF を算出する際に用いる k 最近傍の k については，Breunig らの決定方法 [3] に従ってあらかじめ実験を行い，30 から 50 とした．

3.2 データセット

実験には NASA/WVU IV&V Facility Metrics Data Program (MDP) が公開しているデータセット [16] の一つである，プロジェクト KC1 で記録されたデータセットを用いた．IV&V とは独立検証および妥当性確認のことで，ソフトウェアの信頼性を高めるために第三者機関がソフトウェアの試験や検証を行う活動のことである．KC1 には 2107 個のモジュールについての 20 種類のソースコードメトリクスと，IV&V 工程で得られた各モジュールに含まれる fault の有無の合計 21 種類の特性値が欠損無く記録されている．全モジュールのうち fault-prone モジュールは 325 個（約 15%）である．**表 1** に記録されているソースコードメトリクスを示す．

実験では，fault の有無を目的変数，20 種類のソースコードメトリクスを説明変数として判別モデルを構築する．判別モデルを構築する際には，データセットを二等分し，一方をフィットデータセット，もう一方をテストデータセットとする．

3.3 評価基準

Fault-prone モジュール判別モデルの評価基準として，再現率，適合率，F1 値 [7] を用いる．再現率とは fault-prone と判別されたモジュールのうち，

表 1: KC1 のソースコードメトリクス

	ソースコードメトリクス
m_1	総行数
m_2	実行行数（空行とコメント以外）
m_3	コメント行数
m_4	コメントを含むコード行数
m_5	分岐の数
m_6	Cyclomatic Complexity
m_7	Design Complexity
m_8	Essential Complexity
m_9	Halstead 尺度の Content
m_{10}	Halstead 尺度の Difficulty
m_{11}	Halstead 尺度の Effort
m_{12}	Halstead 尺度の Error Estimate
m_{13}	Halstead 尺度の Length
m_{14}	Halstead 尺度の Level
m_{15}	Halstead 尺度のプログラミング時間
m_{16}	Halstead 尺度の Volume
m_{17}	オペランドの数
m_{18}	オペレータの数
m_{19}	ユニークなオペランドの数
m_{20}	ユニークなオペレータの数

実際に fault を含んでいるモジュールの割合である．再現率の値域は $[0,1]$ であり，値が高いほど判別モデルの見逃しが少ないといえる．適合率とは fault を含んでいるモジュールのうち，正しく fault-prone と判別したモジュールの割合である．適合率の値域は $[0,1]$ であり，値が高いほど判別モデルが正確であるといえる．これら再現率と適合率は一方が高くなると，もう一方が低くなりやすいという関係にある．F1 値とは再現率と適合率の調和平均であり，その計算式は以下の通りである．

$$F1\text{ 値} = \frac{2 \times \text{再現率} \times \text{適合率}}{\text{再現率} + \text{適合率}} \tag{3}$$

F1 値の値域は $[0,1]$ であり，値が高いほど判別モデルの精度が高いといえる．

3.4 実験手順

3.4.1 閾値の決定手順

各外れ値除去法の閾値を決定するために事前実験を行う．事前実験では本実験で用いるフィットデータセットを用いて，判別モデルの構築および評価を行い，最も F1 値が高くなる閾値を求める．この閾値は，fault-prone モジュール判別モデルと外れ値除去法の全ての組み合わせの，6 通りそれぞれについて決定される．例として，外れ値除去法に MOA，fault-prone モジュール判別モデルに線形判別分析（LDA）を用いた場合の閾値 $\theta_{\text{MOA-LDA}}$ の決定手順を以下に示す．

Step 1. フィットデータセット fit を fit_a と fit_b の 2 つにランダムに二等分する．

Step 2. fit_a に対して MOA を適用し，外れ値を除去したデータセット fit'_a を作成する．

Step 3. fit'_a に基づいて，LDA により判別モデルを構築する．

Step 4. fit_b を用いて，構築した判別モデルの判別精度を求める．

Step 5. θ を変化させて，Step 2 から Step 4 を繰り返す．

Step 6. Step 1 から Step 5 を 10 回繰り返し，判別精度の平均値が最も高かった θ を本実験に用いる $\theta_{\text{MOA-LDA}}$ とする．

3.4.2 本実験の手順

3.4.1 項で決定された閾値を用いて，外れ値除去法と fault-prone モジュール判別モデルの全ての組み合わせ 6 通りの判別精度を求める．

例として，外れ値除去法に MOA，fault-prone モジュール判別モデルに LDA を用いた場合の実験の手順を以下に示す．

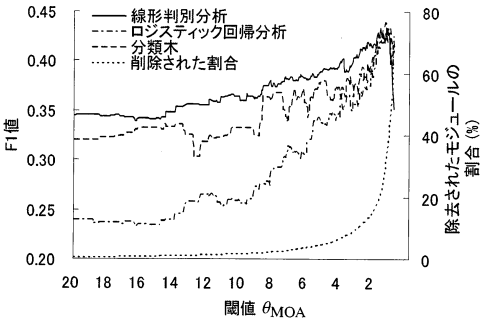
Step 1. データセットをランダムに二等分し，一方をフィットデータセット fit，もう一方をテストデータセット $test$ とする．

Step 2. fit に対して，事前実験で決定した $\theta_{\text{MOA-LDA}}$ を用いた MOA を適用し，外れ値を除去したデータセット fit' を作成する．

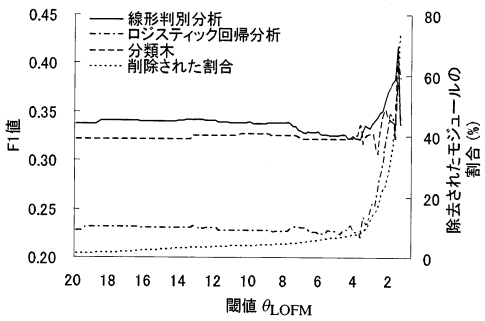
Step 3. fit' に基づいて，LDA により判別モデルを構築する．

Step 4. $test$ を用いて，構築した判別モデルの判別精度を求める．

Step 5. Step 1 から Step 4 を 10 回繰り返し，判別精度の平均値を求める．



(a) MOA



(b) LOFM

図 3: 閾値を変化させたときの判別モデルごとの F1 値と除去されたモジュールの割合

表 2: 判別モデルと外れ値除去法の組み合わせの判別精度

		LDA	LRA	CT
再現率	未適用	0.277	0.154	0.241
	MOA	**0.376**	**0.389**	**0.378**
	LOFM	0.276	**0.160**	**0.301**
適合率	未適用	0.557	0.652	0.483
	MOA	0.452	0.452	0.440
	LOFM	0.515	0.625	0.460
F1 値	未適用	0.367	0.247	0.317
	MOA	**0.409**	**0.416**	**0.402**
	LOFM	0.356	**0.252**	**0.357**

LDA：線形判別分析
LRA：ロジスティック回帰分析
CT：分類木

4 結果と考察

4.1 結果

4.1.1 事前実験の結果

3.4.1 項で行った事前実験の結果を**図 3** に示す．この図は閾値を変化させたときの判別モデルごとの F1 値の変化を示したものであり，**図 3(a)** と**図 3(b)** はそれぞれ MOA を用いた場合と LOFM を用いた場合の結果である．x 軸は外れ値除去法の閾値，y1 軸は F1 値，y2 軸は外れ値として除去されたモジュールの割合を表す．

MOA は全ての判別モデルで θ_{MOA} を小さくする，つまり除去するモジュールの割合を大きくするほど F1 値が向上した．$\theta_{MOA} = 0.9$ のときに，全体の 20.3% が外れ値として除去され，最も F1 値が向上した．

一方，LOFM は $\theta_{LOFM} = 1.2$ のときに，全体の 55.5% が外れ値として除去され，最も F1 値が向上したが，θ_{LOFM} が 8 から 4 の間では，F1 値が緩やかに低下した．

4.1.2 本実験の結果

3.4.2 項で行った本実験の結果を**表 2** に示す．表中の未適用とは，外れ値除去法を適用せずに判別モデルを構築した場合のことであり，太字は未適用と比べて判別精度が向上したものを表す．

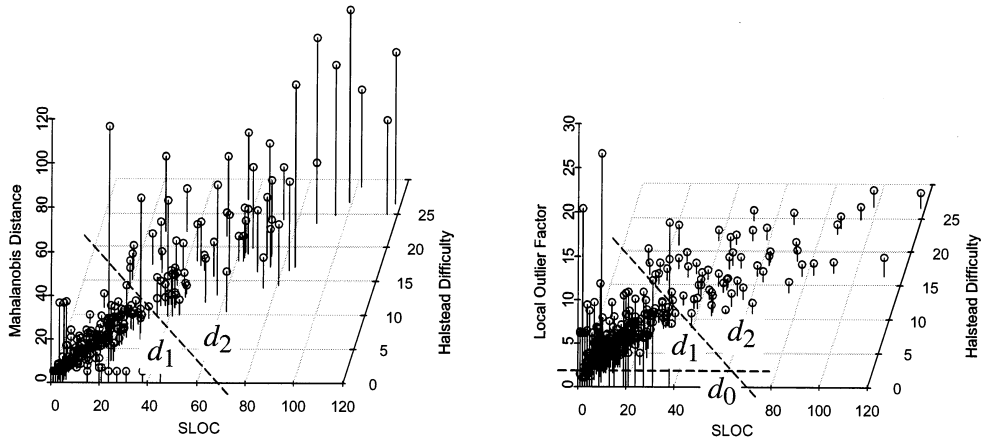
MOA を適用した場合は，全ての判別モデルで適合率は低下したものの，再現率が平均 0.16 向上し，F1 値は平均 0.10 向上した．特に，外れ値を除去しない場合に F1 値が最も低かった LRA と組み合わせた場合，再現率は 0.24，F1 値は 0.17 向上し，再現率と F1 値の向上幅とその値が最も高かった．

一方，LOFM を適用した場合は再現率の向上幅は平均 0.02 と MOA と比べて小さく，F1 値は平均 0.01 の向上であり大きな変化はなかった．

4.2 考察

本節では，外れ値除去法の適用の効果について，not fault-prone モジュールの分布傾向の変化を中心に考察する．not fault-prone モジュールに着目する理由は，not fault-prone モジュールが fault-prone モジュールに比べて 5.5 倍の数があり，判別モデルを構築する際に大きな影響を与えていると考えられるためである．

図 4 にフィットデータセット *fit* の not fault-prone モジュールの散布図を示す．**図 4(a)** と**図 4(b)** は



(a) MOA　　(b) LOFM

図 4: *fit* 内の not fault-prone モジュールの分布と領域の定義

それぞれ，MOA を用いた場合と LOFM を用いた場合の散布図である．x 軸は総行数，y 軸は Halstead Difficulty であり，z 軸は全ての説明変数を用いて算出された各モジュールの外れた度合い（マハラノビス距離，LOF）である．正の相関を持つ変数の組を x 軸と y 軸にとった場合，いずれの組でも似た傾向の散布図が得られるが，**図 4** では一例として，x 軸に総行数，y 軸に Halstead Difficulty をとった．図中の d_0，d_1，d_2 は，x 軸と y 軸の成す平面を破線で区切ったときの各領域のことをさす．

fit におけるモジュールの分布傾向は以下の通りであった．

- Not fault-prone モジュールの数は，fault-prone モジュールの数の 5.5 倍である．
- Not fault-prone モジュールの 80% は，総行数が少なく，かつ Halstead Difficulty が小さい領域 d_1 に分布している．
- Fault-prone モジュールは，総行数が多く，かつ Halstead Difficulty が大きい領域 d_2 に幅広く分布している．
- 領域 d_2 では not fault-prone モジュールの数と fault-prone モジュールの数がほぼ同じである．

・MOA の判別精度が向上した原因

図 4(a) を用いて not fault-prone モジュールのとるマハラノビス距離と判別境界の変化ついて説明する．Not fault-prone モジュールの重心は d_1 にあり，d_2 に属する not fault-prone モジュールはマハラノビス距離が大きな値をとる．このため，θ_{MOA} を小さくするほど，徐々に d_2 に属する not fault-prone モジュールが除去され，d_2 における fault-prone モジュールの占める割合が増える．よって，d_2 にあった判別境界が緩やかに d_1 の方へ移動する．結果として，θ_{MOA} を小さくするほど，構築される判別モデルが全体的に fault-prone モジュールと判別しやすくなり，大幅に再現率が向上し，F1 値が向上したものと考えられる．

これは，**図 3(a)** の F1 値が山なりに向上した結果と一致する．この山なりという点は，最も F1 値の向上する θ_{MOA} を決定する際に大きな利点となる．θ_{MOA} の見積もりを多少誤ったとしても，F1 値が大きく低下しないためである．ただし，事前実験で θ_{MOA} を小さめに見積もった場合（$\theta_{\mathrm{MOA}} = 0$ の近辺）は，急激に F1 値が落ちるが，θ_{MOA} を大きめに見積もることで，本実験では安定した F1 値の向上が見込まれる．

これらのことから，MOA は fault-prone モジュール判別モデルの判別精度向上に有効な手法であるといえる．

・LOFM の判別精度が向上しなかった原因

図 4(b) を用いて not fault-prone モジュールのとる LOF について説明する．d_1 のように，一定の密度でばらついている領域にある not fault-prone モ

ジュールのLOFは1に近い値をとる．また，d_2に属するnot fault-proneモジュールはd_1と比べて若干ばらついており，LOFは1以上の値をとる．一方，Halstead Difficultyが0に近いd_0に属するnot fault-proneモジュールは，密度が極端に高いd_1と比べて数が少なく，密度が小さい．このd_1とd_0の密度の差によってd_0に属するnot fault-proneモジュールのLOFは極端に高い値をとる．つまり各領域のLOFは$d_0 \gg d_2 > d_1 \fallingdotseq 1$という大小関係をとる．

ここで，θ_{LOFM}を小さくした場合，d_0のnot fault-proneモジュールが徐々に除去され，判別境界がMOAとは逆にd_2へ移動する．このため，外れ値除去法を適用しない場合に低かった再現率がさらに低下し，同時にF1値が低下したと考えられる．そして，$\theta_{\mathrm{LOFM}} = 1.2$のときに$d_2$のnot fault-proneモジュールが一気に除去され，一時的に判別境界がMOAと同様にd_1へ移動し，F1値が向上したと考えられる．

このθ_{LOFM}を大きくするほどF1値が低下し，ある局所的な範囲で急激に向上するという傾向は**図 3(b)**からも確認できる．このようなθ_{LOFM}とF1値の関係は，最もF1値を向上させるθ_{LOFM}を適切に決められなかった場合に判別精度の低下を招く．θ_{LOFM}の見積もりが少しでもずれると，高い判別精度を持つ判別モデルを構築できないためである．本稿での，本実験でLOFMを適用した場合にF1値が向上しなかった原因は，最適なθ_{LOFM}を見積りにずれがあったためであると考えられる．

これらのことから，LOFMはfault-proneモジュール判別モデルの判別精度向上に用いることは困難であるといえる．

5 関連研究

様々なデータマイニングの分野で，多数の外れ値除去法・除去法が提案されている．Victoriaらは既存の外れ値検出法についての大規模な調査を行った[8]．外れ値検出法を大きく3つの分類（統計モデル，ニューラルネット，機械学習）に分類した上で，それぞれについて細かく調査を行っている．ただし，各手法の考えや特徴についての調査であり，手法間での性能の良し悪しを比較するものではない．本研究は，判別モデルに対しての各外れ値除去法の適用の効果を比較したという点で異なる．

Khoshgoftaarらはfault-proneモジュール判別モデルの一つであるCase-Based Reasoning (CBR)の判別精度向上のために，Rule-Based Modeling (RBM)を用いた外れ値除去法を提案している[10]．この外れ値除去法はMOAやLOFMなどのような，外れた度合いを算出し閾値によって外れ値を決める手法とは異なり，ある標本が外れているかどうかを一意に決める手法である．Khoshgoftaarらは CBRとRBMを用いた外れ値除去法を組み合わせた結果，判別精度が向上することを確認している．ただし，他の既存の外れ値除去法と比べてどの程度有効であるかの評価は行っていない．本研究は，複数の判別モデルと複数の外れ値除去法の組み合わせの比較という点で異なる．

6 おわりに

本稿では，3つのfault-proneモジュール判別モデル（LDA，LRA，CT）と2つの外れ値除去法（MOA，LOFM）の組み合わせについて比較を行った．実験の結果，外れ値除去法を適用しない場合と比べて，MOAを適用した場合は，F1値が最小0.04，最大0.17，平均0.10向上した．また，MOAでは安定して判別精度の高いfault-proneモジュール判別モデルを構築できることがわかった．一方，LOFMを適用した場合は，F1値が最小-0.01，最大0.04，平均0.01変化し，判別モデルによってはF1値が向上したものの，その向上幅はMOAと比べて小さかった．また，LOFMでは高い判別精度を持つ判別モデルの構築が困難であるということがわかった．本稿はfault-proneモジュール判別モデルを構築する際の前処理としては，MOAが適していると結論付ける．

ただし，この結論は今回用いたデータセットKC1から結論付けられるものであり，今後，他のデータセットを用いて検証する必要がある．また，本稿で用いた外れ値除去法以外にも多数の手法が提案されており，これらの比較を行うという点も今後の重要な課題である．

謝辞

本研究の一部は，文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．

参考文献

[1] Bell, R. M.: Predicting the Location and Number of Faults in Large Software Systems, *IEEE Trans. Softw. Eng.*, Vol. 31, No. 4, pp. 340–355 (2005).

[2] Breiman, L., Friedman, J., Olshen, R. and Stone, C.: *Classification and Regression Trees*, Wadsworth, California (1984).

[3] Breunig, M. M., Kriegel, H. P., Ng, R. T. and Sander, J.: LOF: Identifying Density-based Local Outliers, *Proc. 19th ACM International Conference on Management of Data (SIGMOD'00)*, pp. 93–104 (2000).

[4] David W. Hosmer, S. L.: *Applied Logistic Regression*, Wiley, New York (1989).

[5] Fisher, R. A.: The Use of Multiple Measurements in Taxonomic Problems, *Annals Eugenics*, Vol. 7, No. Part II, pp. 179–188 (1936).

[6] Gray, A. R. and Macdonell, S. G.: Software Metrics Data Analysis-Exploring the Relative Performance of Some Commonly Used Modeling Techniques, *Empirical Software Engineering*, Vol. 4, No. 4, pp. 297–316 (1999).

[7] Herlocker, J. L., Konstan, J. A., Terveen, L. G. and Riedl, J. T.: Evaluating Collaborative Filtering Recommender Systems, *ACM Trans. Inf. Syst.*, Vol. 22, No. 1, pp. 5–53 (2004).

[8] Hodge, V. and Austin, J.: A Survey of Outlier Detection Methodologies, *Artifical Intelligence Rev.*, Vol. 22, No. 2, pp. 85–126 (2004).

[9] Jimenez-Marquez, S. A., Lacroix, C. and Thibault, J.: Statistical Data Validation Methods for Large Cheese Plant Database, *Journal of Dairy Science*, Vol. 85, No. 9, pp. 2081–2097 (2002).

[10] Khoshgoftaar, T. M., Liu, Y. and Seliya, N.: Genetic Programming-Based Decision Trees for Software Quality Classification, *Proc. 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*, pp. 374–383 (2003).

[11] Khoshgoftaar, T. M., Seliya, N. and Gao, K.: Detecting Noisy Instances with the Rule-based Classification Model, *Intelligent Data Analysis*, Vol. 9, No. 4, pp. 347–364 (2005).

[12] Knorr, E. M. and Ng, R. T.: Algorithms for Mining Distance-Based Outliers in Large Datasets, *Proc. 24th International Conference on Very Large Data Bases (VLDB'98)*, pp. 392–403 (1998).

[13] Munson, J. C. and Khoshgoftaar, T. M.: The Detection of Fault-Prone Programs, *IEEE Trans. Softw. Eng.*, Vol. 18, No. 5, pp. 423–433 (1992).

[14] Myers, G. J.: *The Art of Software Testing*, Wiley, New York (1979).

[15] Pighin, M. and Zamolo, R.: A Predictive Metric Based on Discriminant Statistical Analysis, *Proc. 19th International Conference on Software Engineering (ICSE'97)*, pp. 262–270 (1997).

[16] NASA/WVU IV&V Facility, Metrics Data Program: `http://mdp.ivv.nasa.gov/`.

[17] Schneidewind, N. F.: Investigation of Logistic Regression as a Discriminant of Software Quality, *Proc. 7th IEEE International Software Metrics Symposium (METRICS'01)*, pp. 328–337 (2001).

[18] Tang, J., Chen, Z., Fu, A. W. and Cheung, D.: A Robust Outlier Detection Scheme for Large Data Sets, *Proc. 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)* (2002).

[19] Yu, D., Sheikholeslami, G. and Zhang, A.: Findout: Finding Outliers in Very Large Datasets, *Knowledge and Information Systems*, Vol. 4, No. 4, pp. 387–412 (2002).