Fault-proneness モデルへのオーバーサンプリング法の適用

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
E-mail: {yasuta-k,akito-m,matumoto}@is.naist.jp

概要

ソフトウェア信頼性モデルの一つに，モジュールの複雑さや変更頻度に基づいてバグを含みやすいモジュールを判別するモデル（*fault-proneness model*）がある．モデルの種類として，線形判別モデル，ロジスティック回帰モデル，分類木，ニューラルネットなどがあるが，いずれのモデルにおいても，バグを含むモジュール数の少ない学習データセットを用いた場合に精度の高いモデルが構築できないという問題がある．本稿では，少数のクラスのデータを複製する手法であるオーバーサンプリング法を学習データセットに適用することで，予測精度の改善を試みた．評価実験の結果，オーバーサンプリング法を用いたモデルは，従来モデルと比較して予測精度（F 値）が向上した．

1. はじめに

ソフトウェアテストおよび保守において，バグを含みやすい傾向があるモジュール（ファイルやコンポーネント）を特定することは，ソフトウェアの信頼性を向上する上で重要である．そのために，モジュールの複雑さや変更頻度に基づいてバグを含みやすいモジュールを判別するモデル（バグ予測モデル）が多数提案されてきた．モデルの種類としては，線形判別分析 [11] [12]，ロジスティック回帰分析 [10]，ニューラルネットワーク [5] [14]，分類木 [7]，ベイジアンネットワーク [4] などが知られている．

バグ予測モデルでは，モジュールの特性値（プログラム行数やサイクロティック数など）を説明変数とし，目的変数であるバグの有無との関係を，モデルを用いて数学的に表す．モデルの構築は，過去のプロジェクトで計測されたデータセット（多数のモジュールの特性値とバグの有無を記録したもの）を用いて行われる．しかし目的変数であるバグの有無に極端な偏りがある場合，バグ予測モデルの精度が低下するといった問題がある [8]．例えば，バグを含まないモジュールが多いデータセットを用いてバグ予測モデルを構築すると，必要以上にバグを含まないモジュールと判別しがちなモデルが構築されてしまう．このようなバグの有無の偏りは，ソフトウェア開発において一般的に見られるため，バグの有無の偏りにかかわらず高い判別精度が得られるバグ予測モデルが必要である．

本稿では，この問題を解決するための手法として，サンプリング法 [2][9] を用いる．サンプリング法は，データセットに含まれるサンプルを増加もしくは減少させる前処理である．サンプリング法を用いることで，学習データセットの極端な偏りが解消され，判別するモデルの精度を向上させることができる．

我々は，バグの予測モデルの問題において，サンプリング法を適用した場合としない場合における線形判別分析，ロジスティック回帰分析，ニューラルネットワークと分類木の予測精度の比較実験を行った．実験対象は，NASA で公開されているプログラムの 1 つである約 43k ステップのモジュール群である [3]．モジュールの数は 2,108 個であり，それぞれ 21 種類の特性値（プロダクトメトリクス）が収集されている．実験では，バグが含まれているモジュールの判別精度を評価する．

2. Fault-proneness モデル

2.1 線形判別分析

線形判別分析は，説明変数の変動によって目的変数がどのような値をとるかを予測するための手法である．線形判別分析では，説明変数と目的変数の関係が以下の線形式により定義される．

$$y = \beta^T x + \alpha, \tag{1}$$

$$x \in \Re^n, y \in \Re, \alpha \in \Re, \beta \in \Re^n. \tag{2}$$

ここで，y は目的変数，x は説明変数となる入力ベクトルである．また，β は係数ベクトル，α は予測値と実測値の間の残差誤差である．例えば，入力ベクトル x が与えられた場合，式 (1) より，y の値が算出され，y は以下の条件式により，2 つのクラス $X1$ と $X2$ に分類される．本稿では，予測値が 0.5 以上の場合，モジュールがバグを含んでいると予測する．

$$x_i \in X_1, \quad if \quad y \geq 0.5, \tag{3}$$
$$x_i \in X_2, \quad if \quad y < 0.5. \tag{4}$$

2.2 ロジスティック回帰分析

ロジスティック回帰分析は，説明変数の変動によって目的変数が 2 値のどちらを取る確率が高いかを予測するための手法である．ロジスティック回帰分析では，説明変数と目的変数の関係が以下のロジスティック回帰式により定義される．

$$P(y|x) = \frac{1}{1 + e^{-(\beta^T x + \alpha)}} \tag{5}$$

ここで，y は 2 値の目的変数，x は説明変数となる入力ベクトル，β は係数ベクトル，α は係数値である．P は，入力ベクトル x に対して，y が 1 となる確率であり，本稿では P が 0.5 以上の値を取る場合，モジュールがバグを含んでいると予測する．

$$x_i \in X_1, \quad if \quad P \geq 0.5,$$
$$x_i \in X_2, \quad if \quad P < 0.5.$$

2.3 ニューラルネットワーク

ニューラルネットワークは，入力に対する値の重みや，ノード間のリンクの重みを学習データに最適化することで予測モデルを構築する手法である．ここでは本稿で用いた，入力層，中間層，出力層の 3 階層から構築されるニューラルネットワークについて説明する．

まず，入力層から説明変数の値が入力される．入力層は，入力された値を重み付けして，中間層に送る．中間層には，重み付けされた値の和が入力され，非線形変換し，重み付けして，出力層に送る．出力層には，重み付けされた値の和が入力され，出力として値を得る．本稿では，リンクの重みを決定するアルゴリズムには，誤差逆伝播法 [13] を用いた．出力された値は，式 (3)，(4) として判別される．学習回数をそれぞれ 10000，20000，30000，50000，100000 回に設定し仮実験した結果，モデルを構築する際の学習回数として最も予測精度の高かった 30000 回を採用した．

2.4 分類木

分類木とは，ある事項に対する説明変数の結果から、その事項の目的変数に関する結論を導く手法である．

分類木では，ルートのノードに説明変数の値が入力される．入力された値を評価し，その評価に基づいて分類先ノードを決定するという処理を再帰的に繰り返す．最終的にすべての適用事例は，終端のノードに送られ，その事項の目的変数に関する結論が導かれる．本稿では，分類木を構築するためのアルゴリズムとして，CART（Classification And Regression Trees） [7] を用いた．

3. サンプリング法

サンプリング法とは，予測モデルを構築するにあたって，データセットに含まれるサンプルを増加もしくは減少させる前処理である．サンプリング法には，多数派の

サンプルを削除するアンダーサンプリング法，少数派のサンプルを追加するオーバーサンプリング法という 2 つの手法が存在する．しかし，アンダーサンプリング法は予測モデルを構築するために有用なサンプルを削除してしまう可能性があり，オーバーサンプリング法は学習データセットが大きくなる上に予測モデルの汎用性が低下する過学習が発生する可能性がある．

バグが含まれるモジュールであるかの判別では，データセットでのバグを含むサンプルが極端に少ない場合がある [8]．アンダーサンプリング法によりバグを含まないモジュールのサンプルを減少させてしまうと，全体のデータセットにおけるサンプル数が極端に少なくなる恐れがあるため，オーバーサンプリング法を採用する．

本稿では，オーバーサンプリング法の 1 つである SMOTE(Synthetic Minority Over-sampling TEchnique)[2] を用いる．SMOTE は，データセットにおけるそれぞれのデータの k 最近傍を基に新たなデータを生成するアルゴリズムである．SMOTE では，少数派のクラス（今回の場合，バグが含まれているモジュール）のみ抽出した学習データ，オーバーサンプリングする割合 N，k 最近傍の値 k を入力し，オーバーサンプリングされた学習データを出力する．SMOTE は，入力されたデータセットの i 番目のデータ $(i = 1, ..., T)$ とそのデータの k 個の最近傍を基に n 個のデータを新たに生成する．そして，この処理を T 回繰り返す．ここで，T は入力されたデータセットの数，n は $N/100$ から求められる．最終的に，学習データは SMOTE によって新たに生成された少数派のクラスのデータセットと，多数派のクラス（今回の場合，バグが含まれていないモジュール）のデータセットを合わせたものである．

本稿では，Chawla ら [2] が用いた $k = 5$ を採用する．また，k 最近傍を求めるための類似度指標として，それぞれのモジュール間における特徴ベクトルの成す角から類似性を求める Cosine Similarity 法 [1] を用いる．そして，オーバーサンプリングする割合 N は，以下の式により求める．

$$N = \frac{\text{バグを含まないモジュールの個数}}{\text{バグを含むモジュールの個数}} \tag{6}$$

式 (6) でオーバーサンプリングする割合 N を求める理由は，バグを含むモジュールとバグを含まないモジュールのデータの割合を均等にするためである．

4. 実験

4.1. データセット

実験に用いたデータは，NASA IV&V facility Metrics Data Program（MDP） [3] でリポジトリに蓄積された特性値である．NASA IV&V facility MDP とは，NASA の独立検証機構である IV&V facility が過去のプロジェクトにおける成果物の一部をリポジトリに蓄積し，一般に公開している活動である．本稿では，NASA IV&V facility Metrics Data Program（MDP）が公開している KC1 というプロジェクトの特性値を利用した．KC1 は，ある接地装置の内部にある CSCI(Computer Software Configuration Item) の開発プロジェクトである．ソフトウェアは C++で開発され，SLOC は 43k ステップである．また，バグを含まないモジュールを 1,782 個，バグを 1 つ以上含むモジュールを 326 個，合計 2,108 個のモジュールを含んでいる．目的変数は，バグを含まないモジュールを 0 とし，バグを含むモジュールを 1 としている．説明変数に用いた種類の各特性値の統計量を表 1 に示す．表 1 に示すように，説明変数には，SLOC や Cyclomatic 数といったモジュールの複雑さに関する特性値を用い，それぞれのモジュールにおいて特性値を収集した．

4.2. 評価基準

評価基準として，再現率，適合率と F 値 [6] を用いた．再現率は，すべてのバグを含むモジュールのうち，バグを含むモジュールと予測した割合を示し，式（7）として定義される．適合率は，バグモジュールと予測したうち，実際にバグモジュールである割合を示し，式（8）として定義される．F 値は，適合率と再現率の調和平均で与えられ，式（9）として定義される．それぞれの評価基準は，値が大きいほど見積もり精度が高いことを表し，表 4.2 で示された予測したモジュールと実際のモジュールの関係を用いると，以下のように定義される．

再現率

$$\text{再現率} = \frac{n_{22}}{n_{21} + n_{22}} \tag{7}$$

表 1. 各特性値の統計量

	平均値	中央値	標準偏差	最大値	最小値
LOC_BLANK	1.76	0.00	3.86	58.00	0.00
BRANCH_COUNT	4.67	1.00	7.80	89.00	1.00
LOC_CODE_AND_COMMENT	0.13	0.00	0.70	12.00	0.00
LOC_COMMENTS	0.95	0.00	3.09	44.00	0.00
CYCLOMATIC_COMPLEXITY	2.84	1.00	3.90	45.00	1.00
DESIGN_COMPLEXITY	2.55	1.00	3.38	45.00	1.00
ESSENTIAL_COMPLEXITY	1.67	1.00	2.20	26.00	1.00
LOC_EXECUTABLE	14.54	5.00	24.20	262.00	0.00
HALSTEAD_CONTENT	21.26	14.40	21.50	193.06	0.00
HALSTEAD_DIFFICULTY	6.78	3.50	7.87	53.75	0.00
HALSTEAD_EFFORT	5247.36	213.97	17452.51	324803.51	0.00
HALSTEAD_ERROR EST	0.09	0.02	0.17	2.64	0.00
HALSTEAD_LENGTH	49.88	16.00	83.63	1106.00	0.00
HALSTEAD_LEVEL	0.32	0.20	0.32	2.00	0.00
HALSTEAD_PROG_TIME	291.52	11.89	969.58	18044.64	0.00
HALSTEAD_VOLUME	258.94	57.06	516.50	7918.82	0.00
NUM_OPERANDS	18.80	6.00	32.08	428.00	0.00
NUM_OPERATORS	31.07	10.00	51.79	678.00	0.00
NUM_UNIQUE_OPERANDS	9.55	5.00	12.20	120.00	0.00
NUM_UNIQUE_OPERATORS	7.64	6.00	5.73	37.00	0.00
LOC_TOTAL	20.39	9.00	29.76	288.00	1.00

適合率

$$\text{適合率} = \frac{n_{22}}{n_{12} + n_{22}} \tag{8}$$

F 値

$$F\text{ 値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}} \tag{9}$$

表 2. モデルによる予測結果の個数

	バグが無いと予測	バグが有りと予測
実際にバグ無し	n_{11}	n_{12}
実際にバグ有り	n_{21}	n_{22}

4.3. 実験手順

学習データにオーバーサンプリング法を適用しない場合と適用した場合のそれぞれにおいてモデルを構築し，を 4.2 節で説明した指標を用いて精度を評価した．

学習データに対してオーバーサンプリングを適用しない場合の実験手順を以下に示す．

1. 4.1 節で述べたデータセットを無作為に二等分し，予測モデルを構築する学習データと，学習データを用いて予測を行うテストデータとする．
2. 学習データを用いて，バグ予測モデルを構築する．
3. テストデータにおけるバグの有無は未知と仮定し，構築したモデルを用いてバグモジュール予測を行う．
4. 予測したバグの有無と実際のバグの有無から各評価

基準におけるそれぞれのモデルの精度を求める．

オーバーサンプリング法を適用する場合の実験手順を以下に示す．

1. 4.1 節で述べたデータセットを学習データとテストデータに無作為に二等分する．
2. オーバーサンプリング法を適用する割合 N を式 6 より求める．例えば，バグを含んでいるモジュールが 200 個，バグを含んでいないモジュールが 700 個の場合，$N = 700/20 = 7/2$ とする．
3. 手順 2 で求めた N におけるオーバーサンプリング法を適用した学習データを用いて，バグ予測モデルを構築する．
4. 学習データに対して，オーバーサンプリング法を適用する．
5. テストデータにおけるバグの有無は未知と仮定し，構築したモデルを用いてバグモジュール予測を行う．
6. 予測したバグの有無と実際のバグの有無から各評価基準におけるそれぞれのモデルの精度を求める．

5. 結果と考察

学習データに対してオーバーサンプリング法を適用しない場合と適用する場合のモデルの F 値に関する箱ひげ図を図 1 に示す．中央値，最大値，最小値，第三四分位数と第一四分位数において，学習データにオーバーサンプリング法を適用した場合，それぞれのモデルの F 値は適用しない場合のモデルと比較して高い精度であった．例えば中央値の場合，線形判別分析は 0.28 から 0.44 に，ロジスティック回帰分析は 0.25 から 0.41 に，ニューラルネットワークは 0.29 から 0.41 に，分類木は 0.30 から 0.42 に向上した．

また，オーバーサンプリング法を適用しない場合と適用する場合の再現率に関する箱ひげ図を図 2 に示す．F 値と同様に学習データにオーバーサンプリング法を適用した場合のそれぞれのモデルの再現率は，中央値，最大値，最小値，第三四分位数と第一四分位数において，適用しない場合のモデルと比較して高い精度であった．例えば中央値の場合，線形判別分析は 0.18 から 0.79 に，ロジスティック回帰分析は 0.17 から 0.77 に，ニューラルネットワークは 0.18 から 0.81 に，分類木は 0.21 から 0.63 に向上した．

そして，オーバーサンプリング法を適用しない場合と適用する場合の適合率に関する箱ひげ図を図 3 に示す．F 値，再現率と比較して，学習データにオーバーサンプリング法を適用した場合のそれぞれのモデルの適合率は，中央値，最大値，最小値，第三四分位数と第一四分位数において，適用しない場合のモデルと比較して低い精度であった．例えば中央値の場合，線形判別分析は 0.62 から 0.30 に，ロジスティック回帰分析は 0.56 から 0.28 に，ニューラルネットワークは 0.63 から 0.28 に，分類木は 0.55 から 0.30 に低下した．

学習データにオーバーサンプリング法を適用した場合，各バグ予測モデルにおける適合率は低下したものの，再現率は大幅に向上した．この結果から，それぞれのバグ予測モデルにおいて適合率が若干精度が低いものの，実際のバグモジュールをバグが含まれていないと誤って判別されることが減少するという利点を示していると考えられる．また，再現率と適合率の調和平均を表す F 値もまた向上している．このことから，オーバーサンプリング法を学習データに用いることで，それぞれのバグ予測モデルにおいて精度が向上していると考えられる．

6. おわりに

本稿では，NASA IV&V facility Metrics Data Program（MDP）が公開するプロジェクトのデータセットを用いて，正例（今回の場合，バグが含まれているモジュール）が極端に少なく，負例（今回の場合，バグが含まれていないモジュール）が多いデータセットに対してオーバーサンプリング法を適用した場合の予測精度の高さを実験的に評価した．

今後，再現率の高さを保ちつつ，適合率の低下を抑える方法について検討する必要がある．例えば，アプローチとして極端に少ない正例，多い負例それぞれにサンプリング法を適用する，つまり，極端に少ない正例に対してオーバーサンプリング法を適用し，多い負例に対してサンプリング法を適用する手法などを提案していく予定

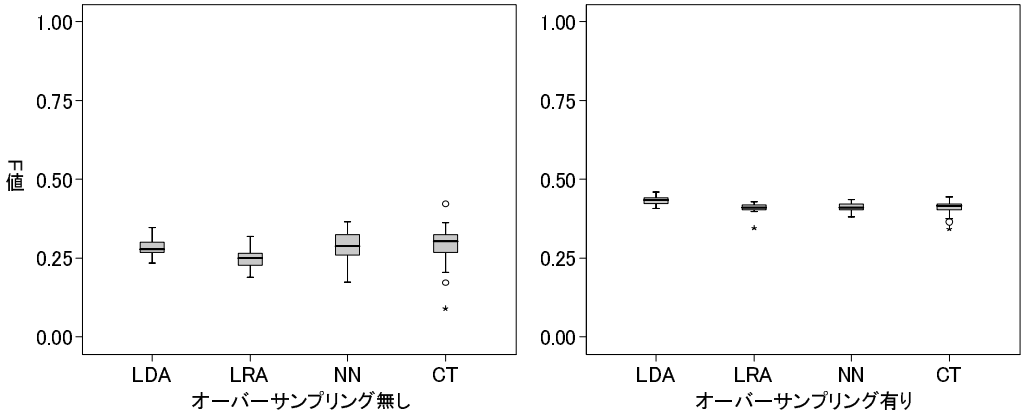

図 1. 各バグ予測モデルにおける F 値

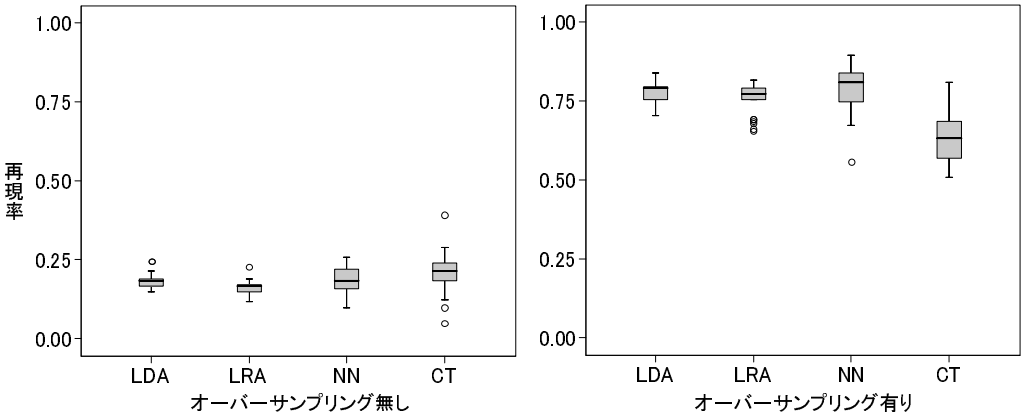

図 2. 各バグ予測モデルにおける再現率

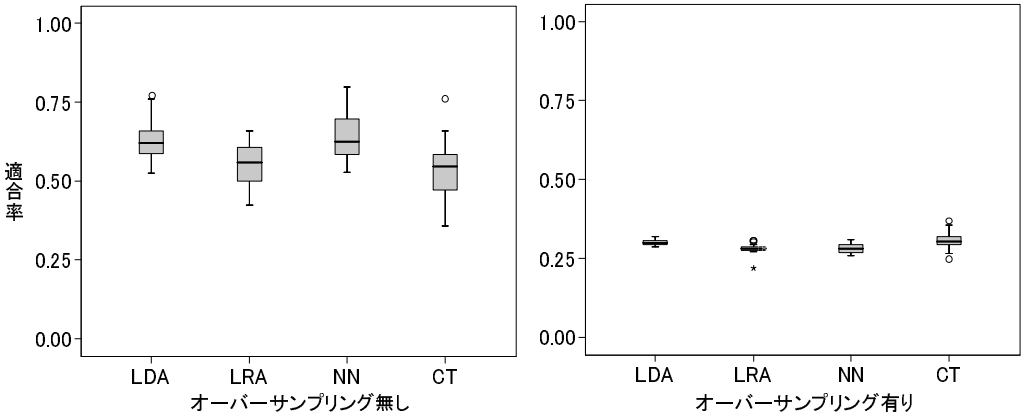

図 3. 各バグ予測モデルにおける適合率

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] S. Badrul, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *In Proc. Int'l World Wide Web Conference*, pp.285–295, Hong Kong, China, May 2001.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and P. W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research (JAIR)*, 16:321–357, 2002.

[3] N. I. Facility. Metrics data program. http://mdp.ivv.nasa.gov/index.html.

[4] N. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Trans on Software Engineering*, 26(5):675–689, Sep. 1999.

[5] A. R. Gray and S. G. MacDonell. Software metrics data analysis - exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, 4:297–316, 1999.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans on Information Systems*, 22(1):5–53, 2004.

[7] T. M. Khoshgoftaar and E. B. Allen. Modeling software quality with classification trees. *Recent Advances in Reliability and Quality Engineering*. World Scientific, pp.247–270, Singapore, 1999.

[8] T. M. Khoshgoftaar, K. Gao, and R. M. Szabo. An application of zero-inflated poisson regression for software fault prediction. In Proc. 12th Int'l Symposium on Software Reliability Engineering(ISSRE'01), pp.66–73, Hong Kong, China, Nov. 2001.

[9] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. *In Proc. 14th Int'l Conf. on Machine Learning*, pp.179–186, Tennessee, USA, 1997.

[10] J. C. Munson and T. M. Khoshgoftaar. The detection of fault-prone programs. *IEEE Trans. on Software Engineering*, 18(5):423–433, 1992.

[11] N. Ohlsson and H. Alberg. Predicting fault-prone software modules in telephone switches. *IEEE Trans on Software Engineering*, 22(12):886–894, 1996.

[12] M. Pighin and R. Zamolo. A predictive metric based on statistical analysis. *In Proc. Int'l Conf. on Software Engineering (ICSE'97)*, pp.262–270, Boston, USA, 1997.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[14] S. Takabayashi, A. Monden, S.Sato , K. Matsumoto, K. Inoue, and K. Torii. The detection of fault-prone program using a neural network. *In Proc. Int'l Symposium on Future Software Technology(ISFST'99)*, pp.81–86, Nanjing, China, Oct. 1999.