SVMに基づくソフトウェア信頼性モデルの定量的評価

Empirical Evaluation of Fault-Proneness Model based on SVM

亀井 靖高*　門田 暁人†　松本 健一‡

あらまし
SVM（Support Vector Machine）は，パターン認識の手法の一手法であり，文字認識などの研究分野において高い精度で適用できたという事例が報告されている．本稿では，SVMを用いてバグが含まれているモジュールを予測するための信頼性モデルを構築し，その性能評価を実験により行った．評価実験における比較対象として，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，分類木を用いた．実験では，あるソフトウェア企業で開発されたCOBOLプログラムの514個のモジュールを対象とした．実験の結果，SVMはF1値の中央値が0.65であり，他のモデルの中央値（0.57，0.54，0.59，0.61）と比べて高い精度を示した．

1　はじめに

ソフトウェアテストおよび保守において，バグを含みやすいモジュールを特定することは，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，従来，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が多数提案されている．モデルの種類としては，線形判別分析 [10] [11]，ロジスティック回帰分析 [9]，ニューラルネットワーク [6] [14]，分類木 [8]，ベイジアンネットワーク [5] などが知られている．

本稿では，高い予測精度（判別精度）の実現を目指して，SVM（Support Vector Machine）をモデル構築のための手法として用いる．SVMは，ベクトルで表される多数のサンプルを，２つのクラスに分類するための非線形のパターン認識手法の一手法であり [4]，ソフトウェアの各モジュールを複数のメトリクスの系列（ベクトル）として表すことで，バグモジュール予測への適用が可能である．SVMの仕組みを図１に示す．図１の左側にある入力空間上に，２つのクラスのうちどちらかに属する入力ベクトルが配置されている．SVMでは，入力ベクトルに対して非線形変換を行い，図１の右側の高次元空間に写像する．そして，その高次元空間上で２つのクラスに判別する分離平面を求める．

ニューラルネットなどの従来の非線形モデルは局所解に陥る可能性があるが，SVMは局所解に陥らないことが特長である．またSVMは，ノイズの多いデータセットに対して頑健である．このことから，SVMは文字認識をはじめとする幅広い研究分野で利用されている [13]．

先行研究として，Xingら [15] はSVMを用いてあるソフトウェアのバグモジュールの予測を行い，線形判別分析や分類木と比較して予測精度が高いことを実験的に示した．バグモジュール予測においては，従来，ロジスティック回帰分析がよく用いられており，予測精度も高いという事例が報告されている [9]．また，非線形モデルとしては，ニューラルネットも従来よく用いられている [6] [14]．ただし，Xingらの研究では，ロジスティック回帰分析やニューラルネットワークとの比較を行っていない．

そこで本稿では，SVMと代表的なバグ予測モデルとの予測精度を比較することを

*Yasutaka Kamei, 奈良先端科学技術大学院大学
†Akito Monden, 奈良先端科学技術大学院大学
‡Ken-ichi Matsumoto, 奈良先端科学技術大学院大学

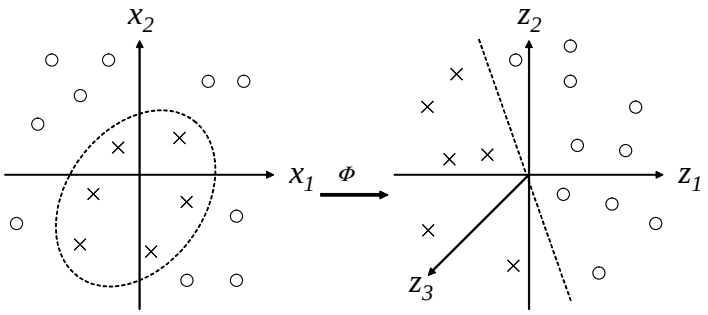

図 1　SVM の基礎概念

目的とする．ロジスティック回帰分析やニューラルネットワークに加えて，線形判別分析や分類木を代表的なバグ予測モデルとして用いて比較実験を行う．実験対象は，あるソフトウェア企業で開発された約 300K ステップのプログラムである．モジュールの数は 514 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）を収集した．実験では，テストにおいてバグが見つかったモジュールの判別精度を評価する．

以降，2 章で実験について説明した後，3 章で実験の結果とその考察について述べる．そして，4 章でまとめを述べる．

2　実験

2.1　データセット

実験で用いたデータセットは，あるソフトウェア企業が COBOL を用いて開発したプログラムの特性値とバグの有無である．ソフトウェアの大きさは，SLOC で約 300k である．そして，バグを含まないモジュールを 277 個，バグを 1 つ以上含むモジュールを 237 個，合計 514 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．これらのバグの有無は，テスト工程中（ユニットテスト，統合テスト，システムテスト）に収集されたバグの記録を基としている．説明変数に用いた 21 種類の各特性値のバグを含まないモジュールの統計量を表 1 に，バグを含むモジュールの統計量を表 2 に示す．表 1，2 に示すように，説明変数には，SLOC やサイクロマティック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つき特性値を収集した．

2.2　評価基準

評価基準として，適合率，再現率と F1 値を用いた．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す．再現率は，すべてのバグモジュールのうち，バグモジュールと予測した割合を示す．F1 値は，適合率と再現率の調和平均で与えられ，式（1）として定義される [7]．それぞれの評価基準は，値が大きいほど予測精度が高いことを表す．

$$F1\text{ 値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}} \tag{1}$$

2.3　実験手順

2.1 節で説明したデータセットを用いて，以下の手順で実験を行った．

表 1　バグを含まないモジュールの各特性値の統計量

	平均値	中央値	分散	最大値	最小値
Source lines of code（SLOC）	411.61	330.00	416.08	5487.00	90.00
コメントの総数	168.84	154.00	124.94	1256.00	37.00
SLOC ごとのコメントの総数	0.57	0.56	0.08	0.84	0.41
新規追加された SLOC	213.90	168.00	142.55	936.00	4.00
再利用された SLOC	27.55	0.00	279.93	4132.00	0.00
テストの総数	20.44	15.00	16.21	152.00	4.00
サイクロマティック数	23.41	16.00	34.36	432.00	2.00
SLOC ごとのサイクロマティック数	2.87	2.25	2.10	15.88	1.25
ループの総数	2.21	2.00	2.95	37.00	0.00
ネストの最大の深さ	5.38	5.00	1.32	12.00	4.00
ネストの平均の深さ	3.62	3.58	0.38	5.12	2.55
ジャンプノードの総数	3.85	1.00	11.26	137.00	0.00
宣言された変数の総数	45.42	25.00	65.91	764.00	4.00
参照された変数の総数	71.85	48.00	110.93	1532.00	4.00
代入された変数の総数	54.60	36.00	94.32	1392.00	4.00
内部呼出しの総数	8.77	8.00	7.13	66.00	0.00
外部呼出しの総数	3.13	1.00	6.85	78.00	0.00
代入されたグローバル変数の総数	28.00	18.00	40.25	528.00	0.00
参照されたグローバル変数の総数	40.52	26.00	55.42	660.00	1.00
マクロの総数	4.28	4.00	2.38	24.00	0.00
パラメータの総数	4.09	1.00	6.80	33.00	0.00

表 2　バグを含むモジュールの各特性値の統計量

	平均値	中央値	分散	最大値	最小値
Source lines of code（SLOC）	707.14	505.00	602.93	4232.00	164.00
コメントの総数	261.14	194.00	193.40	1307.00	75.00
SLOC ごとのコメントの総数	0.61	0.62	0.08	0.84	0.42
新規追加された SLOC	411.46	300.00	389.57	3185.00	13.00
再利用された SLOC	32.19	0.00	204.23	2106.00	0.00
テストの総数	54.22	28.00	84.58	967.00	3.00
サイクロマティック数	42.71	29.00	41.28	279.00	7.00
SLOC ごとのサイクロマティック数	3.75	2.87	3.20	22.50	1.33
ループの総数	3.89	2.00	4.57	28.00	0.00
ネストの最大の深さ	6.29	6.00	1.48	10.00	4.00
ネストの平均の深さ	3.78	3.68	0.48	5.31	2.74
ジャンプノードの総数	4.37	2.00	6.60	46.00	0.00
宣言された変数の総数	94.83	50.00	121.14	697.00	4.00
参照された変数の総数	152.44	98.00	182.60	1662.00	12.00
代入された変数の総数	120.49	79.00	143.37	1241.00	5.00
内部呼出しの総数	16.11	12.00	14.79	108.00	0.00
外部呼出しの総数	5.53	2.00	8.92	46.00	0.00
代入されたグローバル変数の総数	61.20	45.00	65.75	507.00	1.00
参照されたグローバル変数の総数	75.54	50.00	76.84	559.00	2.00
マクロの総数	6.08	5.00	3.83	23.00	1.00
パラメータの総数	4.91	1.00	7.52	25.00	0.00

1. 2.1 節で述べたデータセットを無作為に 257 件ずつに二等分し，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実際に予測を行うテストデータとする．
2. 学習データを使って一つのテストデータに対して SVM を用いた手法によりバグモジュール予測を行い，各評価基準を求める．テストデータのバグの有無は未知としてバグモジュール予測を行う．SVM の学習では一般的にカーネル関数が用いられ，本稿ではカーネル関数として RBF(Radial Basis Function) カーネル関数を用いた．RBF カーネル関数を用いたモデルは他のカーネル関数を用いたモデルと比較してパラメータの設定が容易である点 [2] と RBF カーネルを用いたバグモジュール予測モデルの精度が高かった [15] ためである．
3. 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，分類木でも同様にバグモジュール予測を行う．ニューラルネットワークを構築するためのアルゴリズムには，誤差逆伝播法 [12] を用いた．仮実験の結果最も精度が高かったため，学習回数を 30000 回に，中間層を 3 個に設定し，モデルを構築した．分類木を構築するためのアルゴリズムには，CART（Classification And Regression Trees） [8] を用いた．

手順 1 から 3 を 1 回の試行とし，25 回の試行を行った．

3 結果と考察

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，分類木，SVM で構築したモデルの F1 値の箱ひげ図を図 2 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，SVM の F1 値は他のモデルより高かった．例えばそれぞれのモデルの最小値は，SVM が 0.52 であり，線形判別分析が 0.46 であり，ロジスティック回帰分析が 0.45 であり，ニューラルネットワークが 0.51 であり，分類木は 0.48 であった．また，それぞれのモデルの最大値は，SVM が 0.72 であり，線形判別分析が 0.64 であり，ロジスティック回帰分析が 0.61 であり，ニューラルネットワークが 0.69 であり，分類木は 0.70 であった．

それぞれのモデルにおける適合率と再現率の箱ひげ図を図 3，図 4 に示す．SVM の再現率の中央値は，二番目に高い分類木と 0.06 の差があり，最も低いロジスティック回帰と 0.27 の差があった．一方で，SVM の適合率の中央値は，最も高かった線形判別分析と 0.03 の差のみであり，わずかに差があるのみであった．この結果は，

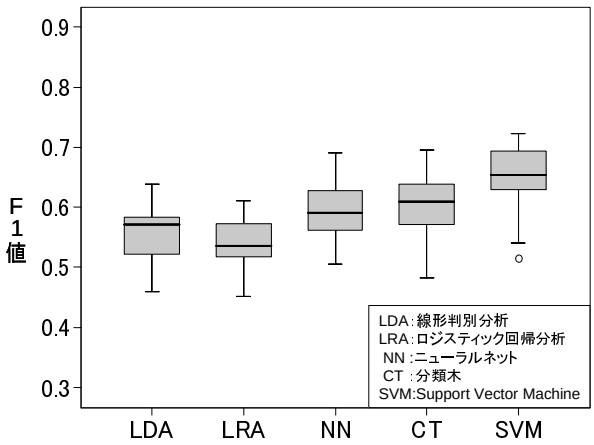

図 2　各モデルにおける F1 値

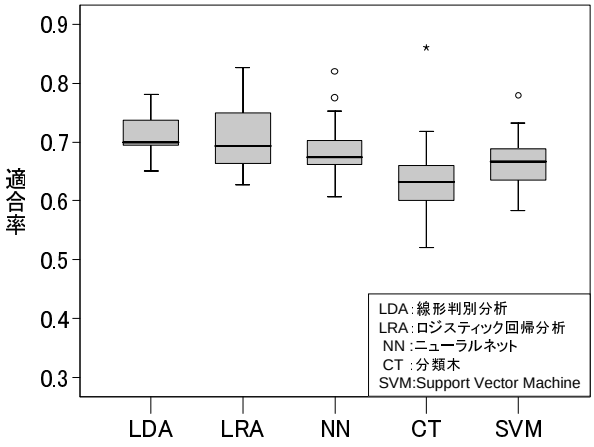

図 3　各モデルにおける適合率

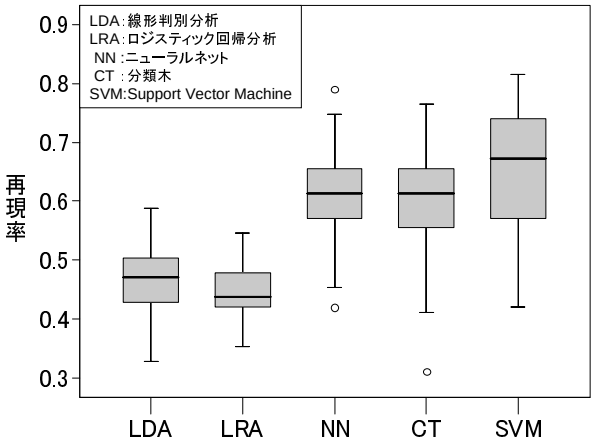

図 4　各モデルにおける再現率

SVM によるバグモジュール予測が適合率において若干精度が低いものの，実際のバグモジュールを見落とすことを避けるという大きな利点を示していると考えられる．

ロジスティック回帰分析は，線形判別分析よりも精度が低く，実験の中で最も精度が低い結果となった．ニューラルネットワークと分類木は，線形判別分析やロジスティック回帰分析よりも F1 値と再現率が高かった．しかしながら，分類木の適合率は，すべてのモデルの中で最も低かった．

SVM を用いたモデルの精度，ロバスト性の高さは，過学習を避けることができるという特徴と，学習データにノイズが混じっている際にも精度の低下を防ぐソフトマージンに起因しているものと考えられる．ノイズとは，例えばソースコードが長く，コメントが少なく，ネストが深いにもモジュールであるにもかかわらず，バグが含まれないような個体である．

4　おわりに

本稿では，あるソフトウェア企業で開発されたプログラムのデータセットを用いて，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，分類木と

いった従来の予測モデルと，SVM を用いた予測モデルの精度の高さを実験的に評価した．結果として，SVM は，従来のモデルと比較して，F1 値において最も高い精度であった．

今後，バグが含まれているモジュールの割合が極端に低いデータセットを用いて，従来の予測モデルと SVM を用いた予測モデルを評価する予定である．一般的に，正例（今回の場合，バグモジュール）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）が多い場合，SVM を用いた予測モデルの精度が極端に低下するという問題が報告されている [1] ．そこで，SVM で予測モデルを構築する前に，バグの含有が不均衡な学習データに対して，リサンプリング法 [1] [3] を適用する予定である．

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] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. *Proc. 15th European Conf. on Machine Learning (ECML'04)*, pp. 39–50, 2004.

[2] Jun Cai and Li Yanda. Classification of nuclear receptor subfamilies with rbf kernel in support vector machine. *Proc. Int'l Symposium on Neural Networks (ISNN'05)*, pp. 680–685, Chongqing, China, May, 2005.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and Philip W Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, Vol. 16, pp. 321–357, 2002.

[4] Corinna Cortes and Vladimir N Vapnik. Support vector networks. *Machine Learning*, Vol. 20, pp. 273–297, 1995.

[5] Norman Fenton and Martin Neil. A critique of software defect prediction models. *IEEE Trans. on Software Engineering*, Vol. 26, No. 5, pp. 675–689, Sep. 1999.

[6] Andrew R Gray and Stephen G MacDonell. Software metrics data analysis - exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, Vol. 4, pp. 297–316, 1999.

[7] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Systems*, Vol. 22, No. 1, pp. 5–53, 2004.

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

[9] John C Munson and Taghi M Khoshgoftaar. The detection of fault-prone programs. *IEEE Trans. on Software Engineering*, Vol. 18, No. 5, pp. 423–433, 1992.

[10] Niclas Ohlsson and Hans Alberg. Predicting fault-prone software modules in telephone switches. *IEEE Trans. on Software Engineering*, Vol. 22, No. 12, pp. 886–894, 1996.

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

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, Vol. 323, pp. 533–536, 1986.

[13] Bernhard Schölkopf, Kah-Kay Sung, Chris JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir N Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. on Signal Processing*, Vol. 45, No. 11, pp. 2758–2765, 1997.

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

[15] Fei Xing, Ping Guo, and Michael R Lyu. A novel method for early software quality prediction based on support vector machine. *Proc. 16th Int'l Symposium on Software Reliability Engineering (ISSRE'05)*, pp. 213–222, 2005.