情報処理学会論文誌　Vol. 52　No. 2　792–800 (Feb. 2011)

fault-prone モジュール判別手法の予測結果の統合方法について

天　嵜　聡　介[†1]

ソフトウェア開発では品質を維持しつつ，コストを削減し開発期間を短縮することが求められている．これらの要求を満たすことを目的として，不具合を含む可能性が高いモジュールを早期に判別する fault-prone モジュール判別手法の研究がさかんに行われている．しかし，同程度の精度を持つ fault-prone モジュール判別手法であっても，同じモジュールに対する予測が一致しない場合がある．この状況では複数の予測結果から専門家が総合的に判断を下す方法が考えられるが，fault-prone モジュール判別手法の利点である高速性・客観性が損なわれる．本研究では複数の fault-prone モジュール判別手法の予測結果を総合的に判断する方法としてアンサンブル学習手法に着目し，その有効性を評価した．実際の開発プロジェクトで収集された 5 つのデータセットを用いた評価実験の結果，4 つのデータセットでアンサンブル学習手法の一種である Stacking の有効性が確認できた．この成果は fault-prone モジュール判別手法の実用性向上に寄与すると考えられる．

On the Method of Synthesizing Multiple Fault-prone Module Detection Methods

Sousuke Amasaki[†1]

Software development companies need a method for reducing cost and shortening duration of a software development project while keeping quality of a product. Fault-prone module detection is one of methods expected to make a testing phase efficient. This approach has advantages of high speed and objective decision. Many fault-prone module detection methods were thus proposed so far. Their predictions on a module are not always identical even if their performance are equivalent. Thus, experts' decision based on their predictions may be suitable for this situation. However, this work-around loses advantages of fault-prone module detection methods. This paper evaluated effectiveness of ensemble-learning methods for decision methods based on predictions of multiple fault-prone module detection methods. As an ensemble learning method, Stacking and Voting were evaluated with five NASA datasets and four conventional fault-prone module detection methods. The results showed Stacking with an appropriate meta-learner could perform appropriate decisions for four of five datasets. The contribution of this paper is that usefulness in practice of synthesizing predictions of multiple fault-prone module detection methods could be suggested.

1. はじめに

ソフトウェア開発プロジェクトにおいてテスト工程は品質を確保するための重要な工程である．よってテスト工程には十分にコストと期間を費やす必要がある．しかし一方ではコスト削減および開発期間の短縮化の要求は高まっている．これらの相反する要求に応えるためにはテスト工程の効率化が重要となる．

品質を確保しつつテスト工程を効率化するため，不具合を含む可能性が高いモジュールへ優先的にテストの工数を割り当てる戦略が考えられる．この戦略を支援するために，テスト工程の早期段階において不具合を含む可能性が高いモジュールを判別する手法の研究がこれまでさかんに行われている[1)–4)]．このような手法を fault-prone モジュール判別手法と呼ぶ．fault-prone モジュール判別手法は線形回帰モデルのように統計学に基づく手法や機械学習，データマイニングに基づいた手法など様々な手法を用いたものが提案されている．これらの fault-prone モジュール判別手法には定量的なデータに基づいて高速に不具合を含む可能性を判別できるという利点がある．

開発プロジェクトで収集される定量的なデータは開発環境や開発するシステムのドメインの違いによって異なる特徴を示す[4)]．また，前述のように fault-prone モジュール判別手法は様々な手法に基づいている．このため，あるデータセットに対して同程度の判別精度を示す手法であっても，個々のモジュールに対する予測は異なる場合がある．この場合の実用上の解決策の 1 つは，複数の判別手法の結果から専門家が総合的な判断を下すことである．しかし，fault-prone モジュール判別の場合は 1 プロジェクトあたりで判別対象となるモジュールが多い．そのため，専門家がすべてのモジュールに対して複数の予測結果から総合的な判断を下すことは効率上の問題が生じる．また，定量的データに基づいた高速な判別という fault-prone モジュール判別手法の利点が損なわれる．利点を考慮すると，fault-prone モジュール判別手法と同様のアプローチで複数の予測結果から総合的な判断を下すことが望ましい．

†1 岡山県立大学
Okayama Prefectural University

© 2011 Information Processing Society of Japan

そこで本研究では,「複数の fault-prone モジュール判別手法の予測結果に基づいて総合的な判断を下す判別手法が構築可能であるか」について検証した．本研究では総合的な判断を下すための方法としてアンサンブル学習手法に着目し，実際のプロジェクトデータを用いた検証を行った．その結果，アンサンブル学習手法の 1 つである Stacking の有効性を確認できた．この成果は fault-prone モジュール判別手法の実用性向上に寄与すると考えられる．

2. アンサンブル学習手法

アンサンブル学習手法は複数のベース学習器の出力を総合して判別を行う手法の総称である．ベース学習器とは入力に基づいて予測を行うものである．データセットで訓練された判別手法などがベース学習器に用いられる．

アンサンブル学習手法は，すべてのベース学習器に同じ判別手法を適用するものとそれぞれに異なる判別手法を適用するものとに分類できる．前者の代表的な手法は Bagging[5] および Boosting[6] である．後者の代表的な手法は Voting および Stacking である．本研究の目的は複数の fault-prone モジュール判別手法の判別結果に基づいて総合的な判断を下す判別手法が構築可能か検証することにある．したがって後者の Voting と Stacking を採用した．

Voting は複数のベース学習器の出力の多数決から判断を下す単純な手法である．一方，Stacking[7] はより複雑な手法である．Stacking の構造を**図 1** に示す．Stacking は複数のベース学習器と 1 つのメタ学習器から構成される．メタ学習器はベース学習器の出力を入力として判別を行う．つまり，メタ学習器に設定した判別手法が総合的な判断を下す．具体的には，まず判別対象の説明変数の値を入力データとして受け取る．すべてのベース学習器は受け取った入力に基づいて予測を行い，その結果をメタ学習器へ渡す．メタ学習器は入力に基づいて最終的な判別結果を出力する．

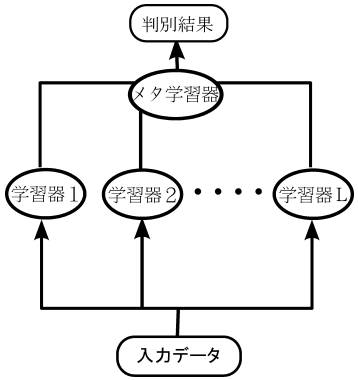


図 1　Stacking の構造
Fig. 1　The structure of Stacking.

ベース学習器はモジュールの特徴と実際の不具合の有無を記録したデータセットを用いて通常の判別手法と同じように訓練する．メタ学習器の訓練には同じデータセットから交差検証法を用いて生成したデータセットを用いる．まずベース学習器に用いる判別手法ごとに交差検証法を適用して判別結果を記録する．この結果を説明変数とし元のデータセットの不具合の有無を目的変数とするデータセットでメタ学習器を訓練する．学習器の数が説明変数の数と一致する．本研究では十重交差検証法を用いた．

3. 評 価 実 験

3.1　fault-prone モジュール判別手法

評価対象のアンサンブル学習手法にはベース学習器ごとに異なる fault-prone モジュール判別手法が必要である．アンサンブル学習手法では不偏性がないベース学習器を用いるのが一般的であるが，本研究の目的は既存の判別手法の予測結果を総合的に判断することにある．よって，ある程度一般的で異なる手法に基づいた fault-prone モジュール判別手法をベース学習器に適用した．既存研究[1)–4)] を調査し，それぞれ線形回帰モデル，回帰木，Analogy-based Reasoning（ABR），ベイズ学習手法に基づいた 4 つの fault-prone モジュール判別手法を選択した．具体的には，それぞれ，ロジスティック回帰モデル，C4.5[8]，ABR[9]，そして Naive Bayesian Classifier（NBC）[10] を用いた．

評価実験ではデータマイニングソフトウェア Weka[11] 上の実装を用いた．Logistic がロジスティック回帰モデル，J48 が C4.5，IB1 が ABR，そして NaiveBayes が NBC に対応する実装である．

3.2　データセット

評価実験で使用したデータセットの特性を**表 1** に示す．5 つのデータセットはすべて NASA の Software Metric Data Program の一環で収集された．これらのデータセットは fault-prone モジュール判別手法の評価実験に広く用いられており，NASA MDP Data Repository や PROMISE Engineering Repository[12] から容易に取得できる．各データセットは 1 つのモジュールに対して 21 個のメトリクスを記録している．これらのメトリクスは McCabe

© 2011 Information Processing Society of Japan

表 1　データセットの特性
Table 1　Datasets characteristics.

データセット	開発言語	不具合を含むモジュール数	モジュール総数
JM1	C	2,106	10,885
CM1	C	49	498
PC1	C	77	1,109
KC1	C++	326	2,109
KC2	C++	107	522

や Halstead のソフトウェアメトリクスなど規模や複雑度に関連するものである．また，モジュールごとに実際の不具合の有無を示す 2 値の離散値を記録している．実験ではデータセット JM1 のうち欠測値を含むデータを除外した．データセットのその他の詳細は PROMISE Engineering Repository から入手できるデータセットのコメントに記載されている．

3.3 実験方法

実験では，アンサンブル学習手法とそのベース学習器に適用したすべての fault-prone モジュール判別手法の性能を比較した．fault-prone モジュール判別手法に用いられる手法の中から Stacking のメタ学習器に適用する判別手法を 1 つ選択した．代表的なアプローチがこれらの判別手法に含まれているためである．また，ベース学習器の数は Voting で多数決を実施することを考慮して 3 つの fault-prone モジュール判別手法を選択して組み合わせた．

ベース学習器のために 4 つの判別手法から 3 つを選ぶ方法は ${}_4C_3 = 4$ 通り，さらに Stacking のメタ学習器のために 1 つを選ぶ方法は ${}_4C_1 = 4$ 通りである．よって Voting と Stacking の組合せを合計すると ${}_4C_3 + {}_4C_3 \times {}_4C_1 = 20$ 通りになる．5 種類のデータセットと組み合わせると 100 通りとなり実験結果の総合的な評価が困難である．

そこでメタ学習器とベース学習器にすべて異なる判別手法を適用した組合せに限定して評価を行った．この限定方法ではベース学習器を 3 つとした場合にすべての判別手法の組合せを用意でき，かつ，メタ学習器にもすべての手法を用いることができる．よってアンサンブル学習手法の特徴を観察するのに適している．この場合ベース学習器が定まると Stacking のメタ学習器は一意に定まる．組合せは ${}_4C_3 \times 2 \times 5 = 40$ 通りとなる．この評価結果から観察できる特徴に応じて追加実験を行いアンサンブル学習手法の有効性を確認した．

表 2 に評価実験を行った組合せを示す．ベース学習器の項では使用した fault-prone モジュール判別手法に印が付いている．No.1–4 は Voting，No.5–8 は Stacking を評価するための組合せである．

表 2　ベース学習器とメタ学習器に対する fault-prone モジュール判別手法の割当て
Table 2　Treatments of fault-prone module detection methods for base-learners and a meta-learner.

No.	アンサンブル学習手法	メタ学習器	ベース学習器 L	C	A	N
1				x	x	x
2	Voting	（なし）	x		x	x
3			x	x		x
4			x	x	x	
5		Logistic (L)		x	x	x
6	Stacking	C4.5 (C)	x		x	x
7		ABR (A)	x	x		x
8		NBC (N)	x	x	x	

実験は 5 種類のデータセットと上記 8 通りの組合せすべてに対して行った．性能評価には十重交差検証法の結果を用いた．この方法は学習セットとテストセットの選び方にランダム性があるため，既存研究[13] と同様に適用を 10 回繰り返した．そして fold ごとに得られる評価指標の平均値で評価を行った．

3.4 評価方法

従来研究[4),14)] と同様に，F-measure，G-mean$_1$，そして G-mean$_2$ の 3 つの評価指標を性能評価に用いた．表 1 が示すような不具合を含むモジュールの割合が低い状況では正確度を評価指標にすることは不適切なためである．たとえば，CM1 の場合，すべてを不具合なしと判別する手法の正確度は約 $498/(498+49) = 91\%$となる．一方，不具合を含むモジュールをすべて正しく判別できるが，不具合を含まないモジュールの 20%を不具合ありと判別する手法の正確度は約 $(49 + 498 \times 0.8)/(498 + 49) = 82\%$となる．後者の方がより有用であるのに評価は低くなるため不適切である．

F-measure，G-mean$_1$，そして G-mean$_2$ は適合率，再現率，そして特異度の 3 つの指標を用いて定義する．これら 3 つの指標は互いにトレードオフの関係にある．適合率は fault-prone であると判別されたモジュールのうち実際に不具合が含まれていたモジュールの割合を示す．再現率は実際に不具合を含んでいたモジュールのうち fault-prone であると判別されたモジュールの割合を表す．特異度は不具合を含んでいなかったモジュールのうち fault-prone ではないと判別されたモジュールの割合を表す．

F-measure，G-mean$_1$，そして G-mean$_2$ の定義を以下に示す．

© 2011 Information Processing Society of Japan

$$\text{F-measure} = \frac{(1+\beta^2) \times \text{適合率} \times \text{再現率}}{\beta^2 \cdot \text{適合率} + \text{再現率}}$$

$$\text{G-mean}_1 = \sqrt{\text{再現率} \times \text{適合率}}$$

$$\text{G-mean}_2 = \sqrt{\text{再現率} \times \text{特異度}}$$

ここで β は適合率と再現率の相対的な重要度を定めるパラメータである．既存研究と同様に，F-measure は $\beta = 1$ として，適合率と再現率を同等に重視する F1 score を採用した．これらの定義は適合率，再現率，特異度のうち 2 つの調和平均もしくは幾何平均である．トレードオフの関係にある指標を平均することで判別手法の性能を総合的に評価している．いずれの評価指標も性能が良いほど大きい値を示す．

複数の fault-prone モジュール判別手法が存在するとき，判別手法の性能に関して以下の 2 通りの状況がありうる．

状況 X　最も良い性能を示す判別手法が 1 つ存在する．

状況 Y　他より良い性能を示す判別手法が存在しない．

状況 X で一番単純な総合的判断の方法は，最も性能が良い判別手法を選んで利用することである．Stacking は学習によってこの単純な方法を採用する可能性がある．このときに Stacking は最も良い性能を示すベース学習器に用いられた判別手法と同じ性能を示すことを考えると，複数のベース学習器による判別結果から総合的な判断を下すように学習されていた場合でも同程度以上の性能であることが望ましい．よって，状況 X では，最も良い性能を示すベース学習器と同程度以上の性能であれば，アンサンブル学習手法による総合的な判断は妥当であると見なせる．

一方，状況 Y では上記の単純な方法をとるのが難しいため，アンサンブル学習手法は複数のベース学習器による判別結果から総合的な判断を下すと考えられる．Stacking ではベース学習器すべての判別結果と実際の不具合の有無の間の関連性をメタ学習器に学習させて総合的な判断を下す．また，Voting では学習を行う代わりにあらかじめ定められた手順に従って総合的な判断を下す．このとき，アンサンブル学習手法は任意の判別手法を選んで利用する場合と同程度以上の性能であることが望ましい．よって，状況 Y では，すべてのベース学習器に適用された判別手法と比較して同程度以上の性能であれば，アンサンブル学習手法による総合的な判断は妥当であると見なせる．

ここで総合的な判断が妥当であると見なす基準を比較すると，状況 X における基準は状況 Y の特別な場合である．本研究では 3 つの評価基準を用いることを考慮して，3 つの評価指標すべてについてすべてのベース学習器に適用された判別手法と同程度以上の性能を示した場合にアンサンブル学習手法が妥当な判断を下したと評価する．

4. 結　　果

この章では実験結果の要点についてデータセットごとに述べる．実験結果の詳細は付録 A.1 に記載した．実験結果を観察すると，4 つの fault-prone モジュール判別手法の中で，すべての評価指標において他より良い性能を示すものが存在することが多かった．つまり，ベース学習器の組合せ 4 通りのうち少なくとも 3 通りは前章で述べた状況 X にある．この 3 通りの組合せを用いるアンサンブル学習手法では一番良い性能を示した判別手法が妥当性評価の基準となる．これらを互いに比較することでアンサンブル学習手法の判別性能におけるベース学習器の組合せによる違いを観察できると考えられる．

以上の観察に基づいて，ベース学習器の組合せを基準に以下の 2 通りの比較評価に分類した．

比較評価 A　最も性能が良い判別手法と，その手法に基づいたベース学習器を含むアンサンブル学習法の比較．

比較評価 B　最も性能が良い判別手法以外の手法と，それらの手法に基づくベース学習器のみ含むアンサンブル学習手法の比較．

JM1 を用いた実験では，fault-prone モジュール判別手法単体で ABR が最も性能が高かった．よって，ABR を適用したベース学習器を含む組合せが比較評価 A に該当する．比較評価 A に該当するアンサンブル学習手法の中では No.8 の Stacking のみが ABR と同程度以上の性能であった．また，No.8 の Stacking のみ対応する Voting より良い結果を示した．比較評価 B に該当する No.3，7 では Stacking である No.7 の方がより良い結果であったが，ベース学習器に適用された判別手法の中で最も良い C4.5 より悪い結果であった．

CM1 を用いた実験では，fault-prone モジュール判別手法単体で NBC が最も性能が高い手法であった．比較評価 A に該当するアンサンブル学習手法の中では No.7 が最も良い結果だったが NBC より悪い結果であった．また，Stacking が Voting より良い結果を示すのは No.7 のみであった．比較評価 B に該当する No.4，8 では Stacking である No.8 がより良い結果を示した．また，No.8 はベース学習器に適用したすべての判別手法と同程度以上の性能であった．

PC1 を用いた実験では，fault-prone モジュール判別手法単体で ABR が最も性能が高い手法であった．比較評価 A に該当するものの中では No.8 が最も良い結果だったが ABR よ

© 2011 Information Processing Society of Japan

り悪い結果であった．また，Stacking が Voting より良い結果を示すのは No.8 のみであった．比較評価 B に該当する No.3，7 では No.7 の Stacking が G-mean$_1$ 以外でより良い結果であった．しかし C4.5 および NBC より悪い結果であった．

KC1 を用いた実験では fault-prone モジュール判別手法単体で ABR が最も性能が高い手法であった．比較評価 A に該当する中では No.8 が最も良く，ABR と同程度以上の性能を示したのは No.8 のみであった．また，Stacking が Voting より良い結果を示すのも No.8 のみであった．比較評価 B に該当する No.3，7 では No.3 がより良い結果であったが，NBC より悪い結果であった．

KC2 を用いた実験では他より良い性能を示す fault-prone モジュール判別手法は他のデータセットほど明確ではなかった．G-mean$_1$ と F-measure は手法間でほぼ差はないといえる．一方，G-mean$_2$ では ABR が他の手法よりやや良い性能を示した．そこで ABR を最も性能が高い手法とした．比較評価 A に該当する中では No.8 が最も良く ABR と同程度以上の性能であった．また，Stacking が Voting より良い結果を示すのは No.8 のみであった．比較評価 B に該当する No.3，7 では No.3 がより良い結果で，ベース学習器に適用した判別手法より性能がやや低かった．

5. 考　　察

表 3 は 4 章の評価結果をまとめたものである．3 列目と 4 列目には比較評価 A および B での結果を同時に記載した．3 列目は比較評価ごとの最も性能が良いアンサンブル学習手法である．すべての評価指標を同程度の重要性で扱い比較した．また評価指標として最も一般的な F-measure の値を併記した．4 列目は 3 列目と対応しており，妥当な判断を下したと評価された場合を “b”，そうでない場合を “w” と記載した．

表 3 より，No.8 に限れば総合的な判断が妥当である場合が多い．原因としてメタ学習器の影響が考えられるため，No.5，6，7 のメタ学習器を NBC に変更して再度評価実験を行った．

表 3　アンサンブル学習手法の評価結果（F-measure）
Table 3　Evaluation results of ensemble-learning methods (F-measure).

データセット	最も性能が良い判別手法	最も性能が良いアンサンブル学習手法	比較結果
JM1	ABR(0.371)/C4.5(0.313)	8/7(0.399/0.293)	b/w
CM1	NBC(0.263)/ABR(0.179)	7/8(0.185/0.242)	w/b
PC1	ABR(0.420)/C4.5(0.324)	8/7(0.369/0.268)	w/w
KC1	ABR(0.428)/NBC(0.394)	8/3(0.465/0.387)	b/w
KC2	ABR(0.500)/NBC(0.501)	8/3(0.550/0.495)	b/w

実験結果を**表 4** に示す．すべての評価指標が同じ傾向を示したため，ここでは評価指標で最も一般的な F-measure の値のみを示す．メタ学習器を NBC に変更した組合せにはダッシュを付与している．比較のため，No.5，6，7，8 の結果および比較評価 A/B それぞれで妥当性の基準となる値を併記した．

実験の結果，No.5，6，7 のすべてで性能が向上した．NBC 適用前と比べて 1%から 34%の向上である．CM1 を用いた比較評価 A で No.6’，7’ による総合的な判断が妥当な結果を示すようになった．また，JM1，KC1，KC2 を用いた比較評価 B では No.7’ の総合的な判断が妥当な結果を示すようになった．PC1 の No.7’ は F-measure と G-mean$_2$ ではより良い結果を示したが，G-mean$_1$ が C4.5 より約 2%低かった．各組合せに対応する Voting である No.1，2，3 と比較した結果，PC1 の No.5’ 以外すべてで 3%から 18%良い性能を示すことが確認できた．以上から，メタ学習器に NBC を選択することが総合的な判断の妥当性に大きく影響したといえる．

表 5 に No.5’，6’，7’ を用いた評価結果を示す．PC1 以外のすべてのデータセットで Stacking による総合的な判断が妥当であったといえる．以上の結果から，メタ学習器に NBC を適用した Stacking は，多くの場合，代表的な fault-prone モジュール判別手法の予測結果を総合的に判断するための手法として妥当であることが確認できた．また，PC1 以外で No.7’，8 がつねに妥当な判断を下しており，安定して妥当な判断を下す組合せが存在することが確認できた．一方，比較評価 A/B それぞれで最も性能が良いものを比較すると，比較評価 A

表 4　メタ学習器に NBC を用いた Stacking の判別精度（F-measure）
Table 4　Performance results of Stacking with NBC meta-learner (F-measure).

No.	Dataset				
	JM1	CM1	PC1	KC1	KC2
5	0.234	0.000	0.274	0.305	0.463
6	0.178	0.014	0.026	0.270	0.524
7	0.293	0.185	0.268	0.323	0.452
5’	0.375	0.256	0.351	0.440	0.542
6’	0.334	0.314	0.366	0.426	0.532
7’	0.386	0.287	0.345	0.437	0.542
8	0.399	0.242	0.369	0.465	0.550
A	0.371	0.263	0.420	0.428	0.500
B	0.313	0.179	0.324	0.394	0.501

© 2011 Information Processing Society of Japan

表 5　メタ学習器に NBC を用いたアンサンブル学習手法の評価結果（F-measure）
Table 5　Evaluation results of ensemble-learning methods with NBC meta-learner (F-measure).

データセット	最も性能が良い判別手法	最も性能が良いアンサンブル学習手法	比較結果
JM1	ABR(0.371)/C4.5(0.313)	8/7'(0.399/0.386)	b/b
CM1	NBC(0.263)/ABR(0.179)	6'/8(0.314/0.242)	b/b
PC1	ABR(0.420)/C4.5(0.324)	6'/7'(0.366/0.345)	w/w
KC1	ABR(0.428)/NBC(0.394)	8/7'(0.465/0.437)	b/b
KC2	ABR(0.500)/NBC(0.501)	8/7'(0.550/0.542)	b/b

に属するものがつねにより良い性能であった．このことはベース学習器に適用する判別手法の性能が Stacking の性能に反映されることを示している．

表 5 に示した NBC を用いた Stacking の結果を観察すると，ベース学習器に適用した判別手法と比べて再現率が高い傾向にあった．これは実際に不具合を含むモジュールが判別手法単体の場合より多く fault-prone であると判別されたことを示す．つまり，1 章で述べたような判別結果が手法によって異なるモジュールの一部も fault-prone であると判別されているといえる．一方，ベース学習器に適用した判別手法すべてと比べて適合率は低下しているが比較的良い方であり，fault-prone と判別されたモジュールは実際に不具合を含むことが多かったといえる．以上の理由から，NBC を用いた Stacking が良い性能を示したのは，個々のベース学習器の判別結果に適切に重みを付け，不具合を含むモジュールをより多くある程度の精度で判別できるようにメタ学習器が構成されたためであると考えられる．

一方，PC1 の比較評価 A の詳細を分析したところ，No.6' は再現率が ABR より約 10%高かったが適合率と特異度が低かった．No.8 は逆の傾向を示し，適合率と特異度は ABR と同等だったが再現率が約 7%低かった．No.5' はその中間であった．表 4 が示すように，NBC を用いた Stacking は PC1 においても他の判別手法をメタ学習器に用いた Stacking および Voting より高い性能を示している．このことは NBC を用いた Stacking が下す総合的な判断に一定の妥当性があることを示す．比較評価 B がほぼ妥当な判断を下したといえる結果であったことを考えると，比較評価 A のみが含む ABR の影響が考えられるが，他のデータセットでは ABR を含むことが問題になっていない．以上から，PC1 でかつ ABR を含むベース学習器の組み合わせ方といった複合的な条件によって妥当な判断を下せなかったと推測される．このことはデータセットによって学習器に適用する判別手法の組み合わせ方を考える必要がある可能性を示している．

NBC を用いた Stacking は今回使用した NASA プロジェクト以外のデータセットについても有効であると考えられる．実験結果から NBC を用いた Stacking はベース学習器の判別結果が一致しない場合を適切に扱うことができているといえる．そして，判別手法が異なる理論に基づくものであれば，不具合を含むモジュールの判別結果が一致しない状況は多くのデータセットで発生すると考えられるからである．ただし，適切な判別手法の組み合わせ方については，他のデータセットによる実験で PC1 のように妥当な判断を下せない例を収集し，その傾向やパターンについて調査する必要がある．

6. 関 連 研 究

現在まで多くの fault-prone モジュール判別手法が提案されており，また，提案された手法の比較評価もさかんに行われている．しかしながら，比較評価の結果は研究ごとに異なっており，一貫して最良といえる手法はない．工数見積り手法の比較研究においても一貫性のない結果が得られることが Myrtveit らによって指摘されている[15]．Shepperd らは，データセットの特徴や開発対象となるソフトウェアのシステムドメインのような個別の事情がこの現象を引き起こしていると指摘している[16]．また，Ma らは本研究で実験に用いたデータセットについて，基本的な統計量を比較してデータセットが異なる特徴を持っていることを指摘している[4]．

アンサンブル学習手法を fault-prone モジュール判別に適用した研究としては Khoshgoftaar らの研究[17),18)] および Ma らの研究[14)] があげられる．アンサンブル学習手法として Khoshgoftaar らは Boosting と Bagging を，Ma らは Random Forests を用いた．いずれの研究も単一の判別手法をベース学習器に採用するアンサンブル学習手法を対象としており，複数の fault-prone モジュール判別手法の予測結果を総合的に判断する本研究とは研究目的が異なる．

異なる種類の fault-prone モジュール判別手法を組み合わせた研究としては文献 19) がある．文献 19) で亀井らは相関ルール分析とロジスティック回帰モデルの中から状況に応じて一方を選び fault-prone モジュールの判別を行っている．また，複数のルールが存在する場合には判別結果の多数決を行っている．本研究は判別手法の予測結果の組み合わせ方を含めて学習するという違いがある．これには採用する判別手法の種類に依存しないという利点があげられる．

7. ま と め

本研究では，複数の fault-prone モジュール判別手法の出力結果を総合的に判断する手法

© 2011 Information Processing Society of Japan

としてアンサンブル学習手法の評価を行った．比較評価の結果，メタ学習器が適切に選択された Stacking が総合的な判断を下す手法として多くの場合に妥当であることを確認できた．

NBC をメタ学習器に使用した実験では最も性能が良い判別手法を単体で利用する場合と比べて F-measure で 2.8%から 4.9%の向上が確認できた．また $G\text{-mean}_1$ で最大 5%程度，$G\text{-mean}_2$ で最大 10%程度の向上が確認できた．これは多くの場合で，評価指標すべてに含まれる再現率が向上したためである．このときの再現率の向上は 4.8%から 10.9%であった．さらに，Voting と比べて F-measure で 3%から 18%の向上が確認された．一般にソフトウェアは出荷後の不具合を修正する費用が高価であるといわれており，再現率の向上による総合評価指標の改善は Stacking の有用性を示しているといえる．

Stacking では特別なデータセットや訓練の方法は不要で既存の fault-prone モジュール判別手法と同様に扱える．たとえば，過去の類似プロジェクトや同一製品の前のバージョンなどから 3.2 節のようなソースコードに基づいたメトリクスやプロセスメトリクスを収集し，そのデータセットで訓練することができる．つまり，利用において費用面での大きな増加はない．ただし，モデルの構築・改善ではベース学習器に適用する判別手法の選択・改善が必要となる．改善では個々の判別手法の性能を改善することが必要である．また，NBC をメタ学習器に用いた実験結果のように Stacking の特徴が再現率向上にあるとすれば，互いの判別結果の違いがあらわれるような判別手法の組合せを採用することが重要だと考えられる．

一方，NBC を用いた実験で再現率が向上する一方で全体的な性能が低下する組合せが存在する可能性があることが分かった．今後の研究では他のデータセットによる実験を行い，同様の結果を示すデータセットからこの組み合わせ方について分析を行う必要がある．また，NBC 以外の判別手法をメタ学習器に適用することでこの問題が解決できるかについても研究を行っていく必要がある．

参 考 文 献

1) Khoshgoftaar, T.M., Ganesan, K., Allen, E.B., Ross, F.D., Munikoti, R., Goel, N. and Nandi, A.: Predicting Fault-Prone Modules with Case-Based Reasoning, *Proc. 8th International Symposium on Software Reliability Engineering*, Los Alamitos, CA, USA, IEEE Computer Society, pp.27–35 (1997).
2) Khoshgoftaar, T.M., Allen, E.B. and Deng, J.: Using Regression Trees to Classify Fault-Prone Software Modules, *IEEE Trans. Reliability*, Vol.51, No.4, pp.455–462 (2002).
3) Munson, J. and Khoshgoftaar, T.: The detection of fault-prone programs, *IEEE Trans. Softw. Eng.*, Vol.18, No.5, pp.423–433 (1992).
4) Ma, Y. and Cukic, B.: Adequate and Precise Evaluation of Quality Models in Software Engineering Studies, *PROMISE'07*, pp.1–9 (2007).
5) Breiman, L.: Bagging Predictors, *Machine Learning*, Vol.24, No.2, pp.123–140 (1996).
6) Freund, Y. and Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, Vol.55, No.1, pp.119–139 (1997).
7) Wolpert, D.H.: Stacked Generalization, *Neural Networks*, Vol.5, No.2, pp.241–259 (1992).
8) Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers (1993).
9) Shepperd, M., Schofield, C. and Kitchenham, B.: Effort estimation using analogy, *Proc. 18th International Conference on Software Engineering*, pp.170–178 (1996).
10) Dura, R.O., Hart, P.E. and Stork, D.G.: *Pattern Classification*, John Wiley & Sons, Inc. (2001).
11) Weka Machine Learning Project: Weka 3: Data Mining Software in Java. http://www.cs.waikato.ac.nz/~ml/weka/
12) Shirabad, J.S. and Menzies, T.J.: *The PROMISE Repository of Software Engineering Databases*, School of Information Technology and Engineering, University of Ottawa, Canada (2005).
13) Guo, L., Ma, Y., Cukic, B. and Singh, H.: Robust Prediction of Fault-Proneness by Random Forests, *Proc. 15th International Symposium on Software Reliability Engineering*, pp.417–428 (2004).
14) Ma, Y., Guo, L. and Cukic, B.: A Statistical Framework for the Prediction of Fault-Proneness, *Advances in Machine Learning Applications in Software Engineering*, Idea Group (2007).
15) Myrtveit, I., Stensrud, E. and Shepperd, M.: Reliability and Validity in Comparative Studies of Software Prediction Models, *IEEE Trans. Softw. Eng.*, Vol.31, No.5, pp.380–391 (2005).
16) Shepperd, M. and Kadoda, G.: Using Simulation to Evaluate Prediction Techniques, *Proc. 7th International Software Metrics Symposium* (2001).
17) Khoshgoftaar, T.M., Geleyn, E., Nguyen, L. and Bullard, L.: Cost-Sensitive Boosting In Software Quality Modeling, *Proc. 7th IEEE International Symposium on High Assurance Systems Engineering* (*HASE'02*), pp.51–59 (2002).
18) Khoshgoftaar, T.M., Galeyn, E. and Nguyen, L.: Empirical Case Studies of Combining Software Quality Classifictaion Models, *Proc. 3rd International Conference on Quality Software* (*QSIC'03*), pp.40–49 (2003).

© 2011 Information Processing Society of Japan

19) 亀井靖高，森崎修司，門田暁人，松本健一：相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別方法，情報処理学会論文誌，Vol.49, No.12, pp.3954–3966 (2008).

付　　録

A.1　実験結果の一覧

表 6，**表 7**，**表 8**，**表 9**，**表 10** はデータセットごとの実験結果を示す．各表の上部は fault-prone モジュール判別手法単体の結果である．下部では同じベース学習器の組合せを持つ Voting と Stacking の結果をまとめて示した．「Voting(-Method)」はベース学習器に Method **以外**を適用した Voting であることを示す．「Stacking(Method)」はメタ学習器に Method を適用した Stacking であることを示す．fault-prone モジュール判別手法の中で最も良い結果と，ベース学習器が同じ組合せの Voting と Stacking のうち性能がより良い方の結果を太字で記載した．

各表の左端の項目「No.」は表 2 の番号と対応する．また，記号 A，B は比較評価 A/B

表 6　JM1 における判別精度
Table6　Performance results on project JM1.

No.	組合せ	適合率	再現率	特異度	G-mean$_1$	G-mean$_2$	F-measure
B	Logistic(L)	0.586	0.115	0.980	0.259	0.335	0.192
B	C4.5(C)	0.459	0.240	0.932	0.330	0.470	0.313
A	ABR(A)	0.374	0.369	0.852	**0.371**	**0.560**	**0.371**
B	NBC(N)	0.484	0.199	0.949	0.310	0.434	0.282
1 A	Voting(-L)	0.534	0.225	0.953	**0.346**	**0.462**	**0.316**
5 A	Stacking(L)	0.550	0.150	0.970	0.286	0.380	0.234
2 A	Voting(-C)	0.561	0.167	0.969	**0.305**	**0.401**	**0.256**
6 A	Stacking(C)	0.609	0.108	0.981	0.249	0.318	0.178
3 B	Voting(-A)	0.509	0.178	0.959	**0.300**	0.412	0.263
7 B	Stacking(A)	0.294	0.293	0.831	0.294	**0.493**	**0.293**
4 A	Voting(-N)	0.506	0.242	0.943	0.350	0.477	0.327
8 A	Stacking(N)	0.458	0.356	0.899	***0.403**	***0.564**	***0.399**

表 7　CM1 における判別精度
Table7　Performance results on project CM1.

No.	組合せ	適合率	再現率	特異度	G-mean$_1$	G-mean$_2$	F-measure
B	Logistic(L)	0.302	0.134	0.969	0.196	0.254	0.178
B	C4.5(C)	0.094	0.055	0.971	0.070	0.108	0.065
B	ABR(A)	0.209	0.175	0.923	0.185	0.318	0.179
A	NBC(N)	0.259	0.298	0.908	***0.270**	***0.437**	***0.263**
1 A	Voting(-L)	0.196	0.095	0.963	**0.131**	**0.191**	**0.120**
5 A	Stacking(L)	0.000	0.000	0.996	0.000	0.000	0.000
2 A	Voting(-C)	0.279	0.146	0.963	**0.196**	**0.271**	**0.182**
6 A	Stacking(C)	0.022	0.010	0.996	0.015	0.022	0.014
3 A	Voting(-A)	0.270	0.124	0.968	0.176	0.237	0.159
7 A	Stacking(A)	0.207	0.188	0.915	**0.191**	**0.328**	**0.185**
4 B	Voting(-N)	0.117	0.042	0.973	0.068	0.087	0.059
8 B	Stacking(N)	0.264	0.246	0.931	**0.248**	**0.399**	**0.242**

表 8　PC1 における判別精度
Table8　Performance results on project PC1.

No.	組合せ	適合率	再現率	特異度	G-mean$_1$	G-mean$_2$	F-measure
B	Logistic(L)	0.344	0.087	0.987	0.166	0.212	0.131
B	C4.5(C)	0.606	0.235	0.989	0.367	0.441	0.324
A	ABR(A)	0.438	0.433	0.955	***0.427**	***0.618**	***0.420**
B	NBC(N)	0.255	0.303	0.934	0.274	0.500	0.270
1 A	Voting(-L)	0.584	0.281	0.984	**0.395**	**0.494**	**0.364**
5 A	Stacking(L)	0.572	0.192	0.991	0.321	0.381	0.274
2 A	Voting(-C)	0.455	0.164	0.985	**0.266**	**0.349**	**0.231**
6 A	Stacking(C)	0.049	0.020	0.996	0.030	0.043	0.026
3 B	Voting(-A)	0.481	0.184	0.985	**0.289**	0.376	0.254
7 B	Stacking(A)	0.288	0.277	0.944	0.275	**0.469**	**0.268**
4 A	Voting(-N)	0.591	0.233	0.988	0.361	0.439	0.321
8 A	Stacking(N)	0.425	0.365	0.958	**0.381**	**0.567**	**0.369**

表 9　KC1 における判別精度
Table9　Performance results on project KC1.

No.	組合せ	適合率	再現率	特異度	G-mean$_1$	G-mean$_2$	F-measure
B	Logistic(L)	0.619	0.217	0.975	0.363	0.454	0.317
B	C4.5(C)	0.478	0.297	0.940	0.373	0.522	0.360
A	ABR(A)	0.465	0.404	0.912	**0.431**	**0.604**	**0.428**
B	NBC(N)	0.426	0.372	0.907	0.397	0.577	0.394
1 A	Voting(-L)	0.542	0.343	0.946	**0.429**	**0.566**	**0.416**
5 A	Stacking(L)	0.603	0.210	0.973	0.351	0.446	0.305
2 A	Voting(-C)	0.571	0.298	0.957	**0.410**	**0.531**	**0.388**
6 A	Stacking(C)	0.576	0.186	0.974	0.318	0.409	0.270
3 B	Voting(-A)	0.527	0.312	0.948	**0.403**	**0.538**	**0.387**
7 B	Stacking(A)	0.329	0.324	0.876	0.325	0.527	0.323
4 A	Voting(-N)	0.590	0.320	0.958	0.432	0.550	0.410
8 A	Stacking(N)	0.473	0.463	0.904	***0.466**	***0.644**	***0.465**

© 2011 Information Processing Society of Japan

表 10　KC2 における判別精度
Table 10　Performance results on project KC2.

No.		組合せ	適合率	再現率	特異度	G-mean$_1$	G-mean$_2$	F-measure
	B	Logistic(L)	0.640	0.407	0.935	0.502	0.607	0.483
	B	C4.5(C)	0.579	0.468	0.901	0.505	0.632	0.489
	A	ABR(A)	0.501	0.525	0.854	0.506	**0.661**	0.500
	B	NBC(N)	0.681	0.411	0.946	**0.523**	0.616	**0.501**
1	A	Voting(-L)	0.633	0.458	0.925	**0.527**	**0.639**	**0.512**
5	A	Stacking(L)	0.686	0.374	0.947	0.493	0.583	0.463
2	A	Voting(-C)	0.666	0.428	0.940	0.526	0.626	0.508
6	A	Stacking(C)	0.665	0.455	0.932	**0.541**	**0.643**	**0.524**
3	B	Voting(-A)	0.670	0.409	0.943	**0.516**	0.613	**0.495**
7	B	Stacking(A)	0.471	0.456	0.859	0.457	**0.617**	0.452
4	A	Voting(-N)	0.609	0.469	0.914	0.525	0.645	0.513
8	A	Stacking(N)	0.558	0.573	0.873	***0.557**	***0.698**	***0.550**

への対応を示す．比較評価 A については，複数のアンサンブル学習手法どうしも比較するため，わかりやすいように各評価指標の最も良い結果に「$*$」を付与した．

(平成 22 年 3 月 27 日受付)
(平成 22 年 11 月 5 日採録)



天嵜　聡介（正会員）

平成 18 年大阪大学大学院情報科学研究科博士後期課程修了．現在，岡山県立大学情報工学部助教．工数見積り手法，fault-prone モジュール判別手法の研究に従事．博士（情報科学）．IEEE-CS，ACM 各会員．

© 2011 Information Processing Society of Japan