

ブースティングに基づいたソフトウェア信頼性予測性能の評価

呉 敬馳
広島大学

kure@hiroshima-u.ac.jp

土肥 正
広島大学

dohi@hiroshima-u.ac.jp

鄭 俊俊
広島大学

jzheng@hiroshima-u.ac.jp

岡村 寛之
広島大学

okamu@hiroshima-u.ac.jp

要旨

過去数十年において、数多くのソフトウェア信頼性モデル (SRM) が提案されてきた。通常、赤池情報量規準 (AIC) などのモデル選択基準を用いて、ソフトウェア故障数の計数データに対する適合性が最も高い SRM を選択し、将来の故障数の予測に使うのが一般的である。しかしながら、過去の観測データに最も良く適合したモデルが、将来発生するソフトウェア故障数に対する最良な予測モデルであるとは限らないことが知られている。本稿では、代表的なアンサンブル学習器のひとつであるブースティングに着目し、ソフトウェア信頼性予測手法を提案する。実際のソフトウェア開発プロジェクトの故障数データを用いた数値実験において、最小の AIC によって選択された単一モデルと比較して、複数モデルの重み付き平均を用いた予測法の方がシステムテストの初期および中期段階における予測性能が高いことを示す。

1 はじめに

ソフトウェア信頼性評価は、プロジェクトマネージャーやテストエンジニアにとって重要で

あり、ソフトウェアシステムのリリース時期を決定するのに有用な指標である。定量的なソフトウェア信頼性であるソフトウェア信頼度は、ソフトウェア故障が一定の期間内に発生しない確率として定義される。テスト工程において発生したソフトウェア故障数の時間的挙動は、通常、非減少整数値確率過程として記述される。よって過去の多くの研究では、さまざまな確率点過程に基づいた SRM が提案されてきた。その中でも、非定常ポアソン過程 (NHPP) に基づいた SRM は、データへの適合性の高さやパラメータ推定の簡便性から、最も汎用性の高いモデルとして認識されている [1, 2, 8–10, 16, 17, 19, 25, 26]。Goel and Okumoto [8] は、すべてのソフトウェア故障発生時刻が独立で同一な指数分布に従うと仮定し、最も基本的な NHPP に基づいた SRM を提案した。その後、パレート分布 [1]、対数正規分布や切断正規分布などの正規分布族 [2, 19]、対数ロジスティック分布や切断ロジスティック分布などのロジスティック分布族 [10, 16]、ワイブル分布やゴンベルツ分布などの極値型分布 [9, 17]、ガンマ分布 [25, 26] など、故障発生時間分布として表現能力の高い NHPP に基づいた SRM が提案されている。Okamura and Dohi [18] は、上記の代表的な 11 種類の SRM を実装したソ

ソフトウェア信頼性評価ツール (SRATS) を開発し、大域的収束性を有する EM (Expectation-Maximization) アルゴリズムを適用することで、ツール上で効率的に最尤推定を行うスキームを開発している。

将来のテスト期間においてソフトウェア故障発生数を予測するためには、これまでに観測した故障数データに基づいて赤池情報量基準 (AIC) [3] などのモデル選択基準を導出し、過去のデータに最も適合した SRM を予測モデルに用いるのが一般的である。しかしながら、Wu et al. [23] において示されたように、AIC、ベイズ情報量基準 (BIC) [20]、その他の関連した情報量基準 [4, 21] を用いてモデル選択したとしても、予測期間における事後的に最適な予測モデルとは必ずしも一致せず、事前に最良の単一予測モデルを知ることは原理的に困難である。このような問題に対して、Lyu and Nikora [14] は、複数の SRM の線形加重結合をメタ予測器として捉え、予測モデルの選択確率である単一 SRM の重みを決定する問題を考察している。その後、Hsu and Huang [11]、Wu and Huang [22] は、複数の NHPP モデルの線形加重結合モデルの重みを求める近似アルゴリズムやニューラルネットワークの誤差逆伝搬アルゴリズムを紹介している。最近、呉ら [24] は赤池ウェイトと呼ばれる加重推論技術を導入し、複数の SRM の線形加重結合を矛盾なく求める方法を提案している。呉ら [24] において紹介された方法は、線形多重回帰モデルの予測において近年注目を集めているモデル平均 (例えば [6] を参照) の手法を確率点過程の予測問題に応用したものである。SRM の線形加重結合は多くの場合、情報量基準に基づいて単一モデルを選択しそれを予測モデルとして用いる方法よりも予測性能が高い反面、赤池ウェイトに基づいた方法でさえも古典的な確率推論の範囲を出ていないことに注意すべきである。

本稿では、近年注目を集めている代表的なア

ンサンブル学習器のひとつであるブースティングに着目したソフトウェア信頼性予測手法を考察する。Li et al. [12] は Ada ブースティングを複数の NHPP モデルの線形加重結合モデルに対する重みを求める問題に適用しており、これは SRM の各候補モデルの重みをリサンプリングされたデータから推定する方法として捉えることができる。元々、機械学習の文脈において Freund and Schapire [7] は、弱学習器と呼ばれる複数の分類器の線形加重結合によって表現される統計的分類器である Ada ブースティングを提案した。後に Drucker [5] は、Freund and Schapire [7] による Ada ブースティングを回帰問題に応用し、AdaBoosting.R2 と呼ばれるアルゴリズムを開発した。これらのブースティングアルゴリズムは決定論的 Ada ブースティングと呼ばれるべきもので、Li et al. [12] によって用いられたアルゴリズムとは基本構造が大きく異なるものである。そこで本稿では、決定論的 Ada ブースティングに基づいたモデル (Deterministic AdaBoosting-based Model: DAM) を提案し、ソフトウェア故障数の予測精度の観点からブースティングによるソフトウェア信頼性予測性能の評価を行う。特に、実際のソフトウェア開発プロジェクトの故障数データを用いた数値実験において、最小の AIC によって選択された単一モデルと比較して、複数モデルの重み付き平均を用いた予測法の方がシステムテストの初期および中期段階における予測性能が高いことを示す。

2 NHPP に基づいた SRM

2.1 定義

時刻 $t (\geq 0)$ までにシステムテストで発生した累積ソフトウェア故障の数を $N(t)$ とする。ここで、 $N(t)$ は非定常ポアソン過程 (NHPP)

表 1: 11 種類の代表的 NHPP に基づいた SRM.

平均値関数	$\Lambda(t; \boldsymbol{\theta})$ ($\boldsymbol{\theta} = a, b, c > 0$)
Exp [8]	$\Lambda(t; \boldsymbol{\theta}) = a(1 - e^{-bt})$
Gamma [25, 26]	$\Lambda(t; \boldsymbol{\theta}) = a \int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto [1]	$\Lambda(t; \boldsymbol{\theta}) = a \left(1 - \left(\frac{c}{t+c} \right)^b \right)$
TruncNormal [19]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{F(t) - F(0)}{1 - F(0)}$, $F(t) = \frac{1}{\sqrt{2\pi b}} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
LogNormal [2, 19]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{1}{\sqrt{2\pi b}} \int_{-\infty}^{\log(t)} e^{-\frac{(s-c)^2}{2b^2}} ds$
TruncLogist [16]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{F(t) - F(0)}{1 - F(0)}$, $F(t) = \frac{1}{1 + e^{-\frac{t-c}{b}}}$
LogLogist [10]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{1}{1 + e^{-\frac{\log(t)-c}{b}}}$
TruncEVMMax [17]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{F(t) - F(0)}{1 - F(0)}$, $F(t) = e^{-e^{-\frac{t-c}{b}}}$
LogEVMMax [17]	$\Lambda(t; \boldsymbol{\theta}) = a e^{-e^{-\frac{\log(t)-c}{b}}}$
TruncEVMMin [17]	$\Lambda(t; \boldsymbol{\theta}) = a \frac{F(0) - F(-t)}{F(0)}$, $F(t) = e^{-e^{-\frac{t-c}{b}}}$
LogEVMMin [9, 17]	$\Lambda(t; \boldsymbol{\theta}) = a \left(1 - e^{-e^{-\frac{-\log(t)-c}{b}}} \right)$

に従うと仮定すると、次式が成り立つ。

$$\Pr \{N(t) = n\} = \frac{\{\Lambda(t)\}^n e^{-\Lambda(t)}}{n!}. \quad (1)$$

ただし、 $\Lambda(t) = \int_0^t \lambda(x) dx = E[N(t)]$ は平均値関数と呼ばれ、 $N(t)$ の期待値を表す。また、 $\lambda(t) (\geq 0)$ は強度関数と呼ばれている。このように、NHPP に基づいた SRM は $\Lambda(t)$ または $\lambda(t)$ によって一意に定まる。過去の文献では、平均値関数を $\Lambda(t; \boldsymbol{\theta}) = aF(t; \boldsymbol{\alpha})$ と仮定することで多数のモデルが提案されている。ここで、 $a (> 0)$ はシステムテスト前にソフトウェア内に残存する総期待フォールト数、 $F(t; \boldsymbol{\alpha})$ は各ソフトウェア故障が発生する時間を表わす連続形確率分布関数、 $\boldsymbol{\theta} = (a, \boldsymbol{\alpha})$ はモデルパラメータである。上述のモデル化の前提として、ソフ

トウェアに内在する一つのフォールトによりシステムテスト期間中ひとつの故障が生じるものと仮定している。表 1 では、SRATS [18] において実装された 11 種類の代表的な NHPP に基づいた SRM を示す。

さらに、モデルパラメータの推定値を $\hat{\boldsymbol{\theta}}$ とすれば、ある時間間隔 $[t_u, t_s)$ におけるソフトウェア信頼度 $R(t_u, t_s; \hat{\boldsymbol{\theta}})$ は次式により定義される。

$$R(t_u, t_s; \hat{\boldsymbol{\theta}}) = e^{-\{\Lambda(t_s; \hat{\boldsymbol{\theta}}) - \Lambda(t_u; \hat{\boldsymbol{\theta}})\}}. \quad (2)$$

すなわち、ソフトウェア信頼度は時間区間 $[t_u, t_s)$ において新たな故障が発生しない確率として定義され、任意の時刻 t_s までに発生する総期待故障数 $\Lambda(t_s; \hat{\boldsymbol{\theta}})$ に依存することがわかる。

2.2 最尤法

次に、モデルパラメータの推定について説明する。\$n\$ 個のソフトウェア故障発生時間データ \$\mathbf{t} = \{t_1, t_2, \dots, t_n\}\$ が観測されたとする。\$n\$ 番目の故障打ち切りの下で、対数尤度関数は次式で与えられる。

$$\ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{t}) = \sum_{i=1}^n \ln \lambda(t_i; \boldsymbol{\theta}) - \Lambda(t_n; \boldsymbol{\theta}). \quad (3)$$

このとき、パラメータ \$\boldsymbol{\theta}\$ の最尤推定値 (MLE) \$\hat{\boldsymbol{\theta}}\$ は次式によって定義される。

$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{t}). \quad (4)$$

仮に、候補となる SRM が \$M\$ 種類 (表 1 では \$M = 11\$) 存在するものとし、\$M\$ 個の SRM の平均値関数の最尤推定値を \$\Lambda_i(t; \hat{\boldsymbol{\theta}}_i)\$ (\$i = 1, 2, \dots, M\$) と表記する。ここで \$\hat{\boldsymbol{\theta}}_i\$ は \$i\$ 番目のモデルのパラメータの最尤推定値である。\$M\$ 個の単一モデルからひとつのモデルを選択しそれを予測に用いる場合、過去の観測データに最も適合したモデルを選択するために情報量基準を用いることが一般的である。多くの場合、以下に示す赤池情報量基準 (AIC) が最も小さくなるようなモデルを選択する。

$$\text{AIC} = -2 \ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{t}) + 2k. \quad (5)$$

ここで、\$k\$ は NHPP に基づいた SRM におけるモデルパラメータ数 (モデル自由度) である。表 1 の各 SRM では \$k = 2\$ もしくは \$k = 3\$ となっていることがわかる。

次に、モデル平均による予測モデルとしての線形加重結合モデル (LWCM) について述べる。\$M\$ 個の NHPP に基づいた SRM ではそれぞれ異なる故障発生時間分布を持つ。各構成モデルを示すモデル指標を \$i\$ (\$i = 1, 2, \dots, M\$) とし、これらの平均値関数を \$\Lambda_i(t; \boldsymbol{\theta}_i)\$ によって表す。\$i\$ 番目のモデルに対する重みを \$\{\omega_i \geq 0; i = 1, 2, \dots, M\}\$ で定義し、

\$\sum_{i=1}^M \omega_i = 1\$ とする。このとき、LWCM は次式で与えられる。

$$\begin{aligned} \Lambda(t; \omega_i, \hat{\boldsymbol{\theta}}_i, i = 1, 2, \dots, M) \\ = \sum_{i=1}^M \omega_i \Lambda_i(t; \hat{\boldsymbol{\theta}}_i). \end{aligned} \quad (6)$$

ここで、\$\hat{\boldsymbol{\theta}}_i\$ は \$i\$ 番目のモデルのパラメータに対する最尤推定である。

2.3 赤池ウェイト [24]

Lyu and Nikora [14] が LWCM における重みを直感的に与えていたのに対し、Hsu and Huang [11] は重み \$\omega_i\$ (\$i = 1, \dots, M\$) とモデルパラメータ \$\boldsymbol{\theta}_i\$ を最尤法で求めることを試みている。例えば表 1 の \$M = 11\$ の場合、式 (6) を単一モデルとみなした場合のモデル次元は \$k = 42\$ となり、対数尤度を最大化する超高次元最適化問題を解く必要がある。文献 [11] ではモデル数 \$M\$ を低めに設定し、各モデルにおけるパラメータ (例えば、表 1 の \$a\$ など) の共通化を行い、最尤推定問題を解くために遺伝的アルゴリズムのようなメタヒューリスティクスを適用している。文献 [22] では最尤推定原理から離れて、重み決定問題を結合加重の誤差最小化問題に置き換えて議論を展開している。

最近、呉ら [24] は赤池ウェイトと呼ばれる LWCM における重み決定法を導入し、単一の予測モデルを選択するよりも予測性能が向上することを示している。赤池ウェイトは次式によって定義される。

$$\begin{aligned} \omega_i &= \frac{\exp(-\Delta \text{AIC}_i/2)}{\sum_{j=1}^M \exp(-\Delta \text{AIC}_j/2)}, \\ \Delta \text{AIC}_i &= \text{AIC}_i - \text{AIC}_{\min}. \end{aligned} \quad (9)$$

ここで、\$\text{AIC}_i\$ は \$i\$ 番目の候補 SRM の AIC 値、\$\text{AIC}_{\min}\$ は候補 SRM の中で最も小さい AIC の値である。式 (9) で与えられた重みはモデル選択確率を意味しており、候補モデルのランキングを表している。これを式 (6) に代入すること

Algorithm 1 DAM (Training).**Input:**

Software time-domain data (training set) : $\mathbf{t} = \{t_1, t_2, \dots, t_n\}$.

M candidate models : $\{\Lambda_i(t; \boldsymbol{\theta})\}, i = 1, 2, \dots, M$.

Maximum training round P .

Initialization: Training round $p \leftarrow 1$,

Resampling constant $\{r_i \leftarrow 1\}_{i=1,2,\dots,n}$,

Resampled data $\mathbf{t}_p = \{t_i^{(p)} \leftarrow t_i\}_{i=1,2,\dots,n}$.

Procedure:**repeat**

- 1: Select the minimum AIC model as a candidate prediction model $\tilde{\Lambda}_p(t; \hat{\boldsymbol{\theta}}_p) \in \{\Lambda_i(t; \boldsymbol{\theta})\}$ among M candidate models with resampled data \mathbf{t}_p , where $\hat{\boldsymbol{\theta}}_p$ is the maximum likelihood estimate with \mathbf{t}_p .
- 2: Calculate the component of likelihood function in Eq.(3) CL_i :

$$CL_i = \lambda_p(t_i^{(p)}; \hat{\boldsymbol{\theta}}_p) \cdot e^{-[\Lambda_p(t_i^{(p)}; \hat{\boldsymbol{\theta}}_p) - \Lambda_p(t_{i-1}^{(p)}; \hat{\boldsymbol{\theta}}_p)]}, \quad (7)$$

where $\lambda_p(t_i; \hat{\boldsymbol{\theta}}_p) = \frac{d\Lambda_p(t; \hat{\boldsymbol{\theta}}_p)}{dt}$.

- 3: Calculate $MCL = \max\{CL_i\}, i = 1, 2, \dots, n$.
- 4: Define likelihood loss functions $g(l_i, D)$:

$$g(l_i, D) = \frac{|MCL - CL_i|}{\max\{|MCL - CL_j|\}_{j=1,2,\dots,n}}. \quad (8)$$

- 5: Calculate the loss $L_i = g(\Delta l_i, D)$ for sample $i = 1, 2, \dots, n$ with any loss function $g(l_i, D)$.
- 6: Calculate an average loss : $\bar{L} = \sum_{i=1}^n L_i \rho_i$, where $\rho_i = r_i / \sum r_j$.
- 7: Calculate the confidence $\beta_p = \frac{\bar{L}}{1 - \bar{L}}$.
- 8: Update the resampling constant $r_i \leftarrow r_i \cdot \beta_p^{(1-L_i)}, i = 1, 2, \dots, n$.
- 9: Sort and relabel the resampling constants $\{r_i\}$ and associated samples $\{t_i\}$ as $\{r_{(i)}\}$ and $\{t_{(i)}\}$, such that

$$r_{(1)} < r_{(2)} < \dots < r_{(n)}.$$
- 10: $p \leftarrow p + 1$.
- 11: The resampling training set \mathbf{t}_p is built as $\{t_{(p)}, t_{(p+1)}, \dots, t_{(n)}, t_{(n-p+1)}, t_{(n-p+2)}, \dots, t_{(n)}\}$,

where the $p - 1$ samples with the minimum resampling constant are removed and the $p - 1$ samples with the maximum resampling constant are added.

until $\bar{L} \leq 0.5$ or $p > P$

Output:

p candidate prediction models $\{\tilde{\Lambda}_j(t)\}$ and associated confidence $\{\beta_j\}, j = 1, 2, \dots, p$.

で、最尤推定法に準拠した予測モデルとしての LWCM を構成することが可能となる。

3 ブースティングに基づいたソフトウェア信頼性予測

3.1 既存の Ada ブースティングアルゴリズム

ここでは Li et al. [12] によって初めて提案されたソフトウェア信頼性予測のための Ada ブースティング技術を適用したモデル (以降、AMCM と略記) を紹介する. n 個の故障発生時間データ $\mathbf{t} = \{t_1, t_2, \dots, t_n\}$ が観測されているとする. この元データをリサンプリングし, $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_P$ という P 組の再標本データを生成する. ただし, P は経験的に前もって定める必要があり, 本稿では $P = 20$ とする. 文献 [12] では AMCM を次式のように定義している.

$$\begin{aligned} \tilde{\Lambda}(t; \omega_p, \hat{\boldsymbol{\theta}}_p, p = 1, 2, \dots, P) \\ = \sum_{p=1}^P \omega_p \Lambda_p(t; \hat{\boldsymbol{\theta}}_p). \end{aligned} \quad (10)$$

ここで, $\Lambda_p(\cdot)$ ($p = 1, 2, \dots, P$) はリサンプリングデータ \mathbf{t}_p に対して 11 個の候補モデルの中で AIC が最小であったモデルの平均値関数であり, $\hat{\boldsymbol{\theta}}_p$ はリサンプリングデータ \mathbf{t}_p を用いて推定されたモデルパラメータの最尤推定値である. 尚, 文献 [12] で用いられたリサンプリングの方法として, 単純なランダムサンプリングではなく前回のリサンプリングした結果 \mathbf{t}_{p-1} の影響を受けるリサンプリング技術を用いており, モデル重み ω_p の決定法についても工夫を凝らしていることに注意する.

3.2 決定論的 Ada ブースティングアルゴリズム

ここでは Drucker [5] によるむしろ標準的な AdaBoosting.R2 アルゴリズムに基づいた尤度

表 2: ソフトウェア故障発生時間データ.

データセット	総故障数	テスト時間 (CPU 時間)	出典	システムの特徴
DS1	73	5090	Project J5 [13]	リアルタイム指揮統制システム
DS2	38	233700	S10 [15]	リアルタイム指揮統制システム
DS3	41	4312598	S27 [15]	軍事用途
DS4	101	19572126	S17 [15]	リアルタイム指揮統制システム

Algorithm 2 DAM (inference).**Input:**

Software time-domain data : $\mathbf{t} = \{t_1, t_2, \dots, t_n\}$.
 Future testing time $t_u > t_n, u = n+1, n+2, \dots$
 P candidate prediction models $\{\tilde{\Lambda}_i(t)\}$ and associated confidence $\{\beta_i\}, i = 1, 2, \dots, P$.

Procedure:

- 1: Obtain P prediction values $\Delta\tilde{y}_p = \tilde{\Lambda}_p(t_u) - \tilde{\Lambda}_p(t_n), p = 1, 2, \dots, P$, for each candidate prediction model $\tilde{\Lambda}_p(t)$.
- 2: Each prediction value $\Delta\tilde{y}_p$ has an associated β_p .
- 3: Relabel $\{\Delta\tilde{y}_p\}$ while retaining the associated $\{\beta_p\}$ as $\{\Delta\tilde{y}_{(p)}\}$ and $\{\beta_{(p)}\}$, such that $\Delta\tilde{y}_{(1)} < \Delta\tilde{y}_{(2)} < \dots < \Delta\tilde{y}_{(P)}$.
- 4: Assign the confidence weight $c_{(p)} = \ln(\frac{1}{\beta_{(p)}}) / \sum \ln(\frac{1}{\beta_{(i)}})$ to the corresponding $\Delta\tilde{y}_{(p)}$.
- 5: Seek the final interval prediction value $\tilde{y}_{(p^*)}$ where the indicator p^* is given by

$$p^* = \min\{p \mid \sum_{i=1}^p c_{(i)} \geq \frac{1}{2} \sum_{i=1}^P c_{(i)}\} \quad (11)$$

(weighted median).

- 6: $\hat{y} = \tilde{y}_{(p^*)} + n$.

Output: \hat{y} , the expectation of cumulative number of software faults detected at testing time t_u .

による損失関数と決定論的なリサンプリング方法を取り入れた決定論的 Ada ブースティングアルゴリズムを提案し、ソフトウェア信頼性予測モデルに適用する。この方法を、AMCM に対比して DAM と表記する。アルゴリズムの詳細な説明は省略するが、Algorithm 1 と Algorithm 2 のように、訓練ステップと予測ステップの 2 段階から構成される。

まず、予測値である $\tilde{\Lambda}(t; \cdot)$ を $\Lambda_p(t; \cdot)$ ($p =$

$1, 2, \dots, P$) の重み付き中央値で定める。次に、標準的な AdaBoosting.R2 アルゴリズムでは再標本データの数 P が予め定められておらず、再標本データ t_p の推定値とデータ t の損失関数によって決まる。さらに、再標本データ t_p を生成するために Li et al. [12] と異なるリサンプリング方法を使っている。AMCM にはもう一つの問題がある。AMCM は確率的なリサンプリングによる手法に基づいており、予測性能はリサンプリングの結果に大きく依存する。

4 数値実験

4.1 実験の設定

本稿では、表 2 に示す 4 件のソフトウェア開発プロジェクトで観測された故障計数データに対して、計 7 種類の予測手法の性能を評価した。各データの観測開始から 20%, 50%, 80% 時点までのデータを訓練データとして用い、残りの期間における総期待故障数を予測する。これらの予測時点をそれぞれソフトウェアテストの初期段階、中期段階、後期段階とみなすことで、テストの進捗状況に応じた予測性能を調べる。

予測性能の評価には、次のような予測平均絶対誤差 (PMAE) を用いる。

$$\text{PMAE} = \frac{1}{l} \sum_{i=1}^l |(n+i) - \tilde{\Lambda}(t_{n+i})|. \quad (12)$$

ここで、 $\{t_1, t_2, \dots, t_n, \dots, t_{n+l}\}$ は故障検出時間データを表し、 l は予測項数、 $\tilde{\Lambda}(t_{n+i})$ は任意の手法により予測された時刻 t_{n+i} における累積故障数である。

4.2 予測性能の評価

表3は、AICを最小にする単一モデルに基づいた予測結果と赤池ウェイトとブースティングに基づいて算出された予測性能を比較した結果を示す。“AIC weight”は赤池ウェイトに基づいて算出されたPMAEを示す。ここで、“Minimum PMAE Model”は、11種類のNHPPに基づいたSRMの中で事後的に最も優れた予測性能を示すモデルPMAEである。また、文献[12]において定義された損失関数 Lineararm Square, Expを用いたAMCMによる予測結果をAMCM(\cdot)で表す。黄色で網掛けされた部分は、AICに基づいて選択した単一モデルが“Minimum PMAE Model”に一致したことを示しており、表中の赤字の値は、それぞれの手法が基準となるAICに基づいた単一モデルよりも良い予測性能を示した場合を表している。特に太字は該当するモデルが“Minimum PMAE Model”をPMAEの観点から上回ったことを示す。

結果として、初期テスト段階においては、AMCM (Linear), AMCM (Square) およびDAMが全4ケース中3件で予測性能の改善を示したが、LWCMとAMCM (Exp)は2件で改善し、赤池ウェイトは全体として1件の改善に留まり、最も悪い結果となった。テスト段階の中期においても、初期テスト段階と似たような結果が観測される。ここで、注意すべき点は、LWCMとAMCM (Square)がDS1において、AMCM (Exp)がDS3において、DAMがDS2において、各々Minimum PMAE Model

よりも高い予測性能を示している。これは同じ候補モデルでも、重みの決め方によっては任意の単一モデルより高い予測性能を成し遂げることを意味し、線型結合モデルがPMAEの

観点から従来の単一モデルを凌駕するポテンシャルを持つ証拠である。テスト段階の後期では初期と中期段階のように予測性能を改善できていないが、AMCM (Square)が4件中2件で予測性能の改善を示し、さらにDS3ではMinimum PMAE Modelよりも高い予測性能を表した。

まとめると、決定論的Adaブースティングを適用すると、テスト初期および中期段階において、4件中3件において単一モデルによる予測性能を上回ることを確認した。既存のブースティング技術もほぼ同じ予測性能を表しているが、これらの結果は確率的なサンプリングの結果に依存するため、予測性能について不安定リスクを含んでいる。赤池ウェイトとLWCMでは、どのテスト段階においても単一モデルより優れた性能を示すことはなかった。これより、本稿で提案した決定論的Adaブースティングの方法は従来手法に比べて有意な予測性能の向上が期待できることを結論付けることができる。

5 結論

本稿では、ソフトウェア信頼性予測のためにブースティング技術を適用することに着目し、決定論的Adaブースティングを用いることを提案した。AICを最小にする単一モデルと従前のAdaブースティングアルゴリズムを適用したモデルと比較することで、初期および中期のテスト段階において、シングルモデルアプローチに比べ有意な予測性能の向上を実現した。今後は、オープンソースプロジェクトに対する実験や、その他のアンサンブル学習手法を用いたソフトウェア信頼性予測手法の開発に取り組む予定である。

6 謝辞

本研究の一部はJST 科学技術イノベーション創出に向けた大学フェローシップ創設事業

表 3: 各テスト段階での予測性能 (PMAE) .

Early Software Testing Phase (20%)				
Data Set	DS1	DS2	DS3	DS4
Minimum PMAE Model	8.85 (LogNormal)	2.27 (Pareto)	3.77 (LogNormal)	16.03 (LogEVMax)
Minimum AIC Model	29.24 (TruncLogist)	8.51 (LogEVMax)	12.38 (Exp)	108.35 (Exp)
LWCM	25.09	10.95	40.59	72.48
AMCM (Linear)	12.59	11.20	6.47	25.26
AMCM (Square)	11.98	6.04	12.68	18.57
AMCM (Exp)	38.37	7.38	6.22	114.47
DAM	23.47	2.45	12.38	108.35
AIC weight	26.18	11.01	14.16	142.69
Middle Software Testing Phase (50%)				
Data Set	DS1	DS2	DS3	DS4
Minimum PMAE Model	14.44 (LogEVMax)	3.01 (LogEVMax)	41.78 (Pareto)	24.65 (TruncEVMin)
Minimum AIC Model	17.14 (Exp)	3.01 (LogEVMax)	131.85 (TruncEVMin)	207.70 (Exp)
LWCM	14.28	5.60	135.60	54.87
AMCM (Linear)	15.98	9.81	41.79	96.54
AMCM (Square)	9.16	9.98	80.69	36.18
AMCM (Exp)	22.07	74.70	6.54	49.30
DAM	16.49	2.63	73.99	212.15
AIC weight	16.25	3.86	178.01	138.47
Late Software Testing Phase (80%)				
Data Set	DS1	DS2	DS3	DS4
Minimum PMAE Model	2.94 (LogEVMax)	1.15 (LogEVMax)	2.28 (LogEVMax)	6.05 (LogEVMax)
Minimum AIC Model	4.53 (Gamma)	1.15 (LogEVMax)	2.63 (LogNormal)	9.93 (Pareto)
LWCM	2.19	2.47	3.06	9.26
AMCM (Linear)	20.60	1.77	2.55	9.78
AMCM (Square)	28.92	2.03	1.45	9.83
AMCM (Exp)	16.49	7.68	4.93	9.72
DAM	6.32	1.27	2.63	9.78
AIC weight	4.77	1.57	2.89	9.80

JPMJFS2129 の支援を受けたものである。

参考文献

- [1] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood (1986), Evaluation of competing software reliability predictions, *IEEE Transactions on Software Engineering*, vol. SE-12, no. 9, pp. 950–967.
- [2] J. A. Achcar, D. K. Dey, and M. Nivertthi (1998), A Bayesian approach using non-homogeneous Poisson processes for software reliability models, In *Frontiers in Reliability*, A. P. Basu, S. K. Basu and S. Mukhopadyyay (eds.), pp. 1–18, World Scientific, Singapore.
- [3] H. Akaike (1973), Information theory and an extension of the maximum likelihood principle, In *Proceedings of the 2nd International Symposium on Information Theory*, B. N. Petrov and F. Caski (eds.), pp. 267–281, Akademiai Kiado, Budapest.
- [4] H. Bozdogan (1987), Model selection and Akaike’s information criterion (AIC): The general theory and its analytical extensions, *Psychometrika*, vol. 52, pp. 345–370.
- [5] H. Drucker (1997), Improving regressors using boosting techniques, In *Proceedings of the 14th International Conference on Machine Learning (ICML 1997)*, D. H. Fisher (ed.), pp. 107–115, Morgan Kaufmann, Nashville.
- [6] D. Fletcher (2018), *Model Averaging*, Springer Berlin, Heidelberg.
- [7] Y. Freund, and R. E. Schapire (1995), A decision-theoretic generalization of online learning and an application to boosting, In *Proceedings of the 2nd European Conference on Computational Learning Theory (ECCLT-1995)*, pp. 23–37, Springer-Verlag, Barcelona.
- [8] A. L. Goel, and K. Okumoto (1979), Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211.
- [9] A. L. Goel (1985), Software reliability models: assumptions, limitations and applicability, *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1411–1423.
- [10] S. S. Gokhale, and K. S. Trivedi (1998), Log-logistic software reliability growth model, In *Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium (HASE-1998)*, pp. 34–41, IEEE CPS.
- [11] C.-J. Hsu, and C.-Y. Huang (2014), Optimal weighted combinational models for software reliability estimation and analysis, *IEEE Transactions on Reliability*, vol. 63, no. 3, pp. 731–749.
- [12] H. Li, M. Zeng, M. Lu, X. Hu, Z. Li (2011), Adaboosting-based dynamic weighted combination of software reliability growth models, *Quality and Reliability Engineering International*, vol. 28, no. 1, pp. 67–84.

- [13] M. R. Lyu (ed.) (1996), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York.
- [14] M. R. Lyu, and A. Nikora (1992), Applying reliability models more effectively, *IEEE Software*, vol. 9, no. 4, pp. 43–52.
- [15] J. D. Musa (1979), *Software Reliability Data*, Technical Report in Rome Air Development Center, New Jersey.
- [16] M. Ohba (1984), Inflection S-shaped software reliability growth model, In *Stochastic Models in Reliability Theory*, S. Osaki and Y. Hatoyama (eds.), pp. 144–162, Springer, Berlin/Heidelberg.
- [17] K. Ohishi, H. Okamura, and T. Dohi (2009), Gompertz software reliability model: Estimation algorithm and empirical validation, *Journal of Systems and Software*, vol. 82, no.3, pp. 535–543.
- [18] H. Okamura, and T. Dohi (2013), SRATS: software reliability assessment tool on spreadsheet, In *Proceedings of the IEEE 24th International Symposium on Software Reliability Engineering (ISSRE-2013)*, pp. 100–107, IEEE CPS.
- [19] H. Okamura, T. Dohi, and S. Osaki (2013), Software reliability growth models with normal failure time distributions, *Reliability Engineering and System Safety*, vol. 116, pp. 135–141.
- [20] G. Schwarz (1978), Estimating the dimension of a model, *Annals of Statistics*, vol. 6, no. 2, pp. 461–464.
- [21] N. Sugiura (1978), Further analysis of the data by Akaike’s information criterion and the finite corrections, *Communications in Statistics*, vol. A7, no. 1, pp. 13–26.
- [22] C.-Y. Wu, and C.-Y. Huang (2021), A study of incorporation of deep learning into software reliability modeling and assessment, *IEEE Transactions on Reliability*, vol. 70, no. 4, pp. 1621–1640.
- [23] J. Wu, K. Daido, T. Dohi, and H. Okamura (2024), Yet another least squares estimation in NHPP-based software reliability models with grouped data, In *Proceedings of 14th International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE-2024)*, vol. 2024, no. 12, pp. 205–211, IET.
- [24] 吳敬馳, 土肥正, 岡村寛之 (2024), マルチモデル推測に基づいたソフトウェア信頼性予測精度の向上に向けて, ソフトウェアシンポジウム 2024 論文集, pp. 102–111, ソフトウェア技術者協会.
- [25] S. Yamada, M. Ohba, and S. Osaki (1983), S-shaped reliability growth modeling for software error detection, *IEEE Transactions on Reliability*, vol. R-32, no. 5, pp. 475–484.
- [26] M. Zhao, and M. Xie (1996), On maximum likelihood estimation for a general non-homogeneous Poisson process, *Scandinavian Journal of Statistics*, vol. 23, no. 4, pp. 597–607.