Alternative AdaBoosting Approaches For NHPP-based Software Reliability Models

呉　敬馳、土肥　正、鄭　俊俊、岡村寛之
広島大学大学院先進理工系科学研究科 *
Jingchi Wu, Tadashi Dohi, Junjun Zheng, Hiroyuki Okamura
Graduate School of Advanced Science and Engineering
Hiroshima University

1 Introduction

Quantitative software reliability, which is defined as a probability that no software fault is detected during a specified time interval under given conditions, is one of the key measures in software quality management and software release decision. Also, it is known that predicting software reliability accurately is equivalent to predicting the number of software faults detected in the long term. To describe the cumulative number of software faults in the system test, non-homogeneous Poisson process (NHPP)-based software reliability models (SRMs) have been widely used in practice. Since Goel and Okumoto [5] developed the seminal NHPP-based SRM with exponential fault-detection time distribution, many authors proposed a number of NHPP-based SRMs with different fault-detection time distributions, such as Pareto distribution [1], lognormal and truncated normal distributions [2, 13], log-logistic and truncated logistic distributions [7, 11], Gompertz and its related extreme-type distributions [6, 12], gamma distribution [16, 18].

Once the model parameters of the candidate NHPP-based SRMs are estimated, it is common to select the best goodness-of-fit SRM uniquely to the historical fault-count data experienced before, and to use it for prediction in the future as *plug-in predictor*. However, the best goodness-of-fit SRM is not always the best prediction model for the future fault counts. In other words, the plug-in prediction implicitly assumes that the probability distributions of software fault counts are exactly same in the past and future, and may often result over-fitting to the past observation.

〒739-8527 東広島市鏡山 1-4-1

To improve the generalization ability by the single use of SRM based on the plug-in predictor, Lyu and Nikora [10] proposed four linearly weighted combination models (LWCMs), where the model weights are all equal or determined by three heuristic/intuitive methods.

Since the full maximum likelihood estimation for LWCMs is quite expensive in computation, our next concern is to develop lightweighted methods to make the prediction of the number of software faults in the LWCMs. Li et al. [9] proposed an AdaBoosting-based LWCM, which was available for only the software fault time-interval (grouped) data, under a very specific resampling scheme. Though it is claimed that the methods employed were motivated by the so-called AdaBoosting.R2 algorithm proposed by Drucker [4], they were rather different from the original AdaBoosting algorithm, and failed to generate sufficient resampling data with significant statistical features. Wu et al. [15] also applied the AdaBoosting.R2 algorithm [4] to NHPP-based LWCM, and attempted to improve the Li et al.'s algorithm [9] by updating the resampling weights and introducing the *weighted median predictor*. However, their method has not been empirically validated with the actual software fault-count data.

The main contribution of this paper is to refine the AdaBoosting approaches for NHPP-based SRMs. Our methods enable to update the resampling weights more appropriately, thereby the issue in [9, 15] with poor resampling scheme is overcome, by taking account of the probability law of the NHPP-based SRMs.

2 Preliminaries

2.1 NHPP-based SRMs

Let $\{N(t), t \geq 0\}$ be a stochastic point process that denotes the cumulative number of software faults detected up to time t during the software system testing. Suppose that $N(t)$ is a non-homogeneous Poisson process (NHPP) with the mean value function $\Lambda(t;\boldsymbol{\theta})$ which is absolutely continuous and a non-decreasing function of t, where $\boldsymbol{\theta}$ is the model parameter vector. Then, the probability mass function of $N(t)$ is given by

$$\Pr\{N(t) = n\} = \frac{\{\Lambda(t;\boldsymbol{\theta})\}^n e^{-\Lambda(t;\boldsymbol{\theta})}}{n!}, \tag{1}$$

where $\lambda(t;\boldsymbol{\theta}) = d\Lambda(t;\boldsymbol{\theta})/dt$ is called the intensity function. It is known to hold that $\mathrm{E}\left[N(t)\right] = \mathrm{Var}\left[N(t)\right] = \Lambda(t;\boldsymbol{\theta})$, so the NHPP-based SRM is uniquely determined by $\Lambda(t;\boldsymbol{\theta})$ or $\lambda(t;\boldsymbol{\theta})$. Table 1. presents the mean value functions of the eleven representative NHPP-based SRMs implemented in SRATS [14]. From a few algebraic manipulations, it is straightforward to derive the quantitative software reliability $R(t_u, t_s;\boldsymbol{\theta})$, which is the probability that no software fault is detected during the time interval $(t_u, t_s]$ after releasing the software at time t_u, as

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

Table 1. Representative eleven NHPP-based SRMs.

Mean Value Function	$\Lambda(t;\boldsymbol{\theta})$ $(\boldsymbol{\theta}=a,b,c>0)$
Exp [5]	$\Lambda(t;\boldsymbol{\theta})=a(1-e^{-bt})$
Gamma [16, 18]	$\Lambda(t;\boldsymbol{\theta})=a\int_0^t \frac{c^b s^{b-1}e^{-cs}}{\Gamma(b)}ds$
Pareto [1]	$\Lambda(t;\boldsymbol{\theta})=a\left(1-\left(\frac{c}{t+c}\right)^b\right)$
TruncNormal [13]	$\Lambda(t;\boldsymbol{\theta})=a\frac{F(t)-F(0)}{1-F(0)}$, $F(t)=\frac{1}{\sqrt{2\pi}b}\int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}}ds$
LogNormal [2, 13]	$\Lambda(t;\boldsymbol{\theta})=a\frac{1}{\sqrt{2\pi}b}\int_{-\infty}^{\log(t)} e^{-\frac{(s-c)^2}{2b^2}}ds$
TruncLogist [11]	$\Lambda(t;\boldsymbol{\theta})=a\frac{F(t)-F(0)}{1-F(0)}$, $F(t)=\frac{1}{1+e^{-\frac{t-c}{b}}}$
LogLogist [7]	$\Lambda(t;\boldsymbol{\theta})=a\frac{1}{1+e^{-\frac{\log(t)-c}{b}}}$
TruncEVMax [12]	$\Lambda(t;\boldsymbol{\theta})=a\frac{F(t)-F(0)}{1-F(0)}$, $F(t)=e^{-e^{-\frac{t-c}{b}}}$
LogEVMax [12]	$\Lambda(t;\boldsymbol{\theta})=ae^{-e^{-\frac{\log(t)-c}{b}}}$
TruncEVMin [12]	$\Lambda(t;\boldsymbol{\theta})=a\frac{F(0)-F(-t)}{F(0)}$, $F(t)=e^{-e^{-\frac{t-c}{b}}}$
LogEVMin [6, 12]	$\Lambda(t;\boldsymbol{\theta})=a\left(1-e^{-e^{-\frac{-\log(t)-c}{b}}}\right)$

where $t_s - t_u$ denotes the prediction length. It can be understood from Eq.(2) that predicting the quantitative software reliability is equivalent to predicting the mean value function in the NHPP-based SRM.

2.2 Plug-in Prediction

Once the mean value function $\Lambda(t;\boldsymbol{\theta})$ is given, it is necessary to estimate the unknown model parameters $\boldsymbol{\theta}$ statistically from the software fault-count data. Although least squares estimation is sometimes used in the parameter estimation, this is completely irrelevant to the probability law in Eq.(1). Hence, the commonly used technique to estimate the model parameters $\boldsymbol{\theta}$ in software reliability is maximum likelihood estimation.

Suppose that the grouped data, $\boldsymbol{x} = \{(t_1, x_1), (t_2, x_2), \ldots, (t_n, x_n)\}$, are available, where x_i is the cumulative number of software faults detected up to the i-th calendar time t_i. The log-

likelihood function with the grouped data is given by

$$\ln\mathcal{L}(\boldsymbol{\theta};\boldsymbol{x}) = \sum_{i=1}^{n}\{(x_i - x_{i-1})\ln\left[\Lambda\left(t_i;\boldsymbol{\theta}\right) - \Lambda\left(t_{i-1};\boldsymbol{\theta}\right)\right] \\ -\ln\left[(x_i - x_{i-1})!\right]\} - \Lambda(t_n;\boldsymbol{\theta}). \tag{3}$$

Then, the maximum likelihood estimates (MLEs) of the model parameters $\boldsymbol{\theta}$ are defined by

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln\mathcal{L}(\boldsymbol{\theta};\boldsymbol{t} \text{ or } \boldsymbol{x}). \tag{4}$$

Since the maximization problems in Eq.(4) are ill-posed nonlinear optimization problems (see [17]) in general, we need to apply any efficient computation algorithm to obtain the MLEs.

Similar to Table 1., consider ϕ candidate (component) NHPP-based SRMs with the mean value functions $\Lambda_m(t;\tilde{\boldsymbol{\theta}}_m)$ $(m = 1, 2, \ldots, \phi)$. It is common to select the best SRM, taking account of the information criteria, which could fit the underlying data set $\boldsymbol{t}$ or $\boldsymbol{x}$. The Akaike information criterion (AIC) [3] is the most well-known information criterion, where

$$\text{AIC} = -2 \cdot \ln\mathcal{L}(\tilde{\boldsymbol{\theta}};\boldsymbol{t} \text{ or } \boldsymbol{x}) + 2\xi \tag{5}$$

with the number of free parameters in the NHPP-based SRM, ξ. Smaller AIC means better goodness-of-fit performance. Once the best fitting model $\Lambda_m^*(t;\tilde{\boldsymbol{\theta}}_m)$ with smallest AIC was selected, we apply $m-$th NHPP-based SRM to the prediction, and obtain the plug-in predictiors; $\Lambda_m^*(t_s;\tilde{\boldsymbol{\theta}}_m)$ and $R(t_u, t_s;\tilde{\boldsymbol{\theta}}_m)$.

3 Adaboosting-based SRMs

3.1 Existing Methods

Li et al. [9] first proposed the AdaBoosting-based linearly weighted combination model (LWCM) for NHPP-based SRMs with grouped data. For ϕ candidate NHPP-based SRMs and the grouped data $\boldsymbol{x}$, ζ resampled data are generated from $\boldsymbol{x}$. The resampled grouped data, $\hat{\boldsymbol{x}}_k = \left\{\left(\hat{t}_1, \hat{x}_1\right), \left(\hat{t}_2, \hat{x}_2\right), \ldots, \left(\hat{t}_{n_k}, \hat{x}_{n_k}\right)\right\}$, with n_k $(\leq n)$ and $\hat{x}_{n_k}$ $(\leq x_n)$, are generated sequentially by repeating the random sampling without replacement n times, according to its corresponding resampling weights $\boldsymbol{r} = \{r_1, r_2, \ldots, r_n\}$, where each data point (t_i, x_i) is picked with normalized probability $r_i/\delta(\boldsymbol{r})$ with $\delta(\boldsymbol{r}) = \sum_{v=1}^{n} r_v$. When the data point was picked, it is discarded. For each grouped resampled data, $\hat{\boldsymbol{x}}_k$, the best fitting model as a weak learner, $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$, among ϕ candidate NHPP-based SRMs, can be obtained. Define $\mathcal{L}_i^k$ as a measure to evaluate the goodness-of-fit between the best fitting model $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$ and the original data $\boldsymbol{x}$ at testing time t_i [9]. The details of $\mathcal{L}_i^k$ are given in Section 4. Then the resampling weights $\boldsymbol{r}$ are updated by

$$r_i = r_i \cdot \frac{\delta(\boldsymbol{r})\mathcal{L}_i^k}{\sum_{v=1}^{n} r_v \mathcal{L}_v^k} \tag{6}$$

in the k-th resampling step. Then, the AdaBoosting-based LWCM in [9] is given by

$$\begin{aligned}\Lambda_{a_1}(t;\zeta,\boldsymbol{\beta},\tilde{\boldsymbol{\theta}}_{a_1}) &= \sum_{k=1}^{\zeta}\beta_k\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*),\\ \boldsymbol{\beta} &= \{\beta_k; k=1,2,\dots\zeta\},\\ \tilde{\boldsymbol{\theta}}_{a_1} &= \{\tilde{\boldsymbol{\theta}}_k^*; k=1,2,\dots\zeta\},\end{aligned} \tag{7}$$

where

$$\beta_k = \frac{\ln\left(\frac{1-\bar{\mathcal{L}}_k}{\bar{\mathcal{L}}_k}\right)}{\sum_{l=1}^{\zeta}\ln\left(\frac{1-\bar{\mathcal{L}}_l}{\bar{\mathcal{L}}_l}\right)}, \tag{8}$$

$$\bar{\mathcal{L}}_k = \sum_{i=1}^{n}\frac{r_i}{\delta(\boldsymbol{r})}\cdot\mathcal{L}_i^k. \tag{9}$$

3.2 Thinning-like Resampling Algorithm

Here, we propose a refined resampling method for the adaboosting approach. The fundamental idea is based on the well-known *thinning* of an NHPP by Lewis [8].

Proposition [8]: Consider an NHPP $\{N(t); t\geq 0\}$ *with intensity function* $\lambda(t;\boldsymbol{\theta})$ *and mean value function* $\Lambda(t;\boldsymbol{\theta}) = \int_0^t \lambda(x;\boldsymbol{\theta})dx$, *so that the number of points,* $N(t_n)$, *in a fixed interval* $(0,t_n]$ *has a Poisson distribution with parameter* $\Lambda(t_n;\boldsymbol{\theta}) - \Lambda(0;\boldsymbol{\theta})$. *Let* $T_1, T_2, \dots, T_{N(t_n)}$ *be the event-occurrence points of the process in the interval* $(0,t_n]$. *Assume that* $0\leq t\leq t_n$, $\underline{\lambda}(t;\underline{\boldsymbol{\theta}}) \leq \lambda(t;\boldsymbol{\theta})$ *with respective model parameters* $\underline{\boldsymbol{\theta}}$ *and* $\boldsymbol{\theta}$. *For* $i = 1,2,\dots,N(t_n)$, *delete the point* T_i *with probability*

$$1 - \underline{\lambda}(T_i;\underline{\boldsymbol{\theta}})/\lambda(T_i;\boldsymbol{\theta}). \tag{10}$$

Then, the remaining points from an NHPP $\{\underline{N}(t); t\geq 0\}$ *with intensity function* $\underline{\lambda}(t;\underline{\boldsymbol{\theta}})$ *in the interval* $(0,t_n]$.

In our problem, we suppose that the original time-domain data $\boldsymbol{t}$ are realizations of an NHPP $\{N(t); t\geq 0\}$ with intensity function $\lambda(t;\boldsymbol{\theta})$. From the above proposition, it can be seen that the resampled time-domain data, $\hat{\boldsymbol{t}}_k$, which are generated by deleting each data point in the original data $\boldsymbol{t}$ with probability $1/n$, can be viewed as realizations of an NHPP $\{N^*(t); t\geq 0\}$ with intensity function $\lambda^*(t;\boldsymbol{\theta}^*)$ with model parameter $\boldsymbol{\theta}^*$. From Eq.(10), it is clear that $\lambda^*(t;\boldsymbol{\theta}^*)$ is the infimum of $\lambda(t;\boldsymbol{\theta})$, so we set

$$1 - \frac{\lambda^*(t;\boldsymbol{\theta}^*)}{\lambda(t;\boldsymbol{\theta})} = \frac{1}{n}. \tag{11}$$

Remind that the weak learner, $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$, is an estimate of $\Lambda^*(t;\boldsymbol{\theta}^*) = \int_0^t \lambda^*(x;\boldsymbol{\theta}^*)dx$ with resampled data $\hat{\boldsymbol{t}}_k$. From a simple algebraic manipulation, the bias for the weak learner $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$ is corrected as

$$\Lambda_{k(c)}^*(t;\tilde{\boldsymbol{\theta}}_k^*, n) = \big(1 + \frac{1}{n-1}\big)\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*). \tag{12}$$

This result can be also applied to the resampled grouped data $\hat{\boldsymbol{x}}$ as

$$\Lambda_{k(c)}^*(t;\tilde{\boldsymbol{\theta}}_k^*, x_n) = \big(1 + \frac{1}{x_n-1}\big)\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*), \tag{13}$$

where x_n is the cumulative number of software faults detected up to the testing time t_n in the original grouped data $\boldsymbol{x}$.

4 Alternative AdaBoosting Approaches for NHPP-based SRMs

From Eqs.(12) and (13), we refine the AdaBoosting-based LWCM with the *weighted thinning resampling*. Suppose that the time-domain data, $\boldsymbol{t} = \{t_1, t_2, \ldots, t_n\}$, and the resampling weights, $\boldsymbol{r}^t = \{r_1, r_2, \ldots, r_n\}$, are available. The resampled data, $\hat{\boldsymbol{t}}_k$, are generated by deleting each data point t_i with probability $r_i/\delta(\boldsymbol{r})$, and are formed with the remaining data point. Let $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$ denote the best fitting model among ϕ candidate NHPP-based SRMs on the resampled data $\hat{\boldsymbol{t}}_k$. The weak learner is obtained by $\Lambda_{k(c)}^*(t;\tilde{\boldsymbol{\theta}}_k, n) = \big(1+\frac{1}{n-1}\big)\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$. On the other hand, suppose that the grouped data, $\boldsymbol{x} = \{(t_1, x_1), (t_2, x_2), \ldots, (t_n, x_n)\}$, and the resampling weights, $\boldsymbol{r}^g = \{r_1, r_2, \ldots, r_{x_n}\}$, are given. For n grouped data with increments $x_i - x_{i-1}$, we label those data points as $\boldsymbol{t}' = \left\{t_1' = t_1^{(1)}, \ldots, t_{x_n}' = t_1^{(n)}\right\}$, where $t_\eta^{(i)}$ $(\eta = 1, 2, \ldots, x_i - x_{i-1})$ denotes the η-th data point in the i-th slot. We delete each point t_j' with probability $r_j / \sum_{\mu=1}^{x_n} r_\mu$ $(j = 1, 2, \ldots, x_n)$. Then, the resampled grouped data, $\hat{\boldsymbol{x}}_k = \{(t_1, \hat{x}_1), (t_2, \hat{x}_2), \ldots, (t_n, \hat{x}_n)\}$, are generated with the remaining data points, where $\hat{x}_i - \hat{x}_{i-1}$ is the remaining number of point in the i-th slot. Similar to the case with time-domain data, for the best fitting model, $\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$, the weak learner is given by $\Lambda_{k(c)}^*(t;\tilde{\boldsymbol{\theta}}_k, x_n) = \big(1 + \frac{1}{x_n-1}\big)\Lambda_k^*(t;\tilde{\boldsymbol{\theta}}_k^*)$.

Dissimilar to Li et al. [9], the fundamental idea in our AdaBoosting approaches is to apply weighted thinning resampling instead of the common random sampling and to update the resampling weights, $\boldsymbol{r}$, by a different way, because solving the recursive formula in Eq.(6) can be quickly made, but the result is poor. For instance, consider an extreme case of $\boldsymbol{r} = \{1, 0, \ldots, 0\}$. Then, it is apparent to see that the resampled data sequence in [9] becomes too short and tends to lose the statistical significant properties of the original data.

Let $\boldsymbol{r}^g = \{r_1 = 1, r_2 = 1, \ldots, r_{x_n} = 1\}$ be the resampling weights in the beginning of training round, say, $\rho = 0$, for grouped data. Let $\mathcal{K}$ be the maximum training round and ϕ candidate NHPP-based SRMs with $\Lambda_m(t;\boldsymbol{\theta}_m)$ $(m = 1, 2, \ldots, \phi)$ are given. Then, we repeat the following steps while the average loss, $\bar{\mathcal{L}}_\rho$, becomes less than a tolerance level, e.g., 0.1, or the training round becomes $\rho = \mathcal{K}$, whichever occurs first.

- Step 1: Obtain the resampled data, $\hat{\boldsymbol{x}}_\rho$, with resampling weights $\boldsymbol{r}^g$, by applying the weighted thinning resampling in Section 3.2 at the ρ-th training round.
- Step 2: Seek the weak learner, $\Lambda^*_{\rho(c)}(t;\tilde{\boldsymbol{\theta}}^*_\rho, x_n) = \big(1+1/(x_n-1)\big)\Lambda^*_\rho(t;\tilde{\boldsymbol{\theta}}^*_\rho)$, for resampled grouped data, where $\Lambda^*_\rho(t;\tilde{\boldsymbol{\theta}}^*_\rho)$ is the best fitting NHPP-based SRM with the minimum AIC, among ϕ candidate models with resampled data, $\hat{\boldsymbol{x}}_\rho$, and $\tilde{\boldsymbol{\theta}}^*_\rho$ are the MLEs of model parameters with $\hat{\boldsymbol{x}}_\rho$.
- Step 3: Calculate the loss function, $\mathcal{L}^\rho_j$, by the weak learner, $\Lambda^*_{\rho(c)}(t;\tilde{\boldsymbol{\theta}}_\rho, x_n)$, for grouped data. Define the error functions $\bar{e}_j$ as

$$\bar{e}_j = \left|\Lambda^*_{\rho(c)}(t_i;\tilde{\boldsymbol{\theta}}^*_\rho, x_n) - x_i\right| / (x_i - x_{i-1}), \tag{14}$$

  where $j = x_{i-1}+1, x_{i-1}+2, \ldots, x_i$ and $i = 1, 2, \ldots, n$. Then, we have the maximum error functions for $\bar{e}_j$ by

$$\gamma = \sup\{\bar{e}_j\}\ (j = 1, 2, \ldots, x_n). \tag{15}$$

  Next, we define the following three loss functions with different sensitivities to the maximum error γ [9] for grouped data.

$$\text{Linear loss function}: \mathcal{L}^\rho_j = \frac{\bar{e}_j}{\gamma}. \tag{16}$$

$$\text{Square loss function}: \mathcal{L}^\rho_j = \left(\frac{\bar{e}_j}{\gamma}\right)^2. \tag{17}$$

$$\text{Exponential loss function}: \mathcal{L}^\rho_j = 1 - \exp\left(-\frac{\bar{e}_j}{\gamma}\right). \tag{18}$$

- Step 4: Calculate the average loss function for grouped data by

$$\bar{\mathcal{L}}_\rho = \sum_{j=1}^{x_n} \mathcal{L}^\rho_j \cdot \frac{r_j}{\sum_{\mu=1}^{x_n} r_\mu}. \tag{19}$$

- Step 5: Update the resampling weight $r_j \in \boldsymbol{r}^g$ as follows.

$$r_j = r_j \left(\frac{\bar{\mathcal{L}}_\rho}{1-\bar{\mathcal{L}}_\rho}\right)^{1-\mathcal{L}_i}. \tag{20}$$

- Step 6: Set the training round $\rho = \rho + 1$.

After terminating the above steps, we obtain ρ^* weak learners. Then our refined AdaBoosting-based LWCM for grouped data is given by

$$\begin{aligned}&\Lambda_{a_2}(t;\rho^*, \boldsymbol{\beta}_{a_2}, \tilde{\boldsymbol{\theta}}_{a_2}, x_n)\\ &= \sum_{\rho=1}^{\rho^*} \beta_\rho \Lambda^*_{\rho(c)}(t;\tilde{\boldsymbol{\theta}}^*_\rho, x_n),\end{aligned} \tag{21}$$

where $\tilde{\boldsymbol{\theta}}_{a_2} = \{\tilde{\boldsymbol{\theta}}^*_\rho; \rho = 1, 2, \dots \rho^*\}$, $\boldsymbol{\beta}_{a_2} = \{\beta_\rho; \rho = 1, 2, \dots \rho^*\}$, and β_ρ is given by

$$\beta_\rho = \frac{\ln\left(\frac{1-\bar{\mathcal{L}}_\rho}{\bar{\mathcal{L}}_\rho}\right)}{\sum_{q=1}^{\rho^*} \ln\left(\frac{1-\bar{\mathcal{L}}_q}{\bar{\mathcal{L}}_q}\right)}. \tag{22}$$

In the above AdaBoosting-based LWCMs with two kinds of fault-count data, the error function, $\bar{e}_j$ in Eq.(14), is based on the cumulative number of software fault counts. However, the above error functions can be defined by removing the cumulative effect. Define

$$\bar{e}_j = \frac{\left|\left[\Lambda^*_\rho(t_i; \tilde{\boldsymbol{\theta}}^*_\rho) - \Lambda^*_\rho(t_{i-1}; \tilde{\boldsymbol{\theta}}^*_\rho)\right] - (x_i - x_{i-1})\right|}{x_i - x_{i-1}}. \tag{23}$$

For each case with/without cumulative effects, we can obtain different ρ^* weak learners; $\Lambda^*_{\rho(c)}(t; \tilde{\boldsymbol{\theta}}^*_\rho, x_n)$ for grouped data.

Table 2. Summary of software fault-count data.

Data Set	Total Number of Faults	Testing Length (month)	Project Name	System description
DS1	3903	89	zig	General-purpose programming language and toolchain
DS2	853	67	ccxt	A multi-programming language cryptocurrency trading API
DS3	254	77	styled-components	A powerful CSS-based framework
DS4	1536	137	netty	An event-driven network application framework

5 Numerical Experiments

5.1 Setup

We investigate the predictive performances of our refined ensemble-based methods and compare them with the existing ones including the plug-in predictors. Table 2. presents the software fault-count data used in the analysis, where four grouped data sets in the GitHub are employed. The grouped data were given by counting the time-domain data which were reported by the software users in each open-source software from the failure issues monthly. Suppose that the first 20%, 50%, and 80% of the entire data points are used for training, and that the remaining data for validation. We view these data points as the early, middle, and late software testing phases, respectively.

Next, we introduce the methods used in our experiments as predictors. The "Best Single Model" (BSM) denotes the NHPP-based SRM with the ex post facto best predictive performance among $\phi = 11$ candidates in Table 1., which cannot be known in advance at each prediction (training) point. The "Single Model Selected by AIC" (SM-AIC) is the NHPP-based SRM

with the minimum AIC at each training point, which is feasible for prediction. The predictive performance is measured by the predictive mean absolute error (PMAE):

$$\text{PMAE} = \frac{1}{k}\sum_{i=1}^{k} |(n+i) - \Lambda(t_{n+i}; \tilde{\boldsymbol{\theta}})|, \tag{24}$$

where $\{t_1, t_2, \ldots, t_n, \ldots, t_{n+k}\}$ represent the software fault detection-time data and k is the prediction length. It is obvious to see that SM-AIC cannot always outperform BSM in terms of PMAE.

Let Model A0, Model A1 and Model A2 be the original AdaBoosting-based LWCM ("AMCM" in [9]), our refined AdaBoosting-based LWCMs with the cumulative effects in Eq.(14) and without the cumulative effects in Eq. (23), respectively. For Model A0 $\sim$ Model A2, we apply three kinds of loss functions; (L), (S), (E) in Eqs. (16), (17) and (18), respectively. Note that the maximum likelihood estimation is applied to Model A0, while Li et al. [9] used the least squares estimation.

Table 3. Predictive performances of LWCMs with grouped data.

Phase	Early Testing Phase (20%)		Middle Testing Phase (50%)		Late Testing Phase (80%)	
Data	BSM	SM-AIC	BSM	SM-AIC	BSM	SM-AIC
DS1	289.84 (TruncLogist)	343.76 (TruncEVMin)	379.81 (LogEVMin)	980.56 (TruncLogist)	313.38 (Gamma)	614.39 (TruncLogist)
DS2	294.23 (Gamma)	300.22 (Exp)	42.56 (LogEVMin)	259.54 (TruncLogist)	59.03 (LogEVMin)	62.10 (TruncEVMax)
DS3	12.02 (LogEVMax)	12.02 (LogEVMax)	3.06 (LogNormal)	6.92 (TruncLogist)	0.49 (TruncEVMin)	2.80 (LogNormal)
DS4	234.26 (Pareto)	709.65 (TruncEVMin)	15.90 (TruncLogist)	107.90 (LogLogist)	0.66 (TruncNormal)	4.84 (LogEVMin)

5.2 Predictive Performances

Table 3. present the predictive performances of the infeasible BSM and the feasible SM-AIC with the grouped data. It can be observed that the NHPP-based SRM with the best goodness-of-fit is not always equivalent to the best prediction model, where the bolded results denote that the corresponding model could give a better predictive performance than the SM-AIC.

Table 4. compares the PMAEs among all the ensemble methods; A0, A1, A2, with the grouped data. The results with yellow color indicate that the corresponding model could provide a better predictive performance than the BSM. Similar to Table 3., the bolded results imply that the corresponding ensemble methods could give better predictive performances than the SM-AIC. In particular, the red results in the table showed the best predictive performance among all the ensemble learning-based methods.

Table 4. Comparison of predictive performances by the ensemble methods with grouped data.

Early Testing Phase (20%)									
Data	A0 (L)	A0 (S)	A0 (E)	A1 (L)	A1 (S)	A1 (E)	A2 (L)	A2 (S)	A2 (E)
DS1	1054.52	982.00	1795.46	**319.54**	**335.60**	**328.67**	**326.25**	**333.55**	**325.08**
DS2	322.79	**278.47**	**292.27**	**299.23**	304.40	**300.21**	**299.82**	**300.11**	**299.58**
DS3	101.85	97.48	201.52	12.36	12.49	12.66	12.72	12.60	12.60
DS4	**174.39**	**635.34**	**596.96**	**700.98**	**701.32**	**698.82**	**701.20**	**703.35**	**701.64**
Middle Testing Phase (50%)									
Data	A0 (L)	A0 (S)	A0 (E)	A1 (L)	A1 (S)	A1 (E)	A2 (L)	A2 (S)	A2 (E)
DS1	**934.51**	**942.62**	**953.49**	**979.84**	**979.80**	**979.40**	**980.30**	**979.93**	**979.02**
DS2	**180.06**	**245.45**	344.53	259.55	**259.33**	**259.42**	**259.35**	**259.32**	259.54
DS3	**4.71**	72.73	142.90	6.94	7.00	6.98	7.01	6.98	7.00
DS4	244.19	**103.93**	297.31	108.24	108.20	108.09	108.07	108.03	108.09
Late Testing Phase (80%)									
Data	A0 (L)	A0 (S)	A0 (E)	A1 (L)	A1 (S)	A1 (E)	A2 (L)	A2 (S)	A2 (E)
DS1	682.17	733.49	838.54	**614.18**	**614.18**	**614.21**	**614.20**	**614.13**	**614.10**
DS2	149.21	73.01	165.54	**61.90**	**61.88**	**61.87**	**61.90**	**61.76**	**61.80**
DS3	12.67	51.59	66.33	2.83	2.82	2.86	2.83	2.85	2.84
DS4	106.35	174.13	331.19	**4.84**	**4.84**	**4.85**	**4.84**	**4.84**	**4.84**

By comparing Table 4. with Table 3., it is found that Model A0 could improve the predictive performances in a few cases. Focusing on the comparison with the SM-AIC, we can observe that Model A2 could also improve the predictive performances than A1(L) and A1(S) in most cases. Overall, it can be seen that the AdaBoosting-based approaches could give better predictive performances in many cases in the early testing phase. This is due to the fact that the single use of NHPP-based SRM strongly depends on the number of data; a small amount of data in the early testing phase and a relatively large amount of data in the late testing phase.

It can be observed that our AdaBoosting-based approaches gave superior predictive performances than LWCM in the middle testing phase. However, those methods showed worse results in the early and late testing phases. Finally, we can conclude that our AdaBoosting-based methods can provide satisfactory predictive performances in most cases than the plug-in predictor, and especially suggest to apply Model A2 under arbitrary loss functions.

6 Conclusions

In this paper, we have developed several AdaBoosting-based LWCMs with NHPP-based SRMs by introducing the thinning-like resampling algorithm, and compared them with the existing

methods [9], in addition to the plug-in predictors. Through numerical experiments with actual software fault-count data, it has been shown that our refined AdaBoosting methods could give the better predictive performances than the best plug-in predictors in almost half cases of time-domain data sets in the early testing phase. Since the resulting ensemble methods are lightweight methods in computation, it is relatively easy to implement the algorithms on software reliability assessment tools which can automate the quantitative software reliability prediction.

In the future, we will apply the clustering technique on LWCM to select the representive SRM in advance and reduce the number of candidate SRMs. By doing the above, it will be possible to implement lightweighted ensemble software reliability prediction.

7 Acknowledgements

This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University.

8 References

[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. Niverthi (1998), A Bayesian approach using nonhomogeneous Poisson processes for software reliability models, *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, *Proceedings of the 2nd International Symposium on Information Theory*, B. N. Petrov, and F. Caski (eds.), pp. 267–281, Akadimiai Kiado, Budapest.

[4] H. Drucker (1997), Improving regressors using boosting techniques, *Proceedings of the 14th International Conference on Machine Learning (ICML-1997)*, pp. 107–115, ACM.

[5] 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.

[6] A. L. Goel (1985), Software reliability models: assumptions, limitations and applicability, *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1411–1423.

[7] S. S. Gokhale, and K. S. Trivedi (1998), Log-logistic software reliability growth model, *Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium (HASE-1998)*, pp. 34–41, IEEE CPS.

[8] P. A. W Lewis, and G. S. Shedler (1979), Simulation of nonhomogeneous Poisson processes

by thinning, *Naval Research Logistics Quarterly*, vol. 26, no. 3, pp. 403–413.

[9] H. Li, M. Zeng, M. Lu, X. Hu, and Z. Li (2012), AdaBoosting-based dynamic weighted combination of software reliability growth models, *Quality and Reliability Engineering International*, vol. 28, no. 1, pp. 67–84.

[10] M. R. Lyu, and A. P. Nikora (1992), Applying reliability models more effectively, *IEEE Software*, vol. 9, no. 4, pp. 43–42.

[11] M. Ohba (1984), Inflection S-shaped software reliability growth model, *Stochastic Models in Reliability Theory*, S. Osaki, and Y. Hatoyama (eds.), pp. 144–162, Springer, Berlin/Heidelberg.

[12] 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.

[13] 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.

[14] H. Okamura, and T. Dohi (2013), SRATS: Software reliability assessment tool on spreadsheet, *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering (ISSRE-2013)*, pp. 100–107, IEEE CPS.

[15] J. Wu, J. Zheng, T. Dohi, and H. Okamura (2024), An alternative boosting-based software reliability prediction method, *Proceedings of The IEEE 29th Pacific Rim International Symposium on Dependable Computing (PRDC-2024)*, pp. 231–234, IEEE CPS.

[16] 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.

[17] H. Yano, T. Dohi and H. Okamura (2024), Performance comparison of software reliability estimation algorithms, *IEEE Computer*, vol. 57, no. 4, pp. 26–36.

[18] 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.