Exploring AdaBoosting Algorithm for Combining Software Reliability Models

Haifeng Li, Min Zeng, and Minyan Lu
Department of Engineering System and Engineering, BeiHang University, China
lihaifeng@dse.buaa.edu.cn

Abstract

Software reliability growth models (SRGMs) are very important for software reliability estimation and prediction and have been successfully applied in software reliability engineering. However, there is no general model which can perform well for different cases. Therefore, several combinational methods of SRGMs have been proposed to improve the reliability estimation and prediction accuracy.

*AdaBoosting (short for Adaptive Boosting) is a commonly used machine learning (ML) algorithm for combining several weak predictors into a single strong predictor to significantly improve the forecasting accuracy. In this paper, an **A**daBoosting-based approach for obtaining a dynamic weighted linear **C**ombinational **M**odel (ACM) is proposed. The key idea of this approach is that we select several SRGMs as the weak predictors and use AdaBoosting algorithm to determine the weights of these models for obtaining the final linear combinational model.*

Finally, two case studies are presented respectively for comparing the proposed ACM with five SRGMs and a neural-network-based combination approach on several real failure data-sets. Preliminary experimental results show that this proposed ACM is fairly effective and applicable since it has: 1) a significantly better goodness-of-fit and prediction capability than the five single SRGMs; 2) a comparable at least or even a little better fitting and prediction performance than the dynamic weighted combinational model based on neural-network in most cases.

1. Introduction

Software reliability is defined as the probability of failure-free software operation for a special period of time in a special usage environment [1]. Over the past 30 years, many time-domain models, also called software reliability growth models have been proposed for the reliability estimation and prediction during software development and operation process [1]. However no one has shown to perform well for all applications. To improve the estimation and prediction accuracy of SRGMs, some important factors which affect the final software reliability, for example, test coverage [3, 4], test effort [5], and some machine learning techniques, such as, GP (Genetic Programming) [9], AdaBoosting [10], SVMs (Support Vector Machines) [11], ANN (Artifical Neural Networks), are considered in the modeling process.

An ideal SRGM should provide consistently accurate reliability estimation and prediction across different projects. However, many researches have shown that there is no single such model which can obtain accurate results for different cases [2]. The reason is that the performance of SRGMs highly depends on the assumptions on the failure behavior and the application data-sets. In other words, many models may be shown to perform well with one failure data-set, but bad with the other data-set. Thus, some researchers proposed to obtain more accurate estimation and prediction than one single model [2, 6~8] by combing some individual SRGMs. For example, Lyu [6] proposed four combinational models, namely, equal, mediam, unequally and dynamical weight. Su [7] used the neural network approach to build a dynamic weighted combinational model (DWCM). Hsu [8] proposed three combinational models, namely, equal, linear, and nonlinear weight, and used genetic algorithm to determine the weight assignment.

AdaBoosting is a commonly used ML algorithm which can combine several weak predictors into a single strong predictor for highly improving the estimation and prediction accuracy. Eduardo [10] has explored GP and AdaBoosting techniques to obtain a non-parameter software reliability model.

In this paper, an AdaBoosting-based approach for obtaining a dynamic weighted linear Combinational Model (ACM) is presented. First, several different SRGMs are selected as the weak predictors. Then, we study how to use AdaBoosting algorithm to determine the weights of these models on the training data-sets for obtaining the final linear combinational model. Finally, two case studies are presented respectively for comparing the proposed ACM with five well-known SRGMs (such as GO) and a neural-network-based combination

approach (DWCM) [7] on some real failure data-sets. Preliminary experimental results are analyzed to investigate the estimation and prediction capability of this proposed AdaBoosting-based combination approach.

2. An Overview of AdaBoosting

AdaBoosting is a commonly used machine learning algorithm for constructing a strong classifier $f(x)$ as linear combination of weak classifiers which is shown as follows:

$$f(x)=\sum_{i=1}^{P} W_i h_i(x) \quad (1)$$

From Eq.1, AdaBoosting calls a weak classifier $h_i(x)$ repeatedly in a series of rounds i=1...P. For each call a distribution of weights W_i is updated that indicates the importance of examples in the data-set for the classification [12].

3. AdaBoosting-based Combinational Model

Many SRGMs may result in estimation or prediction bias since their underlying assumptions are not consistent with the characteristics of the application data. To reduce this bias, combining several different SRGMs together in linear or nonlinear manner is a common and applicable method. Many approaches are proposed to determine the weight assignment for the combinational models, such as equal weight [6], neural-network [7], genetic [8] and etc. In this section, an abstract description of how to use AdaBoosting to obtain the dynamic weighted linear combinational model (ACM) of several SRGMs is shown as follows.

Input:

1. Let $M(t)=u(x_1 \ldots x_k \ldots x_S, t)$ denotes different SRGMs (such as GO model). $M(t)$ is the cumulated faults detected at time t, S is the number of the parameters of $M(t)$, x_k is the k-th parameter of $M(t)$, k=1...S;

2. The failure data-set D_0 is denoted by $(t_1, m_1)...(t_j, m_j)...(t_n, m_n)$. n is the data number of D_0, m_j is the cumulated faults detected at t_j, j=1...n.

Initialize:

Step1: Selecting M different SRGMs (denoted by $M_m(t)$, m=1...M) as the candidate models for the ACM.

Step2: The original weight set of D_0 can be denoted by $K_0=\{k_{01}... k_{0n}\}$, where k_{0j} is initialized by $1/n$.

Circulation:

Step3: New training data set D_i (i=1...P, P is the training rounds) are generated by repeatedly random sampling from D_0 according to its corresponding weight set $K_i=\{k_{i1} \ldots k_{ij} \ldots k_{in}\}$. If there are some same data in D_i, only one of them will be reserved in our approach. Hence the data number of D_i may not be n.

Step4: D_i is used to estimate the parameters of each $M_m(t)$ in the i-th training round. Then the fitness function (Notation 1) of $M_m(t)$ can be determined by D_0. The candidate model whose value of fitness function is the smallest is chosen as the selected model (denoted by $M_{is}(t)$) in this round, i=1...P.

Step5: If $\hat{M}_{is}(t)$ denotes the estimation form of $M_{is}(t)$, the loss function L_{is} (Notation 2) of $M_{is}(t)$ can be calculated by the fitting results of $\hat{M}_{is}(t)$ with D_0. Then K_i can be updated as K_{i+1} by L_{is} (Notation 3). The basic weight β_{is} of $M_{is}(t)$ also can be determined by L_{is} (Notation 4).

Step6: Performing Step3~Step5 repeatedly until i=P, and then turning into Step7.

Output:

Step7: Finally, a combinational linear model (i.e. ACM) is obtained as follows:

$$M_{ACM}(t)=\sum_{i=1}^{P} W_{is} M_{is}(t) \quad (2)$$

The combination weight W_{is} of the selected model $M_{is}(t)$ is $f(\beta_{is})$ (Notation 5), that is, W_{is} is a function of the basic weight β_{is}.

Notation 1:

Fitness function (FF) can be defined according to the estimation method of the parameters of these candidate SRGMs. In this paper, if Maximum Likelihood Estimation is used, FF is equation (3) [8]. If Least Square Estimation is used, FF is equation (4) [5].

$$FF=1/-\log(ML) \quad (3)$$

where ML is the maximum likelihood function.

$$FF=MSE=\sum_{j=1}^{n}\left[m(t_j)-m_j\right]^2/n \quad (4)$$

where m_j is the observed number of faults by time t_j.

Notation 2:

$$L_{is}=\sum_{j=1}^{n} k_{ij} * L_{is}^{j} \quad (5)$$

where k_{ij} is defined in Notation 3.

There are three equations for calculating L_{is}^{j}.

1) Linear: $L_{is}^{j}=AE_{is}^{j}/D_{ent}$;

2) Square: $L_{is}^{j}=(AE_{is}^{j}/D_{ent})*(AE_{is}^{j}/D_{ent})$;

3) Exponential: $L_{is}^{j}=1-\exp(-AE_{is}^{j}/D_{ent})$.

where $AE_{is}^{j} = m_j - \hat{M}_{is}(t_j)$, $D_{ent} = \max\{AE_{is}^{j}\}$.

Notation 3:

$$K_{i+1} = \{k_{i+1,j}\},\ k_{i+1,j} = k_{ij} * L_{is}^{j} / (\sum_{j=1}^{n} k_{ij} * L_{is}^{j}). \quad (6)$$

Notation 4:

$$\beta_{is} = L_{is} / (1 - L_{is}) \quad (7)$$

Notation 5:

$$W_{is} = f(\beta_{is}) = \log\frac{1}{\beta_{is}} / \sum_{i=1}^{P} \log\frac{1}{\beta_{is}} \quad (8)$$

4. Case Studies

In this section, the fitting and prediction performance of the proposed ACM is analyzed by comparing with five single NHPP SRGMs on two failure data-sets first and secondly with a combination approach (DWCM) on another two failure data-sets.

4.1 Case study 1: Comparison with SRGMs

1. Description of this case study

An experiment in this case study contains the following contexts.

1) In this study, two real failure data-sets are selected, Ohba and Wood [5], which are popular and frequently used as benchmark for comparison of SRGMs.

2) Parameters Estimation. Least Square Estimation (LSE) is preferred for it produces unbiased results [7].

3) Fitness Function and Criterion for Model's Comparison. MSE (Eq. 4) is selected as the fitness function and comparison criterion.

4) The following five models (shown in Table 1) are selected as the candidate and comparison models, which have been widely cited by many researchers in the field of software reliability modeling, namely, GO (M_1), MO (M_2), Delayed S-Shaped (M_3), Inflected S-Shaped (M_4) and Generalized GO (M_5).

5) Settings in AdaBoosting. a) Linear equation is used for L_{is}^{j}; b) The training rounds is 20 (i.e. P=20).

2. Comparison of goodness-of-fit performance

1) The estimated parameters of five candidate models are shown in Table 1. Then the fitting results (values of MSE) of these five models are shown in table 2 respectively.

2) With each failure data-set, 100 experiments are performed to get 100 ACMs for comparison. Then the fitting results of these ACMs are obtained. The ACM whose value of MSE (shown in Table 2) is the smallest will be selected for comparison and denoted by ACM^{best}. The training process of the ACM^{best} is shown in Table 3.

The actual data and the estimation data of these five models and ACM^{best} are also plotted in Fig.1 & 2 against the logarithm of the cumulated faults (i.e. m_j).

Table 1 The Estimated Parameters of Five Models

	$M(t)$	Ohba			Wood		
		a	r	c	a	r	c
M_1 [1]	$a(1-\exp(-rt))$	760.53	0.0323	-----	130.2	0.0832	-----
M_2 [1]	$\frac{1}{a}\ln(1+art)$	0.00162	24.852	----	0.01383	12.426	----
M_3 [1]	$a(1-(1+rt)\exp(-rt))$	374.05	0.197	-----	103.98	0.265	-----
M_4 [1]	$\frac{a(1-\exp(-rt))}{1+c\exp(-rt)}$	374.39	0.179	3.01	110.83	0.172	1.2
M_5 [1]	$a(1-\exp(-rt^c))$	427.83	0.0361	1.28	118.56	0.0765	1.11

Table 2 Comparison of Goodness-of-fit

	Ohba	Wood
	MSE (fit)	MSE (fit)
M_1	139.82	11.62
M_2	148.34	15.842
M_3	168.67	25.26
M_4	127.31	8.98
M_5	102.12	10.87
ACM^{best}	**83.55**	**2.05**

Notes: The bold indicates the best fitting result

Table 3 The Training Process of ACM^{best}

	Ohba		Wood	
	$M_{dis}(t)$	W_{is}	$M_{dis}(t)$	W_{is}
1.	M_1	0.264	M_4	0.0522
2.	---	---	M_4	0.0401
3.	M_4	0.283	M_4	0.0429
4.	M_4	0.232	M_5	0.0426
5.	---	---	M_4	0.0303
6.	---	---	M_4	0.0446
7.	M_1	0.221	---	---
8.	---	---	M_4	0.0667
9.	---	---	M_4	0.0614
10.	---	---	M_4	0.0579
11.	---	---	M_4	0.0584
12.	---	---	M_4	0.0580
13.	---	---	M_4	0.0556
14.	---	---	M_4	0.0633
15.	---	---	M_4	0.0544
16.	---	---	M_3	0.0542
17.	---	---	M_4	0.0578
18.	---	---	M_4	0.0546
19.	---	---	M_4	0.0530
20.	---	---	M_4	0.0520

Notes: '---' indicates this training round is invalid for the estimation values of the parameters of models are unreasonable or not existing.

3) From Table 2 and Fig.1 & 2, it should be noted that the values of MSE for the goodness-of-fit of the ACM^{best} on two data-sets are both significantly smaller than these five models. This means the ACM^{best} provides the best descriptive power and implies that the proposed AdaBoosting-based approach can effectively improve the fitting accuracy of individual model since the AdaBoosting algorithm can optimize the weights of the candidate models by training them on data-sets repeatedly. To further validate this conclusion and illustrate the improvement process of the AdaBoosting-based approach, the following two sections 4) and 5) were proposed and analyzed.

4) Let MSE_m denotes the MSE of each model (m=1…5), MSE_t denotes the MSE of each ACM

(t=1…100). Then we have

$$E_m = \{e_{mt} \mid e_{mt} = MSE_m - MSE_t, m = 1,...,5, t = 1,...,100\}$$

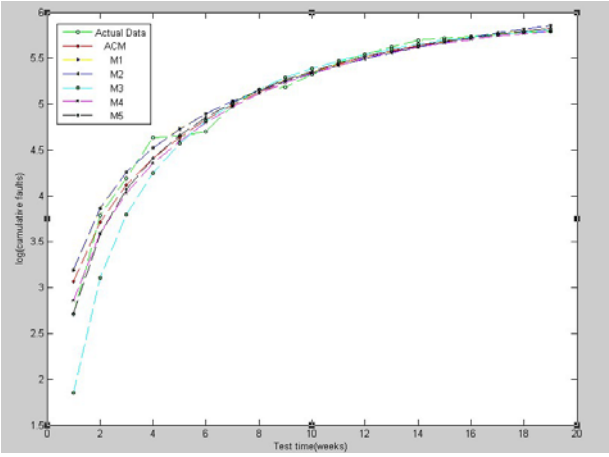

Figure 1 The comparison of fitness with Ohba

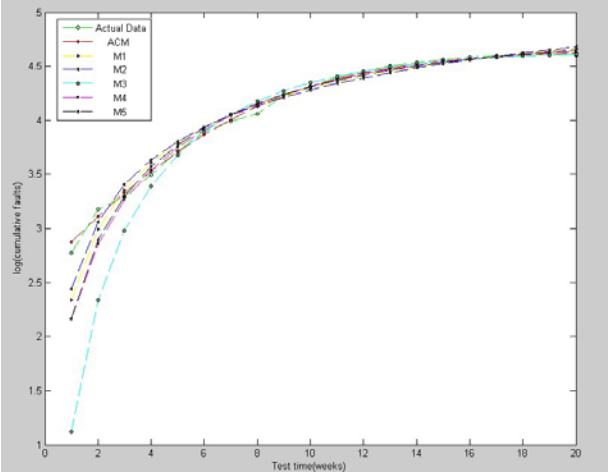

Figure 2 The comparison of fitness with Wood

E_m is used to calculate the following three statistics.① 'Times$_m$' denotes the number of the e_{mt} (t=1…100) whose value is larger than 0 (namely, ACM$_t$ gives a better fitting power than M_m). ② 'MaxError$_m$' denotes the largest value of e_{mt} (t=1…100), where e_{mt}>0. ③ 'AvgError$_m$' denotes the average value of e_{mt} (t=1…100) which can be calculated by the following equation

$$\text{AvgError}_m = \sqrt{\frac{\sum_{t=1}^{Times} e_{mt} * e_{mt}}{Times}}, \text{ where } e_{mt} > 0 \qquad (9)$$

Table 4 shows the results of these above three statistics. From Tab. 4, we note that: ①Almost each 'Times$_m$' is beyond 90 and close to 100 on the two data-sets (only expect the 'Times$_5$' on the Ohba data-set), which means that not only the ACM^{best} but also nearly each ACM$_t$ (t=1…100) is superior to every M_m in the fitting performance in each experiment on each data-set. ②The ACM^{best} has a notable improved result of the fitting performance compared with each M_m on each data-set. For example, compared with M_3, the ACM^{best} improves the fitting performance up to 50.4% on Ohba, and even up to 91.9% on Wood. ③The proposed AdaBoosting-based approach has an effective capability for improving the fitting performance. For example, compared with M_3, the average improved results of ACM are 37.1% on Ohba, and even 81.9% on Wood. As a whole, the proposed AdaBoosting-based approach for combining models is fairly effective for improving the fitting performance of single models.

Table 4 The Results of Three Statistics

	Ohba			Wood		
	Times$_m$	MaxError$_m$	AvgError$_m$	Times$_m$	MaxError$_m$	AvgError$_m$
M_1	99	56.27 (40.2%)	34.93 (25.0%)	98	9.57 (82.3%)	7.36 (63.3%)
M_2	99	64.79 (43.7%)	43.06 (29.0%)	98	13.79 (87.0%)	11.52 (72.7%)
M_3	100	**85.12 (50.4%)**	**62.57 (37.1%)**	100	**23.21 (91.9%)**	**20.68 (81.9%)**
M_4	91	43.76 (34.4%)	24.44 (19.2%)	96	6.93 (77.2%)	4.87 (54.2%)
M_5	64	18.57 (18.2%)	8.53 (8.3%)	97	8.82 (81.1%)	6.66 (61.3%)

Notes: a) The number in bracket is MaxError$_m$ (or AvgError$_m$) divided by MSE$_m$ which means the maximum (or average) improved result of ACM$_t$ compared with M_m. b) The bold means the best improved result.

5) C_j is denoted as the sampling times of data (t_j, m_j) of D_0 in the 100 experiments. C_j is used to describe the sampling distribution (SD) of (t_j, m_j). The absolute error between m_j and its estimated value of M_m is given by Δ_j^m (i.e. Δ_j^m =|$M_m(t_j)$-m_j|). The average absolute error between m_j and its estimated value of ACM$_t$ is given by Δ_j^{ACM}. Δ_j^m and Δ_j^{ACM} are used to describe the fitting error distribution (ED) on M_m and ACM$_t$ of (t_j, m_j). Due to the limited space, only the figures of SD and FE of Ohba and Wood data-sets on M_1 and ACM$_t$ are shown in Fig. 3~4 by normalizing C_j, Δ_j^m and Δ_j^{ACM}.

From Fig.3 and Fig. 4, we note that there is an approximatively proportional relationship between SD and ED of the failure data. In other words, if the Δ_j^m and Δ_j^{ACM} become larger, and then the sampling times of data (t_j, m_j) become more. We think this may be the improvement process of the AdaBoosting-based approach which means that the failure data with large fitting errors on M_m and ACM will be selected and trained by AdaBoosting many times so as to improve the fitting performance of the model on this failure data.

3. Comparison of prediction performance

1) The first 17 failure data of Ohba and the first 18 failure data of Wood are used to estimate the parameters in M_m (m=1…5) and the ACM^{best}. Then the remaining 2 failure data of these two data-sets are used to compare the prediction power of these models. The MSE values for the prediction power are listed in Table 5.

2) From Table 5, we note that the values of MSE for the prediction of the ACM^{best} on two data-sets are both

significantly smaller than four single models (except M_3). This means the ACM^{best} provides a pretty prediction

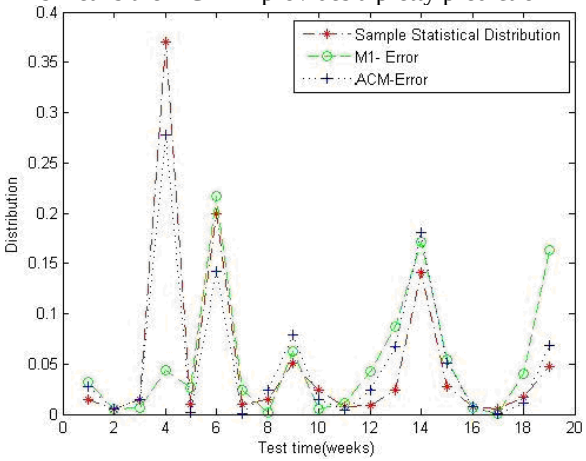

Fig.3 The SD and ED of Ohba Data-set

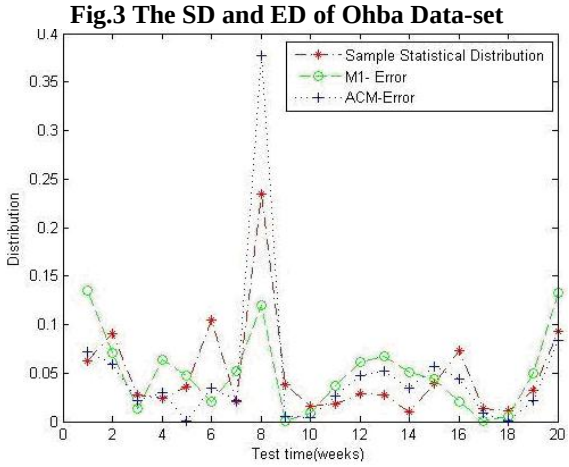

Fig.4 The SD and ED of Wood Data-set

Table 5 Comparison of Prediction

	Ohba	Wood
	MSE (prediction)	MSE (prediction)
M_1	1133.80	63.93
M_2	1202.90	106.77
M_3	**27.17**	**0.45**
M_4	377.35	34.85
M_5	376.65	62.21
ACM^{best}	**90.42**	**2.60**

Notes: The bold indicate the two best fitting results

power and implies that the proposed approach can effectively improve the prediction accuracy of individual model. It should be noted that the values of MSE for the prediction of M_3 on two data-sets are both smaller than the ACM^{best}. However, the fitting performance of M_3 (shown in Tab.2) is significantly worse than the ACM^{best}. Thus we think it is just a coincidence for the number of failure data used to prediction is a little less.

To sum up, according to the section 2 and 3 in 4.1, ACM can improve the fitting and prediction accuracy compared with five single SRGMs effectively.

4.2 Case study 2: Comparison with DWCM

1. Description of this case study

An experiment in this case study contains the following contexts.

1) In this study, two classical failure data-sets used in [7] are selected, namely DS1 and DS2 [1].

2) Parameters Estimation. Least Square Estimation (LSE) is preferred.

3) The dynamic weighted combinational model based on neural-network (DWCM) [7] is selected as the comparison model.

4) Fitness Function and Comparison Criterion. MSE (Eq. 4) is selected as the fitness function. The average relative error (AE, used in [7]) is selected as the comparison criterion for the convenience of comparison.

$$AE = \frac{1}{n}\sum_{j=1}^{n}\left|\frac{m(t_j)-m_j}{m(t_j)}\right|\times 100 \quad (10)$$

5) Settings in AdaBoosting. a) Linear equation is used for L_{is}^{j}; b) The training rounds is 20 (i.e. P=20).

6) GO (M_1), Delayed S-Shaped (M_3) and Inflected S-Shaped (M_4) are selected as the candidate models, which are the selected models in [7].

7) The first x% of the total failure data is used for calculating the parameters and comparing the fitting performance of the DWCM and ACM^{best}. Then the rest of the failure data is used for comparing the prediction performance of the DWCM and ACM^{best}.

2. Comparison on DS1

In this study, we also perform 100 experiments with each data-set and select the ACM^{best} for comparison.

1) Table 6 gives the average relative error for different x% values that are from 50% to 100%, which is used to compare the fitting performance. Table 7 gives the average relative error for different x% values that are from 50% to 90%, which is used to compare the prediction performance.

Table 6 Comparison of Goodness-of-fit on DS1

x%	DWCM	ACM^{best}
	AE (fit)	AE (fit)
50%	2.28	7.9372
60%	3.24	8.5976
70%	1.9	7.4753
80%	1.79	6.9791
90%	1.5	3.5646
100%	1.64	3.3995

Table 7 Comparison of Prediction on DS1

x%	DWCM	ACM^{best}
	AE (Prediction)	AE (Prediction)
50%	13.18	2.6372
60%	7.3	2.4433
70%	3.44	3.8211
80%	2.92	2.103
90%	1.76	1.3275

2) From Tab.6, the values of AE from 50% to 100% of the proposed ACM^{best} are all larger than the DWCM. It means that the ACM^{best} provides a worse fitting performance than DWCM on DS1. However, from Tab.7, we note that the values of AE from 50% to 90% of the ACM^{best} are nearly all smaller than the DWCM. Thus, the ACM^{best} provides a better prediction performance than DWCM on DS1 at least. Tab.6 & 7 also show that the fitting and prediction results are obviously improved with the growth of *x*% values. More information about the failures can help the AdaBoosting optimize the weights of the candidate models more effectively.

3. Comparison on DS2

1) Table 8 gives the average relative error for different *x*% values that are from 50% to 100%, which is used to compare the fitting performance. Table 9 gives the average relative error for different *x*% values that are from 50% to 90%, which is used to compare the prediction performance.

Table 8 Comparison of Goodness-of-fit on DS2

x%	DWCM	ACM^{best}
	AE (fit)	AE (fit)
50%	6.43	5.4372
60%	6.56	5.0749
70%	5.67	4.6119
80%	5.27	4.3877
90%	4.77	3.801
100%	4.3	3.5209

Table 9 Comparison of Prediction on DS2

x%	DWCM	ACM^{best}
	AE (Prediction)	AE (Prediction)
50%	6.7	5.2925
60%	5.51	5.3475
70%	3.69	1.9791
80%	1.01	1.0965
90%	0.22	0.7189

2) From Tab.8, we note that the values of AE from 50% to 100% of the proposed ACM^{best} are all smaller than the DWCM. It means that the ACM^{best} provides a better fitting performance than DWCM on DS2. Furthermore, from Tab.9, the values of AE from 50% to 70% of the proposed ACM^{best} are all smaller than the DWCM, 80% is nearly the same, and only 90% is larger. As a whole, the proposed ACM^{best} provides a better prediction performance than DWCM on DS2.

To sum up, according to the analysis of the section 2 and 3 in 4.2, we suggest that, compared with the DWCM, the proposed ACM has a comparable at least or even a little better fitting and prediction performance on DS1 and DS2 in most cases.

5. Conclusion

In this paper, an AdaBoosting-based approach is presented for obtaining a dynamic weighted linear combinational model. Two case studies are then presented and analyzed for comparing this approach with several SRGMs and a combination approach (DWCM). The results (Tab.2&4, Tab.5~9, Fig.1&2) of case studies show that the proposed approach provides much better accuracy in reliability estimation and prediction compared with five single SRGMs (M_1~M_5) and a comparable or a little better accuracy compared with DWCM in most cases.

Hsu proposed a Genetic-based combination approach [8] (GCM) and applied GCM into a failure data-set of web-based software. However this data-set was not shown in [8], we can't compare the ACM with GCM at present. Thus, the following issues will be further discussed in our future work: 1) Investigating the fitting and prediction performance of the ACM by using maximum likelihood estimation to estimate the parameters of models; 2) Examining the statistical significance of the estimation and prediction results of the ACM; 3) Comparing with the other combination approaches (such as GCM).

References

[1] M. R. Lyu. Handbook of Software Reliability Engineering. McGraw Hill, 1996

[2] S. M. Li, Q. Yin, P. Guo and M. R. Lyu. A hierarchical mixture model for software reliability prediction. Applied Mathematics and Computation, 2007, 185: 1120~1130

[3] X. Cai, M. R. Lyu. Software Reliability Modeling with Test Coverage Experimentation and Measurement with a Fault-Tolerant Software Project. ISSRE, 2007: 17~26

[4] Yashwant K. Malaiya, Michael Naixin Li and etc. Software reliability growth with test coverage. IEEE Transactions on Reliability, 2002, 51(4): 420～426.

[5] C. Y. Huang, S. Y. Kuo and M. R. Lyu. An assessment of testing-effort dependent software reliability growth models. IEEE Transactions on Reliability, 2007, 56(2): 198~211

[6] Lyu, M. R, Nikora, A. Applying Reliability Models More Effective. IEEE Software, 1992, 9(4): 43-52

[7] Y. S. Su, C. Y. Huang. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. The Journal of Systems and Software, 2007, 80: 606-615

[8] C. J. Hsu, C. Y. Huang. Reliability analysis using weighted combinational models for web-based software. WWW 2009, 1131~1132

[9] Eduardo Oliveira Costa, Silvia R. Vergilio, Aurora Pozo, Gustavo Souza. Modeling software reliability growth with Genetic Programming. ISSRE, 2005: 1~10

[10] Eduardo Oliveira Costa, Gustavo Souza, Aurora Pozo, Silvia R. Vergilio. Exploring Genetic Programming and Boosting Techniques to Model Software Reliability. IEEE Transactions on Reliability, 2007, 56(3): 422-434

[11] Ping-Feng Pai and *et al*. Software reliability forecasting by support vector machines with simulated annealing algorithms. The journal of Systems and Software, 2006, 79: 747~755

[12] http://en.wikipedia.org/wiki/AdaBoost