초기하분포 소프트웨어 신뢰성 성장 모델에서의 모수 추정과 예측 방법

박 중 양† · 유 창 열†† · 이 부 권†††

요 약

최근에 개발되어 성공적으로 적용되고 있는 초기하분포 소프트웨어 신뢰성 성장 모델의 모수는 최우추정법으로 추정하기가 쉽지 않으므로 주로 최소자승법으로 추정하고 있다. 본 논문에서는 먼저 기존의 최소자승법에서 사용된 최소화 기준을 비교한 다음, 새로 발견되는 결함수의 분산이 일정하지 않음을 고려한 가중최소자승법을 제안한다. 그리고 두 개의 실제 자료를 분석하여 가중최소자승법이 적합함을 보인다. 마지막으로 임의의 테스팅 시점에서 추가 시험에 의해 발견된 새로운 결함수를 예측하는 방법을 제안한다. 이 예측 방법은 테스팅을 중단하는 시점을 결정할 때 이용될 수 있을 것이다.

Parameter Estimation and Prediction Methods for Hyper-Geometric Distribution Software Reliability Growth Model

Joong-Yang Park† · Chang-Yeul Yoo†† · Bu-Kwon Lee†††

ABSTRACT

The hyper-geometric distribution software reliability growth model was recently developed and successfully applied. Due to mathematical difficulty of the maximum likelihood method, the least squares method has been suggested for parameter estimation by the previous studies. We first summarize and compare the minimization criteria adopted by the previous studies. It is then shown that the weighted least squares method is more appropriate because of the nonhomogeneous variability of the number of newly detected faults. The adequacy of the weighted least squares method is illustrated by two numerical examples. Finally, we propose a new method for predicting the number of faults newly discovered by next test instances. The new prediction method can be used for determining the time to stop testing.

1. Introduction

In recent years software systems have been widely applied to many complex and critical systems. Since the failure of a software system may result in serious damage, software systems are required to be very reliable. Therefore software reliability has become one of major issues in the software system development. In order to quantitatively assess the reliability of a software system during the testing and operational phases, many software reliability growth models (SRGMs) have

† 정 회 원 : 경상대학교 통계학과 교수, 정보통신연구센타
†† 정 회 원 : 남해전문대학 사무자동화과 조교수
††† 정 회 원 : 경상대학교 컴퓨터과학과 교수
논문접수 : 1998년 5월 18일, 심사완료 : 1998년 7월 10일

been proposed in the literature. See, for example, Goel [1], Ramamoorthy and Bastani [11] and Shanthikumar [12] The SRGMs are usually used to estimate the number of remaining faults, software reliability and other software quality assessment measures. Some of currently available SRGMs enable us to predict probabilistically the time to next occurrence of failure in the operational phase. Another class of SRGMs let us estimate the number of software faults still residual after the debugging process. The hyper-geometric distribution software reliability growth model (HGDM) advocated by Tohma et al. [13] belongs to the latter class of SRGMs. A series of studies on the HGDM has been made recently by Hou, Kuo and Chang [2], [3], Jacoby and Tohma [6], Minohara and Tohma [9] and Tohma et al. [14] Hou, Kuo and Chang [4], [5]developed optimal software release policies based on the HGDM.

This paper first considers the problem of estimating the parameters of the HGDM. Then the problem of predicting the number of faults newly discovered by additional test operations is investigated. Section 2 briefly reviews the basic concept and formulation of the HGDM. The parameter estimation problem is discussed and then the weighted least sequares method is proposed in Section 3. Experiments have been performed by using two real data sets and the results are presented in Section 4. The results show that the proposed weighted least squares method is useful. Section 5 suggests a method for predicting the number of faults newly discovered by future test operations. The method is based on the expected value of the number of newly discovered faults obtained on condition that the cumulative number of faults is known.

2. Hyper-Geometric Distribution Software Reliability Growth Model

In this section we briefly review the HGDM. At the beginning of the test-and-debug process a software system is assumed to have m initial faults. Test operations performed in a day or a week may be called a test instance. Test instances are denoted by t_i, $i=1,2,\cdots$ in accordance with the order of applying them. The sensitivity factor, w_i, represents the number of faults discovered by the application of test instance t_i. Some of the faults detected by t_i may have been detected previously by the application of test instances t_j, $j=1,\cdots,i-1$. Hence, the number of faults newly discovered by t_i is not necessarily equal to w_i. That is, each detected fault can be classified into the two categories, newly discovered faults and rediscovered faults. Let N_i denote the number of faults newly discovered by t_i and $C_i=\sum_{j=1}^{i} N_j$ The following assumptions are made on the HGDM.

(1) No new faults are introduced into the software system during the debugging process.
(2) Sensitivity factor w_i, the number of faults discovered by t_i, is the faults taken randomly out of m initial faults.
(3) Sensitivity factor w_i is represented as a function of m and p_i, the progress in test-and-debugging, i.e., $w_i = m\,p_i$

The probability that x_i faults are newly discovered by t_i on the condition that C_{i-1} faults has been discovered up to t_{i-1} is then formulated as

$$P(N_i=x_i \mid C_{i-1}) = \frac{\binom{m-C_{i-1}}{x_i}\binom{C_{i-1}}{w_i-x_i}}{\binom{m}{w_i}}, \quad (1)$$

where $\max(0, w_i-C_{i-1}) \le x_i \le \min(w_i, m-C_{i-1})$ for $i=1,2,\cdots$ $C_0=0$ and $x_0=0$ Thus the con-

ditional expected value of N_i is

$$E(N_i|C_{i-1}) = (m - C_{i-1})p_i$$

The expected values of C_i was derived by Jacoby and Tohma[7] as

$$E(C_i) = m\left[1 - \prod_{j=1}^{i}(1-p_j)\right] \qquad (2)$$

The sensitivity factor plays an important role in the HGDM. Various functions for w_i have been devised and successfully applied to real data sets. Especially Hou, Kou and Chang [2] introduced two types of sensitivity factor based on the learning curve. They are respectively referred to as the exponential sensitivity factor and the logistic sensitivity factor. Functional forms of the sensitivity factor are presented in Table 1 of Jacoby and Tohma [7] and Minohara and Tohma [9].

3. Parameter Estimation

Let c_i and x_i be the observed values of C_i and N_i. Suppose that the software system is tested up to test instance t_n. The number of testers or computer time associated with each test instance is recorded and denoted by u_i. Then the available data consists of c_i (equivalently x_i) and u_i (if available), $i = 1, 2, \cdots, n$ In order to estimate the current number of residual faults and to predict the number of residual faults after applying t_{n+d} for $d \geq 1$, we first need to estimate the parameters in the model. Due to the mathematical difficulty of the maximum likelihood method, the least squares method has been used for the HGDM. Tohma et al. [13] obtained the least squares estimates by minimizing

$$\sum_{i=1}^{n}[c_i - E(C_i)]^2 \qquad (3)$$

This criterion was also employed in Hou, Kuo and Chang [2], [3] and Jacoby and Tohma [7]. However, Tohma et al. [14] minimized

$$\sum_{i=1}^{n}[x_i - E(N_i|C_{i-1})]^2 \qquad (4)$$

The minimization of expression (4) is equiva-lent to the minimization of

$$\sum_{i=1}^{n}[c_i - E(C_i|C_{i-1})]^2$$

since $E(C_i|C_{i-1}) = C_{i-1} + E(N_i|C_{i-1})$ and $C_i = C_{i-1} + N_i$. The distribution of N_i, given by expression (1), is dependent on C_{i-1}. C_{i-1} has been already realized and observed at the time when ith test instance is applied. The information contained in C_{i-1} should be utilized for predicting the expected value of C_i or N_i. Therefore, the minimization of expression (4) is more appropriate than the minimization of expression (3).

We should now note that the above two criteria assume that the variability of N_i, $i = 1, 2, \cdots$ are homogenous. But this assumption does not hold for the HGDM. The variance of N_i is obtained from the distribution of N_i as

$$Var(N_i|C_{i-1}) = \frac{(m - C_{i-1})C_{i-1}p_i(1-p_i)}{m-1}$$

Clearly $Var(N_i|C_{i-1})$ is not constant for all i. In the circumstances the weighted least squares method is ge nerally known to be adequate. Generally observaions s ubject to large variance are not as informative as obser vations subject to small variance. Therefore the weight for an observation should then be the reciprocal of its variance. We may consider the estimates minimizing

$$\sum_{i=1}^{n}\frac{[x_i - E(N_i|C_{i-1})]^2}{Var(N_i|C_{i-1})}$$

4. Application to Real Data Sets

The weighted least squares method is applied to

two real data sets. The first data set is presented in Tohma et al. [14] It was collected from a software system for monitering and real time control. The test data were recorded day by day. Thus the test operations performed in a day were regarded as a test instance. The number of test workers was also recorded for each test instance. Table 1 shows the estimation results of the least squares and weighted least squares methods. We assumed that $w_i = mu_i(\alpha i + \beta)$ as in Tohma et al. [14] Henceforth $\alpha^* = m\alpha$ $\beta^* = m\beta$and $\hat{}$ denotes the estimate.

〈Table 1〉 Estimation results for first data set

	least squares	weighted least squares
$\hat{m}$	497.2745	484.9785
$\hat{\alpha}^*$	0.0554	0.0854
$\hat{\beta}^*$	1.7487	1.1058

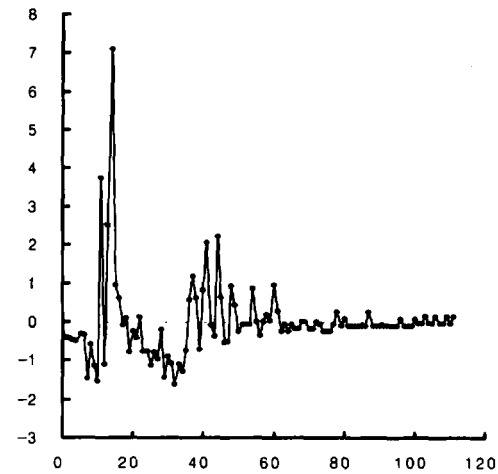
In order to validate and check model as- sumptions and model adequacy, the residual analysis should be considered. The studentized residuals obtained by the least squares method are plotted against i in Fig. 1. The three data points for $i = 11, 13, 14$ appears to be outliers. (The previous studies on this data set did not make any comment on this.) Outliers may occur by chance. Other causes such as the delayed report of fault detection may result in outliers. In order to make our discussion simple, we assume that the three data points are observed by chance. Thus we ignore the three data points and inspect Fig. 1. The variability of N_i is apparently nonhomogenuous. The variability increases in the early phase and then decreases. We thus employ the weighted least squares method. The studentized residuals obtained by the weighted least squares method are plotted in Fig. 2 Notice that three suspicious points are much more clearly revealed. By ignoring the three data points, we can see that the variability has been stabilized. Using the weighted least squares estimates, we can compute the following estimates:

$$\hat{E}(N_i | C_{i-1}) = (\hat{m} - c_{i-1})\hat{p}_i$$

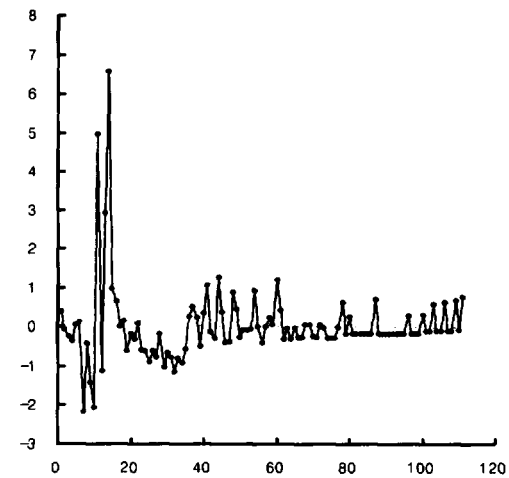
and

$$\hat{E}(C_i | C_{i-1}) = c_{i-1} + \hat{E}(N_i | C_{i-1})$$

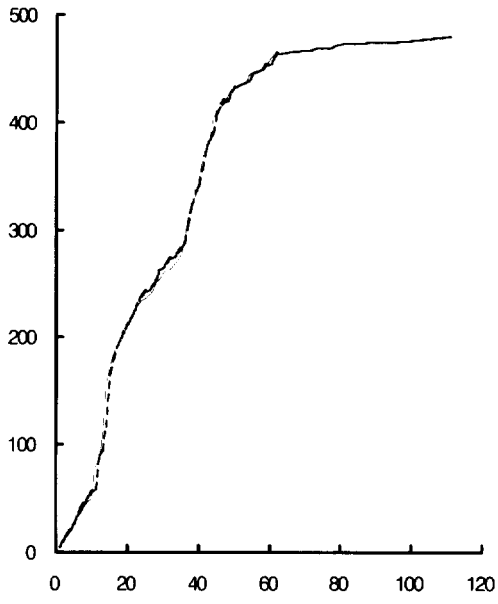
where $\hat{p}_i = u_i(\hat{\alpha} i + \hat{\beta})$. Fig. 3 shows the plots of c_i and $\hat{E}(C_i | C_{i-1})$depicted against i. $\hat{E}(C_i | C_{i-1})$ closely fits to c_i.



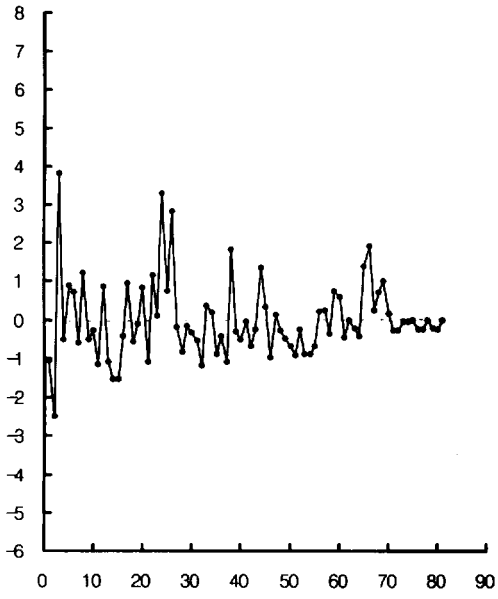
(Fig. 1) Plot of studentized residuals obtained by the least squares method. (first data set)



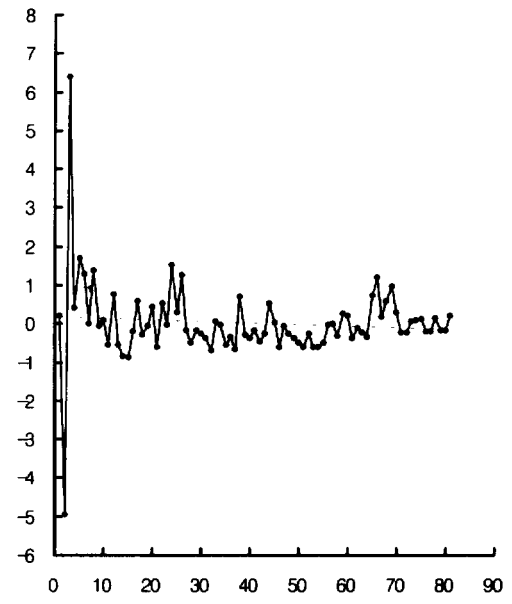
(Fig. 2) Plot of studentized residuals obtained by the weighted least squares method. (first data set)



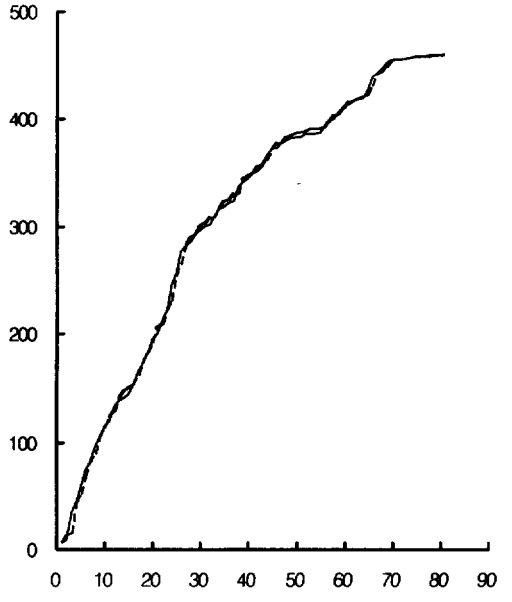
(Fig. 3) Plots of c_i and $\hat{E}(C_i|C_{i-1})$ against i. c_i : solid line, $\hat{E}(C_i|C_{i-1})$: dashed line. (first data set)



(Fig. 4) Plot of studentized residuals obtained by the least squares method. (second data set)



(Fig. 5) Plot of studentized residuals obtained by the weighted least squares method.(second data set)



(Fig. 6) Plots of c_i and $\hat{E}(C_i|C_{i-1})$ against i. c_i : solid line, $\hat{E}(C_i|C_{i-1})$: dashed line. (second data set)

The second data set was gathered from a switching system and presented in Kanoun et al. [8]. It is the collection of the cumulative number of discovered faults for 81 test instances. Since the number of test workers and/or the computer time are not recorded, this data set is analyzed under the assumption that $w_i = m(\alpha i + \beta)$. Table 2 and Fig. 4-6 show the results. The two data points for $i = 2, 3$ seems to be outliers. By simply ignoring the two data points and comparing Fig. 4 and Fig. 5, we find that the stabilization of variability has been achieved. Fig. 6 shows that the assumed model works well.

〈Table 2〉 Estimation results for second data set

	least squares	weighted least squares
$\hat{m}$	478.6827	464.9267
$\hat{\alpha}^*$	0.1787	0.4555
$\hat{\beta}^*$	11.3266	6.3174

5. Prediction

Suppose that we want to predict the number of faults newly detected by next d test instances. That is, we want the estimate of $E(N_{n+1} + \cdots + N_{n+d} | C_n)$. Once the estimate is obtained, we can easily compute the estimate of $E(C_{n+d} | C_n) = C_n + E(N_{n+1} + \cdots + N_{n+d} | C_n)$. Such a prediction problem occurs when we determine the software release time or further test instances required to meet the given software reliability objective. It can be shown that

$$E(N_{n+1} | C_n) = (m - C_n) p_{n+1}$$

$$\begin{aligned} E(N_{n+1} + N_{n+2} | C_n) &= E[E(N_{n+1} + N_{n+2} | C_n, N_{n+1}) | C_n] \\ &= E[(m - C_n - N_{n+1}) p_{n+2} + N_{n+1} | C_n] \\ &= (m - C_n)[p_{n+2} + (1 - p_{n+2}) p_{n+1}]. \\ &= (m - C_n)\left[1 - \prod_{j=1}^{2} (1 - p_{n+j})\right] \end{aligned}$$

and in general

$$E\left(\sum_{j=1}^{d} N_{n+j} | C_n\right) = (m - C_n)\left[1 - \prod_{j=1}^{d} (1 - p_{n+j})\right]. \quad (5)$$

$E(C_i)$ derived by Jocoby and Tohma [7] and given in expression (2), is actually $E(C_i | C_0)$. This can be verified by substituting n and d in expression (5) with 0 and i. By replaceing the parameters in expression (5) with corresponding estimates, we can predict the number of faults newly discovered by next d test instances.

Example Consider the second data set. The data is the collection of C_i, $i = 1, 2, \cdots, 81$. Suppose that we want to predict the number of faults newly discovered by next two test instances. This can be solved by estimating $E(N_{82} + N_{83} | C_{81})$. Since $c_{81} = 461$,

$$\begin{aligned} &\hat{E}(N_{82} + N_{83} | C_{81}) \\ &= (\hat{m} - c_{81})\left[1 - \prod_{j=1}^{2} (1 - \hat{p}_{81+j})\right] \\ &= 0.7065. \end{aligned}$$

6. Conclusion

This paper first proposed the method for estimating paramethers of HGDM. The previous studies suggested the least squares method. We argued that the weighted least squares method is more appropriate then the least squares method. It was illustrated by analyzing two real data set. We then proposed a new method for predicting the number of faults newly discovered by next test instances. It will be useful for determining the time to stop testing.

References

[1] A. L. Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," IEEE Trans. Software Eng., Vol.SE-11, pp.1411-1423, 1985

[2] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Appl ying Various Learning Curves to Hyper-Geometric Distribution Software Reliability Growth Model," Proc. 5th Int. Symp. on Software Reli ab. Eng., pp.7-16, 1994.

[3] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Hyper-Geometric Distribution Software Reliability Growth Model with Imperfect Debugging," Proc. 6th Int. Symp. Software Reliab. Eng., pp.195-200, 1995.

[4] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Opti mal Release Policy for Hyper-Geometric Distri bution Software-Reliability Growth Model," IE EE Trans. Reliability, Vol.45, pp.646-651, 1996.

[5] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Optimal Release Times for Software Systems with Scheduled Delivery Time Based on the HGDM," IEEE Trans. Computers, Vol.46, pp.216-221, 1997.

[6] R. Jacoby and Y. Tohma, "The Hyper-Geometric Distribution Software Reliability Growth Model (HGDM): Precise Formulation and Applicability," Proc. COMPSAC90, Chicago, pp.13-19, 1990.

[7] R. Jacoby and Y. Tohma, "Parameter Value Computation by Least Square Method and Evaluation of Software Availability and Reliability at Service-Operation by the Hyper-Geometric Distribution Software Reliability Growth Model (HGDM)," Proc. 13th Int. Conf. Software Eng., pp.226-237, 1991.

[8] K. Kanoun, M. R. Bastos Martini and J. Moriera de Souza, "A Method for Software Reliability Analysis and Prediction: Application to the Tropico-R Switching System," Research Report from LAAS-CNRS, France, 1989

[9] T. Minohara and Y. Tohma, "Parameter Estimation of Hyper-Geometric Distribution Software Reliability Growth Model by Genetic Algorithms," Proc. 6th Int. Symp. Software Reliab. Eng., pp.324-329, 1995.

[10] J. D. Musa, A. Iannino and K. Okumoto, 'Software Reliability: Measurement, Prediction, Application,' McGraw-Hill, pp.413, 1987.

[11] C. V. Ramamoorthy and F. B. Bastani, "Software Reliability-Status and Perspectives," IEEE Trans. Software Eng., Vol.SE-8, pp.354-371, 1982

[12] J. G. Shanthikumar, "Software Reliability Models: A Review," Microelectron. Reliab., Vol.23, pp.903-943, 1983

[13] Y. Tohma, K. Tokunaga, S. Nagase and Y. M urata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution," IEEE Trans. Software Eng., Vol.15, pp.345-355, 1989.

[14] Y. Tohma, H. Yamano, M. Ohba and R. Jacoby, "The Estimation of Parameters of the Hypergeometric Distribution and Its Application to the Software Reliability Growth Model," IEEE Trans. Software Eng., Vol.17, pp.483-489, 1991.

박 중 양

1982년 연세대학교 응용통계학과 (학사)

1984년 한국과학기술원 산업공학과 응용통계전공(석사)

1994년 한국과학기술원 산업공학과 응용통계전공(박사)

1984년~1989년 경상대학교 전산통계학과 교수

1989년~현재 경상대학교 통계학과 교수

관심분야 : 소프트웨어 신뢰성, 신경망, 선형 통계 모형, 실험계획법

유 창 열

1987년 경상대학교 전산통계학과 (학사)

1994년 경상대학교 대학원전자계산학과(석사)

1994년~1997년 경상대학교 대학원 박사과정 수료

1996년~현재 남해전문대학 사무자동화과 조교수

관심분야 : 소프트웨어 신뢰성, 객체지향 소프트웨어공학, 멀티미디어통신

이 부 권

1967년 진주농과대학

1978년 Michigan State Univ. 공과대학 전기 및 시스템 공학과 시스템공학전공

1980년~1989년 경상대학교 전산통계학과 교수

1989년~현재 경상대학교 컴퓨터과학과 교수

관심분야 : 시뮬레이션, 멀티미디어 통신