基于EM方法的隐Markov软件可靠性模型

张婷婷 张德平 刘国强
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 针对单一软件可靠性模型不能准确描述软件失效行为、无法合理准确地评估预测出软件可靠性的问题,将变点分析引入软件可靠性建模,提出了一种基于隐Markov过程的软件可靠性模型。该模型采用隐变量来描述影响软件可靠性的多种因素,通过隐变量的状态变化刻画出软件过程中各种因素的变化情况,构建出隐Markov链软件可靠性模型,并采用EM算法进行求解,通过实例分析来验证其有效性。实验结果表明,隐Markov链软件可靠性模型具有较强的变点检测能力,并能显著提高软件可靠性拟合精度。

关键词 软件可靠性,隐马尔科夫链模型,EM算法,变点分析

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.033

Hidden Markov Software Reliability Model with EM Method

ZHANG Ting-ting ZHANG De-ping LIU Guo-qiang
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract In view of the problem that single software reliability model doesn't precisely describe the failure behavior of the software, and doesn't accurately predict the software reliability, this paper studied a hidden Markov chain software reliability model incorporating the change point analysis. The formulation of the hidden Markov chain software reliability prediction approach involves a hidden state variable that indicates the regime change. This variable is specified to be detected by software failure data in each regime. The model parameters are estimated using expectation/maximization (EM) algorithm. Some numerical examples were performed based on some real software failure data sets. Experimental results show that the proposed framework to incorporate multiple change points for software reliability model has fairly accurate and efficient change-point detection capability, and can significantly improve software reliability fitting accuracy.

Keywords Software reliability, Hidden Markov chain model, Expectation/Maximization algorithm, Change point analysis

随着计算机技术的日益成熟,各种结构复杂、功能强大的计算机系统被广泛应用到航空航天、交通运输、核电能源和医疗卫生等关键安全领域。软件质量已经严重影响着人们的生活、学习和工作,并日益成为影响和推动社会生产力发展的重要因素,软件可靠性作为衡量软件质量的重要指标,其定量评估和预测已成为人们关注和研究的焦点[1]。

软件可靠性预测分析是利用软件可靠性模型对软件可靠性进行度量与预测的过程,软件可靠性模型是其核心与关键,对软件产品质量的度量和评估有重要的意义。最常见的软件可靠性模型是非齐次泊松过程(NHPP)模型[1,11,12],这类模型大多假设在软件故障检测期间,任意时刻软件故障引起的软件失效都相互独立且服从相同分布。这意味着在软件测试过程中,所有软件故障被检测到的概率相等,并且故障间的时间间隔变换速率保持不变。但实际上失效数据的分布由多种因素影响,例如运行环境、测试策略、资源分配等。一旦这些因素在软件测试阶段发生改变,将会导致软件失效强度函数非单调地递增或者递减,这就是所谓的变点问题。在软件可靠性评估中,如果存在变点,则应该同时考虑变点的影响,否则评估模型不能准确描述软件失效行为,从而影响软件可靠性评估与预测的精确性,可靠性建模中的变点问题已经开始引起广泛关注[13-27]。

已有的带变点可靠性建模研究主要包括:基于NHPP软件可靠性模型考虑各种失效度量指标函数带变点的可靠性模型,如Zhao[13]研究的失效强度函数带变点的JM模型;Chang[14]研究的失效分布函数为带变点Weibull分布的非齐次泊松过程可靠性模型;Jeong[15]和Zou[16]分别研究的失效率函数带变点的可靠性模型;Shyur[17]研究的不完美调试变点模型;Zhao[18]等提出的故障检测率带变点软件可靠性增长模型和引入带变点的测试工作量函数的NHPP软件可靠性模型。基于Markov链转换软件可靠性模型[27]以及隐Markov链的软件可靠性模型[29]用隐藏的状态过程来描述软件失效行为的改变,通过Bayesian方法或EM方法进行参数估计。本文在文献[29]的基础上,提出了一种带改变点连续隐Markov链可靠性模型,结合前向后向算法和Viterbi算法提

到稿日期:2015-07-27 返修日期:2015-11-30

张婷婷(1991-),女,硕士生,主要研究领域为软件测试与软件可靠性建模,E-mail:zhang7_ting@163.com;张德平(1973-),男,博士,硕士生导师,主要研究领域为软件测试与软件可靠性建模,E-mail:depingzhang@nuaa.edu.cn;刘国强(1990-),男,硕士生,主要研究领域为软件测试与软件可靠性建模,E-mail:liuguoqiangde@126.com。

出一种新的 EM 算法，其能快速有效地检测出模型的变点，并能有效地提高软件可靠性评估与拟合精度。

1 软件失效的隐 Markov 模型

为了便于应用隐 Markov 来描述软件失效行为，假定软件失效受一个潜在的过程 $\Lambda=\{\Lambda_t\}_{t\geqslant 1}$ 控制，Λ_t 称为“状态过程”；给定一个失效时刻 t，在 t 时刻的观察值失效时间间隔用 X_t 表示，观察到的软件失效数据序列记为 $X=\{X_t\}_{t\geqslant 1}$，则软件失效过程可以看作为一个隐马尔科夫链，其在有限的状态空间 $E=\{1,2,\cdots,K\}$ 上的转移矩阵为 P，其中 K 表示状态的个数，且 X_t 对应的隐藏状态的值为 i，X_t 服从参数为 λ_i 的指数分布，由式(1)给出：

$$X_t\mid\Lambda_t=i\sim\varepsilon(\lambda_i),i\in E \tag{1}$$

根据各个失效时间点，软件失效过程 $X=\{X_t\}_{t\geqslant 1}$ 可以划分为相应的不同时间段，每个时间段内的软件失效率 λ_i 保持不变。假定环境因素或其他因素的影响对软件失效产生的改变发生在相应的失效时间点 τ_i 上，即在第 i 次失效时，从状态为 Λ_i、失效率为 λ_i 转到状态为 Λ_{i+1}、失效率为 $\lambda_j(\lambda_i\neq\lambda_j)$，则时间 τ_i 可定义为一个变点(Change Point，CP)。因此整个软件失效过程 X 可用隐 Markov 过程描述，如图 1 所示。

$$X:X_1,\cdots,X_{\tau_1}\,;\,X_{\tau_1+1},\cdots,X_{\tau_2}\,;\,X_{\tau_2+1},\cdots,X_{\tau_{K-2}}\,;\,X_{\tau_{K-2}+1},\cdots X_{\tau_{K-1}}\,;\,X_{\tau_{k-1}+1},\cdots,X_n$$

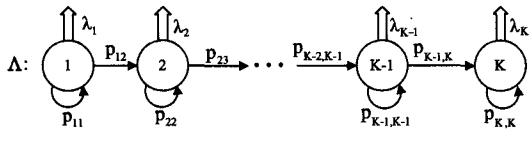

图 1 软件失效的隐 Markov 过程

形式上，整个软件失效过程可以描述为一个二元随机过程$\{(\Lambda_t,X_t),t=0,1,\cdots,N\}$，定义如下：

(1)隐状态过程 Λ_t 的初始概率向量为 π 并且满足：$\Lambda_0=1$，即 $\pi_1=1,\pi_k=0$，其中 $k=2,3,\cdots,K$。

(2)给定隐状态过程 Λ_t，X_t 是条件独立的，即：

$$\pi(X_1,\cdots,X_N\mid\Lambda_1,\cdots,\Lambda_N)=\prod_{t=1}^{N}\pi(X_t\mid\Lambda_1,\cdots,\Lambda_N) \tag{2}$$

(3)转移概率为 $P(\Lambda_{t+1}=j\mid\Lambda_t=i)=p_{ij}^{(t)}$，$t\geqslant 1,1<i,j<K$。即在时刻 $t+1$，系统以概率 $p_{ij}^{(t)}$ 由状态 i 变化为状态 j，则整个软件失效过程的状态转移矩阵记为 $P=(p_{ij}^{(t)})$。

(4)$\lambda_1,\cdots,\lambda_K$ 为不同状态下的软件失效率，则不同状态下的观察概率为 $P(X_t\mid\Lambda_t)$，由于 $X_t\mid\Lambda_t$ 服从参数为 λ_i 的指数分布，观察概率可表示为：

$$P(X_t\mid\Lambda_t)=\lambda_i e^{-\lambda_i x_t},\Lambda_t=i \tag{3}$$

记隐状态序列为 $z=(z_1,z_2,\cdots,z_N)$，易知 $z=(z_1,z_2,\cdots,z_N)$ 与改变点列 $t=(t_0,t_1,\cdots,t_K)$ 具有一一对应关系，给定一个序列 z，便可以获得对应的改变点列 t，即改变点列元素 t_k 满足 $z_{t_k}\neq z_{t_k+1}$，其中 $k=1,2,\cdots,K-1$，并且$(t_0=0,t_K=N)$，因此 $K\leqslant N$，即长度为 N 的失效序列最多具有 N 个改变点。每一个时间段都与一个状态唯一相关联，则变点确定问题可以转化为估计潜在状态序列 $z_1,z_2,\cdots,z_N$ 问题。

2 参数估计及模型选择

易知，观察序列 $x=(x_1,x_2,\cdots,x_N)$ 是由 3 个参数 $K,P,\theta=(\lambda_1,\lambda_2,\cdots,\lambda_K)$ 确定的隐 Markov 模型生成。由观察序列 x 可以计算每一个状态序列 z 的可能性(即条件概率)，从而可以获得状态序列的最大似然估计 $\hat{z}$ 以及改变点列 t 的最大似然估计 $\hat{t}$。其中改变点列 t 的最大似然估计 $\hat{t}$ 可以通过状态序列的最大似然估计 $\hat{z}=(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)$ 获得。由于状态序列是不可能被观察的，只能根据观察序列 $x=(x_1,x_2,\cdots,x_N)$ 以及参数 K,P,θ 估计 $\hat{z}$。实际上，由于 K,P,θ 未知，对该隐 Markov 模型时间分段进行最大似然估计时，采用 EM 算法进行：1)参数估计；2)时间序列分段，并且这种过程不断重复直到收敛。

2.1 状态估计

假设观察序列 $x=(x_1,x_2,\cdots,x_N)$ 以及参数 K,P,θ 是确定的，则 $\hat{z}=(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)$ 可以通过最大似然估计法获得。状态序列 z 以及观察序列 x 的联合似然概率记为 $\ell(z,x;P,\theta,K)$，则

$$\begin{aligned}\ell(z,x;P,\theta,K)&=\ell(z_1,\cdots,z_N,x_1,\cdots,x_N;P,\theta,K)\\&=\lambda_{z_1}e^{-\lambda_{z_1}x_1}\prod_{t=2}^{N}(P_{z_{t-1},z_t}\cdot\lambda_{z_t}e^{-\lambda_{z_t}x_t})\end{aligned} \tag{4}$$

根据前面的假设可知 $z_1=1$。

给定一个观察序列 x 对应的状态序列 z 的条件似然函数，记为：$\ell(z\mid x;P,\theta,K)$，易得

$$\ell(z\mid x;P,\theta,K)=\frac{\ell(x,z;P,\theta,K)}{A(x;P,\theta,K)} \tag{5}$$

其中，$A(x;P,\theta,K)$ 是序列 $x_1,x_2,\cdots,x_N$ 的边缘概率密度，并且与 z 无关。

给定一个失效时间序列 x，其对应状态序列的最大似然估计 $\hat{z}$ 应为使得关于 z 的条件似然函数 $\ell(z\mid x;P,\theta,K)$ 值最大的序列，即

$$(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)=\arg\max_{z_1,\cdots,z_N}\ell(z_1,,z_N\mid x,P,\theta,K) \tag{6}$$

状态序列的最大似然估计 $\hat{z}=(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)$ 可以由 Viterbi 算法[30]获得。式(6)可改写为：

$$\begin{aligned}(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)&=\arg\max_{z_1,\cdots,z_N}\ell(z_1,\cdots,z_N,x_1,\cdots,x_N;P,\theta,K)\\&=\arg\max_{z_1,\cdots,z_N}\prod_{t=1}^{N}(P_{z_{t-1},z_t}\cdot\lambda_{z_t}e^{-\lambda_{z_t}x_t})\end{aligned} \tag{7}$$

对于给定的 $t=1,2,\cdots,N$ 以及 $k=1,2,\cdots,K$，定义：

$$q_{k,t}=\max_{z_1,\cdots,z_{t-1}}\ell(z_1,\cdots,z_{t-1},k,x_1,\cdots,x_t;P,\theta,K) \tag{8}$$

则最佳隐藏状态序列 $\hat{z}=(\hat{z}_1,\hat{z}_2,\cdots,\hat{z}_N)$ 以及 $q_{k,t}$ 都可以通过算法 1 递归计算。

算法 1 Viterbi 算法

输入：时间序列 $x_1,x_2,\cdots,x_N$ 以及参数 P,θ,K

前向递归

初始化：$q_{1,0}=1,q_{2,0}=q_{3,0}=\cdots=q_{k,0}=0$

For t=2,…,N

For k=1,2,…,K

$q_{k,t}=\max_{1\leqslant j\leqslant K}(q_{j,t-1}\cdot P_{j,k}\cdot\lambda_k e^{-\lambda_k x_t})$

$\tau_{k,t}=\arg\max_{1\leqslant j\leqslant K}(q_{j,t-1}\cdot P_{j,k}\cdot\lambda_k e^{-\lambda_k x_t})$

EndFor

EndFor

后向递归

$\hat{l}=\max_{1\leqslant k\leqslant K}(q_{k,N})$

$\hat{z}_N = \arg\max_{1\leqslant k\leqslant K}(q_{k,N})$

For t=N,N−1,…,2

$\hat{z}_{t-1} = r_{\hat{z}_t,t}$

EndFor

输出：$\hat{z} = (\hat{z}_1, \hat{z}_2, \cdots, \hat{z}_N)$

2.2 参数估计

当变点时间序列 $t=(t_0,t_1,\cdots,t_K)$ 确定后，根据给定的分段可以对 $\theta=(\lambda_1,\lambda_2,\cdots,\lambda_K)$，$k=1,2,\cdots,K$ 进行合理的估计：

$$\hat{\lambda}_k = \frac{T_k}{\sum_{t=t_{k-1}+1}^{t_k} x_t} \tag{9}$$

其中，$T_k=t_k-t_{t-1}$ 表示第 k 个时间段的失效次数，x_t 表示第 t 次失效的时间间隔。

由图 1 可知，对于 $k=1,2,\cdots,K$，每一个状态至少都会遍历一次，并且所有的 j 不等于 k 和 $k+1$ 时 $p_{k,j}=0$，同时满足对于 $k=1,2,\cdots,K-1$ 有 $p_{k,k+1}=1-p_{k,k}$。因此矩阵 P 只有 $K-1$ 个自由参数，即 $p_{1,1},p_{2,2},\cdots,p_{K-1,K-1}$，其中 $p_{K,K}=1$。对于矩阵 P，比较常见的估计方法是前向后向算法[29]。

对于 $k=1,2,\cdots,K$ 以及 $t=1,2,\cdots,N$，定义前向概率 $\alpha_{k,t}$ 以及后向概率 $\beta_{k,t}$ 如下：

$$\alpha_{k,t}=P(X_1=x_1,X_2=x_2,\cdots,X_t=x_t,Z_t=k) \tag{10}$$

$$\beta_{k,t}=P(X_{t+1}=x_{t+1},\cdots,X_N=x_N\,|\,Z_t=k) \tag{11}$$

记 p 第 i 次估计为 $p^{(i)}$，并且注意

$$p_{k,k}^{(i)}=p^{(i)},\ p_{k,k+1}^{(i)}=1-p^{(i)},\ p_{k,n}^{(i)}=0 \tag{12}$$

其中，$n\neq k$ 且 $n\neq k+1$。很容易得到下列递推公式：

$$\alpha_{1,1}=\lambda_1 e^{-\lambda_1 x_1},\ \alpha_{k,1}=0,k=2,3,\cdots,K \tag{13}$$

$$\alpha_{k,t+1}=\sum_{n=1}^{K}\alpha_{n,t}\cdot p_{n,k}^{(i)}\cdot\lambda_{z_{t+1}}e^{-(\lambda_{z_{t+1}}x_{t+1})}$$

$$k=1,2,\cdots,K,t=1,2,\cdots,N-1 \tag{14}$$

$$\beta_{k,N}=1,k=1,2,\cdots,K \tag{15}$$

$$\beta_{k,t}=\sum_{n=1}^{K}p_{k,n}^{(i)}\cdot\lambda_{z_{t+1}}e^{-(\lambda_{z_{t+1}}x_{t+1})}\cdot\beta_{n,t+1}$$

$$k=1,2,\cdots,K,t=N-1,N-2,\cdots,1 \tag{16}$$

进一步可以发现：

$$P(O)=\sum_{k=1}^{K}P(O,Z_t=k)=\sum_{k=1}^{K}\alpha_{k,t}\beta_{k,t},t=1,2,\cdots,N \tag{17}$$

用 $S_{t,k,n}$ 表示 t 时刻状态由 k 转移到 $t+1$ 时刻的 n 的概率，即

$$S_{t,k,n}=P(Z_t=k,Z_{t+1}=n|O)=\frac{P(Z_t=k,Z_{t+1}=n,O)}{P(O)}=\frac{\sum_{t=1}^{N-1}\alpha_{k,t}\cdot p_{k,n}\lambda_{z_{t+1}}e^{-(\lambda_{z_{t+1}}x_{t+1})}\cdot\beta_{n,t+1}}{\sum_{k=1}^{K}\alpha_{k,t}\beta_{k,t}} \tag{18}$$

用 $W_{t,k}$ 表示 t 时刻状态为 k 的概率，即

$$W_{t,k}=P(Z_t=k|O)=\frac{P(Z_t=k,O)}{P(O)}=\frac{\alpha_{k,t}\beta_{k,t}}{\sum_{k=1}^{K}\alpha_{k,t}\beta_{k,t}} \tag{19}$$

给定 $p^{(i)}$，根据前向后向算法[29]，$p^{(i+1)}$ 的估计表达式为：

$$p^{(i+1)}=\frac{\text{状态由 } k \text{ 转到 } k \text{ 的平均次数}}{\text{由状态 } k \text{ 转到其他状态的平均次数}}=\frac{\sum_{t=1}^{N-1}S_{t,k,k}}{\sum_{t=1}^{N-1}W_{t,k}} \tag{20}$$

参数估计以及变点的估计可以采用改进的 EM 算法（见算法 2）实现，此时最大似然估计为隐藏序列 $z=(z_1,z_2,\cdots,z_N)$ 和参数 θ 同时取值使得似然函数达到最大。

算法 2 改进的 EM 算法

输入：观察状态序列，即失效时间间隔序列 $x_1,x_2,\cdots,x_N$，参数 K 的值，以及终止条件 ε 的值

初始化：随机生成一组 K 个隐藏状态的隐藏序列 $z^{(0)}=(z_1^{(0)},z_2^{(0)},\cdots,z_N^{(0)})$，并且随机生成一个转移矩阵 P

For i=1,2,…

E 步：参数估计

通过 $z^{(i-1)}$ 计算 $t^{(i)}$；

通过 $t^{(i)}$ 以及式(8)计算 $\theta^{(i)}$；

通过前向后向算法计算 $P^{(i)}$；

M 步：状态估计

根据 X,K,$P^{(i)}$,$\theta^{(i)}$，用 Viterbi 算法算出最有可能的隐藏序列 $z^{(i)}$

终止条件：

If $|\ell(z^{(i)},x;P^{(i)},\theta^{(i)})-\ell(z^{(i-1)},x;P^{(i-1)},\theta^{(i-1)})|<\varepsilon$

$\hat{z}=z^{(i)}$

$\hat{\theta}=\theta^{(i)}$

Exit the loop

Endif

EndFor

根据 $\hat{z}$ 计算 $\hat{t}$

输出：$\hat{t}$ 和 $\hat{\theta}$

与 EM 算法在 HMM 中的经典应用 Baum-Welch 算法[29]相比，提出的改进的 EM 算法主要在以下 3 个方面进行改进：

1）改进的 EM 算法在进行参数估计的同时对最佳隐藏序列估计（数据集分段），每次迭代中修正最佳隐藏序列的估计更新都用于参数估计值的更新，使算法更易收敛；

2）改进的 EM 算法在用前向后向算法以及 Baum-Welch 公式计算转移矩阵 P 时，充分考虑参数取值与最佳隐藏序列对其的影响，每次迭代都利用最近更新参数值与最佳隐藏序列的估计进行更新修正，使算法更灵敏；

3）改进的 EM 算法估计使用的似然函数与经典 EM 算法不同，经典 EM 算法采用观察概率作为似然函数，本文采用观察状态与隐藏状态的联合概率作为似然函数，充分利用有用信息来指导参数估计，使模型更加准确。

2.3 模型选择

该算法在不断增值的 $K=1,2,3,\cdots$ 失效时间序列上运行，并且可以获得每个 K 值的最佳时间段。对于每一个 K 值，对应的隐马尔科夫链有 K 个隐藏的状态。确定 K 的值非常困难，并且这是模型选择中的一个特殊情况。本文采用贝叶斯信息（BIC）准则，该准则定义为：

$$BIC(K)=\log\ell(z,x,P,\theta,K)-\frac{\upsilon(K)}{2}\times\log(n) \tag{21}$$

其中，$\log\ell(z,x,P,\theta,K)$ 为具有 K 个隐藏状态的 Markov 链的对数似然估计。$\upsilon(K)=K+(K-1)=2K-1$ 为隐 Markov 模型独立参数的个数，K 个时间段的失效数据分别服从参数为 λ_i 的指数分布，其参数的个数为 K，转移矩阵的参数个数为 $K-1$。n 为观察序列的长度，即失效数据的序列长度。

将 K 分别从 1 到 $K_{\max}$ 进行取值来计算 BIC 值，选择使得 BIC 值最大的一组 K,P,θ 参数值作为最佳估计结果。

3 实例验证与结果分析

为了更好地验证本文所提的带改变点的隐 Markov 软件

可靠性模型的有效性，将该变点检测方法运用到软件可靠性领域常用的两个模型(JM 模型和 SVR 模型)中。同时，为了比较本文提出的变点检测方法的有效性，将其与基于算术平均因子和 Laplace 因子[31]的变点检测方法进行对比。对比实验包括 6 组模型之间的比较：基于本文提出的隐 Markov 模型变点检测方法的 JM 模型和 SVR 模型(分别记为 HM-JM 和 HM-SVR)、基于算术平均因子和 Laplace 因子[31]的 JM 模型和 SVR 模型(分别记为 AM&L-JM 和 AM&L-SVR)、JM 模型、SVR 模型。数据集采用文献[29]中的部分数据集(M1、M2 以及 M40)进行验证。

带变点的软件可靠性模型的验证步骤分为 3 步：1)变点位置检测，将数据分段；2)针对数据中的每一段分别用基本模型进行拟合；3)拟合有效性分析。

采用软件可靠性领域中常用的 4 种拟合或预测评价指标[32]：*AE*、*MAE*、*MSPE*、*R-Square*。其中，*AE* 值、*MAE* 值和 *MSPE* 值越小，*R-Square* 值越接近于 1，表明模型拟合或者预测值与真实值越接近，拟合或者预测性能越高。

3.1 M1 应用分析

失效数据集 M1 共有 136 个数据，其不同 K 值的最佳分段如表 1 所列，由表 1 可看出，当 $K=3$ 时，*BIC* 值最大，即隐藏状态个数为 3，且分段[16,78,136]作为最终的分段估计，变点位置为 16 和 78。

表 1　数据集 M1 的不同 K 值的最佳分段

K	分段边界(变点位置)	BIC 值
1	136	−1019.7577244
2	78,136	−987.7612364
3	16,78,136	**−986.1606577**
4	16,17,78,136	−990.5436752
5	1,14,16,78,136	−995.4698686

M1 的最佳分段结果如图 2 所示，该分段比较符合原失效序列的趋势。

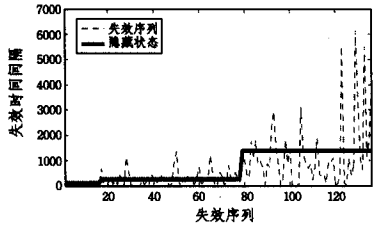

图 2　数据集 M1 的隐藏状态划分

对于失效数据集 M1，采用算术平均因子和 Laplace 因子检测的变点位置为 16，即分段为[16,136]。其分别采用 JM，SVR，HM-JM，HM-SVR，AM&L-JM 模型以及 AM&L-SVR 模型进行拟合，拟合精度分析的结果如表 2 所列。

表 2　数据集 M1 的 6 种拟合模型有效性比较分析

预测模型	评价指标			
	R-Square	AE	MSPE	MAE
JM	0.899	1261.090	7.101	157.577
AM&L-JM	0.925	912.682	5.298	148.892
HM-JM	**0.947**	**894.547**	**5.276**	**123.104**
SVR	0.929	911.863	5.298	142.430
AM&L-SVR	0.935	908.391	5.293	138.024
HM-SVR	**0.981**	**783.949**	**4.939**	**68.145**

由表 2 可知，首先，带变点的 JM 模型和 SVR 模型的拟合精度优于原 JM 模型和原 SVR 模型。其次，HM-JM 模型和 HM-SVR 模型分别与对应的 AM&L-JM 模型以及 AM&L-SVR 模型都具有明显的优势。本文提出的基于 HMM 的变点检测方法检测出数据集 M1 存在两个变点，而文献[31]的单变点检测方法只能检测出一个变点位置，这表明当数据序列可能存在多个变点时，提出的变点检测方法具有一定的优势。

3.2 M2 应用分析

失效数据集 M2 共有 54 个数据，其不同 K 值的最佳分段如表 3 所列。由表 3 可看出，当 $K=2$ 时，*BIC* 值最大，即隐藏状态个数为 2，且分段[25,54]为最终的分段估计，变点位置为 25。

表 3　数据集 M2 的不同 K 值的最佳分段

K	分段边界(变点位置)	BIC 值
1	54	−466.7960696
2	25,54	**−456.7722094**
3	25,41,54	−461.6244041
4	12,25,41,54	−465.7783718
5	1,26,33,41,54	−466.8338778

M2 的最佳分段结果如图 3 所示，该分段比较符合原失效序列的趋势。

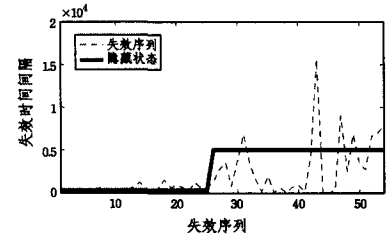

图 3　数据集 M2 的隐藏状态划分

对于失效数据集 M2，采用算术平均因子和 Laplace 因子检测的变点位置为 26，即分段为[26,54]。其分别采用 JM，SVR，HM-JM，HM-SVR，AM&L-JM 以及 AM&L-SVR 模型进行拟合精度分析，其结果如表 4 所列。

表 4　数据集 M2 的 6 种拟合模型的有效性比较分析

预测模型	评价指标			
	R-Square	AE	MSPE	MAE
JM	0.880	5018.482	53.995	685.098
AM&L-JM	0.938	3568.673	27.872	366.863
HM-JM	**0.957**	**1259.804**	**12.143**	**174.139**
SVR	0.955	1268.887	12.701	180.311
AM&L-SVR	0.959	1245.734	10.701	146.774
HM-SVR	**0.998**	**220.839**	**1.558**	**4.492**

由表 4 可知，其与数据集 M1 的实验结果一致。首先，不论是 JM 模型还是 SVR 模型，带变点的模型与原模型相比具有一定的优势。其次，HM-JM 模型和 HM-SVR 模型分别与对应的 AM&L-JM 模型以及 AM&L-SVR 模型相比效果更优。这表明，同样是在单变点的情况下，本文基于 HMM 的变点检测方法检测出的变点位置可能更加精确，但仍有待进一步研究。

3.3 M40 应用分析

仿真数据集 M40 共有 101 个数据，其不同 K 值的最佳分段如表 5 所列。由表 5 可看出，当 $K=3$ 时，*BIC* 值最大，即隐藏状态个数为 3，且分段[53,67,101]作为最终的分段估计，变点位置为 53,67。

表 5　数据集 M40 的不同 K 值的最佳分段

K	分段边界(变点位置)	BIC 值
1	101	−1332.9317021
2	61,101	−1251.6815092
3	53,67,101	**−1249.8782837**
4	1,53,67,101	−1251.3250856
5	43,44,61,67,101	−1258.6403824

M40 的最佳分段结果如图 4 所示，该分段比较符合原失效序列的趋势。

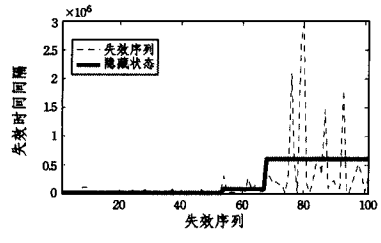

图 4　数据集 M40 的隐藏状态划分

对于失效数据集 M40，采用算术平均因子和 Laplace 因子检测的变点位置为 53，即分段为[53,101]。其分别采用 JM,SVR,HM-JM,HM-SVR,AM&L-JM 以及 AM&L-SVR 模型进行拟合精度分析，其结果如表 6 所列。

表 6　数据集 M40 的 6 种拟合模型有效性比较分析

预测模型	评价指标			
	R-Square	AE	MSPE	MAE
JM	0.883	450.939	5.983	23829.238
AM&L-JM	0.943	205.090	0.488	13787.330
HM-JM	**0.983**	**55.214**	**0.283**	**9072.129**
SVR	0.942	204.793	0.476	15698.587
AM&L-SVR	0.950	171.260	0403	12464.172
HM-SVR	**0.997**	**50.258**	**0.282**	**6756.905**

由表 6 可知，带变点的 JM 模型和 SVR 模型仍然优于不带变点的 JM 模型和 SVR 模型。对两种变点检测结果进行建模，实验结果同样与前文一致，HM-JM 模型和 HM-SVR 模型分别与对应的 AM&L-JM 模型以及 AM&L-SVR 模型相比优势更明显。

3.4　实验结果及分析

首先，将所提的带改变点的隐 Markov 链的变点检测方法与基于算术平均因子和 Laplace 因子[31]的变点检测方法进行对比。主要结果有以下几个：

1)针对提出的基于 EM 算法的隐 Markov 模型变点检测方法，根据表 1、表 3、表 5 可以看出，对于不同的 K 值，某些位置会多次被检测为变点，说明提出的变点检测方法的参数初始值对于最终的变点检测结果影响不大。

2)采用了 3 组数据集，最终 K 值的取值都较小，在 1～3 之间，说明 BIC 准则对于 K 值较小的占有优势，这一点在分析 BIC 准则公式时亦可得出。

3)提出的基于 EM 算法的隐 Markov 模型变点检测方法是典型的多变点检测方法。而基于算术平均因子和 Laplace 因子[31]的变点检测方法是典型的基于软件可靠性增长模型的单变点检测方法。其劣势有两个：1)没有考虑不完美调试的存在，对于非可靠性增长数据不适用；2)该方法是单变点检测。3 组数据集中，对于数据集 M1、M40，隐 Markov 模型变点检测结果是多变点。根据表 2、表 4 可知，对于明显存在多个变点的数据而言，提出的多变点检测方法与算术平均和 Laplace 因子[31]的变点检测方法相比具有一定的优势。

同时为了验证提出的改进的 EM 算法与经典的 Baum-Welch 算法[29]相比在收敛速度上的有效性，实验采用了大量的数据集进行验证，并且对两种算法在整个程序执行过程中的时间进行了统计，实验结果中绝大部分数据集上两种算法的分段结果完全一致，只有少部分数据集上的分段结果略有差异。但在整个过程的执行时间上，提出的算法不及经典 EM 算法的一半，具有明显的优势。这表明，提出的改进的 EM 算法虽在每次重估参数的同时由于采用 Viterbi 算法对隐藏状态进行估计而耽误了一定的时间，但是通过 Viterbi 算法对隐藏状态估计的结果进行参数更新，以能够快速准确地识别最终的变点位置，加快收敛速度，从而大大缩减了整个过程的时间，具有一定的应用前景。

结束语　软件失效数据的分布受多种因素影响，因此近年来越来越多的学者专注于失效数据变点的研究。本文在文献[29]的基础上，提出了一种带改变点的连续隐 Markov 链可靠性模型，并且结合前向后向算法和 Viterbi 算法对传统的 EM 算法进行了改进，以准确快速地检测出数据序列的变点位置。本文还将基于该变点检测方法与文献[31]的算术平均和 Laplace 因子的单变点检测方法的变点模型以及基本模型进行对比实验。实验结果表明，在基本模型中加入变点可明显提高数据集的拟合精度，并且与采用文献[31]的单变点检测方法相比，不管是对于数据集的适应性、变点位置的准确性还是变点个数，本文提出的变点检测方法都具有明显的优势，进一步提高了软件可靠性的拟合精度。

参 考 文 献

[1] Lyu M R. Handbook of software reliability engineering [M]. New York:McGraw-Hill,1996

[2] Yang B,Li X,Xie M,et al. A generic data-driven software reliability model with model mining technique[J]. Reliability Engineering and System Safety,2010,95(6):671-678

[3] Chatterjee S,Misra R B,Alam S S. Predeiction of software reliability using an auto regressive process[J]. International Journal of Systems Science,1997,28(2):211-216

[4] Guo J H,Liu H W,Yang X Z. An autoregressive time series software reliability growth model with independent increment [C]//Proceedings of the 7th International Conference on Mathematical Methods and Computational Techniques In Electrical Engineering. World Scientific and Engineering Academy and Society (WSEAS). 2005:362-366

[5] Raja U,Hale D P,Hale J E. Modeling software evolution defects:a time series approach[J]. Journal of Software Maintenance and Evolution:Research and Practice,2009,21(1):49-71

[6] Bao Y K,Yi D B,Chen J H. Forecasting software reliability using ARIMA with ensemble emirical mode decomposition[J]. International Journal of Modeling and Simulation,2012,32(2):104-110

[7] Amin A,Grunske L,Colmn A. An approach to software reliability prediction based on time series modeling[J]. The Journal of Systems and Software,2013,86(7):1923-1932

[8] Zheng J. Predicting software reliability with neural network ensembles[J]. Expert Systems with Applications,2009,36(2):2116-2122

[9] Hu Q P, Xie M, Ng S H. Software Reliability Predictions using Artificial Neural Networks[M]// Computational Intelligence in Reliability Engineering. Springer Berlin Heidelberg, 2007: 197-222

[10] Moura M C, Zio E, Lins I D, et al. Failure and reliability prediction by support vector machines regression of time series data [J]. Reliability Engineering & System Safety, 2011, 96(11): 1527-1534

[11] Jelinski Z, Moranda P. Software reliability research[J]. Statistical Computer Performance Evaluation. Freiberger W, Ed. Academic Press: New York, 1972: 465-484

[12] Goel A L, Okumoto K. Time dependent error detection rate model for software reliability and other performance measures [J]. IEEE Transactions on Reliability, 1979, 28(3): 206-211

[13] Zhao M. Change-point problems in software and hardware reliability[J]. Communications in Statistics-Theory and Methods, 1993, 22(3): 757-768

[14] Chang Y P. Estimation of parameters for nonhomogeneous Poisson process: Software reliability with change-point model[J]. Communications in Statistics-Simulation and Computation, 2001, 30(3): 623-635

[15] Jeong K M. An adaptive failure rate change-point model for software reliability[J]. International Journal of Reliability and Applications, 2001, 2(3): 199-207

[16] Zou F Z. A change-point perspective on the software failure process[J]. Software Testing, Verification and Reliability, 2003, 13(2): 85-93

[17] Shyur H J. A stochastic software reliability model with imperfect debugging and change-point[J]. Journal of Systems and Software, 2003, 66(2): 135-141

[18] Zhao J, Liu H W, Cui G, et al. Software reliability growth model with change-point and environmental function[J]. Journal of Systems and Software, 2006, 79(11): 1578-1587

[19] Huang C Y. Performance analysis of software reliability growth models with testing-effort and change-point[J]. Journal of Systems and Software, 2005, 76(2): 181-194

[20] Wang Z, Wang J. Parameter estimation of some NHPP software reliability models with change-point[J]. Communications in Statistics-Simulation and Computation, 2005, 34(1): 121-134

[21] Kapur P K, Singh V B, Anand S, et al. Software reliability growth model with change-point and effort control using a power function of testing time[J]. International Journal of Production Research, 2008, 46(3): 771-787

[22] Lin C T, Huang C Y. Enhancing and measuring the predictive capabilities of the testing-effort dependent software reliability models[J]. Journal of Systems and Software, 2008, 81(6): 1025-1038

[23] Li X, Xie M, Ng S H. Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points[J]. Applied Mathematical Modelling, 2010, 34(11): 3560-3570

[24] Huang C Y, Hung T Y. Software reliability analysis and assessment using queueing models with multiple change-points[J]. Computers and Mathematics with Applications, 2010, 60(7): 2015-2030

[25] Huang C Y, Lyu M R. Estimation and analysis of some generalized multiple change-point software reliability models[J]. IEEE Transactions on Reliability, 2011, 60(2): 498-514

[26] Singh O, Anand A, Singh J, et al. Assessment of distribution based SRGM with the effect of change-point and imperfect debugging incorporating irregular fluctuations[J]. Journal of Pure and Applied Science & Technology, 2012, 2(1): 37-49

[27] Ravishanker N, Liu Z H, Ray B K. NHPP models with Markov switching for software reliability[J]. Computational Statistics and Data Analysis, 2008, 52(8): 3988-3999

[28] Nam S, Cha J H, Cho S. A Bayesian Change-Point Analysis for Software Reliability Models[J]. Communications in Statistics-Simulation and Computation, 2008, 37(9): 1855-1869

[29] Durand J B, Gaudoin O. Software reliability modelling and prediction with hidden Markov chain[J]. Statistical Modelling, 2005, 5(1): 75-93

[30] Forney Jr G D. The Viterbi algorithm[J]. Proceedings of the IEEE, 1973, 61(3): 268-278

[31] Inoue S, Hayashida S, Yamada S. Toward Practical Software Reliability Assessment with Change-Point Based on Hazard Rate Models[C]// 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2013: 268-273

[32] Wang Shuai. Software Reliability Forecasting Method Based on Decomposition and Reconstruction of Series[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2014 (in Chinese)
汪帅. 基于序列分解与重构的软件可靠性预测方法[D]. 南京: 南京航空航天大学, 2014

(上接第 141 页)

[10] Chai M, Schlingloff B H. Monitoring Systems with Extended Live Sequence Charts[C]// 14th International Conference on Runtime Verification. Springer, 2014: 48-63

[11] Kugler H, Harel D, Pnueli A, et al. Temporal logic for scenario-based specifications[C]// Proc. of the 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS05). 2005, 3440: 445-460

[12] Clavel, Duran, Marti-Oliet, et al. Maude Manual (version 2.6) [M]. University of Illinois, Urbana-Champaign 1, 2011: 1-491

[13] Martí—Oliet N. An Introduction to Maude and Some of Its Applications[M]// Practical Aspects of Declarative Languages. Springer Berlin Heidelberg, 2010: 4-9

[14] Zhao Lin, Tang Tao, Xu Tian-hua, et al. Runtime Verification and its Applications in Train Control System[J]. Railway Society, 2011, 33(12): 65-71 (in Chinese)
赵琳, 唐涛, 徐田华, 等. 运行时验证及其在列车运行控制系统中的应用[J]. 铁道学报, 2011, 33(12): 65-71

[15] Niu Ru, Cao Yuan, Tang Tao. Formal Modelling and Analysis of RBC Handover protocol for ETCS Level 2 Using Stochastic Petri Nets[J]. Railway Society, 2009, 31(4): 52-58 (in Chinese)
牛儒, 曹源, 唐涛. ETCS-2 级列控系统 RBC 交接协议的形式化分析[J]. 铁道学报, 2009, 31(4): 52-58

[16] ERTMS/ETCS Subset-098: ERTMS/ETCs Class 1 RBC-RBC safe communication interface[EB/OL]. http://www.aeif.org/db/docs/ccm/SUBSET-052 v222. 2005

[17] Zhao Lin, Tang Tao, Xu Tian-hui, et al. Runtime Verification and its Applications in Train Control Systems[J]. Journal of The China Railway Society, 2011, 33(12): 65-71 (in Chinese)
赵琳, 唐涛, 徐田华, 等. 运行时验证及其在列车运行控制系统中的应用[J]. 铁道学报, 2011, 33(12): 65-71