A Bayesian hidden Markov model for software failures

A. Pievatolo\(^1\), F. Ruggeri\(^1\) and R. Soyer\(^2\)

\(^1\) CNR IMATI, Via Bassini 15, 20133 Milano, Italy
\(^2\) Department of Decision Sciences, George Washington University Washington, D.C. 20052, U.S.A.

Abstract. The paper presents the Bayesian analysis of a new model in software reliability, aimed to detect deterioration due to potential introduction of new bugs into the software during the development phase. Since the introduction of bugs is an unobservable process, latent variables are introduced to incorporate this characteristic into the proposed hidden Markov model. The state of the latent process reflects the effectiveness of the interventions, that is, the design changes to the software after each observed failure time. The Bayesian estimation (via MCMC) of the state of the latent process is a straightforward matter, if the dimension of the state space is known. Otherwise, we consider dimension selection as a Bayesian model selection problem and we also examine the relationship between possible overfitting and the choice of hyperparameters.

Keywords. Software reliability, Hidden Markov model, Bayesian model selection.

1 Introduction

Many papers have been published on software reliability during the last several decades; see Jelinski and Moranda (1972) and Musa and Okumoto (1984) as examples of early work. Bayesian methods have been widely used in this field as discussed in Singpurwalla and Wilson (1999). In this paper we present a new model motivated by potential introduction of new bugs to the software when fixing the current ones. The proposed model, based on a hidden Markov chain, assumes that times between failures are exponentially distributed with parameters depending on an unknown latent state variable which, in turn, evolves as a Markov chain. The model takes into account, implicitly, the possibility that a new bug is added during software re-design after each observed failure time.

Other earlier works on imperfect debugging are those by Gaudoin (1999) and Kremer (1983). More recently, Durand and Gaudoin (2005) considered a hidden Markov model similar to ours (but in a non-Bayesian framework) and used an EM algorithm to obtain maximum likelihood estimates. Finally, Ravishanker et al. (2008) defined a non-homogeneous Poisson process with Markov switching intensity to model sequences of software failure counts.

In Section 2 we describe the hidden Markov model (HMM). In Section 3 we present a numerical example based on Musa’s System 1 data and we address very briefly the problem of selecting the state-space cardinality, using Chib’s (1995) method for approximating the marginal likelihood. Finally, we dwell upon the effect of hyperparameters on such selection.

2 A hidden Markov model for software failures

We assume that, during the testing stages, the failure rate of the software is governed by a latent process \( Y \). Let \( Y_t \) denote the state of the latent process at time \( t \) and, given that \( Y_t = i \), assume that \( X_t \), the failure time for period \( t \), follows an exponential model given by

\[
X_t | Y_t = i \sim \mathcal{E}(\lambda(i))
\]
Conditionally on hidden states, failure times are independent with marginal distribution as above. Finally, we let the latent process be a Markov chain with a transition matrix \( P \) on a finite state space \( E = \{1, \ldots, k\} \). The initial state \( Y_1 \) is given a uniform distribution on \( \{1, \ldots, k\} \).

In the Bayesian setup, the transition matrix \( P \) and the failure rate \( \lambda(i) \), for \( i = 1, \ldots, k \), are all unknown random quantities. The rows of the transition matrix are independent and the \( i \)-th row of \( P \) follows a Dirichlet distribution: \( P_i \sim \text{Dir}(\alpha_{i1}, \ldots, \alpha_{ik}) \), \( i = 1, \ldots, k \). The rates are a priori independent as well and gamma distributed: \( \lambda(i) \sim \mathcal{G}(a(i), b(i)) \).

Having the Bayesian model so specified, it is relatively easy to build a Gibbs sampler with state space \( (\lambda, P, Y^{(n)}) \), where \( \lambda = (\lambda(1), \ldots, \lambda(k)) \) and \( Y^{(n)} = (Y_1, \ldots, Y_n) \). Letting \( x^{(n)} = (x_1, \ldots, x_n) \) denote the observed failure times, we list the needed full conditional distributions below:

\[
\lambda(i)|Y^{(n)}, x^{(n)} \sim \mathcal{G}(a^*(i), b^*(i))
\]

where \( a^*(i) = a(i) + \sum_{t=1}^n 1(Y_t = i) \) and \( b^*(i) = b(i) + \sum_{t=1}^n 1(Y_t = i) x_i \), with \( 1(\cdot) \) denoting the indicator function:

\[
P_i|Y^{(n)}, \lambda(Y_t), x^{(n)}, P \sim \text{Dir}\{\alpha_{ij} + \sum_{t=1}^n 1(Y_t = i, Y_{t+1} = j); j \in E\} ;
\]

\[
\pi(Y_t|Y^{(-t)}, \lambda(Y_t), x^{(n)}, P) \propto P_{Y_{t-1}, Y_t} \lambda(Y_t) e^{-\lambda(Y_t) x_t} P_{Y_t, Y_{t+1}}, \quad Y_t \in E
\]

where \( Y^{(-t)} = \{Y_s; s \neq t\} \).

From the output of the Gibbs sampler we can approximate quantities such as the posterior probability that, for any \( t \), \( Y_t \) is in a state associated with the lowest failure rate or the posterior predictive distribution of \( X_{n+1} \).

3 Analysis of software reliability data

Musa’s System 1 data consists of 136 software failure times. At first, we consider two states for \( Y_t \), and assume uniform distributions for the row vectors \( P_t \) of the transition matrix, and diffuse gamma distributions for the \( \lambda \)'s: \( \lambda(i) \sim \mathcal{G}(0.01, 0.01) \), \( i = 1, 2 \). We run 5000 iterations of the Gibbs sampler and this led to convergence for all the quantities.

A plot of the pairs \( (t, x_t) \), \( t = 1, \ldots, 136 \), shows that the time between failures tend to increase over time implying an overall reliability growth. This is reflected in the posterior distributions of \( \lambda_1 \) and \( \lambda_2 \) (not presented here for lack of space): the posterior distribution of \( \lambda_1 \) is concentrated around lower values than that of \( \lambda_2 \). Thus environment 1 is the more desirable of the two environments. In Figure 1 we present the posterior probabilities \( P(Y_t = 1|x^{(n)}) \) for the “good” environment, that is, for environment 1, for time periods \( t = 1, \ldots, 136 \). As we can see from the figure, the posterior probability is rather low for most of the first 80 testing stages implying that modifications which are made to the software during these stages have not improved the reliability from one period to the next. On the other hand, the posterior probabilities for environment 1 wander around values higher than 0.85 for most of the stages, implying the improvement in the reliability achieved during the later stages. We note that the higher posterior probabilities are associated with longer failure times. If we plotted also the posterior distributions of transition probabilities, we would see that the process \( Y_t \) tends to stay in the same state from one testing stage to the next one.

The analysis with two environments is justified by the values of the marginal likelihood computed with Chib’s method, even though the marginal likelihood is highest with three environments. In fact, a run of the Gibbs sampler with three hidden states produces identical
posterior summaries for the two smallest rates, whereas the third hidden state is included only to capture three null failure times at epochs 33, 61 and 104. Furthermore, the posterior probability that \( Y(t) \) occupies a hidden state associated with the lowest failure rate is identical to that of Figure 1, except for deeper valleys at the epochs of the null failure times.

With the possible exceptions due to very unusual observed failure times, such as the null ones in Musa’s data, further experiments with simulated data showed that diffuse priors on rates induce a faithful estimation of the number of hidden states, both when there is only one state and when states are more than two. If we reduce the prior variance of rates, for example \( \alpha(i) = b(i) = 0.1, i = 1, \ldots, k \), the model tends to favour a larger number of environments, because a single environment can accommodate a less wide range of failure times. Therefore, diffuse priors on rates provide a more parsimonious modelling while still indicating whether the environment is good enough for ending the software testing phase.

![Time Series Plot of Posterior Probabilities of \( Y(t)=1 \)](image)

**Fig. 1.** Posterior probability of \( Y_t = 1 \).

**References**


