A Hierarchical Mixture Model for Software Reliability Prediction

Shaoming Li, Qian Yin and Ping Guo

Department of Computer Science,
Beijing Normal University, Beijing 100875, China
Laboratory of Computer Science, Institute of Software,
The Chinese Academy of Sciences, Beijing 100080, China
pguo@bnu.edu.cn

Michael R. Lyu*

Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong, SAR China
*lyu@cse.cuhk.edu.hk

It is important to develop general prediction models in current software reliability research. In this paper, we propose a hierarchical mixture of software reliability model (HMSRM) for the software reliability prediction. It is an application of the hierarchical mixtures of experts (HME) architecture. In HMSRM, individual software reliability model is used as an expert. During the training of HMSRM, Expectation-Maximizing (EM) algorithm is employed to estimate the parameters of the model. Experiments illustrate that our approach performs better than single classical software reliability models and the result of HMSRM is quite well in reliability prediction. It shows that the method can select the fittest lower-level model for the data automatically and performances are well in prediction.

Keywords: Software Reliability Model, HME, EM algorithm, Prediction.

AMS Subject Classification: 22E46, 53C35, 57S20

1. Introduction

Software reliability is one of a number of aspects of computer software which can be taken into consideration when determining the quality of the software. At the 1997 International Symposium on Software Reliability Engineering, Tierney reported the results of a survey taken in late 1997 that showed that Microsoft has applied software reliability engineering in 50 percent of its software development groups, including projects such as Windows NT and Word. Build good reliability models are one of the key problems in the field of software reliability.

A good software reliability model should give good predictions of future failure behavior, compute useful quantities as well as widely applicable. In current software reliability research, it is very important to develop general prediction models[1].
Existing models typically rely on assumptions about development environments, the nature of software failures, and the probability of individual failures occurrence. Thus each model could be shown to do well with a specific failure data sets, but no model appears to perform well for all cases.

To solve this problem, in this paper a hierarchical mixture of software reliability model (HMSRM) is proposed, which is based on the principle of “divide-and-conquer”. HMSRM can be considered as an application of the hierarchical mixtures of experts (HME) architecture[2]. The model takes the outputs of serval “experts” as inputs and integrates them together effectively by the gating networks. Unlike many other divide-and-conquer algorithms, HME makes use of “soft” splits of data. Thus it can save as much input information as possible to make decisions.

HMSRM is used to take the advantages of several classical software reliability models as “experts”. By this means, we can get more accurate predictions. In order to adjust the “contribution” of each expert, Expectation-Maximization (EM) algorithm[2][5] is employed. The experimental results show that the generalization performance of HMSRM is better than individual classical software reliability models, and it is a good approach to model selection.

2. Classical Software Reliability Models

As the development of software reliability research, many models are built to predict future failures. Some of the models are called Nonhomogeneous Poisson Process (NHPP), because the mean value function $M(t)$ represents the cumulative number of faults exposed by time $t$ [3]. Many of the NHPP models are proved to be effective only at a particular environment in practice. In order to construct HMSRM, we choose four classical NHPP models as the “experts” in the HMSRM. They are listed in the following:

(1) GOEL AND OKUMOTO MODEL (G-O model)

$$M(t) = a(1 - e^{-bt})$$

(2) DUANE MODEL

$$M(t) = at^b$$

(3) S-SHAPED MODEL

$$M(t) = a(1 - (1 + bt)e^{-bt})$$

(4) K-STAGE ERLANGIAN (GAMMA) GROWTH CURVE MODEL (k=3)

$$M(t) = a(1 - (1 + bt + \frac{(bt)^2}{2})e^{-bt})$$

In these NHPP models, usually parameter $a$ represents the mean number of software failures to be detected eventually and parameter $b$ represents the probability of a failure is detected in a constant period.
3. HMSRM and the EM Algorithm

The HMSRM we used here is based on the HME architecture. The failure data are given in pairs as $X = \{(x^{(t)}, d^{(t)})\}$, where $x^{(t)}$ represents the executive time and $d^{(t)}$ represents the failure number. In the training algorithm, we divide the input space into a set of sub regions and assign each “expert” to certain regions. The bounds of the regions are adjusted dynamically by the algorithm.

The architecture of a two-level HMSRM[2][6] is shown in figure 1. It has exactly the same structure as the HME model. Classical reliability models, which are expert networks in the HME model, sit at the leaves of the tree and gating networks control the nonterminals of the tree. Each classical model receives the $x$(executive time) as input and produces a result $\mu_{i,j}$(failure number). In the meanwhile, the gating networks receive the same $x$ and compute the weights of each lower-level model.

In the architecture, lower-level model produces its output $\mu_{i,j}$ as a function of the input $x$:

$$\mu_{i,j} = f_{i,j}(x).$$  \hspace{1cm} (3.1)
The gating networks are considered as generalized linear. Define the intermediate variables:
\[ \lambda_i = v_i^T x, \quad (3.2) \]
where \( v_i \) is a weight vector. The top-level gating network is defined as the following:
\[ g_i = \frac{e^{\lambda_i}}{\sum_k e^{\lambda_k}}. \quad (3.3) \]
Similarly, the lower-level networks are defined as following:
\[ g_{ji} = \frac{e^{\lambda_{ji}}}{\sum_k e^{\lambda_k}}, \quad \lambda_{ij} = v_{ij}^T t. \quad (3.4) \]
Thus, the output of the whole HMSRM is:
\[ y = \sum_i g_i \mu_i, \quad \mu_i = \sum_j g_{ji} \mu_{ij}. \quad (3.5) \]

As \( g_i \) and \( g_{ji} \) sums to one for each \( x \) respectively, they can be considered as “prior” probabilities. Using Bayes’ rules, we can define the posterior probabilities as follows:
\[ h_i = \frac{g_i \sum_j g_{ji} P_{ij}(y)}{\sum_i g_i \sum_j g_{ji} P_{ij}(y)}, \quad h_{ji} = \frac{g_{ji} P_{ij}(y)}{\sum_j g_{ji} P_{ij}(y)}; \quad (3.6) \]
where \( P_{ij} \) is the distribution of the outputs of the classical models. Also, joint posterior probabilities are defined as:
\[ h_{ij} = \frac{g_i g_{ji} P_{ij}(y)}{\sum_i g_i \sum_j g_{ji} P_{ij}(y)}. \quad (3.7) \]

We employ the EM algorithm to solve the learning problem of HMSRM. The EM algorithm is a general technique for maximum likelihood estimation, it is an iterative approach. Each iteration of an EM algorithm has two steps: an Estimation (E) step and a Maximization (M) step. The M step involves the maximization of a likelihood function that is redefined in each iteration by the E step[2].

In order to apply the EM algorithm to HMSRM, we define indicator variables \( z_i \) and \( z_{ji} \), that one and only one of the \( z_i \) is equal to one, and one and only one of the \( z_{ji} \) is equal to one. Thus the expectation of them can be written as follows:
\[ E[z_{ij}^{(t)} | X] = \frac{P(z_{ij}^{(t)} = 1 | y^{(t)}, x^{(t)}, \theta^{(p)})}{P(y^{(t)} | x^{(t)}, \theta^{(p)})} \]
\[ = \frac{P(y^{(t)} | z_{ij}^{(t)} = 1, x^{(t)}, \theta^{(p)}) P(z_{ij}^{(t)} = 1 | x^{(t)}, \theta^{(p)})}{P(y^{(t)} | x^{(t)}, \theta^{(p)})} \]
\[ = h_{ij}^{(t)}, \quad (3.8) \]
where \( \theta \) stands for the parameters of the expert networks. Also that \( E[z_i^{(t)} | X] = h_i^{(t)} \) and \( E[z_{ji}^{(t)} | X] = h_{ji}^{(t)} \).
The complete-data likelihood:

\[ l_{c}(\theta; y) = \sum_{t} \sum_{i} \sum_{j} z_{ij}^{(t)} \ln(g_{i}^{(t)} g_{j|i}^{(t)} P_{ij}(y^{(t)})), \]  

(3.9)

then the expectation of the complete-data likelihood is:

\[ Q(\theta, \hat{\theta}(n)) = \sum_{t} \sum_{i} \sum_{j} E[z_{ij}^{(t)}] (\ln g_{i}^{(t)} + \ln g_{j|i}^{(t)} + \ln P_{ij}(y^{(t)})) \]

\[ = \sum_{t} \sum_{i} \sum_{j} h_{ij}^{(t)} (\ln g_{i}^{(t)} + \ln g_{j|i}^{(t)} + \ln P_{ij}(y^{(t)})). \]  

(3.10)

\[ Q(\theta, \hat{\theta}(n)) = \sum_{t} \sum_{i} \sum_{j} h_{ij}^{(t)} (\ln g_{i}^{(t)} + \ln g_{j|i}^{(t)} + \ln P_{ij}(y^{(t)})). \]  

(3.11)

The M step requires to maximize the \( Q(\theta, \hat{\theta}(n)) \). According to Equation 3.11, we can adjust the lower-level model parameters and the gating network weights respectively. Thus the M step reduces to the following optimization problems:

\[ w_{ij}^{(p+1)} = \arg \max_{w_{ij}} \sum_{t} h_{ij}^{(t)} \ln P_{ij}(y^{(t)}), \]  

(3.12)

\[ v_{i}^{(p+1)} = \arg \max_{v_{i}} \sum_{t} \sum_{k} h_{ik}^{(t)} \ln g_{k}^{(t)}, \]  

(3.13)

\[ v_{ij}^{(p+1)} = \arg \max_{v_{ij}} \sum_{t} \sum_{k} h_{ik}^{(t)} \sum_{l} h_{lj}^{(t)} \ln g_{l|k}, \]  

(3.14)

where \( w_{ij} \) is the parameters of the \((i, j)\) lower-level model. To simplify the problem, we assume the distribution of \( P_{ij} \) is normal.

\[ P_{ij}(y^{(t)}) = k e^{-(d^{(t)} - y_{ij}^{(t)})^2}, \]  

(3.15)

where \( k \) is an constant value. Thus Equation 3.12 reduces to:

\[ w_{ij}^{(p+1)} = \arg \min_{w_{ij}} \sum_{t} h_{ij}^{(t)} (d^{(t)} - y_{ij}^{(t)})^2, \]  

(3.16)

where \( y_{ij} \) is the output of the \((i, j)\) lower-level model and \( d^{(t)} \) is the corresponding result of \( x^{(t)} \).

4. Experiments

The training and test data SYS1 and CSR1 are selected from the CD-ROM of “Handbook of Software Reliability Engineering” [4]. In data SYS1, there are 136 samples that contain time and failure numbers while there are 397 failure numbers and corresponding time-to-failure samples in CSR1. We take early part of the software running data to train the HMSRM.

In the experiments, we construct a two level HMSRM. At the terminal nodes sit the four single models as shown in Figure 1. In this part, two experiments are
Table 1. The prediction result based on 60% data

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Variance (training data)</th>
<th>Variance (test data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOEL AND OKUMOTO MODEL</td>
<td>a=0.8014</td>
<td>7.4913e-004</td>
<td>0.0195</td>
</tr>
<tr>
<td></td>
<td>b=5.8456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUANE MODEL</td>
<td>a=1.1054</td>
<td>7.5958e-004</td>
<td>0.0021</td>
</tr>
<tr>
<td></td>
<td>b=0.4897</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-SHAPED MODEL</td>
<td>a=0.7354</td>
<td>0.0040</td>
<td>0.0374</td>
</tr>
<tr>
<td></td>
<td>b=15.1317</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-STAGE MODEL (k=3)</td>
<td>a=0.7256</td>
<td>0.0070</td>
<td>0.0411</td>
</tr>
<tr>
<td></td>
<td>b=23.0171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMSRM</td>
<td></td>
<td>3.2064e-004</td>
<td>0.0022</td>
</tr>
</tbody>
</table>

Table 2. The prediction result based on 75% data

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Variance (training data)</th>
<th>Variance (test data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOEL AND OKUMOTO MODEL</td>
<td>a=0.8703</td>
<td>0.0014</td>
<td>0.0133</td>
</tr>
<tr>
<td></td>
<td>b=5.0047</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUANE MODEL</td>
<td>a=1.0976</td>
<td>6.8968e-004</td>
<td>0.0045</td>
</tr>
<tr>
<td></td>
<td>b=0.4860</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-SHAPED MODEL</td>
<td>a=0.7911</td>
<td>0.0056</td>
<td>0.0339</td>
</tr>
<tr>
<td></td>
<td>b=13.7296</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-STAGE MODEL (k=3)</td>
<td>a=0.7742</td>
<td>0.0089</td>
<td>0.0403</td>
</tr>
<tr>
<td></td>
<td>b=21.7276</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMSRM</td>
<td></td>
<td>3.6517e-004</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

performed to verify the model using data SYS1. Before the experiment, the data is normalized. In the first experiment, the model is trained with the first 60% of the samples in time axis, while remaining of failure data are used for test. The data is divided into 114 training samples and 22 test samples respectively. In the second experiment, the model is trained with the first 75% of the samples in time axis, which means there are 128 training samples and 8 test samples. The data are used to train the HMSRM with EM algorithm mentioned above. As a comparison, we also train individual NHPP models. In order to simplify the problem, the gradient descent learning algorithm is adopted to estimate the parameters of the classical models. Also, the variance is used to measure the quality of the models. The results are shown in Table 3 and Table 2.
Table 3. The prediction result (data normalized)

<table>
<thead>
<tr>
<th>Model</th>
<th>60% Variance (training data)</th>
<th>60% Variance (test data)</th>
<th>75% Variance (training data)</th>
<th>75% Variance (test data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOEL AND OKUMOTO</td>
<td>7.4913e-004</td>
<td>0.0195</td>
<td>0.0014</td>
<td>0.0133</td>
</tr>
<tr>
<td>DUANE</td>
<td>7.5958e-004</td>
<td>0.0021</td>
<td>6.8968e-004</td>
<td>0.0045</td>
</tr>
<tr>
<td>S-SHAPED</td>
<td>0.0040</td>
<td>0.0374</td>
<td>0.0056</td>
<td>0.0339</td>
</tr>
<tr>
<td>K-STAGE (k=3)</td>
<td>0.0070</td>
<td>0.0411</td>
<td>0.0089</td>
<td>0.0403</td>
</tr>
<tr>
<td>HME</td>
<td>3.2064e-004</td>
<td>0.0022</td>
<td>3.6517e-004</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Fig. 2. The prediction curve of HMSRM in 75% prediction: the solid line stands for the HMSRM prediction result and the dashed line stands for the real failure data.

Figure 2 and Figure 3 show the curves of the HMSRM and individual models in comparison with the real data respectively. The two figures indicate that HMSRM works well in the prediction and the DUANE model is much better than other individual models. From Table 3 and Table 2, we can find that HMSRM is as good as the DUANE model in prediction, or even better than it in some condition. That proves HMSRM is an effective method in failure prediction.

Also, Figure 4 and Figure 5 show “contribution” curves of the final result of the HMSRM in the experiments. They represent the effects of every single models in the final result in various time periods. As DUANE model works relative well for a longer time period, it has the largest contribution. G-O model also works well with specific time period. The other two models are not suitable in this region, therefore they are restrained by the gating networks.
As shown in Equation 3.3 and 3.4, $g_i$ and $g_{ji}$ have exponential function form, models that are not quite suitable are restrained gradually, and one model has the biggest weight as time $t$ increases. Thus the most suitable model has dominating effects in prediction as the DUANE model above. That makes the prediction close to that of the best model. As the best model is selected automatically by the gating networks during the training, HMSRM is a good approach to select proper lower-level models in software reliability. Besides the dominate one, other lower-level software reliability models in the HMSRM may work for part of the data just as the G-O model in the above experiment. It may help the HMSRM fix part of the prediction data, thus the HMSRM may get better results than the best single lower-level model.

Table 3 and Table 2 also show the fact that if we have a larger set of failure data, the accuracy of prediction would be improved. More information from the failures can help the gating networks choose the best lower-level model and decide the most suitable regions for the lower-level models more accurately. More information is also useful to train the lower-level models, consequently, which will lead to better predictions. The result can also be found in the comparison of Figure 4 and Figure 5. In Figure 5, the HMSRM depends upon DUANE model much more and G-O model has a relatively low effect. We can see that leads to a better prediction according to the tables.
HMSRM is applied to solve the problem and the result is shown in Table 4. Though DUANE model is very effective in the SYS1 data experiments and not suitable for current CSR1 samples. From the result, we can see HMSRM is not affected by the irrelevancy model and can still produce a good prediction. That proves our proposed technique is not depended on single models and can select best “contribution curve” for various data sets automatically.

5. Conclusions

In this paper, we apply the HME architecture to mix classical software-reliability models and that forms HMSRM. During the training phase, the EM algorithm is used to ensure the convergence of the training. The experimental results show the HMSRM can select the best lower-level models for the given failure data sets automatically and produce a good prediction.

Another experiment is done to verify the model using data set CSR1. There are 397 samples in the data set. First the data set is converted to the form of time and corresponding failure numbers and normalized. Similarly, we use the earlier 300 samples as the training set and the remain 97 samples to verify the prediction result. The prediction results of individual models in shown in Figure 6. We can see the most serious problem in the individual models is the DUANE model, which is not suitable for this data. It meets some of the early data, but the prediction result has a large disagree from the real data.

Fig. 4. The contribution curves of the lower-level models in the HMSRM in 60% prediction. (a) G-O model (b) DUANE model (c) S-SHAPED model (d) K-STAGE model
Shaoming Li, Qian Yin, Ping Guo and Michael R. Lyu

Fig. 5. The contribution curves of the lower-level models in the HMSRM in 75% prediction. (a) G-O model (b) DUANE model (c) S-SHAPED model (d) K-STAGE model

Table 4. The prediction result for CSR1 based on the first 300 samples

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Variance (training data)</th>
<th>Variance (test data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOEL AND OKUMOTO MODEL</td>
<td>a=0.8079</td>
<td>0.0011</td>
<td>0.0101</td>
</tr>
<tr>
<td></td>
<td>b=9.9644</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUANE MODEL</td>
<td>a=2.8549</td>
<td>0.0127</td>
<td>1.0798</td>
</tr>
<tr>
<td></td>
<td>b=0.8165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-SHAPED MODEL</td>
<td>a=0.7478</td>
<td>7.6480e-004</td>
<td>0.0222</td>
</tr>
<tr>
<td></td>
<td>b=25.7710</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-STAGE MODEL (k=3)</td>
<td>a=0.7545</td>
<td>0.0031</td>
<td>0.0205</td>
</tr>
<tr>
<td></td>
<td>b=38.3136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMSRM</td>
<td></td>
<td>7.9633e-004</td>
<td>0.0098</td>
</tr>
</tbody>
</table>

As we known, the prediction quality is based on the lower-level models, HMSRM can be built with a lot of successful reliability models to achieve a good performance since HMSRM has an expandable architecture. Also it can restrain the irrelevancy models in the architecture. From the above discussion, we conclude that HMSRM is a promising method in reliability prediction for the case of later software devel-
Fig. 6. The prediction curves of individual models in CSR1 prediction: the solid lines stand for the prediction result and the dashed lines stand for the real failure data. (a) G-O model (b) DUANE model (c) S-SHAPED model (d) K-STAGE model

Acknowledgements

This work was fully supported by a grant from the National Natural Science Foundation of China (Project No. 60275002) and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHAAAA/xxxx).

References

6. Yin Qian, Li Shaoming, Guo Ping, “A Case Study on the Hierarchical Software Reliabili-
12 Shaoming Li, Qian Yin, Ping Guo and Michael R. Lyu
