Empirical Bayes Estimation of Software Failures

Néstor Ruben Barraza *

Universidad Nacional de Tres de Febrero
nbarraza@untref.edu.ar

Abstract. The empirical Bayes estimator is applied to software failures production. The time between failures data registered up to a given time, are used in order to estimate the probability of failure appearance during the next interval time. This method is similar to the estimation of n-grams in natural language processing. A modified expression to the estimator usually used in language and speech processing is introduced in order to follow the failures production curve. Results of simulations comparing well with experimental data are also shown.

Keywords: Software reliability, empirical Bayes, growth model

1 Introduction

The empirical Bayes estimator is obtained from the Bayes rule applied to the conditional expectation of the unknown parameter given the samples. The conditional expectation minimizes the mean square error function as it is well known. Several applications can be found in different areas of Engineering, specially in Speech recognition and word processing, [10], [13], as well as in Reliability, [7], [8]. Many simple forms of this estimator were proposed decades ago. One of them was the Good-Turing estimator introduced as an alternative to the maximum likelihood estimator in order to avoid assigning zero probability to samples that never occurred, [6]. Improvements on the Good-Turing estimator were proposed in the literature of speech processing, [4]. A general form of the empirical Bayes estimator using mixed distributions was proposed in a previous work [2]. The interest in speech or word processing is to estimate the probability of the appearance of a n-gram given it has appeared r times. We proposed to apply this concept in software reliability considering software failures instead of n-grams. By time discretization, we can ask about the probability of a software failure arrival during the next interval time, conditioned to r failures has occurred in the past. It is well known that a nonlinearity is expected in this probability since the probability of failure decreases as testing time progresses. In order to follow this behavior, several software reliability models were proposed in the literature, the so called software reliability growth models. Non homogeneous Poisson and Compound Poisson models were proposed decades ago as software reliability growth models.

* The author is also with the School of Engineering. University of Buenos Aires.

The mentioned alternatives introduced in the Empirical Bayes estimator used in word processing have a linear dependence on r. Since a nonlinearity dependence on r is required for software failures in order to follow the software failure cumulative curve, we used the modified expression introduced in [2].

The main motivation of this work is to take advantage of the nonlinearity property of the modified expression of the Empirical Bayes estimator, in order to predict the probability of failure in the next interval time, provided this probability decreases. Modeling this way the reliability growth. This prediction is based on previously reported data of failures of the same project. We will show that having just few reports of failures is enough to perform a good prediction of future failures during the whole phase either testing or operations. This is the main advantage of the proposed method.

This paper is organized as follows: The empirical Bayes estimator as it is usually used to predict the probability of the occurrence of an n-gram is presented in section (2), the modified expression of the Empirical Bayes estimator with a nonlinearity characteristic is shown in section (3), the theoretical foundations that supports our main motivation are presented in section (4), a simulation of the proposed model and a comparison with experimental data is shown in section (5), conclusions are presented in section (6).

2 The Empirical Bayes estimator

The empirical Bayes estimator $\hat{\theta}$ of the probability θ of the occurrence of a single event after r outcomes in n experiments were obtained can be defined as:

$$\hat{\theta} = E[p|r, n] \; . \tag{1}$$

There are well known forms of (1) like the Good-Turing estimate:

$$\hat{\theta} = \frac{r+1}{n} \frac{N_r}{N_{r+1}} \tag{2}$$

where N_r and N_{r+1} are the number of events that occurred r and $r+1$ times, or the Laplace law of succession:

$$\hat{\theta} = \frac{r+1}{n+2} \; . \tag{3}$$

Other simple expressions have been proposed for applications in word processing were calculus must be performed rapidly. Some of these proposals are the Knesser Ney discount estimates [11]:

$$\begin{aligned} \hat{\theta} &= (1-k)\,\frac{r+b}{n} && \text{Linear discount} \\ &= \frac{r-b}{n} && \text{Absolute discount} \end{aligned}$$

and the generalized law of successions [12]:

$$\hat{\theta} = \frac{r+b}{n+s\,b} \tag{4}$$

where k, b and s are constants. All of these forms of the estimates depend linearly on r, in order to get other dependencies on r, a generalized expression is introduced in the next section.

3 A modified expression of the Empirical Bayes estimator using mixed distributions

A general expression of the Empirical Bayes estimator can be introduced from a mixed Poisson distribution:

$$P(r/\lambda) = \frac{\lambda^r}{r!}\exp^{-\lambda}\ . \tag{5}$$

Since we are interested in a binary probability, i.e. a probability of success or fault, then, the approximation of the binomial by the Poisson

$$\hat{\theta} = \frac{\hat{\lambda}}{n} \tag{6}$$

valid for large values of n must be taken into account. Being $S(\lambda)$ the probability density function of λ and applying the Bayes rule, the λ estimate results, (see [2] for details):

$$\hat{\theta} = \frac{1}{n}\frac{\int \lambda \frac{\lambda^r}{r!}\exp^{-\lambda}\,dS(\lambda)}{\int \frac{\lambda^r}{r!}\exp^{-\lambda}\,dS(\lambda)}\ . \tag{7}$$

Despite the last approximation is valid for large values of n and small values of r, the previous expression can be considered as a general definition of the estimator. The last equation can be written as:

$$\hat{\theta} = \frac{r+1}{n}\frac{P(r+1)}{P(r)}\ . \tag{8}$$

From (8), the introduction of the smoothing nonlinear factor $\frac{P(r+1)}{P(r)}$ must be noted. The effect and necessity of the smoothing factor in software reliability is analyzed in the next section.

4 Software failures prediction using the Empirical Bayes Estimator

The cumulative number of software failures collected during the testing phase of a software product has several characteristics. We expect an increasing number

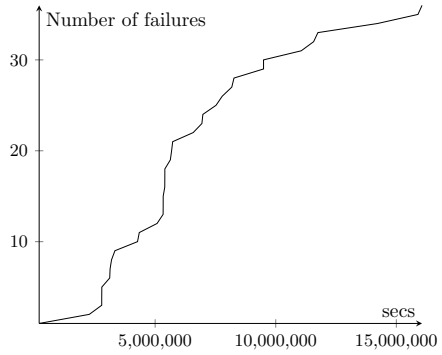

Fig. 1. Typical cumulative number of failures curve.

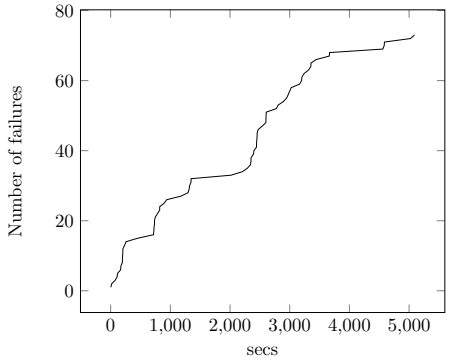

Fig. 2. Cumulative number of failures curve with an inflexion point.

of failures per time unit at first, and a low rate of failures at the end, after some corrections are introduced. A typical software failures cumulative forms revealing the reliability growth taken from [9] are shown in Fig. 1 and Fig. 2.

The curve shown in Fig. 2 presents an inflexion point due perhaps to new code addition. Software reliability models based on the non homogeneous Poisson process were proposed in the 1970s, see a revision in [1]. There were also proposed other models like that based on a Compound Poisson distribution, see [3] for a summary. All of these models try to follow the cumulative number of failures curve by estimating the number of remaining failures from the number of failures previously reported. A different approach is proposed in this work, instead of predicting the total number of remaining failures or the time to next failure, we propose to estimate the probability of a failure arrival during the next interval time.

In order to apply the Empirical Bayes estimator to software failures, we propose to estimate the probability of a failure arrival during the next interval time given r failures have appeared before. In this approach, no more than one failure per interval time is allowed. Since time are usually recorded in seconds, this requirement is accomplished. The smoothing factor introduced in (8) plays an important role in software failures since we can model the cumulative number of failures curve having $P(r+1) > P(r)$ at first and $P(r+1) < P(r)$ at the end. This behavior can be achieved by choosing properly the mixing probability density function $S(\lambda)$. A similar approaches were presented in [7] and [8], where the empirical Bayes is applied to estimate the mean time to failure modeled by a mixed exponential distribution. The main difference between those approaches and our proposal is given by the expression (6), i.e., we propose to estimate the binary probability of failure occurrences. According to the Empirical Bayes estimation assumptions, failures, like n-grams in speech processing or time between failures in [7] and [8], are considered to be statistically independent. Two simulations of the cumulative number of failures are presented in the next section.

5 Simulation

In order to test the goodness of fit of the proposed model we present two simulations based on real data for comparison. Failures data based on execution time were collected by Prof. John D. Musa from several projects, they are available in [9]. Despite these data were registered decades ago when the interest in Software Reliability began, they are still relevant in order to see the software failures behavior and test models since they show the main characteristic in software reliability, i.e. the reliability growth. Because of this, these datasets are still used in recent publications. New software failures data collections and adaptation of the SR models to modern software development methodologies are needed in order to improve software quality metrics, see for example the analysis presented in [5].

Instead of performing a progressive test and estimation as time progresses, in this first approach of our proposal we proceed to simulate failures production

from a starting point obtained experimentally. Thus, taking the arrival time of the first two failures from the corresponding data set, we estimate the probability θ of failure arrival during the next interval time. From eq. (8) we get for the starting point:

$$\hat{\theta}_0 = \frac{r_0 + 1}{n_0} \frac{P(r_0 + 1)}{P(r_0)} \,, \tag{9}$$

where $r_0 = 2$ and n_0 is the time up to the first two failures arrival. From this starting point, the simulation progresses generating a failure with probability θ in the next interval time, if a failure is produced, the value of r is increased by one, else, keeps its value. Whatever the result of the failure generation is, the value of n is always increased by a new interval time. A pseudo code of the simulation procedure is shown next.

Simulation procedure

```
SET the initial count of seconds and failures

SET r equal to the initial number of failures

FOR each second n in the total time interval

    SET p = (r + 1)/n P(r+1)/P(r)

    GENERATE a failure with probability p

    IF a failure arrived THEN

          INCREMENT r

    ENDIF

    STORE number of arrived failures r and seconds

ENDFOR
```

(Failure generation code)

As the mixing distribution, the Generalized Inverse Gaussian probability function used in insurance and queuing models, given by:

$$S(\lambda) = \frac{\mu^{-\alpha} \ \lambda^{\alpha-1} \ \exp -(\lambda^2 + \mu^2)/2\beta\lambda}{2K_\alpha(\mu \ \beta^{-1})} \,, \tag{10}$$

was arbitrarily chosen. Where μ, α and β are parameters, and K_α is the Modified Bessel Function of the third Kind, see [14].

Results of simulations are shown in fig. (5) and (6) for projects 14C and SS1A from the mentioned dataset. Both data were taken during the operations phase.

Data 14C corresponds to a real time system of hundred of thousands of instructions. Data SS1A corresponds to an operating systems of hundred of thousands of instructions. We have chosen these data since they show the software reliability growth characteristic and the 14C project shows also an inflexion point. Software reliability growth with an inflexion point is usually modeled with the S-shaped stochastic model. The parameters of the Inverse Gaussian mixing distribution were also arbitrarily chosen as those which fit better the actual data. Parameter values and the resultant form of the mixing distribution are shown in Fig. 3) and Fig. 4).

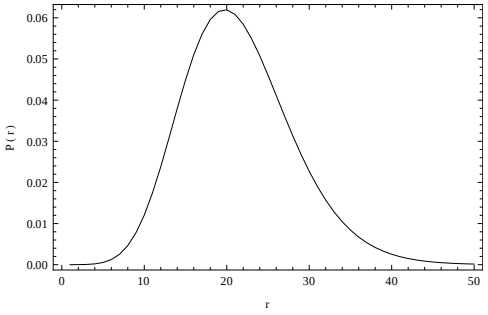

Fig. 3. Generalized Inverse Gaussian for the SS1A project, $\beta = 1$, $\mu = 20$, $\alpha = 1$.

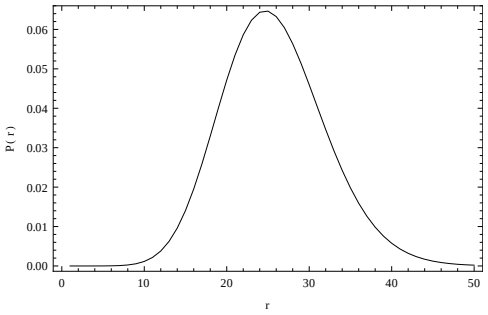

Fig. 4. Generalized Inverse Gaussian for the SS1A project, $\beta = 0.5$, $\mu = 25$, $\alpha = 1$.

From Fig. 5 we can see that the real data curve has an inflexion point at secs 5000000. It is seen that the simulated curve follows the real data up to the inflexion point, then, separates a little and adjusts better at the end.

From Fig. 6 it is seen that the curve fits well the real data all the time and follows the curvature.

Since the only estimation point is that at the first two failures, we can conclude that failures production can be modeled well enough by this procedure.

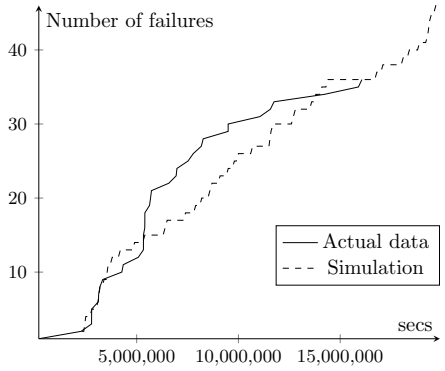

Fig. 5. Actual and simulated cumulative failures curve for the 14C project.

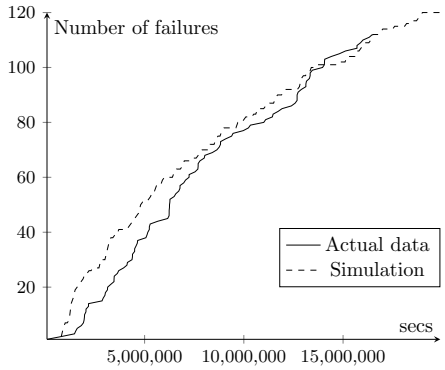

Fig. 6. Actual and simulated cumulative failures curve for the SS1A project.

It could indicate that data at the start of the project together with the mean value of the mixing distribution $S(\lambda)$ in (7) have a lot of information in order to predict the future behavior. All simulations have been performed using Mathematica v.8. Despite of the complex form of the Inverse Gaussian distribution, the simulation gives results in few seconds. Then, we can expect that for any mixing distribution this procedure run fast enough.

6 Conclusion

A new method in order to estimate software failures production has been proposed. This method is based on The Empirical Bayes estimator usually used in other areas of Engineering. A specially smoothing nonlinear factor was introduced in the estimate. This factor is based on a mixed Poisson distribution. A simulation of the cumulative failures curve using a Poisson mixed by a Generalized Inverse Gaussian probability density function has been performed with good agreement with reported data. Some criteria of choosing the mixing distribution and methods to estimate its parameters will be reported in a future work.

Acknowledgments. The author wants to thank Universidad Nacional de Tres de Febrero for support. The author would also like to thank the anonymous reviewers for their valuable comments that helped to improve this paper.

References

1. Almering, V., van Genuchten, M., Cloudt, G., Sonnemans, P.J.: Using software reliability growth models in practice. IEEE Software 24(6), 82–88 (2007)
2. Barraza, N.R.: The empirical bayes estimator and mixed distributions. AIP Conference Proceedings 1073(1) (2008)
3. Barraza, N.R.: Parameter estimation for the compound poisson software reliability model. International Journal of Software Engineering and its Applications 7(1), 137–148 (2013)
4. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. In: Joshi, A., Palmer, M. (eds.) Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics. pp. 310–318. Morgan Kaufmann Publishers, San Francisco (1996)
5. Far, B.: Software reliability engineering for agile software development. In: Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on. pp. 694–697 (April 2007)
6. Good, I.J.: The population frequencies of species and the estimation of population parameters. Biometrika 40(3-4), 237–264 (1953), `http://biomet.oxfordjournals.org/content/40/3-4/237.abstract`
7. Lynn, K., Tae, Y.Y.: Bayesian computation for nonhomogeneous poisson processes in software reliability. Journal of The American Statistical Association 91(434), 763–773 (1996)
8. Mazzuchi, T.A., R., S.: A Bayes Empirical-Bayes Model for Software Reliability. IEEE Trans. on Reliability 37, 248–254 (Jun 1988)

9. Musa, J.D.: Cyber security & information systems. information analysis center. https://sw.thecsiac.com/databases/sled/swrel.php (1970)
10. Nadas, A.: On Turing's formula for word probabilities. IEEE Trans. Acoust. Speech and Signal Processing 33, 1414–1416 (Dec 1985)
11. Ney, H., Essen, U., Kneser, R.: On the estimation of 'small' probabilities by leaving-one-out. IEEE Trans. Pattern Anal. Mach. Intell. 17(12), 1202–1212 (1995)
12. Ristad, E.S.: A natural law of succession. Tech. Rep. TR-495-95, Princeton University (1995)
13. Vaithyanathan, S., Mao, J., Dom, B.: Hierarchical bayes for text classification. In: PRICAI Workshop on Text and Web Mining. pp. 36–43 (2000)
14. Willmot, G.: Mixed Compound Poisson Distributions. Astin Bulletin 16, 59–79 (1986)