Count Models for Software Quality Estimation

Kehan Gao
Eastern Connecticut State University, USA

Taghi M. Khoshgoftaar
Florida Atlantic University, USA

INTRODUCTION

Timely and accurate prediction of the quality of software modules in the early stages of the software development life cycle is very important in the field of software reliability engineering. With such predictions, a software quality assurance team can assign the limited quality improvement resources to the needed areas and prevent problems from occurring during system operation. Software metrics-based quality estimation models are tools that can achieve such predictions. They are generally of two types: a classification model that predicts the class membership of modules into two or more quality-based classes (Khoshgoftaar et al., 2005b), and a quantitative prediction model that estimates the number of faults (or some other quality factor) that are likely to occur in software modules (Ohlsson et al., 1998).

In recent years, a variety of techniques have been developed for software quality estimation (Briand et al., 2002; Khoshgoftaar et al., 2002; Ohlsson et al., 1998; Ping et al., 2002), most of which are suited for either prediction or classification, but not for both. For example, logistic regression (Khoshgoftaar & Allen, 1999) can only be used for classification, whereas multiple linear regression (Ohlsson et al., 1998) can only be used for prediction. Some software quality estimation techniques, such as case-based reasoning (Khoshgoftaar & Seliya, 2003), can be used to calibrate both prediction and classification models, however, they require distinct modeling approaches for both types of models. In contrast to such software quality estimation methods, count models such as the Poisson regression model (PRM) and the zero-inflated Poisson (ZIP) regression model (Khoshgoftaar et al., 2001) can be applied to yield both with just one modeling approach. Moreover, count models are capable of providing the probability that a module has a given number of faults. Despite the attractiveness of calibrating software quality estimation models with count modeling techniques, we feel that their application in software reliability engineering has been very limited (Khoshgoftaar et al., 2001). This study can be used as a basis for assessing the usefulness of count models for predicting the number of faults and quality-based class of software modules.

BACKGROUND

Software Metrics and Software Quality Modeling

Software product and process metrics are essential in the software development process. With metrics, the software development team is able to evaluate, understand, monitor and control a software product or its development process from original specifications all the way up to implementation and customer usage.

In the software reliability engineering literature, the relationship between software complexity metrics and the occurrence of faults in program modules has been used by various metrics-based software quality estimation models, such as case-based reasoning (Khoshgoftaar & Seliya, 2003), regression trees (Khoshgoftaar et al., 2002), fuzzy logic (Xu et al., 2000), genetic programming (Liu & Khoshgoftaar, 2001) and multiple linear regression (Ohlsson et al., 1998). Typically, a software quality model for a given software system is calibrated using the software metrics and fault data collected from a previous system release or similar project. The trained model can then be applied to predict the software quality of a currently under-development release or comparable project. Subsequently, the resources allocated for software quality improvement initiatives can then be targeted toward program modules that are of low quality or are likely to have many faults.

Count Modeling Techniques

Count models have many applications in economics, medical and health care, and social science (Cameron & Windmeijer, 1996; Deb & Trivedi, 1997; Mullahy, 1997). Count models refer to those models where the values of the dependent variable are count numbers, i.e., zeros or positive integers. The typical count models include Poisson regression models (Khoshgoftaar et al., 2005a), negative binomial regression models (Cameron & Trivedi, 1998), hurdle models (Gurmu, 1997) and zero-inflated models (Lambert, 1992).

The Poisson regression method is the basis in count modeling approach. It requires *equidispersion*, i.e., equality of mean and variance of the dependent variable. However in real life systems, the variance of the dependent variable often exceeds its mean value. This phenomenon is known as *overdispersion*. In software quality estimation models, overdispersion is often displayed by an excess of zeros for the dependent variable, such as number of faults. In such cases, a pure PRM fails to make an accurate quality prediction. In order to overcome this limitation of the pure PRM, a few but limited numbers of modifications or alternatives to the pure PRM have been investigated.

Mullahy (1986) used a with-zeros (WZ) model instead of a standard PRM. In his study, it was assumed that the excess of zeros resulted from a mixture of two processes, both of which produced zeros. One process generated a binary outcome, for example: perfect or non-perfect, while the other process generated count quantities which might follow some standard distribution such as, Poisson or negative binomial. Consequently, the mean structure of the pure Poisson process was changed to the weighted combination of the means from each process.

Lambert (1992) used a similar idea to Mullahy's when introducing the zero-inflated Poisson (ZIP) model. An improvement of ZIP over WZ is that ZIP establishes a logit relationship between the "probability that a module is perfect" and the "independent variables of the data set". This relationship ensures that the probability distribution is between 0 and 1.

Similar to the zero-inflated model, another variation of count models is the hurdle model (Gurmu, 1997) which also consists of two parts. However, instead of having perfect and non-perfect groups of modules, the hurdle model divides them into a lower count group and a higher count group, based on a binary distribution. The dependent variable of each of these groups is assumed to follow a separate distribution process. Therefore, two factors may affect predictive quality of the hurdle models: the crossing or threshold value of the dependent variable that is used to form the two groups, and the specific distribution each group is assumed to follow.

Preliminary studies (Khoshgoftaar et al., 2001; Khoshgoftaar et al., 2005a) performed by our research team examined the ZIP regression model for software quality estimation. In the studies, we investigated software metrics and fault data collected for all the program modules from a commercial software system. It was found that over two thirds of the program modules had no faults. Therefore, we considered the ZIP regression method to develop the software quality estimation model.

MAIN FOCUS

Poisson Regression Model

The Poisson regression model is derived from the Poisson distribution by allowing the expected value of the dependent variable to be a function associated with the independent variables. Let $(y_i, \mathbf{x}_i)$ be an observation in a data set, such that y_i and $\mathbf{x}_i$ are the dependent variable and vector of independent variables for the i^{th} observation. Given $\mathbf{x}_i$, assume y_i is Poisson distributed with the probability mass function (*pmf*) of,

$$\Pr(y_i \mid \mu_i, \mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} \quad \text{for } y_i = 0, 1, 2, \cdots, \tag{1}$$

where μ_i is the mean value of the dependent variable y_i. The expected value (E) and the variance (Var) of y_i are identical, i.e., $E(y_i|\mathbf{x}_i) = Var(y_i|\mathbf{x}_i) = \mu_i$.

To ensure that the expected value of y_i is nonnegative, the link function, which displays a relationship between the expected value and the independent variables, should have the following form (Cameron & Trivedi, 1998):

$$\mu_i = E(y_i|\mathbf{x}_i) = e^{\mathbf{x}_i'\boldsymbol{\beta}} \tag{2}$$

where $\boldsymbol{\beta}$ is an unknown parameter vector and $\mathbf{x}_i'$ represents the transpose of the vector $\mathbf{x}_i$.