An Approach to Improving Parametric Estimation Models in case of Violation of Assumptions

Author: Salvatore Alessandro Sarcia[1,3]

Advisors: Victor R. Basili[1,2] Giovanni Cantone[3]

[1]Dept. of Computer Science, University of Maryland, A.V. Williams Bldg. 115, College Park 20742, MD, USA
[2]Fraunhofer Center for Experimental Software Engineering Maryland, College Park, Maryland, 20742
{basili, sarcia}@cs.umd.edu
[3]DISP, Università di Roma Tor Vergata, via del Politecnico 1, 00133 Rome, Italy
sarcia@disp.uniroma2.it, cantone@uniroma2.it

Abstract— parametric models such as least squares regression functions are the most applied methods for prediction and inference in Software Engineering. They have been used for predicting effort, size, fault proneness, defects, and many other variables as to software processes and products. Although the high popularity of those parametric models, sometimes they fall short of providing good results because of violations of the assumptions on which they are built. In this work, we present an approach to improving parametric estimation models when regression assumptions on which the model is built are violated (i.e. errors are not independent, the model is not linear, the sample is heteroscedastic, and the error probability distribution is not Gaussian). We show that, if we violate the regression assumptions and do not deal with the consequences of these violations, we cannot improve the estimation model. Since models can be used for prediction and inference, our proposal refers to both those aspects. The approach provides a way of improving parametric models without making any specific assumption. Since errors can never be removed entirely, we show a risk mitigation strategy that includes uncertainty considerations into the estimation process. The proposed approach can be completely automated and implemented as a support tool for organizations that aims at improving their estimation process over time. Since the proposed approach does not care about the parametric model kind and data sets, to validate the methodology, we calibrated log-linear regression functions by using a well-known data set (COCOMO NASA), even though it is no longer in use. We prove the statement that, violating assumptions and pretending that no consequence affects the estimation model performance is not a suitable way of dealing with improvement issues of mature organizations.

Index Terms—Multi-layer feed-forward neural networks, non-linear regression, Bayesian learning, prediction intervals for neural networks, risk analysis and management, learning organizations, COCOMO

1. Introduction

This research refers to support learning organizations [1] in achieving their business goals and gaining competitive advantage. These organizations need to manage projects effectively and deliver products on time, on budget, and with all functions and features as required. To this end, one of the most important keystones for their success is to be able to estimate correctly the variables of interest of the project, task, and module (e.g., effort, fault proneness, and defect slippage). For instance, a software organization may need to quantify the cost of developing a software system in order to bid on the contract. So, the success (winning the contract or delivering the sub-system as required) would depend on the capability to get the most accurate software cost estimate. Consequently, getting accurate estimates is a strategic goal for these organizations.

Estimation accuracy is not only about yielding estimates as closer to the actual value as possible, but also estimating the estimate variability. In this work, we refer to the improvement issue in a twofold way, (1) improving correctness of estimates (i.e. shrinking the prediction error) and (2) improving the way of calculating the spread of the prediction error (i.e. improving the inference about the predicted variable). The latter is also addressed as estimating the estimation model (EM) uncertainty (e.g., quantifying the risk) or evaluating prediction intervals (PIs) of estimates (e.g., what is the variability of the next predicted value?).

Kitchenham et al. [10] refer to several sources of error, i.e. errors found in the measurements (Measurement error), produced by unsuitable mathematical models (Model error), wrongly assumed input values (Assumption error), violations of regression assumptions (Violation error), or inadequacy of the projects chosen for building such estimates (Scope error). Errors can be represented by (stochastic) variables that we can study and even try to predict, but we cannot avoid. For this reason, in dealing with these issues, we have realized that improving estimation models over time is not enough for supporting those software organizations. They also need to measure the impact of the error on the stated software goals when using the estimation model in their own environment. In other words, software organizations need to both improve their estimation models over time and analyze the risk for planning suitable mitigation strategies.

Researchers and practitioners prefer violating assumptions (e.g. homoscedasticity, model linearity, and normality of the distributions) and ignoring error sources, rather than dealing with the consequences of such violations and errors. Keeping on violating assumptions and ignoring errors pretending that everything is fine "for the sake of simplicity" is not a good way of managing learning organizations and improving estimation models. Conversely, as we propose in this work, investigating empirically consequences of those violations can improve both estimation and inference of parametric models. For this reasons, even though we refer to parametric estimation models, the proposed improvement strategy is mainly based on non-linear models, and non-parametric statistics.

Organizations should be able to integrate the estimation improvement into their general improvement process (e.g., QIP [1]). By contrary, over the last three decades, scientists and practitioners have been trying to support software organizations in finding the best estimation model instead of trying to improve those models that organizations have been using. The result of this huge effort is that currently software engineering practitioners neither have the best estimation model nor appropriate improvement techniques for those models. In other words, as argued in [14] and [12], this thirty-year research effort has been practically disappointing.

We have organized the work in some parts. First, we have presented linear and non-linear estimation models, known improvement strategies, and techniques for evaluating PIs (i.e. uncertainty and risk) for both linear and non-linear models. Secondly, we have defined an error taxonomy showing errors that we really need to worry about and their consequences. Subsequently, we have presented the problem, i.e. we have answered the question why currently used parametric estimation models fall short of providing valid and reliable results when assumptions are violated. We have proceeded with defining the mathematical solution and its application to software engineering. Finally, we have tried out the proposed methodology in a real case (i.e., log-linear regression functions calibrated by using the COCOMO NASA data set).

2. PREDICTION AND INFERENCE OF ESTIMATION MODELS

Mathematically, an estimation model is a regression function f_R such that $y = f_R(x,\beta) + \varepsilon$, where x represents a set of independent variables, y is the dependent variable and β is a set of parameters defining f_R. The component ε is the aleatory part of the model standing for our uncertainty on the relationship between independent and dependent variables. Function f_R cannot be calculated usually, i.e. we cannot calculate parameters β. This is because we cannot know every point of the population. We can estimate β finding a set of estimators b such that they minimize an error function, e.g. least squares (LS). To estimate b, we consider the relationship $O_{act} = f_R(I_{act},b)$, where O_{act} represent the actual values of y, I_{act} are the actual values of x (note that x is a vector of variables hence a matrix of values), and b is the vector of parameters being estimated. For instance, using the COCOMO variables, the estimation model would be, Effort = f_R({KSLOC, 15 COCOMO multipliers}, b). If we were interested in predicting the expected number of defects (y) according to a set of variables x (e.g., size, complexity, and programming language), function f_R would be the parametric model providing the sought estimate. Since b is different from β, f_R provides $O_{est} = f_R(I_{act},b)$, not O_{act}. Then, the difference $e = O_{act} - O_{est}$ is a vector of errors representing ε (called residuals, with $e \neq \varepsilon$ and ε unknown). The most important part in modeling is to find the best estimates for β. For calculating b, there exist some strategies, which are based on selecting b in such a way as the function best fits the observations. If the error function is composed of a linear combination of the sought parameters, the EM is linear in the parameters, and we have a closed solution. If the equation system is composed of non-linear equations of the sought parameters, the EM is non-linear in the parameters and the solution can be found iteratively.

A linear-in-the-parameter function is the following $y = \beta_0 + \beta_1 x_1 + \ldots + \beta_Q x_Q$ (or any polynomial). Non-linear-in-the-parameter models look like the following function $y = \beta_0 + \beta_1 \mathrm{tgh}(w_{01} + w_{11}x_1 + \ldots + w_{Q1}x_Q) + \beta_2 \mathrm{tgh}(w_{02} + w_{12}x_1 + \ldots + w_{Q2}x_Q)$, where tgh(.) is the hyperbolic tangent function. Therefore, a non-linear-in-the-parameter model is composed of functions having adjustable parameters (i.e., parameters w), while a linear-in-the-parameter model is composed of fixed functions (i.e. variables x having only parameters β).

Although we cannot make sure that the "true" regression function is linear, when the size of the sample N goes to infinity, traditional least squares estimates applied to linear-in-the-parameter models provide the "true" parameters β of the regression function (i.e., b = β). Conversely, non-linear-in-the-parameter models cannot provide the true parameters of the regression function because there is no closed solution to the least squares problem for such models (i.e., there is an iterative solution). However, when N is a finite number (as usual in software engineering and in many other fields), linear-in-the-parameter models are not able to provide the true value of the regression function as happening for non-linear-in-the-parameter models. Then, since a non-linear-in-the-parameter model can be made indefinitely flexible for a fixed number of input variables and a linear one does not, the former is more parsimonious and flexible than the latter. Operatively we cannot consider models having an infinitive number of variables, but we can increase the number of parameters of non-linear-in-the-parameter models. Fixing the number of variables and increasing the number of parameters of the model is the essence of the parsimony [2]. Therefore, because of the increased parsimony of non-liner models, in cases where we do not have an infinitive number of variables and observations (i.e., in real cases), non-linear-in-the-parameter models can provide better estimates than the linear ones. We mainly refer to Multi-layer Feed-forward Neural Networks (MFNNs) trained with Backpropagation [3], which are non-linear-in-the-parameter models having indefinite flexibility. MFNNs are called arbitrary approximators (or universal approximators) not because they provide arbitrary outcomes. The "arbitrary" aspect is only about their unlimited flexibility as explained above.

The reasons why parameters of a regression function may be biased, its estimates inaccurate, and the inference drawn from it incorrect are the following:

(a) *Variable model error (VrblME)*: The model is missing some relevant variables; hence, it is not able to explain the output.

(b) *Redundancy model error (RdndME)*: The model includes too many variables that negatively affect the correctness of the model parameters

(c) *Complexity model error (CplxME)*: The model is not enough flexible to represent the relationship between inputs and output. For instance, this error can happen when we choose models that are linear in the parameters, e.g. ordinary least squares (OLS)

(d) *Violation model error (VltnME)*: If we want to get the best estimators of β (Gauss-Markov theorem [11]) and use the model for inference (i.e., estimating the variability of the independent variable), LS requires some assumptions. If we violate assumptions, the model parameters may be biased and inference incorrect. We call these assumptions as "regression assumptions". They are:

    (1) Errors ε are not x correlated

    (2) The variance of the errors is constant (homoscedasticity), $\text{cov}(\varepsilon) = \sigma^2 I$

    (3) Errors ε are not autocorrelated (not worry for software engineering data)

    (e) The probability density of the error is a Gaussian, $\varepsilon \sim NID(0, \sigma^2 I)$, i.e. there are no outliers, skewed/kurtotic distributions, and measurement error.

Note that, *Assumption error* and *Scope error* do not affect the correctness of the estimation model because the former is about wrongly assumed values of the project being estimated and the latter is about the unsuitability of the model in estimating new projects because observations used for calibrating the model are different from projects being estimated.

To improve parametric estimation models, improvements can be done for each bullet of the list above. To deal with *VrblME*, there is no way actually. We must find the right variables of the model. If we do not find those variables, the model will not be correct. To deal with *RdndME*, the most applied technique is stepwise regression, which can be forward or backward [11]. It requires knowing the relevance of variable being removed. If we do not know the least relevant variables, we have to consider all the possible models. For instance, if we have Q variables, there will be 2^Q different models. However, this procedure is usually too expensive to be executed in real cases. Stepwise regression is based on the assumption that the model does not suffer of multicollinearity [11]. If we would like to avoid this assumption, we can apply techniques of feature reduction such as Principal component analysis (PCA) [*7*] or Curvilinear component analysis (CCA) [*3, pp. 310-319*]. Since CCA is able to perform PCA as well, we will focus on CCA. CCA is a technique that is able to shrink the initial input variable set into an equivalent set having a fewer variables called curvilinear components [13]. CCA removes any linear and non-linear redundancy but it turns the data set into an equivalent one where the variables have no correspondence with the initial ones. Since CCA does no need any assumption, we choose it as a feature reduction technique. To deal with *CplxME*, the procedure is to (1) consider different families of function, (2) compare them to each other, and (3) select the best, i.e., the one yielding the least generalization error. Vapnik proves that leave-one-out cross-validation (LOOCV) provides an unbiased estimate of the generalization error [16]. Therefore, if we apply LOOCV and CCA to non-liner models we can find a model both flexible and parsimonious without making any specific assumption.

TABLE 1

IMPROVING INFERENCE OF LINEAR AND NON-LINEAR MODELS

Violation	Improvement	
	Bias	Spread
x-correlation of ε	Estimate parameters u such that $e=f_e(x,u)^{(1)}$	No improvement needed
Heteroscedasticit y	No improvement needed	Consider the variance as a x-dependent function and/or apply the weighted regression$^{(2)}$
ε are autocorrelated $^{(3)}$	Use ARCH models for time series autocorrelation	Apply bootstrap to ARCH models
Non-normality of the distributions	Calculate the x-dependent median minimizing the Minkovski R-distance (R=1, robust regression) of non-linear models instead of the sum-of-squares (mean)	Use empirical distributions (non-parametric spread statistics) and avoid using either t-student or z percentiles

$^{(1)}$ f_e is a non-linear regression function treated with LOOCV and CCA, e = residuals, x = independent variables of the model. $^{(2)}$ Variance may be affected by further variables different from x

$^{(3)}$ Autocorrelation does not usually affect software engineering data.

When regression assumptions do not hold, there are a number of consequences for inference to be aware of. Before dealing with improvements, let us recall some concepts about PIs. A PI is a range where, most probably, the next estimates will fall with a specific confidence (e.g., 95%). A PI is different from a confidence interval (CI) because the latter refers to a parameter of the distribution, while a PI refers to the next estimate (PI ≥ CI). When dealing with linear-in-the-parameter models, we can estimate PIs as follows. If the regression model relies upon regression assumptions, the PI is readily available, i.e. $(y') \pm t_{1-\alpha/2}(N-Q-1) \cdot S\sqrt{1 + x'^T (X^T X)^{-1} x'}$. Where, $y' = f_R(x',b) = b_0 + b_1 x'_1 + \ldots + b_Q x'_Q = bx'$ is the expected value (mean) of the dependent variable (y) when the independent variables get the next value $x' = (1\ x'_1, \ldots, x'_Q)^T$, i.e., $(x_1 = x'_1 \ldots, x_Q = x'_Q)$. $t_{1-\alpha/2}(N\text{-}Q\text{-}1)$ is a two tail t-value (Student's percentile) with N-Q-1 degrees of freedom. X is the observation matrix of independent variables where the first column is composed of only 1s. N is the number of observations. Q is the number of the independent variables. $S = \sqrt{\frac{Y^T(I-H)Y}{N-Q-1}} = \sqrt{MSE}$ is an unbiased estimator of the standard deviation of the population (σ). Y is the observation vector of the dependent variable. I is the identity matrix $H = X(X^T X)^{-1}X^T Y$, and MSE stands for Mean Squared Error. For non-linear models, a slightly different formula can be applied [6]. The formula for calculating PIs of MFNNs is $(y') \pm t_{1-\alpha/2}(N-K-1) \cdot S\sqrt{(1+g(x)^T(J^TJ)^{-1}g(x))}$, where $y' = f_R(x',b)$ is the expected value (mean) of the dependent variable (y) when the independent variables get the next value $x' = (1\ x'_1, \ldots, x'_Q)^T$. $t_{1-\alpha/2}(N-K-1)$ is a two tail t-value (Student's percentile) with N-K-1 degrees of freedom. N is the number of observations, K is the number of parameters (note that K > Q).

$S = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - y')^2}{N-K-1}}$ is an unbiased estimator of the standard deviation of the population (σ). Y is the observation vector of the dependent variable. g(x) is a vector whose ith element is the partial derivative $\partial f_R(x',b)/\partial b_i$ evaluated at its true value. J is a matrix whose ijth element is the partial derivative $\partial f_R(x_i,b)/\partial b_j$. J can be calculated iteratively through the training procedure (i.e. Backpropagation) [3]. Table 1 summarizes some improvements for estimating PIs of linear and non-linear models when regression assumptions do not hold.

To deal with heteroscedasticity (i.e. a non-constant variance), we have to consider the variance of the dependent variable as an x-dependent function. Further, if the probability distributions of variables and errors are not Gaussian, PI formulas reported above cannot be applied anymore. Conversely, non-parametric statistics should be preferred. In this work, we mainly refer to the empirical methodology defined by Jørgensen et al. [9] rearranged for parametric models.

3. The Problem

So far, we have seen that for improving the prediction capability of linear models, we can use non-linear models because of their parsimony and indefinite flexibility that linear models having the same number of variables as the non-linear models do not have [4, chapter 1]. Further, we have seen that, for improving inference (e.g., estimating PIs) of both linear and non-linear models, when regression assumptions are violated, a number of measures can be taken into account (Section 1, Table 1). Since we aim at improving estimation and inference at the same time, we should consider non-linear models (i.e., MFNNs).

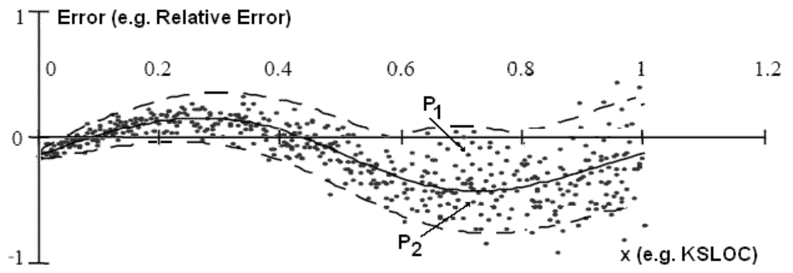

Fig. 1. Errors are x correlated, with increasing variance, biased, and with outliers. The solid line is the expected relative error (x-dependent median) and the region bounded by dashed lines is the associated 95% prediction interval

The mainly problem that we deal with in this work is about estimating PIs of linear and non-linear models (MFNNs) when regression assumptions are violated (Section 2). Although for illustrating the problem we refer to a two-dimensional space, considerations made below can be applied, without generality loss, to an N-dimensional space, where the variance depends on $x = x_1, \ldots, x_Q$, $Q > 2$[1].

Even though we can apply improvements to a linear and non-linear estimation model as shown in Table 1, when regression assumptions are violated, we may have a situation like the one in Figure 1. We have considered a relative (i.e. weighted) measure of the error, i.e. RE = e/Actual = (Actual – Estimated)/ Actual, instead of the residuals (i.e., e) to avoid that the error increases as x values grow. Further, RE is well known and applied in software cost estimation. Note that, other error measures can be used (e.g. BRE) [12], [9]. Since almost never we can assume that the error distribution is a Gaussian with fixed parameters $(0, \sigma^2 I)$, calculating PIs by formulas based on t-student or z percentiles leads to making errors. Consequently, we may have type I or II errors. As an example (Figure 1), if x = 0.8 KSLOC, the variance of the RE is expected to be greater than the average variance of the sample (i.e. σ^2). Therefore, if we estimated the spread by the average variance we would underestimate the real uncertainty. In Figure 1, when x = 0.8, the expected bias is not zero. It should be calculated by the non-linear regression function in Figure 1 (solid line).

4. The mathematical solution

Approach that we propose is an alternative to the non-parametric bootstrap method [5] and it is an evolution of the Jørgensen's approach for calculating empirical PIs for both regression-based models [8] and human judgment [9]. The proposed methodology may be bootstraped as well. In this first definition, however, we do not consider the bootstrap procedure, for the sake of clearness. The proposed strategy for estimating PIs is based on removing as many regression assumptions as possible and considering the error sample in Figure 1 (weighted

[1] Before calculating the model parameters, independent and dependent variables are transformed to the same ratio scale.

residuals/relative error, RE). Since we do not assume that the distribution is a Gaussian, we consider the distribution asymmetric and affected by outliers. Moreover, we assume an x-correlated RE. The solution is to calculate the non-linear robust regression function of y (standing for RE) with respect to x (standing for KSLOC), i.e. the solid line in Figure 1. Note that, in real cases, we would have more than one x-variable. That regression function provides an x-dependent median, minimizing the Minkowski R-distance (R = 1). It is called robust regression because it is less sensitive to outliers and asymmetric distributions.

To deal with the heteroscedasticity issue, we estimate the x-dependent variance empirically. The strategy is based on turning the problem into a two-class discrimination problem (Bayesian approach). In particular, we use the x-dependent median for splitting up the sample into two classes (solid line in Figure 1). Class A (upper side in Figure 1) and class B (lower side in Figure 1). We use elements of classes A and B as representatives of the unobserved data points of each class respectively. Then, we train a Multi-layer Feed-forward Neural Network for Discrimination (MFNND) in such a way that its output provides a classification decision (i.e. a new data point is classified as belonging to A or B according to its similarity to the data used for training). MFNND is a generalization of the traditional logistic regression, logit [3], [4], and [6]. We call MFNND as Bayesian Discrimination Function (BDF) because its output can be interpreted as the posterior probability that any input belongs to class A [3], [4]. Therefore, an input is classified as belonging to class A, if the BDF output is between [0.5, 1], e.g. 0.85. It is classified as belonging to class B otherwise, i.e. the BDF output is in [0,0.5[, e.g. 0.25, where [.,.] is a closed interval and [.,.[is a right-open interval. The defined BDF is expressed by the following relationship, $f_{BDF}(x_1 = RE, x_2=KSLOC) = [0,1]$, i.e. $y = f_{BDF}(x_1, x_2) = Pr(y=1|x_1, x_2)$, where the interval [0, 1] points out any real number in [0,1], x_1 and x_2 represent the sample information, and y = 1 represents class A (y = 0 represents class B). Assume that, the BDF yields $f_{BDF}(P_1) = 0.85$ (> 0.5) and $f_{BDF}(P_2) = 0.25$ (< 0.5). Then, project P_1 would be classified as belonging to class A, and project P2 would be classified as belonging to class B.

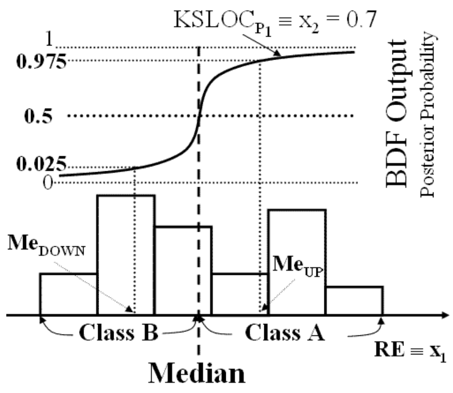

Fig. 2. Posterior probability density obtained by fixing KSLOC and letting RE vary

Assume now that instead of fixing both values x_1 = RE and x_2=KSLOC, we fix only x_2 (= constant c), and let x_1 vary. Then, BDF turns into $\Pr(y=1|x_2)=f_{|x_2=c}(x_1)=y(x_1)$. Note that, in this 2-variable example, the estimation model (EM) would have only one independent variable (x_2=KSLOC). In case of an N-dimensional space (with N > 1), we would fix all variables except x_1 (the relative error RE), i.e. $\Pr(y=1|x_2..x_N)=f_{|x_2=c_1,\dots,x_N=c_{N-1}}(x_1)=y(x_1)$ and the EM would have (N – 1) variables, i.e. (x_2, …, x_N). Then, we predict the RE prediction interval of the EM by feeding the project values (i.e., the EM inputs for the project) into the BDF and applying the solution in Figure 2. Once we build the posterior probability density (solid line in Figure 2), we can obtain a (Bayesian) PI by fixing a 95% confidence, i.e. (0.025, 0.975), and picking the corresponding values of RE on the x-axis, i.e. (Me_{DOWN}, Me_{UP}). This interval represents the expected range where the next RE will fall. The posterior probability density in Figure 2 has an important characteristic. Its slope gets steeper as the variance in Figure 1 decreases. It gets flatter as the variance increases [6]. To calculate the PI corresponding to the RE range, i.e. (Me_{DOWN}, Me_{UP}), we first consider the formula RE = (Actual – Estimated)/Actual and then, we deduce Actual = Estimated/(1 – RE). As shown by Jørgensen et al. [9], the PI is then $[O_{est}^{N+1}/(1-Me_{DOWN}), O_{est}^{N+1}/(1-Me_{UP})]$, where $O_{est}^{N+1}=f_R(x',b)$, e.g. $f_R(x' = 0.7,b)$, see Section 2. For instance, assume that the RE interval obtained from Figure 2 is [-0.9, 0.1] and Estimated = 3 person months, then the PI is [3/(1-(-0.9)), 3/(1-0.1)] =[1.6, 3.4] person months.

6. DISCUSSION AND CONCLUSION

Based on results in Section 5, we define an improvement and risk mitigation strategy. Gray rectangles in Figure 3 are all the possible kinds of RE ranges that we can obtain from the error analysis in Figure 2. An error PI is unbiased, if it includes zero ([a] and [b]). It is biased otherwise ([c] and [d]). A PI is useful if it is within stated thresholds ([a] and [c]). It is useless otherwise ([b] and [d]). An error PI is acceptable if it is useful and unbiased at the same time ([a]). It is unacceptable otherwise ([b], [c], and [d]). Figure 3 allows us to figure out the improvement needs (e.g. [a] does not need any improvement, while [b], [c], and [d] do). When dealing with biased intervals, we highlight the fact that, the model is incorrect (e.g., missing variables, lack of flexibility). When the interval is too wide, further variables (e.g., dummy variables) should be considered. Analysis in Figure 3 can be used for defining a mitigation strategy as well, e.g. we may choose an estimate that minimizes the error of underestimates. Note that, PI $[O_{est}^{N+1}/(1-Me_{DOWN}), O_{est}^{N+1}/(1-Me_{UP})]$ is a way of making the estimates unbiased, even though the formula cannot improve the estimation model (the estimation model keeps biased).

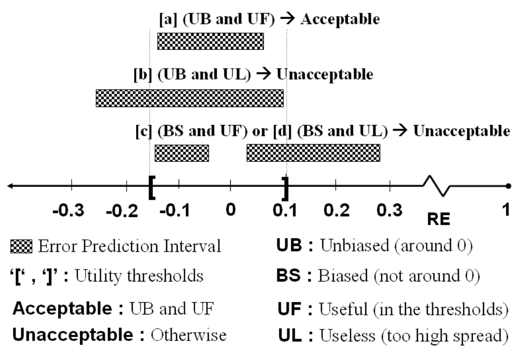
[a] (UB and UF) → Acceptable
[b] (UB and UL) → Unacceptable
[c] (BS and UF) or [d] (BS and UL) → Unacceptable
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
RE
1
▨ Error Prediction Interval
'[' , ']' : Utility thresholds
Acceptable : UB and UF
Unacceptable : Otherwise
UB : Unbiased (around 0)
BS : Biased (not around 0)
UF : Useful (in the thresholds)
UL : Useless (too high spread)

Fig. 3. Acceptability of error prediction intervals

We tested our improvement strategy in a real case based on the COCOMO NASA data set [15] and log-linear regression functions. Although the COCOMO NASA data set is more than twenty years old, we used it anymore because what we really needed to test was the proposed mathematical model improvement, not the relevance of the COCOMO variables and related data. The COCOMO data set seemed to be a good choice also because a well-known example could better enhance comprehension, discussion, and replication than recent models whose validity has not been fully accepted yet. Therefore, we do not suggest using the COCOMO model for improving estimates, but we use its variables and data for illustrating our solution on an estimation model where regression assumptions are violated. The analysis starts at the beginning of 1985, when NASA has already developed 77 software systems, which represent their experience. The NASA's goals are to (1) estimate the cost of 16 next software systems (from 1985 to 1987), (2) exploit such experience to improve their estimation process, and (3) state suitable mitigation strategy when estimating the remaining 16 projects. The expected range of the relative error for each project being estimated (16 overall) is reported in Figure 4. In particular, the letter 'S' shows a possible *scope error* it happens when the RE falls out of the PI. For the remaining cases, A *square* expresses that the interval is too wide (i.e. useless for inference) and the model needs to be improved (finding dummy variables). A *circle* expresses that the magnitude of the PI is useful. The letter 'U' means that the interval is unbiased (includes zero) and the letter 'B' means that the interval is biased. If the model is biased, it needs some improvement (e.g., increasing the flexibility, adding variables).

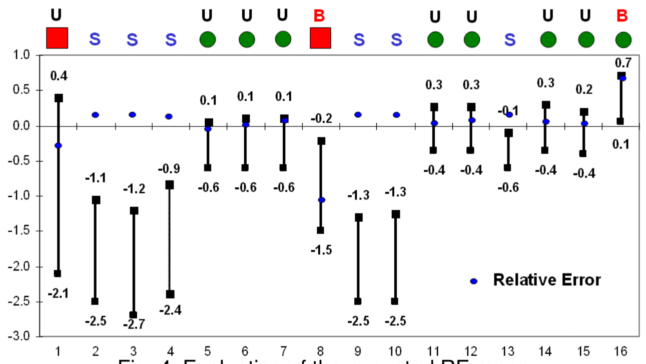

Fig. 4. Evaluation of the expected RE range

Based on the analysis in Figure 4, we infer that the EM does not need any improvement for estimating projects 5, 6, 7, 11, 12, 14, and 15 (negligible risk). The EM needs some dummy variables for shrinking the PI on project 1 (low risk) as well as the EM needs some improvements for estimating projects 16 (high risk) and project 8 (very high risk). The EM cannot be used for estimating projects (2, 3, 4, 9, and 10) because a *scope error* may occur (unpredictable risk).

8. References

[1] V. R. Basili, G. Caldiera, H. D. Rombach, "The Experience Factory," In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons, 1994.

[2] A. Barron, "Universal approximation bounds for superposition of a sigmoidal function", IEEE Transaction on Ifromation Theory, 39, pp. 930-945, 1993.

[3] Bishop C., "Neural Network for Pattern Recognition," Oxford University Press, 1995.

[4] G. Dreyfus, "Neural Networks Methodology and Applications," Springer, 2005.

[5] B. Efron and R.J. Tibshirani, "An Introduction to the Bootstrap," Chapman & Hall, NY, 1993.

[6] D. Husmeier, R. Dybowski, and S. Roberts, "Probabilistic Modeling in Bioinformatics and Medical Informatics," Springer, 2004.

[7] I.T. Jollife, "Principal Component Analysis," Springer, 1986.

[8] M. Jørgensen, "Regression Models of Software Development Effort Estimation Accuracy and Bias," Empirical Software Engineering, Vol. 9, 297-314, 2004.

[9] M. Jørgensen and D.I.K. Sjøberg, "An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy" Journal of Information Software and Technologies 45: 123-136, 2003.

[10] B. Kitchenham and S. Linkman, "Estimates, Uncertainty, and Risk." IEEE Software, 14(3), pp. 69-74, May-June 1997.

[11] A.D.R. McQuarrie, C. Tsai, "Regression and Time Series Model Selection," World Scientific, 1998.

[12] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models," IEEE Trans. Software Eng., vol. 31, no. 5, pp. 380-391, May 2005.

[13] S.A. Sarcia, G. Cantone and V. R. Basili, "Adopting Curvilinear Component Analysis to Improve Software Cost Estimation Accuracy," EASE08, Bari (Ialy), 2008.

[14] M. Shepperd, "Software project economics: a roadmap," FOSE'07, IEEE, 2007.

[15] J.S. Shirabad, and T. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca /SERepository, 2005.

[16] V.N. Vapnik, "The Nature of Statistical Learning Theory," Springer, 1995.