A new boosting-based software reliability growth model

Lev V. Utkin[1], Frank P.A. Coolen[2]

[1] *Telematics Department, Central Scientific Research Institute of Robotics and Technical Cybernetics, Peter the Great Saint-Petersburg Polytechnic University, St. Petersburg, Russia*
[2] *Department of Mathematical Sciences, Durham University, United Kingdom*

Preprint submitted to Communications in Statistics - Theory and Methods *February 19, 2020*

A new boosting-based software reliability growth model

Lev V. Utkin[1], Frank P.A. Coolen[2]

[1] *Telematics Department, Central Scientific Research Institute of Robotics and Technical Cybernetics, Peter the Great Saint-Petersburg Polytechnic University, St. Petersburg, Russia*
[2] *Department of Mathematical Sciences, Durham University, United Kingdom*

Abstract

A new software reliability growth model (SRGM) called RBoostSRGM is proposed in this paper. It can be regarded as a modification of the boosting SRGMs through the use of a reduced set of weights to take into account the behaviour of the software reliability during the debugging process and to avoid overfitting. The main idea underlying the proposed model is to take into account that training data at the end of the debugging process may be more important than data from the beginning of the process. This is modelled by taking a set of weights which are assigned to the elements of training data, i.e., to the series of times to software failures. The second important idea is that this large set is restricted by the imprecise ε-contaminated model. The obtained RBoostSRGM is a parametric model because it is tuned in accordance with the contamination parameter ε. As a variation to this model, we also consider the use of the Kolmogorov-Smirnov bounds for the restriction of the set of weights. Various numerical experiments with data sets from the literature illustrate the proposed model and compare it with the standard non-parametric SRGM and the standard boosting SRGM.

Keywords: software reliability growth model; boosting; support vector regression; extreme points; imprecise contaminated model; Kolmogorov-Smirnov bounds; pairwise comparisons.

1. Introduction

Reliability of software is crucial in many application areas, yet assuring high reliability is difficult. Lyu (1996) defined software reliability as the probability of failure-free software operation for a specified period of time in a specified environment. Many models for software reliability estimation and prediction have been proposed, where predicting the software reliability based on information from the software debugging process is usually efficient. The model assumption that software errors are removed during debugging or testing leads to the term 'software reliability growth model' (SRGM). Although assumptions for SRGMs are often doubtful in practice (Cai et al, 1991), they have become popular models and they can mostly be divided into two groups: parametric and non-parametric models (Hua et al, 2007; Li et al, 2012a; Roy et al, 2014).

Parametric SRGMs (e.g. Jelinski and Moranda (1972); Goel and Okomoto (1979); Li et al (2012a); Tian and Noore (2005)) are based on assumptions about the behaviour of the software faults and failure processes, typically reflected through parametric probability distributions for time between failures or related process characteristics. Non-parametric SRGMs utilize methods such as artificial neural networks, genetic programming or support vector machines (Hastie et al, 2001; Vapnik, 1998) (SVMs) to predict the software reliability. An advantage of such methods is that they limit prior assumptions and are mainly based on fault history data. Many non-parametric SRGMs have been proposed in recent years (Amina et al, 2013; Cai et al, 2001; Chiu et al, 2008; das Chagas Moura et al, 2011; Govindasamy and Dillibabu, 2018; Hua et al, 2007; Jaiswal and Malhotra, 2018; Jin and Jin, 2014; Kim et al, 2015; Kumar and Banerjee, 2015; Kumar and Singh, 2012; Li et al, 2012b,a; Lou et al, 2010; Pai and Hong, 2006; Sun et al, 2018; Tian and Noore, 2005; Xing et al, 2005; Yang et al, 2010). A review of using soft computing approaches in non-parametric SRGMs is provided by Lohmor and Sagar (2018).

Many SRGMs are based on regression models. We consider a support vector regression (SVR) which can be regarded as a special case of the SVM with a loss function of a specific form for constructing regression functions. Due to many important features of the SVR, many authors (das Chagas Moura et al, 2011; Jin and Jin, 2014; Tian and Noore, 2005; Xing and Guo, 2005) have applied this learning technique to software reliability modelling. A second important approach to constructing non-parametric SRGMs is by using ensemble-based methods, in particular, boosting algorithms. We will study a boosting algorithm which uses the SVR for constructing the so-called weak regressor.

Vamsidhar and Raju (2011) proposed to combine well-known parametric NHPP SRGMs, including the Goel-Okumoto model, the Musa-Okumoto model, the delayed S-shaped model, the inflected S-shaped model, and the generalized Goel-Okumoto model, by means of the AdaBoost algorithm. The proposed use of the AdaBoost algorithm is very similar to the well-known models based on combination of predictions obtained from different SRGMs and proposed by Littlewood et al (1993). Similar SRGMs were studied by Hsu and Huang (2009). A short review of SRGMs based on boosting methods is provided by Vamsidhar et al (2012).

Li et al (2012b) point out that the AdaBoost algorithm as learning technique has not commonly been applied to software reliability engineering. They propose two interesting AdaBoosting-based combination approaches and investigate the effectiveness of these two approaches for improving the estimation and prediction power of the SRGMs. The first one, named 'self-combination approach', selects several variations of one original SRGM as the weak predictors, and uses the AdaBoost algorithm to determine the weights of these variations, by training these variations on the failure data sets in order to obtain the final linear self-combination model to improve the performance of this original SRGM. The second approach, named 'multi-combination approach', selects several different candidate SRGMs as the weak predictors, and uses the AdaBoost algorithm to determine the weights of these models by training these SRGMs on the failure data sets, in order to obtain the final linear multi-combination model to improve the performance of all candidate models.

The second combination approach proposed by Li et al (2012b) is similar to that proposed by Vamsidhar and Raju (2011). The use of boosting techniques to improve the performance of software reliability models was proposed by Costa et al (2007).

In spite of the successful application of the SRGMs based on using the AdaBoost algorithm, we meet all problems intrinsic to the AdaBoost method itself. It should be noted that the main advantage of AdaBoost over other boosting techniques is that it is adaptive, i.e., it is able to take advantage of weak hypotheses that are more accurate than was assumed a priori. Adaptation is a very important feature of the AdaBoost and other ensemble-based algorithms. At the same time, it has been reported by many authors that the main reason for poor learning results of AdaBoost in the high-noise regime is that the algorithm produces a skewed data distribution, by assigning too large weights to a few hard-to-learn data points. This leads to overfitting. In order to overcome this difficulty, we should somehow restrict the set of possible weights assigned to data points. Several modifications of boosting algorithms, which restrict the set of weights of examples in training data, have been proposed (Domingo and Watanabe, 2000; Nakamura et al, 2004; Servedio, 2003; Utkin, 2013, 2015). In fact, every modification is a choice of some rule restricting the weights. However, the available approaches to restrict the weights do not take into account peculiarities of the software debugging process and their application to the software reliability modelling may lead to unsatisfactory results. Therefore, in this paper, we propose a new non-parametric SRGM based on the AdaBoost algorithm with the SVR, called AdaBoost.R2, which restricts the set of weights by combining two rules.

It should be noted that in order to reduce the set of all possible weights represented as the unit simplex in AdaBoost.R2, Utkin and Wiencierz (2015) proposed to apply several well-known imprecise statistical models, including the linear-vacuous mixture or imprecise ε-contaminated model (Walley, 1991), the pari-mutuel model (Walley, 1991), the constant odds-ratio model (Walley, 1991, Subsection 3.3.5.) and an imprecise model (Utkin and Coolen, 2014) using the well-known Kolmogorov-Smirnov bounds (Johnson and Leone, 1964, Subsection 8.9.3.). The choice of a suitable model depends on the specific application. We show below that a good choice for constructing SRGMs is the imprecise ε-contaminated model which allows us to tune an analyzed SRGM by looking for the optimal value of the parameter ε.

Typically in software development and testing, training data from the later stages of the debugging process are likely to be more important for prediction of software reliability than data from the early stages of the process. Simple errors in software are removed at the beginning of the debugging process and the software may actually be changed, so early failures are less relevant for predicting future reliability.

In this paper, we introduce additional weights which reflect the comparative information underlying the debugging process. The comparative information is represented by pairwise comparisons. At first sight, we could use many approaches to formalize the pairwise comparison information, for example, Fishburn's representation of preferences (Fishburn, 1999). However, every such approach leads to assignment of precise weights which make it impossible to use the boosting algorithms. Therefore, we propose to consider a set of

weights produced by pairwise comparisons. This set is also a subset of the unit simplex and it can be combined with the subset produced by the imprecise ε-contaminated model in order to improve the AdaBoost.R2 algorithm applied to SRGMs. We should emphasize that this second subset enables training data from the later stages of the debugging process to have more weight than data from earlier stages of the process, but this is not required. The set of weights allows all data points to have the same weight, or later ones to have more weight. Numerical experiments will show that optimisation of the weights over the intersection of the two subsets used tends to lead to good results. The obtained modification of the AdaBoost.R2 algorithm uses the weighted SVR as a weak regressor in order to implement non-linear regression functions. In addition, we also consider the intersection of a set of weights produced by the well-known Kolmogorov-Smirnov bounds and the pairwise comparison information. The idea to reduce the set of weights by means of two subsets gives the corresponding name of the proposed SRGM: RBoostSRGM (Reduced Boosting SRGM).

This paper is organized as follows. A general approach to the non-parametric SRGM is presented in Section 2. Section 3 contains a short description of the original AdaBoost.R2 (Drucker, 1997) and its modification taking into account the reduced set of possible weights. The introduced sets of weights for data points corresponding to times to software failures, and their extreme points required for implementing the modified AdaBoost.R2, are studied in Section 4. In Section 5 we consider the intersection of a set of weights produced by the well-known Kolmogorov-Smirnov bounds and the pairwise comparison information. Numerical experiments with public software reliability data illustrating the proposed SRGMs and investigating their performance are given in Section 6.

2. A general approach to the non-parametric SRGM and the weighted SVR

The non-parametric SRGM is formulated in the general framework of predictive learning problems (Hastie et al, 2001; Vapnik, 1995, 1998) as follows. Parameters of the regression models are computed by minimizing a risk functional defined by a certain loss function and by a probability distribution of the noise. In regression analysis, we try to estimate a functional dependency $f(\mathbf{x})$ between a set of sampled points $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ taken from $\mathbb{R}^m$, and target values $Y = (y_1, y_2, ..., y_n)$ with $y_i \in \mathbb{R}$. In the framework of machine learning the regression problem can be stated in the same way. Given n training data (also called 'examples') $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathbb{R}^m$ represents a feature vector involving m features and $y_i \in \mathbb{R}$ is an output variable. Let us write the function $f(\mathbf{x})$ in the form $f(\mathbf{x}) = \langle a, \phi(\mathbf{x})\rangle + b$, where $a = (a_1, ..., a_m)$ and b are parameters of the function; $\phi(\mathbf{x})$ is a feature map $\mathbb{R}^m \to G$ such that the data points are mapped into an alternative higher-dimensional feature space G; $\langle \cdot, \cdot \rangle$ denotes the canonical dot product. The map ϕ is used to get the non-linear regression function f. A learning machine has to choose a function $f(\mathbf{x})$, from a given set of functions, which best approximates the unknown dependency. This can be done in different ways, for example, by minimizing the empirical expected loss, where the loss function represents a measure of difference between

the estimate f and the actual value y given by the unknown function at a point $\mathbf{x}$. The standard SVR technique is to assume that the probability distribution of the training points is the empirical (non-parametric) probability distribution function whose use leads to the empirical expected risk

$$R_{\text{emp}} = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f(\mathbf{x}_i)).$$

Here l is the so-called ϵ-insensitive loss function with parameter ϵ, which is defined as

$$l(y, f(\mathbf{x})) = \begin{cases} 0, & |y - f(\mathbf{x})| \leq \epsilon, \\ |y - f(\mathbf{x})| - \epsilon, & \text{otherwise.} \end{cases}$$

Many realizations of the SVR use some distribution of weights $w = (w_1, ..., w_n)$, $w_1 + ... + w_n = 1$, instead of the uniform distribution $(1/n, ..., 1/n)$ over the sampled points S, in order to incorporate prior knowledge about importance of the data points. The expected risk in this case is of the form:

$$R_w = \sum_{i=1}^{n} w_i l(y_i, f(\mathbf{x}_i)). \tag{1}$$

The parameters a and b of the function f are estimated by minimizing the following regularized risk function:

$$R_{\text{SVR},w} = \frac{1}{2}\langle a, a\rangle + C\sum_{i=1}^{n} w_i l(y_i, f(\mathbf{x}_i)).$$

Here the first term is the standard Tikhonov regularization or smoothness term (Tikhonov and Arsenin, 1977); $C > 0$ is the constant "cost" parameter which determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated (Smola and Scholkopf, 2004).

If we introduce slack variables ξ_i and ξ_i^*, then the problem of minimizing the regularized risk function can be written as follows:

$$\text{minimize } \frac{1}{2}\langle a, a\rangle + C\sum_{i=1}^{n} w_i\,(\xi_i + \xi_i^*) \tag{2}$$

subject to $\xi_i \geq 0$, $\xi_i^* \geq 0$,

$$y_i - \langle a, \phi(\mathbf{x})\rangle - b \leq \epsilon + \xi_i, \tag{3}$$

$$\langle a, \phi(\mathbf{x})\rangle + b - y_i \leq \epsilon + \xi_i^*. \tag{4}$$

This optimisation problem can be written in its dual formulation utilizing Lagrange multipliers α_i, α_i^*, $i = 1, ..., n$, as (Tay and Cao, 2002)

$$\text{maximize } \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) - \epsilon\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)$$

$$-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j), \tag{5}$$

subject to

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0, \tag{6}$$

$$0 \leq \alpha_i \leq Cw_i,\ 0 \leq \alpha_i^* \leq Cw_i,\ i = 1, ..., n. \tag{7}$$

Here $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ is the kernel function that is the inner product of the points $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ mapped into feature space. We do not need to explicitly compute the function $\phi(\mathbf{x})$ due to the kernels. It can be seen from the optimization problem that only the kernels are used in the objective function. Typical examples of kernel functions are linear, polynomial and Gaussian (Scholkopf and Smola, 2002). In this paper, we will use the Gaussian kernel which is defined as

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\left\|\mathbf{x} - \mathbf{y}\right\|^2 / \sigma^2\right),$$

where σ is the kernel parameter determining the geometrical structure of the mapped samples in the kernel space. The regression function f can now be written in terms of Lagrange multipliers as

$$f(\mathbf{x}) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(\mathbf{x}, \mathbf{x}_i) + b.$$

A simple expression for computing b can be found in Smola and Scholkopf (2004).

3. AdaBoost.R2 and its modification

A typical boosting algorithm for constructing a regression model consists of the following steps. First, the regression model is constructed by means of a number (say, T) of weak regressors f_t, $t = 1, ..., T$. The term weak is taken from the classification ensemble-based methods. Second, weights of data points in the training set for every weak regressor are updated in accordance with certain rules taking into account the accuracy of the regressor. Third, various weak regressors are combined in order to form a final regressor. According to Rokach (2010), a boosting algorithm improves the performance accuracy because it generates a final classification or regression algorithm whose accuracy can be increased by combining many weak algorithms whose accuracies may be low.

There are several modifications of ensemble-based regression methods. The simplest one is called AdaBoost.R2 Drucker (1997). Initially, identical weights $h_1 = (1/n, ..., 1/n)$ are assigned to all data points in the training set. In each iteration, the weights of all points with a large error measure are increased while the weights of points corresponding to small values of the error measure are decreased. As a consequence, the weak regressor is forced to focus on the "hard" data points of the training set by performing additional iterations and creating more regressors. The "hard" data points are characterized by the large error measure. According to AdaBoost.R2 (Drucker, 1997), the largest error $D_t = \max_{j=1,...,n} |y_j - f_t(\mathbf{x}_j, \mathbf{w})|$ is computed at each boosting iteration in order to normalize the real-valued error $|y_j - f_t(\mathbf{x}_j, \mathbf{w})|$ in such a way that each adjusted error is in the interval from

0 to 1. As a result, we get the error measure $e_j(t) = |y_j - f_t(\mathbf{x}_j, \mathbf{w})|/D_t$ of each data point at the t-th iteration. The distribution h_t is updated using a rule modifying the weights of data points. One of the well-known rules is of the form $h_{t+1}(i) = h_t(i) \cdot \exp(-\alpha_t(1 - e_i(t)))$, where α_t is a weight assigned to every weak regressor. This weight measures the overall accuracy of the regressor and it depends on the error measures $e_1(t), ..., e_n(t)$ of all data points. In fact, the weight α_t measures also the importance that is assigned to the regressor, i.e., higher weights are given to more accurate classifiers. The final regression function f is a weighted majority vote of the T weak regressors with coefficients α_t. Detailed steps of AdaBoost.R2 are represented as Algorithm 1.

Algorithm 1 AdaBoost.R2

Require: T (number of iterations), S (training set)
Ensure: $f_t, \alpha_t,\ t = 1, ..., T$
  $t \leftarrow 1$; $h_i(t) \leftarrow 1/n$; $i = 1, ..., n$.
  repeat
    Build the regression model f_t using distribution h_t
    $D_t \leftarrow \max_{j=1,...,n} |y_j - f_t(\mathbf{x}_j, \mathbf{w})|$.
    Calculate the adjusted error $e_i(t)$ for each instance: $e_i(t) \leftarrow |y_i - f_i(\mathbf{x}_i, \mathbf{w})|/D_t$
    Calculate the expected error of f_i: $\epsilon_t \leftarrow \sum_{i=1}^{n} e_i(t) h_i(t)$
    if $\epsilon_t > 0.5$ **then**
      $T \leftarrow t - 1$
      exit Loop.
    end if
    $\alpha_t \leftarrow \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    $h_{t+1}(i) \leftarrow h_t(i) \cdot \exp(-\alpha_t(1 - e_i(t)))$
    Normalize $h(t+1)$ to be a proper distribution.
    $t++$
  until $t > T$

It can be seen from Algorithm 1 that it does not substantially differ from the standard AdaBoost for classification (Freund and Schapire, 1997). The main difference is in determining the expected error ϵ_t for each iteration. We can observe that the weights h_t are restricted by the unit simplex denoted as $S(1, n)$. This is the main reason for overfitting because the weights of data points tend to concentrate on "hard" points and weak regressors may be constructed by taking into account only these "hard" data points.

In order to overcome this difficulty and to improve AdaBoost.R2, Utkin and Wiencierz (2015) proposed the following modification of AdaBoost.R2, restricting the set of weights of data points by some convex set which is a subset of the unit simplex. Denote a convex and compact subset of the unit simplex as $\mathcal{P}$. The set $\mathcal{P}$ is totally defined by its extreme points due to its convexity. Denote the extreme points $q_k = (q_1^{(k)}, ..., q_n^{(k)})$, $k = 1, ..., r$. Then every weight $h = (h_1, ..., h_n)$ from $\mathcal{P}$ can be represented as the linear combination of

the extreme points

$$h = \sum_{k=1}^{r} \lambda_k \cdot q_k. \tag{8}$$

Here $\lambda = (\lambda_1, ..., \lambda_r)$ is a vector of non-negative weights such that $\lambda_1 + ... + \lambda_r = 1$. The main idea of the AdaBoost.R2 modification is to modify the weight vector λ instead of h. Due to this replacement, a new modified vector h computed from (8) will be inside the set $\mathcal{P}$. At the same time, the weight vector λ can be arbitrarily changed in the unit simplex having r vertices denoted $S(1, r)$.

The proposed modification is implemented as follows. Initially, we take identical weights $\lambda_k(t) = 1/r$, $k = 1, ..., r$. Then we compute the vector h from $\mathcal{P}$ through λ by means of (8). The vector h is applied to the weighted regression procedure in order to get the t-th regressor. In order to get the error measure of the t-th regressor, the k-th extreme point mean error, denoted by ε_k, is introduced as follows:

$$\varepsilon_k = \sum_{i=1}^{n} e_i(t) q_i^{(k)}.$$

Here $e_i(t)$ has been introduced in Algorithm 1. The overall error of the t-th regressor is determined as

$$\epsilon_t = \sum_{k=1}^{r} \lambda_k \varepsilon_k.$$

The updating rule from AdaBoost.R2 can be modified in the following way:

$$\lambda_k(t+1) = \lambda_k(t) \cdot \exp(-\alpha_t(1 - \varepsilon_k)),$$

where

$$\alpha_t = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \ln\left(\frac{1 - \sum_{k=1}^{r} \lambda_k \varepsilon_k}{\sum_{k=1}^{r} \lambda_k \varepsilon_k}\right).$$

The detailed description and explanation of the proposed algorithm can be found in (Utkin and Wiencierz, 2015). It is represented here in the form of Algorithm 2.

Utkin and Wiencierz (2015) point out that the quality of regression can be controlled by choosing the optimal parameters of the set $\mathcal{P}$. Another advantage of the proposed algorithm is that arbitrary constraints for the set $\mathcal{P}$ can be introduced as additional prior information about the training set. It is an important property which allows us to take into account peculiarities of the software debugging process in the methods presented in this paper.

It should also be noted that there are various learning techniques to realize the weak regressors in the AdaBoost.R2 and its modification, for example, the Lasso or ridge regression methods (Hastie et al, 2001) and Bayesian linear regression (Carlin and Louis, 2008). However, we select the weighted SVR method which can efficiently deal with non-linear regression functions.

Algorithm 2 Imprecise AdaBoost.R2

Require: T (number of iterations), S (training set), $q_1, ..., q_r$ (extreme points of $\mathcal{P}$)
Ensure: $f_t, \alpha_t,\ t = 1, ..., T$

$t \leftarrow 1$
$\lambda_k(1) \leftarrow 1/r$; $k = 1, ..., r$, or it can be randomly selected from the unit simplex.
repeat
  Compute the vector $h \leftarrow \sum_{k=1}^{r} \lambda_k(t) \cdot q_k$
  Build the regression model f_t using distribution h_t
  $D_t \leftarrow \max_{j=1,...,n} |y_j - f_t(\mathbf{x}_j, \mathbf{w})|$.
  Calculate the adjusted error $e_i(t)$ for each instance: $e_i(t) \leftarrow |y_i - f_t(\mathbf{x}_i, \mathbf{w})|/D_t$
  Calculate the k-th extreme point mean error $\varepsilon_k \leftarrow \sum_{i=1}^{n} e_i(t) q_i^{(k)}$
  Calculate the adjusted error of f_t: $\epsilon_t \leftarrow \sum_{k=1}^{r} \varepsilon_k(t)\lambda_k(t)$
  if $\epsilon_t > 0.5$ **then**
    $T \leftarrow t - 1$
    exit Loop.
  end if
  $\alpha_t \leftarrow \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  $\lambda_k(t+1) \leftarrow \lambda_k(t) \cdot \exp(-\alpha_t(1 - \varepsilon_k))$
  Normalize $\lambda(t+1)$ to be a proper distribution.
  $t++$
until $t > T$

4. Weights for observations and their extreme points

Several well-known imprecise statistical models have been studied by Utkin and Wiencierz (2015) as the rules for reducing the unit simplex of weights in the AdaBoost.R2 algorithm. These include the linear-vacuous mixture or imprecise ε-contaminated model (Walley, 1991), the pari-mutuel model (Walley, 1991), the constant odds-ratio (π, ε) model (Walley, 1991, Subsection 3.3.5) and an imprecise model (Utkin and Coolen, 2014) using the well-known Kolmogorov-Smirnov bounds (Johnson and Leone, 1964, Subsection 8.9.3). We select two models which are simply explained from a reliability point of view and are used in reliability engineering. These are the imprecise ε-contaminated model, considered in this section, and the model using Kolmogorov-Smirnov bounds, considered in Section 5. The imprecise ε-contaminated model for the training set consisting of n points produces a set of weights having n extreme points. In fact, this is a small simplex included in the unit simplex. Moreover, the parameter ε of the model can be used for tuning the modified AdaBoost.R2 algorithm and the proposed SRGM based on the boosting method. The case $\varepsilon = 1$ produces the unit simplex and corresponds to the standard AdaBoost.R2 algorithm.

However, we have pointed out that in spite of the advantages of the imprecise ε-contaminated model, it does not take into account the different importance of the debugging data points, where training data from the later stages of the debugging process may be considered to be more important for prediction than data from the early stages of the debugging process. In order to take into account the difference in importance of the data points, we introduce additional weights which reflect the comparative information underlying the debugging process. Since the comparative information may be represented by pairwise comparisons, we consider a set of weights produced by the pairwise comparisons. This set is also a subset of the unit simplex and it can be combined with the subset produced by the imprecise ε-contaminated model in order to improve the AdaBoost.R2 algorithm applied to SRGMs. So we use the intersection of two subsets as a possible set of weights in the AdaBoost.R2 algorithm. This gives us a new modified AdaBoost.R2 algorithm.

4.1. The linear-vacuous mixture

The linear-vacuous mixture, or imprecise ε-contaminated model, produces the set $\mathcal{P}(\varepsilon, w)$ of probabilities $w = (w_1, ..., w_n)$ such that $w_i = (1-\varepsilon)p_i + \varepsilon h_i$, where $p = (p_1, ..., p_n)$ is an elicited probability distribution, $h_i \geq 0$ is arbitrary with $h_1 + ... + h_n = 1$, and $0 < \varepsilon < 1$. The set $\mathcal{P}(\varepsilon, p)$ is a subset of the unit simplex $S(1, n)$. Moreover, it coincides with the unit simplex when $\varepsilon = 1$. We denote the set $\mathcal{P}(\varepsilon, p)$ for the special case with $p = (1/n, ..., 1/n)$ as $\mathcal{P}(\varepsilon)$. This set can be produced by $n + 1$ hyperplanes

$$w_i \geq (1 - \varepsilon)n^{-1}, \ i = 1, ..., n, \ w_1 + ... + w_n = 1. \tag{9}$$

4.2. The set produced by comparative information

Let us consider the following comparative information:

$$0 \leq w_1 \leq w_2 \leq ... \leq w_n, \tag{10}$$

and the condition $w_1+...+w_n=1$. A set $\mathcal{M}$ of weights $w=(w_1,...,w_n)$, which is produced by $n-1$ inequalities of the form $w_i-w_{i-1}\geq 0$, n inequalities $w_i\geq 0$ and one equality $w_1+...+w_n=1$, has extreme points are of the form:

w_1	w_2	...	w_{n-2}	w_{n-1}	w_n
0	0	...	0	0	1
0	0	...	0	1/2	1/2
0	0	...	1/3	1/3	1/3
...	...	...	...	...	...
0	$1/(n-1)$	...	$1/(n-1)$	$1/(n-1)$	$1/(n-1)$
$1/n$	$1/n$	...	$1/n$	$1/n$	$1/n$

It is interesting to see from the extreme points that they correspond to cases when only several elements of training data are used for constructing a regression model. Moreover, we have two extreme cases. The first one is the distribution of weights having only one non-zero element. In this case, the decision is made on the basis of the last point. Another extreme case coincides with the uniform distribution accepted in the empirical expected risk.

4.3. Intersection of two sets

Before we study the intersection of sets $\mathcal{P}(\varepsilon)$ and $\mathcal{M}$ in general, we consider this intersection for the case $n=3$ by using the standard unit simplex. Fig. 1 illustrates the unit simplex in which every point is a possible weight vector (w_1,w_2,w_3). The set $\mathcal{M}$ corresponds to the area restricted by the triangle ABC. The set $\mathcal{P}(\varepsilon)$ corresponds to the area restricted by the small simplex. Their intersection is restricted by the triangle BED (the shaded area). We can see from the figure that the obtained set of weights is rather small. The weights used in the model are shifted towards the last (the third) vertex of the simplex. At the same time, the small simplex corresponding to $\mathcal{P}(\varepsilon)$ does not allow us to assign too large weights to the third vertex or to other vertices.

Proposition 1. *A set of extreme points of the intersection $\mathcal{P}(\varepsilon)\cap\mathcal{M}$ consists of n elements of the form:*

$$w_1=w_2=...=w_{i-1}=\frac{1}{n}-\frac{\varepsilon}{n},$$
$$w_i=w_{i+1}=...=w_n=\frac{1}{n}+\frac{\varepsilon}{n}\cdot\frac{i-1}{n-i+1},$$
$$i=1,...,n.$$

The proof of Proposition 1 is provided in the Appendix. Proposition 1 gives us the set of extreme points of the set $\mathcal{P}(\varepsilon)\cap\mathcal{M}$, which are necessary for implementing the reduced boosting algorithm 2. The set $\mathcal{P}(\varepsilon)\cap\mathcal{M}$ is very interesting. Indeed, condition (10) produces a rather large set $\mathcal{M}$ of weights and its use may lead to overfitting because this set contains

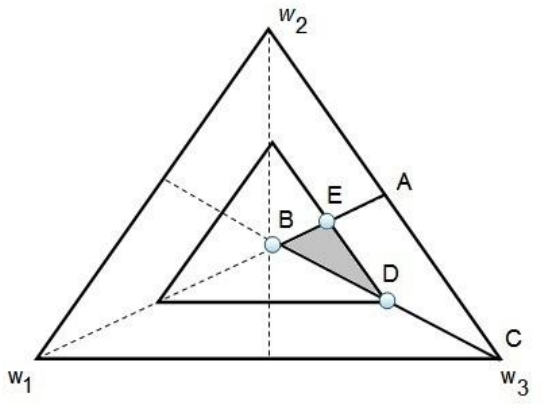

Figure 1: Intersection of sets of weights in the unit simplex for $n = 3$

many points with unit weights (see, for example, point C in Fig. 1), i.e., the regression is constructed by using one of the elements of the training set. It should be noted that one of the extreme points of $\mathcal{M}$ is $(0, ..., 0, 1)$. So, the use of $\mathcal{M}$ does not help us to solve the problem of overfitting. The set $\mathcal{P}(\varepsilon)$ itself has also some shortcomings. Indeed, its extreme points consist of identical elements except for a single element which is larger than other elements. We again construct the regression by using one of the elements of the training set. This may lead to overfitting. At the same time, the intersection of sets $\mathcal{P}(\varepsilon)$ and $\mathcal{M}$ is a set which has the best properties of both the sets. Let us consider the special cases. If $\varepsilon = 0$, then the set $\mathcal{P}(\varepsilon) \cap \mathcal{M}$ is reduced to the single point $(1/n, ..., 1/n)$, which corresponds to the standard non-parametric SRGM with identical weights of all elements of the training set. Another case is $\varepsilon = 1$. In this case, we get $\mathcal{P}(1) \cap \mathcal{M} = \mathcal{M}$, i.e., we use only comparative information. The algorithm implementing the RBoostSRGM is a modification of Algorithm 2, where n extreme points of $\mathcal{P}(\varepsilon) \cap \mathcal{M}$ denoted $q_1, ..., q_n$ are taken in accordance with Proposition 1.

5. An upper bounded imprecise model, Kolmogorov-Smirnov bounds and pairwise comparisons

Before considering the Kolmogorov-Smirnov bounds, we study the so-called upper bounded imprecise model which has been considered in detail in Utkin (2014). In contrast to many well-known imprecise statistical models, including Walley's imprecise Dirichlet model and the imprecise ε-contaminated model, the upper bounded model does not appear to have a clear statistical meaning. However, we will show that this model is closely linked to the Kolmogorov-Smirnov bounds (Johnson and Leone, 1964, Subsection 8.9.3.) which can be regarded as the best bounds for the empirical cumulative distribution. Moreover, the

Kolmogorov-Smirnov bounds have a strong statistical explanation and justification which can be used for constructing the proposed SRGM.

5.1. The upper bounded robust model

Let us consider a simple generalization of the vacuous model by introducing upper bounds for the probabilities $w_1, ..., w_n$, namely, we state that $w_i \le \vartheta$, $i = 1, ..., n$, where ϑ is some number between $1/n$ and 1. We denote the reduced set of probability distributions as $\mathcal{P}_U(\vartheta)$. The lower possible bound for ϑ is $1/n$ because the set $\mathcal{P}_U(\vartheta)$ becomes empty when $\vartheta < 1/n$. The set $\mathcal{P}_U(\vartheta)$ is produced by the following system of $2n$ inequalities and one equality:

$$0 \le w_i \le \vartheta,\ i = 1, ..., n,\ w_1 + ... + w_n = 1. \tag{11}$$

It is not difficult to find all extreme points of the set $\mathcal{P}_U(\vartheta)$, which strongly depend on the value ϑ.

Proposition 2. *Let k be an integer from 1 to $n-1$ such that the following condition is fulfilled:*

$$\frac{1}{n-k+1} < \vartheta \le \frac{1}{n-k}. \tag{12}$$

Then the set $\mathcal{P}_U(\vartheta)$ has $t = k \cdot \binom{n}{k}$ extreme points such that every extreme point consists of exactly $n-k$ elements ϑ, $k-1$ elements 0 and one element $1-(n-k)\vartheta$.

The proof of Proposition 2 can be found in (Utkin, 2014). It follows from condition (12) that the value k is determined from the condition:

$$n - \vartheta^{-1} \le k < n + 1 - \vartheta^{-1}.$$

5.2. Kolmogorov-Smirnov bounds

One of the ways for taking into account the amount of statistical data and for constructing bounds for the set of probability distributions is using the Kolmogorov-Smirnov confidence limits for the empirical cumulative distribution function $F_n(\mathbf{x})$ constructed on the basis of n observations. It is a quite different model which does not have anything common with the upper bounded robust model at first glance. However, we will see an interesting link between the models.

First, we briefly introduce the Kolmogorov-Smirnov confidence limits. Suppose that function $F(\mathbf{x})$ is a true probability distribution function of points from the training set, which is unknown. Then a critical value of the test statistic $d_{n,1-\gamma}$ can be chosen such that a band of width $\pm d_{n,1-\gamma}$ around $F_n(\mathbf{x})$ will entirely contain $F(\mathbf{x})$ with probability $1-\gamma$, which is to be interpreted as a confidence statement in the frequentist statistical framework. In other words, we can write $\Pr\{D_n \ge d_{n,1-\gamma}\} = \gamma$, where the quantity $D_n = \max_{\mathbf{x}} |F_n(\mathbf{x}) - F(\mathbf{x})|$ is called the Kolmogorov-Smirnov statistic. Denote the $(1-\gamma)$-quantile of the Kolmogorov distribution by $k_{1-\gamma}$. The ways for computing $d_{n,1-\gamma}$ for given n and γ as well as the values of $k_{1-\gamma}$ can be found in the book (Johnson and Leone,

1964, Subsection 8.9.3.). In particular, according to Johnson and Leone (1964), a good approximation for the test statistic for $n > 10$ is given by

$$d_{n,1-\gamma} \approx k_{1-\gamma}/\sqrt{n}.$$

For $n \leq 10$ another approximation can be used Johnson and Leone (1964):

$$d_{n,1-\gamma} \approx k_{1-\gamma}\left(\sqrt{n} + 0.12 + 0.11/\sqrt{n}\right)^{-1}.$$

Taking into account that the bounds are cumulative distribution functions, we write the following bounds $\underline{F}_n(\mathbf{x})$ and $\overline{F}_n(\mathbf{x})$ for some unknown distribution function $F(\mathbf{x})$:

$$\underline{F}_n(\mathbf{x}) \leq F(\mathbf{x}) \leq \overline{F}_n(\mathbf{x}), \tag{13}$$

where

$$\underline{F}_n(\mathbf{x}) = \max(F_n(\mathbf{x}) - d_{n,1-\gamma}, 0),$$
$$\overline{F}_n(\mathbf{x}) = \min(F_n(\mathbf{x}) + d_{n,1-\gamma}, 1).$$

It can be seen from the above inequality that the left tail of the upper probability distribution is $d_{n,1-\gamma}$. The right tail of the lower probability distribution is $1 - d_{n,1-\gamma}$. It has been shown by Utkin and Coolen (2011) that the largest value of the expected risk is achieved at the probability distribution which coincides with the lower Kolmogorov-Smirnov confidence limit. Moreover, it has been shown that this distribution has $k-1$ jumps of size 0, one jump of size $k/n - d_{n,1-\gamma}$ and the other jumps all of size $1/n$, where k is determined from the condition

$$d_{n,1-\gamma}n < k \leq d_{n,1-\gamma}n + 1.$$

It is assumed here that there are no coinciding points, i.e., $\mathbf{x}_j \neq \mathbf{x}_i$ for every $j \neq i$. This assumption can be relaxed for the case of coinciding points, in which case the number of jumps is reduced.

It is important to point out that Kolmogorov-Smirnov bounds use an assumption that there is a jump $1 - d_{n,1-\gamma}$ which is located at boundary points of the sample space far from all data points (see Utkin and Coolen (2011) for details). Therefore, in order to normalize the sizes of jumps at points $\mathbf{x}_1, ..., \mathbf{x}_n$, every size has to be divided by $1 - d_{n,1-\gamma}$. As a result, we get one jump of size

$$(k - nd_{n,1-\gamma})n^{-1}(1 - d_{n,1-\gamma})^{-1},$$

$n - k$ jumps of size

$$n^{-1}(1 - d_{n,1-\gamma})^{-1},$$

and $k - 1$ jumps of size 0.

Let us denote

$$\vartheta = n^{-1}(1 - d_{n,1-\gamma})^{-1}. \tag{14}$$

Then we get one jump equal to $1-(n-k)\vartheta$, $n-k$ jumps equal to ϑ, $k-1$ jumps of size 0. Moreover, the value ϑ fulfills condition (12). It follows from the above that we have obtained extreme points of the upper bounded vacuous model. In other words, the set of probability distributions produced by Kolmogorov-Smirnov bounds and by the upper bounded vacuous model coincide. This is an important feature which provides a way for interpretation and choice of the bound ϑ in terms of critical values of the test statistic $d_{n,1-\gamma}$.

The upper bounded model and the model using Kolmogorov-Smirnov bounds have another interesting property. Every extreme point has $k-1$ zero-valued elements. This implies that weights of $k-1$ points from the training set are 0, i.e., these points are not used in computing the optimal parameters. Of course, every point from the training set is used, but it is used by taking other extreme points. The number of the zero-valued points strongly depends on the total number of points n in the training set and on the parameter ϑ or the confidence probability $1-\gamma$.

5.3. Intersection of two sets

We consider the intersection of the sets of probabilities distributions produced by Kolmogorov-Smirnov bounds and pairwise comparisons. First of all, we consider the intersection of sets $\mathcal{P}_U(\vartheta)$ and $\mathcal{M}$ by using the standard unit simplex which contains all possible weight vector (w_1, w_2, w_3) for the case $n=3$. Fig. 2 illustrates two cases of the value ϑ. The first case (see the left simplex in Fig. 2) is when $\vartheta > 1/2$. The set $\mathcal{M}$ corresponds to the area restricted by the triangle ABC. The set $\mathcal{P}_U(\vartheta)$ corresponds to the area restricted by the hexagon with vertices $DEFGKI$. Their intersection is restricted by the tetragon $ABKL$ (the shaded area). The second case (see the right simplex in Fig. 2) is when $1/3 \le \vartheta \le 1/2$. The set $\mathcal{P}_U(\vartheta)$ corresponds to the area restricted by the small simplex with vertices DEF. The intersection of $\mathcal{P}_U(\vartheta)$ and $\mathcal{M}$ is restricted by the triangle AFG (the shaded area). In both cases, the weights used in the model are shifted towards the last (the third) vertex of the simplex. However, the set $\mathcal{P}_U(\vartheta)$ restricts this shift.

It follows from (14) that the parameter ϑ decreases as the number of observations n increases. One can see from Fig. 2 that the intersection of sets $\mathcal{P}_U(\vartheta)$ and $\mathcal{M}$ is reduced with decrease of ϑ or increase of n. It is also interesting to note that increase of n leads to increase of the dimensionality of the simplex of weight (we get a heigher dimensionality) and simultaneously leads to reduction of the set $\mathcal{P}_U(\vartheta)$. The limiting case when $n \to \infty$ reduces the set $\mathcal{P}_U(\vartheta)$ as well as the set $\mathcal{P}_U(\vartheta) \cap \mathcal{M}$ to the point $(1/n, ..., 1/n)$.

Proposition 3. *If we define k from (12), then a set of extreme points of the intersection $\mathcal{P}_U(\vartheta) \cap \mathcal{M}$ consists of $k(n-k+1)$ elements of the form:*

$$(\underbrace{0, ..., 0}_{k-s}, \underbrace{\frac{\varphi}{s+t}, ..., \frac{\varphi}{s+t}}_{s+t}, \underbrace{\vartheta, ..., \vartheta}_{n-k-t}), \tag{15}$$

where $\varphi = 1-(n-k-t)\vartheta$, $t = 0, ..., n-k$, $s = 1, ..., k$.

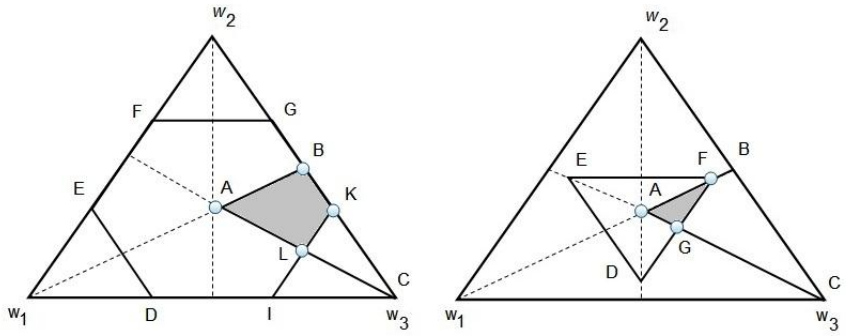

Figure 2: Intersections of sets of weights in the unit simplex for $n = 3$

The proof of Proposition 3 is provided in the Appendix. Let us consider the case $n = 3$ and $\vartheta = 0.8$ as an example. Then it follows from (12) that $k = 2$ and four extreme points can be written by using Proposition 3:

$$\begin{aligned}(\varphi/3, \varphi/3, \varphi/3,) &= (1/3, 1/3,\ 1/3),\\ (0, \varphi/2, \varphi/2,) &= (0, 1/2, 1/2),\\ (0, 1-\vartheta, \vartheta) &= (0, 0.2, 0.8),\\ ((1-\vartheta)/2, (1-\vartheta)/2, \vartheta) &= (0.1, 0.1, 0.8).\end{aligned}$$

These points correspond to points A, B, K, L in the left simplex depicted in Fig. 2. If we take $\vartheta = 0.4$, then $k = 2$ and we get three extreme points:

$$\begin{aligned}(\varphi/3, \varphi/3, \varphi/3,) &= (1/3, 1/3,\ 1/3),\\ (1-2\vartheta, \vartheta, \vartheta) &= (0.2, 0.4, 0.4),\\ ((1-\vartheta)/2, (1-\vartheta)/2, \vartheta) &= (0.1, 0.1, 0.8).\end{aligned}$$

These points correspond to points A, F, G in the right simplex depicted in Fig. 2.

Proposition 3 gives us the set of extreme points of the set $\mathcal{P}_U(\vartheta) \cap \mathcal{M}$, which will be used for solving $v = k(n-k+1)$ programming problems. Every optimization problem is defined by one of the extreme points $w^{(k)}$, $k = 1, ..., k(n-k+1)$. As a result, we have $k(n-k+1)$ standard weighted SVRs whose solutions do not meet any difficulties.

In contrast to the set $\mathcal{P}(\varepsilon)$, presented in Section 4, the set $\mathcal{P}_U(\vartheta)$ depends strongly on the number of training data. If the training set is small, then the Kolmogorov-Smirnov bounds produce a very large and non-informative interval which leads to very large values of ϑ. Both the sets $\mathcal{M}$ and $\mathcal{P}_U(\vartheta)$ compensate each other. Their intersection allows us to avoid some extreme cases when only small part of training data determines the SRGM. It should be also noted that the point $(1/n, ..., 1/n)$, corresponding to the standard approach accepted in the non-parametric SRGMs, belongs to the set $\mathcal{P}_U(\vartheta) \cap \mathcal{M}$. This implies that reliability measures of the software reliability modelling, obtained by means of the standard non-parametric SRGM on the basis of the empirical expected risk measure, belong to the intervals of the reliability measures obtained by means of the proposed model. By using the Kolmogorov-Smirnov bounds, this model allows statements about the software reliability with a predefined level of confidence $\gamma = \Pr\{D_n \geq d_{n,1-\gamma}\}$.

6. Numerical experiments

The illustration of our new method and the investigation of its performance is mostly done using the model presented in Section 4. We report in less detail on the performance of the model presented in Section 5 at the end of this section. For all data we split the data set into two subsets. One of them (training set having $n - n_{test}$ examples) is used to train the model while the other (test set having n_{test} examples) is used to validate the model. We use about 20% of examples for testing, so n_{test} is about equal to $0.2n$, and these test examples are the final data points in the data set, so with labels $n - n_{test} + 1, ..., n$. Every regressor is realized by means of the weighted SVR (Support Vector Regression) with parameter $\epsilon = 0$ of the so-called ϵ-insensitive loss function (for details see Smola and Scholkopf (2004)).

Since the purpose of these examples is mainly to show the application of the method on simple and easy to visualize problems, the hyperparameters are chosen without fine tuning. For all performed experiments, we quantify the prediction performance with the normalized root mean square error measure (MSE), which is defined as

$$MSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2}{n_{test}}}.$$

Here y_i, $f(\mathbf{x}_i)$ are the actual value and the forecasted value, respectively. We will also use the relative absolute difference between MSEs (RAMSE), which is defined as

$$RAMSE = \left| MSE_{\varepsilon=0} - \min_{\varepsilon} MSE_{\varepsilon} \right| / \min_{\varepsilon} MSE_{\varepsilon} \times 100.$$

Here we have to point out that the MSE depends on the parameter ε.

All experiments use a standard Gaussian radial basis function (RBF) kernel with the kernel parameter σ. Different values for the kernel parameter σ and the cost parameter C have been tested, choosing those which lead to the best results. This procedure is

realized by considering all possible values of σ and C in a predefined grid. The grid for σ is determined as 2^v, where $v = -15, ..., 0, ..., 15$. The values of C are taken in accordance with the expression $C_0 + iC_s$, where C_0 and C_s are experimentally determined parameters, $i = 1, ..., 40$.

6.1. Data sets

The proposed RBoostSRGM has been evaluated and investigated by application to the following publicly available data sets:

1. The first data set is the software inter-failure times y_i taken from a telemetry network system by AT&T Bell Laboratories, published by Pham and Pham (2000). The data set contains 22 observations of the actual time series.
2. The second data set is obtained from Pham and Pham (2001) and contains 101 observations of the software inter-failure time series.
3. The third data set is reported by Ohba (1984) and is recorded from testing an on-line data entry software package developed at IBM. The data set contains 15 observations.
4. The fourth failure data set (NTDS - failure data) was first reported by Jelinski and Moranda (1972) and contains 34 failure data. The set can be also found in (Pham, 2006, Subsection 4.8).
5. The fifth data set is reported by Musa et al (1987) and evaluated by Liu and Xu (2011) (JDM-I failure data). The set contains 17 observations of the software inter-failure time series.
6. The sixth data set is also reported by Musa et al (1987) and evaluated by Liu and Xu (2011) (JDM-II failure data). The set contains 15 observations of the software inter-failure time series.
7. The seventh data set is reported by Kanoun et al (1991) and is called TUSER. The set contains 26 observations of the software inter-failure time series.
8. The eighth data set called TDOC is also obtained from Kanoun et al (1991). It consists of 34 observations.

6.2. Numerical results

First, we investigate the proposed model by using the first data set. Fig.3 presents an example of the fault detection prediction results with three SRGMs, the proposed RBoost-SRGM (the thick curve - 1) by $\varepsilon = 0.2$, the boosting SRGM (the thin curve - 2) which is a special case of the proposed RBoostSRGM with $\varepsilon = 1$, and the standard non-parametric SRGM based on the SVR (the middle curve - 3), together with the actual results (dashed curve). It can be seen from Fig. 3 that the observation data have visible heteroscedasticity, i.e., changes of variance especially after the 15-th failure detection. It is obvious in this case that observations from the beginning of the debugging process should not strongly influence the later predictions. The training data are depicted by triangle markers, the testing data are depicted by circle markers. One can also see from Fig. 3 that all three

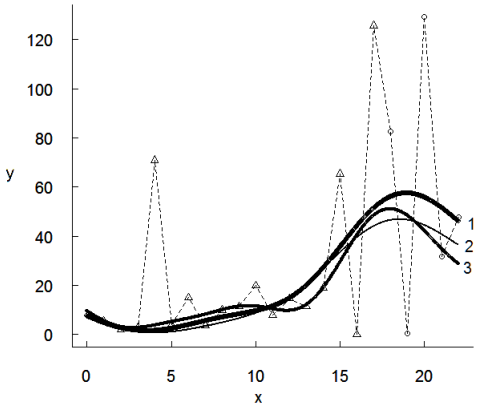

Figure 3: Fault detection prediction results with three SRGMs and actual results for the first data set

models behave differently especially at the testing period where the large variation of times to failure is observed.

We investigate how the prediction performance depends on the contamination parameter ε which takes values from 0 to 1. The case $\varepsilon = 0$ corresponds to the standard non-parametric SRGM based on the SVR, the case $\varepsilon = 1$ corresponds to the boosting SRGM reducing only by the set $\mathcal{M}$. Moreover, we also compare the above models with the standard boosting SRGM (Vamsidhar and Raju, 2011) without reduction of the unit simplex as the weight space. Dependence of the MSE on the parameter ε is shown in Fig. 4. The solid curve with triangle markers corresponds to the MSE obtained by using the RBoostSRGM. The dashed line shows the MSE for $\varepsilon = 0$, i.e., for the standard non-parametric SRGM. The dotted line illustrates the MSE for $\varepsilon = 1$, i.e., for the standard boosting SRGM. The solid curve with circle markers illustrates the MSE of the standard boosting SRGM. We will use these lines in all data set analyzed.

The dotted and dashed lines almost coincide here, i.e., the boosting SRGM reducing by the set $\mathcal{M}$ does not outperform the standard SRGM for the first data set. At the same time, one can see that the solid curve achieves its smallest value at $\varepsilon = 0.2$, i.e., the highest accuracy can be obtained by taking $\varepsilon = 0.2$. The largest RAMSE is about 9%.

By considering the second data set, we see that the best results are provided by the standard boosting SRGM. Moreover, the RBoostSRGM increases the MSE (see Fig. 6). This is because the training set contains many elements. Moreover, if we look at the data set (see Fig. 5), then we can find that the observations are characterized by large variety and it is difficult to distinguish different debugging stages from their importance point of view. Therefore, the proposed model does not provide the best performance.

The corresponding dependences of the MSE of the RBoostSRGM, the standard SRGM, the boosting SRGM reduced by $\mathcal{M}$ and the standard boosting SRGM on the parameter ε obtained for all data sets are shown in Figs. 6-12. The optimal values of the parameter ε

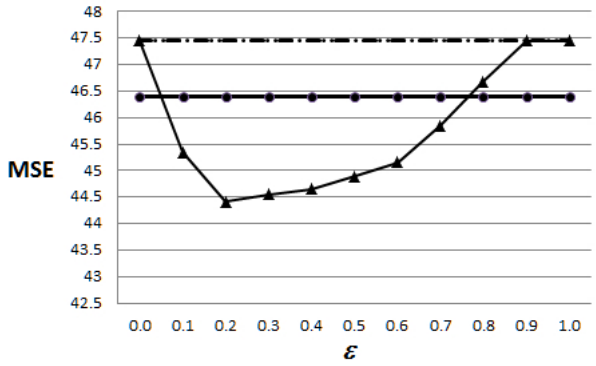

Figure 4: Dependence of the MSE on the parameter ε for the first data set

Table 1: Numerical results with real data sets

Data Set	Figure	n	Optimal parameter ε	The largest RAMSE (%)
1	4	22	0.2	9
2	6	101	1	0
3	7	15	0.4	28.8
4	8	34	0.4	4.3
5	9	17	0.3	7.3
6	10	15	0.7	27.2
7	11	26	0.7	4.5
8	12	34	0.4	0.6

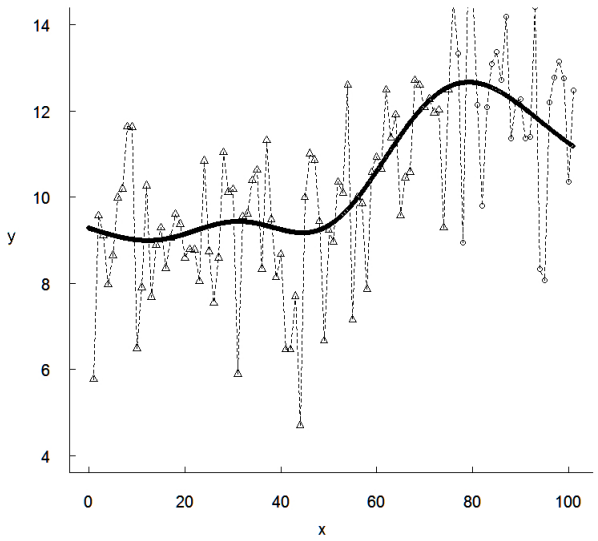

Figure 5: Fault detection prediction results with the proposed SRGM and actual results for the second data set

the RBoostSRGM and the largest RAMSE for every data set are given in Table 1.

It is interesting to point out how the largest RAMSE depends on the number n of training examples. One can see from Table 1 that the RAMSE decreases as n increases. This implies that the proposed model can be successful in comparison with the available SRGMs when the number of observations is rather small. Of course, there are other factors which impact on the model prediction accuracy, including the parameters of models, for example, ε for the imprecise ε-contaminated model or γ for the Kolmogorov-Smirnov bounds. Another one of the important factors is the peculiarity of a dataset analyzed. For example, the fourth and the eighth datasets have the same number of observations, but quite different accuracy measures. This implies that their structure is differently adopted for the proposed model. In most cases, the number of observations is an important factor.

Next we consider the application of the Kolmogorov-Smirnov bounds and pairwise comparisons, as presented in Section 5, for constructing the boosting SRGM. In order to apply the Kolmogorov-Smirnov bounds, we have to define the critical value of the test statistic $d_{n,1-\gamma}$ as a function of number n of training data and the confidence level $1-\gamma$. Let us take $\gamma = 0.1$. According to Johnson and Leone (1964, Subsection 8.9.3.), the 0.9-quantile of the Kolmogorov distribution $k_{1-\gamma}$ is equal to 1.22 in this case. The computed values of $d_{n,1-\gamma}$ are given in Table 2. It should be noted that they are computed by taking into account only training data, i.e., $n - n_{test}$ which is about equal to $0.8n$ examples. The corresponding MSE measures are also shown in Table 2. The RAMSE measure (see the fifth column of Table 2) is computed as the relative absolute difference between the MSE for our method

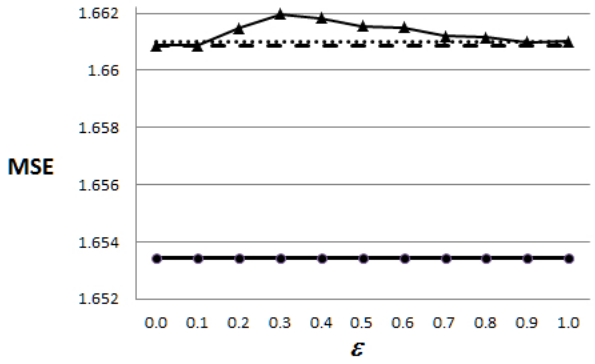

Figure 6: Dependence of the MSE on the parameter ε for the second data set

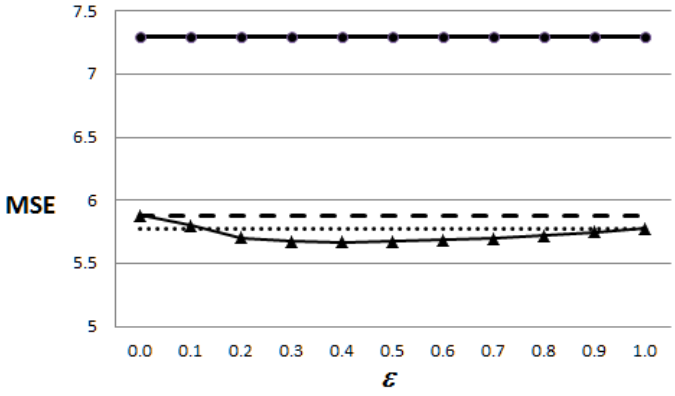

Figure 7: Dependence of the MSE on the parameter ε for the third data set

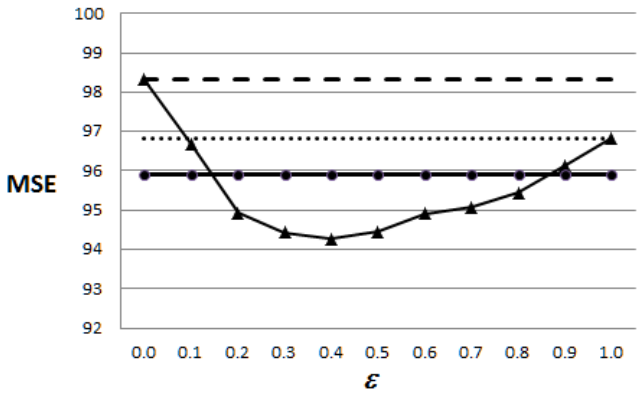

Figure 8: Dependence of the MSE on the parameter ε for the fourth data set

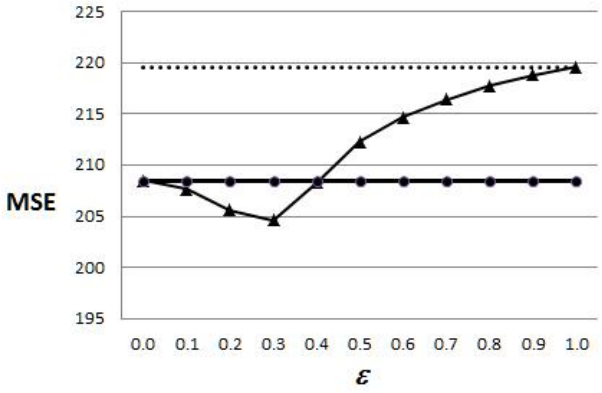

Figure 9: Dependence of the MSE on the parameter ε for the fifth data set

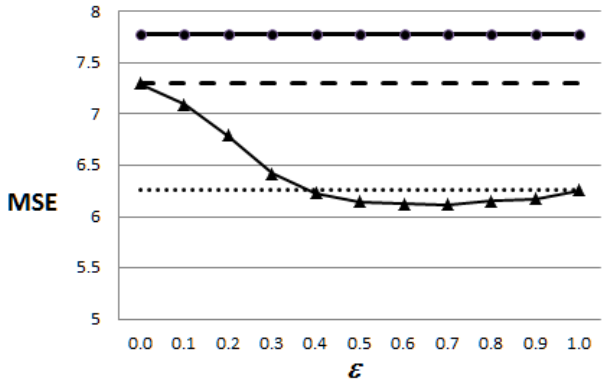

Figure 10: Dependence of the MSE on the parameter ε for the sixth data set

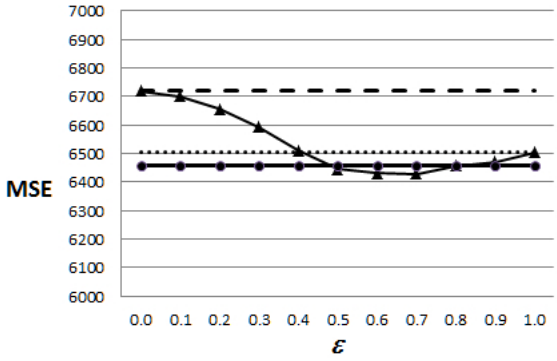

Figure 11: Dependence of the MSE on the parameter ε for the seventh data set

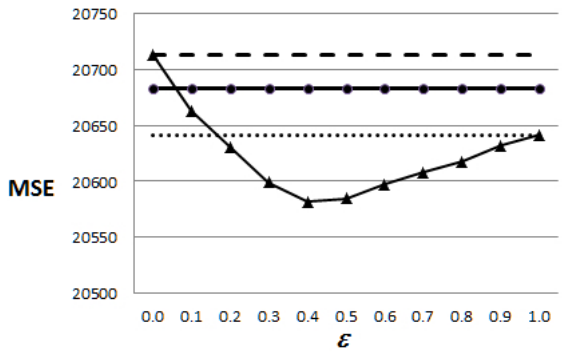

Figure 12: Dependence of the MSE on the parameter ε for the eighth data set

Table 2: Numerical results with real data sets by using the Kilmogorov-Smirnov bounds

Data Set	n	$d_{n,1-\gamma}$	MSE	RAMSE (%)
1	22	0.281	43.9	8.1
2	101	0.134	1.638	1.4
3	15	0.337	5.784	26.2
4	34	0.228	92.91	5.8
5	17	0.318	200.6	9.5
6	15	0.337	6.245	24.6
7	26	0.259	6227	7.9
8	34	0.228	19897	4.1

and the MSE of the standard non-parametric SRGM obtained in the previous numerical experiments.

If we compare Tables 1 and 2, we can see that the results are very similar. We can point out that the RAMSEs by using the Kolmogorov-Smirnov bounds are larger in comparison with the RBoostSRGM using the imprecise ε-contaminated model for large data sets. However, small data sets illustrate opposite results which can be explained by the fact that the optimal value in the imprecise ε-contaminated model is chosen, but the Kolmogorov-Smirnov bounds have the predefined parameter $\gamma = 0.1$. At the same time, we can see from Tables 1 and 2 that the behaviour of the RAMSEs for the Kolmogorov-Smirnov bounds and the imprecise ε-contaminated model are similar. However, it should be noted that the model based on the Kolmogorov-Smirnov bounds has a very important advantage in comparison with the imprecise ε-contaminated model. It is explainable from statistical point of view and this model has a strong interpretation based on the test statistic $d_{n,1-\gamma}$ as a function of the training data number n.

7. Conclusion

In this paper we have presented a new software reliability growth model, called RBoost-SRGM, which can be viewed as a modification of the boosting SRGMs using a reduced set of weights in order to take into account the behaviour of the software reliability during the debugging process. We have used the main idea that training data from the later stages of the debugging process may be more important for software reliability prediction than data from the early stages of the debugging process. This leads to assigning increasing weights to the elements of training data from the time series consisting of consecutive times to software failures. The second important idea used in the paper is that we do not assign precise weights but we apply a set of weights produced by the comparative information about training data elements. Moreover, we avoid getting a very large set of "comparative" weights by use of the set produced by the imprecise ε-contaminated model. As a result, we have obtained an interesting intersection of sets, which can be used as the reduced set of weights in the boosting algorithm.

In fact, the obtained RBoostSRGM is a parametric model because it is tuned in accordance with the parameters of the SVR (C and σ) and with the contamination parameter ε. The last parameter is very important because the above numerical experiments show that in each application there is an optimal value of ε providing the best results.

We have applied the well-known AdaBoost.R2 algorithm for modification. However, there are other boosting regression algorithms whose use can be regarded as a direction for future research. Moreover, we have applied both the imprecise ε-contaminated model and the Kolmogorov-Smirnov bounds for restricting the set of "comparative" weights. Other imprecise models could also be used here, for example Walley's imprecise pari-mutuel and constant odds-ratio models (Walley, 1991). General guidance on the choice of a suitable imprecise model providing optimal results is another direction for future research.

Our newly proposed method does not perform best for all considered data sets. It can be seen from Table 1 that the relative absolute difference between MSEs is not so large. Nevertheless, the proposed idea of reducing the set of weights in the boosting method in a specific manner may can be regarded as an opportunity for developing new classes of boosting-based models.

Appendix: Proofs

Proof of Proposition 1.

Denote the set of inequalities (9) by P and the set of inequalities (10) by M. We do not include into P inequalities $w_i \geq 0$ because all points with $w_i = 0$ at least for one i do not belong to the intersection $\mathcal{P}(\varepsilon) \cap \mathcal{M}$. Let us consider the system of $2n-1$ inequalities from P and M. It is know that every extreme point satisfies $n-1$ equalities from $P \cap M$. We study the following cases.

Case 1. All $n-1$ equalities are from P. It is obvious that $w_1 = ... = w_n = 1/n$.

Case 2. $n-2$ equalities are from P, and 1 constraint is from M. This implies that there is one strong inequality $w_{i-1} < w_i$ from P and one equality $w_k = n^{-1} - \varepsilon n^{-1}$ from M. Here we have to consider two subcases. The first subcase is $k \geq i$. Then

$$w_{i-1} < w_i = w_k = n^{-1} - \varepsilon n^{-1}.$$

However, $w_i = w_{i+1} = ... = w_n$. Hence, $w_1 + ... + w_n = 1 - \varepsilon < 1$. We get a contradiction. Therefore, this subcase does not give extreme points.

The second subcase is $k < i$. Then

$$w_1 = w_2 = ... = w_{i-1} = \frac{1}{n} - \frac{\varepsilon}{n}.$$

Then there holds

$$w_i + ... + w_n = 1 - \left(n^{-1} - \varepsilon n^{-1}\right) i.$$

Hence, we get

$$w_i = \frac{1 - \left(n^{-1} - \varepsilon n^{-1}\right) i}{n - i + 1} = \frac{1}{n} + \frac{\varepsilon}{n} \cdot \frac{i-1}{n-i+1}.$$

This is an extreme point for a fixed i. Note that the extreme point obtained in Case 1 can be regarded as a special case of Case 2 when $i = 1$.

Case 3. $n-3$ equalities are from P, and 2 constraints are from M. This implies that there are two strong inequalities $w_{i-1} < w_i$ and $w_{j-1} < w_j$, $i < j$, from P and two equalities $w_k = n^{-1} - \varepsilon n^{-1}$ and $w_l = n^{-1} - \varepsilon n^{-1}$, $k < l$. The case $k \geq i$ and $l \geq i$ is not considered here (see the similar case above). Suppose that $k < i$ and $l < i$. Then we can write

$$w_1 = w_2 = ... = w_{i-1} = \frac{1}{n} - \frac{\varepsilon}{n}.$$

Suppose that $w_s = a$, $s = i, ..., j-1$, and $w_s = b$, $s = j, ..., n$, i.e.,

$$w_i = ... = w_{j-1} = a, \quad w_j = ... = w_n = b.$$

Here due to inequalities $w_{i-1} < w_i$ and $w_{j-1} < w_j$, we can write

$$\frac{1}{n} - \frac{\varepsilon}{n} < a < b.$$

The numbers a and b satisfy the following obvious condition:

$$\left(\frac{1}{n}-\frac{\varepsilon}{n}\right)(i-1)+a(j-i)+b(n-j+1)=1.$$

It can be seen that there are infinitely many values of a and b satisfying the above condition. This implies that we get an edge of the corresponding polytope. The same can be obtained for other cases when we take $n-r$ equalities from P, and $r-1$ constraints are from M. Consequently, Case 2 totally defines all extreme points, as was to be proved.

Proof of Proposition 3.

It is simple to prove that an extreme point of a set, which belongs to another set, is an extreme point of the intersection of these sets. Therefore, we first consider the extreme points of $\mathcal{M}$ which belong to $\mathcal{P}_U(\vartheta)$. It is obvious that extreme points of $\mathcal{M}$, having the form

$$(\underbrace{0,...,0}_{n-r},\underbrace{r^{-1},...,r^{-1}}_{r}),$$

belong to the set $\mathcal{P}_U(\vartheta)$ if $w_i \le \vartheta$ (see the definition of the set $\mathcal{P}_U(\vartheta)$). This implies that $r^{-1} \le \vartheta$. Hence $r \ge \vartheta^{-1}$. This provides the first subset of extreme points.

We now consider extreme points of $\mathcal{P}_U(\vartheta)$ belonging to $\mathcal{M}$. In order to ensure that a point belongs to $\mathcal{M}$, we select extreme points of $\mathcal{P}_U(\vartheta)$ whose elements do not decrease with increase of the element index. Since every extreme point of $\mathcal{P}_U(\vartheta)$ consists of exactly $n-k$ elements ϑ, $k-1$ elements 0 and one element $\varphi = 1-(n-k-t)\vartheta$ for $t=0$, then we can write only one extreme point

$$\underbrace{0,...,0}_{k-1},\varphi,\ \underbrace{\vartheta,...,\vartheta}_{n-k}.$$

Indeed, we can write $1-(n-k)\vartheta < \vartheta$. This inequality follows from the condition for k which is $\vartheta > 1/(n-k+1)$. Therefore, the element $1-(n-k)\vartheta$ cannot be located in another place and we have one extreme point.

Let us consider now extreme points produced by the intersection of sets. Suppose that $n-k-t$ elements of one of the extreme points of $\mathcal{P}_U(\vartheta)\cap\mathcal{M}$ correspond to the equalities $w_i=\vartheta$. Here $t=1,...,n-k-1$. It is obvious that the indices of these elements are $i=k+t+1,...,n$, because $w_i \le \vartheta$ and $w_{i-1} \le w_i$. We also suppose that $k-s$ elements of one of the extreme points of $\mathcal{P}_U(\vartheta)\cap\mathcal{M}$ correspond to the equalities $w_i=0$. Here s is some integer. The indices of these elements are $i=1,...,k-s$. Then elements with indices $i=k-s+1,...,k+t$ are $(1-(n-k-t)\vartheta)/(s+t)$.

Let us prove that the points with arbitrary elements $w_{k-s+1}=\alpha_1,...,w_k=\alpha_s$ satisfying all conditions (10) and (11), i.e., the point

$$(\underbrace{0,...,0}_{k-s},\underbrace{\alpha_1,...,\alpha_{s+t}}_{s+t},\underbrace{\vartheta,...,\vartheta}_{n-k-t}),$$

can be obtained as a linear combination of two extreme points:

$$(\underbrace{0,...,0}_{k-s},\underbrace{0,...,0,\varphi}_{s+t},\ \underbrace{\vartheta,...,\vartheta}_{n-k-t}),$$

and

$$(\underbrace{0,...,0}_{k-s},\underbrace{\varphi/(s+t),...,\varphi/(s+t)}_{s+t},\ \underbrace{\vartheta,...,\vartheta}_{n-k-t}).$$

On the one hand, it follows from the normalization condition of weights that $\alpha_1 + ... + \alpha_{s+t} = \varphi$. On the other hand, we can write

$$\alpha_i = \gamma\varphi/(s+t),\ i = 1,...,s+t-1,\ \alpha_{s+t} = \gamma\varphi/(s+t) + (1-\gamma)\varphi.$$

Here γ is in $[0,1]$. Hence, the sum of α_i, $i = 1,...,s+t$, is

$$(s+t-1)\gamma\varphi/(s+t) + \gamma\varphi/(s+t) + (1-\gamma)\varphi = \varphi.$$

The above implies that $\alpha_1,...,\alpha_{s+t}$ can be obtained as the linear combination of the corresponding elements of two extreme points and the points with these elements belong to an edge of the corresponding polytope produced by the intersection $\mathcal{P}_U(\vartheta) \cap \mathcal{M}$. It should be noted that a point with $n-k+t$ elements ϑ does not exist if $t \geq 1$. Indeed, it follows from the condition for ϑ that $(n-k+1)\vartheta > 1$. Then the sum of all elements $(n-k+t)\vartheta > 1$ by $t \geq 1$.

We can see that the points

$$(\underbrace{0,...,0}_{n-r},\underbrace{r^{-1},...,r^{-1}}_{r}),$$

belong to the set of points (15). Indeed, if $t = n-k$, then $\varphi = 1$. It is easy to prove that $r = n-k+s$. Moreover, it can be seen that the point

$$\underbrace{0,...,0}_{k-1},\varphi,\ \underbrace{\vartheta,...,\vartheta}_{n-k}$$

belongs to the set of points (15) too.

In summary, for every t from the set $\{0,...,n-k\}$ we have k extreme points corresponding to $s = 1,...,k$. Hence, we can write that the total number v of extreme points of the form (15) is $v = k(n-k+1)$.

Acknowledgement

The presentation of this paper has been improved by valuable suggestions from an associate editor and a reviewer, for which we are grateful. Part of the research reported in this paper took place during a visit of Prof. Utkin to Durham University in November 2016, funded through a 'Research in Pairs' (Scheme 4) grant from the London Mathematical Society.

References

A. Amina, L. Grunskeb, and A. Colman. Anvapproach to software reliability prediction based on time series modeling. *The Journal of Systems and Software*, 86:1923–1932, 2013.

S. Brocklehurst, K. Kanoun, J.-C. Laprie, B. Littlewood, S. Metge, P. Mellor, and A. Tanner. Analyses of software failure data. Technical Report 91173, The Centre for Software Reliability, City University, London, UK and LAAS-CNRS, Toulouse, France, May 1991.

K.-Y. Cai, L. Cai, W.-D. Wang, Z.-Y. Yu, and D. Zhang. On the neural network approach in software reliability modeling. *The Journal of Systems and Software*, 58:47–62, 2001.

K.Y. Cai, C.Y. Wen, and M.L. Zhang. A critical review on software reliability modeling. *Reliability Engineering & System Safety*, 32:357–371, 1991.

B.P. Carlin and T.A. Louis. *Bayesian Methods for Data Analysis*. CRC Press, Boca Raton, 2008.

K.-C. Chiu, Y.-S. Huang, and T.-Z. Lee. A study of software reliability growth from the perspective of learning effects. *Reliability Engineering & System Safety*, 93:1410–1421, 2008.

E.O. Costa, G.A. de Souza, A.T.R. Pozo, and S.R. Vergilio. Exploring genetic programming and boosting techniques to model software reliability. *IEEE Transactions on Reliability*, 56:422–434, 2007.

M. das Chagas Moura, E. Zio, I.D. Lins, and E. Droguett. Failure and reliability prediction by support vector machines regression of time series data. *Reliability Engineering & System Safety*, 96:1527–1534, 2011.

C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pp 180–189, San Francisco, 2000. Morgan Kaufmann.

H. Drucker. Improving regressor using boosting techniques. In *Proc. of the 14th Intenational Conferences on Machine Learning*, pages 107–115, San Francisco, CA, USA, 1997. Morgan Kaufmann.

P.C. Fishburn. Preference relations and their numerical representations. *Theoretical Computer Science*, 217:359–383, 1999.

Y. Freund and R.E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

A.L. Goel and K. Okomoto. Time dependent error detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, R-28:206–211, 1979.

P. Govindasamy and R. Dillibabu. Development of software reliability models using a hybrid approach and validation of the proposed models using big data. *The Journal of Supercomputing*, 1–14, 2018.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York, 2001.

C.J. Hsu and C.Y. Huang. Reliability analysis using weighted combinational models for web-based software. In *WWW 2009*, pp. 1131–1132, Madrid, Spain, April, 20-24 2009. ACM.

Q.P. Hua, M. Xie, S.H. Ng, and G. Levitin. Robust recurrent neural network modeling for software fault detection and correction prediction. *Reliability Engineering & System Safety*, 92:332–340, 2007.

A. Jaiswal and R. Malhotra. Software reliability prediction using machine learning techniques. *International Journal of System Assurance Engineering and Management*. 9(1):230–244, 2018.

Z. Jelinski and P.B. Moranda. Software reliability research. In W. Greiberger, editor, *Statistical Computer Performance Evaluation*, pp. 464–484. Academic Press, New York, 1972.

C. Jin and S.-W. Jin. Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. *Applied Soft Computing*, 15:113–120, 2014.

N.L. Johnson and F. Leone. *Statistics and Experimental Design in Engineering and the Physical Sciences*, volume 1. Wiley, New York, 1964.

T. Kim, K. Lee, and J. Baik. An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm. *Journal of Systems and Software*, 102:134–144, 2015.

N. Kumar and S. Banerjee. Measuring software reliability: A trend using machine learning techniques. In Quanmin Zhu and Ahmad Taher Azar, editors, *Complex System Modelling and Control Through Intelligent Soft Computations*, volume 319 of *Studies in Fuzziness and Soft Computing*, pp. 807–829. Springer, 2015.

P. Kumar and Y. Singh. An empirical study of software reliability prediction using machine learning techniques. *International Journal of System Assurance Engineering and Management*, 3:194–208, 2012.

H.-F. Li, M.-Y. Lu, M. Zeng, and B.-Q. Huang. A non-parametric software reliability modeling approach by using gene expression programming. *Journal of Information Science and Engineering*, 28:1145–1160, 2012.

H.F. Li, M. Zeng, M. Lu, X. Hu, and Z. Li. Adaboosting-based dynamic weighted combination of software reliability growth models. *Quality and Reliability Engineering International*, 28:67–84, 2012.

S. Lohmor and B.B. Sagar. A comprehensive review on software reliability growth models utilizing soft computing approaches. In *The International Symposium on Intelligent Systems Technologies and Applications*. Springer, pp. 509–523, 2018

B. Littlewood, M. Lu, and S. Brocklehurst. Combination of predictions obtained from different software reliability growth models. *Journal of Computer and Software Engineering*, 1:303–324, 1993.

J. Liu and M. Xu. Function based nonlinear least squares and application to Jelinski–Moranda software reliability model. *arXiv preprint arXiv:1108.5185*, 2011.

J. Lou, J. Jiang, C. Shuai, and Y. Wu. A study on software reliability prediction based on transduction inference. In *Test Symposium (ATS), 2010 19th IEEE Asian*, pp. 77–80, Shanghai, 2010.

M.R. Lyu. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, 1996.

J.D. Musa, A. Iannino, and K. Okumoto. *Software Reliability: Mesurement, Prediction, Application*. McGraw-Hill, New York, 1987.

M. Nakamura, H. Nomiya, and K. Uehara. Improvement of boosting algorithm by modifying the weighting rule. *Annals of Mathematics and Artificial Intelligence*, 41:95–109, 2004.

M. Ohba. Software reliability analysis models. *IBM Journal of Research and Development*, 28:428–443, 1984.

P.-F. Pai and W.-C. Hong. Software reliability forecasting by support vector machines with simulated annealing algorithms. *Journal of Systems and Software*, 79:747–755, 2006.

H. Pham. *System Software Reliability*. Springer, London, 2006.

L. Pham and H. Pham. Software reliability models with time-dependent hazard function based on Bayesian approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 30:25–35, 2000.

L. Pham and H. Pham. A Bayesian predictive software reliability model with pseudo-failures. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 31:233–238, 2001.

L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33:1–39, 2010.

P. Roy, G.S. Mahapatra, P. Rani, S.K. Pandey, and K.N. Dey. Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction. *Applied Soft Computing*, 22:629–637, 2014.

B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, Massachusetts, 2002.

R.A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

A.J. Smola and B. Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

Y. Sun, L. Wang, X. Wang and X. Zhao. Reliability growth prediction method based on GA-Elman neural network. In *The 12th International Conference on Reliability, Maintainability, and Safety (ICRMS)*, Shanghai, China, pp. 135-141, 2018.

F.E.H. Tay and L.J. Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48:847–861, 2002.

L. Tian and A. Noore. Dynamic software reliability prediction: an approach based on support vector machines. *International Journal of Reliability, Quality and Safety Engineering*, 12:309–321, 2005.

A.N. Tikhonov and V.Y. Arsenin. *Solution of Ill-Posed Problems*. W.H. Winston, Washington DC, 1977.

L.V. Utkin. An imprecise boosting-like approach to classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 27:1–24, 2013.

L.V. Utkin. A framework for imprecise robust one-class classification models. *International Journal of Machine Learning and Cybernetics*, 5:379–393, 2014.

L.V. Utkin. The imprecise Dirichlet model as a basis for a new boosting classification algorithm. *Neurocomputing*, 151:1374–1383, 2015.

L.V. Utkin and F.P.A. Coolen. On reliability growth models using Kolmogorov-Smirnov bounds. *International Journal of Performability Engineering*, 7:5–19, 2011.

L.V. Utkin and F.P.A. Coolen. Classification with support vector machines and Kolmogorov-Smirnov bounds. *Journal of Statistical Theory and Practice*, 8:297–318, 2014.

L.V. Utkin and A. Wiencierz. Improving over-fitting in ensemble regression by imprecise probabilities. *Information Sciences*, 317:315–328, 2015.

Y. Vamsidhar and P.S. Raju. Comparison of software reliability growth models by using adaboosting algorithm. *International Journal of Computer Science & Technology*, 2:18–21, 2011.

Y. Vamsidhar, P.S. Raju, and T.R. Kumar. Performance analysis of reliability growth models using supervised learning techniques. *International Journal of Scientific & Technology Research*, 1:1–7, 2012.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

F. Xing and P. Guo. Support vector regression for software reliability growth modeling and prediction. In *Advances in Neural Networks – ISNN 2005*, volume 3496 of *Lecture Notes in Computer Science*, pp. 925–930. Springer, 2005.

F. Xing, P. Guo, and M.R. Lyu. A novel method for early software quality prediction based on support vector machine. In *Proccedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE 2005)*, pp. 1–10. IEEE, 2005.

B. Yang, X. Li, M. Xie, and F. Tan. A generic data-driven software reliability model with model mining technique. *Reliability Engineering & System Safety*, 95:671–678, 2010.