Machine learning methods and asymmetric cost function to estimate execution effort of software testing

Daniel G. e Silva∗, Mario Jino† and Bruno T. de Abreu‡

∗†School of Electrical and Computer Engineering
State University of Campinas - UNICAMP, Campinas, SP, Brazil
E-mail: danielgs@dca.fee.unicamp.br, jino@dca.fee.unicamp.br†
‡Sofist - Intelligent Software Testing
Estrada da Telebras km 0.97, Campinas, SP, Brazil
E-mail: bruno.abreu@sofist.com.br

Abstract—Planning and scheduling of testing activities play an important role for any independent test team that performs tests for different software systems, developed by different development teams. This work studies the application of machine learning tools and variable selection tools to solve the problem of estimating the execution effort of functional tests. An analysis of the test execution process is developed and experiments are performed on two real databases. The main contributions of this paper are the approach of selecting the significant variables for database synthesis and the use of an artificial neural network trained with an asymmetric cost function.

Keywords—software testing; prediction; estimate; effort; neural networks; asymmetric function.

I. INTRODUCTION

Independent Software Verification & Validation (ISV&V) is an engineering practice devised to improve quality and reduce costs of a software product [1]. This practice also helps to reduce development risks by having an organization independent of the software development team performing verification and validation on software specification and source code.

From the definition of ISV&V, an Independent Test Team (ITT) consists of a team of test engineers exclusively focused on test planning, design, execution and result analysis. The team tests software systems from different development teams to find their faults, which are reported back to the development teams.

An ITT requires a proper management in order to make decisions such as when the test cycles will end and how many testers will be involved. Correct planning of test process is a key task that impacts the development schedule of all software systems under ITT’s analysis. A recent research [2] in Australia with 65 companies has shown that nearly 68% of them have ITTs, confirming the activity’s growing importance in software process and the increasing demand for support tools to manage the test process.

Machine learning techniques have been successfully used to solve several software engineering challenges, including the effort estimation problem. Specifically, recent papers have addressed test effort estimation with different methods such as support vector regression [3] and metrics based approaches [4], [5], [6]. However, while there are many studies on effort and cost estimation techniques for software development [7], [8], research in test effort estimation is a relatively open field, with opportunities for simple and efficient tools that estimate effort to execute a test case suite.

In this work we study the application of artificial neural networks and support vector regression to estimate execution effort of functional tests. After an analysis of the test process, we concluded that there are larger impacts on software quality when effort is underestimated. In order to cope with this, we have developed a modified cost function to train an artificial neural network, aiming to bias the predictive model to overestimate, instead of underestimating.

Two real case studies are conducted to observe and compare each model’s performance. A variable selection process is performed to improve data quality and, consequently, reduce the effort of training and estimation activities. The main purpose of this work is to compare the methods’ efficacy and the feasibility of results.

This paper is organized as follows: Section II presents related work on test effort estimation; Section III presents basic definitions about the machine learning and variable selection tools that are considered in the paper; Section IV introduces the asymmetric cost function; Section V presents the case studies, followed by our conclusions and future improvements in our work.

II. RELATED WORK

Despite the recent interest on research in test effort estimation, there are already important studies on the problem. Different theories and methods have been used, always attempting to maintain quality of estimates and taking into consideration practical applicability.

NAGESWARAN has presented one of the first attempts to address this task [9]. It is a modification of the original Use
Case Points (UCP) Analysis, where the result is proportional to the required effort in test activities.

This method has the drawback that it estimates the effort of the whole test process, without discriminating the test phases; besides, it requires subjective judgment to perform the estimate. In an attempt to address some of these issues, modifications on Nageswaran’s method were proposed by ALMEIDA et al. [5].

In a preliminary study, SILVA et al. [6] have proposed the application of an efficiency metric to estimate the execution effort of functional tests. To assure its simplicity, the model construction assumes a stable test environment and does not require information on the system to be tested or on the test team composition. However, this precludes a robust behavior in face of the different characteristics of organizations.

ARANHA & BORBA [4] propose a different metric based approach, defining the concept of “test case complexity”. Each test case (TC) is evaluated within a group of characteristics to obtain a relation between the complexity value and the effort actually spent. Another proposal of complexity metric to estimate effort has been recently presented by KUSHWAHA [10]; there are no experimental results to validate it.

Another approach to deal with the problem consists on using models that are inspired in dynamic systems theory to describe the test process. CALZOLARI et al. [11] propose to describe the evolution of the maintenance and testing effort by means of the predator-prey dynamic model, analyzing the linear and non-linear mathematical variations. STIKKEL [12] proposes an adaptation and extension to Calzolari’s linear model to provide effort tracking and estimates of test completion for system testing process.

Finally, a usual approach to estimate effort of software development process has been recently considered to tackle the test effort problem. ZHU et al. [3] train a Support Vector Machine with a database containing test cycle’s characteristics and effort information.

To estimate execution effort of functional tests, this paper adopts the same principle of training a machine learning model to learn from past data, with an additional process of selecting the best input variables.

III. MACHINE LEARNING TOOLS

Machine learning techniques have been recommended when it is hard to describe all variables that model the problem. The test process can be considered such a kind of problem, since human intervention considerably increases its complexity as there is no way to know exactly all determinant factors of test execution effort.

The following subsections briefly introduce the machine learning tools that we have chosen to analyze with respect to the effort estimation problem. First, the wrapper variable selection process is aimed to improve data quality; after that, an Artificial Neural Network model and a Support Vector Machine model are synthesized to have their estimates compared in two real case studies.

A. Variable selection

Nowadays, machine learning problems are common that explore data spaces containing hundreds, maybe thousands of variables. New techniques have emerged to eliminate irrelevant and redundant variables in databases, to reduce the analysis effort and to improve the learning algorithms’ performance. In situations where the number of data records is relatively small, the importance of a variable selection process may be even greater because of the noise introduced by irrelevant variables; in addition to that, less variables means lower overall dimension of the problem and less data required.

Variable selection have many benefits, such as: better data understanding and visualization; simpler infrastructure for information collection and storage; time required to train and operate learning machines is reduced. This paper applies the wrapper method of variable selection to prepare the two databases of our case studies. The implementation is in conformance with the proposal of KOHAVI & JOHN [13], and its basic principle can be observed in Figure 1. The figure shows a search problem where a pre-defined induction model is used as a black box to provide a performance measure for each candidate variable subset that is evaluated during the search process.

![Figure 1. Variable selection by the wrapper method.](image)

B. Support Vector Regression

Support Vector Machines (SVMs) had their development started in 1992, by Vapnik and his colleagues at Bell Labs. They are based on Statistical Learning Theory, also known as Vapnik-Chervonenkis Theory, whose fundamental principle is the Structural Risk Minimization [14]. SVMs have been used initially in pattern classification problems such as OCR (Optical Character Recognition). Competitive results were quickly achieved in OCR and other problems of pattern recognition [15] and data regression [7].

We focus on SVMs to solve non-linear regression problems. This approach is typically known as Support Vector Regression (SVR), and a good introduction to this specific
technique is presented by SMOLA & SCHOLKOPF [16]. We propose to use the traditional \(\varepsilon\)-SV method [14], which defines the \(\varepsilon\)-insensitive loss function. The idea of this loss function is that errors smaller than a threshold \(\varepsilon > 0\) are ignored, while bigger errors are measured by the slack variables \(\xi\) and \(\xi^*\), as shown in Figure 2.

Consider a training data set

\[
\{(x_1, y_1), (x_2, y_2), \ldots, (x_1, y_1)\} \subset X \times \mathbb{R},
\]

where \(X\) is the input data space (e.g. \(X = \mathbb{R}^d\)). The objective is to find an approximation function

\[
f(x) = \langle w, \Phi(x) \rangle + b \quad \text{with} \quad w \in X, \quad b \in \mathbb{R},
\]

where \(\Phi : X \rightarrow F\) is a non-linear map into some feature space \(F\), that makes the value of \(\varepsilon\)-insensitive loss function minimum and, simultaneously, keeps \(f(x)\) as flat as possible. This search task can be converted into the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) \\
\text{subject to} & \quad \begin{cases}
    y_i - \langle w, \Phi(x_i) \rangle - b \leq \varepsilon + \xi_i \\
    \langle w, \Phi(x_i) \rangle + b - y_i \leq \varepsilon + \xi_i^* \\
    \xi_i, \quad \xi_i^* \geq 0
\end{cases}
\end{align*}
\]

The constant \(C > 0\) establishes a trade-off between the flatness of the approximation function and the tolerance to errors larger than \(\varepsilon\). To solve this optimization problem, the design of the SVR algorithm comprises the use of Lagrange multipliers and kernel functions. These functions are defined as \(K(x, x') : = \langle \Phi(x), \Phi(x') \rangle\); they avoid the computation of \(\Phi(x)\) explicitly during the optimization process.

We consider the Radial Basis Function (RBF) kernel in this work, which is computed as:

\[
K(x, x') = \exp \left( -\frac{||x - x'||^2}{2\sigma^2} \right).
\]

The parameters \(C\) and \(\varepsilon\) influence the generalization performance of the model, so a careful choice of their values is important. For the RBF kernel, the parameter \(\sigma\) must be chosen carefully too.

C. Artificial Neural Networks

Artificial Neural Networks (ANNs) are massively parallel and distributed information processing systems, composed by simple processing units (neurons), that have the intrinsic property of storing knowledge by experience and then make it available to be used [15].

ANNs’ adoption offers several useful properties, such as non-linearity, input-output mapping, adaptivity and fault-tolerance. Besides, the architectural possibilities, the different choices of neuron and learning strategies create multiple approaches, with distinct practical applications such as function approximation, temporal series prediction, pattern classification, associative memory, data clustering, etc..

We use in this work the Multilayer Perceptron (MLP), a feedforward ANN with multiple layers of neurons and usually trained with the supervised learning algorithm known as backpropagation [15]. Its architecture comprises: (1) one input layer; (2) at least one hidden layer composed by neurons with sigmoid activation function; and (3) an output layer. Figure 3 presents an example of an MLP network, with two hidden layers. There are \(n\) input values and \(m\) outputs, and the value +1 as input in each neuron layer corresponds to the bias value.

The training process of an MLP network is a non-linear and non-constrained optimization problem, where a search
takes place for a minimum of the error function between the network output and the desired output. This cost function traditionally is the mean square error (MSE), that is, the average of the instantaneous square error values for all training patterns that are presented to the ANN.

Equation 4 defines the instantaneous square error, while Equation 5 computes the MSE. \( N \) is the output layer’s number of neurons, \( d_k(t) \) is the desired output for the neuron \( k \) when it is provided an input in time \( t \), \( y_k(t) \) is the output computed by neuron \( k \) when it is provided the same input in time \( t \), and \( M \) is the number of training patterns.

\[
J(t) = \frac{1}{2} \sum_{k=1}^{N} [d_k(t) - y_k(t)]^2 
\tag{4}
\]

\[
J_{av} = \frac{1}{M} \sum_{t=1}^{M} J(t) 
\tag{5}
\]

IV. MLP ASYMMETRIC COST FUNCTION

Based on the authors’ experience as members of the ITT of Brazilian Federal Revenue Service’s HARPIA Project, and on a research conducted by JØRGENSEN & SØRBØG in 2001 [17], it is possible to conjecture on the impact of effort estimates on the test process.

Consider a scenario where there is a pessimistic estimate (also called an underestimate). Two case studies of JØRGENSEN & SØRBØG’s work point out that the remaining time when the job is completed is spent on improvements in software product. Analogously, during test execution in the HARPIA project the same situation happened: deeper examinations and other types of tests, such as usability and performance, were executed until the deadline. These behaviors comply with Parkinson’s principle [18], which states that “Work expands so as to fill the time available for its completion”.

Now consider a scenario where there is an optimistic estimate (also called an overestimate). The same paper shows that the software is simplified, quality requirements are ignored, and/or the test effort is decreased to deliver the product on schedule. Considering this situation in the HARPIA project, the most common decision was to prioritize test cases that could verify the most important functionalities, ignoring software’s minor functions.

We can assume that overestimating usually leads the test team to do extra work to fill the remaining time, and consequently that improves the system verification. On the other hand, underestimating increases the risk of not uncovering important faults, due to the test cases prioritization and effort cutting, which may affect software’s quality.

Pessimistic or optimistic estimates undoubtedly lead to losses, but if we prioritize the quality of delivered products, underestimating will cause worse damages than overestimating. We can conclude that it is prudent to overestimate, because underestimated effort increases the cost to finish the job on expected time, generates stress in people involved on tasks execution, in addition to increasing the risk of delivering poor quality products [19].

The traditional loss function (see Equation 5) of an MLP network is symmetric in the \( y \) axis, i.e., negative or positive error values results in the same function value. To minimize the impact of bad estimates, it would be better to build a model that “prefers” overestimation over underestimation.

We propose the definition of an asymmetric square error function to be used during the training of an MLP network. This can make the weight of an underestimate bigger than that of an overestimate in the computation of the loss function. We can modify Equations 4 and 5 to define the new error function and its corresponding average value.

**Definition 1**: Sum of Weighted Square Errors.

Consider the function \( p : \mathbb{R}^4 \to \mathbb{R} \)

\[
p(d, y, \alpha, \beta) = \begin{cases} 
\alpha (d - y)^2 & \text{if } (d - y) > 0 \\
\beta (d - y)^2 & \text{if } (d - y) \leq 0 
\end{cases}
\]

we define that the sum of weighted square errors is

\[
J_p(t, \alpha, \beta) = \frac{1}{2} \sum_{k=1}^{N} p(d_k(t), y_k(t), \alpha, \beta).
\]

**Definition 2**: Average of Weighted Square Errors Sums.

\[
J_{pav} = \frac{1}{M} \sum_{t=1}^{M} J_p(t, \alpha, \beta)
\]

Parameters \( \alpha \) e \( \beta \) must be constant; they determine the weights of a negative or positive error, respectively. In order to have a loss function that gives a bigger weight to underestimates, we must choose values so that \( \alpha > \beta \). As an example, consider \( \alpha = 1.5 \) e \( \beta = 1 \); thus we have a function \( p \) that computes the square value if the error is negative (overestimate), and the square value increased in 50% if the error is positive (underestimate). Figure 4 illustrates this scenario.

![Figure 4. Function \( p(d, y, \alpha, \beta) \) with \( \alpha = 1.5 \) and \( \beta = 1 \).](image-url)
CRONE [20] presents a generalization of the backpropagation algorithm calculation to allow the use of any differentiable (analytically or numerically) loss function. Based on this work, we have implemented a modified backpropagation algorithm in MATLAB® to train an ANN and perform the experiments on test execution effort estimation.

V. EXPERIMENTS

In order to evaluate our proposal, real case studies have been conducted with two different databases:
- Database #1: data from the HARPIA project and from a test specialized enterprise that was created by former members of HARPIA’s ITT. Both organizations followed the same test process, and collected the same metrics.
- Database #2: data from the Information Technology Department of a Chinese financial services organization, provided by Xiaochun Zhu.

Each case study consists of two experiments with the respective databases. We consider the problem of estimating execution effort for a test execution cycle, i.e., the complete execution of a test case suite. Suppose there is an input variable “Total number of Test Cases”; a record of this variable in the database, for instance, would be the number of test cases that were executed during one test cycle for a software.

Four models are compared in the experiments: (1) the Sequential Minimal Optimization (SMO) algorithm [21] of ε-SV regression; (2) a traditional MLP network with the second order Levenberg-Marquardt training algorithm; (3) the MLP network with the weighted square errors cost function and the generalized first order backpropagation algorithm; (4) and a basic least-squares linear regression model.

The training algorithms of both MLP networks run a fixed number of epochs, and choose the configuration that has achieved the best performance for the validation set. We have not found quantitative studies about effort estimation impacts to determine parameters α and β of the weighted loss function; we use α = 1.5 and β = 1 so that the underestimated value has a weight 50% larger than the overestimated value.

It is also important to emphasize that before the beginning of the experiments with each model, the wrapper variable selection is executed to select the appropriate variable subset for that model.

The first experiment uses cross-validation and aims to answer the question: “What machine learning model has the best performance?” Measures of performance for predictive models can be done with several formulas; we have chosen the most adopted ones: the mean absolute relative error (MARE) and PRED(0.25). The MARE is given by the following equation:

\[
MARE = \frac{1}{n} \sum_{i=1}^{n} \frac{|d_i - y_i|}{d_i}
\]

where \( n \) is the number of records, \( d_i \) is the desired output value and \( y_i \) is the estimated output value. PRED(0.25) is defined as the percentage of estimates with an absolute relative error less than or equal to 25%.

We have a limitation due to the number of available records: there are only 30 in database #1 and 70 in database #2. We were unable to collect more data with other researchers and organizations; possible reasons for this may be secrecy restrictions and absence of well defined processes for data collection [19].

This situation increases the risk that data are not representative enough to provide general conclusions on the problem. Several proposals and studies deal with the same scenario by using local approaches, i.e., they apply methods with adjustments and tailorings so that they function according to data and environment of the specific organization.

As a way to cope with our restrictions, Kirsopp & Shepperd [22] proposed a large number of experiment’s measure repetitions. We adopted this approach; thus, the first experiment can be detailed by the following steps:
1) Randomly split data into 80% for the training set and 20% for the validation set.
2) Considering the same division, train the 4 models and record the performance (MARE and PRED(0.25)) with the validation set.
3) Repeat steps one and two a thousand times.
4) Compare the average values of each measure.

The second experiment has a slightly different purpose, which consists of evaluating each model as the estimation tool of a test organization. The idea is to simulate a real scenario, through the following steps:
1) Split data into 70% for the training set and 30% for the test set, according to the chronological order of test cycles occurrence.
2) Train the inductive model with the training set. If a validation set (traditional MLP and the weighted MLP networks) is necessary, the training set is split again into 75% for training and 25% for the validation set, which is used for the training stopping criteria.
3) Estimate the effort using the trained model and the records of the test set.
4) Compare the estimates with the actual values, considering again the metrics MARE and PRED(0.25).

For the traditional MLP and weighted MLP the second step of the second experiment is repeated ten times and the best result is chosen as the model configuration, in order to minimize the variability of training results due to local optimum points. The wrapper execution and both experiments use the same parameter settings for each model, defined by eight previous training runs.
A. Case study: Brazilian Federal Revenue Service

Database #1 contains data from 25 test cycles of six software systems that were executed by the ITT of the HARPIA project, plus data obtained from 5 test cycles of a seventh system, provided by a software testing organization which was created by former employees of HARPIA. There are 30 test cycles from 7 software systems, a total of 3,017 test case executions between August 2007 and November 2008. Six software systems are web-based and one is a desktop application. The development language in all projects is Java.

After an analysis of the organizations’ test processes and the factors that might influence test effort, eleven variables were collected to be used by the inductive models:

1) Total number of TCs;
2) Total number of TCs’ steps;
3) Number of TCs to execute;
4) Number of TCs’ steps to execute;
5) Number of TCs to re-execute;
6) Number of TCs’ steps to re-execute;
7) Number of testers in charge;
8) Experience of testers, measured by the number of days working at the organization;
9) Number of system’s use cases;
10) Number of system’s Lines of Code (LOC);
11) Number of system’s code classes.

The output variable consists of the total effort to execute the test cycle, in person-hours. The wrapper algorithm was executed for each of the four models under analysis. Thus, each model has its specific input variable subset as shown in Table I.

Table I

<table>
<thead>
<tr>
<th>Model</th>
<th>Selected variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR</td>
<td>{2,7,9,10,11}</td>
</tr>
<tr>
<td>Linear regression</td>
<td>{1,2,7}</td>
</tr>
<tr>
<td>MLP</td>
<td>{2,10}</td>
</tr>
<tr>
<td>Weighted MLP</td>
<td>{2,7,9}</td>
</tr>
</tbody>
</table>

Unfortunately, statistical analysis can not be applied for the first experiment due to the fact that the number of records is not large enough to satisfy criteria such as normal distribution or independence of the experimental measures [23].

Nonetheless, it can identify patterns that could provide information on the comparative behavior of the models. Table II presents the mean ($\mu$) and standard deviation ($\sigma$) of the 1,000 experiment’s means of MARE and PRED(0.25), abbreviated as P(0.25).

The bold values indicate that the Weighted MLP (abbreviated as W-MLP in the Table) network has achieved the highest mean of PRED(0.25) and also the smallest mean of MARE. This means that W-MLP has the best performance among all four inductive models, in both metrics. The SVR model’s performance has achieved the second highest value of average PRED(0.25) — 66.9% — while the linear regression model has obtained the second smallest value of average MARE — 26.7%.

Table II

<table>
<thead>
<tr>
<th>Model</th>
<th>Lin. Reg. $\mu$</th>
<th>MLP $\mu$</th>
<th>SVR $\mu$</th>
<th>W-MLP $\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\sigma$</td>
<td>$\sigma$</td>
<td>$\sigma$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>P(0.25)</td>
<td>57.5</td>
<td>18.9</td>
<td>62.0</td>
<td>22.8</td>
</tr>
<tr>
<td>MARE</td>
<td>26.7</td>
<td>10.1</td>
<td>27.4</td>
<td>15.4</td>
</tr>
</tbody>
</table>

Apparenty, the proposed modification in cost function has not impaired the Weighted MLP network’s performance. The better performance of the modified MLP model over the common MLP network reinforces our proposal.

For the second experiment, we include the proposal of test execution effort estimation by the accumulated efficiency metric [6]. The 21 first records are used as training set, and the last nine records of the database are the test set. The results are shown in Table III, where the first column indicates the machine learning model, with EM representing the Accumulated Efficiency Model; the central columns, for which labels “SxCy” mean “Cycle #y of software #x”, contain the relative error values for each of the nine test cycles, computed as:

$$E_{rel} = \frac{E_{est} - E_{act}}{E_{act}}$$

(7)

where $E_{est}$ is the effort estimate and $E_{act}$ is the actual effort; the last columns are the MARE and PRED(0.25) metrics.

The first remark is on the bad performance of the accumulated efficiency model: the MARE value was considerably high, 125.5%, and no absolute relative error is below 25%. Although SILVA et al. [6] reported a satisfactory case study, our current experiment shows that the model contains limitations leading to the worst performance among the models.

The linear regression model has the smallest MARE value, 31.1%, which is aligned to the good performance achieved in the first experiment. Considering PRED(0.25), the best performance was achieved by the SVR model, 77.8%, representing 7 test cycles in 9 with relative absolute errors under or equal to 25%. Still, the SVR has the second smallest MARE, 36%. The Weighted MLP network, despite not being the winner as in the first experiment, has the second highest value of PRED(0.25).

PRED(0.25) metric provides a better analysis than the MARE metric on the regularity of models’ estimates. A graphic with bars of the relative errors is another way to observe this property in the experiment’s results. Figure 5 shows the relative error values for each machine learning model in each test cycle.

We notice that no model obtained a relative error under 50% in estimates four and one, which correspond to the
Table III
USE SIMULATION (EXPERIMENT 2) FOR DATABASE #1 (PERCENTAGE VALUES).

<table>
<thead>
<tr>
<th>Model</th>
<th>S4_C8</th>
<th>S5_C1</th>
<th>S5_C2</th>
<th>S5_C3</th>
<th>S5_C4</th>
<th>S6_C2</th>
<th>S5_C5</th>
<th>S7_C5</th>
<th>MARE</th>
<th>PRED(0.25)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM</td>
<td>177.1</td>
<td>139.6</td>
<td>81.6</td>
<td>172.4</td>
<td>135.1</td>
<td>-56.5</td>
<td>152.6</td>
<td>97.9</td>
<td>116.5</td>
<td>125.5</td>
</tr>
<tr>
<td>SVR</td>
<td>87.4</td>
<td>21.5</td>
<td>23.1</td>
<td>112.0</td>
<td>-13.9</td>
<td>17.3</td>
<td>-10.4</td>
<td>19.0</td>
<td>19.5</td>
<td>36.0</td>
</tr>
<tr>
<td>MLP</td>
<td>92.6</td>
<td>-2.1</td>
<td>-5.1</td>
<td>90.9</td>
<td>-31.8</td>
<td>16.7</td>
<td>95.7</td>
<td>-3.6</td>
<td>59.0</td>
<td>44.2</td>
</tr>
<tr>
<td>Lin. Reg.</td>
<td>52.6</td>
<td>45.1</td>
<td>2.1</td>
<td>66.8</td>
<td>7.8</td>
<td>-62.9</td>
<td>11.2</td>
<td>-14.6</td>
<td>17.0</td>
<td>31.1</td>
</tr>
<tr>
<td>W-MLP</td>
<td>142.0</td>
<td>18.4</td>
<td>16.4</td>
<td>139.6</td>
<td>-16.8</td>
<td>-4.7</td>
<td>53.8</td>
<td>16.5</td>
<td>18.1</td>
<td>47.4</td>
</tr>
</tbody>
</table>

Figure 5. Relative error for each cycle in the second experiment for Database #1.

The chart in Figure 5 makes clear that the SVR model, indicated by the darkest gray color, has the most regular behavior among all models. Except for the previously mentioned cases, all estimates are within the range $[-25\%, 25\%]$, including five positive estimates in seven, which means that the model overestimated more times than underestimated.

Continuing with the regularity analysis, the Weighted MLP comes right after the SVR model, with six measures within the absolute error of 25%. However, the model has the highest errors for the first and fourth measures and, in the sixth cycle, the Weighted MLP model underestimates the effort, while the common MLP model overestimates the effort; that is, the modification of the cost function does not guarantee that the model will always “prefer” to overestimate. However, we expect the Weighted MLP to overestimate more frequently than the common MLP, given that both models have been sufficiently trained.

B. Case study: Chinese financial organization

Database #2 contains data from 70 test cycles that were executed in eighteen releases of a single software system, developed by the IT department of a chinese financial services organization. These test cycles account for 41,139 test case executions.

Due to the exclusive dependence on the research collaborator, it was not possible to perform a detailed test process analysis to define variables using the same criteria of Database #1. Only three variables are available to be used by the inductive models:

1) Total number of TCs;
2) Total number of TCs’ steps;
3) Number of system’s modified LOCs.

While the two first ones have the same definition used for Database #1, the third variable is exclusive of this case study. It consists of the number of lines of code that were updated since the previous release of the software.

Despite the several differences between Databases #1 and #2, those can be considered a positive fact, since the reality of software organizations is heterogeneity [19]. To study and propose an effort estimation methodology, it is valuable that experiments are performed using data in conformance with
this diversity. We are aware that the number of available records and databases for this experiment is not enough for general conclusions; but it may be sufficient to provide directions for the solution to test effort estimation problem.

Application of the wrapper algorithm for variable selection yielded the same result for the four models: variables one and two were chosen, and variable three was discarded. Thus, the same subset of variables was used to compare the four models, differently from the first case study.

The first experiment done in the first case study was performed for Database #2. Table IV presents the mean (μ) and standard deviation (σ) of the 1,000 experiment’s measures of MARE and PRED(0.25), abbreviated as P(0.25).

The linear regression model is clearly the winner. The bold values in the table show that the technique has achieved the highest value of P(0.25) and the lowest value of MARE. The common MLP network took the second place in the experiment, with the second highest value of P(0.25), 50%, and the second smallest value of MARE, 39.9%.

<table>
<thead>
<tr>
<th>Model</th>
<th>MARE</th>
<th>PRED(0.25)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR</td>
<td>91.8%</td>
<td>38.1%</td>
</tr>
<tr>
<td>MLP</td>
<td>128.4%</td>
<td>33.3%</td>
</tr>
<tr>
<td>Lin. Reg.</td>
<td>38.4%</td>
<td>57.1%</td>
</tr>
<tr>
<td>W-MLP</td>
<td>82.0%</td>
<td>28.6%</td>
</tr>
</tbody>
</table>

The Weighted MLP network model achieved the second lowest MARE value. However, its result was extremely bad (82.0%). The SVR has obtained the second place for the PRED(0.25) metric: 38.1%, also bad. The common MLP network has the worst performance.

Figure 6 provides the observation of regularity for each model, presenting the relative error values of each estimate, for each model. The linear regression model superiority is well illustrated in the chart, represented by the line with square shaped markers. Two estimates in 21 were bigger than 100% of the relative error. On the other hand, the common MLP network, represented by the line with triangle shaped markers, has a large prediction dispersion, with three estimates bigger than 500% of the relative error. The SVR and Weighted MLP models present similar curves, with a slight advantage for the second one.

Database #2 made available a higher number of records to train the inductive models when compared to Database #1. However, less information was available, only three variables. The overall result indicates a failure for non-linear models, and a reasonable performance just for the least squares linear regression model.

C. Summary of experimental results

Analyzing the general aspects and results of both case studies, two main limitations were faced during the process: (1) two databases and the small number of records in both of them is an environment insufficient to design experiments that can be statistically validated; (2) the second database had just three input variables.

The first case study provided good performances for the SVR model and the Weighted MLP network, when compared to other previous researches. Possibly, SVR good results may be due to its recognized capability of automatically creating sophisticated non-linear maps, without great adjustments or manual configuration settings required usually by ANNs. The risk of over-fitting is also better controlled in SVR models than in ANNs.

Despite MLPs’ setup being more difficult than that of SVRs, the Weighted MLP network achieved a good performance. The asymmetric loss function used to train the model may have been responsible for this improvement, but more experiments are necessary to obtain a definite conclusion.

Considering the second case study, non-linear models had a very poor performance. Only the linear regression model
had reasonable results; it is interesting to notice that the proposal had competitive results also in first case study. Thus, it is not possible to exclude the hypothesis that the problem can be solved by this approach, if a proper database is considered. The synthesis of an inductive model requires data with quality, both in number of records and in available information; a different situation might have occurred in the second case study if there were variables such as those in Database #1.

VI. CONCLUSIONS AND FUTURE WORK

This work investigated the application of machine learning and variable selection tools to solve the problem of estimating functional test execution effort. We chose artificial neural networks, support vector machines and linear regression as models to be analyzed. The wrapper method was the variable selection technique adopted.

Based on the analysis of impacts that effort estimation may cause in software engineering, we have proposed a modification in the loss function of the MLP network. The model derived from the proposal is called Weighted MLP network; its implementation has been evaluated and compared to other models.

Two case studies using two real databases have been performed. As a pre-processing activity, the variable selection had a useful role of complementing the variable definition activity. While the process of defining variables to compose a database is typically human based, the automatic process of variable selection refines the previous phase and probably selects variables that are really relevant to the specific inductive model.

To keep the cost of data collection low, we defined just variables that do not depend on subjective judgment to compose both databases. Teams with well defined test processes will have no difficulty to implement the same data collection scheme.

We performed two experiments in each case study. Unfortunately, we did not arrive to the best proposal to solve the problem due to the discrepancy of the results.

The computational cost to apply each model was low, due to the small number of records and variables that the databases contained. Considering that the priority of our work was to compare the models’ prediction efficacy, all implementations were done in the MATLAB® environment to facilitate analysis of results. If future work confirms the adoption of machine learning models to predict test effort, a new implementation will be required of the chosen model, and as a user-friendly test tool.

For projects where the test phase is executed traditionally, i.e., not by an ITT, application of the proposed methodology is more limited, as estimation is more useful at earlier stages, when test planning is being done. However, our approach can still be used after test design phase as a complement to previously applied estimation methods. For independent test teams and/or projects that perform several test regression cycles, our models are more useful — in this scenario, different systems are repeatedly tested and/or the retest cycles demand design of few additional test cases.

Despite the absence of a general conclusion, we can make some hypothesis from the results:

- The application of the wrapper algorithm to select effort database variables can improve the quality of estimates; at least it can validate the variables that were defined
to be part of the database;

- The performance of the linear regression model was competitive in the first case study; in the second case study, where a very restrictive database was available, it was the winner. Thus, we can conjecture that the linear regression model may succeed as a prediction tool where there is little information available in database and where the adopters prioritize simplicity over efficacy of estimates;
- In environments where robust databases are available, containing ten or more test process metrics and at least 30 records, the SVR model and the Weighted MLP model are recommended to provide more accurate results, instead of the linear regression model.

In short, this work confirms the high complexity of the effort estimation problem. Managers and other testing professionals require easy to use and efficient estimation methods. However, our experiments do not point to any approach to play such a role.

As future work, we intend to: repeat the experiments with more databases, from different organizations; perform a deeper analysis on the application of asymmetric loss functions to predict effort; and study the application of Machine Committee techniques [15].

ACKNOWLEDGMENTS

The authors thank Mr. Xiaochun Zhu, who gently shared the data required to perform the second case study.

REFERENCES