ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 3, No. 1, 49-65 (2018)
www.astesj.com
Special issue on Advancement in Engineering Technology

ASTES Journal
ISSN: 2415-6698

Improving System Reliability Assessment of Safety-Critical Systems using Machine Learning Optimization Techniques

Ibrahim Alagöz*[1], Thomas Hoiss[2], Reinhard German[1]

[1]*Department of Computer Science 7, FAU Erlangen-Nuremberg, 91058, Germany*

[2]*Automotive Safety Technologies GmbH, 85080, Gaimersheim, Germany*

A R T I C L E I N F O

Article history:
Received: 14 November, 2017
Accepted: 10 January, 2018
Online: 30 January, 2018

Keywords:
Safety-Critical System
Black Box Regression Testing
Linear Classifier
Selection
Prioritization

A B S T R A C T

Quality assurance of modern-day safety-critical systems is continually facing new challenges with the increase in both the level of functionality they provide and their degree of interaction with their environment. We propose a novel selection method for black-box regression testing on the basis of machine learning techniques for increasing testing efficiency. Risk-aware selection decisions are performed on the basis of reliability estimations calculated during an online training session. In this way, significant reductions in testing time can be achieved in industrial projects without uncontrolled reduction in the quality of the regression test for assessing the actual system reliability.

1 Introduction

Reliability assessment of safety-critical systems is becoming an almost insurmountable challenge. In the near future, the engineering of new applications for vehicles such as driving assistance functions or even autonomous driving systems will inevitably incur significantly increased engineering sophistication and longer test cycles. Thus, in the automotive domain, functional safety continues to be ensured on the basis of the international ISO 26262 standard. As both the levels of functionality such systems provide and their degree of interaction with their environment increases, an adequate increase in system safety assessment capabilities is required.

This paper is an extension of work originally presented at the 10th IEEE International Conference on Software Testing, Verification and Validation (ICST 2017) [1] and describes a methodology for efficiently assessing system safety. The focus of the paper is on regression testing of safety-critical systems consisting of black-box components. This scenario is common for automotive electronic systems, where testing time is expensive and should be reduced without an uncontrolled reduction in reliability.

The work reported here correspondingly seeks to increase testing efficiency by reducing the number of selected test cases in a regression test cycle. When a selection decision is made, the following two types of errors are possible:

- a test case is selected but would pass (type-I-error, false-positive case) and
- a test case is not selected but would fail (type-II-error, false-negative case).

Accordingly, we model a classifier $\hat{H}$ for solving the following optimization problem.

$$\begin{aligned} \min\ & p_{FP} = P(\hat{H} = H_1 | H_0) \\ \text{subject to } & p_{FN} = P(\hat{H} = H_0 | H_1) \leq p_{FN,MAX} \end{aligned} \tag{1}$$

A good standard of test efficiency calls for the avoidance of false-positives. This requires minimization of the probability of mistakenly assuming the rival hypothesis (H_1 : test case fails) even though the null hypothesis (H_0 : test case passes) is correct. Conversely, false-negatives mean that system failures remain undetected; the occurrence of this type of error must therefore be avoided with very stringent requirements. Thus, a predefined limit $p_{FN,MAX}$ for the probability of a false-negative is defined.

[1] proposed a concept for the selection of test cases based on a stochastic model. However, this paper proposes a holistic optimization framework for the safety assessment of safety-critical systems based on machine learning optimization techniques. We

*Corresponding Author: Ibrahim Alagöz, Hornstr. 1 85051 Ingolstadt, ibrahim.alagoez@gmail.com

https://dx.doi.org/10.25046/aj030107

suggest an incrementally and actively learning linear classifier whose parameters are estimated on the basis of Bayesian inference rules. As a result, our novel approach for modeling a linear classifier outperforms other machine learning approaches in terms of sensitivity.

Furthermore, this paper deals with the following fundamentally important research question: The machine learning approach is trained with data (test evaluations) obtained during a concurrently running regression test. How much training data is enough? When does regression test selection actually start?

We extend the proposed selection method [1] by introducing suitable test case features that are used in the machine learning approach for increasing performance (see [2]). Therefore, each feature introduced increases the complexity of the optimization problem (cf. Eq. 1) as a new dimension for optimization is introduced. Thus [3] and [4] suggest that high dimensional optimization problems can be solved in reasonable timeframes by using evolutionary algorithms instead of a (grid)search-based approach as given in [1]. Accordingly, we propose an evolutionary optimization approach for increasing testing efficiency.

Further extensions, such as the introduction of a prioritization strategy for test cases in order to select higher-priority test cases, will also be presented within this paper. In our novel approach, a linear classifier is trained in an online session; the ordering of the training data on the basis of a prioritization strategy therefore has the potential to improve our classifiers' performance.

We also provide an industrial case study to show the advantages of the suggested selection method. The study uses data from several regression test cycles of an ECU of a German car manufacturer, showing how test effort can be reduced significantly whereas the rates of both false-negatives and false-positives can be kept at very low values. In this example, we can quadruple test efficiency by keeping the false-negative probability at 1%.

We first discuss related work in Sec. 2 and explain basic definitions in Sec. 3. Accordingly, we motivate our research topic in Sec. 4 by giving some background information on regression tests and referring to the challenges. In Sec. 5, we give a brief overview of known machine learning methods' performance in solving safety-critical binary classification tasks. The concept of our novel machine learning approach is presented in Sec. 6. Sec. 7 discusses optimization strategies, and Sec. 8 focuses on the importance of the learning phase for the success of our approach. An industrial case study with real data is then given in Sec. 9. Finally, Sec. 10 presents the paper's conclusions.

2 Related Work

The automotive industry is currently engaged in a laborious quality assessment process around new engineered driver assistance and active and passive safety functions, while functional safety is ensured according to the international ISO 26262 standard [5]. Reliability assessment of systems is therefore, possible through both model-checking and testing.

Model-checking is used for verifying conditions on system properties. Thus [6] states that system requirements can also be validated by model-checking techniques. The idea is to check the degree to which system properties are met and to deduce logical conclusions on the basis of the satisfaction of system requirements. Model-checking has therefore gained wide acceptance in the field of hardware and protocol verification communities [7]. Motivated by the fact that numerical model-checking approaches cannot be directly applied to black-box components as a usable formal model is not available, we focus on model-checking driven black-box testing [6] and statistical model-checking techniques [8]. However, there exist some approaches for interactively learning finite state systems of black-box components (see [9] and [10]), which are proposed as *black box checking* in [11]. Learning a model is an expensive task, as the interactively learned model has to be adapted due to inaccuracy reasons. Nevertheless, some assumptions about the system to be checked, such as the number of internal states, are necessary; furthermore, conformance testing for ensuring the accuracy of the learned model has to be iteratively performed [9].

Therefore, [8] outlines the advantages of statistical model-checking as being simple, efficient and uniformly applicable to white- and even to black-box systems. [6] motivates on-the-fly generation of test cases for checking system properties; here, a test case is generated for simulating a system for a finite number of executions. All these executions are used as individual attempts to discharge a statistical hypothesis test and finally for checking the satisfaction of a dedicated system property.

Model-checking driven testing, or even simply testing a system in order to validate its requirements, is an expensive task, especially where safety-critical systems are concerned. However, the focus is on regression testing, which means that the entire system under test has already been tested once but has to be tested again due to system modifications that have been carried out. The purpose of regression testing is to provide confidence that unchanged parts within the system are not affected by these modifications [12]. White-box selection techniques have been comprehensively researched [13, 14]. However, we are here considering black-box components, and hence selecting test cases that only check modified system blocks gets difficult.

Since the implementation of black-box systems and moreover, the information on performed system modifications is not available [12], reasonably conducting a regression test becomes impossible.

Accordingly, regression testing of safety-critical black-box systems ends up in simply executing all existing test cases; this is a *retest-all* approach [12].

This is also motivated by the fact that in the automotive industry, up to 80% of system failures [1] that are detected during a regression test have not occurred previously. The reason behind this fact is that often many unintended bugs are introduced during a bug-fixing process. So between two system releases many new unknown errors are often introduced.

For reducing the overall test effort, we apply a test case selection method [1] based on hypothesis tests. Those test cases that are assumed to fail on their executions are accordingly selected. However, type errors while performing hypothesis tests are possible, as, for instance, in statistical model-checking.

We extend the proposed selection method into a holistic machine learning-driven optimization framework that utilizes suitable test case features for increasing testing efficiency (see [2]). Machine learning methods are often trained in so-called *batch* modes. Nevertheless, many applications in the field of autonomous robotics or driving are trained on the basis of continuously arriving training data [15]. Thus, incremental learning facilitates learning from streaming data and hence is exposed to continuous model adaptation [15]. Especially handling non-stationary data assumes key importance in applications like voice and face recognition due to dynamically evolving patterns. Accordingly, many adaptive clustering models have been proposed, including incremental K-means and evolutionary spectral clustering techniques [16].

Furthermore, labeling input data is often awkward and expensive [17] and hence accurately training models can be difficult. Therefore, semi-supervised learning techniques are developed for learning from both labeled and unlabeled data [17]. Motivated by these techniques, we propose a similar approach for effectively learning from labeled data. Hence, we cluster binary labeled data in more than two clusters for improving a classifier's learning capability due to the optimization of an objective function. Our optimization framework thus utilizes evolutionary optimization algorithms for handling the optimization complexity. Minimization of labeling cost on the basis of active learning strategies [18, 19] will also be dealt with in this paper.

3 Basic Definitions

We define the test suite $T = \{t_i \mid 1 \leq i \leq M\}$ consisting of a total of M test cases. $T_{Exec} \in T$ and $T_{\overline{Exec}} \in T$ are subsets of T that contain test cases that are executed and deselected in a current regression test respectively. Based on the test case executions ($\forall t_i \in T_{Exec}$), a system's reliability is actually learned, and thus the machine learning algorithm is trained.

The focus in supervised learning is on understanding the relationship between feature and data (here test case evaluation) [4]. Therefore, a test case needs to code a feature vector so that the indication of the coded features for a system failure can be learned in a supervised fashion. Such an indication is not just a highly probable forecast of an expected system failure, it is rather a particular risk-associated recognition.

First of all, a feature can be any individual measurable property of a test case. The data type of a feature is mostly numeric, but strings are also possible. However, such features need to be informative, discriminative and independent of one another if they are to be relevant and non-redundant The definition of suitable features increases the classifier performance [20]. In our application, a feature can be varied, such as a

- subjective ranking of a test case based on expert knowledge. Such rankings can hint at the error susceptibility of verified parts of the system;
- verified function's safety integrity level, known as the *ASIL* in automotive applications [5];
- name of a function whose reliability is assured;
- reference to any hardware component of a circuit board that is being tested in a hardware-in-the-loop (HiL) test environment;
- number of totally involved electronic control units during the testing of a networked functionality; Such a number can hint at the complexity of the networked functionality and hence at its error susceptibility.

We define the entire set of features $\Phi = \{\phi_f \mid 1 \leq f \leq F\}$ of test cases that might be relevant for understanding the behavior of test cases. Thus, features may be e.g. $\phi_1 = \{'QM','A','B','C','D'\}$ (*ASIL*) or $\phi_2 = \{f_1, f_2, f_3\}$ (function name). Hence, a test case can verify a function f_3 that has an *ASIL* A.

The following passages discuss the selection of suitable features, which is an important strategy for improving a classifier's performance.

- Sometimes less is more - If the defined set Φ is too large it can cause huge training effort, high dimensionality of the optimization problem and overfitting. Thus we define a selection mask $\boldsymbol{b}_s = \begin{bmatrix} 1 & 0 & 0 \cdots 1 \end{bmatrix}$ of length F for selecting relevant features Φ_s. If the $f-th$ matrix entry of $\boldsymbol{b}_s$ is greater than or equal to 1, then the corresponding feature $\phi_f \in \Phi$ is selected and added to Φ_s, otherwise not.
- The set of main features $\Phi_m \subseteq \Phi_s$ is coded as follows: If the $f-th$ matrix entry of $\boldsymbol{b}_s$ is equal to 2, then the corresponding feature $\phi_f \in \Phi_s$ is at the same time a main feature $\phi_f \in \Phi_m$, otherwise not. The main features are used to establish the overall training data set: The training data is adapted to each test case, and thus it is $T_{t_i} = \{t_j \mid t_j \in T_{Exec} \wedge t_i \overset{\Phi_m}{\equiv} t_j\}$. Hence, we define that two test cases t_i and t_j are equivalent $t_i \overset{\Phi}{\equiv} t_j$ if their features $\forall \phi_f \in \Phi$ have identical values.

Additionally, a cross-product transformation of features $\forall \phi_f \in \Phi_s \setminus \Phi_m$ is performed. Thus we define $\Psi = \phi_{f_1} \times \phi_{f_2} \times ... \times \phi_{f_h} \times ... \times \phi_{f_H}$ with $\phi_{f_h} \in$

$\Phi_s \setminus \Phi_m, 1 \leq h \leq H = |\Phi_s \setminus \Phi_m|$ consisting of features ψ_l that represent individual combinational settings for features $\phi_f \in \Phi_s \setminus \Phi_m$. In simple terms, the cross-product of our sample features is $\phi_1 \times \phi_2 = \{('QM', f_1), ('QM', f_2), ('QM', f_3), ('A', f_1), ...\}$ Finally, a Boolean function $check: T \times \Psi \rightarrow \mathbb{B}$ with $check(t_i, \psi_l) = \begin{cases} 0, & \text{if } t_i\text{'s features are given by } \psi_l \\ 1, & otherwise \end{cases}$ is defined.

In addition, the function $state: T \times R \rightarrow S$ is defined; it returns the state of a dedicated test case in a concrete regression test. The state has to be either 'Pass' or 'Fail', except for cases where the test case has not been executed so that its state is undefined. Therefore, $S = \{\text{'Pass'}, \text{'Fail'}, \text{'Undefined'}\}$ defines the set of possible states. Furthermore, the set $R = \{r_k \mid 0 \leq k \leq K\}$ includes r_0 which is the current regression test and older regression tests starting from the last regression r_1 to the first considered regression r_K. Lastly, we define the tuple $history(t_i) = \{state(t_i, r_1), ..., state(t_i, r_K)\}$ containing t_i's previous test results.

4 Motivation

In practice, finding suitable features is a difficult task. Since we focus here on black-box systems, system-internal information is not available that might be useful for understanding the system behavior. As a reason, we can only define the above listed features, which might be too high-level for classifying system failures. To illustrate this fact, Fig. 1 a) shows a typical situation: The behavior of test cases in relation to arbitrarily defined features ϕ_1 and ϕ_2 is given. Passed and failed test cases are presented by green squares and red diamonds respectively, and white circles stand for test cases yet to be executed.

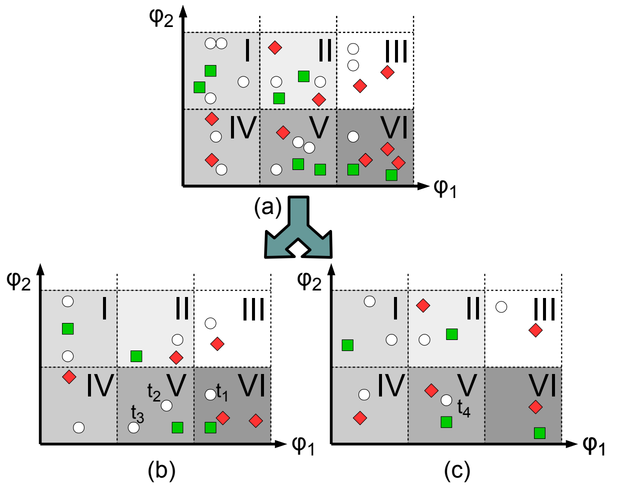


Figure 1: An artificial regression test with test cases.

We can see that the green squares and the red diamonds are widely scattered. Thus, defining a hyperplane in order to set two acceptance regions for passing and failing test cases is no easy matter. In order to solve this complex task, we develop a novel approach that is basically motivated by the following thought experiment: All test cases that are represented in Fig. 1 a) are now either assigned to 1 b) or 1 c) according to a certain mapping. The individual mappings of test cases will be discussed later, in Sec. 6. In the next step, a cross-product transformation is performed in order to group test cases into sub-regions (we refer these later as sub-clusters). In our example, we create six sub-regions. Table 1 lists the empirical failure probabilities of each sub-region.

In order to keep our thought experiment very simple, we will neglect statistical computations for now and focus only on the main idea of our novel approach. The introduction of Bayesian networks and hence the derivation of weights for linear classifiers will be discussed later, in Sec. 6. We assume for now that the calculated failure probabilities of test cases in Fig. 1 b) and Fig. 1 c) are correlated. So our example remains very simple, we also require that the failure probabilities of the corresponding sub-regions are equal. This assumption reduces the complexity of the following classification task enormously. We will classify the following test cases t_1, t_2, t_3 and t_4 in accordance with whether a selection is necessary or not.

Table 1: Failure probability of each sub-region.

$P(H_1)$	I	II	III	IV	V	VI
Fig. 1 a)	0	0.5	1	1	1/3	3/5
Fig. 1 b)	0	0.5	1	1	0	2/3
Fig. 1 c)	0	0.5	1	1	0.5	0.5

Only if t_1 passes will the failure probability of sub-region VI in Fig. 1 b) be equal to the failure probability of sub-region VI in Fig. 1 c). According to this fact, t_1 is assumed to pass, and hence it is deselected. Furthermore, t_2 will be selected as a fail of a test case inside sub-region V is expected. However, t_2 passes, and, based on the same consideration, t_3 is also selected and finally fails. Since now a failure probability of 1/3 is expected in sub-region V, t_4 is assumed to pass, and therefore it does not need to be selected. Table 2 summarizes all decisions executed.

Table 2: Test case states and algorithm decisions.

Test Case	State	Decision	Type of Decision
t_1	Pass	Deselected	True-Negative
t_2	Pass	Selected	False-Positive
t_3	Fail	Selected	True-Positive
t_4	Pass	Deselected	True-Negative

So our novel approach for solving binary classification tasks is based on calculated empirical probabilities and empirically evaluated correlations among those probabilities. The behavior of test cases can be precisely estimated on the basis of the calculated correlations. In practice, the failure probabilities of same sub-regions in Fig. 1 b) and Fig. 1 c) is often not exactly equal, but these failure probabilities are correlated. So the main task is to find *good* sub-regions for maximizing the empirically evaluated correlations and thus for precisely estimating the behavior of test cases. A more detailed explanation of our novel approach will follow in Sec. 6.

5 Performance of Known Machine Learning Methods

We have already indicated, by showing the example regression test in Sec. 4 (see Fig. 1), that according to the distribution of the input data, many machine learning methods cannot be reasonably applied for solving the constrained optimization problem (cf. Eq. 1). We will now demonstrate briefly that training linear classifiers in the classical sense by minimizing a loss function cannot perform well for solving safety-critical binary classification tasks. The situation is that only a small percentage of the data is actually labeled with one ('Fail'). Furthermore, failed and passed test cases are widely scattered in the feature space, which means detecting failing test cases becomes impossible. Additionally, the performance of deep neuronal networks is validated in the following.

The evaluation results (precision/recall) of these machine learning methods are given in Table 3. Each machine learning method is trained in the *batch* mode. The training data consists of all obtained test evaluations of a special regression test that will also be analyzed in our industrial case study in Sec. 9. For evaluating the machine learning methods, we used the training data first for training and later for testing (training data = test data). Even so, the sensitivity of both machine learning methods is zero, and thus we propose a novel approach for determining a linear classifier's parameters in Sec. 6.

Table 3: Performance of known machine learning methods in solving safety-critical binary classification tasks.

Machine Learning	Precision	Recall / Sensitivity
Linear Classifier (Trained by Minimizing a Loss-Function)	0	0
Deep Neuronal Network	0	0

6 Concept

The concept of our novel approach is shown in Fig. 2. We start with specifying a feature set Φ, and taking its subset Φ_s and finally constitute a cross-product feature transformation to obtain the set Ψ. Based on Ψ and by applying the *check*-function on T_{Exec}, test cases can be grouped. If we look back to the example where we grouped test cases in Fig. 1 a), then we will see that there is a relationship between test cases' features and their assignments to sub-regions.

Correspondingly, we introduce the definitions of clusters and sub-clusters of test cases. In the first step, test cases $\forall t_k \in T_{t_i}$ are assigned to clusters based on their history-tuples. A cluster is basically a partition of T_{t_i} and consists of test cases that have the same history-tuples. Accordingly, the number of distinct history-tuples N determines the total number of clusters. This is the step that has already been shown in Sec. 4, where test cases inside Fig. 1 a) were individually mapped into Figs. 1 b) and 1 c). In this way, already executed test cases depicted in Fig. 1 b) belong to one cluster and, analogously, those executed test cases that are depicted in Fig. 1 c) belong to another cluster.

In the next step, each cluster C_n is subdivided into L sub-clusters. A test case $t_k \in C_n$ is an element of the l-th sub-cluster $C_{n,l}$ if $check(t_k, \psi_l)$ is true. We originally introduced the terminology of sub-regions in Sec. 4. However, we focus in what follows on discrete valued features, which means that grouping test cases into sub-clusters is more appropriate. By introducing the function $eval : T_{Exec} \rightarrow \{0,1\}$ that is defined as follows

$$eval(t_i) = \begin{cases} 0, & \text{for } state\{t_i, r_0\} = \text{'Pass'} \quad (2) \\ 1, & \text{for } state\{t_i, r_0\} = \text{'Fail'} \quad (3) \end{cases}$$

the calculation of failure probabilities can be given in Eq. 4.

$$p_{n,l} = \frac{1}{|C_{n,l}|} \sum_{\forall t_i \in C_{n,l}} eval(t_i) \tag{4}$$

The given selection decisions in our example in Sec. 4 (cf. Fig. 1) were taken based on calculated failure probabilities. Additionally, the correlations between the failure probabilities were considered. Accordingly, we need a stochastic model for estimating the classifier's sensitivity and specificity. We propose a univariate and also a multivariate stochastic model. The short-comings of the univariate stochastic model for solving the optimization problem (cf. Eq. 1) will be discussed later to motivate the introduction of a multivariate stochastic model. First of all, the next step introduces a multidimensional Gaussian distribution that constitutes a distribution for the failure probabilities of test cases. Based on this distribution, two distinct Bayesian Belief networks for both stochastic models will be introduced.

In the following, we interpret $p_{n,l}, 1 \leq l \leq L$ as realizations of a random variable X_n. X_n is Gaussian distributed based on the following assumption:

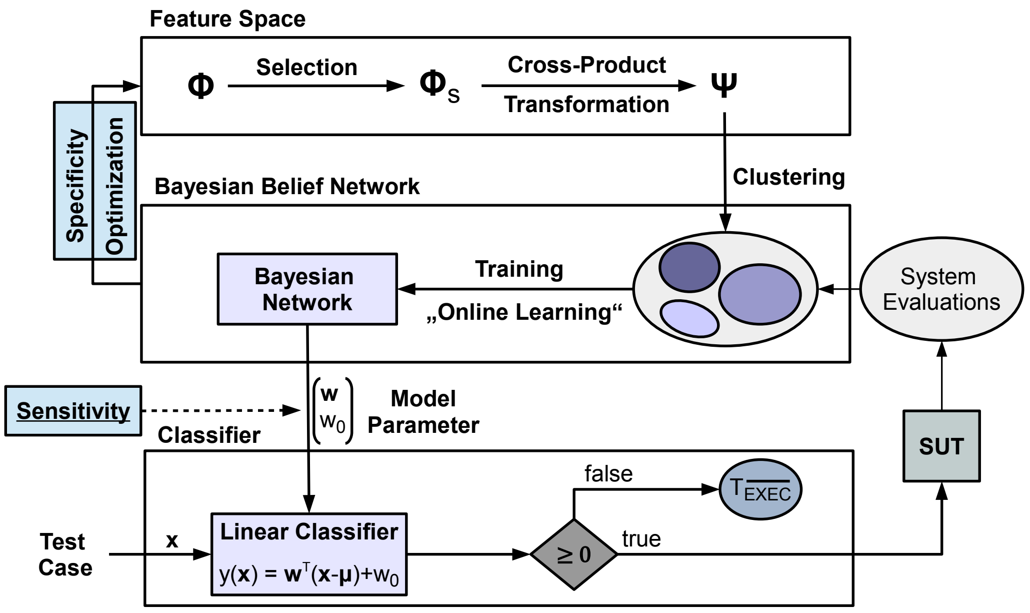


Figure 2: Determining the weights of a linear classifier for maximizing its specificity under the constraint of a specific sensitivity.

Since each test case evaluation is a binary experiment with two possible outcomes ('0' or '1'), it can be regarded as a realization of a binary random variable. As the sum of independent random variables results into a Gaussian random variable according to the central-limit theorem [21], considering test case evaluations as independent random experiments justifies X_n's assumed distribution. However, test cases are executed on the same system and there may be some dependencies between test case evaluations that cannot be directly validated by such means as performing code inspections. As a result, we assume a mix of dependent and independent test case evaluations, and hence the Gaussian assumption is still valid. The moments of X_n are $E[X_n] = \mu_n = \frac{1}{L}\sum_{l=1}^{L} p_{n,l}$ and $E[(X_n - E[X_n])^2] = \sigma_n^2 = \frac{1}{L-1}\sum_{l=1}^{L}(p_{n,l} - E[X_n])^2$. As we introduced in total N Gaussian random variables, the moments of the multidimensional Gaussian distribution are $\boldsymbol{\mu} = E[\boldsymbol{X}] = [E[X_1], E[X_2], ..., E[X_N]]^T$ and $\Sigma = E[(\boldsymbol{X}-\boldsymbol{\mu})(\boldsymbol{X}-\boldsymbol{\mu})^T]$.

Since the constraint of the optimization problem (cf. Eq. 1) has to be fulfilled, an accurate sensitivity estimation has to be iteratively performed.

6.1 Sensitivity Estimation

The formula for calculating the classifier's false-negative selection probability is given in Eq. 5.

$$p_{FN} = \frac{N_{FN}}{N_{FN} + N_{TP}} \le p_{FN,MAX} \tag{5}$$

However, $\hat{p}_{FN} = \frac{\hat{N}_{FN}}{\hat{N}_{FN}+N_{TP}}$ has to be estimated, since the number of mistakenly deselected failing test cases N_{FN} is unknown, and thus it is estimated by $\hat{N}_{FN}$. The number of already detected failing test cases is given by N_{TP}. Before a decision can be taken on whether a test case t_i can be deselected, the currently allowed risk of taking a wrong decision has to be estimated in advance. The estimation of $\hat{N}_{FN}$ has to be adjusted by the term $xP(\hat{H} = H_0|H_1)$, where x is the failure probability of t_i and $P(\hat{H} = H_0|H_1)$ is the estimated false-negative probability if t_i is deselected. Accordingly, the recursive formulation $\hat{N}_{FN,new} = \hat{N}_{FN,old} + xP(\hat{H} = H_0|H_1)$ is continuously updated whenever an arbitrary test case is deselected. As $\hat{p}_{FN} \le p_{FN,MAX} \Leftrightarrow \frac{\hat{N}_{FN,new}}{\hat{N}_{FN,new}+N_{TP}} \le p_{FN,MAX}$ is required, the maximum allowed false-negative probability for deselecting the next test case t_i is given in Eq. 6.

$$\begin{aligned} P(\hat{H} = H_0|H_1) &\le \frac{\frac{N_{TP}\cdot p_{FN,MAX}}{1-p_{FN,MAX}} - \hat{N}_{FN,old}}{x} \\ &=: p_{FN,Limit} =: \frac{p_{FN,Bound}}{x} \end{aligned} \tag{6}$$

6.2 Univariate Stochastic Model

We model the Bayesian Network that consists of the random variables X, H and $\hat{H}$, in Fig. 3. In the univariate stochastic model, the focus is on modeling of only one failure probability distribution. Thus the random variable X stands for the previously defined X_1 and its realization x is given by $p_{1,l}$ where l is the index of that sub-cluster $C_{1,l}$ that fulfills $t_i \in C_{1,l}$.

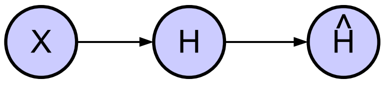


Figure 3: Bayesian Network consisting of the random variables X, H and $\hat{H}$.

H and $\hat{H}$ are binary random variables for modeling test case states and classifier decisions. As the state of a test case t_i is a-priori unknown, it needs to be modeled by a corresponding random variable. According to the realization of X, a pass or a fail of the corresponding test case t_i, whose failure probability distribution is modeled by X, is expected. Finally, $\hat{H}$ takes a decision for t_i based on its failure probability x:

$$\hat{H}(x) = \begin{cases} H_0, & \text{if } x \in \mathcal{X}_0 = [0; p_{TH}[\quad (7) \\ H_1, & \text{if } x \in \mathcal{X}_1 = [p_{TH}; 1] \quad (8) \end{cases}$$

According to $\hat{H}$'s selection rule a very simple *hyperplane* $y(x) = x - p_{TH}$ is derived where in the case of $y(x) \geq 0$ a selection decision is taken. A particularly important factor is the definition of the threshold probability p_{TH}, as its setting determines the classifier's sensitivity and specificity. The common ways of estimating false-negative and false-positive probabilities are given in equations 9 and 10, respectively.

$$\hat{p}_{FN} = \int_{\mathcal{X}_0} p(X|H_1)dx \tag{9}$$

However, we could only estimate the probability density function (pdf) $p(X)$ in contrast to the conditional probability density functions $p(X|H_0)$ and $p(X|H_1)$. The reason for this is that pdf estimations are based on mean calculations of test case evaluations. Hence passing and failing test cases are both considered in calculating average failure probabilities. Thus, $p(X)$ is a distribution over failure probabilities of passing as well as failing test cases. Accordingly, $p(X|H_0)$ and $p(X|H_1)$ cannot be estimated and in conclusion, $\hat{p}_{FN}$ and $\hat{p}_{FP}$ cannot be estimated as in Eq. 9 and 10.

$$\hat{p}_{FP} = \int_{\mathcal{X}_1} p(X|H_0)dx \tag{10}$$

Fig. 4 shows the important probability distribution functions that are used for estimating $\hat{p}_{FN}$ and $\hat{p}_{FP}$.

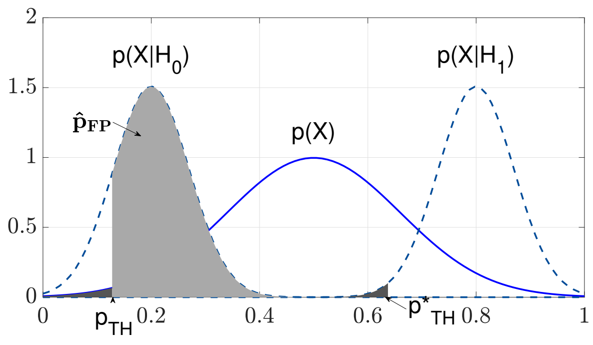


Figure 4: Considered probability distribution functions in the univariate stochastic model.

The threshold probability p_{TH} is calculated based on the estimation of $\hat{p}_{FN}$ as the following relation in Eq. 11 holds.

$$\hat{p}_{FN} = P(X < p^{*}_{TH}|H_1) \tag{11}$$

As Eq. 11 cannot be directly estimated, the following relation in Eq. 12 is used for estimating $\hat{p}_{FN}$ and finally for p_{TH}.

$$\hat{p}_{FN} = P(X < p_{TH}|H_1) \leq P(X < p_{TH}) \leq p_{FN,Limit} \tag{12}$$

$\hat{p}_{FN}$ is estimated in Eq. 12 according to the assumption that the quantiles of $p(X|H_1)$ are larger than the quantiles of $p(X)$. By solving Eq. 12 the threshold probability is computed as given in Eq. 13

$$p_{TH} = \text{erfinv}(2p_{FN,Limit} - 1)\sigma\sqrt{2} + \mu \tag{13}$$

with $\mu = E[X]$ and $\sigma = \sqrt{VAR(X)}$. As a result, the classifier's sensitivity is larger than $1 - p_{FN,Limit}$, since its false-negative selection probability is smaller than $p_{FN,Limit}$. Finally, the decision regions of the linear classifier are defined (cf. Eq. 7 and 8) by determining p_{TH}.

Furthermore, the minimization of the classifier's specificity is required by the definition of the constraint optimization problem (cf. Eq. 1). Accordingly, the classifier's false-positive selection probability is estimated as given in Eq. 14 and shown in Fig. 4.

$$\hat{p}_{FP} = P(X \geq p_{TH}|H_0) \tag{14}$$

However, $p(X|H_0)$ is not given and thus $\hat{p}_{FP}$ cannot reasonably be estimated. Furthermore, $\hat{p}_{FP}$ cannot be reasonably minimized as an optimization parameter is not defined; consequently, we need a so-called multivariate stochastic model for performing this. In the first instance, the idea of minimizing $\hat{p}_{FP}$ and hence gaining testing efficiency by regarding several distribution functions is explained.

6.3 Preliminaries

Let us assume that two dependent Gaussian random variables X and X' are given. The focus is again on estimating $\hat{p}_{FN}$ and $\hat{p}_{FP}$. Fig. 5 shows the probability distribution functions $p(X)$, $p(X|H_0)$ and $p(X|H_1)$ as in Fig. 4. Additionally, the a-posteriori failure probability distribution function $p(X|X')$ is shown.

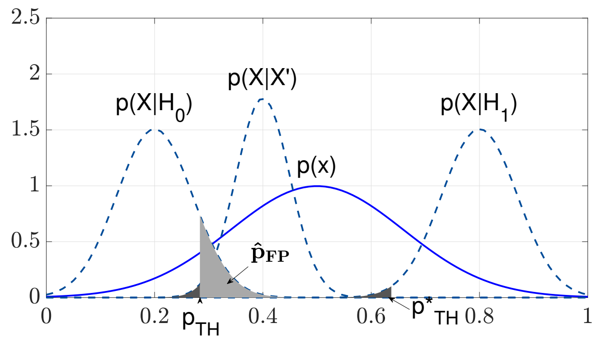


Figure 5: Qualitatively minimizing $\hat{p}_{FP}$ by introducing $p(X|X')$ and hence by using conditional informations.

The main idea is to use several observations of distinct dependent random variables to achieve a considerably more representative a-posteriori failure probability distribution function that is relatively narrow within a certain range. So $p(X|X')$ is considered as the more representative distribution for the failure probabilities and hence $\hat{p}_{FN}$ and p_{TH} are estimated by using this distribution function. Comparing Figs. 4 and 5, it can easily be seen that $\hat{p}_{FP}$ is basically minimized, since the risk of a false-negative selection probability is computed based on $p(X|X')$, which allows a more representative risk estimation.

All in all, by regarding a set of dependent Gaussian random variables and by using the information about their observations, a more representative a-posteriori failure probability distribution function is achieved, which allows a more precise risk estimation. Accordingly, the probability of false-positive selection can be minimized. As a result, a multivariate stochastic model is created to exploit the dependency information between random variables for finally achieving testing efficiency.

6.4 Multivariate Stochastic Model

By using the dependency between the random variables $X_n, 1 \le n \le N$, a considerably more accurate estimation of $\hat{p}_{FN}$ is achieved and hence $\hat{p}_{FP}$ is minimized. Fig. 6 shows the modeled Bayesian network consisting of the random variables $X_n, 1 \le n \le N$, H and $\hat{H}$.

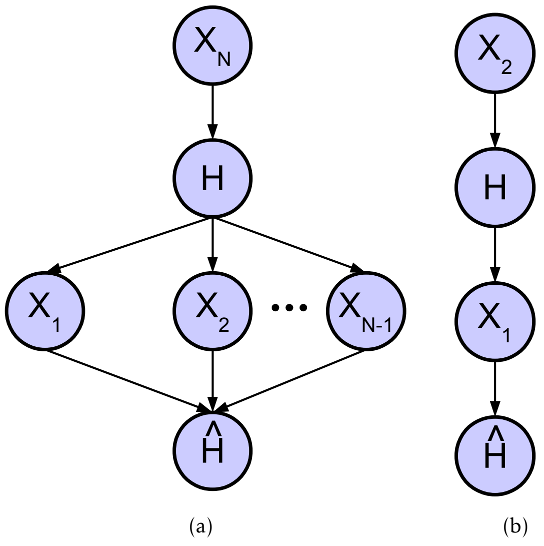


Figure 6: Bayesian Network consisting of an (a) indefinite and (b) definite number of random variables $X_n, 1 \le n \le N$.

The focus is on taking a selection decision for an arbitrary test case t_i. X_N now models the failure probability distribution of t_i. In our previous example, as shown in Fig. 1 c), the failure probability distribution of test case t_4 was calculated based on the empirically evaluated failure probabilities of test cases inside Fig. 1 c). Thus t_4 was an element of C_2 ($N = 2$), and its failure probability was modeled by X_2. Analogously, X_1 was defined by the empirically evaluated failure probabilities of test cases inside Fig. 1 b). In the interest of simplification, we always assume that the currently focused test case t_i is an element of cluster C_N and thus X_N models its failure probability distribution.

Furthermore, we can calculate the dependency among $X_n, 1 \le n \le N$. However, the Bayesian network in Fig. 6 models the statistical dependency between H and further random variables $X_n, 1 \le n \le N-1$. These dependencies cannot be calculated, but have to be modeled for estimating $\hat{p}_{FN}$ and $\hat{p}_{FP}$.

First of all, we model the classifier $\hat{H}(\cdot)$ as follows

$$\hat{H}(x_{ML}) = \begin{cases} H_0, & \text{if } x_{ML} \in \mathcal{X}_0 = [0; p_{TH}[\quad (15) \\ H_1, & \text{if } x_{ML} \in \mathcal{X}_1 = [p_{TH}; 1] \quad (16) \end{cases}$$

where $x_{ML} = \underset{x_N}{\operatorname{argmax}}\Big[\ln[\mathcal{L}(x_N|x_1,...,x_{N-1})]\Big]$ (consult [1]) is the maximum likelihood estimation. Accordingly, the likelihood estimation is a weighted sum as given in Eq. 17

$$x_{ML} =: \sum_{n=1}^{N-1} w_n(x_n - \mu_n) + \mu_N \tag{17}$$

with weights $w_n, 1 \le n \le N-1$ as given in Eq. 18.

$$w_n = -\frac{(\Sigma^{-1})_{n,N}}{(\Sigma^{-1})_{N,N}} \tag{18}$$

Further, p_{TH} has to be calculated based on a precise estimation of $\hat{p}_{FN}$. Thus we derive a calculation formula for $\hat{p}_{FN}$ for the case $N = 2$, but we will also provide a general calculation formula of $\hat{p}_{FN}$ for an arbitrary number N of random variables.

6.4.1 Derivation of probability distribution functions

In the following, some probability distributions are driven that are used for estimating $\hat{p}_{FN}$. First of all, the joint pdf $p(\hat{H}X_1HX_2)$

$$p(\hat{H}X_1HX_2) = p(\hat{H}|X_1)p(X_1|H)p(H|X_2)p(X_2) \tag{19}$$

and the conditional pdf $p(\hat{H}X_1H|X_2)$ are given in Eq. 19 and 20, respectively.

$$\begin{aligned} p(\hat{H}X_1H|X_2) &= \frac{p(\hat{H}X_1HX_2)}{p(X_2)} \\ &= \underbrace{p(\hat{H}|X_1)p(X_1|H)}_{p(\hat{H}X_1|H)}p(H|X_2) \end{aligned} \tag{20}$$

In the next step Eq. 21 is obtained by setting the equation $p(\hat{H}|X_1)p(X_1|H) = p(\hat{H}X_1|H)$ into Eq. 20.

$$p(\hat{H}X_1|H) = \frac{p(\hat{H}X_1H|X_2)}{p(H|X_2)} \tag{21}$$

Furthermore, the most important relation $p(\hat{H}|H) \le \frac{p(\hat{H}|X_2)}{p(H|X_2)}$ is driven in Eq. 22.

$$p(\hat{H}X_1|H) \le p(\hat{H}|H) \le \frac{p(\hat{H}H|X_2)}{p(H|X_2)} \le \frac{p(\hat{H}|X_2)}{p(H|X_2)} \tag{22}$$

Thus, the probability calculation $P(\hat{H} = H_0|H_1)$ can be estimated by using the relation in Eq. 22 as given in Eq. 23.

$$\hat{p}_{FN} = P(\hat{H} = H_0|H_1) \leq \frac{P(\hat{H} = H_0|X_2 = x_2)}{P(H_1|X_2 = x_2)} \tag{23}$$

Since $\hat{p}_{FN}$ cannot be directly estimated, as the conditional pdf $p(\hat{H}|H)$ is not given for performing the probability calculation $P(\hat{H} = H_0|H_1)$, the relation in Eq. 23 is used for estimating an upper bound for $\hat{p}_{FN}$. However, the linear classifier's actual false-negative deselection probability would be smaller than the calculated upper bound.

Since the constraint in Eq. 24 has to be fulfilled,

$$P(\hat{H} = H_0|H_1) \leq p_{FN,Limit} \tag{24}$$

we solve the inequality in Eq. 25.

$$\frac{P(\hat{H} = H_0|X_2 = x_2)}{P(H_1|X_2 = x_2)} \leq p_{FN,Limit} \tag{25}$$

As $x_2 = P(H_1|X_2 = x_2)$ holds, the following inequality is finally solved.

$$P(\hat{H} = H_0|X_2 = x_2) \leq p_{FN,Bound} \tag{26}$$

Eq. 26 is driven for the case $N = 2$ but in the general case, where the number of random variables $X_n, 1 \leq n \leq N$ is given by an arbitrary N, the following inequality has to be solved.

$$P(\hat{H} = H_0|X_N = x_N) \leq p_{FN,Bound} \tag{27}$$

By solving Eq. 27, the threshold probability

$$p_{TH} = -w_N(x_N - \mu_N) - w_0 + \mu_N \tag{28}$$

is obtained with weights

$$w_N = -\frac{e^{2I} - 1}{e^{2I}} \tag{29}$$

and

$$w_0 = -\frac{\sqrt{2}\sigma_N \sqrt{e^{2I} - 1}\,\mathrm{erfinv}(2p_{FN,Bound} - 1)}{e^{2I}} \tag{30}$$

Thus, the differential mutual information is defined in Eq. 31.

$$\mathcal{I} := \mathcal{I}(X_1, \ldots, X_{N-1}; X_N) \tag{31}$$

6.4.2 Conditional Independence

We have already motivated and introduced the following dependent random variables $X_n, 1 \leq n \leq N$. We have explained the fact that test case failure probabilities are correlated, since test cases are executed on the same system, and thus they show a dependent behavior.

However, the random variables $X_n, 1 \leq n \leq N$ are conditionally independent. This means that the information about a test case evaluation dominates such that a test case's originally calculated failure probability becomes irrelevant after observation of its state. Accordingly, the dependency among failure probabilities vanishes after observation of test case evaluations. This means that a fail of a test case t_m is actually expected based on the information about the evaluation of another test case t_n and no longer on t_n's originally calculated failure probability. Thus the remaining random variables $X_n, 1 \leq n \leq N-1$ become independent of the random variable X_N after observation of H's realization (cf. Fig. 6).

6.4.3 Specificity Estimation

The specificity is given by the term $1 - \hat{p}_{FP}$. As extensive mathematical derivations are needed for obtaining a calculation formula of $\hat{p}_{FP}$, these derivation steps are given in the appendix and in what follows here only the result is given.

Theorem 1 (False-Positive Probability Estimation). *$\hat{p}_{FP}$ is estimated as given in Eq. 32*

$$\hat{p}_{FP} = P(Z \geq -w_0 + \boldsymbol{w}^T \Delta\boldsymbol{\mu}) = \frac{1}{2}\left[1 - erf\left(\frac{-w_0 + \boldsymbol{w}^T \Delta\boldsymbol{\mu}}{\sqrt{2}\sigma_Z}\right)\right] \tag{32}$$

with $\sigma_Z = [w_1 \cdots w_{N-1}]\Sigma_{1,1}[w_1 \cdots w_{N-1}]^T + w_N^2\sigma_N^2$ and $\Delta\boldsymbol{\mu} = \boldsymbol{\mu} - \boldsymbol{\mu}_{H_0}$. The conditional moment has the following definition $E[\boldsymbol{X}|H_0] = \boldsymbol{\mu}_{H_0}$

For the case N = 2, Eq. 32 can be simplified; after several calculation steps the following Eq. 54 results

$$\hat{p}_{FP} = \frac{1}{2}\left[1 - erf\left(\psi\right)\right] \tag{33}$$

with

$$\psi = \frac{\sqrt{e^{2\mathcal{I}} - 1}\,erfinv(2p_{FN,Bound} - 1) + \frac{\sqrt{e^{4\mathcal{I}} - e^{2\mathcal{I}}}}{\sigma_1\sqrt{2}}\Delta\mu_1 - \frac{\sqrt{e^{2\mathcal{I}} - 1}}{\sigma_2\sqrt{2}}\Delta\mu_2}{\sqrt{2e^{4\mathcal{I}} - 3e^{2\mathcal{I}} + 1}} \tag{34}$$

6.4.4 Small Dimension Validation

Fig. 7 shows five plots of $\hat{p}_{FP}$ for different values of displacements $\Delta = \mu_n - \mu_{n,H_0}$. Indeed, the actual value of Δ is unknown. However, the focus is on the minimization of $\hat{p}_{FP}$. Accordingly, $\hat{p}_{FP}$ decreases in each sub-figure of Fig. 7. The actual value of Δ only determines how fast $\hat{p}_{FP}$ decreases. So we can solve the optimization problem (cf. Eq. 1) by minimizing $\hat{p}_{FP}$. We considered two random variables X_1 and X_2, as in our example in Sec. 4 where we created two clusters. A very important factor here is the underlying strategy for clustering test cases. As the distribution of the random variables X_1 and X_2 is directly related to the clustering strategy the main focus is on the maximization of the differential mutual information $\mathcal{I}(X_1; X_2)$. Accordingly, $\mathcal{I}$ is an optimization parameter for effectively reducing $\hat{p}_{FP}$. Lastly, we chose the

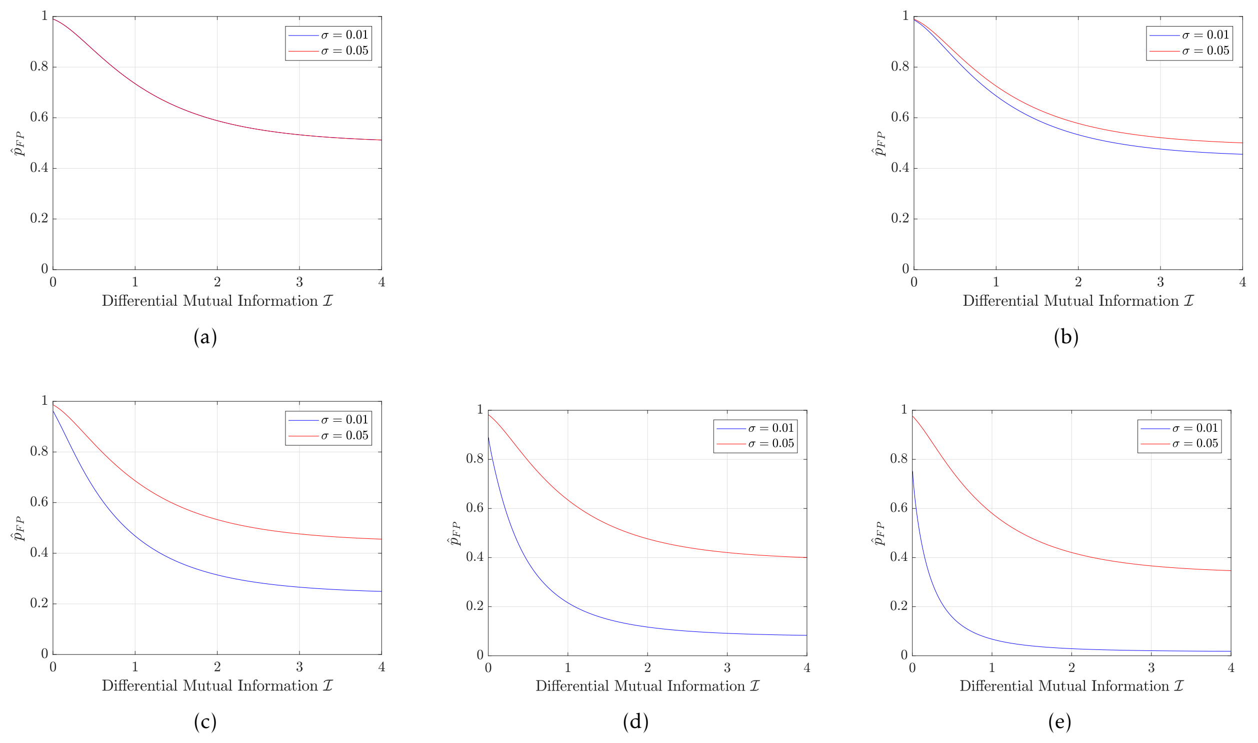


Figure 7: Estimation of $\hat{p}_{FP}$ for different values of $\mathbf{\Delta}$ and σ with $\mathbf{\Delta} = \alpha \cdot \boldsymbol{v} \cdot 10^{-3}$. The value of α is 0, 1, 5, 10 and 15 in Fig. 7 a), Fig. 7 b), Fig. 7 c), Fig. 7 d) and Fig. 7 e) respectively.

following values $\{0.01, 0.05\}$ for $\sigma = \sigma_1 = \sigma_2$ and selected the following displacements $\mathbf{\Delta} = \alpha \cdot \boldsymbol{v} \cdot 10^{-3}$ with $\alpha \in \{0; 1; 5; 10; 15\}$ and $\boldsymbol{v} = [1, -1]^T$.

7 Optimization

The first strategy is to optimize the feature selection. Optimal features are learned in an unsupervised learning session where an evolutionary optimization framework is applied to search for optimal features. The next strategy is to improve the labeling of test cases through an active learning strategy.

7.1 Evolutionary Optimization

Clustering (and sub-clustering) of test cases is performed based on features. Therefore, different clusterings for different selections of feature subsets (Φ_m, Φ_s) are possible. Accordingly, a different statistical model is obtained, as it reflects the failure frequencies in clusters. Furthermore, the differential mutual information (cf. Eq. 31) depends on the statistical dependencies and thus changes for different clusterings.

Sec. 6 proposed a calculation formula for the weights $w_n, 0 \leq n \leq N$, of a linear classifier. However, those formulas still depend on the differential mutual information I. A desired sensitivity has to be guaranteed, and thus the hyperplane is adjusted according to the value of I. It can be shown that for small values of I, the position of the hyperplane still guarantees a desired sensitivity but the false-positive selection probability increases. To minimize the false-positive selection probability, the differential mutual information has to be maximized, which is the final strategy for solving the constrained optimization problem (cf. Eq. 1).

First, clustering depends on the history-tuples of test cases as, for example, the length of the history-tuples determines the maximum number $|S|^K$ of clusters. Second, feature selection is optimized. All in all, we have summarized that K (number of considered previous regressions) is an optimization parameter and $\boldsymbol{b}_s$ (for coding selected and main features) is an optimization matrix. However, this is a large-scale high dimensional optimization problem, as there exist many possible settings for K and $\boldsymbol{b}_s$. Thus, [3] and [4] suggest that the high dimensional optimization problem can be solved in a reasonable time by using evolutionary algorithms. Accordingly, an evolutionary optimization framework is applied for solving the mentioned high dimensional optimization problem. As each setting for K and $\boldsymbol{b}_s$ is one possible solution for clustering test cases, which is the basis for derivation of a stochastic model, the fitness of this solution can be evaluated by calculating the extracted information $\mathcal{I}$ in Eq. 31. Thus, the optimal parameter and matrix setting with the best fitness will survive and will be returned by the evolutionary optimization algorithm.

Fig. 8 shows the overall flow chart of the evolutionary optimization framework. First of all, a new population consisting of several genotypes is initialized. Each genotype stands for a possible setting of K and $\boldsymbol{b}_s$. In the next step, the corresponding phenotypes of the genotypes are derived. Hence each phenotype encodes a stochastic model. Accordingly, the population is evaluated, wherein the fitness of each phenotype is calculated. However, a *bad* fitness is also possible due to *bad* statistical properties of the underlying stochastic model. This means that statistical calculations based on the stochastic model that a phenotype encodes cannot guarantee desired statistical confidence bounds. This will be explained in more detail in Sec. 8. Those phenotypes with *bad* fitness cannot survive and hence are eliminated.

Accordingly, remaining genotypes (phenotypes) are stochastically selected, and successively new genotypes are generated due to *crossover* and *mutation* operations. After a certain number of iterations, the phenotype with the best fitness will be selected, and this will be used in the selection algorithm. However, if the population is empty since all phenotypes were of *bad* fitness, then the training mode is activated, in which test cases are still executed without running the selection algorithm.

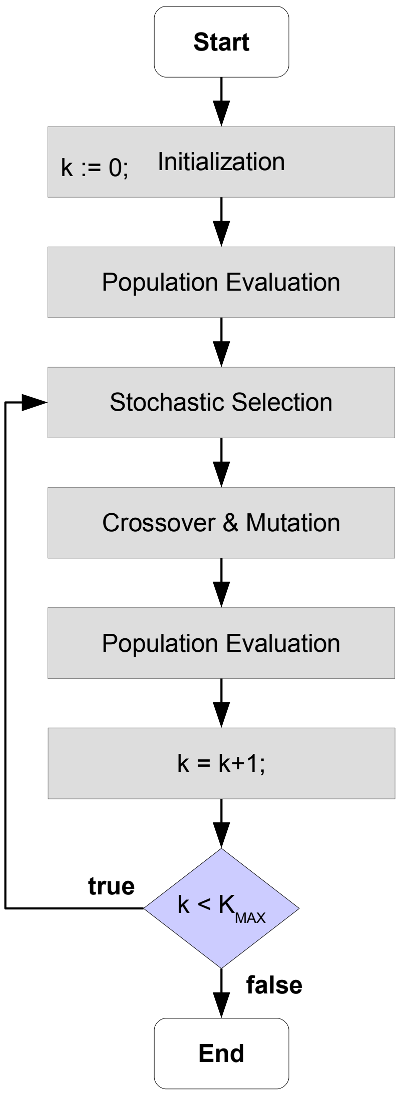


Figure 8: Evolutionary Optimization Framework.

7.2 Active Learning

Our classifier's conducted decisions can be regarded as *hard* or even as *soft* decisions. Once taken, *hard* decisions are never changed later on, in contrast to *soft* decisions. The test efficiency can be significantly increased by conducting *soft* decisions as opposed to *hard* decisions.

7.2.1 Hard Decision

[1] performs *hard decisions* since a selected test case is automatically executed and a once deselected test case is never selected again in the current regression test. In the following passages, the disadvantage of conducting *hard decisions* will be explained in detail in relation to classifier decisions.

The linear classifier's decision depends on the current estimation of $\hat{N}_{FN}$ as it calculates the allowed residual risk $p_{FN,Limit}$ (cf. Eq 6) of potentially taking a wrong decision. $\hat{N}_{FN}$ returns the number of supposedly unrevealed system failures that would be detected by those already deselected test cases that are elements of $T_{\overline{Exec}}$. Accordingly, the linear classifier's decision depends on the decisions it has already taken ($T_{\overline{Exec}}$) and hence it is memory driven.

Each deselected test case t_j has an individual additional contribution $\hat{N}_{FN,j}$ (cf. Eq. 36) to the overall estimation $\hat{N}_{FN}$ such that the relation in Eq. 35 holds.

$$\hat{N}_{FN} = \sum_{\forall t_j \in T_{\overline{Exec}}} \hat{N}_{FN,j} \tag{35}$$

$\hat{N}_{FN,j}$ is the product of t_j's failure probability x and the false-negative probability $P(\hat{H} = H_0|H_1)$ by deselecting t_j as given in Eq. 36.

$$\hat{N}_{FN,j} = P(H_1)P(\hat{H} = H_0|H_1) \tag{36}$$

Because of this fact, a deselection of an arbitrary test case can cause that the residual risk $p_{FN,Limit}$ reaches zero as $\hat{N}_{FN}$ increases (cf. Eq. 6). This means that no more risk ($p_{FN,Limit} = 0$) is allowed, and all remaining test cases have to be consequently selected.

Indeed, selecting test cases even if their deselection is allowed according to risk calculations is sometimes the better choice. In fact, this is the case if $p_{FN,Limit}$ is zero and thus it can be significantly increased by selecting and executing an already deselected test case in order to eliminate its risk. When this is done, $\hat{N}_{FN}$ decreases and hence $p_{FN,Limit}$ increases and thus a residual risk for further deselections is obtained.

However, the amount $\Delta\hat{N}_{FN}$ of how much $\hat{N}_{FN}$ can be decreased by selecting an arbitrary test case is significant. If later more than one test case can be deselected, and these deselected test cases add the same amount of expected unrevealed system failures $\Delta\hat{N}_{FN}$ to $\hat{N}_{FN}$ is in fact a gain in terms of reducing the regression test effort. So the strategy is to deselect primarily those test cases with fewer failure probabilities in order to increase testing efficiency.

As a result, the regression test efficiency can be increased. Therefore, the proposed selection method [1] is extended by a *soft decision* methodology. So each decision for deselecting a test case is now regarded as a *soft decision* that might be changed later. (We note here that the other way round is impossible since an already selected test case is automatically executed on the system under test and hence deselecting it later does not make sense).

7.2.2 Soft Decision

Fig. 9 shows the logic for managing soft selection decisions: Let us assume that t_i is the next test case that is analyzed by the linear classifier. If t_i is deselected, then it is queued into a priority queue whereby its priority is calculated as given in Eq. 37.

$$prio(t_i) = \hat{N}_{FN,i} = P(H_1)P(\hat{H} = H_0|H_1) \qquad (37)$$

In the other case, if t_i is selected then test cases deselected up to this point are analyzed to the end of improving the trade-off between the assumed risk and the total number of deselected test cases. As a consequence, the most probable failing test case t_j is obtained by taking the peek-operation on the priority queue. The priority of t_i and t_j is compared, and the test case with the higher priority is selected and executed on the system under test.

If t_j is executed, then it is removed from the set $T_{\overline{Exec}} \leftarrow T_{\overline{Exec}} \backslash t_j$ and added into the set $T_{Exec} \leftarrow T_{Exec} \cup t_j$. Furthermore, t_j's state is evaluated $eval(t_j)$ and accordingly the empirical failure probabilities of test cases are updated in algorithm 1. Since the calculated failure probabilities are averages of test case evaluations, the failure probabilities of those sub-clusters (see Eq. 4) have to be updated where t_j is an element of them. Accordingly, the failure probabilities of $\forall t_k \in T_{\overline{Exec}}$ are updated in algorithm 1.

Algorithm 1 Test case selection algorithm

```
procedure UPDATE STATISTICS($T_{\overline{Exec}}$, $t_j$)    ▷ $T_{\overline{Exec}}$ contains already deselected test cases; $t_j$ is executed
    for each test case $t_k \in T_{\overline{Exec}}$ do
        $\exists! C_{n,l} \Longrightarrow t_k \in C_{n,l}$    ▷ Find sub-cluster of $t_k$ and thus determine $n$ and $l$
        if $t_j \in C_{n,l}$ then
            $p_{n,l} \leftarrow \frac{1}{|C_{n,l}|} \sum_{\forall t_i \in C_{n,l}} eval(t_i)$    ▷ see Eq. 4
            $P(H_1) \leftarrow p_{n,l}$
            update $t_k$'s priority: $prio(t_k)$    ▷ see Eq. 37
        end if
    end for
    if $eval(t_j) == 1$ then
        $N_{TP} \leftarrow N_{TP} + 1$
    end if
end procedure
```

The important point is that even the failure probability of t_i is computed again. In most cases, t_i would be deselected. Nevertheless, it could be possible that the execution of t_j has failed, such that a further system failure has been found. In such a case, even t_i's failure probability may have increased such that its deselection has to be checked again by the linear classifier.

All in all, testing efficiency can be significantly increased by performing *soft selection* decisions. The performance of both selection strategies (*hard decision* and *soft decision*) will be compared in Sec. 9.

8 Learning Phase

The learning phase is of essential importance due to the fact that during this phase, the system reliability is actually learned. Test case selection is a safety-critical binary classification task as probably system failures would remain undetected and hence, a corresponding quality measure of wrong decisions is required. Accordingly, risk estimations on probably undetected system failures due to deselection of test cases have to be as accurate as possible. The more the system is learned during a regression test, the more precise the risk estimations are. However, learning a system in terms of understanding its reliability is a costly process, as it requires test cases to be executed. The fundamentally important research question is how much training data is enough for safely selecting test cases with a desired sensitivity.

8.1 Statistical Sensitivity Estimation

We have already required a specific sensitivity in the constraint optimization problem (cf. Eq. 1). Accordingly, we define the following confidence level in Eq. 38, which is basically driven from the constraint of Eq. 1.

$$P\Big(\Psi \leq \gamma\Big) \geq 1 - \alpha \qquad (38)$$

Ψ is an estimator for the number of false-negatives $\hat{N}_{FN} = \sum_{t_i \in T_{\overline{EXEC}}} \hat{N}_{FN,i}$ and the bound is given as $\gamma = \frac{N_{TP} \cdot p_{FN,MAX}}{1 - p_{FN,MAX}}$. $\Psi = \sum_i \psi_i$ is composed of several random variables ψ_i standing for the distribution of each $\hat{N}_{FN,i}$. ψ_i's distribution is complex, since the individual contribution of a deselected test case t_i is given by $\hat{N}_{FN,i} = x_N \hat{p}_{FN}$ where x_N is t_i's failure probability and $\hat{p}_{FN}$ is the corresponding estimated false-negative probability: The following theorem is already proved in [1] and gives the formula for the false-negative probability estimation.

Theorem 2 (False-Negative Probability)**.** *For a given p_{th} the calculation formula of the false-negative probability $P(\hat{H} = H_0|H_1)$ has the form*

$$\hat{p}_{FN} = \frac{1}{2}\left(1 + erf\left(\frac{1}{\sqrt{2}} \frac{x_N - (x_N - p_{th})e^{2\mathcal{I}} - \mu_N}{\sigma_N \sqrt{e^{2\mathcal{I}} - 1}}\right)\right) \qquad (39)$$

where x_N is the failure probability of a test case, whereas μ_N, σ_N are parameters of the probability distribution function $\mathcal{N}(\mu_N, \sigma_N)$.

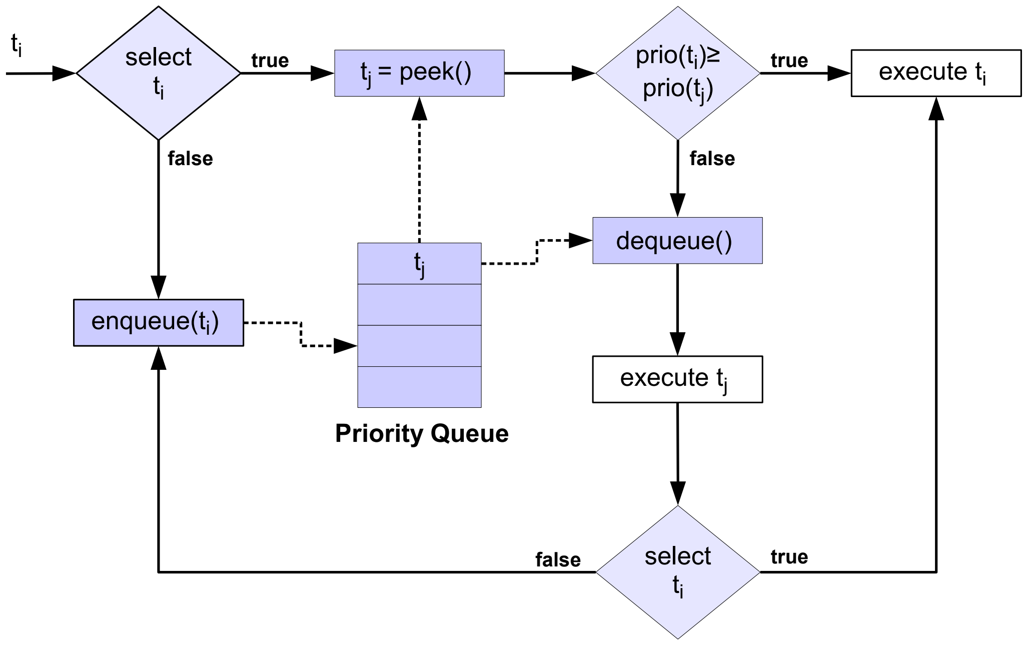


Figure 9: The extended concept for conducting soft decisions

These statistics are created on the basis of sample averages, such that the consideration of sample variances becomes inevitable during the estimation of confidence intervals. For instance, the false-negative probability estimation $\hat{p}_{FN}$ (see Eq. 39) is composed of the statistics x_N, μ_N, σ_N and I. Therefore, its variance depends on the individual variances of each statistic, including the sample variance of x_N. Especially the probability distribution of I is complex as it is non-linearly composed of a set of multivariate distributed Gaussian random variables.

Therefore, we choose the following approach to solving Eq. 38. We simplify the definition of ψ_i as follows $\psi_i = \hat{p}_{FN} \cdot X_N$ where $\hat{p}_{FN}$ is assumed to be a constant value without any distribution. This step simplifies the calculation complexity of Eq. 38 significantly, as Ψ becomes simply a weighted sum of Gaussian random variables. However, the variance of $\hat{p}_{FN}$ is of course relevant and should not be easily neglected. Accordingly, we require a maximum confidence interval width for $\hat{p}_{FN}$ such that the estimated false-negative probability is quite accurate and hence can be assumed to be just like a constant value without any statistical deviation. We calculate the confidence interval $[\hat{p}_{FN}^{(l)};\hat{p}_{FN}^{(u)}]$ and its width $\delta = \hat{p}_{FN}^{(u)} - \hat{p}_{FN}^{(l)}$ and require a maximal confidence interval width of δ_{max}.

The Wilson score interval [22] delivers confidence bounds for binomial proportions. Therefore, we calculate the following confidence intervals $[x_n^{(l)};x_n^{(u)}]$ (confidence level: $1-\alpha = 99\%$) for each failure probability estimation $x_n, 1 \leq n \leq N$. Each bound is consequently used for building the bounds of the composed statistics σ, Σ and I. By doing this, we obtain the following bounds: $\sigma^{(b)}$, $\Sigma^{(b)}$ and $I^{(b)}$ with $b = \{u, l\}$. Accordingly, we calculate $\hat{p}_{FN}^{(b)}$ by consequently inserting the bounds $\sigma^{(b)}$, $\Sigma^{(b)}$ and $I^{(b)}$ for the statistics σ, Σ and I respectively.

8.2 Criteria for Training

In order to guarantee a statistical bound on the sensitivity with a 99% confidence level, the following conditions have to be checked.

1. $\delta \leq \delta_{max}$
2. $P\Big(\Psi \leq \gamma\Big) \geq 1 - \alpha = 99\%$

If both conditions are fulfilled, then these risk calculations in the selection algorithm are reasonably accurate and hence selection decisions can be performed. However, if one condition is not fulfilled then the training mode is just active, such that test cases still have to be executed.

9 Industrial Case Study

A German premium car manufacturer constitutes each regression test as being a system release test, and thus the system test takes up to several weeks according to [5]. However, a first detected system failure makes a system release impossible so optimizing the current regression for achieving high efficiency in reducing the regression effort becomes justified.

It is often the case that close to the so-called start of production (SOP) of a vehicle, many electronic control units (ECU) have only some critical spots and thus each regression test is expected to be a system-release test. Since many test cases pass, a lot of time is spent in observing passing test cases. Therefore, reducing the number of executed passed test cases (since a system failure is detected and a system release is no longer possible) and keeping the limited testing time back for fault-revealing test cases decreases the regression test effort significantly. In any case, a final regression test will succeed after further system updates have been conducted; this will be constituted as a final release-test that meets the high-quality standards of [5].

In our industrial case study, we applied our selection method to a production-ready controller that implements complex networked functionalities for the protection of passengers and other road users. Therefore its test effort is immense, and hence we apply our regression test selection method for accelerating its testing phase. In Fig. 10 the right-hand side of the well-known *V-Model* (see [23]) is shown, whereas the focus is on system testing in our case study.

A hardware-in-the-loop simulator (HiL) [24] is used for validating an ECU's networked complex functionalities as well as its I/O-interaction and its robustness during voltage drops, as it provides an effective platform for testing complex real-time embedded systems.

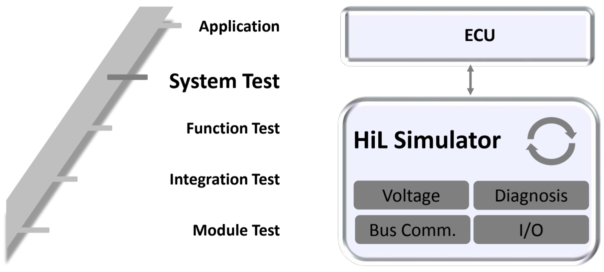


Figure 10: A HiL simulator is used for performing the system test.

Further, we selected for the following test case features for training the machine learning algorithm:

- Name of verified system parts
- Name of a function for which reliability is assured
- Number of totally involved electronic control units during the testing of a networked functionality
- Error type (broken wire etc.) in hardware robustness tests
- Set of checked diagnostic trouble codes, *DTC*s
- Number of checked diagnostic trouble codes, *DTC*s

Since the quality of our selection decisions is hedged on a stochastic level, it can appear that during different runs of our selection method, a statistical deviation of the false-positive probabilities could occur. Therefore, we constitute several independent runs of a regression test, where we set $p_{FN,MAX} = 1\%$. The boxplots and the quantiles of the false-positive probabilities are given in Fig. 11 and in Table 4, respectively.

Fig. 11 shows the overall boxplots of the false-positive probabilities achieved during the regression test replications. To compare the *hard* with the *soft decision* strategy we performed distinct regression test replications where we disabled and enabled the parameter for *'soft decision'*, respectively.

It can be seen from Fig. 11a) and Fig. 11b) that the average false-positive probability is about 74% and 23% for *hard* and enabled *soft decisions* respectively. As already mentioned, conducting *hard decisions* does not allow for global optimization of the trade-off between an already assumed risk and the corresponding number of totally deselected test cases. Global optimization hence requires the analysis of all test cases deselected thus far over and over again, and, if necessary, the selection of an already deselected test case. Therefore, test cases with a higher failure probability should be considered again for eventual selection in an ongoing regression test in order to potentially deselect further less risky test cases. As a result, the regression test effort can be reduced much more by applying *soft decisions*.

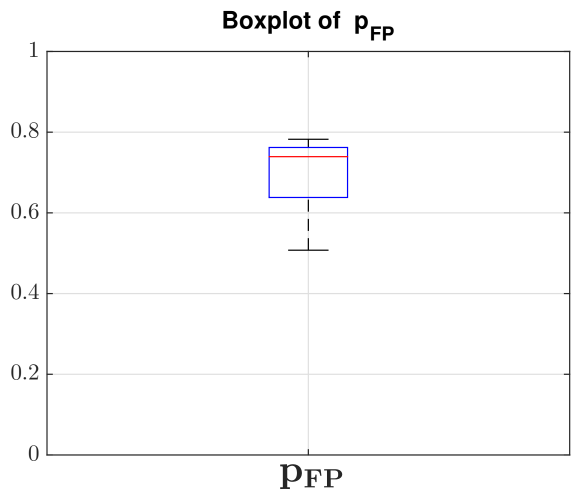
Furthermore, the condition in Eq. 40 on the false-negative probability p_{FN} or on the number of actually occurring false-negatives N_{FN} was fulfilled in all conducted regression test replications.

$$\begin{aligned} &p_{FN} \leq p_{FN,MAX} = 1\% \text{ or} \\ &N_{FN} \leq 1 \end{aligned} \tag{40}$$

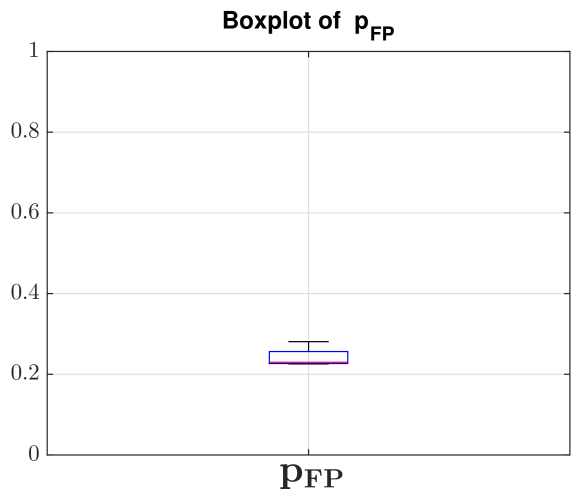
Our implemented algorithm for selecting test cases runs on a desktop CPU that is specified in Table 5. We decided to conduct a multithreaded execution of the evolutionary algorithm such that the fitness of all phenotypes in a population is computed in a multithreaded manner (in total 32 threads). Thus the average CPU load is approximately 95% and the maximum memory allocation is about 4GB. We need a mean analysis time of 0.9s for deciding whether a test case should be selected or not.

10 Conclusion and Future Work

We proposed a holistic optimization framework for the safety assessment of systems during regression testing. To this end, we designed a linear classifier for (de-) selecting test cases according to a classification due to a risk-associated recognition. Therefore we defined an optimization problem, since the classifier's specificity has to be maximized whereas its sensitivity still has to exceed a certain threshold $1 - p_{FN,MAX}$. Accordingly, we developed a novel method for determining the weights of a linear classifier that solves the above optimization problem. We have theoretically shown that the classifier performance is directly interrelated with the success of selected relevant features of test cases. Lastly, we applied our method to a production-ready controller and analyzed the overall regression test effort subject to an active learning strategy. We have demonstrated that, in the regression testing of safety-critical systems, significant sav-



(a) hard decision



(b) soft decision

Figure 11: Boxplots of false-positive probabilities in case of a) hard and b) soft decisions

Table 4: Quantiles of the false-positive probabilities in each replicated regression test

	Quantiles of False-Positive Probability				
	0.025	0.25	0.5	0.75	0.975
Hard Decision	50.8%	63.8%	73.9%	76.2%	78.2%
Soft Decision	22.6%	22.7%	23%	25.6%	28.1%

ings can be achieved. As feature selection is a complex task, and thus an evolutionary optimization supposedly finds local optima, more thorough research in this field may indeed allow higher-order reductions of the classifier's false-positive selection probability.

11 Appendix

In the following, a detailed proof of Theorem 1 is given, relating to the proofs given in [1].

Proof of Theorem 1

Proof. According to [1], the maximum likelihood estimation $x_{N,ML}$ (abbreviated x_{ML} in the following) is given in Eq. 41.

$$x_{ML} = \mu_N - \frac{(\boldsymbol{x}_D - \boldsymbol{\mu}_D)^T \boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\phi}^T}{\boldsymbol{\phi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\phi}^T} \tag{41}$$

As $\boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\phi}^T = \boldsymbol{P}_1\Sigma^{-1}\boldsymbol{P}_2^T$ and $\boldsymbol{\phi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\phi}^T = \boldsymbol{P}_2\Sigma^{-1}\boldsymbol{P}_2^T = \left(\Sigma^{-1}\right)_{N,N}$ holds (consult Proof of Theorem 3 in [1]) x_{ML} can be written as given in Eq. 42.

$$x_{ML} =: \sum_{n=1}^{N-1} w_n(x_n - \mu_n) + \mu_N \tag{42}$$

with $w_n = -\frac{\left(\Sigma^{-1}\right)_{n,N}}{\left(\Sigma^{-1}\right)_{N,N}}$. Furthermore, the threshold probability p_{th} was calculated in [1] as given in Eq. 43.

$$\begin{aligned} p_{th} &= x_N \\ &+ \frac{\sqrt{2}\sigma_N\sqrt{e^{2\mathcal{I}}-1}\operatorname{erfinv}(2p_{FN,Bound}-1) + \mu_N - x_N}{e^{2\mathcal{I}}} \end{aligned} \tag{43}$$

By introducing the definitions of the weights $w_0 := -\frac{\sqrt{2}\sigma_N\sqrt{e^{2\mathcal{I}}-1}\operatorname{erfinv}(2p_{FN,Limit}-1)}{e^{2\mathcal{I}}}$ and $w_N := -\frac{e^{2\mathcal{I}}-1}{e^{2\mathcal{I}}}$ the threshold probability can be written as $p_{TH} = -w_0 + \mu_N - w_N(x_N - \mu_N)$.

Eq. 44 derives the final definition of the hyperplane $y(\boldsymbol{x})$ and the acceptance region of the rival hypothesis H_1 by putting the definitions of x_{ML} and p_{TH} together.

$$\begin{aligned} x_{ML} &\geq p_{TH} \\ \mu_N + \sum_{n=1}^{N-1} w_n(x_n - \mu_n) &\geq -w_0 + \mu_N - w_N(x_N - \mu_N) \\ w_0 + \sum_{n=1}^{N} w_n(x_n - \mu_n) &\geq 0 \\ \underbrace{w_0 + \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu})}_{y(\boldsymbol{x})} &\geq 0 \end{aligned} \tag{44}$$

The estimation of the false-positive probability is

Table 5: Computational/memory requirements and algorithm performance

CPU	Intel Core i7-4800MQ @ 2.7 GHz, 8 Threads
CPU Load	95%
RAM	4 GB
# Allocated Threads	32 Threads
Response Time	0.9 seconds

performed by calculating $\hat{p}_{FP} = P(\hat{H} = H_1|H_0)$. However, as already explained in Sec. 6, the conditional pdf $p(\hat{H}|H)$ is not given and therefore $\hat{p}_{FP}$ cannot be directly estimated. Thus, we assume that the conditional pdf $p(\hat{H}|H)$ is of a Gaussian distribution with mean $E[\boldsymbol{X}|H_0] = \boldsymbol{\mu}_{H_0}$ and covariance matrix $E[(\boldsymbol{X} - \boldsymbol{\mu}_{H_0})(\boldsymbol{X} - \boldsymbol{\mu}_{H_0})^T|H_0] = \Sigma$. However, $\boldsymbol{\mu}_{H_0}$ is an unknown parameter vector; additionally, the second-order moments are assumed to be invariant of an event H_0. We will calculate $\hat{p}_{FP}$ in dependency on the unknown vector $\boldsymbol{\mu}_{H_0}$ and will qualitatively show that $\hat{p}_{FP}$ can be minimized independently of $\boldsymbol{\mu}_{H_0}$ due to an optimization strategy. In showing this, we demonstrate that the concept of our work is validated and mathematically proved.

First of all, Eq. 44 is given in Eq. 45 with an additional term.

$$\begin{aligned} w_0 + \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu}) &\geq 0 \\ w_0 + \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu} + \boldsymbol{\mu}_{H_0} - \boldsymbol{\mu}_{H_0}) &\geq 0 \end{aligned} \tag{45}$$

Furthermore, by introducing the definition $\Delta\boldsymbol{\mu} = \boldsymbol{\mu} - \boldsymbol{\mu}_{H_0}$ Eq. 45 can be written as given in Eq. 46.

$$\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu}_{H_0}) \geq -w_0 + \boldsymbol{w}^T\Delta\boldsymbol{\mu} \tag{46}$$

Accordingly, $\hat{p}_{FP}$ is estimated as given in Eq. 47.

$$\hat{p}_{FP} = P\Big(\boldsymbol{w}^T(\boldsymbol{X} - \boldsymbol{\mu}_{H_0}) \geq -w_0 + \boldsymbol{w}^T\Delta\boldsymbol{\mu}\Big|H_0\Big) \tag{47}$$

By substituting $U := \sum_{n=1}^{N-1} w_n(X_n - \mu_{n,H_0})$ and $V := w_N(X_N - \mu_{N,H_0})$ with $\mu_{n,H_0} = \left(\boldsymbol{\mu}_{H_0}\right)_n$ the false-positive estimation is given as follows.

$$\hat{p}_{FP} = P\Big(U + V \geq -w_0 + \boldsymbol{w}^T\Delta\boldsymbol{\mu}\Big|H_0\Big) \tag{48}$$

Based on that fact, that H_0 is given U and V are conditionally independent (see explanation in subsection 6.4.2) of each other and hence the random variable $Z := U + V$ has the mean $E[Z] = E[U] + E[V]$ and the variance $\sigma_Z^2 = \sigma_U^2 + \sigma_V^2$. Since $E[X_n|H_0] = \mu_{n,H_0}$ holds the mean of U and V is zero ($E[U] = 0$; $E[V] = 0$).

The variances of U and V are given in Eq. 49

$$\sigma_U^2 = [w_1 \cdots w_{N-1}]\Sigma_{1,1}[w_1 \cdots w_{N-1}]^T \tag{49}$$

and Eq. 50 respectively.

$$\sigma_V^2 = w_N^2\sigma_N^2 \tag{50}$$

As $[w_1 \cdots w_{N-1}]^T = -\frac{\boldsymbol{\Phi\Lambda}^{-1}\boldsymbol{\phi}^T}{\boldsymbol{\phi\Lambda}^{-1}\boldsymbol{\phi}^T}$ holds and by substituting $\beta := \boldsymbol{\Phi\Lambda}^{-1}\boldsymbol{\phi}^T$ the term $\beta^T\Sigma_{1,1}\beta$ can be simplified in Eq. 51 as already explained in the proof of Theorem 3 in [1].

$$\boldsymbol{\beta}^T\Sigma_{1,1}\boldsymbol{\beta} = -\frac{\det(\Sigma_{1,1})}{\det(\Sigma)} + \sigma_N^2\left[\frac{\det(\Sigma_{1,1})}{\det(\Sigma)}\right]^2 \tag{51}$$

It can easily be seen that the variance of U is equal to $\sigma_U^2 = \frac{\beta^T\Sigma_{1,1}\beta}{\left[\left(\Sigma^{-1}\right)_{N,N}\right]^2}$. Furthermore, $\left(\Sigma^{-1}\right)_{N,N} = \frac{\det(\Sigma_{1,1})}{\det(\Sigma)}$ holds and thus the variance is further simplified and is given in Eq. 52.

$$\sigma_U^2 = -\frac{\det(\Sigma)}{\det(\Sigma_{1,1})} + \sigma_N^2 \tag{52}$$

Finally, the false-positive probability is estimated as follows:

$$\hat{p}_{FP} = P(Z \geq -w_0 + \boldsymbol{w}^T\Delta\boldsymbol{\mu}) = \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{-w_0 + \boldsymbol{w}^T\Delta\boldsymbol{\mu}}{\sqrt{2}\sigma_Z}\right)\right] \tag{53}$$

For the case $N = 2$, Eq. 53 can be simplified and after several calculation steps the following Eq. 54 results

$$\hat{p}_{FP} = \frac{1}{2}\left[1 - \operatorname{erf}\Big(\psi\Big)\right] \tag{54}$$

with

$\psi =$

$$\frac{\sqrt{e^{2\mathcal{I}} - 1}\operatorname{erfinv}(2p_{FN,Bound} - 1) + \frac{\sqrt{e^{4\mathcal{I}} - e^{2\mathcal{I}}}}{\sqrt{2}\sigma_1}\Delta\mu_1 - \frac{\sqrt{e^{2\mathcal{I}} - 1}}{\sqrt{2}\sigma_2}\Delta\mu_2}{\sqrt{2e^{4\mathcal{I}} - 3e^{2\mathcal{I}} + 1}} \tag{55}$$

□

References

[1] I. Alagöz, T. Herpel, and R. German, "A selection method for black box regression testing with a statistically defined quality level," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, March 2017, pp. 114–125, doi: 10.1109/ICST.2017.18.

[2] F. Bürger and J. Pauli, "Representation optimization with feature selection and manifold learning in a holistic classification framework," in *Proceedings of the International Conference on Pattern Recognition Applications and Methods*, 2015, pp. 35–44, doi: 10.5220/0005183600350044.

[3] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996, doi: 10.1108/k.1998.27.8.979.4.

[4] F. Bürger and J. Pauli, "Understanding the interplay of simultaneous model selection and representation optimization for classification tasks," in *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods*, 2016, pp. 283–290, doi: 10.5220/0005705302830290.

[5] "ISO/DIS 26262-10 - Road vehicles — Functional safety," http://www.iso.org, Tech. Rep., 2012, doi: doi:10.3403/30205385.

[6] G. Xie and Z. Dang, "Model-checking driven black-box testing algorithms for systems with unspecified components," *CoRR*, vol. cs.SE/0404037, 2004. [Online]. Available: http://arxiv.org/abs/cs.SE/0404037

[7] R. Grosu and S. A. Smolka, *Monte Carlo Model Checking*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 271–286, doi: 10.1007/978-3-540-31980-1_18. [Online]. Available: https://doi.org/10.1007/978-3-540-31980-1_18

[8] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *International Conference on Runtime Verification*. Springer, 2010, pp. 122–135, doi: 10.1007/978-3-642-16612-9_11.

[9] E. Elkind, B. Genest, D. Peled, and H. Qu, *Grey-Box Checking*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 420–435, doi: 10.1007/11888116_30. [Online]. Available: https://doi.org/10.1007/11888116_30

[10] K. Sen, M. Viswanathan, and G. Agha, "Statistical model checking of black-box probabilistic systems," in *International Conference on Computer Aided Verification*. Springer, 2004, pp. 202–215, doi: 10.1007/978-3-540-27813-9_16.

[11] D. Peled, M. Y. Vardi, and M. Yannakakis, *Black Box Checking*. Boston, MA: Springer US, 1999, pp. 225–240, doi: 10.1007/978-0-387-35578-8_13. [Online]. Available: https://doi.org/10.1007/978-0-387-35578-8_13

[12] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012, doi: 10.1002/stvr.430. [Online]. Available: http://dx.doi.org/10.1002/stvr.430

[13] G. Rothermel and M. J. Harrold, "A safe, efficient regression test selection technique," *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 2, pp. 173–210, Apr. 1997, doi: 10.1145/248233.248262. [Online]. Available: http://doi.acm.org/10.1145/248233.248262

[14] A. Orso, M. J. Harrold, D. Rosenblum, G. Rothermel, M. L. Soffa, and H. Do, "Using component metacontent to support the regression testing of component-based software," in *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, 2001, pp. 716–725, doi: 10.1109/ICSM.2001.972790.

[15] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, ser. DLRS 2016, vol. abs/1606.07792. New York, NY, USA: ACM, 2016, pp. 7–10, doi: 10.1145/2988450.2988454. [Online]. Available: http://doi.acm.org/10.1145/2988450.2988454

[16] R. Langone, O. M. Agudelo, B. De Moor, and J. A. Suykens, "Incremental kernel spectral clustering for online learning of non-stationary data," *Neurocomputing*, vol. 139, pp. 246–260, 2014, doi: 10.1016/j.neucom.2014.02.036 .

[17] S. Mehrkanoon, O. M. Agudelo, and J. A. Suykens, "Incremental multi-class semi-supervised clustering regularized by kalman filtering," *Neural Networks*, vol. 71, pp. 88–104, 2015, doi: 10.1016/j.neunet.2015.08.001.

[18] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009. [Online]. Available: http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf

[19] C. Lin, M. Mausam, and D. Weld, "Re-active learning: Active learning with relabeling," 2016. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12500

[20] F. Bürger and J. Pauli, "Automatic representation and classifier optimization for image-based object recognition," in *VISAPP 2015 - Proceedings of the 10th International Conference on Computer Vision Theory and Applications, Volume 2, Berlin, Germany, 11-14 March, 2015.*, 2015, pp. 542–550, doi: 10.5220/0005359005420550. [Online]. Available: https://doi.org/10.5220/0005359005420550

[21] *Handbook of Mathematics*. Springer Berlin Heidelberg, 2007, doi: 10.1007/978-3-540-72122-2. [Online]. Available: https://doi.org/10.1007%2F978-3-540-72122-2

[22] M. Thulin, "The cost of using exact confidence intervals for a binomial proportion," *Electron. J. Statist.*, vol. 8, no. 1, pp. 817–840, 2014, doi: 10.1214/14-EJS909. [Online]. Available: https://doi.org/10.1214/14-EJS909

[23] I. Sommerville, *Software Engineering: (Update) (9th Edition) (International Computer Science)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2011.

[24] M. Schlager, *Hardware-in-the-Loop Simulation: A Scalable, Component-based, Time-triggered Hardware-in-the-loop Simulation Framework*. VDM Verlag Dr. Müller E.K., 2013.