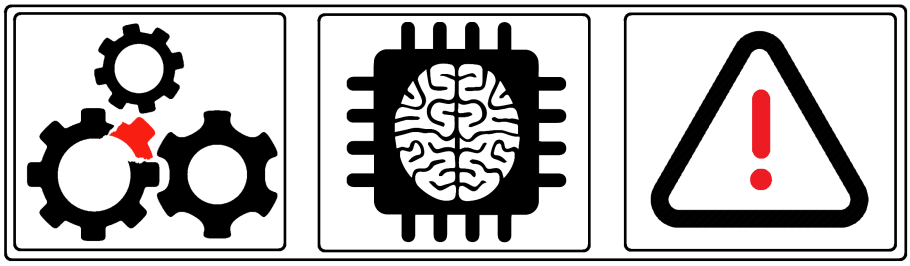
Advanced Set Bounding Methods for Fault Detection

F.R. Ritsma

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Advanced Set Bounding Methods for Fault Detection

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

F.R. Ritsma

June 14, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Advanced Set Bounding Methods for Fault Detection

by

F.R. Ritsma

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: June 14, 2019

Supervisor(s):

Riccardo Ferrari

Zaid Al-Ars

Reader(s):

Jan-Willem van Wingerden

Alfredo Nunez Vicencio

Abstract

Performance of set based fault detection is highly dependent on the complexity of the set bounding methods used to bound the healthy residual set. Existing methods achieve robust performance with complex set bounding that narrowly define healthy system behavior, yet at the cost of higher computation times. In this thesis a major improvement is reached in both accuracy and computation time by applying machine learning methods to set bounding. A method is developed which achieves fault detection at several orders of magnitude the speed of an existing set based fault detection method without sacrificing a robust performance.

Table of Contents

List of Figures

Preface

Notations and Naming Conventions

Throughout the thesis whenever an important concept is first introduced it is displayed in **bold font**. Acronyms are written out in full before they are used in shortened form.

Sets are denoted in mathematical calligraphy, e.g. $\mathcal{X}$, with an element of that set denoted with the same letter uncapitalized; $x \in \mathcal{X}$. Set elements are numbered using brackets; the n-th element of set $\mathcal{X}_y$ set is denoted $x_y\{n\}$.

Symbols denoted with a tilde $\tilde{x}$ denote hypothetical values that are obtained from a priori knowledge or Monte Carlo simulation.

Some naming conventions are in conflict e.g. x/y for both a system's state/output and a function's input/output. Variable and constant names are therefore defined after an equation if the symbol's definition has changed since its last use. To differentiate between the major fields of machine learning and control theory, variables and constants relevant to control theory all are denoted with a specialized type setting, e.g. x, y.

Of all cited sources [1] is most important and is therefore deliberately referred to with [1].

Figures which are not attributed to any source or mentioned to be in the public domain are the authors own work.

The proceeding chapters are divided into four parts. Parts I and II are introductory to fault detection and machine learning respectively, whereas parts III and IV are the authors own innovations. Part III contains incremental improvements on [1]. Part IV develops a fundamentally new approach to set based fault detection.

Acknowledgements

Several people have made the creation of this thesis an enjoyable process through their friendship and support;

Arnau, Auke, Ben, Carlo, Chris, Coen, Eva, Patrick and Rens.

For their academic assistance I want to thank my supervisors Riccardo Ferrari and Zaid Al-Ars .

Delft, University of Technology F.R. Ritsma
June 14, 2019

"I have a suggestion for you. Start using your brains."
— *prof.dr.ir. Michel Verhaegen*

Part I

Part I introduces general principles of fault detection, and offers detailed explanation of one specific approach to set based fault detection.

Chapter 1

Introduction

Many of the technical processes on which our society is built are growing evermore automated and technically complex. It is important to ensure safety and reliability for these systems that are too complicated for human supervision. This is the motivation for the field of fault detection ever since its emergence in the 1970s [2]. In that time fault detection has become invaluable in vehicle control systems, robots, transport systems, among other sectors. A fault is "any type of process degradation, or degradation in equipment performance because of changes in the process's physical characteristics, process inputs or environmental conditions" [3]. "Fault isolation" and "fault identification" are complementary fields to fault detection but have no relevance to this thesis.

Scope of Thesis

Fault detection can be approached from different directions. Fault detection using spectral data [4] is used for systems which exhibit periodic system dynamics, such as rotating machines [5]. This is distinct from fault detection using time domain data [6]. Knowledge based fault detection [7] uses heuristic symptoms and knowledge of faults from domain experts. In a survey spanning the field of fault detection, [8] mentions 10 ways of data generation and 4 ways of data evaluation.

The scope of this thesis is limited to "Set Based Fault Detection" (section 2-3) a subfield of "Model Based Fault Detection" (section 2-2). The type of fault detection developed in this thesis is characterized by being performed in the time domain, under the assumption that an accurate model of the system is available with some additional a priori knowledge on physiological uncertainties. Furthermore, there is no requirement for any knowledge on the dynamics of a faulty system, or even manifestations of faulty dynamics.

Performance Metrics in Fault Detection

The three goals for any fault detection method are to minimize the MDR, i.e. Missed Detection Rate (of faults), to minimize the FAR, i.e. False Alarm Rate, and to minimize the time

between when relevant data is available and when that data has been evaluated to conclude the presence or absence of a fault in the system, i.e. Computation Time.

To an extent all the performance metrics are mutually exclusive. Because during the majority of a system's operating no fault is present, it is required that the user be able to enforce a maximum FAR. Under the constraint that this FAR is probabilistically guaranteed, both MDR and Computation Time are minimized. In practical application, the Computation Time should on average be as fast as the time step of the system.

Starting Point of the Thesis

"A Set Based Probabilistic Approach to Threshold Design for Optimal Fault Detection" [1] by R. Ferrari, T. Keviczky and V. Rostampour was the starting point for this thesis (chapter 3). The main contribution of that work is to maximize MDR for a probabilistic guarantee on minimum FAR. This is done using complex set bounding methods to narrowly define all healthy system behavior. Those set bounding methods are computationally taxing, so the improved MDR comes at a great cost to Computation Time.

Application of Machine Learning to Set Bounding

Given the prominence of machine learning it is no surprise that some efforts have already been undertaken to apply machine learning to fault detection. Specifically supervised learning has been applied to fault detection [9]. However, this has been done for circumstances in which data is available describing system behavior in both the presence and absence of faults, which is in general a rarity for fault detection. It should also be noted that healthy and faulty dynamics can overlap, in which case the performance of a supervised learning algorithm becomes dubious. If both healthy and faulty system dynamics are known and do not overlap, it is trivial to create a binary classifier with supervised learning.

In cases outside of fault detection machine learning has been applied with very similar aims, such as in network intrusion detection or credit card fraud detection [10]. Major differences are that these methods are model free and rely on an abundance of historical data, and mostly do not depend on something analogous to a continuous state as an input. These methods are therefore not directly applicable to fault detection.

A fundamentally different mindset exists between fault detection and machine learning; whereas in fault detection a focus exists on rigorous methods of data creation, in machine learning an abundance of data is usually the starting point and the challenge exists in finding the most accurate evaluation of the data. This thesis aims to bridge the efforts of both fields, by combining the data creation of fault detection with the effective data evaluation of machine learning.

In machine learning (chapter 4) and specifically in the subfield of anomaly detection (chapter 5) a variety of set bounding methods are in use, although for similar subjects analogous language is used; set bounding is typically referred to as anomaly detection, outlier detection or one class classification. Several of these methods are adapted for fault detection (chapter 7).

Research Objective

With [1] as a benchmark for performance, success is achieved if for a FAR performance equal to or smaller than user requirements, a set based fault detection method is developed with a faster Computation Time and a MDR smaller than or equal to the benchmark method.

This thesis aims to maintain or improve the MDR performance achieved with complex set bounding methods, but drastically reduce the Computation Time required to create a bounding set by applying techniques native to anomaly detection (part III).

In addition to offering incremental improvements on a preexisting approach to fault detection, a fundamentally different approach is developed (part IV) using methods from supervised learning, a field of machine learning which could previously only be used if both data for healthy and faulty system dynamics are available. By innovating a new method of data creation a single classifier can be made which is used for many time steps, while requiring only a single training and optimization phase.

Chapter 2

Introduction to Fault Detection

This chapter serves to explain some of most general concepts in fault detection (section 2-1), followed by the principles of **model based fault detection** (section 2-2) and its subfield, **set based fault detection** (section 2-3). The importance of, and challenges to set bounding are explained in section 2-4. Because this chapter is meant to be understandable without the general introduction to the thesis, minor repetitions of the introduction exist in section 2-1.

2-1 Fault Detection in General Form

Fault detection (FD) is a subfield of control theory, where the aim is to detect faults. A **fault** is "any type of process degradation, or degradation in equipment performance because of changes in the process's physical characteristics, process inputs or environmental conditions" [3]. A system in which a fault exists, is called "faulty", or is said to exhibit "faulty behavior". The opposite to "faulty" is **"healthy"**. The terms "fault" and "error" are not to be confused. In control theory, an error is the difference between a system's current state and the desired state. This thesis concerns itself with **set based fault detection**, which is in turn a specialization of **model based fault detection**. An explanation of both model based and set based fault detection follows in sections 2-2 and 2-3.

The approach taken to fault detection is heavily dependent on what information is known to the user. Because faults are rare by nature, so is data relevant to faults. Generally, a fault is to be detected based on evaluation of the input to the system $u[t]$, output of the system $y[t]$ and whatever a priori information is available.

The two metrics for the accuracy of a FD scheme are the **Missed Detection Rate (MDR)** and the **False Alarm Rate (FAR)**. They are defined as:

$$\begin{aligned} \text{MDR} &= \frac{\text{Amount of Faults Not Reported}}{\text{Total Amount of Faults}} \\ \text{FAR} &= \frac{\text{Amount of Falsely Reported Faults}}{\text{Total Amount of Faults Reported}} \end{aligned} \tag{2-1}$$

The goal in fault detection is to have both FAR and MDR be as low as possible. FAR and MDR are equivalents of false positive and false negative rates. In practice, minimizing one quantity conflicts with minimizing the other. A scheme with a lower FAR at the cost of higher MDR is called **conservative**.

The third metric of importance is computation time. New data $y[t]$ and $u[t]$ for which a decision on whether or not a fault acts on the system is available at every time step. This requires that on average all data is analyzed in the duration of a time step.

2-2 Model Based Fault Detection

For explanations in this section, most notations are adapted from [1]. For a certain system $\mathcal{S}$ fault detection is required. System $\mathcal{S}$ is described as follows (equation 2-2):

$$\mathcal{S} : \begin{cases} x[t+1] = f_{\mathcal{S}}(x[t], u[t], p[t]) \\ y[t] = x[t] + n_y[t] \end{cases} \tag{2-2}$$

With state x, input u, output y, model parameters $p[t]$, measurement noise n_y and time step t. To perform model based fault detection, an accurate state observer $\mathcal{M}$ of the system $\mathcal{S}$ is required, see equation 2-3:

$$\mathcal{M} : \begin{cases} \hat{x}[t+1] = f_{\mathcal{S}}(\hat{x}[t], u[t], \hat{p}) + \Lambda(y[t] - \hat{y}[t]) \\ \hat{y}[t] = \hat{x}[t] \end{cases} \tag{2-3}$$

Variables with a circumflex $\hat{x}$ denote an estimate of that variable x. The model parameters $\hat{p}$ differ from the true parameters by a quantity $n_p[t] = \hat{p} - p[t]$. Λ is an observer gain that stabilizes the model.

If the observer $\mathcal{M}$ and therefore the estimate $\hat{y}[t]$ is accurate, there should be almost no difference between prediction and measurement. This leads to one of the most important concepts in model based fault detection, the **residual**. The residual is the difference between the model's prediction and the system's output (equation 2-4):

$$r[t] = y[t] - \hat{y}[t] \tag{2-4}$$

If the magnitude of the residual is not sufficiently small, a significant difference exists between $\mathcal{S}$ and $\mathcal{M}$. It is reasoned in model based fault detection that this difference between $\mathcal{S}$ and $\mathcal{M}$ is caused by a fault. This is the central concept behind model based fault detection, so it is stated with extra emphasis:

A model does not suffer physical degradation. Therefore, if predictions made by an accurate model differ too much from the system's measurements, the assumption is that this discrepancy is caused by a fault in the system.

An important point about **detectability of faults** should be made. It has been established that if a system's measurements do not conform to predictions, a fault is present. However, some faults do not manifest themselves under certain system conditions, in which case predictions and measurements conform despite the presence of a fault. An accessible example would be that of a car with faulty brakes, which are not detectable until the brakes are used. Up to that moment, predictions and measurements did conform. The logic in FD is not reversible: presence of a faulty residual proofs the presence of a fault, yet absence of a faulty residual does not proof absence of a fault.

2-3 Set Based Fault Detection

Factors other than a fault will cause a difference between $y[t]$ and $\hat{y}[t]$. For accurate fault detection it is necessary to know whether a certain residual $r[t]$ *could* have been caused by these physiological *uncertainties*. To illustrate this point, the residual is first written as a function of system dynamics:

$$r[t+1] = f_S(x[t], u[t], p[t]) + n_y[t+1] - \Big(f_S(\hat{x}[t], u[t], \hat{p}) + \Lambda(y[t] - \hat{y}[t])\Big) \tag{2-5}$$

All unknown quantities can be expressed as a function of known quantities and uncertainties:

$$r[t+1] = f_S(y[t] - n_y[t], u[t], \hat{p} - n_p[t]) + n_y[t+1] - \Big(f_S(\hat{x}[t], u[t], \hat{p}) + \Lambda(y[t] - \hat{y}[t])\Big) \tag{2-6}$$

The residual $r[t+1]$ is a function of three uncertainties: $n_p[t]$, $n_y[t]$ and $n_y[t+1]$. A Monte Carlo simulation is used to simulate all three uncertainties. A priori knowledge is required of a confidence interval or a probability distribution of the inaccuracies. For explanation purposes, consider that all inaccuracies occur within a finite domain:

$$n_p \in \mathcal{G}_p, \quad n_{y0} \in \mathcal{G}_y, \quad n_{y1} \in \mathcal{G}_y \tag{2-7}$$

with $\mathcal{G}_p$ the domain of n_p, and $\mathcal{G}_y$ the domain of n_{y0} and n_{y1}. Using this a priori knowledge a hypothetical residual can be generated:

$$\tilde{r}_H = f_S(y[t] - \tilde{n}_{y0}, u[t], \hat{p} - \tilde{n}_p) + \tilde{n}_{y1} - \Big(f_S(\hat{x}[t], u[t], \hat{p}) + \Lambda(\tilde{n}_{y0} - \tilde{n}_e)\Big) \tag{2-8}$$

with $\tilde{n}_p \in \mathcal{G}_p$, $\tilde{n}_{y0} \in \mathcal{G}_y$, $\tilde{n}_{y1} \in \mathcal{G}_y$. Values denoted with a tilde $\tilde{x}$ are hypothetical values, that is values that are not known or that cannot be computed, yet their existence can be assumed for analysis purposes. This process of creating a hypothetical residual is denoted as follows:

$$\tilde{r}_H = g(\hat{x}, \hat{p}, u, \tilde{n}_p, \tilde{n}_{y0}, \tilde{n}_{y1}) \tag{2-9}$$

For a given state-input pair $\langle x, u \rangle$, a set $\mathcal{G}_r \in \mathbb{R}$ exists, which is the set of all hypothetical healthy residuals for that state-input pair:

$$\mathcal{G}_r = \{g(\hat{x}, \hat{p}, u, \tilde{n}_p, \tilde{n}_{y0}, \tilde{n}_{y1}) : \tilde{n}_p \in \mathcal{G}_p, \tilde{n}_{y0} \in \mathcal{G}_y, \tilde{n}_{y1} \in \mathcal{G}_y\} \tag{2-10}$$

By collecting sufficient hypothetical residuals, i.e. performing Monte Carlo simulation, the space $\mathcal{G}_r$ is progressively filled with samples until the space of all healthy residuals has been approximated with a finite set. This set is called the **healthy residual set**, and is denoted with $\mathcal{R}_H$. The creation of the healthy residual set is performed as follows:

$$\begin{aligned}
\mathcal{R}_H &= \langle g(\hat{x}, \hat{p}, u, \mathcal{N}_p, \mathcal{N}_{y0}, \mathcal{N}_{y1}) \rangle \\
\mathcal{N}_p &= \langle \tilde{n}_p\{1\}, \tilde{n}_p\{2\}, ..., \tilde{n}_p\{s\} \rangle \\
\mathcal{N}_{y0} &= \langle \tilde{n}_{y0}\{1\}, \tilde{n}_{y0}\{2\}, ..., \tilde{n}_{y0}\{s\} \rangle \\
\mathcal{N}_{y1} &= \langle \tilde{n}_{y1}\{1\}, \tilde{n}_{y1}\{2\}, ..., \tilde{n}_{y1}\{s\} \rangle
\end{aligned} \tag{2-11}$$

with s the sample count. For an increasing s, $\mathcal{G}_r$ is progressively more accurately approximated by $\mathcal{R}_H$.

2-4 Set Bounding Methods and Their Influence on Performance

In brief summary, it has been explained how residuals can be indicative of a faulty system, and how to create a healthy residual set $\mathcal{R}_H$ which approximates a region of residuals associated with healthy system behavior. However, $\mathcal{R}_H$ is a discrete set of randomly sampled points, so a received residual r will never be exactly equal to any element in $\mathcal{R}_H$. Formally speaking, the received residual will never be in the healthy residual set: $r \notin \mathcal{R}_H$. Instead, a **bounding set** $\mathcal{B}$ is created, an open set such that:

$$r \in \mathcal{B}, \quad \mathcal{R}_H \in \mathcal{B} \tag{2-12}$$

In the contents of this thesis, all bounding sets can be expressed in the form:

$$\mathcal{B} = \{x \in \mathbb{R} : f_{\mathcal{B}}(x, c) \geq \tau\} \tag{2-13}$$

Where $\mathcal{B}$ is defined by some set bounding function $f_{\mathcal{B}}(x, c)$, with parameters c. It should be noted that there is some analogous language on this subject in FD and ML. In ML, the set bounding function $f_{\mathcal{B}}$ is typically called a **classifier**. The output of this classifier is known as an "outlier score" or "outlier probability". Further attention is given to the subject in chapters 4 and 5.

2-4-1 Optimization Goals for Bounding Sets

For the creation of this bounding set there are two conflicting goals. $\mathcal{B}$ should include *all* possible healthy residuals at the risk of causing false alarms (increasing FAR), yet simultaneously $\mathcal{B}$ should *only* include healthy residuals, at the risk of missing detections (increasing MDR).

As was mentioned previously, the accuracy of a FD scheme is determined by its MDR and FAR. The FAR is more easily evaluated than MDR, because healthy residuals can be generated (see section 2-3) with which FAR can be established. Due to their nature faults are rare, and consequently data relevant to faulty system dynamics is also rare. It is not known how and with which frequency faults manifest, so it is not possible to guarantee a general detection rate. However, [1] introduces set volume $\text{Vol}(\mathcal{B})$ as a heuristic for MDR performance. Set volume is defined as:

$$\text{Vol}(\mathcal{B}) = \int_{\mathcal{B}} dx \tag{2-14}$$

In [1] MDR performance is used as a heuristic measure proportional to set volume. For lower set volumes more residuals are classified as faulty. If, for a constant FAR, set volume decreases it can be expected but *not guaranteed* that MDR decreases.

Under the constraint of a minimal FAR, attaining minimal MDR can be seen as an optimization problem:

$$\begin{aligned} &\min \text{Vol}(\mathcal{B}) \\ &\text{subject to: } \mathcal{R}_H \in \mathcal{B} \end{aligned} \tag{2-15}$$

The optimization problem posed in equation 2-15 is simplified for explanation purposes in the following section. In chapters 3 and 7 a more detailed view is taken.

2-4-2 Performance Limitations of Simple Set Bounding Methods

So far, it has been explained that a bounding set $\mathcal{B}$ is required to perform set based fault detection. It is explained in this section how simple set bounding methods limit the performance of FD in either FAR and/or MDR.

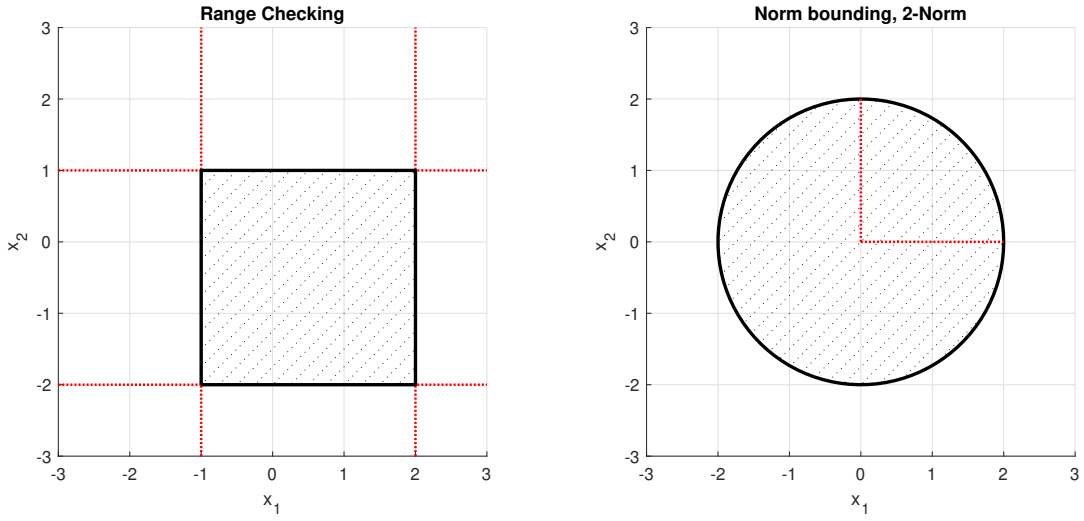
Two often used methods for set bounding are range checking and 2-norm bounding. The bounding set for range checking is defined as:

$$\mathcal{B} = \{x \in \mathbb{R}^D : \tau_d^{min} \leq x \leq \tau_d^{max}, \forall\, 1 \leq d \leq D\} \tag{2-16}$$

The bounding set for 2-norm bounding is defined as:

$$\mathcal{B} = \{x \in \mathbb{R}^D : ||x||_2 \leq \tau\} \tag{2-17}$$

With D the dimension, and τ a threshold. Both methods of set bounding lead to a very specific shape of bounding set. For range checking, the shape of the bounding set is a hypercube with the location of the vertices set by the threshold values. The bounding set in 2-norm bounding is a hypersphere of radius τ. This is visualized in figure 2-1:



(a) Range checking, bounding set **(b)** 2-Norm bounding set

Figure 2-1: Visualization of range checking (left) and 2-norm bounding (right)

Although the location of the vertices can be changed (range checking) or the radius of the hypersphere (2-norm bounding) the fundamental shape of these sets can not be changed.

To understand the drawbacks of this, consider the healthy residual set in figure 2-2 as an example:

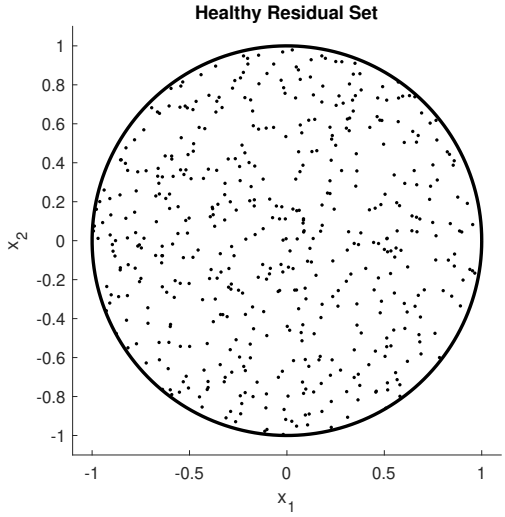


Figure 2-2: A healthy residual set $\mathcal{R}_H$ represented with black points and the ideal bounding set $\mathcal{B}$, the boundary of which is displayed with a black line.

The healthy residual set $\mathcal{R}_H$ in figure 2-2 is deliberately created in a circular pattern. Predictably, a bounding set $\mathcal{B}$ created with 2-norm bounding fits the data very well (black line). Consider however the resulting bounding set created by range checking 2-3:

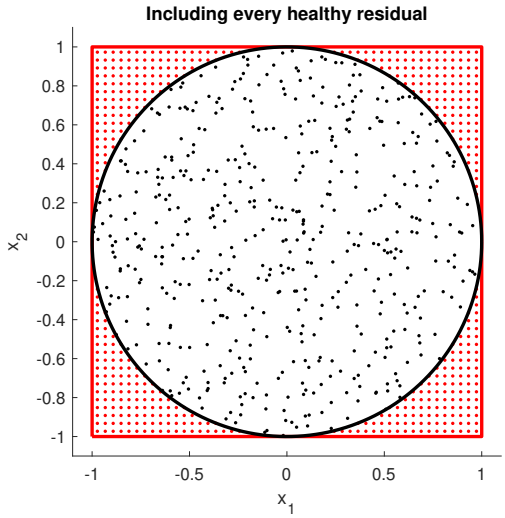


Figure 2-3: Bounding $\mathcal{R}_H$ using range checking. The set boundaries of the range checking set are indicated with red lines. The red dots indicate residuals erroneously classified as healthy.

All healthy residuals are included in the bounding set $\mathcal{R}_H \in \mathcal{B}$, but it can be seen in the corners that a region indicated by red dots is erroneously included. These represent residuals that would be erroneously classified as healthy, causing a missed detection (MDR). Consider the user prioritizes a decrease in MDR. Figure 2-4 depicts a bounding set that achieves this:

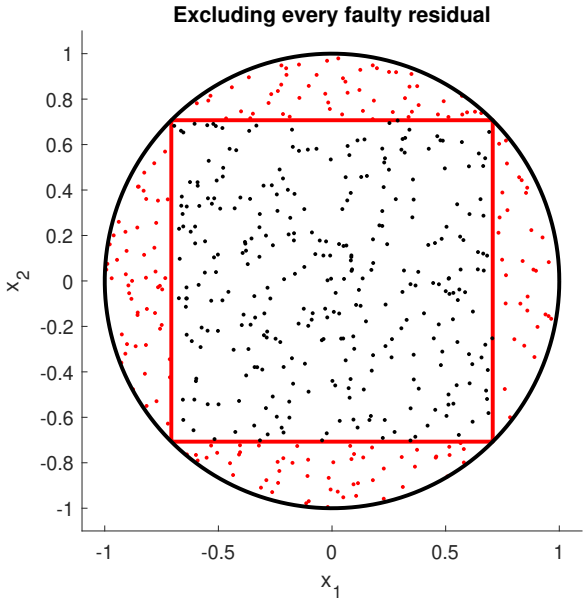


Figure 2-4: A bounding set $\mathcal{B}$ excluding every faulty residual, but misclassifying some healthy residuals (red dots)

Indeed, the same faulty residuals that were classified as healthy, are now no longer included in the bounding set. Yet it can be seen that some of the healthy residuals, indicated by red dots, are now classified as faulty causing false alarms (FAR).

Because in the examples of figures 2-3 and 2-4 the shape of the bounding set $\mathcal{B}$ was fundamentally misaligned with the shape of $\mathcal{R}_H$, a situation arises in which either FAR and/or MDR performance is suboptimal, and improving performance in one metric is at the cost of performance in the other. Summarizing, to attain good performance in FAR and MDR, a bounding set $\mathcal{B}$ needs to match the shape of healthy residual set $\mathcal{R}_H$.

In the previously used examples the shape of $\mathcal{R}_H$ was a circle, but in practical applications this shape is more complex. The set shape of $\mathcal{B}$ needs to be complex in response. Xiong [11] lists ellipsoids, parallelotopes, polytopes and zonotopes as more complex methods of bounding sets, yet even these methods result in comparatively simple shapes. Forming bounding sets of increased complexity is one of the results achieved in [1], as explained in chapter 3.

2-4-3 Outliers and Robust FAR Performance

An additional challenge arises from healthy residual sets with **outliers**. An outlier is a sample that is not characteristic of the set. A more narrow definition depends on context. An example of a healthy residual set with outliers is given in figure 2-5:

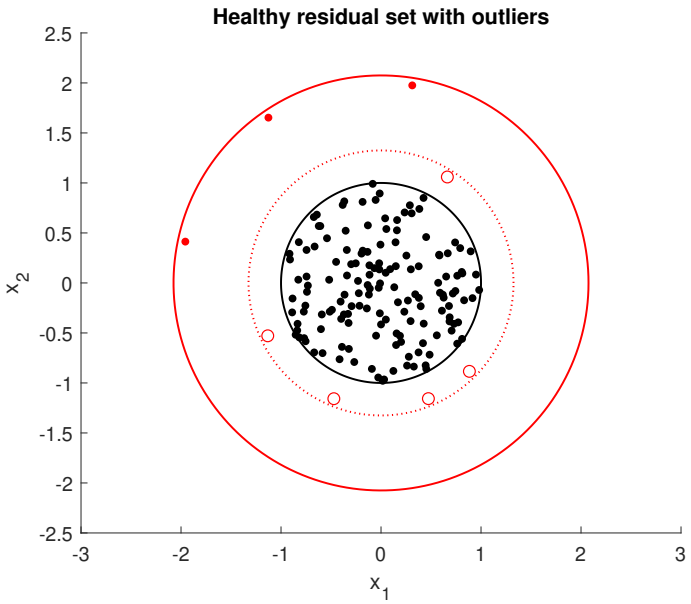


Figure 2-5: Example of a healthy residual set with outliers depicted in red.

In figure 2-5 the black dots represent those samples that are without controversy characteristic of the set. The red dots, if included in set bounds, would drastically increase the Vol($\mathcal{B}$). Whether to include the red hollow dots is a less obvious choice. The presence of outliers calls for a performance which is FAR robust. A requirement to include *all* possible healthy residuals is in such a circumstance too strict, and will deteriorate MDR performance for only minor improvements in FAR.

In [1] deals with outliers by creating a bounding set which is probabilistically α robust. The concept is explained in closer detail in chapter 3. In this thesis the problem is handled differently, by introducing a new sampling technique (chapter 6) and by setting a threshold on test data performance (chapter 7).

Chapter 3

Detailed View of Existing Set Based Fault Detection

To evaluate performance of a novel approach in contrast to established methods it is first needed to understand the starting point of this thesis. As has been mentioned previously, the research objective is to improve on "A set based probabilistic approach to threshold design for optimal fault detection" by Vahab Rostampour, Riccardo M.G. Ferrari and Tamas Keviczky [1]. This chapter presents the most important subjects of the paper with relevance to the thesis.

3-1 Polynomial Super Level Sets

It was explained in section 2-3, that if the shape of $\mathcal{B}$ does not align with the shape of set $\mathcal{R}_H$ FAR and/or MDR performance deteriorates as a consequence. All of the mentioned examples are limited in the shapes they can take on. A major innovation in [1] is the more complex set shapes used, achieved with polynomial super level sets. A polynomial super level set is defined as follows:

$$\mathcal{B} = \{x \in \mathbb{R}^n : f(x, c) \geq \tau\} \tag{3-1}$$

With $f(x, c)$ a polynomial function, and c the coefficients of the polynomial function. The shape of $\mathcal{B}$ can be made more complex by increasing the degree of the polynomial. Coefficients c depend on $\mathcal{R}_H$, as will be explained in section 3-2. The set bounding methods mentioned in section 2-3 are restricted to being convex and connected, reducing their complexity. Polynomial super level sets need be neither convex nor connected. As a set bounding method is is referred to as **super level set bounding (SLSB)** throughout the thesis.

3-2 Optimization Procedure to Compute Super Level Sets

This section serves to explain the method in which the coefficients c of the polynomial $f(x, c)$ are computed. In its most rudimentary form c is the result of a minimization problem:

$$\begin{aligned} &\arg\min_{c}(\text{Vol}(\mathcal{B})) \\ &\text{subject to: } \mathcal{R}_H \in \mathcal{B} \end{aligned} \tag{3-2}$$

The notation in equation 3-2 is deliberately verbose. Because this section is rife with detail, it serves to emphasize the end goal of the optimization before a more in depth explanation is offered. The method for creating super level sets in [1] is based on work by Dabbene and Henrion [12]. Using constraints and approximations the minimization in equation 3-2 is made into a linear programming problem. A linear program is a minimization problem with linear constraints and a linear objective which takes the following form (equation 3-3):

$$\begin{aligned} &\min_{z}(c^T z) \\ &\text{subject to: } Az \leq b, \qquad z \geq 0 \end{aligned} \tag{3-3}$$

The function $f(x, c)$ is separated into monomials and coefficients. A monomial is a product of variables only, $x^{n_1}y^{n_2}z^{n_3}$, where the sum of the exponents $n_1 + n_2 + n_3$ is the degree of the monomial, n. The function $f(x, c)$ is a sum of monomials multiplied by coefficients, as in equation 3-4:

$$f(x,c) = \begin{bmatrix} x^2 & xy & x & y^2 & y & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = c_1x^2 + c_2xy + c_3x + c_4y^2 + c_5y + c_6 \tag{3-4}$$

It can be seen that $f(x, c)$ is linear in the parameters. The following notation is used to denote $f(x, c)$:

$$f(x,c) = m(x)c^T \tag{3-5}$$

With $m(x)$ the monomials in $f(x, c)$. The first constraint is:

$$m(\tilde{r}_H)c^T \geq \tau, \quad \forall\ \tilde{r}_H \in \mathcal{R}_H \tag{3-6}$$

Which guarantees inclusion of $\mathcal{R}_H \in \mathcal{B}$. The function $f(x, c)$ is required to be non negative. This is enforced by sampling in a grid around the set $\mathcal{R}_H$. With these samples $x_G \in \mathcal{X}_G$ non negativity is enforced:

$$m(x_G)c^T \geq 0, \quad \forall\ x_G \in \mathcal{X}_G \tag{3-7}$$

Equations 3-6, 3-7 complete the constraints for the linear program. The objective function is the minimization was stated as Vol($\mathcal{B}$). This volume is not linearly dependent on the coefficients c. However, an upper bound is formulated on Vol($\mathcal{B}$):

$$\int_{\mathcal{R}_H} f(x,c)dx \geq \text{Vol}(\mathcal{B}) \tag{3-8}$$

Where the integration of $f(x)dx$ is approximated with the integration of $f(x)dx$ over a bounding box. The upper bound $\int_{\mathcal{R}_H} f(x)dx$ on Vol($\mathcal{B}$) is linearly dependent on parameters in c, as is illustrated by integrating a monomial in equation 3-9

$$\int_{y_0}^{y_1}\int_{x_0}^{x_1} c_n x^a y^b dxdy = \frac{c_n}{(a+1)(b+1)}(x_1^{a+1} - x_0^{a+1})(y_1^{b+1} - y_0^{b+1}) \tag{3-9}$$

The integrated monomials are denoted as:

$$M(x_1 - x_0) = \int_{x0}^{x_1} m(x)dx \tag{3-10}$$

All constraints and the objective linearly depend on the coefficients in the polynomial $f(x, c)$. The optimization problem as a linear program is as follows:

$$\begin{aligned} &\arg\min_{c^T}([M(\max(\mathcal{R}_{H,i}) - \min(\mathcal{R}_{H,i}))]c^T) \\ &\text{subject to: } \begin{bmatrix} m(\tilde{r}_H) \\ m(x_G) \end{bmatrix} c^T \geq \begin{bmatrix} \tau \\ 0 \end{bmatrix},\ \forall\ r_H \in \mathcal{R}_H,\ \forall x_G \in \mathcal{X}_G \end{aligned} \tag{3-11}$$

3-3 Summarizing Results

It is explained in [1] why lower set volume of the bounding set $\mathcal{B}$ is expected to decrease MDR. SLSB is successfully applied to reduce set volume Vol($\mathcal{B}$) on healthy residual sets created using a simulated three tank system, a test case which is often used as a bench mark for fault detection methods (see section 8-1-2).

Having established the success of polynomial super level sets, this thesis seeks to improve on the results in [1] by evaluating different set bounding methods. Improvement is sought in the minimization of computation time required to compute set bounds and minimization of set volume Vol($\mathcal{B}$).

Some differences in approach between this thesis and [1] should also be noted. In [1] the concept of α-robustness is introduced. α is a parameter which governs the probability of future healthy residuals being bounded by $\mathcal{B}$, with probability of violation $1 - \alpha$. A relation is given between α, ξ the degree of polynomial $f(x, c)$, and s the sample count. It is how the paper deals with outliers (section 2-4-3) by not requiring all possible healthy residuals to be bounded by $\mathcal{B}$. The concept of α-robustness is not continued in the thesis, though different approaches are taken to ensure a robust FAR performance.

It has already been noted that this thesis is strictly concerned with fault detection and not fault isolation. For fault isolation, faulty residual sets $\mathcal{R}_F$ are used. Some attention is given to the subject in [1], but it will not be considered in this thesis. Chapter 6 explains the different approach taken in the thesis for sampling healthy residual sets. A different approach is taken when evaluating volume Vol($\mathcal{B}$), as is explained in chapter 7.

Part II

Part II explains machine learning terminology relevant to the thesis. An overview is given of machine learning algorithms applicable to fault detection.

Chapter 4

Machine Learning Procedures and Terminology

The aim of this thesis is to improve on current methods (chapter 3) using machine learning for set bounding. Before introducing specific methods of machine learning in chapter 5, an explanation of relevant terminology is provided and some widely followed procedures are explained.

4-1 Introduction to Machine Learning

The field of machine learning has become ever more prominent in many applications. In academia this is evidenced by an annually growing prominence of the term in publications (figure 4-1).

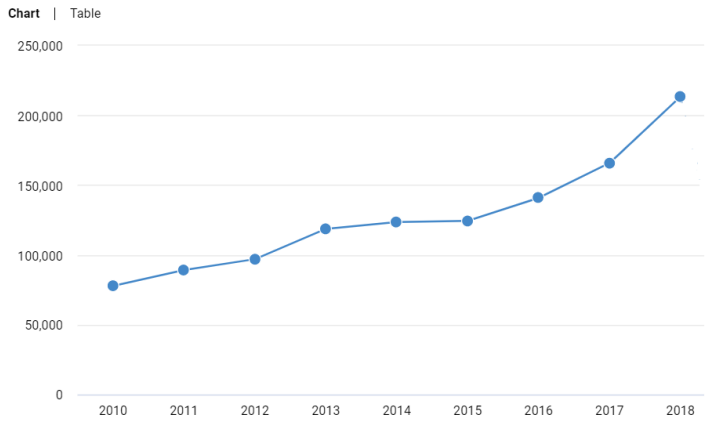


Figure 4-1: Number of annual publications in which the term "machine learning" is used (From app.dimensions.ai/analytics).

In industry the same pattern holds. Global funding for AI based companies has grown annually by at least 15 percent from 2012 to 2016 (figure 4-2):

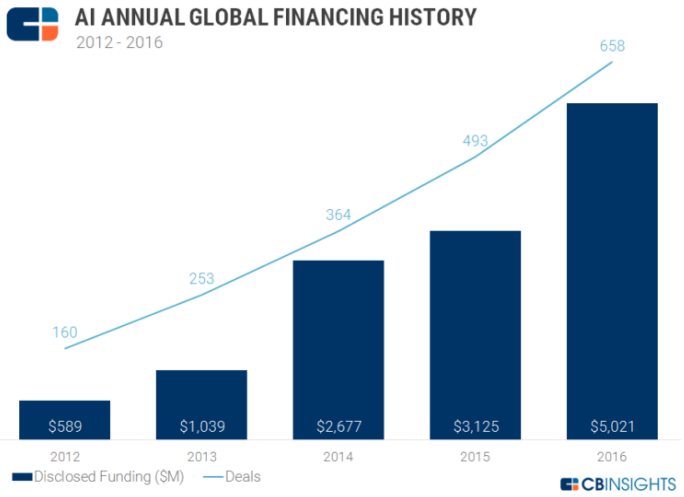


Figure 4-2: Annual funding for AI based companies. A steady increase is seen of at least 15 percent between years (From aiimpacts.org).

In view of this rise to prominence, it is evident that contributions of machine learning to the field of fault detection can be expected. Machine learning is defined in [13] as follows:

"Machine learning is programming computers to optimize a performance criterion using example data or past experience"

This optimizing of a performance criterion is referred to interchangeably as "learning" or "training". It is clear that in set based fault detection the "example data or past experience" is a healthy residual set. In view of this, set based fault detection can be expected to overlap greatly with machine learning.

4-2 Relevant Subfields

As machine learning is a very broad field of study, it is useful to define which sub-disciplines of machine learning are applied in this thesis.

Machine learning can be parsed in multiple ways, e.g. by application or method. The larger categories of machine learning are **regression** versus **classification** and **supervised learning** versus **unsupervised learning**. What type of machine learning is to be used depends on the availability of data and the desired output. In regression, outputs are a continuous quantity. In classification, the output is a label. For fault detection, the aim is to classify a residual with a label "healthy" or "faulty", making fault detection a problem of classification.

In supervised learning, data consists of an input X for which a desired output Y is known. The supervised method is trained to recognize and reproduce the relation between X and Y. In unsupervised learning, the desired output for a set X is not known. In this thesis it is assumed that only a healthy residual set is available. This calls for unsupervised machine

learning as the output of the method needs to identify residuals as either "faulty" or "healthy" having only been trained on examples of healthy system behavior.

4-3 Anomaly Detection

Anomaly detection (AD), known under several synonyms (outlier detection, surprise detection, discord detection, etc. [14]), is a subfield of machine learning where the goal is to identify anomalies in a data set or future observations. For fault detection, this data set is $\mathcal{R}_H$, and future observations are residuals r. Anomaly detection is used to in credit card fraud detection, insurance fraud detection, intrusion detection in cyber-security, among other applications [15]. An anomaly is defined in [16] as:

An anomaly is an observation or event that deviates qualitatively from what is considered to be normal, according to a domain expert.

This definition may seem rather broad, but a more narrow definition of anomaly is highly dependent on the field of application. Anomaly detection can be performed using either supervised or unsupervised methods, but the nature of fault detection requires unsupervised methods be used (section 4-2). "Precision" and "recall" are often used in one class classification as a performance metric, see equation 4-1.

$$\begin{aligned} \text{FAR} &= \frac{fp}{tp+fn} \\ \text{MDR} &= \frac{fn}{tn+fp} \\ \text{Precision} &= \frac{tp}{tp+fp} \\ \text{Recall} &= \frac{tp}{tp+fn} \end{aligned} \tag{4-1}$$

With fp, tp being false- and true positives, and fn, tn being false- and true negatives. The notation given in equation 4-1 for MDR and FAR is consistent with the definition provided earlier in chapter 2. It is decided to maintain FAR and MDR as performance metrics throughout the thesis.

Several surveys were consulted to determine which anomaly detection methods are suitable for fault detection ([17], [18], [19],[20]). A detailed overview of methods deemed suitable is given in the next chapter.

4-4 General Procedures and Terminology

Chapters discussing proposed machine learning methods (chapter 5) or performance measures (chapter 7) require understanding of terminology used in machine learning. To introduce all relevant terminology, the process of using an arbitrary anomaly detection method is described step by step.

The first step in machine learning is the acquisition and preprocessing of data. Several steps are not relevant for anomaly detection such as de-noising [21], or data augmentation. In [22]

it is advised that data first be scaled to a hyperbox with edges [-1,1] or [0,1]. Incidentally, this is also applied in [1], where the data is scaled to a hyperbox with edges [-1,1]. In the thesis the data is scaled to hyperbox with edges [0,1]. Scaling data improves performance if the data is unbalanced, meaning the variance of the data is much larger in one dimension than in another.

After data preprocessing, the available data is randomly divided into a **train set**, **test set**, and **validation set**. The train set is used to train on. The test set is used to check if whether the trained method is **overfit**. A method which is overfit, has learned features from the train set which are specific to that train set only, rather than a general truth. Typically this means a good performance on the train set, but a bad performance on the test set. Conversely, a method that has not learned enough is **underfit**, manifesting in a bad performance on both the train and test set. The validation set is a hybrid between the train and test set. The validation set is not used to train on directly, but is used during training to verify whether an update on trainable parameters improves performance outside train data.

A machine learning method has two kinds of parameters: **hyper parameters**, and **trainable parameters**. Hyper parameters are set by the user before training, and do not change during training. Trainable parameters are adapted during training to optimize performance. Hyper parameters often control the complexity of the machine learning algorithm. Machine learning algorithms with to much complexity are liable to overfitting. This can be combated by either decreasing the complexity, or increasing the amount of train data. To find the optimal hyper parameter settings, **hyper parameter optimization** may be performed. Alternatively, because hyper parameter optimization is computationally expensive, a **hyper parameter heuristic** can be used. A heuristic is a less computationally expensive method of determining hyper parameters, often based on e.g. sample count or variance of the data.

Chapter 5

Anomaly Detection Methods

In chapter 4 machine learning and specifically anomaly detection were proposed for application to fault detection. This chapter provides detailed explanations of methods which are considered suitable to set bounding, followed by a more brief overview of rejected methods (section 5-4). Test results of suitable methods are included in chapter 8.

5-1 Parzen windows

The **Parzen window** (**PW**), also known as Parzen-Rosenblatt window, or Kernel Density Estimation, is a way of estimating the probability density function of a random variable. This is done by summing kernels (also known as window functions) centered on samples of that random variable.

A slightly different definition exists for kernels when applied to statistics (PW) compared to OSVM (see section 5-2). However, the kernels used throughout the thesis fit both the definitions used in section 5-1 as well as the definition used in section 5-2. For this reason, the term kernel will be used throughout both sections. In the context of kernel density estimation, a kernel is a function which adheres to the following definitions:

$$\begin{aligned} &\int_{-\infty}^{+\infty} K(x)dx = 1 \\ &K(\mu - x) = K(\mu + x) \\ &K(x) \geq 0, \forall x \end{aligned} \tag{5-1}$$

Meaning a kernel has an integral of 1 over the entire domain, is symmetrical over the mean μ, and is non negative over the entire domain. In this thesis the Epanechnikov-, Triangular- and Gaussian kernels are considered. Mathematical equations for these functions are stated

below (equation 5-2):

$$\begin{aligned} K_{\text{Gaussian}}(x,\mu,\sigma) &= \frac{1}{\sqrt{|\Sigma|(2\pi)^d}}\exp(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T), \quad \Sigma = \sigma I \\ K_{\text{Epanechnikov}}(x,\mu,\sigma) &= \max(1-\sigma(x-\mu)(x^T-\mu^T),0) \\ K_{\text{Triangle}}(x,\mu,\sigma) &= \max(1-\sigma\sqrt{(x-\mu)(x^T-\mu^T)},0) \end{aligned} \tag{5-2}$$

With mean μ, kernel size σ -also called bandwidth and d dimension of $x \in \mathbb{R}^d$ for K_{Gaussian}. When using a PW the assumption is that elements of the train set $\mathcal{X}_{train}$ are samples of an unknown probability density function $p(x)$. A reconstructed probability density function $\hat{p}(x)$ is created as follows:

$$\hat{p}(x) = \frac{1}{N}\sum_{n=1}^{N} K(x, \mathcal{X}_{train}\{n\}, \sigma) \tag{5-3}$$

With N the amount of samples in train set $\mathcal{X}_{train}$. In the context of fault detection the reconstructed probability function $\hat{p}(x)$ is a residual's probability of being the result of healthy system dynamics. The bounding set is defined as follows:

$$\mathcal{B} = \{x \in \mathbb{R}^n : f_{\text{PW}}(x) > \tau\} \tag{5-4}$$

With $f_{\text{PW}}(x) = \hat{p}(x)$ instead of $\hat{p}(x)$ for a uniform notation throughout the chapter. Parameter τ is a user set threshold. The single hyper parameter in PW is σ, the bandwidth of the window function. Because decreasing the size of σ is equivalent to moving from underfitting to overfitting, σ can be smaller if more data is available (see also section 7-3-1). In PW no trainable parameters exist.

Illustrated Example

To illustrate the workings of a PW the distribution in equation 5-5, illustrated in figure 5-1, will be recreated. A multivariate Gaussian distribution p_E is described with:

$$p_E(x, \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} \exp(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T),$$
$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{5-5}$$

This probability function is depicted in the following figure (5-1):

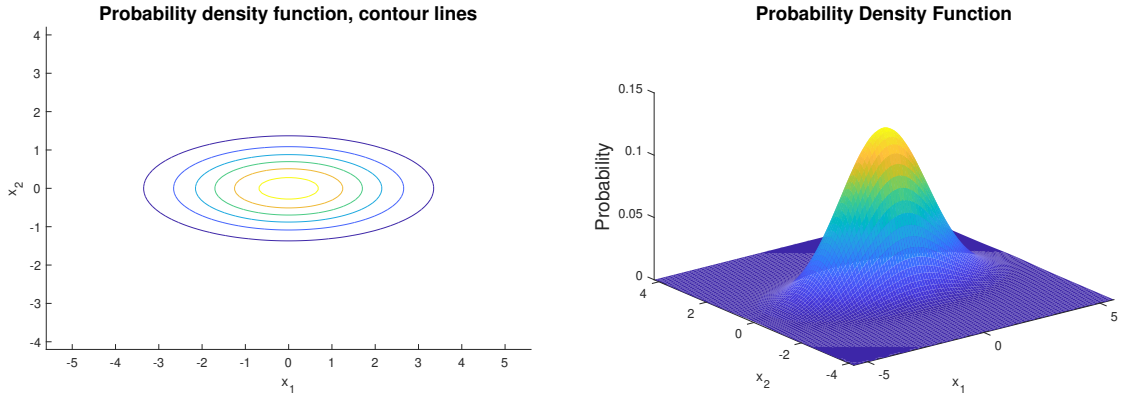


Figure 5-1: Distribution of the random variable

In the following pictures it can be seen how this distribution is reconstructed with increasing accuracy for an increasing amount of train samples. A Gaussian kernel is used for reconstruction.

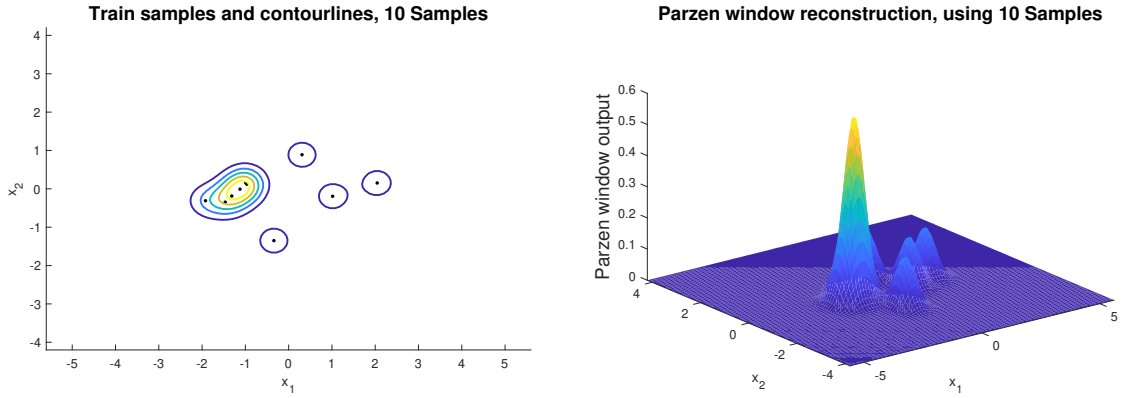


Figure 5-2: Kernel density estimation, example 1

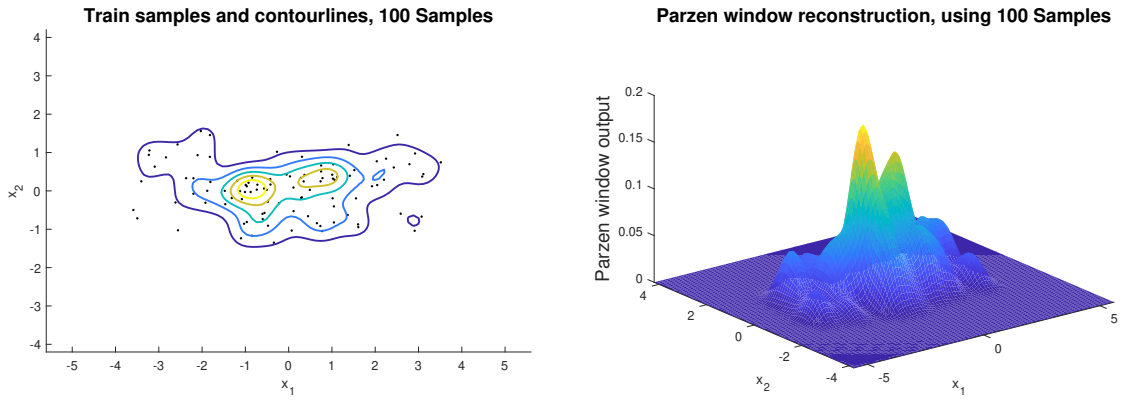


Figure 5-3: Kernel density estimation, example 2

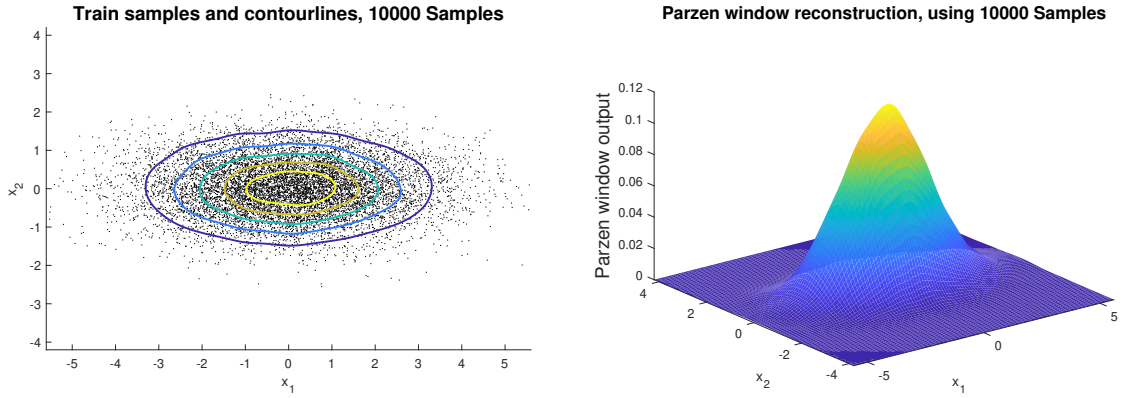


Figure 5-4: Kernel density estimation, example 3

It can be seen that an initially too small kernel size becomes more accurate for an increasing amount of train samples, illustrating the relation between overfitting, complexity and sample size.

5-2 One Class Support Vector Machine

One Class Support Vector Machines (**OSVM**) are an adaptation of Support Vector Machines (SVM) for anomaly detection. SVMs were originally devised for binary classification of data with a linear classifier. The goal is to find the line or hyperplane which most optimally separates two classes. Of the two classes, one is assigned the label 1, the other is assigned the label -1. To understand the minimization problem that results in the optimal separation of classes, it is helpful to understand the **hinge loss function**, which is explained in the following section (section 5-2-1). The minimization of the SVM specifically is explained in section 5-2-2.

5-2-1 Hinge Loss for Classification

Loss functions are an integral part of machine learning in both regression and classification. Suppose some function $f(x, c) = \hat{y}$, with input x and tuning parameters c, is to be used for either regression or classification. The user has a data set of inputs $\langle x_1, x_2, x_{...}, x_n \rangle$ and for those inputs associated outputs $\langle y_1, y_2, y_{...}, y_n \rangle$. A loss function is the minimization problem which when solved returns as arguments of the minimum the optimal parameters for $f(x, c)$ such that the function performs the desired task optimally.

In classification purposes the **hinge loss** is widely used. To illustrate why it is well suited to classification it is contrasted to the **mean squared error loss** function (equation 5-6), which is used for regression problems:

$$\begin{aligned} L_{\text{MSE}} &= \frac{1}{N} \sum_{n=1}^{N} ((\hat{y}\{n\} - y_{train}\{n\})^2) \\ \hat{y}\{n\} &= f(x_{train}\{n\}, c) \end{aligned} \tag{5-6}$$

with for input values $x_{train} \in \mathcal{X}_{train}$ corresponding output values $y_{train} \in \mathcal{X}_{train}$. The aim in regression is that the function's output $\hat{y}$ and true output y have as little difference as possible. This difference is minimized in minimizing the mean square error loss.

In binary classification problems an exact value is not required. Typically, one class is assigned the negative label $l = -1$ and the other class is assigned the positive label $l = +1$. A function is required that for inputs of the negative class returns a negative value, and vice versa for the positive class. In contrast to the regression problem, any negative/positive value will do, provided it outputs for the corresponding class. To maximize a margin between the classes negative train samples are optimized to have an output equal to or lesser than -1, positive inputs a value equal to or greater than $+1$. This can be achieved minimizing the hinge loss function:

$$\begin{aligned} L_{\text{hinge}} &= \sum_{n=1}^{N} (\max(0, 1 - \hat{y}\{n\} \cdot l_{train}\{n\})) \\ \hat{y}\{n\} &= f(x_{train}\{n\}, c) \end{aligned} \tag{5-7}$$

with for input values $x_{train} \in \mathcal{X}_{train}$ corresponding output labels $l_{train} \in \mathcal{L}_{train}$. The hinge loss returns a positive value when $f(x, c)$ outputs a value on the incorrect side of a margin, i.e. (equation 5-8):

$$\begin{aligned} f(x_n, c) &< +1, \quad l_n = +1 \\ f(x_n, c) &> -1, \quad l_n = -1 \end{aligned} \tag{5-8}$$

If a function $f(x, c)$ returns an output on the correct side of the margin, it is of no importance by how much distance it is placed on the correct side. This is reflected by the $\max(0, x)$ operation. Any output $f(x, c)$ with a correct output beyond the margin is set by the max operation at zero. This decreases the complexity of the optimization. By not following the strict requirement $f(x_{train}\{n\}, c) = l_{train}\{n\}$, a greater amount of functions can fit to the data resulting in an easier optimization problem.

5-2-2 SVM Minimization Problem

As was mentioned previously, the aim of the SVM is to separate two classes with the widest possible margin using a hyperplane. Figure 5-5 depicts two classes and the line which separates them with the widest possible margin.

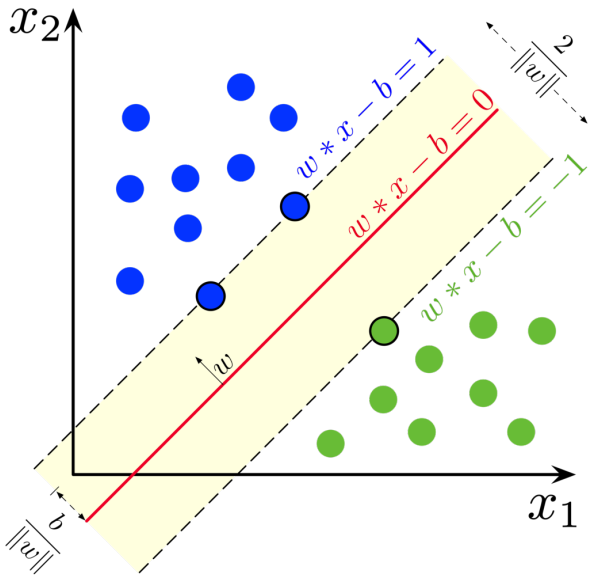


Figure 5-5: SVM separating two classes with the widest possible margin (Public Domain Image)

Mathematical notations in figure 5-5 are continued in this section. The optimal hyperplane is expressed in the form:

$$w \cdot x - b = 0 \tag{5-9}$$

The goal is to maximize the distance between hyper planes $w \cdot x - b = 1$ and $w \cdot x - b = -1$ under the condition that all samples are classified correctly. This results in the optimization problem of equation 5-10:

$$\begin{aligned} &\min ||w||^2, \quad \text{subject to:} \\ &l_{train}\{n\}(w \cdot \mathcal{X}_{train}\{n\} - b) \leq 1, \quad \forall n \in N \end{aligned} \tag{5-10}$$

With l_{train} the label of a sample $\mathcal{X}_{train}$. However, in some cases the widest possible margin does not result in the best classification, or it may not be possible at all to separate classes completely if some train samples are in clusters of the opposite class.

In these cases, the constraint in the optimization problem (equation 5-10) is relaxed and implemented as a hinge loss [23]. λ is a parameter which trades off importance of widest margin for adhering to the constraint. The minimization problem becomes (equation 5-11):

$$\text{minimize:} \left[\frac{1}{N}\sum_{n=1}^{N} \max(0, 1 - l_{train}\{n\}(w \cdot \mathcal{X}_{train}\{n\} - b))\right] + \lambda||w||^2 \tag{5-11}$$

Increasing λ moves emphasis to maximizing the margin between classes.

5-2-3 Kernel Trick

A separate challenge is that classes may be distributed such that a hyperplane is not capable of separating them irregardless of noise. Figure 5-6 depicts such a set.

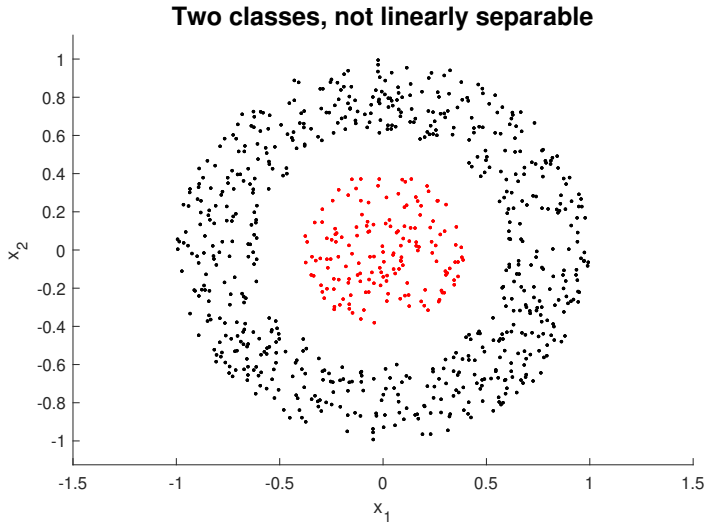


Figure 5-6: Not-linearly separable data set

To separate this data, it is needed to transform it first. This transformation is called the "kernel trick" [24]. Every sample in an n-dimensional data set is augmented with an extra dimension, which is given by a kernel function. A kernel function, in the context of the SVM, is a positive definite function, which depends on the in product of two input vectors.

$$K(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{5-12}$$

An often used kernel function is the Gaussian (see section 5-1) especially in anomaly detection. The transformed data is separable with a hyperplane:

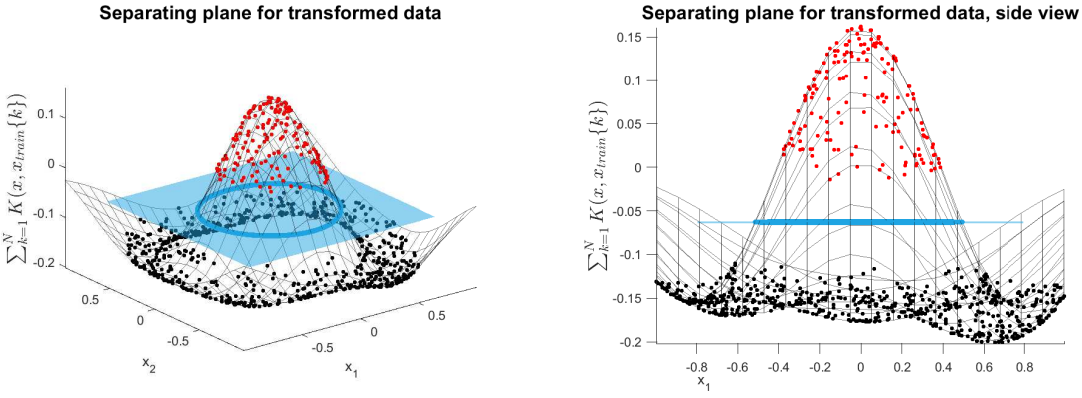


Figure 5-7: Data transformed with the kernel trick

This decision boundary in the original coordinates is depicted in figure 5-8.

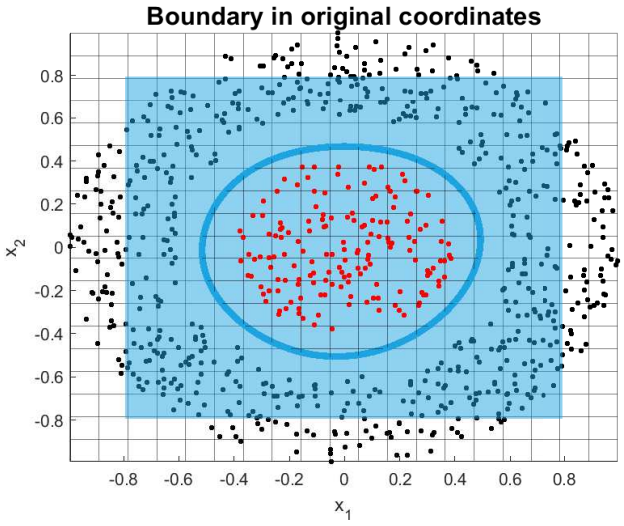


Figure 5-8: Separating hyperplane in the original coordinates

5-2-4 SVM Applied to Anomaly Detection

The SVM in combination with the kernel trick can be applied to anomaly detection. In this context the SVM is known as the **One class Support Vector Machine, OSVM**. The objective function of the OSVM is of course slightly different, as the aim of maximizing distance between a hyperplane and two classes is no longer valid. In [25] the distance between the origin and a hyper plane is maximized, under the constraint that samples need on the side of the hyperplane opposite the origin. In [26] a hypersphere is formed around the samples. The volume of this hypersphere is then minimized. In this thesis the matlab toolbox for SVMs is used, which cites [25] in the OSVM documentation.

The OSVM has two hyper parameters, σ and ν. σ is the kernel scale which is the radius of the kernel function, a Gaussian kernel in this thesis. The ν-parameter is an upper bound on the fraction of samples not included in the boundary and a lower bound on the amount of samples included in the support vector. As stated in the matlab documentation; "A small value of ν leads to fewer support vectors and, therefore, a smooth, crude decision boundary. A large value of ν leads to more support vectors and, therefore, a curvy, flexible decision boundary". Both σ and ν can be decreased to increase complexity.

The following figure (5-9) demonstrates the effect of both parameters on the classification of a set.

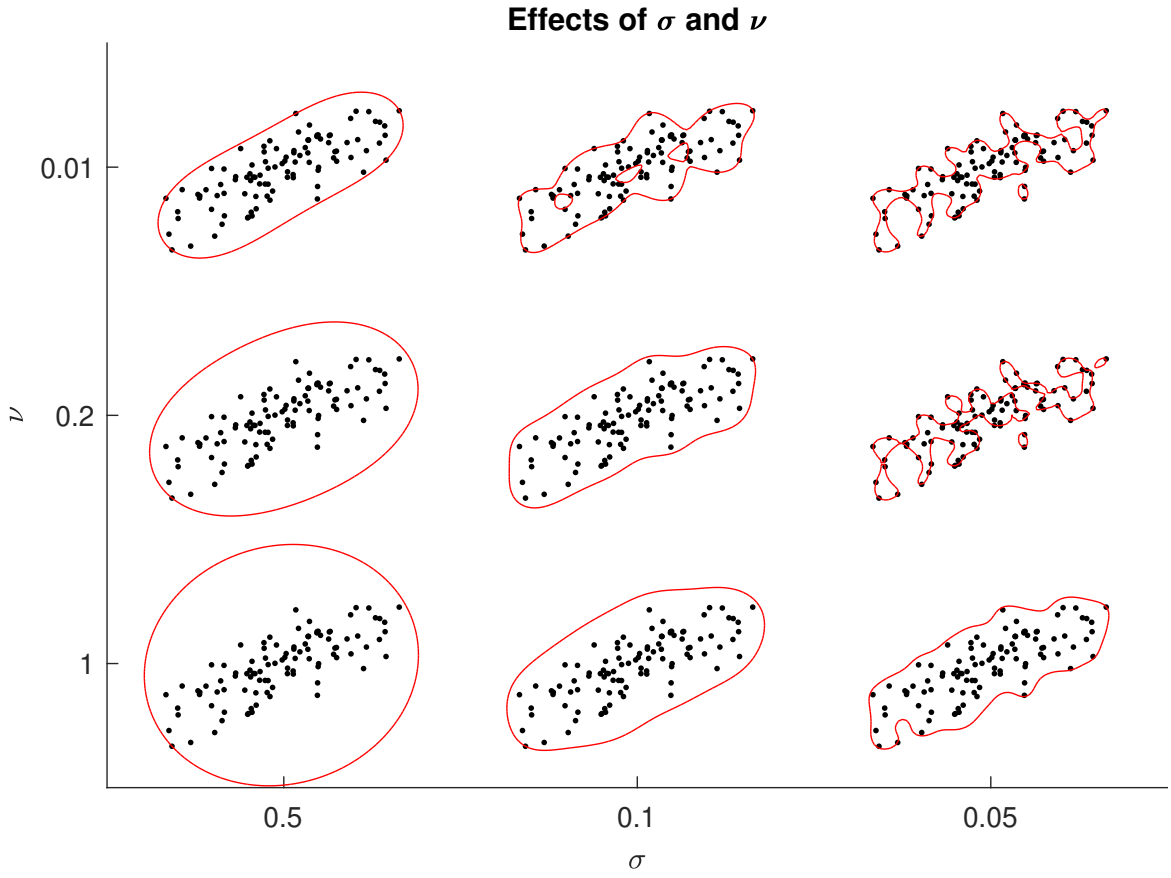


Figure 5-9: The effects of tuning parameters σ and ν on set bounds for identical data sets.

For an observation x the OSVM can give either of two outputs; a label or an anomaly score. The label is a binary score depending on what side of the hyperplane x is on. The anomaly score is the signed distance of x to the hyperplane. The latter is used in the thesis, in which case the bounding set for an OSVM is:

$$\mathcal{B} = \{x \in \mathbb{R}^n : f_{\text{OSVM}}(x) > \tau\} \tag{5-13}$$

With τ a user set threshold.

5-3 K-th Nearest Neighbor

The k-th nearest neighbor is, as the name implies, for an integer value k and observation x the sample in set $\mathcal{X}_{train}$ which is for $k = 1$ the closest neighbor, for $k = 2$ the second-to-closest neighbor, etc. The following notations are used:

$$\begin{aligned} &\mathrm{NN}(x, \mathcal{X}_{train}, k) \\ &d(x, \mathrm{NN}(x, \mathcal{X}_{train}, k)) \end{aligned} \tag{5-14}$$

With NN denoting the sample $\mathcal{X}_{train} \in \mathcal{X}_{train}$ which the k-th closest to x, and d the distance between x and its k-th nearest nearest neighbor. The k-th nearest neighbor distance is inversely proportional to the local density of samples $\rho(x_i, k)$.

$$\rho(x_i, k) = \frac{\text{No. Samples}}{\text{Volume}} = \frac{k}{c_d \cdot \Big(|xi - d(x_i, \mathrm{NN}(x_i, k))|\Big)^n}, \quad c_d \approx \frac{1}{\sqrt{n\pi}}\left(\frac{2\pi e}{n}\right)^{\frac{n}{2}} \tag{5-15}$$

With n the dimension of $x \in \mathbb{R}^n$. By increasing k the local estimate is made less sensitive to noise. For example, with $k = 1$ a density estimate next to an outlier is misleading because the distance to this single outlier is measured, but by increasing $k = 2$ this outlier is ignored. Yet increasing k also makes the density estimate less local.

5-3-1 K-th Nearest Neighbor Anomaly Detection

To avoid confusion between the concept of a k-th nearest neighbor and the machine learning algorithms based on it, only the latter will be referred to with an acronym; **KNN**. Dating back to 1951, KNN is one of the oldest machine learning methods [27], originally used for multiple class classification. In [28] KNN is adapted for anomaly detection using the following formula for an anomaly score:

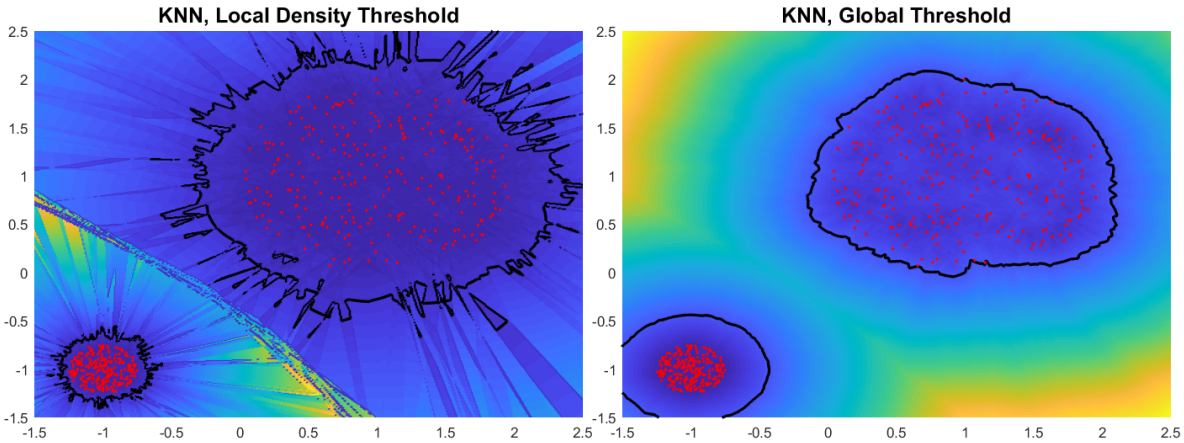
$$\phi_{\mathrm{KNND}}(x, \mathcal{X}_{train}, k) = \frac{d(x, \mathrm{NN}(x, \mathcal{X}_{train}, k))}{d(\mathrm{NN}(x, \mathcal{X}_{train}, k), \mathrm{NN}(\mathrm{NN}(x, \mathcal{X}_{train}, k), \mathcal{X}_{train}, k))} \tag{5-16}$$

As explained previously, k-nearest neighbor distance is inversely proportional to local density. In equation 5-16 the local density of x (in the numerator) is compared to the local density of its own k-th nearest neighbor (in the denominator). This is useful for data sets with multiple clusters with different densities. If the observation x is in the vicinity of a dense cluster it is required to be in close proximity to other samples to not be anomalous and vice versa for observations near sparse clusters. This is called a **Local Outlier Factor (LOF)**, a strategy in which observations are compared to their direct neighborhood to decide whether an observation is anomalous.

In a healthy residual set multiple clusters with varying density are not likely. The healthy residual set is a non linear mapping of a data set with continuous density. Rather, the goal of maintaining a low set volume calls for a global assessment of anomalies. KNN has been found to perform better without the LOF compensation in the denominator (equation 5-17):

$$f_{\mathrm{KNN}}(x, \mathcal{X}_{train}, k) = d(x, \mathrm{NN}(x, \mathcal{X}_{train}, k)) \tag{5-17}$$

Figure 5-10 visually compares the two KNN set bounding methods:



(a) Heatmap of LOF KNN, anomaly score **(b)** Heat of global threshold KNN, anomaly score

Figure 5-10: Heatmap of anomaly scores using LOF-KNN (left) and global threshold KNN (right)

A data set is created consisting of two clusters, bottom left with a high density, top right with a low density. Samples of the data set are depicted in red, black lines represent set bounds and the heat map depicts anomaly scores. The LOF-KNN method tightly bounds the high density cluster, and bounds the low density cluster more loosely. The global threshold KNN method tightly bounds the top right, but the low density in the top right cluster negatively affects the bounding of the bottom left cluster. In the example of figure 5-10 it appears that LOF-KNN achieves a better classification, but the data set has been constructed specifically to demonstrate the advantages of LOF-KNN.

The KNN based anomaly detection method is the one described in equation 5-17. In KNN an observation is anomalous if its local density is too low. The criterion for classifying an observation as anomalous is therefore:

$$\mathcal{B} = \{x \in \mathbb{R}^n : f_{\text{KNN}}(x) < \tau\} \tag{5-18}$$

With τ a user set threshold.

5-3-2 K-D Tree

KNN can be performed with or without training. Training consists in the construction of a K-D tree. A K-D tree is a space partitioning method which allows for searching through a data set in logarithmic time, see figure 5-11.

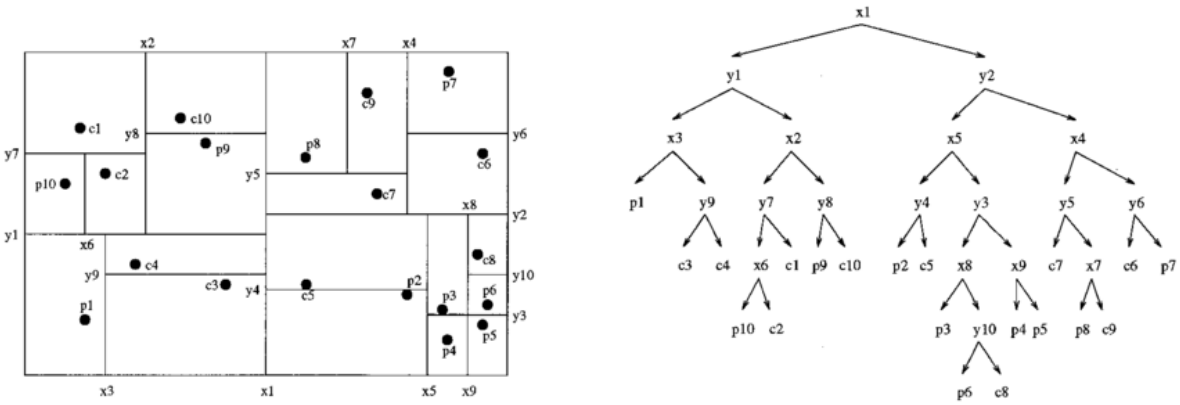


Figure 5-11: Space partitioning with a K-D tree, (Adapted from [29])

KNN search can also be performed without constructing a K-D tree first. Because this tree has to be constructed and then searched through, it is sometimes faster to not create the tree. This can be seen in figure 5-12. The efficiency of searching using K-D tree increases as the amount of test samples increases. For this thesis, the amount of train samples is such that creating a K-D tree is always the better option.

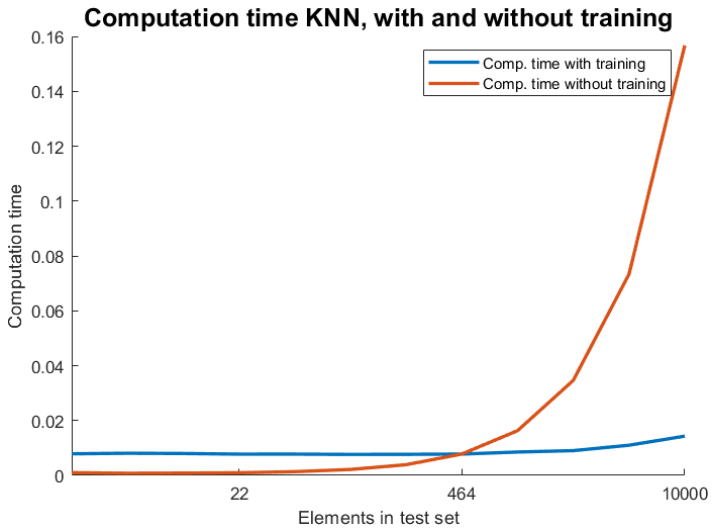


Figure 5-12: Effect of constructing a K-D tree on computation time. For an increasing amount of data (x-axis) an increasing computation time is required (y-axis), yet less so for the KNN with K-D tree (blue), than the untrained KNN (red)

5-4 Rejected Methods

In the early phases of the thesis, many methods of anomaly detection were considered. Several surveys of anomaly detection methods were used as a catalog of possible options. This section serves as a brief overview of the methods which were rejected from further consideration in this thesis. Descriptions of these methods are deliberately brief, focusing mainly on the motivations for rejection.

5-4-1 Autoencoder

Autoencoders are a specialized form of neural networks. Autoencoders consist of an encoder and a decoder. The encoder reduces the dimensionality of its input as much as possible, creating an encoded version of the input. The decoder restores the encoded version to its original form. In figure 5-13 the structure of an auto encoder is depicted:

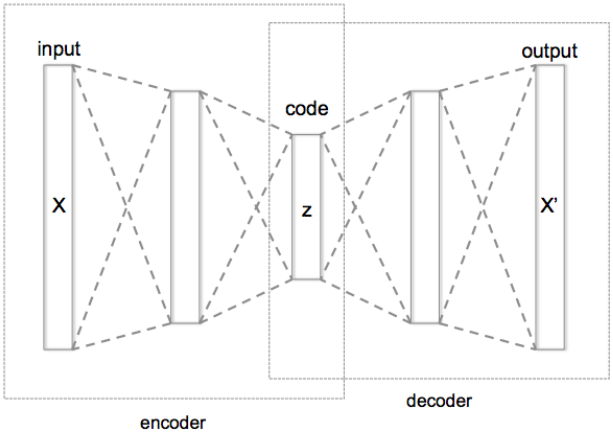


Figure 5-13: Structure of an autoencoder. Consecutive layers are structured from left to right, vertical height represents proportionally the amount of neurons per layer (Public Domain Image)

Consecutive layers are depicted left to right. Vertical height represents proportionally the amount of neurons in a layer. A simplified explanation of the effect of encoding data in a compact form and restoring it, is that only "room" for the most essential features of data is left in the encoded form. Any noise in the input data is not counted among these most essential of features, which is why autoencoders have found application in image reconstruction and noise rejection. Outliers can be viewed as a form of noise, which is why auto encoders have been suggested for the application of outlier detection. Figure 5-14 depicts denoising of handwriting images:

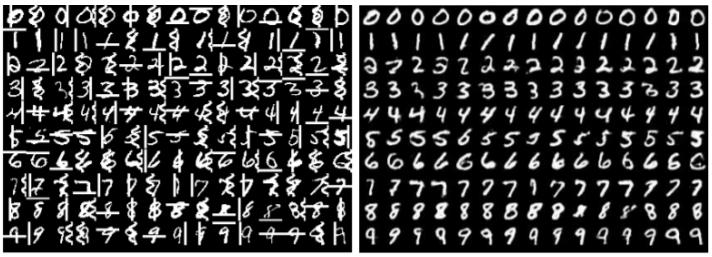


Figure 5-14: Image denoising using an auto encoder. Corrupted images on the left, restored images on the right. (Adapted from [30])

Several drawbacks have made autoencoders, in the view of the author, not worth pursuing. Neural networks are trained by an extensive optimization process on a large train data set. In fault detection, a single residual set is analyzed once in as short a time frame as possible. For neural networks a large amount hyper parameters and design choices influence performance; activation functions, amount of layers, amount of neurons per layer, etc. With such a large amount of options it is hard to make a conclusive statement about the performance of an auto encoder compared to other methods. Finally, to achieve good results it is necessary to discretize the data, which decreases precision.

5-4-2 Decision Trees

Decision Trees are a concept well known outside the field of machine learning. A decision tree simplifies the taking of a complicated decision by posing the user multiple simpler questions, ordered in a flowchart. The answer to each question leads the user to another question until at a terminal point the answer to the decision is provided. An example is included in figure 5-15.

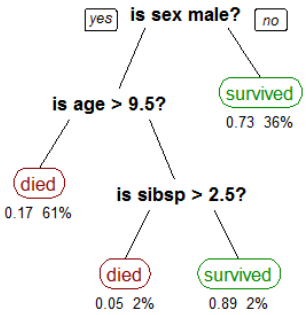


Figure 5-15: Decision tree for survival rate of titanic passengers. (Public Domain Image)

In machine learning each instance of a question is called a node, the lines connecting nodes are called branches. Decision trees are widely used in machine learning, often in combination with gradient boosting or bootstrap aggregating. A strength of decision trees is their ability to handle categorical data. As an example, the color of a car or the country of birth may be used as an input for a decision tree. In classifying faults however this is irrelevant, as in this thesis all data is continuous.

5-4-3 Distance Based Clustering

In distance based clustering, anomaly scores are based on the distance of an observation to a cluster center. Different vector norms can be used (L^1, L^2, etc.), possibly in combination with a covariance matrix. The major drawback is that this technique can, in its most advanced form, created only simple, convex and symmetrical shapes. In view of this, distance based clustering is barely an improvement over the methods mentioned in chapter 2, which were described as not complex enough.

Part III

Part III offers improvements on fault detection as performed in [1]. A new sampling strategy is presented. Various machine learning algorithms are optimized for set based fault detection. Using several innovations an improvement is reached over [1] in performance metrics introduced in this part.

Chapter 6

Improved Sampling Technique

In this section an alternative sampling technique to the one used in [1] is introduced. It is the only chapter which discusses an innovation not directly related to machine learning. Sections 6-2, 6-3, explain shortcomings of the current method, and present improvements.

6-1 Drawbacks of Current Sampling Method

In the introduction to fault detection (chapter 2) it was explained how healthy residual sets can be created using Monte Carlo sampling. Some notations are briefly restated. A healthy residual set is produced by simulating the uncertain processes underlying the creation of a residual:

$$\begin{aligned} \mathcal{R}_H &= \langle g(\hat{x}, \hat{p}, u, \mathcal{N}_p, \mathcal{N}_{y0}, \mathcal{N}_{y1}) \rangle \\ \mathcal{N}_p &= \langle \tilde{n}_p\{1\}, \tilde{n}_p\{2\}, ..., \tilde{n}_p\{s\} \rangle \\ \mathcal{N}_{y0} &= \langle \tilde{n}_{y0}\{1\}, \tilde{n}_{y0}\{2\}, ..., \tilde{n}_{y0}\{s\} \rangle \\ \mathcal{N}_{y1} &= \langle \tilde{n}_{y1}\{1\}, \tilde{n}_{y1}\{2\}, ..., \tilde{n}_{y1}\{s\} \rangle \end{aligned} \tag{6-1}$$

With $\mathcal{N}_p$, $\mathcal{N}_{y0}$ and $\mathcal{N}_{y1}$ sets of hypothetical uncertainties. Uncertainties are sampled from their associated domain:

$$\tilde{n}_p \in \mathcal{G}_p, \quad \tilde{n}_{y0} \in \mathcal{G}_{y0}, \quad \tilde{n}_{y1} \in \mathcal{G}_{y1} \tag{6-2}$$

Some properties of the sampling as it is performed in [1] should be noted. The uncertainties are not necessarily sampled in a finite domain:

$$\mathcal{G}_e \subseteq \mathbb{R}, \quad \mathcal{G}_p \subseteq \mathbb{R}, \quad \mathcal{G}_{y0} \subseteq \mathbb{R}, \quad \mathcal{G}_{y1} \subseteq \mathbb{R} \tag{6-3}$$

Furthermore, it is assumed that for each noise n a probability density function $p_n(x)$ is known. A noise set $\mathcal{N}$ is formed by sampling from this distribution, typically a Gaussian distribution. It is argued that there are several flaws in this approach, which are presented before an alternative method is proposed.

Infinite Support

For probability density functions with infinite support like the Gaussian distribution the probability of sampling an outlier increases proportionally to the sample count. At the same time modeling noise with infinite distributions is not an accurate representation of reality. Sensors have a limited output range, an output noise beyond a certain range can not be recorded. Additionally, sensor readings outside a certain range can be contradictory to a model. As an example, in the three tank system (see chapter 8) the water level in a tank can not be below the bottom of a tank, so it is not necessary to simulate this scenario when creating the healthy residual set.

Center Heavy Distribution

An inefficient use of data arises from the center heavy distribution used to model noises. It is to be expected that when heavy distributions are used to create the noise sets $\mathcal{N}$, the resulting healthy residual set $\mathcal{R}_H$ is center heavy as a result. The AD methods considered in this thesis all rely in some way on local sample density to output an outlier score. Outlier score is lower for a higher sample density. The bounding set is a region where all outlier scores are below a threshold. For a center heavy distribution of data, the center outputs an outlier score below the threshold with a large margin, due to sample density being higher. By removing some samples from the center, the center would still be an inlying region, but with a smaller margin. The margin by which an inlier is classified inlying is not important, provided the classification is correct in the first place. Rather than removing samples, by uniform sampling this superfluous data is not created in the first place, and is instead sampled at the edges of the residual set where more detail actually improves performance.

Lack of Guarantee on FAR Robustness

A relation exists between FAR, MDR and sample count. In the following section (section 6-2) this relation is defined in a more rigorous manner, yet the basic relation is that for a higher sample count FAR decreases, but MDR increases. The healthy residual set fills an ever growing region for increasing sample count. This increases the probability of overlapping with the received residual, but also the probability of overlapping with a faulty residual. An important concept in [1] is increasing detection for a probabilistically robust FAR performance, yet the influence of sampling in this regard is not addressed.

6-2 FAR Robust Sampling

Having noted some of the drawbacks in the current sampling method, an alternative method is proposed. To avoid confusion, variables and constants relevant to the system are subscripted with sys, Monte Carlo variables and constants are subscripted with mc.

Consider subject to some noise $n_{sys}[t]$, with a probability density function $p_{sys}(x)$ in domain $[-x_{sys}, +x_{sys}]$. Hypothetical noises are created with a Monte Carlo simulation. In the current sampling technique, hypothetical noises $\tilde{n}_{mc}$ are created with a Monte Carlo simulation where

the probability density function of the Monte Carlo noise $p_{mc}(x)$ is the same as the one in the system, $p_{mc}(x) = p_{sys}(x)$, also being sampled in the same domain $[-x_{mc}, +x_{mc}] = [-x_{sys}, +x_{sys}]$. There is however no requirement to sample with the same distribution or in the same domain. It is proposed that a uniform distribution function is used for $p_{sys}(x)$. The choice of sampling domain is less obvious. For $x_{mc} > x_{sys} > 0$ FAR decreases because the probability of overlapping with the received residual is increased. For $0 < x_{mc} < x_{sys}$ MDR decreases because of a decreased overlap with faulty residuals. A method is proposed where the domain of the Monte Carlo noises is the result of an optimization problem which minimizes MDR for a probabilistic guarantee on FAR and a given sample count.

6-2-1 Assumptions and Notations

A system is subject to $J \in \mathbb{Z}$ uncorrelated one dimensional noises, represented by vector $n_{sys} \in \mathbb{R}^J$:

$$n_{sys} = \langle n_{sys}\{1\}, n_{sys}\{2\}, ..., n_{sys}\{J\} \rangle \tag{6-4}$$

The probability density function of the j-th element in the vector is denoted $p_{sys}\{j\}$. It is assumed that each system noise $p_{sys}\{j\}$ has a finite support, denoted with $[-B_{sys}\{j\}, B_{sys}\{j\}]$. For ease of notation, all system noises have symmetrical support and zero mean, and symmetrical probability density functions.

Preposition 1. *The probability density functions in p_{sys} have distributions symmetrical over the mean, $p_{sys}(x) = p_{sys}(-x)$.*

The set of all Monte Carlo noise vectors is denoted with $\mathcal{N}_{mc}$:

$$\begin{aligned} \mathcal{N}_{mc} &= \langle \mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_J \rangle \\ \mathcal{N}_j &= \langle n_j\{1\}, n_j\{2\}, ..., n_j\{s\} \rangle \end{aligned} \tag{6-5}$$

The Monte Carlo noise set $\mathcal{N}_{mc}$ contains J amount of sets $\mathcal{N}_j$, with each set $\mathcal{N}_j$ having s elements, s being the sample count set by the user. The following assumptions are posed to define when a systems residual is bounded by the healthy residual set:

Assumption 1. *For healthy system dynamics, if n_{sys} is bounded by $\mathcal{N}_{mc}$, then $\mathcal{R}_H$ also bounds r.*

It should be noted that if $\mathcal{R}_H$ does not bound r, this does not in itself mean a false alarm will be given. For an inadequate healthy residual set $\mathcal{R}_H$ a healthy residual r might still be bounded if the method of set bounding is overly conservative.

Assumption 2. *The system noise n_{sys} is bounded by the set of Monte Carlo noises $\mathcal{N}_{mc}$ if the hyper box that bounds all Monte Carlo noises $\mathcal{N}_{mc}$ also bounds n_{sys}.*

For all Monte Carlo noises the uniform probability density function $p_{mc}\{j\}$ is defined as:

$$p_{mc}\{j\}(x) : \begin{cases} \frac{1}{2B_{mc}\{j\}} & \forall x \quad \in [-B_{mc}\{j\}, B_{mc}\{j\}] \\ 0 & \forall x \quad \notin [-B_{mc}\{j\}, B_{mc}\{j\}] \end{cases} \tag{6-6}$$

The support of each $p_{mc}\{j\}$ is the result of an optimization problem which will be explained in a later section 6-3.

6-2-2 Probability of Bounding System Noise

To calculate the probability of n_{sys} not being bounded, a mathematical definition for the assumption 2 is given.

Lemma 1. *n_{sys} is not bounded by $\mathcal{N}_{mc}$ if at least one element j of $n_{sys}\{j\}$ is not bounded by $\mathcal{N}_{mc}$. An element $n_{sys}\{j\}$ is not bounded if either of the following conditions are met:*

$$\max(\mathcal{N}_j) < n_{sys}\{j\}$$
$$\min(\mathcal{N}_j) > n_{sys}\{j\}$$

With $\mathcal{N}_j$ the set of noises created to approximate element $n_{sys}\{j\}$.

Calculating violation probabilities is straightforward when using the notation of lemma 1.

Definition 1. *The probability of not bounding $\gamma(x)$ by a specific Monte Carlo noise $\mathcal{N}_j$ is defined as:*

$$\gamma(x) = \min\left(\left[\frac{B_{mc}\{j\}+x}{2B_{mc}\{j\}}\right]^s + \left[\frac{B_{mc}\{j\}-x}{2B_{mc}\{j\}}\right]^s, 1\right)$$

Some additional clarification for definition 1 is provided. The probability of sampling a single value lower than x is easily calculated due to the uniform probability density function of the Monte Carlo noise:

$$p_{low} = \min\left(\left[\frac{B_{mc}\{j\}+x}{2B_{mc}\{j\}}\right], 1\right) \tag{6-7}$$

If *all* sampled values $n_j \in \mathcal{N}_j$ are lesser than x, than one of the conditions in lemma 1 is not met, namely $\max(\mathcal{N}_j) < x$. The probability of this occurrence is the consecutive intersection of p_{low} for every sample $n_j \in \mathcal{N}_j$.

$$P_{low} = p_{low,1} \cap p_{low,2} \cap ... \cap p_{low,s} = \min\left(\left[\frac{B_{mc}\{j\}+x}{2B_{mc}\{j\}}\right]^s, 1\right) \tag{6-8}$$

The same logic applies to the probability of sampling only greater than x:

$$\begin{aligned} p_{high} &= \left(\left[\frac{B_{mc}\{j\}-x}{2B_{mc}\{j\}}\right], 1\right) \\ P_{high} &= \left(\left[\frac{B_{mc}\{j\}-x}{2B_{mc}\{j\}}\right]^s, 1\right) \end{aligned} \tag{6-9}$$

The probability of not bounding is the union of P_{high} and P_{low}. The min operation in definition 1 is applied to maintain a probability $\gamma(x) = 1$ when $|x| > B_{mc}\{j\}$.

Definition 2. *The probability of $n_{sys}\{j\}$ being sampled and not being bounded is defined as ϕ_{nb}:*

$$\phi_{nb}\{j\} = \int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} \gamma(x) p_{sys}\{j\}(x) dx$$

For a generic time step, the probability of not bounding $\phi_{nb}\{j\}$ an element $n_{sys}\{j\}$ is the intersection of the probability of sampling $n_{sys}\{j\}$ with the probability of not bounding $n_{sys}\{j\}$ integrated over the domain of $p_{sys}\{j\}$.

Definition 3. *The probability of not bounding n_{sys} for a generic time step is defined as Φ_{nb}:*

$$\Phi_{nb} = 1 - \prod_{j=1}^{J}(1 - \phi_{nb}\{j\})$$

As was stated in lemma 1, not bounding occurs if one or more elements of $n_{sys}\{j\}$ is not bounded. Φ_{nb} is the complement of the intersections of the complement of $\phi_{nb}\{j\}$ for all elements $1 \leq j \leq J$. In summary, Φ_{nb} is the probability of $\mathcal{N}_{mc}$ not bounding n_{sys} for any time step, under assumption 2. Under assumption 1, this is also the probability of $\mathcal{R}_H$ not bounding r. User controlled variables with an influence on Φ_{nb} are sample count s and Monte Carlo supports B_{mc}.

Having explained how to calculate the probability of not bounding Φ_{nb}, another aspect of set bounding is expanded on. As was explained in chapter 2, for set bounds on $\mathcal{R}_H$ with a lower set volume, a lower MDR is to be expected. In chapter 2 this is brought up as a motivation for more complex set bounds. However, by sampling more strategically it is possible to create a set $\mathcal{R}_H$ for which a smaller bounding set is theoretically possible.

Assumption 3. *The set volume of $\mathcal{N}_{mc}$ is approximated with the volume of the bounding hyper box.*

Assumption 3 is motivated by the specific circumstance of $\mathcal{N}_{mc}$ being constructed with uncorrelated noises. In a multivariate approach, assumption 3 would not hold.

Assumption 4. *Decreasing the set volume of $\mathcal{N}_{mc}$ decreases the set volume of $\mathcal{R}_H$.*

From assumption 3 it follows:

Lemma 2. *The set volume of $\mathcal{R}_H$ is proportional to the product of all Monte Carlo noise supports B_{mc}.*

To illustrate lemma 2, for any element of j the expected maximum and minimum values in the set $\mathcal{N}_{mc}$ are:

$$\begin{aligned} E[\min(\mathcal{N}_{mc,j})] &= B_{mc}\{j\}\left(\frac{2}{s+1} - 1\right) \\ E[\max(\mathcal{N}_{mc,j})] &= B_{mc}\{j\}\left(1 - \frac{2}{s+1}\right) \end{aligned} \tag{6-10}$$

The expected volume of the hyper box v_{hb} is:

$$\begin{aligned} v_{hb} &= \prod_{j=1}^{J}\Big(E[\max(\mathcal{N}_j)] - E[\min(\mathcal{N}_j)]\Big) \\ v_{hb} &= \prod_{j=1}^{J}\left(B_{mc}\{j\}\left[2 - \frac{4}{s+1}\right]\right) \\ v_{hb} &= \left(2 - \frac{4}{s+1}\right)^{J}\prod_{j=1}^{J} B_{mc}\{j\} \end{aligned} \tag{6-11}$$

From assumptions 4 and lemma 2 it follows that minimizing the product of all Monte Carlo noise supports $\prod_{j=1}^{J} B_{mc}\{j\}$ decreases set volume of $\mathcal{R}_H$.

6-2-3 Performance Objectives

Φ_{nb} and v_{hb} are dependent on both B_{mc} and s. This leads to conflicting objectives, as both Φ_{nb} and v_{hb} ought to be as low as possible. s is a fixed user defined parameter. The user sets a constraint α, the maximum allowable value for Φ_{nb}. Under this constraint v_{hb} is minimized over B_{mc}. The minimization problem is posed as follows (equation 6-12):

$$\begin{aligned} &\underset{B_{mc}\{1\},\ldots,B_{mc}\{J\}}{\text{minimize}} && v_{hb} \\ &\text{subject to:} && \Phi_{nb}(B_{mc}) < \alpha \end{aligned} \tag{6-12}$$

Several constraints can be added to the minimization problem, which is done in section 6-3. Using the constraints an initial search region can also be defined. However, the mathematics motivating these constraints should first be explained, which is done in section 6-2-4.

6-2-4 Upper and Lower Bounds on Monte Carlo Support

It was previously explained how $\phi_{nb}\{j\}$ is a function of $B_{mc}\{j\}$. An exact solution to $\phi_{nb}\{j\}(B_{mc}\{j\}) = \xi$ can only be approximated numerically. However, because the aim is to formulate constraints on a search region an upper and lower bound on $B_{mc}\{j\}$ is sufficient.

Preposition 2. *For an equality $\phi_{nb}\{j\}(B_{mc}\{j\}) = \xi$ a lower bound ${}^{*}B_{mc}\{j\}$ and an upper bound $B^{*}_{mc}\{j\}$ fulfill the property:*

$$\begin{aligned} &\phi_{nb}\{j\}(B_{mc}\{j\}) = \xi \\ &{}^{*}B_{mc}\{j\} < B_{mc}\{j\} < B^{*}_{mc}\{j\} \end{aligned}$$

Upper Bound

In this section it is explained how to give an upper bound on the required Monte Carlo supports for probability of not bounding $n_{sys}\{j\}$ for a generic time step. This is done by formulating a worst case scenario probability density function and circumventing a complicated equality by solving an inequality.

Lemma 3. *The probability of not bounding x is given by $\gamma_{nb,j}(x)$ (definition 1) The probability of not bounding is proportional to absolute value of x:*

$$|x^{+}| > |x^{-}|, \qquad \gamma_{nb}\{j\}(|x^{+}|) \geq \gamma_{nb}\{j\}(|x^{-}|)$$

To proof lemma 3, the derivative of $\gamma_{nb}\{j\}(x)$ with respect to x is evaluated. The $\min(x, 1)$ is disregarded, as this is a step in establishing a global minimum where the discontinuity has no effect.

$$\frac{\partial \gamma_{nb}\{j\}(x)}{\partial x} = \frac{s}{2B_{mc}\{j\}}\left(\left[\frac{B_{mc}\{j\} + x}{2B_{mc}\{j\}}\right]^{s-1} - \left[\frac{B_{mc}\{j\} - x}{2B_{mc}\{j\}}\right]^{s-1}\right) \tag{6-13}$$

A unique extreme value exists at $x = 0$. This is a global minimum, which can be proven using the second derivative:

$$\frac{\partial^2 \gamma_{nb}\{j\}(x)}{\partial^2 x} = \frac{s(s-1)}{4B_{mc}^2\{j\}}\left(\left[\frac{B_{mc}\{j\}+x}{2B_{mc}\{j\}}\right]^{s-2} + \left[\frac{B_{mc}\{j\}-x}{2B_{mc}\{j\}}\right]^{s-2}\right) \tag{6-14}$$

At $x = 0$, the second derivative has a positive value:

$$\frac{\partial^2 \gamma_{nb}\{j\}(0)}{\partial^2 x} = \frac{s(s-1)}{2B_{mc}^2\{j\}}\left[\frac{B_{mc}\{j\}}{2B_{mc}\{j\}}\right]^{s-2} \tag{6-15}$$

Which confirms the extreme value at $x = 0$ as a minimum. The function $\gamma_{nb}\{j\}$ is a symmetrical function ($\gamma_{nb}\{j\}(x) = \gamma_{nb}\{j\}(-x)$) with a global minimum at $x = 0$, which proves lemma 3. Following lemma 3, the maximums of the $\gamma_{nb}\{j\}$ can be found at the extremes of the domain, at $x = \pm B_{sys}\{j\}$.

Now, rather than the probability of not bounding for a specific value n_{sys}, let us revisit the probability $\phi_{nb}\{j\}$ of not bounding n_{sys} for a generic time step, which is restated below:

$$\phi_{nb}\{j\} = \int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} p_{sys}\{j\}(x)\gamma_{nb}\{j\}(x)dx$$

It is known for $\gamma_{nb}\{j\}(x)$ which value of x gives the highest probability of not bounding (lemma 3). A worst case distribution function can be formulated.

Definition 4. *The worst case distribution $f_{wc}\{j\}(x)$ is defined as:*

$$f_{wc}\{j\}(x) = \tfrac{1}{2}(\delta(x - B_{sys}\{j\}) + \delta(x + B_{sys}\{j\}))$$

With $\delta(x)$ the Dirac delta function.

Lemma 4. *If $\mathcal{N}_{mc}$ bounds $\int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} f_{wc}\{j\}(x)\gamma_{nb}\{j\}(x)dx$, then it bounds all other possible distributions with the same support.*

The distribution $f_{wc}\{j\}$ has full probability $p = 1$ of sampling values with the lowest possible probability of being bounded. This means that a Monte Carlo which guarantees a probability of not bounding $\phi_{nb}\{j\} = \xi$ for $f_{wc}\{j\}(x)$, will guarantee a lower probability of not bounding $\phi_{nb}\{j\} < \xi$ for any other distribution with the same support.

In equation 6-16 the integration for $\phi_{nb}\{j\}$ with the worst case probability distribution is performed:

$$\begin{aligned} &\int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} f_{wc}\{j\}(x)\gamma_{nb}\{j\}(x)dx = \xi_{wc}, \\ &\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s + \left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s = \xi_{wc} \end{aligned} \tag{6-16}$$

Unfortunately, stating the general solution to the equality in equation 6-16 for $B_{mc}\{j\}$ is not feasible, this equation being the sum of two very high order binomials. However, a more conservative estimate for $B_{mc}\{j\}$ can be given.

$$\begin{aligned}
&\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s+\left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s=\xi_{wc},\\
&\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s>\left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s,\\
&2\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s>\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s+\left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s,\\
&2\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s>\xi_{wc}
\end{aligned} \tag{6-17}$$

Solving the last inequality in formula 6-17, as an equality yields the following expression:

$$\begin{aligned}
&2\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s=\xi_{wc}\\
&B_{mc}\{j\}=\frac{-B_{sys}\{j\}}{1-2\left(\frac{\xi_{wc}}{2}\right)^{\frac{1}{s}}}
\end{aligned} \tag{6-18}$$

Definition 5. *For a probability of not bounding ξ, an upper bound of the required Monte Carlo support is given by:*

$$B^*_{mc}\{j\}(\xi)=\frac{-B_{sys}\{j\}}{1-2\left(\frac{\xi}{2}\right)^{\frac{1}{s}}}$$

From the inequalities in formula 6-17 it follows that:

$$\begin{aligned}
&\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s+\left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s=\xi\\
&\left[\frac{B^*_{mc}\{j\}(\xi)+B_{sys}\{j\}}{2B^*_{mc}\{j\}(\xi)}\right]^s+\left[\frac{B^*_{mc}\{j\}(\xi)-B_{sys}\{j\}}{2B^*_{mc}\{j\}(\xi)}\right]^s<\left[\frac{B_{mc}\{j\}+B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s+\left[\frac{B_{mc}\{j\}-B_{sys}\{j\}}{2B_{mc}\{j\}}\right]^s
\end{aligned} \tag{6-19}$$

And from equation 6-19 and lemma 4, it follows:

$$\begin{aligned}
&\phi_{nb}\{j\}(B_{mc})=\xi\\
&\phi_{nb}\{j\}(B^*_{mc})<\phi_{nb}\{j\}(B_{mc})
\end{aligned} \tag{6-20}$$

So far, it has been demonstrated how to obtain a conservative estimate of the Monte Carlo supports for which the probability of not bounding is smaller than required. It is proven that this is in fact an upper bound for $B_{mc}\{j\}$.

Lemma 5. *The quantity $\phi_{nb}\{j\}$ is inversely proportional to $B_{mc}\{j\}$.*

To proof lemma 5, apply a scaling factor β to $B_{mc}\{j\}$ in $\gamma_{nb}\{j\}$, such that increasing β is equivalent to increasing $B_{mc}\{j\}$:

$$\gamma_{nb}\{j\}=\left[\frac{\beta B_{mc}\{j\}+x}{\beta 2B_{mc}\{j\}}\right]^s+\left[\frac{\beta B_{mc}\{j\}-x}{\beta 2B_{mc}\{j\}}\right]^s \tag{6-21}$$

Equation 6-21 is equivalent to the following notation:

$$\gamma_{nb}\{j\} = \left(\frac{B_{mc}\{j\} + \frac{x}{\beta}}{2B_{mc}\{j\}}\right)^s + \left(\frac{B_{mc}\{j\} - \frac{x}{\beta}}{2B_{mc}\{j\}}\right)^s \tag{6-22}$$

Note the following limit:

$$\lim_{\beta \to \infty} \frac{x}{\beta} = 0 \tag{6-23}$$

From equations 6-23, 6-22, it is obvious that increasing the Monte Carlo supports is equivalent to decreasing x. In 6-14 and 6-15 it was proven that decreasing x steadily lowers the probability of not bounding. This proves lemma 5. Using lemma 5 and equation 6-20, it follows that the conservative estimate $B^*_{mc}\{j\}$ is an upper bound on the Monte Carlo supports required for a given probability of not bounding $\phi_{nb}\{j\}$.

Lower Bound

In this section it is explained how to obtain a lower bound on Monte Carlo supports, ${}^*B_{mc}\{j\}(\phi_{nb}\{j\})$. Using the same logic for formulating an upper bound in reverse to obtain a lower bound is unfortunately not feasible. The upper bound was found with a function f_{wc} which is the hardest possible function to bound. However, for the lower bound, the easiest possible function would be $\delta(x)$, a Dirac delta at zero, a function which is bounded for any Monte Carlo support $B_{mc}\{j\} > 0$. A different, more ad hoc approach for obtaining a lower bound ${}^*B_{mc}\{j\}(\phi_{nb}\{j\})$ is formulated. For an arbitrary distribution p_{sys}, a function f_{lb} can be formulated which if bounded provides a lower bound:

Definition 6. *The function f_{lb} is a function which satisfies the following requirement:*

$$\int_{-(Z-\epsilon)}^{+(Z-\epsilon)} f_{lb}(x)dx \geq \int_{-Z}^{+Z} p_{sys}(x)dx, \quad \forall Z \in [-B_{sys}\{j\}, +B_{sys}\{j\}], \quad \epsilon > 0$$

Lemma 6. *If $\mathcal{N}_{mc}$ bounds:*

$$\int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} p_{sys}\{j\}(x)\gamma_{nb}\{j\}(x)dx$$

then $\mathcal{N}_{mc}$ bounds:

$$\int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} f_{lb}\{j\}(x)\gamma_{nb}\{j\}(x)dx.$$

It was established in the previous section on the subject of upper bounding that probability of not bounding $\gamma_{nb}\{j\}(x)$ becomes lower proportionally for lower values of $|x|$ (lemma 3). The function f_{lb} has a higher or equal proportion of its distribution in intervals with lower values of $|x|$ compared to p_{sys}.

A function f_{lb} is constructed which is easily integrated and adheres to lemma 6. First, a user parameter $M \in \mathbb{Z}$ is introduced. A vector $\mathcal{X}_G$ is constructed, which consists of M values uniformly spaced in the interval $[0, B_{sys}\{j\}]$.

$$\mathcal{X}_G = \langle x_G\{1\}, x_G\{2\}, ..., x_G\{M\}, \rangle \tag{6-24}$$

The function f_{lb} consists in a sum of $M-1$ Dirac deltas (equation 6-25), multiplied with integrations of p_{sys} in a short interval:

$$\begin{aligned} f_{lb}(x) &= \sum_{m=1}^{M-1} b_m \delta_m(x) \\ \delta_m(x) &= \delta(x - x_G\{m\}) \\ b_m &= 2 \int_{x_G\{m\}}^{x_G\{m+1\}} p_{sys}\{j\}(z)dz \end{aligned} \tag{6-25}$$

${}^*\phi_{nb}\{j\}$ is the product of $\gamma_{nb}\{j\}(x)$ and $f_{lb}\{j\}$ integrated over $[-B_{sys}\{j\}, +B_{sys}\{j\}]$. Because $f_{lb}\{j\}$ is constructed using Dirac deltas, the integration is performed easily (equation 6-26):

$$\begin{aligned} {}^*\phi_{nb}\{j\} &= \int_{-B_{sys}\{j\}}^{+B_{sys}\{j\}} f_{lb}\{j\}(x)\gamma_{nb}\{j\}(x)dx = \\ &\sum_{m=1}^{M-1} \min\left(\left[\frac{B_{mc}\{j\} + x_G\{m\}}{2B_{mc}\{j\}} \right]^s + \left[\frac{B_{mc}\{j\} - x_G\{m\}}{2B_{mc}\{j\}} \right]^s, 1 \right) b_m \end{aligned} \tag{6-26}$$

For a given probability of not bounding a generic time step ξ, a lower bound on the required Monte Carlo supports ${}^*B_{mc}\{j\}$ is acquired by solving the optimization problem in equation 6-27 for $B_{mc}\{j\}$:

$$\begin{aligned} {}^*B_{mc}\{j\}(\xi) &= \arg\min_{B_{mc}\{j\}} {}^*\phi_{nb}\{j\} \\ &\text{subject to: } {}^*\phi_{nb}\{j\} \geq \xi \end{aligned} \tag{6-27}$$

It should be noted that the equality ${}^*\phi_{nb}\{j\} = \xi$ can not be solved directly, and neither can it be replaced by an inequality as was the case for the upper bound. Rather, an iterative method is used to approximate $\phi_{nb}\{j\} = \xi$. To guarantee ${}^*B_{mc}\{j\}$ being a lower bound, an iteration point at which $\phi_{nb}\{j\} \geq \xi$ is chosen as a value for ${}^*B_{mc}\{j\}$. This is the motivation for the notation in equation 6-27.

To approach a solution the property described in lemma 3 is exploited. A bi-sectional search method is used. The value for ${}^*B_{mc}\{j\}$ can be found in the interval $[0, B_{mc}\{j\}^*]$. Iterations of the search method are performed as follows (algorithm 1):

$y := \frac{1}{2} B_{mc}\{j\}^*$
$\Delta y := y$
while ${}^*\phi_{nb}\{j\}(y) < \xi$ *or* ${}^*\phi_{nb}\{j\}(y) > \xi + tolerance$ **do**
  $\Delta y := \frac{1}{2}\Delta y$
  if ${}^*\phi_{nb}\{j\}(y) > \xi$ **then**
    $y := y + \Delta y$
  else
    $y := y - \Delta y$
  end
end
${}^*B_{mc}\{j\} := y$

Algorithm 1: Iteration steps for finding the highest lower bound on $B_{mc}\{j\}$

Further Comments About Bounds

The lower bound is the highest possible lower bound, whereas the upper bound is not the lowest possible upper bound. These bounds are used to define a search space (section 6-2-5 and 6-3), so decreasing the upper bound would decrease the search space and accelerate minimization. However, calculating a lower upper bound would involve an iterative process. In the minimization algorithm (section 6-3) search space is eliminated in an iterative process, too. This means that less iterations in reducing the search space come at the cost of more iterations in lowering the upper bound, which is why a lower upper bound is not pursued.

6-2-5 Additional Constraints

A lower bound constraint can be formulated on any Monte Carlo support $B_{mc}\{j\}, \forall j : 1 \leq j \leq J$, and an upper bound constraint can be formulated on the product of all Monte Carlo supports $\prod_{j=1}^{J} B_{mc}\{j\}$. Both are derivations from the constraint on probability of not bounding:

$$\Phi_{nb} = \alpha \tag{6-28}$$

Φ_{nb} depends on $\phi_{nb}\{j\}$ as follows (equation 6-29)

$$\Phi_{nb} = 1 - \prod_{j=1}^{J} \left(1 - \phi_{nb}\{j\}\right) \tag{6-29}$$

Lower Bound

Because all $\phi_{nb}\{j\}$ are probabilities, they are bounded in the interval $[0, 1]$. Consequently, the value of the equation 6-29 is equal to or greater than the largest element $\phi_{nb}\{j\}$ for all j, or:

$$\Phi_{nb} \geq \max(\phi_{nb}\{j\} \forall j : 1 \leq j \leq J) \tag{6-30}$$

Using both equation 6-28 and 6-30, a constraint can be placed on $\phi_{nb}\{j\}$:

$$\phi_{nb}\{j\} < \alpha, \forall j : 1 \leq j \leq J \tag{6-31}$$

An upper bound constraint on $\phi_{nb}\{j\}$ is translated into a lower bound constraint on $B_{mc}\{j\}$:

$$B_{mc}\{j\} > {}^{*}B_{mc}(\alpha), \forall j : 1 \leq j \leq J \tag{6-32}$$

Upper Volume Bound

The constraint α can be translated into a maximum volume constraint. For the case $\phi_{nb}\{1\} = \phi_{nb}\{2\} = \phi_{nb}\{...\} = \phi_{nb}\{j\}$ a maximum probability of not bounding for every element α_j can be obtained:

$$\begin{aligned} \alpha &= 1 - (1 - \alpha_j)^J \\ (1 - \alpha_j)^J &= 1 - \alpha \\ \alpha_j &= 1 - (1 - \alpha)^{\frac{1}{J}} \end{aligned} \tag{6-33}$$

This maximum probability for every element can be translated into a maximum volume constraint:

$$\prod_{j=1}^{J} B_{mc}\{j\} < \prod_{j=1}^{J} B_{mc}\{j\}^{*}(\alpha_j) \tag{6-34}$$

Revisiting the optimization problem with the two final constraints added:

$$\begin{aligned} \underset{B_{mc}\{1\},\ldots,B_{mc}\{J\}}{\text{minimize}} \quad & \prod_{j=1}^{J} B_{mc}\{j\} \\ \text{subject to:} \quad & \Phi_{nb}(B_{mc}) < \alpha \\ & B_{mc}\{j\} > {}^{*}B_{mc}\{j\}(\alpha), \forall j : 1 \leq j \leq J \\ & \prod_{j=1}^{J} B_{mc}\{j\} < \prod_{j=1}^{J} B_{mc}\{j\}^{*}(1-(1-\alpha)^{\frac{1}{J}}) \end{aligned} \tag{6-35}$$

The constraint $B_{mc}\{j\} > 0, \forall j : 1 \leq j \leq J$ is replaced with the second constraint, as ${}^{*}B_{mc}\{j\}(\alpha) > 0, \forall j : 1 \leq j \leq J$.

6-3 The Minimization Algorithm

An algorithm has been made to perform the minimization problem given in 6-35. It is for several reasons not feasible to use a gradient descent, as in calculating $\Phi_{nb}(B_{mc})$ a discontinuity exists because of the $\min(x, 1)$ component. Additionally, it would require integration of an exponent multiplied by an exponential function, with the exponent $s \gg 1$, with the resulting function differentiated by B_{mc}. This is, for matlab at least, to advanced a problem to perform using symbolics. Rather, a strategy is adopted where iteratively search space is eliminated.

The initial search space Σ can be defined by the constraints:

$${}^{*}B_{mc}\{j\}(\alpha) < \Sigma\{j\} < \frac{{}^{*}B_{mc}\{j\}(\alpha)\prod_{j=1}^{J} B_{mc}\{j\}^{*}(1-(1-\alpha)^{\frac{1}{J}})}{\prod_{j=1}^{J} {}^{*}B_{mc}\{j\}(\alpha)}, \forall j : 1 \leq j \leq J \tag{6-36}$$

Finding supports for the probability density functions of the Monte Carlo is done using an iterative process. Using square brackets $B_{mc}[i]$ denotes the vector B_{mc} at iteration i. When the chance of bounding the residual $\Phi_{nb}(B_{sys}[i])$ is calculated, the constraint $\Phi_{nb} < \alpha$ can be either fulfilled or not. In both cases, a section of the search region can be eliminated.

In the case that for $B_{sys}[i]$ the constraint is met (i.e. $\Phi_{nb} < \alpha$), it is not needed to consider supports with a higher volume than $\prod_{j=1}^{J} B_{sys}\{j\}[i]$. If the constraint is not met, $\Phi_{nb} \geq \Phi$, then supports which are smaller than $B_{mc}\{j\}[i]$ for every element of $i[n]$, can not fulfill the constraint. In mathematical notation:

Region excluded when $\Phi_{nb} < \alpha$:

$$E_s = \{B_{mc}\{j\} \in B_{mc} : \prod_{j=1}^{J} B_{sys}\{j\}[i] > \prod_{j=1}^{J} B_{sys}\{j\}[i]\} \tag{6-37}$$

Region excluded when $\Phi_{nb} \geq \alpha$:

$$E_f = \{B_{mc}\{j\} \in B_{mc} : B_{mc}\{j\} < B_{sys}\{j\}[i], \forall j : 1 \geq j \geq J\} \tag{6-38}$$

Figure 6-1) is included as a visualization of the sets E_s and E_f. For an iteration at black point near the center of the figure, blue region represents E_s and the orange region represents E_f.

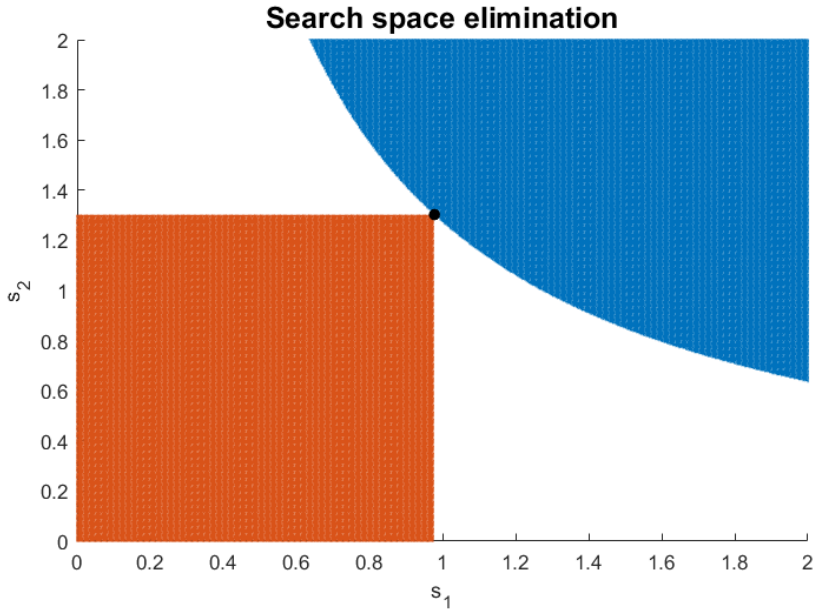


Figure 6-1: Illustration of two regions of which one can be excluded, depending on the output at the point indicated by the black dot

For every iteration, either E_f or E_s will be eliminated depending on Φ_{nb}. $B_{mc}[i]$ is chosen to maximize minimal return, i.e. $\max(\min(\text{Vol}(E_f), \text{Vol}(E_s)))$. Figure 6-1 illustrates such a point for the first iteration. By consecutively eliminating search space, an optimal value B_{opt} is found for the Monte Carlo noise supports B_{mc}.

The minimization method is given in algorithm 2:

Σ as defined by equation 6-36,
$B_{opt}\{j\} := \infty, \quad \forall 1 < j < J$
while $i < i_{max}$ **do**
  $\underset{B_{mc}[i]}{\text{maximize}} \min(\text{Vol}(E_f \cap S), \text{Vol}(E_s \cap S))$
  if $\Phi_{nb} < \alpha$ **then**
    if $\prod_{j=1}^{J} B_{opt}\{j\} > \prod_{j=1}^{J} B_{mc}\{j\}[i]$ **then**
      $B_{opt} := B_{mc}[i]$
    end
    $\Sigma := \Sigma \cap \bar{E}_s$
  else
    $\Sigma := \Sigma \cap \bar{E}_s$
  end
  i := i + 1
end

Algorithm 2: Algorithm for finding optimal B_{mc} by iterative search space elimination.

With i_{max} the maximum number of iterations and $\bar{E}_s, \bar{E}_f$ denoting the complement of E_s, E_f respectively.

6-4 Summary

A new sampling strategy is used which offers several advantages to the one used previously. A healthy residual set is created with a more uniform distribution, which means that a similar accuracy can be achieved using less samples compared to a data set with a center heavy distribution. By sampling in a finite domain fewer outliers are in the healthy residual set. Volume of this domain is minimized under a probabilistic constraint $r \in \mathcal{B}$, $\mathcal{R}_H \in \mathcal{B}$.

Further reduction in the volume of the domain of the Monte Carlo distributions could be achieved by using multivariate probability density functions for the Monte Carlo simulation. This was however outside the scope for this thesis.

Chapter 7

Anomaly Detection for Fault Detection

In previous chapters machine learning was discussed in general terms (chapter 4) and the anomaly detection algorithms used in this thesis were discussed (chapter 5). Having introduced both subjects, this chapter explains some of the procedures taken specific to the application of these methods in a fault detection context. Additionally, for the hyper parameters of all AD algorithms tuning and optimization strategies are presented.

7-1 FAR Robustness in Fault Detection

An important property of [1] is the α-robustness, a probabilistic guarantee on bounding a fraction $1-\alpha$ of future residuals. A completely analogous approach is not found in this thesis. This is partly because of the different sampling technique (chapter 6) causing outliers in the healthy residual set to be much less likely.

In this thesis, it is during the sampling that a probabilistic guarantee on bounding a fraction $1-\alpha$ of future residuals is guaranteed. The aim in tuning the AD methods is to maintain this FAR performance. This is done using the threshold τ common to every AD algorithm used. Recall from chapter 5 the following definitions for bounding sets $\mathcal{B}$ for all AD methods:

$$\begin{aligned} \mathcal{B}_{\text{PW}} &= \{x \in \mathbb{R}^n : f_{\text{PW}}(x) > \tau_{\text{PW}}\} \\ \mathcal{B}_{\text{OSVM}} &= \{x \in \mathbb{R}^n : f_{\text{OSVM}}(x) > \tau_{\text{OSVM}}\} \\ \mathcal{B}_{\text{KNN}} &= \{x \in \mathbb{R}^n : f_{\text{KNN}}(x) < \tau_{\text{KNN}}\} \end{aligned} \tag{7-1}$$

Tuning of the threshold τ in anomaly detection is usually a pay-off between MDR and FAR. A common approach for tuning τ is to assign a cost to both false alarms and missed detections, and reach as a function of τ a minimal cost [31]. For the thesis, the aim is to maintain FAR $= \alpha$ after set bounding, so the effect of τ on MDR is not considered. To reduce MDR, the remaining parameters of the machine learning method are tuned.

To make statements about FAR performance on future residuals an assumption is made:

It is assumed that performance of a classifier on a sufficiently large test set is identical to all future performance

The difference between performance on train and test sets is of major importance in guaranteeing performance on future inputs. This is illustrated in figure 7-1:

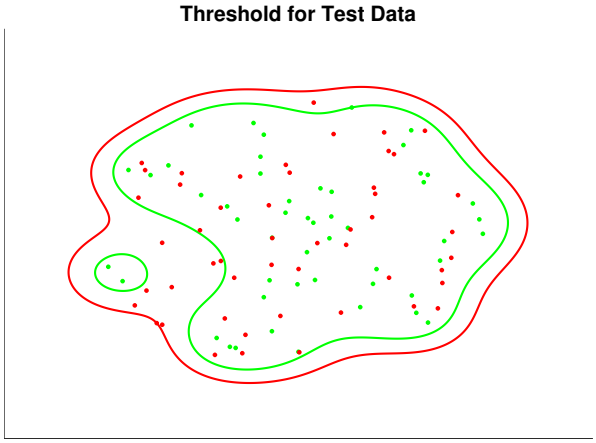
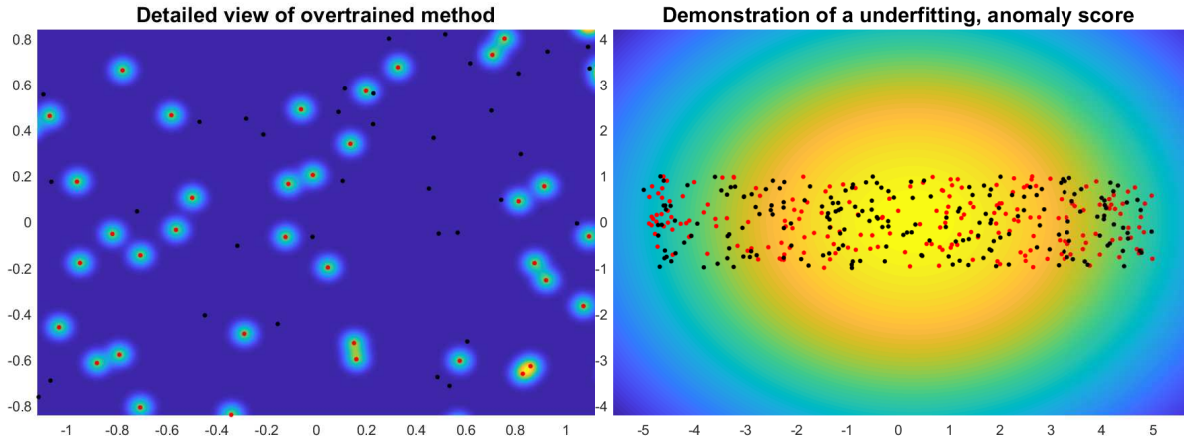


Figure 7-1: Resulting bounding sets for bounding all test samples (red), in contrast to bounding all train samples (green)

The sets in figure 7-1 result from the same classifier using two different thresholds. The green lines indicate a bounding set with a threshold based on train data, and consequently misidentifies certain test samples. The red line indicates a bounding set with a threshold based on test data, and bounds both all test and all train data.

Most machine learning methods perform markedly better on train data than on test data. In extreme cases, this is called overfitting, as opposed to underfitting. The following figures depict heat maps with outlier scores for an overfit and an underfit method (figure 7-2):



(a) Detailed view of overfit method **(b)** View of underfit method

Figure 7-2: Heatmap of anomaly scores for an underfit and an overfit classifier. Test data in black, train data in red

Both illustrations in figure 7-2 are a heatmap of anomaly scores. Red dots are train samples, black dots are test samples. For ML in general, overfitness is ascertained when a trained algorithm performs well on train data and significantly worse on test data. Underfitness manifests as a poor performance on both test and train data. This principle is seen in figure 7-2, where the overtrained algorithm (left) classifies all train samples as inliers, and virtually none of the test samples. The underfit algorithm (right) misidentifies both train and test samples. Underfitness and overfitness are both results of "complexity", with an underfit algorithm having too little, and an overfit algorithm having too much, suggesting an ideal fit in between.

To maintain FAR performance, the threshold in an AD algorithm is set at whatever value bounds all test data. This leads to the following rules for setting thresholds (equation 7-2):

$$\begin{aligned} \tau_{\text{PW}} &= \min \left[f_{\text{PW}}(x_{test}), \forall x_{test} \in \mathcal{X}_{test} \right] \\ \tau_{\text{OSVM}} &= \min \left[f_{\text{OSVM}}(x_{test}), \forall x_{test} \in \mathcal{X}_{test} \right] \\ \tau_{\text{KNN}} &= \max \left[f_{\text{KNN}}(x_{test}), \forall x_{test} \in \mathcal{X}_{test} \right] \end{aligned} \tag{7-2}$$

With thresholds a function of test data performance, a bounding set $\mathcal{B}$ is formed with FAR $= \alpha$ for user controlled probability. With FAR as a constant value subject to user requirements, the sole remaining metric relevant to classifier accuracy is $\text{Vol}(\mathcal{B})$. Set volume is approximated numerically with a grid:

$$\text{Vol}(\mathcal{B}) = \frac{\text{card}(\{x_G \in \mathcal{X}_G : x_G \in \mathcal{B}\})}{\text{card}(\mathcal{X}_G)} \tag{7-3}$$

Set volume is to be minimized using the hyper parameters of a respective AD algorithm; k for KNN, σ and ν for OSVM and σ for PW.

7-2 Hyper Parameter Optimization

The previous section introduced set volume as the single performance metric relevant to the quality of the classification. This section presents strategies for hyper parameter optimization for the AD algorithms used in this thesis. Because for kernel size σ the same optimization method is used for both PW and OSVM, these subjects are discussed together.

7-2-1 Golden Section Optimization

For both PW and OSVM, Vol($\mathcal{B}$) has a characteristic performance curve as a function of σ. Kernel scale σ controls complexity; by increasing σ the classifier performance goes from overfit to underfit. An ideal fit can be expected between overfit and underfit. This is illustrated in figure 7-3.

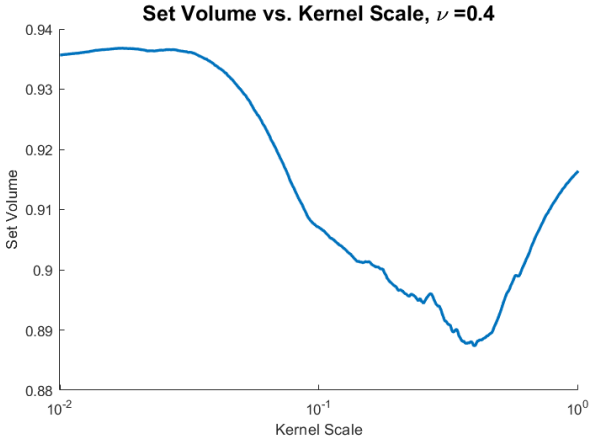


Figure 7-3: Set volume as a function of σ

Performance of kernel based methods (PW, OSVM) as a function of σ is unimodal and largely continuous. Small deviations can be seen. These are explained by inaccuracies caused by calculating set volume using a grid, and because the optimization of the OSVM has a stopping criterion which stops some hyperplane optimizations sooner than others. In this thesis this characteristic of kernel based algorithms is exploited instead of performing a grid search, a very common approach in hyper parameter optimization [32]. When viewed as an optimization problem, set volume is not differentiable as a function of σ, meaning iterative methods have to be used to minimize set volume as a function of σ. The problem is one dimensional and approximately convex. In view of these properties, golden section search [33] is the best optimization method. Golden section search is a bisectional search method for finding a minimum or maximum of a unimodal function. Each iteration removes a section of the search space from further consideration.

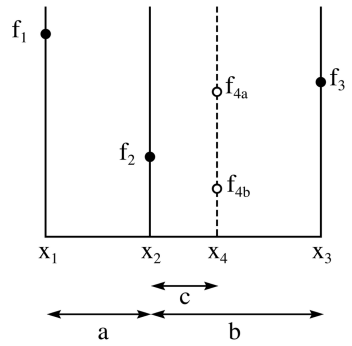


Figure 7-4: Search space elimination with golden section optimization (Public Domain Image)

In figure 7-4 three iterations have already been made, at x_1, x_2 and x_3. The values f_1, f_2 and f_3 are the outputs of a f which is minimized over x. A new iteration is taken at x_4. f_4 can be a either greater or smaller than f_2. If $f_4 < f_2$, (f_{4b} in the image), the lowest point can not be between x_1 and x_2. If $f_4 > f_2$, (f_{4a} in the image), the same goes for the interval between x_4 and x_3. A new iteration will be made in the region left after elimination of the interval where the minimum can not be, and the process of taking new iterations is repeated.

The golden section algorithm in pseudo code is:

$[x_a, x_b] = [\min(x_a, x_b), \max(x_a, x_b)]$
$\bar{\phi}_1 = \frac{\sqrt{5}-1}{2}$
$\bar{\phi}_2 = \frac{3-\sqrt{5}}{2}$
$h = x_b - x_a$
while $|(x_c - x_d)| > tolerance$ **do**
  $x_c = x_a + \bar{\phi}_2 h$
  $x_d = x_a + \bar{\phi}_1 h$
  $y_c = f(x_c)$
  $y_d = f(x_d)$
  if $y_c < y_d$ **then**
    $x_b = x_d$
    $x_d = x_c$
    $h = \bar{\phi}_1 h$
    $x_c = x_a + \bar{\phi}_2 h$
    $y_c = f(x_c)$
  else
    $x_a = x_c$
    $x_c = x_d$
    $h = \bar{\phi}_1 h$
    $x_d = x_a + \bar{\phi}_1 h$
    $y_d = f(x_d)$
  end
end

Algorithm 3: Golden section algorithm

With x_a and x_b initial points required to surround the minimum, and *tolerance* the stopping criterion.

It should be noted that for PW, this optimization has to be repeated for every kernel function because for different kernel functions different optima exist (see figure 7-5).

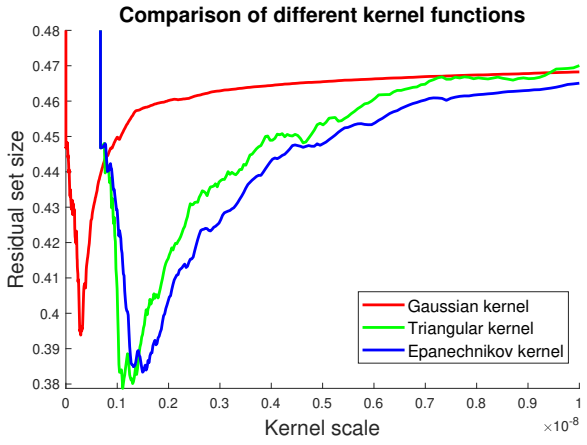


Figure 7-5: Performance of different kernel functions on the same residual set

7-2-2 Nu Optimization

In the previous section optimization of σ was discussed for both PW and OSVM. In the case of OSVM ν is an additional hyper parameter. It is suggested that optimizing over ν becomes redundant when σ-optimization is performed well. The following figure depicts a heat map of set volumes as a function of both σ and ν.

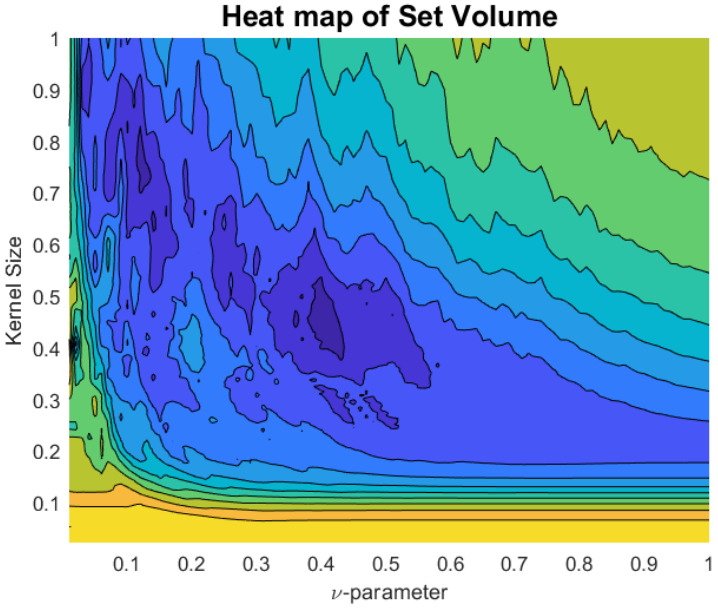


Figure 7-6: Heat map, increasingly blue colors indicate decreasing set volume

In figure 7-6 cooler tints indicate decreasing set volume. A relation can be seen between σ and ν. As nu decreases, better performances can be found for higher values of σ. Because lower

values of both parameters increase complexity, not both of them can be low simultaneously, illustrating why optimizing both is redundant. Note also the minimum set volume as a function of ν in figure 7-7.

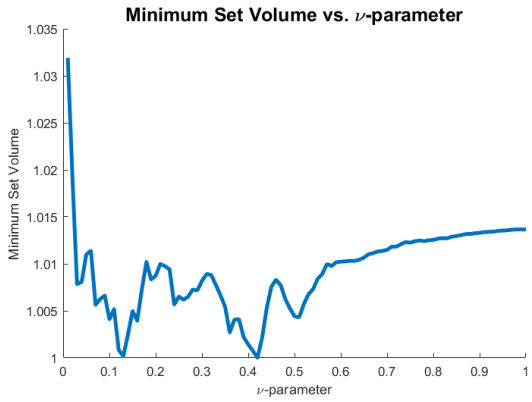
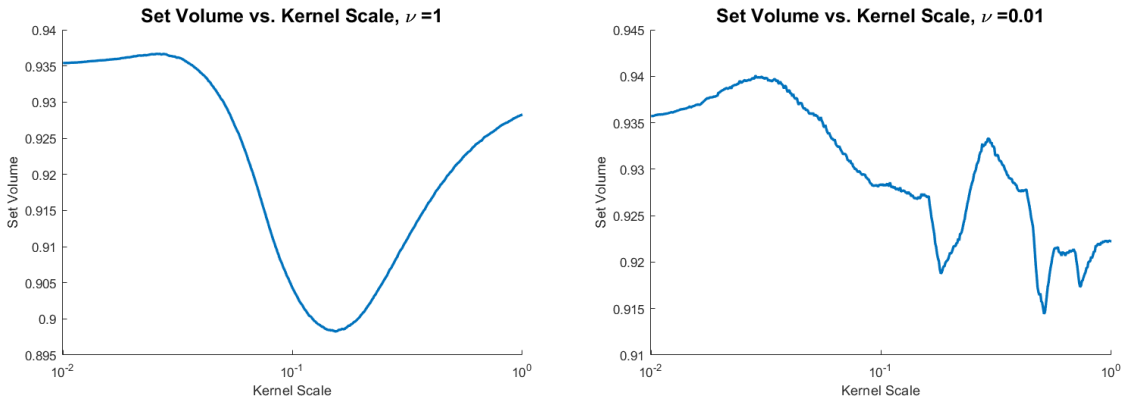


Figure 7-7: Minimum set volume as a function of ν

Minimum set volume as a function of ν steadily decreases for lower values of ν in the interval $0.5 < \nu < 1$. In the interval $0 < \nu < 0.5$ minimum set volume as a function of ν becomes much more irregular, suggesting the optimal combination of ν and σ can only be found with a grid search. However, optimizing for both σ *and* ν instead of optimizing for σ results in reducing set volume with at most 0.68% in this example. This suggests optimizing over σ only, and maintaining a constant value for ν. To motivate the choice of this constant ν an additional effect of ν is considered.



(a) A smooth performance curve for a high value of ν **(b)** An irregular performance curve for a low value of ν

Figure 7-8: Set volume as a function of σ, for extreme values of ν

It can be seen in figures 7-8a and 7-8b how set volume as a function of kernel size follows the expected trajectory closely in 7-8a. As values for ν decrease, this trajectory becomes less predictable (see figure 7-8b). Because the golden section optimization only works under the assumption that the performance curve is unimodal, a value of 0.4 is chosen for ν, a value for which the curve is still suitable for golden section iterations (see figure 7-9).

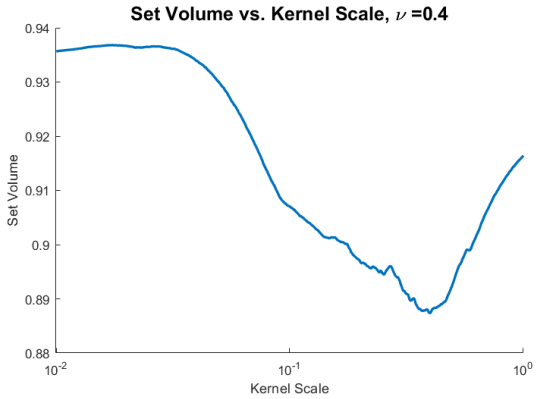


Figure 7-9: Performance curve for the chosen value of ν

7-2-3 KNN Optimization

The performance of KNN as a function of k is irregular. In the author's view this is because k is a parameter controlling noise rejection, rather than complexity. Consequently k performs less predictably. Also due to the sampling strategy (chapter 6) healthy residual sets are relatively noise free.

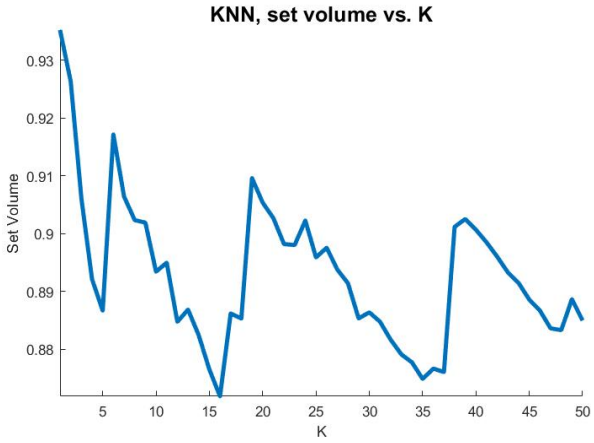


Figure 7-10: Set volume as a function of K

Grid search for k is more feasible because k is an integer value. Additionally, the same KD-tree can be used for all values of k, meaning training does not need to be repeated. This is in contrast to ν and σ, parameters which are an integral part of the training phase.

7-3 Hyper Parameter Heuristics

With hyper parameter optimization a near optimal set volume is reached at the cost of additional computation time. Computation time is a relevant metric when a different residual set needs to be classified at every time step. Set volume after hyper parameter optimization is a best case scenario for specifically the set volume performance metric. In using a hyper parameter heuristic, the best case scenario for computation time is reached.

Extensive literature on heuristics for kernel scale can be found. In contrast, literature for k was rare, and for ν non existent. With the same reasoning of section 7-2-2, ν is set $\nu = 0.4$ and left unchanged. Literature can be found on choosing k for multi class classification. An often cited heuristic in multi class classification KNN is $k = \sqrt{s}$, with s the amount of train samples. On healthy residual sets this heuristic consistently places k too high. The inaccuracy of this heuristic can be explained by it not being intended for anomaly detection and the absence of noise in the data set, with k being mostly a parameter filtering noise. Parameter k was set $k = 4$ as a heuristic.

7-3-1 Kernel Scale Heuristics

In [34] the following heuristic is given for σ in the context of OSVM:

"[...] the optimal values of the width of the hyper-parameter σ are shown to lie in between the 0.1 and 0.9 quantile of the $||x - x'||^2$ statistics."

$||x - x'||^2$ denotes a Gram matrix. In this thesis the 0.5 quantile is used as the value for σ. It should be noted that in this heuristic σ is not dependent on sample size, because variance remains unaffected by sample size. It is to be expected that for an increasing sample count σ should decrease because a relation exists between complexity, sample count and quality of fit. Several heuristics for kernel size do indeed decrease kernel size for an increasing sample size, such as Silverman's rule (equation 7-4, [27]) or Scott's rule (equation 7-5, [35]).

$$\sqrt{H_{ii}} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} s^{\frac{-1}{d+4}} \delta_i \tag{7-4}$$

$$\sqrt{H_{ii}} = s^{\frac{-1}{d+4}} \delta_i \tag{7-5}$$

With δ_i the standard deviation of the i-th variable, and s the sample count. Heuristics apply to a Guassian kernel function. These heuristics have shortcomings when applied to fault detection, which is why they were not applied. Firstly, the heuristic is only valid for Gaussian kernel functions. Only one of the three kernel functions used for PW is Gaussian. It can be seen in chapter 8, that even an optimally tuned PW with Gaussian kernel performs sub par. Secondly, in using these heuristics, the objective is to minimize the mean integrated square error (MISE) between the reconstructed distribution $\hat{p}(x, \sigma)$ and the true distribution $p(x)$ (see equation 7-6), instead of finding a kernel function for which $\mathrm{Vol}(\mathcal{B})$ is minimal.

$$\mathrm{MISE}(\sigma) = \mathrm{E}\left[\int (\hat{p}(x, \sigma) - p(x))^2\right] \tag{7-6}$$

Finally, the assumption is that the true distribution $p(x)$ is a Gaussian distribution. The true distribution $p(x)$ is known to be a non linear mapping of uniform probability density

functions. In view of these objections, the heuristic for kernel size as is applied to OSVM is applied to PW. Additional comments will be given in the discussion on the results in chapter 8.

Chapter 8

Performance Evaluation of Set Bounding Methods

This chapter builds up to a conclusion on which AD method is the best for set based fault detection, based on several test metrics and a theoretical evaluation of computation time performance. In addition, the new method for Monte Carlo sampling (chapter 6) is compared with the old method for Monte Carlo sampling.

8-1 Test Cases

It was explained in chapter 6 how a healthy residual set can be made for a general model. Two different models are used to create residual sets (section, 8-1-1, 8-1-2). Systems are discretized using Euler's method. For a given state equation:

$$\frac{\partial x}{\partial t} = f_s(x) \tag{8-1}$$

Euler discretization gives the discrete time system:

$$x[t+1] = x[t] + f_s(x)t_\Delta \tag{8-2}$$

With t_Δ a sample time chosen by the user. For simplicity all states are measured directly, meaning no observer is needed. Measurements are corrupted with additive noise:

$$y[t] = x[t] + n_y[t] \tag{8-3}$$

Both systems have at least one parameter subject to modeling uncertainties.

8-1-1 Van der Pol Oscillator

The van der Pol oscillator is a non linear system, which is expected to give challenging healthy residual set shapes. State equations for the van der Pol oscillator are as follows:

$$f_S(x, p) = \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = p(1 - x_1^2)x_2 - x_1 \end{cases} \tag{8-4}$$

With states x and parameter p, subject to modeling uncertainties. The Van der Pol oscillator can be used to demonstrate how detectability of faults varies over states. As an example, consider an instance of a Van der Pol oscillator with a corrupted parameter p_c:

$$f_S(x, p_c) = \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = p_c(1 - x_1^2)x_2 - x_1 \end{cases} \tag{8-5}$$

It can be seen that for $x_2 = 0$ and $x_1 = \pm 1$ the effect of the corrupted parameter does not propagate to the next state. As these states are approached, a fault in p becomes less detectable.

8-1-2 Three Tank System

The three tank system is widely used in fault detection simulations. The tree tank system models three connected tanks with a water inlet in the first and third tank. It is used in [1] as the test case.

$$f_S(x, p) = \begin{cases} \dot{x}_1 = \dfrac{u_1[t]}{S} - \dfrac{p_1}{S}\text{sign}[x_1 - x_2]\sqrt{2g|x_1 - x_2|} \\ \dot{x}_2 = \dfrac{p_1}{S}\text{sign}[x_1 - x_2]\sqrt{2g|x_1 - x_2|} - \dfrac{p_2}{S}\text{sign}[x_2 - x_3]\sqrt{2g|x_2 - x_3|} \\ \dot{x}_3 = \dfrac{u_2[t]}{S} - \dfrac{p_2}{S}\text{sign}[x_2 - x_3]\sqrt{2g|x_2 - x_3|} - \dfrac{p_3}{S}\sqrt{|2gx_3|} \end{cases} \tag{8-6}$$

With x_n the water level in tank n, u_1 and u_2 inputs, S the cross section of the tank, p_n the outflow coefficient and g the gravitational constant. The water level in each tank is limited in the interval $0 < x < h_{max}$. The system is discretized using Euler's method. Similarly to the Van der Pol oscillator, all states are measured directly with additive noise. The outflow coefficients p_1, p_2, p_3 are the parameters to which a modeling uncertainty applies.

8-2 Set Volume and Computation Time

This section contains performances of AD methods on set volume and computation time metrics. For MDR and FAR in simulated fault detection see section 8-3. Set volume was introduced as a heuristic for MDR performance. Computation time is chosen as the other important test metric. In practical applications a hard limit exists on the time available for performing fault detection. Computation time and set volume can not be viewed entirely separately. Set volume improves and computation time worsens for increasing train set sizes, so to a degree performance in one metric can be traded for performance in another.

In addition to set volume and computation time, two scenarios are considered, as were introduced in chapter 7. In the first scenario exhaustive optimization methods for the AD methods are completed to find the best possible tuning parameters for a minimum set volume. The computation time includes the entire optimization process. This category is a best case scenario for set volume. Its relevancy pertains to processes in which computations can be front loaded, as in chapter 9.

In the second scenario parameters are found with a heuristic method. The complexity of an optimization becomes less of a problem if performance without optimization is satisfactory. This is a best case scenario for computation time. Summarizing, all methods are compared on two metrics in two scenarios:

Smallest Set Volume		**Fastest Computation Time**	
Set Volume (SS)	Computation Time (SS)	Set Volume (FC)	Computation Time (FC)

For SLSB no parameter optimization is performed. The aim of this thesis is to find an improvement over methods already in use, not to improve methods already in place. This is why in tables 8-2, 8-1 the same results are used both for heuristic and optimization for SLSB. Sample count and polynomial degree for SLSB are identical to the settings in [1].

8-2-1 Set Volume and Computation Time Results

In the following tables the results of AD methods on all performance metrics are recorded for both test cases. The results are normalized to the best result in any metric. This is also why heuristic and optimal values for SLSB appear different, despite being the same values.

Table 8-1: AD performance metrics on Van der Pol oscillator data, 500 train samples

	Set Vol. (SS)	Comp. Time (SS)	Set Vol. (FC)	Comp. Time (FC)
KNN	1	1	1	1
OSVM	1.0088	35.1594	1.3598	1.5554
PW, Epan.	1.0158	37.2049	1.3421	8.2768
PW, Triang.	1.0204	45.4374	1.4047	10.0754
PW, Gaussian	1.0167	221.2599	1.3421	49.0123
SLSB	1.3623	2.7855	1.2882	5.2436

Table 8-2: AD performance metrics on three tank system data, 256 train samples

	Optim. Volume	Optim. Time	Heur. Volume	Heur. Time
KNN	1.0097	1	1.1198	1
OSVM	1.028	29.3436	1.1247	2.9467
PW, Epan.	1.0368	58.9086	1.1231	14.1575
PW, Triang.	1.0331	77.7773	1.1282	17.4895
PW, Gaussian	1.0276	349.4771	1.1192	97.4862
SLSB	1	6.5783	1	16.9628

On the Van der Pol oscillator data KNN has superior performance to all other AD methods in every performance metric. On three tank data KNN is outperformed by SLSB on set volume performance, in both scenarios. It should be noted however that on set volume performance KNN and SLSB differ with a very small margin, whereas KNN outperforms SLSB on computation by a large margin. In section 8-3 it can be seen that for an increased amount of train data KNN does outperform SLSB on all metrics.

8-3 MDR, FAR and Computation Time

In the previous section AD methods are compared on set volume, which ultimately is a heuristic for MDR performance. In this section data sets are created for simulated healthy and simulated faulty systems. KNN and SLSB are compared on FAR and MDR for these sets. As is typical in machine learning, confusion tables are used to compare performance, with a blue cell indicating FAR performance and a red cell indicating MDR performance.

8-3-1 Van der Pol Oscillator Simulation

A fault is introduced as a 50% decrease in μ. Performance on 500 instances of faulty behavior, and 500 instances of healthy behavior.

Table 8-3: Confusion table, Van der Pol data classified by KNN in 2.6324 seconds:

KNN	Healthy Residuals	Faulty Residuals
Classified as Faulty	0	0.4478
Classified as Healthy	1	0.5522

Table 8-4: Confusion table, Van der Pol data classified by SLSB in 68.2866 seconds:

SLSB	Healthy Residuals	Faulty Residuals
Classified as Faulty	0	0.4458
Classified as Healthy	1	0.5542

8-3-2 Three Tank System Simulation

A fault is introduced as a 10% decrease in the cross sectional area connecting the second to the third pipe. Performance on 500 instances of faulty behavior, and 500 instances of healthy behavior. The most important observation is that for an increased train sample count (256 to 500), KNN now outperforms SLSB on MDR.

Table 8-5: Confusion table, three tank data classified by KNN in 4.2531 seconds

KNN	Healthy Residuals	Faulty Residuals
Classified as Faulty	0	0.3320
Classified as Healthy	1	0.6680

Table 8-6: Confusion table, three tank data classified by SLSB in 214.6230 seconds

SLSB	Healthy Residuals	Faulty Residuals
Classified as Faulty	0.0040	0.1760
Classified as Healthy	0.9960	0.8240

8-4 Comparing Sampling Methods

The sampling method used in [1] is compared with the sampling method explained in chapter 6 on fault detection performance for a simulated three tank scenario. The Monte Carlo supports in the new sampling method were minimized for a probability of not bounding $\alpha = 0.01$. Using identical measurements 500 healthy residual sets were created for healthy and faulty system behavior each using both sampling methods. Fault detection was performed with KNN. MDR and FAR performance is recorded in table 8-7:

Table 8-7: MDR and FAR performance on healthy residual sets using old and new sampling methods

	Old Sampling Method	New Sampling Method
FAR	0.0280	0
MDR	0.002	0.002

For FAR a marked improvement is seen. The old sampling method gives 14/500 false alarms, with the new sampling method giving 0/500. For MDR the old sampling method outperforms the new sampling method slightly, with 2/500 missed detections compared 2/500 missed detections. The new sampling method, although slightly outperforming the old method, seems not to provide a major improvement. It is the author's expectation that using multivariate noise is essential in truly reducing

8-5 Computational Complexity of Anomaly Detection Methods

Computation time is used throughout the previous sections as a performance metric. Using empirical methods only a relatively limited amount of test cases can be analyzed. Computational complexity gives a broader view of computation time, indicating what performance to expect for increasing sample size and dimensions.

8-5-1 Discrepancy Between Complexity and Time

Computational complexity is typically denoted as $O(x)$, with O the number of operations to achieve some result as a function of x. Although computational complexity is strongly correlated with computation time additional factors exist which influence computation time. Some context is given for computational complexity.

Computational complexity is given in literature as either a worst case scenario or an average scenario. Obviously the worst case is not representative of what is to be expected during operation. The average case, though more representative, is often a theoretical performance

on a completely random input. The total number of operations affects computation time in one of two ways. If operations can be performed in parallel (e.g. matrix multiplication), the computation time can be diminished proportionally to how much computational power is available. If not, a fundamental limit exists on the minimal computation time.

Memory plays an integral part in computation time. Certain algorithms (e.g. sorting) can be performed faster if more memory is available [36]. However, CPU speed is typically much faster than RAM speed, so memory intensive algorithms decrease in speed especially for larger data sets [37].

8-5-2 Computational Complexity for Super Level Set Bounding

For many AD methods the computational complexity can be found in literature. SLSB is a comparatively obscure method so a few additional steps are needed to estimate its computational complexity. SLSB is performed using linear programming. The computational complexity of linear programming using the interior programming algorithm from [38] is as follows (equation 8-7):

$$\begin{aligned} &L := L_0 + n_m n_n + n_m + n_n \\ &O(\sqrt{n_n} L) \end{aligned} \tag{8-7}$$

With n_n the amount of variables to be optimized and n_m the amount of constraints. L_0 is the length of the binary encoding of variables and constants. Put simply, L_0 is the numerical accuracy.

Both the amount of constraints and the amount of variables scale poorly with dimension. A set of grid points $\mathcal{X}_G$ is used to enforce non-negativity. For each grid point a constraint is made in the linear program. The amount of grid points scales exponentially with dimensions. Each point in the train set is required to be larger than a threshold value. The total amount of constraints is:

$$n_m = n_g^d + n_{train} \tag{8-8}$$

With n_g the amount of grid points per dimension, d the dimension and n_{train} the amount of train samples. The amount of variables is a function of maximum degree of the monomials and the dimension, see equation 8-9:

$$n_n = \binom{n_p + d}{d} = \frac{(n_p + d)!}{d! n_p!} \tag{8-9}$$

With n_p the maximum degree of the monomial and d the dimension of the data.

The function in equation 8-9 is not intuitive, so n_n is given for some values of n_p and d in table 8-8:

	d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8	d=9	d=10
n_p=1	2	3	4	5	6	7	8	9	10	11
n_p=2	3	6	10	15	21	28	36	45	55	66
n_p=3	4	10	20	35	56	84	120	165	220	286
n_p=4	5	15	35	70	126	210	330	495	715	1001
n_p=5	6	21	56	126	252	462	792	1287	2002	3003
n_p=6	7	28	84	210	462	924	1716	3003	5005	8008
n_p=7	8	36	120	330	792	1716	3432	6435	11440	19448
n_p=8	9	45	165	495	1287	3003	6435	12870	24310	43758
n_p=9	10	55	220	715	2002	5005	11440	24310	48620	92378
n_p=10	11	66	286	1001	3003	8008	19448	43758	92378	184756

Table 8-8: Amount of tuneable parameters n_n as a function of dimension d and maximum monomial degree n_p

Summarizing all constraints and variables, the computational complexity of bounding a set using SLSB is:

$$\begin{aligned} n_n &= \frac{(n_p + d)!}{d! n_p!} \\ n_m &= n_c^d + n_{train} \\ &O_{train}\left(\sqrt{n_n}(L_0 + n_m n_n + n_m + n_n)\right) \end{aligned} \tag{8-10}$$

With n the maximum degree of the monomials, d the dimension of the data, n_{train} the amount of train samples used and n_c the amount grid points per dimension. Given that the application SLSB is very specific to [1], no literature exists on the test complexity. However, it can be reasoned what the complexity is. Once the coefficients have been computed, testing involves computing all monomials and and a matrix multiplication with the coefficients, bringing the test efficiency to:

$$O_{test}\left(n_{test} \sum_{i=2}^{n_p} \left[\frac{(i+d-1)!}{d!(i-1)!} L_0^i\right] + n_n n_{test}\right) \tag{8-11}$$

With n_{test} the amount of test samples. Computational complexity of scalar multiplication differs depending on the algorithm used [39]. In 8-11 a worst case scenario is used.

8-5-3 Computational Complexity in Context

Computational complexity of SLSB is compared with KNN and OSVM. Because of the prominence of KNN and OSVM values for computational complexity can be obtained directly from literature.

Complexity SLSB:

$$
\begin{aligned}
&n_n = \frac{(n_p + d)!}{d! n_p!} \\
&n_m = n_c^d + n_{train} \\
&O_{train}\left(\sqrt{n_n}(L_0 + n_m n_n + n_m + n_n)\right) \\
&O_{test}\left(n_{test} \sum_{i=2}^{n_p} \left[\frac{(i+d-1)!}{d!(i-1)!} L_0^i\right] + n_n n_{test}\right)
\end{aligned} \tag{8-12}
$$

Complexity KNN [40]:

$O_{train}(n_{train} \log n_{train})$

$O_{test}(\log n_{train})$

Complexity OSVM [41],[42]:

$O_{train}(n_{train}^3)$

$O_{test}(n_{test} \cdot d)$

It can be seen that KNN is the most efficient in the train phase, and OSVM is the most efficient in the test phase. Depending on the choice of parameters SLSB can train faster than the OSVM, yet train performance of SLSB deteriorates quickly for increasing dimensions. Further comments on the computational complexity of AD methods will be given in section 8-7 of this chapter.

8-6 Summary

Super Level Set Bounding, K-Nearest Neighbor, One class Support Vector Machines and Parzen Window Density Estimation have been compared on performance metrics relevant to set based fault detection. A new method of sampling was introduced to minimize missed detection rate for a user controlled maximum false alarm rate. Two test cases were used to create data sets.

It was found that KNN outperforms the rival methods on the following performance measures; false alarm rate, missed detection rate, set volume and computation time. It is concluded that KNN is a superior set bounding method for set based fault detection. An analysis of computational complexity suggests that KNN is more efficient in higher dimensional data than SLSB. The new sampling method proved successful in reducing MDR for a user controlled probabilistic guarantee on FAR.

8-7 Further Comments

As an addendum to the summary of results some further comments are given on the performance of several AD methods, and to note some additional results.

8-7-1 Computational Complexity

KNN has the best scaling of computational complexity as a function of dimensions when compared with SLSB and OSVM. The amount of monomials used in SLSB grows as a function of dimension and maximum monomial degree. This creates a growing amount of variables for the optimization problem. Additionally, to compute an anomaly score a growing amount of scalar operations needs to be performed. In contrast, using OSVM it is only required to compute the distance from a point to a hyperplane. It follows that although SLSB has a reasonable performance for lower dimensions, as evidenced by the results on the three tank system data set, for growing dimensions worse and worse performance is to be expected. The circumstances in which SLSB performs well are therefore rather niche.

8-7-2 Sampling Method

A new sampling method has been proposed to decrease FAR (for more detail see chapter 6). Both methods of set creation were compared on a simulated fault detection for a three tank system. The new sampling method achieves lower FAR than the old sampling method. MDR performance was equal for both methods, although the aim was for the MDR to be lower. It is expected that when using multivariate Monte Carlo noises this can be improved. By using uncorrelated uniform noises, the set of all Monte Carlo noises is a hyper box, which analogously to set bounding of residuals, is not a complex enough shape to bound with minimum volume. The importance of sampling in a wider range is especially important for lower sample counts.

8-7-3 Computation Time Performance of Parzen Window

PW methods perform poorly in computation time metrics. One would expect a PW to have lower computation time than the OSVM for lack of an optimization process. Several factors may explain the inferior performance.

The matlab documentation for SVMs mentions a paper [43] which mentions matrix factorization in order to reduce the amount of data used to represent the Gram matrix. The amount of elements in the Gram matrix scales quadratically with sample count and is therefore a major bottleneck on performance. Similar methods of reducing data requirements have not been implemented for the PW, as it is considered out of scope for the thesis. Additional code optimization can also be expected to decrease required computation time. However, a faster PW would still be outperformed by KNN on a set volume performance. This is why, in the view of the author, improving PW computation time performance is not worth pursuing.

8-7-4 Kernel Scale Heuristics

As was noted in the section on kernel scale heuristics (section 7-3-1), it is expected that better heuristics exist for kernel based methods. It can be seen however that even for a fully optimized kernel size, set volume is still inferior. In view of these results it is not worth finding a better heuristic, which would possibly be an adaptation of Scott's rule or Silverman's rule.

Part IV

Whereas part III offers incremental improvements to existing set based fault detection, part IV explores a fundamentally different approach to set based fault detection.

Chapter 9

An Alternative Method for Data Generation and Classification

The method explained previously was one in which a residual set is created for a specific time set, followed by data preparation, a train phase and a test phase and optionally meta optimization. A new method is proposed, in which all computations except for testing a residual need be performed only once to create a single classifier which is suited to all future time steps. Optimization strategies are presented for this classifier. Finally, the novel approach is compared to the previous method of fault detection.

9-1 Introduction to Residual Space

Throughout this chapter the previous method of creating a healthy residual set at every time step will be compared to the new method. A healthy residual set created during a time step will be referred to as **TGR** (Time step Generated Residual sets). Performing fault detection using TGR will be referred to as **TGR Fault Detection (TGR FD)**. In equation **??** a representation is given of how healthy residual sets are created in TGR:

$$\begin{aligned}
\mathcal{R}_H &= \langle g(\hat{\chi}, \hat{p}, u, \mathcal{N}_p, \mathcal{N}_{y0}, \mathcal{N}_{y1}) \rangle \\
\mathcal{N}_p &= \langle \tilde{n}_p\{1\}, \tilde{n}_p\{2\}, ..., \tilde{n}_p\{s\} \rangle \\
\mathcal{N}_{y0} &= \langle \tilde{n}_{y0}\{1\}, \tilde{n}_{y0}\{2\}, ..., \tilde{n}_{y0}\{s\} \rangle \\
\mathcal{N}_{y1} &= \langle \tilde{n}_{y1}\{1\}, \tilde{n}_{y1}\{2\}, ..., \tilde{n}_{y1}\{s\} \rangle
\end{aligned} \tag{9-1}$$

A healthy residual set $\mathcal{R}_H$ is created for one specific state estimate $\hat{\chi}[t]$ and the current input $u[t]$. For $\langle \hat{\chi}[t], u[t] \rangle$ all feasible residuals are simulated by using sets of noises $\mathcal{N}_p$, $\mathcal{N}_{y0}$, $\mathcal{N}_{y1}$. Function g_{res} denotes the process of creating hypothetical residuals as explained in section 2-3.

It is proposed in the new approach that residuals not be created for a specific $\langle \hat{\chi}[t], u[t] \rangle$, but rather for randomly generated sets $\mathcal{X}$, $\mathcal{U}$ sampled in the expected range of $\langle \hat{\chi}[t], u[t] \rangle$ during operation. The resulting set is defined in equation 9-2:

$$\mathcal{M}_H = \langle \mathcal{X}, \mathcal{U} \langle g(\mathcal{X}, \hat{p}, \mathcal{U}, \mathcal{N}_p, \mathcal{N}_{y0}, \mathcal{N}_{y1}) \rangle \rangle \tag{9-2}$$

with $\tilde{m}_H \in \mathcal{M}_H$ a tuple $\langle \tilde{\chi}, \tilde{u}, r_H \rangle$. Whereas the healthy residual set $\mathcal{R}_H$ filled the space of all possible healthy residuals for a specific state-input pair $\langle \hat{\chi}[t], u[t] \rangle$, the set $\mathcal{M}_H$ should be thought of as filling the space of all possible healthy residuals, for all possible state-input pairs. An element $\tilde{m}_H \in \mathcal{M}_H$ is a point in a space of d_m dimensions, as defined in the following equation (equation 9-3):

$$\begin{aligned} x &\in \mathbb{R}^{d_x}, \quad d_\chi \in \mathbb{Z} \\ u &\in \mathbb{R}^{d_u}, \quad d_u \in \mathbb{Z} \\ r &\in \mathbb{R}^{d_r}, \quad d_r \in \mathbb{Z} \\ m_H &\in \mathbb{R}^{d_m}, \quad d_m = d_\chi + d_u + d_r \end{aligned} \tag{9-3}$$

For a sufficient amount of samples the set $\mathcal{M}_H$ bounds all possible healthy combinations of state $\tilde{\chi}$, input $\tilde{u}$ and residual $\tilde{r}_H$. It will be referred to as the **Residual Space (RS)** and fault detection using such a set will be referred to as **RS Fault Detection (RS FD)**.

9-1-1 Motivations for the Novel Approach

The advantages of RS fault detection should be viewed in context of its alternative, TGR fault detection. Let us list all the processes that take place during every time step of TGR fault detection:

1. Creating a healthy residual set $\mathcal{R}_H = \langle g(\hat{x}[t], \hat{p}, u[t], \mathcal{N}_p, \mathcal{N}_{y0}, \mathcal{N}_{y1}) \rangle$
2. Training an AD classifier on $\mathcal{R}_H$
3. Performing hyper parameter optimization, possibly involving the training of additional AD classifiers
4. Testing $r[t+1]$ on the optimal classifier classifying it as either healthy or faulty
5. Clearing all data and classifiers from memory

Several characteristics of TGR fault detection contribute to either wasteful use of computations or decreasing accuracy of residual evaluation. The accuracy of the AD classifier is limited by the available computation time in two ways:

- Hyper parameter optimization can be performed more exhaustively depending on available time.
- Accuracy can be increased by using more train data, which also increases train time.

Instances of wasteful use of computational power occur in TGR fault detection. Consider a state being approximately equal to a past state; $\langle \hat{\chi}[t], u[t] \rangle \approx \langle \hat{\chi}[t-n], u[t-n] \rangle$. The residual $r[t]$ could accurately be classified with the classifier of time step $t-n$, yet TGR fault detection requires that past data and classifiers be cleared. A repetition of calculations already performed is required.

The aforementioned drawbacks motivate an approach in which only step 4 of TGR fault detection (classifying the residual) is performed at every time step. The RS is a set which encompasses all expected states and therefore suffers none of the aforementioned drawbacks to TGR Fault Detection. Set creation, classifier training and optimization need only be performed once. Because these steps are carried out once only, the strategy in RS fault detection is to create one extremely optimized classifier. The performance of this classifier is evaluated on test time and accuracy.

Another factor is the ability to incorporate historical data in fault detection. Aside from using known distributions or confidence intervals, it is noted in [1] that historical data can be incorporated when creating healthy residual sets. In TGR FD it is infeasible to compare a received residual against healthy residuals from historical data, because it is rare to find healthy residuals created for the exact same state-input pair. Rather, healthy residuals are used to create exact values for uncertainties, such that hypothetical residuals can be created. In RS FD a residual is not compared with residuals created under identical state-input pairs, but with residuals sampled in a continuum of states and inputs. Incorporation of healthy residuals is therefore easier.

9-1-2 Challenges to the Novel Approach

Having explained what motivates the new approach, attention is given to the challenges, which are mainly consequences of the "curse of dimensionality". The curse of dimensionality is a phrase which, although not necessarily exclusive to machine learning, is often used in reference to machine learning problems which become exponentially harder in proportion to the dimension of the data, such as anomaly detection. The relevancy of the curse of dimensionality to this chapter is that RS by definition has a higher dimension than TGR; $d_m = d_\chi + d_u + d_r$.

Typically $d_x > d_r$, in which case the dimension of the RS is at least double that of TGR. As an example, the three tank system which was used previously as a test case has three states, two inputs and three outputs, creating an 8-dimensional RS. In [44] the observation is made that pair wise distances between samples tend to concentrate around the same value for increasing dimensions.

Sample density decreases as the amount of dimensions increases. Strictly speaking the volume of a unit hypercube remains constant as a function of dimension, so the average samples per volume remains constant also. Yet the density measures used in the used AD algorithms are based on inverse Euclidean distance between samples, which increases proportionally to dimension. Consider a uniform distribution in a unit cube of d dimensions, using n samples. The minimum distance between samples as a function of dimension is given in equation 9-4:

$$D_{min} = \frac{1}{\sqrt[d]{n} + 1} \tag{9-4}$$

With D_{min} the minimum distance between samples. For increasing dimensions density measures become lower, decreasing the contrast between $\mathcal{B}$ and its complement. Similar problems occur when numerically computing set volume with a grid, as distance between grid points increases exactly as in equation 9-4.

The reduced accuracy for both the AD classifier and the measurement of set volume need to be compensated for with an increased amount of train samples and grid points respectively. The increasing data requirements create secondary problems. The goal for RS is to have as fast a test time as possible. Test time increases proportionally to how much train data is used in constructing a classifier. Although train time is of lesser importance if a train phase is performed only once, it is certainly desirable to keep the train time within practical limits. Section 9-4 is dedicated to selecting from available train data a subset of data such that with minimal data usage the best possible classifier is created. The proposed method optimizes both train- and test time, as well as set volume. Methods for set volume computation beyond grid based are developed and tested in section B of the appendices.

9-2 Proposed Anomaly Detection Procedure for Large Train Sets

Summarizing the previous sections, the classifier's most important performance metrics are test time and set volume. Train- and optimization time are of some but lesser importance. Due to the increased dimensionality of the problem, a large train set is required to achieve good performance.

A method is proposed to meet the requirements which is to the best of the authors knowledge original. A brief introduction is given in this section because the explanation of the complete method requires two sections (sections 9-3, 9-4). A Two Class Support Vector Machines **(BSVM)** is used as the classifier for RS fault detection. Note that in literature BSVM is never used as an abbreviation as typically SVM is used. However, "SVM" will be used to denote one specific support vector machine while explaining algorithms, as opposed to the discussions of general properties of the BSVM. Section 9-3 explains how to make an unsupervised machine learning problem into a supervised one and how to minimize set volume. Section 9-4 builds on the notion of anomaly detection as supervised learning and explains how for a supervised learning method large train sets can be reduced to so called minimal train sets resulting in a low test time BSVM .

9-3 Anomaly Detection as Artificial Binary Classification

This section is focused on a proposed method for using the BSVM in an iterative method for decreasing set volume of the bounding set. The reader might recall that supervised methods (chapter 4) such as the BSVM are considered by the author unsuitable for anomaly detection, due to only one class having labels which would suggest the need for unsupervised machine learning. This class is denoted as $\mathcal{I}$, for inliers. It is proposed that any set based fault detection problem can be made into a two class classification problem.

A minimum volume bounding set $\mathcal{B}_{min}$ is required for some set of known inliers $\mathcal{I}$. The complement of $\mathcal{I}$ is the set of outliers $\mathcal{O}$. The first step in the approach is to construct a

conservative bounding set $\mathcal{B}_0 \supset \mathcal{I}$. This set need not be optimized for set volume, but is strictly required to completely bound $\mathcal{I}$. Given that $\mathcal{B}_0$ is conservative, for an arbitrary point $x \in \mathcal{B}_0$ in bounds it is not known whether x is correctly classified as an inlier, $i \in \mathcal{I}$, or an outlier, $o \in \mathcal{O}$. However, an arbitrary point $x \notin \mathcal{B}_0$ *not* in bounds, is certainly an outlier. This can be used to create for $\mathcal{B}_0$ a set of samples which are certainly anomalous. This set is defined as:

$$\mathcal{O}_0 = \{x \in \mathbb{R}^n : x \notin \mathcal{B}_0\}, \quad \mathcal{O}_0 \subset \mathcal{O} \tag{9-5}$$

The set $\mathcal{O}_0$ is created by random sampling and rejection. A Two Class SVM is then trained on the available sets:

$$\text{SVM}_0 := \text{Train}(\langle \mathcal{I}, l_i \rangle, \langle \mathcal{O}_0, l_o \rangle) \tag{9-6}$$

The previous notation (equation 9-6) is used for indicating the training of an SVM. In the brackets of "Train" the train sets are indicated in angle brackets $\langle\rangle$ with their respective label either l_i or l_o. SVM_0 provides a new bounding set, $\mathcal{B}_1 \supset \mathcal{I}$. A property of key importance is:

$$\mathcal{I} \subset \mathcal{B}_1 \subset \mathcal{B}_0 \tag{9-7}$$

The SVM separates the classes by the widest possible margin, leading to a new decision boundary in between $\mathcal{O}_0$ and $\mathcal{B}_0$, closer to the inliers than the previous bounding set. A new set of outliers $\mathcal{O}_1$ is created using bounding set $\mathcal{B}_1$, and the entire process is repeated for a certain amount of iterations:

$$\begin{aligned} \mathcal{O}_n &= \{x \in \mathbb{R}^n : x \notin \mathcal{B}_n\} \\ \text{SVM}_n &= \text{Train}(\langle \mathcal{I}, l_i \rangle, \langle \mathcal{O}_n, l_o \rangle) \end{aligned} \tag{9-8}$$

With n denoting the current iteration. The consecutive bounding sets adhere to the properties:

$$\begin{aligned} &\mathcal{I} \subset \mathcal{B}_n \subset \mathcal{B}_{...} \subset \mathcal{B}_1 \subset \mathcal{B}_0 \\ &\text{Vol}(\mathcal{I}) < \text{Vol}(\mathcal{B}_n) < \text{Vol}(\mathcal{B}_{...}) < \text{Vol}(\mathcal{B}_1) < \text{Vol}(\mathcal{B}_0) \end{aligned} \tag{9-9}$$

In application deviations from the trend in equation 9-9 occur due to finite and random sampling. A visualization of the principle is included in figure 9-1:

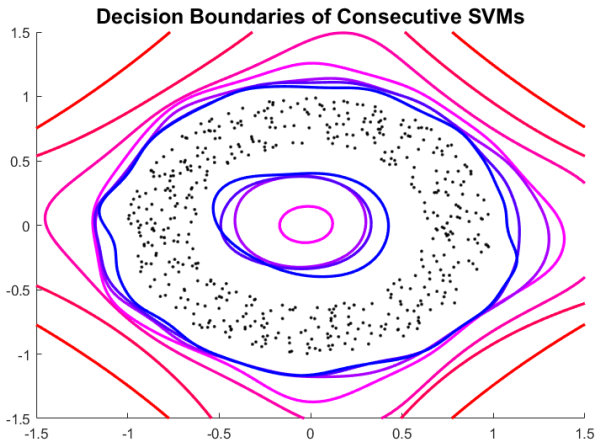


Figure 9-1: A demonstration of consecutive SVMs reaching an increasingly accurate fit. Lines are increasingly blue for later iterations.

In the example of figure 9-1 the shape of $\mathcal{I}$ consists in a hollow circle. The initial bounding set $\mathcal{B}_0$ is a bounding box with vertices $\{-1.5, +1.5\}$. The set $\mathcal{O}_0$ consists in four samples at the four corners of the bounding box. Consecutive sets $\mathcal{O}_n$ were generated using random sampling and rejection. Colors of consecutive decision bounds move from red to blue. It can be seen how bounding sets adhere to the properties described in equation 9-9.

A certain limit exists on the gains that can be made on set volume with consecutive SVMs. The volume difference $\text{Vol}(\mathcal{B}_{n+1}) - \text{Vol}(\mathcal{B}_n)$ decreases for increasing n. As the decision bound nears the set of inliers $\mathcal{I}$, an increasing amount of train samples is required to prevent an intersection of $\mathcal{O}_n$ with $\mathcal{I}$. Optional stopping criteria are therefore:

$$\begin{aligned} &\text{Vol}(\mathcal{B}_{n+1}) - \text{Vol}(\mathcal{B}_n) < c_{\text{vol}} \\ &\frac{\text{Card}(\{i_{test} \in \mathcal{I}_{test} : \text{SVM}_n(i_{test}) \neq l_i\})}{\text{Card}(I_{test})} > c_{\text{FAR}} \\ &n \geq n_{max} \end{aligned} \tag{9-10}$$

Which are, in order, a lack of gains in set volume minimization, an exceedingly high FAR performance on test data or reaching maximum iterations as set by the user. The notation $\text{SVM}_n(x)$ represent the output label of the trained SVM of iteration n on a sample x.

9-4 Iterative Train Data Selection

In the previous section (section 9-3) the choice of BSVM was motivated as a way of decreasing $\text{Vol}(\mathcal{B})$. This section is dedicated to explaining how the BSVM can accurately classify a region using less train samples than an anomaly detection method. The proposed approach requires the method explained in the previous section, and continues with the notations used in the previous section.

The abundance of data in model based fault detection is equally an asset and a challenge. Both the inliers $\mathcal{I}$ and the outliers $\mathcal{O}_n$ are the samples of a stochastic process for which an unlimited amount of samples can be generated. Accuracy of BSVMs increases proportionally to how much train data is used, yet both train and test time increase and the total amount of train data that can be used is limited. An algorithm is proposed which selects a subset of the available train data in such a way that classification accuracy does not decrease. A key factor in this algorithm is that the test time of a BSVM is much faster than the train time of a BSVM on sets of equal size.

It should first be explained why BSVMs are able to classify regions using much less data than anomaly detection methods. A fundamental difference exists between how supervised and unsupervised methods define a region. All of the anomaly detection methods used in previous chapters are directly or indirectly based on sample density. Consider a bounding set $\mathcal{B}$ formed for some arbitrary train set $\mathcal{I}$. All points $x \in \mathcal{B}$ must have a local density higher than the global threshold. It is possible for some train samples to be removed, yet only under the constraint that the density at every point $x \in \mathcal{B}$ remains equal to or higher than the threshold. Reduction of train data in this manner only serves to make the density of samples in $\mathcal{B}$ more uniform. The BSVM functions fundamentally different. It requires mostly samples near the decision boundary separating classes. Within the regions defined by the separating

hyperplane no minimum sample density is required. This is why a BSVM can classify the same region using less data.

A procedure is proposed for selecting the minimum required train data for a well performing BSVM. The subject of creating a BSVM using minimal train data is not new, as will be discussed in section 9-5. However, to the best knowledge of the author the specific method proposed in this section is original. In keeping with the notation used in the previous section, two classes are considered; an inlying class $\mathcal{I}$ and an outlying class $\mathcal{O}_n$. Part of the set of inliers is separated for use as a validation set $\mathcal{I}_{val}$. In an iterative process subsets of the available train data are created:

$$\mathcal{I}_{min} \subset \mathcal{I}_{train}, \quad \mathcal{O}_{min} \subset \mathcal{O}_n \tag{9-11}$$

$\mathcal{I}_{min}$ and $\mathcal{O}_{min}$ denote minimal train sets. At first, a simplified version of the algorithm is explained to clarify the general concept. After this a version will be presented which uses additional optimizations.

Random batches of $\mathcal{I}_{train}$ and $\mathcal{O}_{train}$ are sampled to initialize $\mathcal{I}_{min}$ and $\mathcal{O}_{min}$. An SVM is trained on the data:

$$\begin{aligned}
&\mathcal{I}_{min} := R(\mathcal{I}_{train}, b) \\
&\mathcal{O}_{min} := R(\mathcal{O}_{train}, b) \\
&\mathcal{I}_{train} := \{i_{train} \in \mathcal{I}_{train} : i_{train} \notin \mathcal{I}_{min}\} \\
&\mathcal{O}_{train} := \{o_{train} \in \mathcal{O}_{train} : o_{train} \notin \mathcal{O}_{min}\} \\
&\text{SVM}_1 := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)
\end{aligned} \tag{9-12}$$

$R(\mathcal{X}, b)$ denotes sampling b, the batch size, amount of random samples from set $\mathcal{X}$. The first SVM_1 is constructed on a small subset of all available train data. The next step is to seek train data which most increases accuracy if it were added to the current minimal train set. To this end, all train data is tested on SVM_1. Train data correctly classified by SVM_1, is considered data from which little can be learned. A new random batch is sampled from the wrongly classified train data, and added to the current minimum train data.

$$\begin{aligned}
&\mathcal{I}_{min} := \mathcal{I}_{min} \cup R(\{i_{train} \in \mathcal{I}_{train} : f_{svm_1}(i_{train}) \neq l_i\}, b) \\
&\mathcal{O}_{min} := \mathcal{O}_{min} \cup R(\{o_{train} \in \mathcal{O}_{train} : f_{svm_1}(o_{train}) \neq l_o\}, b) \\
&\text{SVM}_2 := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)
\end{aligned} \tag{9-13}$$

With the l_i, l_o the labels for the inlier and outlier class respectively.

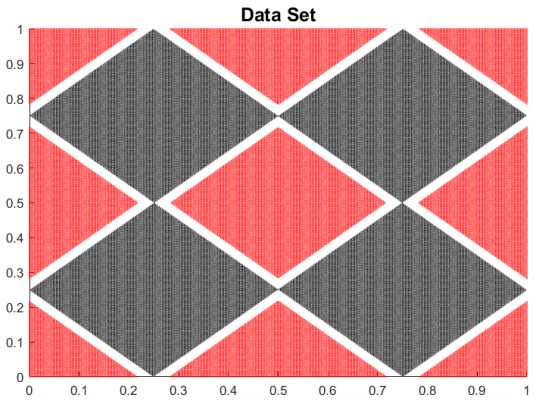
The entire algorithm is as follows:

$\mathcal{I}_{min} := R(\mathcal{I}_{train}, b)$
$\mathcal{O}_{min} := R(\mathcal{O}_{train}, b)$
$\mathcal{I}_{train} := \{i_{train} \in \mathcal{I}_{train} : i_{train} \notin \mathcal{I}_{min}\}$
$\mathcal{O}_{train} := \{o_{train} \in \mathcal{O}_{train} : o_{train} \notin \mathcal{O}_{min}\}$
$\text{SVM}_1 := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)$
for $n = 1 : n_{max}$ **do**
  $\mathcal{I}_{min} := \mathcal{I}_{min} \cup R(\{i_{train} \in \mathcal{I}_{train} : f_{svm_n}(i_{train}) \neq l_i\}, b)$
  $\mathcal{O}_{min} := \mathcal{O}_{min} \cup R(\{o_{train} \in \mathcal{O}_{train} : f_{svm_n}(o_{train}) \neq l_o\}, b)$
  $\mathcal{I}_{train} := \{i_{train} \in \mathcal{I}_{train} : i_{train} \notin \mathcal{I}_{min}\}$
  $\mathcal{O}_{train} := \{o_{train} \in \mathcal{O}_{train} : o_{train} \notin \mathcal{O}_{min}\}$
  $\text{SVM}_{n+1} := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)$
end

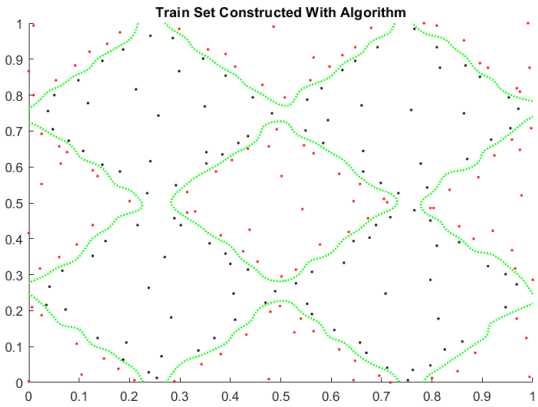
Algorithm 4: Algorithm for selecting a minimal train set for SVM

9-4-1 Demonstration of Sample Strategy

A data set is created for demonstration of the algorithm. Figure 9-2a depicts the complete data set, figure 9-2b depicts the data points that make up the minimal train set.



(a) Two class data set for demonstration purposes



(b) Minimal train set and decision boundary

Figure 9-2: Complete data set (left) and the minimal train set required for accurate classification (right) with decision boundary in green

The minimal train set is depicted with the decision boundary in green. It can be seen that samples close to boundary between classes tend to be picked more frequently. This is to be expected, as this is the region where an incorrectly drawn boundary immediately causes miss classifications. Performance of the algorithm throughout iterations is depicted in below (figure 9-3), compared with a randomly sampled train set of equal size:

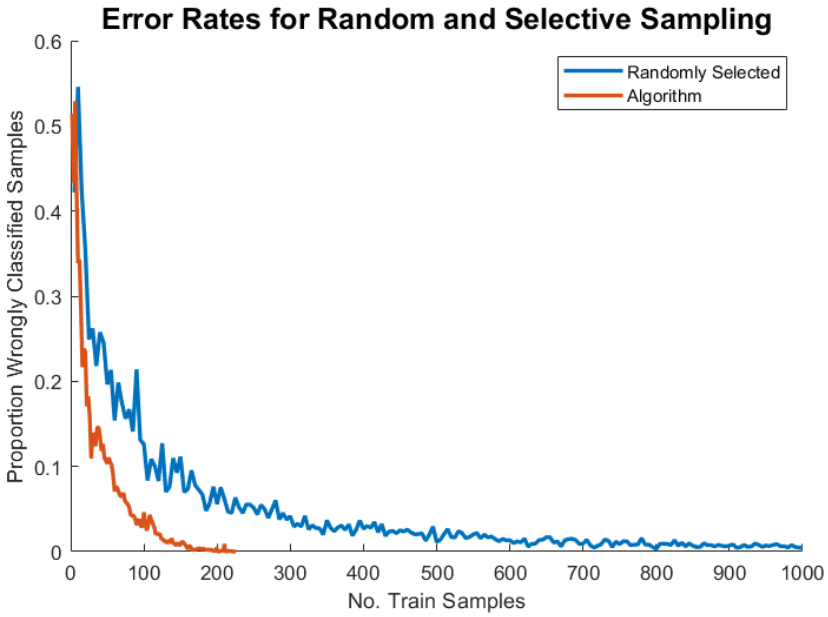


Figure 9-3: The blue line represents error rates for randomly selected train sets, the orange line represents error rates for selective sampling

Both methods increase in accuracy proportionally to the amount of train samples used, evidenced by the gradual decrease in the proportion of wrongly classified samples. The more effective nature of selective sampling can be seen in the faster decrease of the orange line. The algorithm reaches an error rate of zero using 226 train samples.

9-4-2 Further Optimizations

Having explained a simplified version of the algorithm, a more complicated version is presented with some optimizations. Previously an algorithm was shown in which at every iteration all available train data is tested to decide which train samples ought to be included in the minimal train set. The batch size b is the maximum amount of new samples added to the minimal train set every iteration. Especially in earlier iterations, the amount of misclassified train data is much larger than the batch size. A smaller subset of train data can be tested to fill the batch every iteration. Before the loop starts it is measured which fraction of train data is wrongly classified for class $\mathcal{I}$ and $\mathcal{O}$:

$$\begin{aligned} p_i &:= \frac{\text{Card}(\{i_{train} \in \mathcal{I}_{train} : \text{SVM}_n(i_{train}) \neq l_i\})}{\text{Card}(I_{train})} \\ p_o &:= \frac{\text{Card}(\{o_{train} \in \mathcal{O}_{train} : \text{SVM}_n(o_{train}) \neq l_o\})}{\text{Card}(O_{train})} \end{aligned} \tag{9-14}$$

With p_i and p_o the proportion in each class of wrongly classified data. The current classifier would require a subset of train data of cardinality $\frac{b}{p}$ to generate enough wrongly classified

data to fill a batch. However, a larger proportion of data is sampled the next iteration:

$$\begin{aligned} b_i &:= 2\frac{b}{p_i} \\ b_o &:= 2\frac{b}{p_o} \end{aligned} \tag{9-15}$$

With b_i and b_o the batch size for each class. A larger proportion of data is chosen because the next iteration SVM_{n+1} is expected to perform better than the previous SVM_n and therefore miss-classify a lower proportion of data. Both p_i, p_o and b_i, b_o are updated at every iteration. The update procedure for iterative SVMs is given in algorithm 5:

$\mathcal{I}_{candidate} := \emptyset$
$m := 0$
while $Card(\mathcal{I}_{candidate}) < b$ **do**
  $m := m + 1$
  $\mathcal{I}_{candidate} := \mathcal{I}_{candidate} \cup (\{i_{train} \in R(\mathcal{I}_{train}, b_i) : \text{SVM}_n(i_{train}) \neq l_i\})$
end
$p_i := \dfrac{\text{Card}(\mathcal{I}_{candidate})}{m \cdot b_i}$
$b_i := 2\dfrac{b}{p_i}$
$\mathcal{I}_{min} := \mathcal{I}_{min} \cup R(\mathcal{I}_{candidate}, b)$
$\mathcal{I}_{train} := \{i_{train} \in \mathcal{I}_{train} : i_{train} \notin \mathcal{I}_{min}\}$
$\text{SVM}_{n+1} := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)$

Algorithm 5: The update procedure for SVM_n and the minimal train set.

An identical procedure as described algorithm 5 is used for class $\mathcal{O}$. It is enforced that the FAR of the final SVM be under a user set minimum c_{FAR}. A "**while**" loop activates if the FAR of the current iteration's classifier is higher than the user set minimum. FAR is decreased by adding exclusively samples from the set $\mathcal{I}$ to the minimum train set during the "**while**" loop.

Pseudo code of the algorithm as it is used in the thesis is included in algorithm 6:

$\mathcal{I}_{min} := R(\mathcal{I}_{train}, b)$
$\mathcal{O}_{min} := R(\mathcal{O}_{train}, b)$
$\mathcal{I}_{train} := \{i_{train} \in \mathcal{I}_{train} : i_{train} \notin \mathcal{I}_{min}\}$
$\mathcal{O}_{train} := \{o_{train} \in \mathcal{O}_{train} : o_{train} \notin \mathcal{O}_{min}\}$
$\text{SVM}_1 := \text{Train}(\langle \mathcal{I}_{min}, l_i \rangle, \langle \mathcal{O}_{min}, l_o \rangle)$
for $n = 1 : n_{max}$ **do**
    Using only $\mathcal{O}_{train}$, update SVM_n and $\mathcal{O}_{min}$ with algorithm 5
    while $\frac{Card(\{i_{val} \in \mathcal{I}_{val} : SVM_{n+1}(i_{val}) \neq l_i\})}{Card(I_{val})} > c_{FAR}$ **do**
        Using only $\mathcal{I}_{train}$, update SVM_n and $\mathcal{I}_{min}$ with algorithm 5
    end
end

Algorithm 6: Algorithm for selecting a minimal train set for Two Class SVM, adapted for Fault Detection

9-5 Final Remarks on The Proposed Methodology

The successful implementation of RS space classifier is proven by the results of section 9-7. Sections 9-3, 9-4 are summarized below. The complete method for constructing the RS classifier involves the combination of methods discussed in the two previous sections. For visual aid a schematic overview of the procedure is included in figure 9-4:

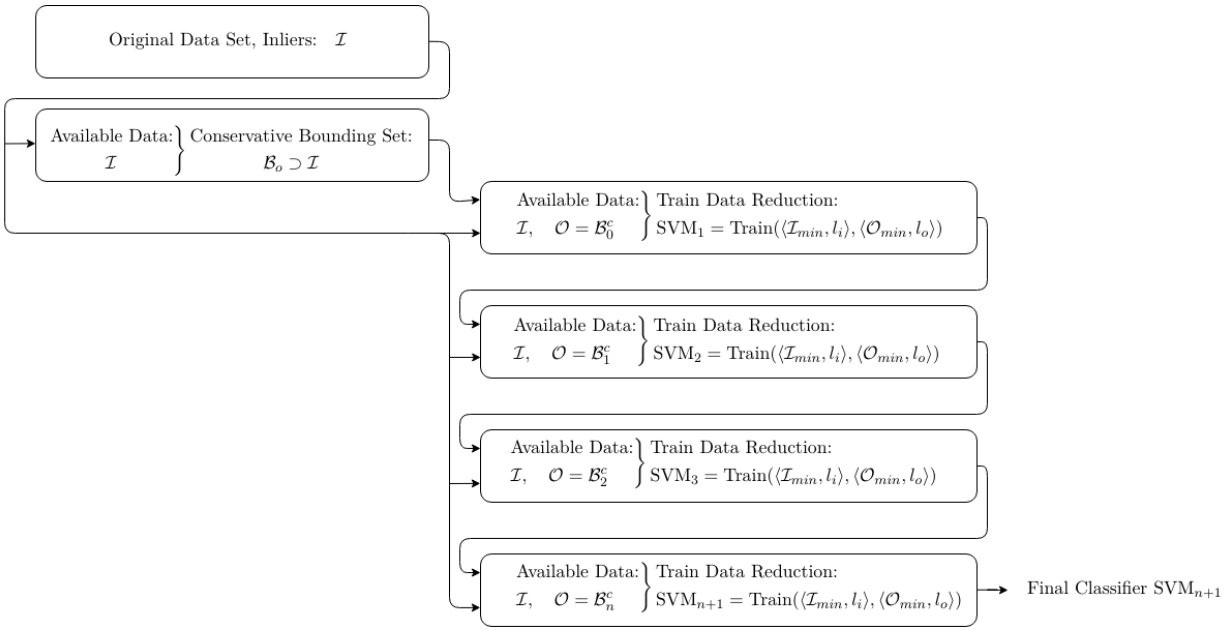


Figure 9-4: Schematic overview of anomaly detection for large data sets using the methods introduced in this chapter

Initially only one labeled class $\mathcal{I}$ is available. Section 9-3 discusses how an unsupervised problem can be made in to an artificial supervised problem by creating bounding sets of decreasing conservativeness. An initial conservative bounding set $\mathcal{B}_0$ is created to start training supervised BSVMs. Section 9-4 describes how each BSVM is trained on a subset of available train data. The BSVM trained in the final iteration is used for RS fault detection.

9-5-1 Comments on Train Data Selection

A significant decrease in required train data is achieved by the data selection algorithm. Additionally, the duration of creating a classifier, i.e. the sum of all train times and test times performed during the algorithm is lower compared to random sub sampling. The efficiency of creating a classifier is greatly increasing due to a large proportion of samples being tested *instead* of trained. The test complexity of a BSVM with kernel functions is $O(n \cdot d)$ [41], the train complexity is $O(n^3)$ [42], with n the amount of train samples and d the dimension of the data. It is for data sets of increasing size exponentially more efficient to test than to train.

A feature unique to model based fault detection is that as much data can be generated as is required. The samples in $\mathcal{I}$ are in the case of RS fault detection elements of $\mathcal{M}_H$ which is generated by a Monte Carlo process. The set of outliers $\mathcal{O}$ is generated by rejection sampling.

In the implementation currently in the thesis, data is generated once and stored in memory. It is the expectation of the author that additional efficiency can be achieved by generating data during iterations rather than retrieving it from memory. As it stands, not enough train data is available to enforce FAR with $\mathcal{I}_{train}$ during later iterations of SVM$_n$, making a minimum FAR requirement the main constraint to further optimization.

9-5-2 Previous Research on Train Data Selection

Previous research exists on selecting from a train set an effective smaller subset for training an accurate SVM. From [45] an overview of available techniques is given in figure 9-5:

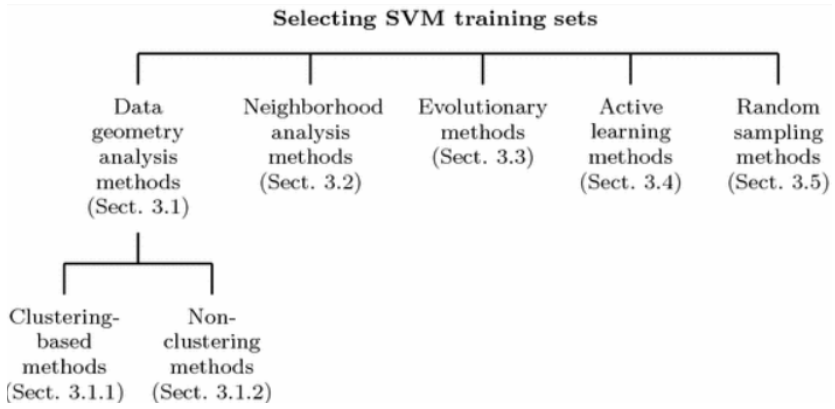


Figure 9-5: Data sub sampling techniques for application in SVM

The method as proposed in the thesis was not found among the literature survey [45]. It is considered out of scope of the thesis to compare the author's method to the methods in the survey, but it should be noted that a difference exists between the aims of the thesis and those in the survey. The methods in the survey mainly focus on decreasing time required to create a classifier. Although to an extent this is achieved by the method presented in this chapter, it is a secondary priority, with the main focus on minimizing test time.

9-6 Ensemble Approach to Minimum Volume

The approach of anomaly detection as a supervised learning problem provides a great improvement in set volume performance. An even greater reduction can be achieved by combining multiple BSVMs in an ensemble (section 9-6-1). To an extent this is a brute force approach to fault detection. In section 9-6-2 a refinement is presented to minimize test time of an ensemble.

9-6-1 Brute Force Ensemble

The gains in set volume Vol($\mathcal{B}$) decrease as meta-iterations progress. A certain minimum is reached which can be surpassed only by increasing the amount of train data allotted to the training of BSVMs. This characteristic can be seen in figure 9-6 where the black line

illustrates the set volume achieved by BSVMs in further iterations. When the individual BSVMs are combined however, a marked improvement is seen in set volume performance.

An important property of each BSVM in the combined approach is that they are forced to be conservative in labeling outliers. Being classified an outlier by the BSVM is therefore a much stronger statement than being labeled an inlier. This quality is used in creating an ensemble approach to set bounding. Multiple BSVMs classify an observation. If at least one BSVM classifies the observation as an outlier, it is classified as an outlier. The bounding set of the ensemble is denoted $\mathcal{B}_E$, and is defined as:

$$\mathcal{B}_E = \mathcal{B}_1 \cap \mathcal{B}_2 \cap \mathcal{B}_{...} \cap \mathcal{B}_n \tag{9-16}$$

Ensemble voting is not uncommon in machine learning. Usually, the average output of multiple models is used as a final output. In the proposed approach the "vote" of one classifier can overrule the votes of all other classifiers making it more analogous to a veto. In the figures below (figures 9-6, 9-7) performances of an ensemble of BSVMs is contrasted with the performance of individual BSVMs.

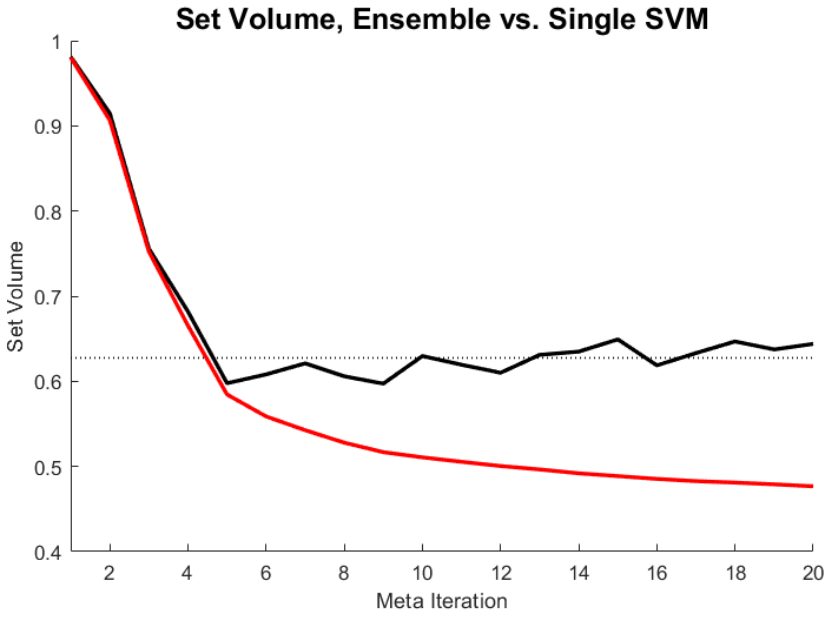


Figure 9-6: Set volume performance of single BSVMs compared with set volume performance of an ensemble of BSVMs. The black line indicates performance of single BSVMs throughout meta iterations, the red line indicates the combined performance of all BSVMs until that meta iteration.

Figure 9-6 is the performance of BSVMs on three tank system data. The black line depicts set volume of BSVMs for consecutive meta iterations. The red line depicts the set volume of an ensemble of BSVMs using all BSVMs created until the current meta iteration. It can be seen that set volume for the individual BSVMs of the meta iterations fluctuates around a constant value after the first five iterations. Yet the volume of the ensemble steadily decreases as more BSVMs are created.

It should be noted that the effectiveness of this approach has its limitations. A theoretical minimum exists on the volume of bounding set $\text{Vol}(\mathcal{B}_E)$. Computation time increases as more

BSVMs are included in the ensemble. Although every individual BSVM has a low FAR, the FAR of the ensemble is approximately the sum of the FAR of each individual BSVM in the ensemble. Figure 9-7 depicts the FAR of each individual BSVM in comparison to the FAR of the ensemble.

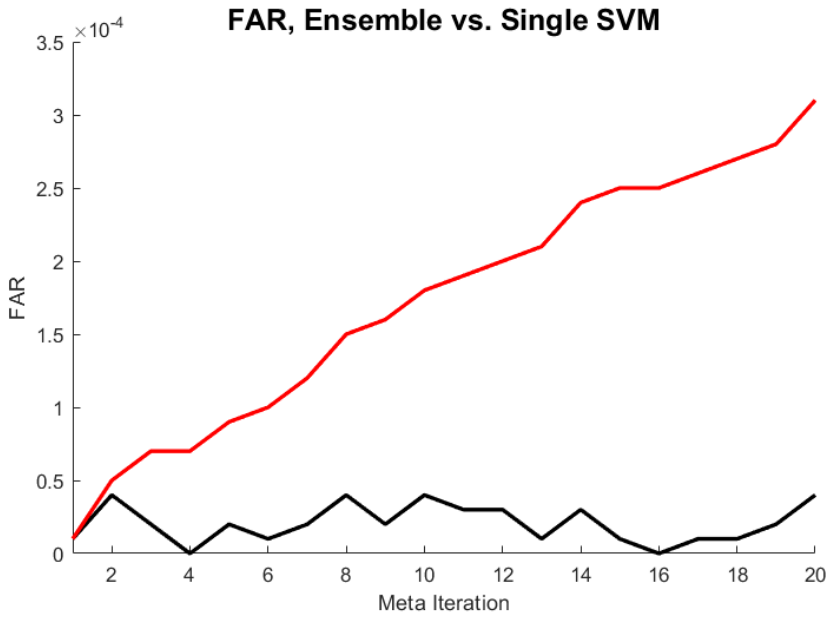


Figure 9-7: FAR performance of single BSVMs compared with FAR performance of an ensemble of BSVMs. The black line indicates performance of single BSVMs throughout meta iterations, the red line indicates the combined performance of all BSVMs until that meta iteration.

It can be reasoned that FAR_E of the ensemble, as a function of its component classifiers is:

$$\min(\text{FAR}_1, \text{FAR}_2, ..., \text{FAR}_N) \leq \text{FAR}_E \leq (\text{FAR}_1 + \text{FAR}_2 + ... + \text{FAR}_N) \tag{9-17}$$

If at least one of the classifiers classifies an inlying sample as outlying, this will be the output of the ensemble, so FAR_E is at least as high as the highest component FAR. However, more than one classifier can mis classify the same inlier, in which case FAR_E is less then the sum of its components.

In the following figure (9-8) the bounding sets achieved by a single BSVM and the ensemble are compared:

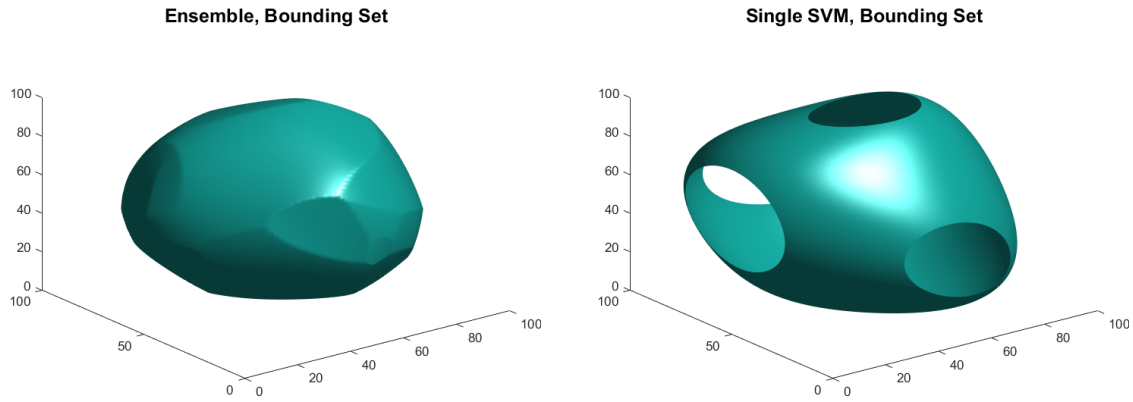


Figure 9-8: Bounding set for single BSVM compared with an ensemble

The bounding sets are visualizations of the bounded healthy residuals for a random initial state in residual space for a three tank system. The ensemble bounding has sharp edges caused by the intersection of multiple bounding sets. The single BSVM bounding set does not entirely fit in the bounding box which causes the "holes" in the visualization.

9-6-2 Faster Classification with a Decision Tree Approach

Combining BSVMs in an ensemble greatly improves accuracy, one of the goals of RS fault detection. Yet the test time is increased, especially for inlying data, i.e. healthy residuals. To see why test time for specifically inlying data is increased, note the ensemble's structure in figure 9-9:

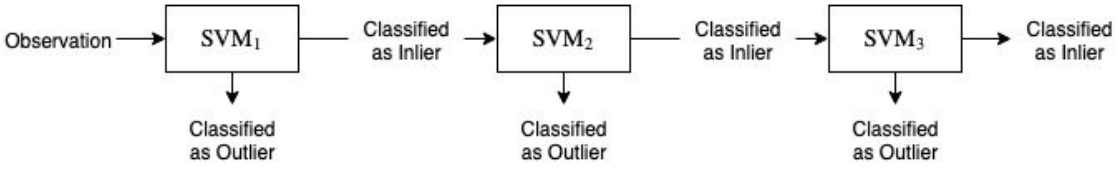


Figure 9-9: The structure of the SVM ensemble, and the steps required for a correct classification of an inlying observation

Classification by the ensemble is complete either if one of the BSVMs classifies an observation as outlying, or all of the BSVMs classify the observation as inlying. In the vast majority of cases an observation is inlying, a circumstance where it is necessary for a single observation to be classified by every single BSVM in the ensemble. Because in application most observations will be inlying the average test time in use will be close to the worst case test time.

A final step in creating the RS classifier is a restructuring of classifiers to increase average test time. Knowing that most observations will by inlying, an additional BSVM is trained which is conservative in classifying inliers; a maximum set volume BSVM with MDR ≈ 0. By initiating the ensemble with this BSVM most inliers will be correctly classified without

having to be tested on every BSVM in the ensemble. This improved structure is depicted in figure 9-10:

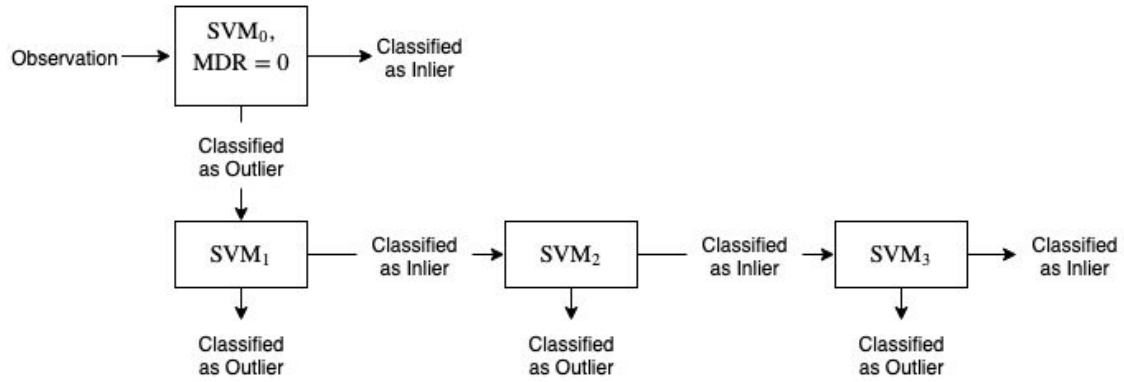


Figure 9-10: Improved ensemble structure for faster test time

In table 9-1 computation times of the brute force and decision tree ensemble are compared on identical data sets:

Table 9-1: Test time for inlying and random data using the brute force ensemble, and the improved decision tree structure

	Brute Force Ensemble	Decision Tree
Classifying Inlying Data	14.177580 sec.	0.959901 sec.
Classifying Random Data	7.967172 sec.	4.432730 sec.

Test data sets healthy samples in residual space for a three tank system, and randomly sampled data in the same dimensions. An identical ensemble of BSVMs is used in both instances. In the decision tree this ensemble is preceded by the zero MDR BSVM. Results conform to expectations. The decision tree is faster in general with the discrepancy especially noticeable on inlying data. FAR and set volume performance is identical.

9-7 Test Results

Three methods are compared on three tank system data: TGR FD-SLSB, TGR FD-KNN and RS FD-BSVMs (using six BSVMs). Because TGR FD and RS FD use data in different dimensions set volume is not a feasible test metric so the methods are compared on simulated fault detection. A total of 500 data sets are created, 250 under faulty system conditions and 250 under healthy system conditions. A fault is introduced as a decrease of 10% in the cross sectional area of the pipe connecting the second tank to the third tank. TGR methods are created using 500 train samples and 500 test samples. Results for MDR, FAR and computation time are included below:

Table 9-2: Comparison of fault detection methods on the FAR, MDR and Computation Time. Included methods are the bench mark (TGR FD-SLSB), the incremental improvement on the bench mark (TGR FD-KNN) and the fundamentally new approach to fault detection (RS FD-BSVM)

	FAR	MDR	Computation Time	Comp. Time, Normalized
TGR FD-SLSB	0.0040	0.8240	214.6230 sec.	9395.2324
TGR FD-KNN	0	0.6680	4.2531 sec.	186.1830
RS FD-BSVM	0	0.6600	0.0228 sec.	1

A universally superior performance by RS FD-BSVM can be concluded. It outperforms TGR FD-KNN mainly on computation time, and outperforms TGR FD-KNN on MDR by a small margin. As was established previously, SLSB is outperformed by KNN. Computation times normalized to the lowest value are included to emphasize the large margin by which computation time has been decreased from the TGR FD-SLSB as introduced in [1].

For both TGR FD methods MDR can be further reduced by using more train data, yet only at the cost of a further increase in computation time. The relation between FAR, MDR and computation time is not this straightforward when using the BSVM ensemble. In addition to train data, ensemble size is a factor. Increasing ensemble size reduces MDR but increases FAR and average computation time. For an equal ensemble size, increasing train sample count decreases MDR without increasing FAR. Although increasing train sample count increases the worst case computation time, the effects on average computation time are more complex. It will be more expensive to test on each individual BSVM in the ensemble, but the probability of exiting the ensemble increases, meaning the additional BSVMs are not activated.

Conclusion and Recommendations

Conclusion

The research objective set out for the thesis was to, with [1] as a benchmark for performance, develop for a FAR performance equal to or smaller than user requirements, a set based fault detection method with a faster Computation Time and a MDR smaller than or equal to the benchmark method. This improvement is achieved by employing machine learning techniques and innovating a new set bounding method which is founded on transforming set bounding from a one class learning problem to a two class learning problem. A new method of data creation is innovated such that one classifier is valid for multiple time steps.

The two class learning problem is solved by combining multiple support vector machines in an SVM-Ensemble. The SVM-Ensemble is structured to classify healthy residuals faster than faulty residuals. A special method of train data selection is developed to select the smallest subset of train data with the maximum effect on classification accuracy. The combination of these methods culminates in a set based fault detection method that performs fault detection at a speed several orders of magnitude that of the bench mark method with an improved false alarm rate and missed detection rate.

When directly comparing computation time of the benchmark method to that of the SVM-Ensemble one thing should be noted. To a degree the SVM-Ensemble moves the required computation time from during the fault detection method being in operation, to before the fault detection method being in operation. To definitively state that the new method is faster, it must be presupposed that the fault detection method is used during at least a long enough time span for the reduced computation time during operation to warrant the much increased computation time before operation.

The SVM-Ensemble method can effectively process a very large amount of train data by creating smaller subsets, but it is not known whether the method only works for a very large amount of train data, which would make difficult the creation of a classifier based on historical data only. Set based fault detection with a minimum volume residual bounding set effectively detects faults outside the set of accepted uncertainties. Yet the method is not sensitive to systematic error, so long as a systematic error is strictly inside the set of accepted

uncertainties. The approach of transforming a one class problem into a two class problem opens up set bounding to a wide array of supervised learning algorithms. Neural networks are an especially promising alternative. Despite good preliminary results, it was not in scope of this thesis to include them.

A separate approach was investigated for set based fault detection without building a single classifier valid for all state-input pairs. Improvements over state of the art fault detection have been achieved by using an adapted K-Nearest-Neighbor algorithm for set bounding and a new sampling technique to reduce the probability of sampling outliers, generate a more uniform data set, and maximize detection for a probabilistic guarantee on false alarm rate. Though an improvement over existing methods, this approach is outperformed by the classifier designed for a range of state-input pairs, on all performance metrics.

Recommendations

Preliminary work has been done using neural networks with promising results, though several potential drawbacks can be identified. An advantage of the SVM-Ensemble is that, when presented with an observation, any member of the ensemble can veto all other members of the ensemble. By placing the ensemble in a sequential structure, it is no longer required to access the remaining ensemble as soon as any SVM has pronounced a veto. This drastically shortens the computation time required to process data with the ensemble. This is not possible using a neural network. Another drawback of neural networks is that they, as opposed to SVMs, do not guarantee separation between classes by the widest possible margin (see section 5-2). The separation of classes by the widest margin is what caused a fast convergence to a minimal volume bounding set when using SVMs. Despite these drawbacks, neural networks may have a faster average test time and perhaps a faster best case test time, and in the author's expectation likely a better MDR performance, and should therefore be investigated as an alternative.

It is to be expected, given the curse of dimensionality, that beyond a certain amount of dimensions the current approach is no longer feasible. This subject has not been relevant for the scale of problems considered in the thesis, but is likely unavoidable for the application of fault detection to increasingly complex systems. It should be investigated what the exact limits are of the current approach in terms of dimensions. Should dimensionality issues limit the use of current methods to practical applications, it is recommended that principal component analysis or self organizing maps be investigated.

Typically system identification methods return fixed parameter models which minimizes mean square error or standard deviation when fitting a model to observation data. For optimal detectability of faults it is preferable to have confidence intervals for parameters. Estimation of parameter intervals for system identification has been studied before [46], but not with the aim of maximizing detectability of faults. An optimization problem should be formulated to return a parameter interval for maximum detectability.

Appendix A

Reducing Train Data for KNN

Test time for KNN can be reduced by decreasing the cardinality of the train set X_{train}. This concept was introduced by Hart [47] for multi class classification. Techniques for anomaly detection mainly consist in de-noising the data set [48].

Using less train data usually deteriorates performance for machine learning methods. In this thesis, a subset $M_{train} \subset X_{train}$ is constructed, for which performance is not worse than for KNN using X_{train}. This is ensured by requiring identical performance of a test set X_{test} on both X_{train} and M_{train}, or in mathematical notation:

$$\begin{aligned} &\underset{M_{train} \subset X_{train}}{\text{minimize}} \ (card(M_{train})), \quad \text{subject to:} \\ &\frac{f_{\text{KNN}}(X_{test}, X_{train}, k) - \tau_n}{|f_{\text{KNN}}(X_{test}, X_{train}, k) - \tau_n|} = \frac{f_{\text{KNN}}(X_{test}, M_{train}, k) - \tau_n}{|f_{\text{KNN}}(X_{test}, M_{train}, k) - \tau_n|} \end{aligned} \tag{A-1}$$

The following algorithm is used to approach minimum cardinality for M_{train}:

(1) $M_{test} := \{X_{test} : f_{\text{KNN}}(X_{test}, X_{train}, k) < \tau_n\}$
(2) $M_{train} := \{X_{train} : (\exists m_{test} \in M_{test})[card(\{X_{train} : ||X_{train} - m_{test}||^2 \leq \tau_n\}) = k]\}$
(3) $M_{test} := \{M_{test} : f_{\text{KNN}}(M_{test}, M_{train}, k) > \tau_n\}$
while $M_{test} \neq \emptyset$ **do**
  (4) $c_k := \underset{0<n_k\leq k, n_k\in\mathbb{Z}}{\text{minimize}}(n_k)$, subject to:
  $\left\{M_{test} : card(\{M_{train} : ||M_{train} - M_{test}||^2 \leq \tau_n\}) = k - n_k\right\} = \emptyset$
  (5) $U_{test} := \left\{M_{test} : card(\{M_{train} : ||M_{train} - M_{test}||^2 \leq \tau_n\}) = k - c_k\right\}$
  (6) $N_{train} := \{X_{train} \cap \bar{M}_{train} : \min card(N_{train}) \wedge$
  $card(\{X_{train} : ||X_{train} - U_{test}||^2 \leq \tau_n\}) = k - c_k + 1\}$
  (7) $M_{train} := M_{train} \cup N_{train}$
  (8) $M_{test} := \{M_{test} : f_{\text{KNN}}(M_{test}, M_{train}, k) > \tau_n\}$
end

The circled numbers are annotations to help explain the functioning of the algorithm. The core of the algorithm are the two sets M_{test} and M_{train}. M_{test} is a set of all samples that are not yet correctly classified by M_{train} using f_{KNN}, as per the requirements in equation A-1. During each iteration of the loop samples from X_{train} are added to M_{train} and correctly classified samples are removed from M_{test}. The iterations end when M_{test} is empty, meaning all test samples are correctly classified using M_{train}.

The loop starts at (1) with removing all anomalies from the test set to create M_{test}. Henceforth, train samples are added to M_{train} until every sample $m_{test} \in M_{test}$ is classified as not anomalous. At (2), the first additions to M_{train} are samples that can't be removed from X_{train} without causing miss classifications in X_{test}. These are test samples m_{test} that have in a radius τ_n exactly k amount of train samples x_{train}. By definition, at least one such sample exists. After any additions to M_{train}, samples that are classified non anomalous with the current set M_{train} are removed from M_{test}. This happens at (3) and (8).

What follows are iterations of steps (4) to (8) until M_{test} is empty. If $c_k = 1$, then at (4) and (5) samples in M_{test} are found which need the fewest amount of train samples added to M_{train} to be non anomalous. Usually $c_k = 1$, however, c_k may also be of higher value to avoid an unsolvable loop. At (6), a set N_{train} of minimum cardinality is constructed under the condition that $M_{train} \cup N_{train}$ increases the amount of train samples in a radius of τ_n around all elements of U_{test} by one. At (7) these samples are added to the existing set M_{train}. As was mentioned before, (8) removes samples classified as non anomalous from M_{test}.

Appendix B

Advanced Methods for Computing Set Volume

An accurate measurement of set volume $\text{Vol}(\mathcal{B})$ is of key importance to optimize performance. A formalization of the problem is given in equation B-1:

$$\begin{aligned} & x \in \mathbb{R}^d \\ & f_I(x) = 1, \quad \forall x \in \mathcal{B} \\ & f_I(x) = 0, \quad \forall x \notin \mathcal{B} \\ & \text{Vol}(\mathcal{B}) = \int_0^1 f_I(x) dx \end{aligned} \tag{B-1}$$

The domain of integration is a unit hyper box (a hyper box with all edges in $[0, 1]$). Consequently the set volume will always be a value in the interval $(0, 1]$. For low dimensional problems $\text{Vol}(\mathcal{B})$ is easily approximated using a static grid, as defined in equation B-2:

$$\text{Vol}(\mathcal{B}) \approx \frac{\text{Card}(\{x_{grid} \in \mathcal{X}_{grid} | f_I(x_{grid}) = 1\})}{\text{Card}(\mathcal{X}_{grid})} \tag{B-2}$$

With $\mathcal{X}_{grid}$ the set of grid points. Because previously all sets have been of relatively low volume, the subject of set volume computation has been of little importance. It is primarily because of the use of RS (see chapter 9) that sets of higher volume are used, and the static grid approach is no longer sufficient. It should be noted that a *relative* rather than an *exact* measurement of set volume is required. Between several classifiers the classifier with lowest set volume is required.

In section 9-1-2 chapter 9 the problem of dimensionality is discussed. In brief summary, as a function of dimension the distance between grid points increases (equation B-3).

$$\Delta = N^{-\frac{1}{d}} \tag{B-3}$$

With dimension d, number of grid points N and inter-point distance Δ. A method is proposed in section B-1-2 to combat the reduced accuracy caused by increasing dimensions. As an introduction to this section numerical integration and adaptive quadrature are briefly discussed.

B-1 Numerical Integration

Computing set volume is a specific application of numerical integration. In general numerical integration is concerned with integrating an arbitrary function over a finite domain. In its most simple form numerical integration is performed as follows:

$$\int_a^b f(x)dx \approx \Delta \sum_{n=0}^{N-1} \frac{f(a+n\Delta) + f(a+(n+1)\Delta)}{2}, \quad \Delta = \frac{b-a}{N} \tag{B-4}$$

With N the amount of intervals chosen by the user. The assumption is that in a small enough interval Δ the function f behaves as a linear function. This approximation is progressively more accurate for smaller values Δ. Equation B-3 and equation B-4 in concert illustrate why numerical integration is heavily affected by the curse of dimensionality.

B-1-1 Adaptive Quadrature

Adaptive quadrature is an algorithm for increasing accuracy of numerical integration without the brute force method of increasing N. Rather, detail is applied locally to regions where the numerical integration is less accurate than the user defined tolerance ϵ. A single iteration of adaptive quadrature is explained in the following:

First, an interval $[a, a+\Delta_a]$ is numerically integrated:

$$Q = \Delta_a \frac{f(a) + f(a+\Delta_a)}{2} \tag{B-5}$$

The same interval is split into two smaller intervals $[a, a+\frac{1}{2}\Delta_a]$ and $[a+\frac{1}{2}\Delta_a, a+\Delta_a]$. The two intervals are integrated:

$$\begin{aligned} Q_1 &= \frac{\Delta_a}{2} \frac{f(a) + f(a+\frac{1}{2}\Delta_a)}{2} \\ Q_2 &= \frac{\Delta_a}{2} \frac{f(a+\frac{1}{2}\Delta_a) + f(\frac{1}{2}\Delta_a+\Delta_a)}{2} \end{aligned} \tag{B-6}$$

The sum of the two smaller intervals provide a more accurate measure of the integral due to their interval being smaller. If the increase in accuracy is minor it is assumed that the step size Δ_a is small enough. This is true if the condition in equation B-7 holds:

$$\text{Accuracy is sufficient if: } |Q - (Q_1 + Q_2)| < \epsilon \tag{B-7}$$

If the condition in equation B-7 is not fulfilled, the integrations Q_1 and Q_2 themselves are split in to two new regions and the procedure is repeated. Integration is complete once all intervals are accurate to the requirement set by the user.

B-1-2 Adaptive Quadrature for Set Volume

An adaptation of adaptive quadrature is presented to measure set volume in a d-dimensional domain. The domain is divided into d-dimensional intervals. Each d-dimensional interval

consists of 2^d points at the vertices of the d-dimensional interval jointly forming the set $\mathcal{I}_\Delta$, with Δ the length of the edges of the interval. It is assumed that if all points in $\mathcal{I}_\Delta$ are in $\mathcal{B}$, the entire interval is in $\mathcal{B}$. Vice versa, if all points in $\mathcal{I}_\Delta$ are out of $\mathcal{B}$, the entire interval is out of $\mathcal{B}$. This stopping criterion is formalized as:

$$\text{Accuracy is sufficient if: } |2^{d-1} - \sum_{n=1}^{2^d} [f(\mathcal{I}_\Delta\{n\})]| = 2^{d-1} \tag{B-8}$$

If either condition is not true, the interval is divided into 2^d number of new intervals with edge length $\Delta_2 = \frac{\Delta}{2}$. The difference between measuring set volume and generic numerical integration is that in generic numerical integration eventually the desired accuracy is reached in all intervals. However, when measuring set volume there remains a region at the boundary of $\mathcal{B}$ where part of the interval is in the set and part of the interval is out of the set regardless of how small Δ is. To prevent an exponential increase in the amount of grid points, a tolerance is introduced:

$$\text{Accuracy is sufficient if: } |2^{d-1} - \sum_{n=1}^{2^d} [f(\mathcal{I}_\Delta\{n\})]| = 2^{d-1} - \epsilon \tag{B-9}$$

The tolerance $\epsilon \in \mathbb{Z}$ causes an interval with all *but* ϵ amount of samples in or out of the set to be considered completely in or out of the set. The tolerance is activated whenever the total amount of grid points exceeds the user set maximum.

B-1-3 Monte Carlo Integration

A comparatively simpler approach to set integration is Monte Carlo integration. The procedure in its entirety is depicted below (algorithm 7) and explained in the following.

$e_{1,2,\dots,M} := \dfrac{\text{Card}(\{x_{rand} \in \mathcal{X}_{rand} | f(x_{rand}) = 1\})}{\text{Card}(\mathcal{X}_{rand})}$
$\mu_e := \frac{1}{M} \sum_{m=1}^{M} e_m$
$D_e := \frac{1}{\mu_e} \sum_{m=1}^{M} (\mu_e - e_m)^2$
$i := 1$
while $D_e > \epsilon$ **do**
  $e_{1,2,\dots,M} := \dfrac{i}{i+1} e_{1,2,\dots,M} + \dfrac{1}{i+1} \dfrac{\text{Card}(\{x_{rand} \in \mathcal{X}_{rand} | f(x_{rand}) = 1\})}{\text{Card}(\mathcal{X}_{rand})}$
  $\mu_e := \frac{1}{M} \sum_{m=1}^{M} e_m$
  $D_e := \frac{1}{\mu_e} \sum_{m=1}^{M} (\mu_e - e_m)^2$
  $i := i + 1$
end
$\text{Vol}(\mathcal{B}) \approx \mu_e$

Algorithm 7: Integrating a set with Monte Carlo integration

An estimate e_m of random set $\text{Vol}(\mathcal{B})$ is made with a random set $\mathcal{X}_{rand}$. In total M different uncorrelated estimates $e_{1,2,\dots,M}$ are made. D_e is the index of dispersion; a normalized measure of variance between all estimates of the set volume. The set volume estimate is

considered accurate if $D_e < \epsilon$, with ϵ set by the user. While the accuracy of the estimate is not high enough, all estimates are refined with additional samples. Advantages of Monte Carlo integration will be discussed in closer detail in section B-2.

B-2 Comparing Adaptive Quadrature and Monte Carlo Integration

A test case is constructed for both methods of set volume computation. The set $\mathcal{B}$ is a hypersphere with radius $r = \frac{1}{2}$. Performances of static grid, adaptive quadrature and Monte Carlo are compared. Due to differences in scale the static grid and adaptive quadrature performance are depicted in figure B-1 without the performance of Monte Carlo integration:

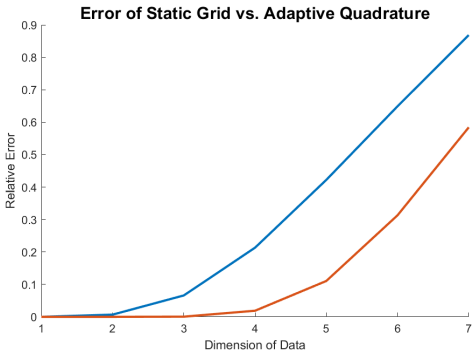
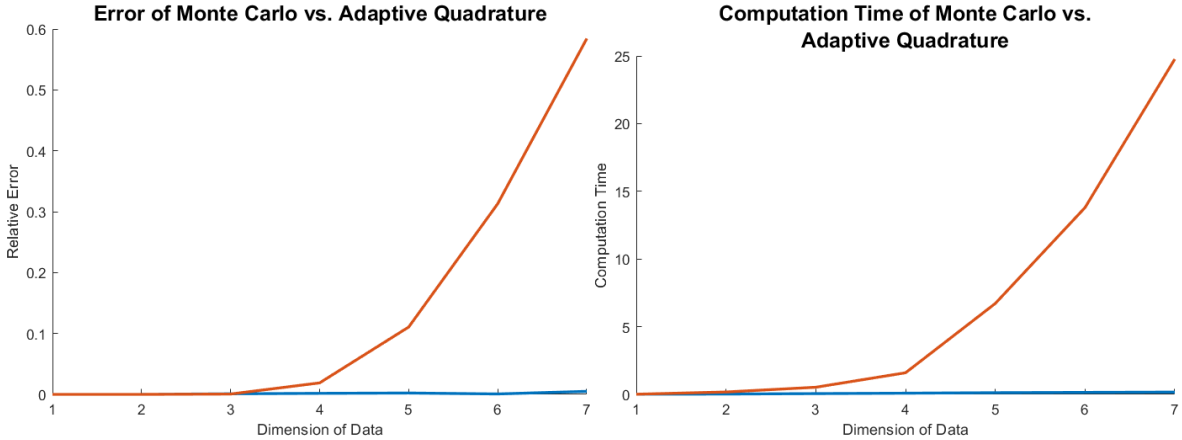


Figure B-1: Set volume error, static grid (blue) vs. adaptive quadrature (orange)

Both methods are afforded the same maximum grid size. It can be seen that using adaptive quadrature a higher accuracy is achieved. For increasing dimensions adaptive quadrature, suffers from the curse of dimensionality as well. In figure B-2a performances of Monte Carlo and adaptive quadrature are depicted together.



(a) Relative error Monte Carlo integration (blue) vs. adaptive quadrature (orange)

(b) Computation time of Monte Carlo integration (blue) vs. adaptive quadrature (orange)

Figure B-2: Comparison of set volume measurement using Monte Carlo and adaptive quadrature on accuracy (left) and computation time (right)

Monte Carlo integration outperforms adaptive quadrature on both accuracy and computation time by increasingly large margins for ascending dimensions. It is perhaps not surprising that Monte Carlo integration is more accurate, as a minimum accuracy is a stopping criterion for the process.

What merits closer attention is the difference in computation time. Adaptive quadrature is designed to eliminate a growing space from further consideration. New iteration points are created such that the most information can be expected to be retrieved from them. In contrast, the Monte Carlo integration places samples completely randomly at every iteration in a much more brute force approach.

The pitfall of adaptive quadrature is that it requires a number of complex operations on large arrays and it requires that large arrays be saved to and retrieved from memory every iteration. The Monte Carlo integration requires neither. Additionally, Monte Carlo integration can be performed with parallel computing.

B-3 Conclusion

Set volume computation becomes highly inaccurate for higher dimensions if a grid is used. Although better results are achieved with adaptive quadrature it too suffers poor performance for increasing dimensions. This phenomena is in the view of the author an inescapable problem for any numerical integration method that in any way is based on a grid. It can be seen that Monte Carlo integration is a highly efficient and accurate manner of set volume computation.

The user controls the accuracy of the Monte Carlo integration. This is an asset when a relative measure of set volume is required, i.e. which classifier has the smallest set volume rather than what exactly the set volume is. A suitable stopping criterion when comparing two classifiers is:

$$\epsilon < |\text{Vol}(\mathcal{B}_1) - \text{Vol}(\mathcal{B}_2)| \tag{B-10}$$

It strongly recommended that Monte Carlo integration be applied for fault detection applications in high dimensions.

Bibliography

[1] V. Rostampour, R. Ferrari, and T. Keviczky, "A set based probabilistic approach to threshold design for optimal fault detection," *Proceedings of the American Control Conference*, pp. 5422–5429, 2017.

[2] R. Martinez-Guerra and J. L. Mata-Machuca, "Understanding Complex Systems Fault Detection and Diagnosis in Nonlinear Systems A Diff erential and Algebraic Viewpoint,"

[3] S. X. Ding, "Model-based fault diagnosis techniques: design schemes, algorithms, and tools," 2008.

[4] J. Cusido, L. Romeral, J. A. Ortega, J. A. Rosero, and A. Garcia Espinosa, "Fault Detection in Induction Machines Using Power Spectral Density in Wavelet Decomposition," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 2, pp. 633–643, 2008.

[5] J. Dikun, L. Urmoniene, and D. Stanelyte, "Spectral Ratio Method for Fault Detection in Rotating Machines," *Balkan Journal of Electrical and Computer Engineering*, pp. 61–63, apr 2018.

[6] X. Li and K. Zhou, "A time domain approach to robust fault detection of linear time-varying systems," *Automatica*, vol. 45, pp. 94–102, jan 2009.

[7] R. Isermann, "Fault diagnosis of machines via parameter estimation and knowledge processing—Tutorial paper," *Automatica*, vol. 29, pp. 815–835, jul 1993.

[8] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

[9] N. Saravanan, V. N. S. K. Siddabattuni, and K. I. Ramachandran, "Fault diagnosis of spur bevel gear box using artificial neural network (ANN), and proximal support vector machine (PSVM),"

[10] M. Zareapoor and P. Shamsolmoali, "Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier," *Procedia Computer Science*, vol. 48, pp. 679–685, 2015.

[11] J. Xiong, "Set-membership state estimation and application on fault detection," 2013.

[12] F. Dabbene and D. Henrion, "Set approximation via minimum-volume polynomial sublevel sets," *ieeexplore.ieee.org*, no. European Control Conference (ECC), 2013.

[13] E. Alpaydin, *Introduction to machine learning*. 2010.

[14] H. A. Dau, V. Ciesielski, and A. Song, "Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class," pp. 311–322, 2014.

[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing Surveys*, 2009.

[16] J. Janssens, "Outlier selection and one-class classification," tech. rep.

[17] S. Khan and M. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374.

[18] R. G. Brereton, "One-class classifiers," *Journal of Chemometrics*, vol. 25, no. 5, pp. 225–246, 2011.

[19] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques in financial domain," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[20] C. Aggarwal and P. Yu, "Outlier detection for high-dimensional data," *ACM Sigmod Record*, vol. 30, no. 2, pp. 37–46, 2001.

[21] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining," *Applied Artificial Intelligence*, vol. 17, pp. 375–381, may 2003.

[22] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," 2003.

[23] C. Cortes, V. Vapnik, and L. Saitta, "Support-Vector Networks Editor," tech. rep., 1995.

[24] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," tech. rep.

[25] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Advances In Neural Information Processing Systems 12*, vol. 12, pp. 582–588, 2000.

[26] D. M. Tax and R. P. Duin, "Support Vector Data Description," *Machine Learning*, vol. 54, pp. 45–66, jan 2004.

[27] B. W. Silverman, *Density estimation for statistics and data analysis*.

[28] R. Duin, D. de Ridder, and D. Tax, "Featureless Pattern Classification," *Kybernetika*, vol. 34, pp. 399–404, 1998.

[29] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, pp. 170–231, jun 1998.

[30] G. Chen and S. N. Srihari, "Removing Structural Noise in Handwriting Images using Deep Learning," 2014.

[31] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal Thresholds for Anomaly-Based Intrusion Detection in Dynamical Environments," tech. rep.

[32] P. M. Martínez, "Hyperparameter optimization of svm stochastic processes," 2017.

[33] J. Kiefer, "Sequential Minimax Search for a Maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, p. 502, 2006.

[34] A. Karatzoglou, D. Meyer, W. Wien, and K. Hornik, "Journal of Statistical Software Support Vector Machines in R," tech. rep., 2006.

[35] D. W. Scott, "Scott's rule," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, pp. 497–502, jul 2010.

[36] J. Pagter and T. Rauhe, "Basic Research in Computer Science," 1998.

[37] W. A. Wulf and S. A. Mckee, "Hitting the Memory Wall: Implications of the Obvious," tech. rep., 1994.

[38] K. Fukuda, "Interior-Point Methods for Linear Programming," tech. rep.

[39] D. Harvey, J. Van Der Hoeven, and G. Lecerf, "Even faster integer multiplication," tech. rep., 2014.

[40] I. Wald and V. Havran, "On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N)," tech. rep., 2006.

[41] M. Claesen, F. De Smet, J. A. K. Suykens, and B. De Moor, "Fast Prediction with SVM Models Containing RBF Kernels *," tech. rep., 2014.

[42] A. Bordes and J. Weston, "Fast Kernel Classifiers with Online and Active Learning," tech. rep., 2005.

[43] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working Set Selection Using Second Order Information for Training Support Vector Machines," tech. rep., 2005.

[44] D. François, V. Wertz, M. Verleysen, and S. Member, "The Concentration of Fractional Distances,"

[45] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: a review," *Artificial Intelligence Review*, pp. 1–44, jan 2018.

[46] M. C. Campi, G. Calafiore, and S. Garatti, "Interval predictor models: Identification and reliability," *Automatica*, vol. 45, pp. 382–392, 2009.

[47] P. E. Hart, "The Condensed Nearest Neighbor Rule," *IEEE*, 1967.

[48] H. Liu and S. Zhang, "The Journal of Systems and Software Noisy data elimination using mutual k-nearest neighbor for classification mining," *The Journal of Systems and Software*, vol. 85, pp. 1067–1074, 2011.