CHALMERS

Anomaly Detection in PowerCells Auxiliary Power Unit

Master's Thesis in Computer Science and Engineering

HAMPUS HJORTBERG

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Anomaly Detection in PowerCells Auxiliary Power Unit

Hampus Hjortberg

Anomaly Detection in PowerCells Auxiliary Power Unit

HAMPUS G. HJORTBERG

Examiner: DEVDATT DUBHASHI

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

In the paper of Hayton et.al [1], *One-class Support Vector Machine* is used for health monitoring of a jet engine in order to discover when and if an abnormal event has occured. Hayton et.al used the amplitude of the vibration data from the engine shaft as the feature data to the *One-class Support Vector Machine* algorithm. This approach works well when the sensor data is known to be periodic, with a certain frequency; however it can not be used if the sensor data has an irregular shape. In this paper we will extend the concept of Hayton et.al [1] and use the *Discrete Wavelet Transform* coefficients as input data to the *OCSVM*, rather than the *Fourier Transform*. This way we are able to classify more arbitrary sensor data found in PowerCells Auxilliary Power Unit (APU). We will also introduce a novel approach of how to select the hyperparameter σ for the Radial Basis Function Kernel, in order to avoid both overfitting and underfitting.

Contents

1 Background

Anomaly detection is a field within machine learning where the goal is to identify if any data in the dataset is unexpected given the history of previous data. Unexpected events are important to detect because they could be an indication that something is wrong. During recent years there has been an immense increase of interest within this field both in industry and in the accademic society. The enhanced computation and storage capabilities has elevated the applicability for all machine learning techniques. Anomaly detection is not a specific method or algorithm but rather a collection of them, each with different pros and cons. The underlying assumption is that only data of normality is provided to the algorithm of choice. From this data, the task will be to find a subspace where data is classified as normal and a complementary subspace where it is classified as abnormal. Given this proceducer, anomaly detection is often regared as a semi-supervised learning algorithm. The reason why it is not entirely unsupervised is because you are bound to check that no abnormal data is in the training set.

In this report we will focus on one perticular method called *One-class Support Vector Machine (OCSVM)*. It borrows a lot of the same concepts from the widely known *Support Vector Machine* classification algorithm. The *OCSVM*, has been used frequently both in industry and in the academic society during recent years, for the purpose of novelty detection.

In the paper of Hayton et.al [1], *OCSVM* is used for health monitoring of a jet engine in order to discover when and if an abnormal event has occured. Hayton et.al used the amplitude of the vibration data from the engine shaft as the feature data to the *OCSVM* algorithm. This approach works well when the sensor data is known to be periodic, with a certain frequency; however it can not be used if the sensor data has an irregular shape. In this paper we will extend the concept of Hayton et.al [1] to find a method that could be used for health monitoring of every type of sensor data; not only the ones who are periodic. In order to optain amplitudes of a vibration you are bound to make a fourier transform of the raw data. But rather than using a *Fourier Transform* to get the amplitudes, which are used as feature data, we will we use a wavelet transform where properties of the wavelet coefficients can be used as input feature data to the classifying algorithm.

This paper is made in colaboration with a company, called *PowerCell Sweden AB*. There mission is to design an Auxiliary Power Unit (APU) that can convert low sulfur automotive diesel into electricity without the use of an Internal Combustion Engine (ICE) generator. Instead of using combustion, they use a series of chemical catalytical processes to reform hydrogen gas from the low sulfur automotive diesel. Subsequently, the hydrogen gas is then used as an input fuel to a Proton Exchange Membrane (PEM) fuel cell that produces electricity. The major advantages with a system like this compared to a regular diesel driven Internal Combustion Engine (ICE) generator, is the gained efficiency in fuel consumption and lowered emissions.

The traditional ICE has been developed and optimized over a century. It is well established how the ICE works and operate eventhough the chemical processes in the cumbustion is not jet explored. The difference between an ICE and PowerCells APU, is that an ICE works fine without knowing what comes out of the exhaust. The approach that PowerCell has is different. The chemical processes are at the basis of the opertation and if some unexpected process

occurs, the hole unit could possibly malfunction. The chemistry in the APU is complex and sometimes difficult to model; using machine learning and novelty detection techniques one can deal with complexity issue. It will not tell you the reason why something has occurd but it can tell you that something has occurd. If something unexpected occurs there is a high chance that this will lead to a degredation of the entire unit; some of the components could be damaged when exposed to impurities. The components that PowerCell uses are expensive and non dispensable; for that reason it's more important to remain the integrity of the components rather than to continue with operation. PowerCell is in need of a method to monitor the system and detect when any unforeseen event is happening, that might harm the system. It's not feasible to have human monotoring due to the shear amount of data. Therefore it will be nessesary to implement machine supervision. In order to have a machine that supervise the process we will need to learn the machine 'what is a state of normality' to be able to find states of abnormality. The colaboration with PowerCell also involved the design of a framework, to detect novel events, that could be fully integrated in their sensor data collection system.

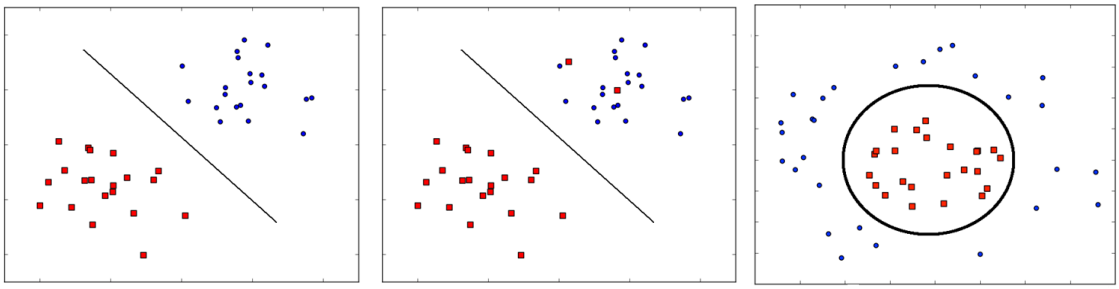
The *Discrete Wavelet Transform (DWT)* has many applications in the field signal processing. It can be used to compress or denoise a signal but it has also been used in combination with the *OCSVM* for anomaly detection in Wireless Sensor Networks [2]. The idea is to: select an interval of the time signal, perform *DWT* decomposition, extract the important properties of the wavelet coefficients and lastly feed those important features into the *OCSVM*. The reason why this could be advantageous is because the wavelet coefficients contains information about the overall behaviour of the signal on differnet time scales.

In this report we will present a novel approach on how to combine *OCSVM* with *DWT* in order to create a tool for anomaly detection in a discrete time signal data. The basic linear implementation of *OCSVM* will optimaly seperate a fraction of the data with a hyperplane; it is only when kernels are introduce that a nonlinear decision boundary could be obtained. More precisely we will use the *Radial Basis Function* (RBF) kernel. The drawback with a kernel is that extra hyperparameters are introduced; in perticular the RBF-kernel has one hyperparameter σ that needs to be determined. There is little theory of how to set these hyperparameters but normaly they are decided through cross validation. In this report we will also present a novel approach on how to set the hyperparameter σ in order to avoid overfitting and underfitting of the *OCSVM* algorithm.

2 Theory

2.1 Support Vector Machine

The Support Vector Machine, in its origianl form, is a binary classifier that is able to seperate two clases of data in a high dimensional space. The SVM algorithm is all about constructing a hyperplane that distinguish two differnt clases of from each other. The principal of how this works can be seen in figure 2.1 a), where we illustrate the results from 2-dimensional linear test case. The boundary decision line has a maximum margin to each of the two classes. It is easy to see how this works in the simple case of figure 2.1 a) but the SVM machinery will also be handle the case when is not possible to seperate the the data points with a hyperplane, as in figure 2.1 b). With the use of the kernel trick it can even deal with nonlinear decision boundaries, as in figure 2.1 c). In section 2.1.1 we will discribe the mathematics of how this is done.



(a) linear SVM (b) linear SVM with outliars (c) non-linear SVM

Figure 2.1: *Illustration of how the SVM is capable of classifying two classes, in three different scenarios.*

The Support Vector Machine for novelty detection is slightly different from the ordinary binary classification that the regular *SVM* algorithm deals with. When working with novelty or anomaly detection there will only be positive examples of what is 'normal'; therefore we will only feed one class of datapoints to the algorithm. The algorithm that handles this type of problem is called the *One Class Support Vector Machine* (OCSVM). Rather than finding a hyperplane that optimaly seperates two classes, it finds a hypersphere that optimaly engulfs a fraction ν of the datapoints, from the rest of the 'abnormal' datapoints. If that fraction ν is chosen to be small one will get an anomaly detection algorithm.

2.1.1 One-Class Support Vector Machine

We begin to define our training set where we denote the number of training data as $l \in \mathbb{N}$. The data points x_i are multidimensional vectors that belongs to the input feature space χ

$$x_1, ..., x_l \in \chi \tag{2.1}$$

The true power of SVM only becomes apparent when kernels are introduced. Let Φ be denoted as a maping function such that $\chi \to F$, where F is a inner product space and the

image of Φ can be calculated using a kernel function [4]

$$k(x,y) = (\Phi(x) \cdot \Phi(y)). \tag{2.2}$$

The trick of any *SVM* algorithm is that they never explicitly require to merely perform the mapping $\Phi(x_i)$; it is only the inner products $\Phi(x) \cdot \Phi(y)$ that are essential to calculate. Thus replacing the innerproduct with a kernel function, that fulfills certain requirements, will map χ to an unknown inner product space F; this approach is called the kernel trick. F is unknown because the maping function Φ is unknown but for some kernels it is still possible to derive the properties of F.

One of those is the *Radial Basisi Function (RBF)* kernel,

$$k(x,y) = e^{-\|x-y\|^2/\sigma} \tag{2.3}$$

which is one of the more frequently used kernels. It can be shown that the transform function $\Phi(x)$ maps the feature data in χ into $F \in R^{\infty}$. If the feature data is maped into an infinte dimonsional space F, the *OCSVM* algorithm can always find a seperation of the two classes [5]; however in the feature space χ the hypersphere will be a nonlinear decision boundary. In practice the *OCSVM* in combination with the RBF-kernel becomes a method of finding high density [1], for that reason it is sometimes regared as a unsupervised learning algorithm [5].

The primal form of the *OCSVM* is

$$\begin{aligned} \min_{\omega \in F, \xi \in R^l, \rho \in R} \quad & \frac{1}{2}\|\omega\|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \\ \text{subject to} \quad & (\omega \cdot \Phi(x_i)) \geq \rho - \xi_i,\ \xi_i \geq 0 \end{aligned} \tag{2.4}$$

This quadratic program (QP) minimization problem will seperate the data from the origin. If the problem (2.4) is not possbile to define, beacause of outliars, one needs to introduce soft margins. This is done with the use of slack variables ξ_i. Note that these slack variables ξ are penalized in the objective function. The trade off between how much $\|\omega\|^2$ should be minimized in relation to slack variables $\sum_i \xi_i$ is regulated by the parameter ν. The decision function

$$f(x) = sgn(\omega \cdot \Phi(x) - \rho) \tag{2.5}$$

will be 1 if x is classified as normal and -1 if its abnormal. Note that, in the primal form we are forced to calculate $\phi(x)$, which is not possible if we use the *RBF*-kernel. Therefore we will derive the dual form of the *OCSVM*. A common aproach when dealing with constrained optimization problem is to introduce the *Lagrangian function*

$$L(\omega, \xi, \rho, \alpha, \beta) = \frac{1}{2}\|\omega\|^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho - \sum_i \alpha_i(\omega \cdot \Phi(x_i) - \rho + \xi_i) - \sum_i \beta_i \xi_i \tag{2.6}$$

The *Lagrangian multipliers*, $\alpha_i, \beta_i \geq 0$ will penalize the *Lagrangian*, L, if the constraints are violated. The objective is now to find the minimum of the *Lagrangian*, L. In order to find the minimum of $L(\omega, \xi, \rho, \alpha, \beta)$, all the partial derivatives with respect to the primal variables $\omega,\ \xi,\ \rho$ will need to be zero.

$$\frac{\partial L}{\partial \omega} = 0 \quad \Rightarrow \quad \omega = \sum_i \alpha_i \Phi(x_i) \tag{2.7}$$

$$\frac{\partial L}{\partial \xi} = 0 \quad \Rightarrow \quad \alpha_i = \frac{1}{\nu l} - \beta_i \leq \frac{1}{\nu l} \tag{2.8}$$

$$\frac{\partial L}{\partial \rho} = 0 \quad \Rightarrow \quad \sum_i \alpha_i = 1 \tag{2.9}$$

The substitution of (2.7) into (2.5) yeilds

$$f(x) = sgn\left(\sum_i \alpha_i k(x_i, x) - \rho\right) \tag{2.10}$$

The substitution of (2.7), (2.8),(2.9) into (2.5) gives the dual form of the *OCSVM*.

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l}, \sum_i \alpha_i = 1 \end{aligned} \tag{2.11}$$

According to Scholkopf et al. one can show that at the optimum, the two inequality constraints (2.11) become equalities if α_i and β_i are nonzero, i.e. if $0 < \alpha_i < 1/(\nu l)$. Therefore, we can recover ρ by exploiting that for any such α_i , the corresponding data point x_i satisfies

$$\rho = (\omega \cdot \Phi_i) = \sum \alpha_j k(x_j, x_i) \tag{2.12}$$

2.1.2 Sequential Minimum Optimazation

At the basis of any type of *SVM* algorithm is a quadratic optimization problem. Normaly these are solved using an of the shelf quadratic programming algorithm. Studies have shown that these algorithms performs very porly when they are applied to a *SVM* optimization problem, [5]. The main reason is because the kernel matrix, $K_{ij} = \Phi(x_i) \cdot \Phi(x_j)$, becomes to large to be stored in the cache memory. Basically the kernel matrix will need to be recomputed or obtained in each iteration when quadratic programming is used. Scholkopf et al. solved this issue by contructing an algorithm called *Sequential Minimum Optimazation* (SMO).

The novel approach of SMO, rather than standard quadratic programming , is to break up the constrained optimization over all Lagrangian multipliers into the smallest optimization steps possible. The trick is to select two Lagrangian multipliers, i.e. α_1, α_2 and regard the rest of the multipliars as constant. The constraind minimization (2.11) can now be rewritten as

$$\min_{\alpha_1, \alpha_2 \in R} \frac{1}{\nu l} \sum_{i,j=1}^{2} \alpha_i \alpha_j K_{ij} + \sum_{i=1}^{2} \alpha_i C_i + C \tag{2.13}$$

Scholkopf et al. makes a derivation of how to explicitly find the optimum of the reduced problem (2.13), given the equality constriant. The final result is

$$\begin{aligned} \alpha_1 &= \Delta - \alpha_2 \\ \alpha_2 &= \frac{\Delta(K_{11} - K_{12}) + C_1 - C_2}{K_{11} + K_{22} - 2K_{12}} \end{aligned} \tag{2.14}$$

Scholkopf et al. also describes a hierarchy of how to select α_1 and α_2 from the entire set of *Lagrangian multipliers* α_i. In generall, a multiplier is more likely to be selected if its corresponding data point is a support vector. For this reason the *SMO* algorithm requires only a smaller part of the kernel matrix, K_{ij}, to be stored in the cache memory, which was the main issue with the standard QP-solver.

2.2 Wavelets

Wavelets have many application within the field of signal processing. It can be used for denoising, compression and signal analysis. Especially the signal analysis property is something that will be used in this report. The applications and use of the Wavelet transform are in many ways similar to the *Fourier Transform*. The *Wavelet Transform* is however superior to the *Fourier Transform*, for signal analysis purposes, when the signal has an irregular shape. If the signal is non periodic and irregular, the *Fourier Transform* could still be used for reconstruction purposes but the fourier coefficients will not be particularly useful for any type of signal analysis. The amplitude of the fourier coefficents gives information about which frequencies that can be found in the signal. Eventhough this could be useful information when the signal is periodic, it is not useful when the signal is irregular. The reason is because the phase of the coefficients are random, and its the phase that contains the information about of how too reconstruct the signal. For that reason the fourier coefficents can not be trusted for signal analysis on an irregular signal. The Wavelet coeffecients on the other hand carries information about the properties of even irregular signals. Another useful feature of the Wavlet coefficients is its localization property; this implies that a coefficient gives information about how the signal behaves at a specific place and time scale.

Time series signals are discrete and for that reason the *Discrete Wavelet Transform (DWT)* will be used. Much like the *Fast Fourier Transform (FFT)* it is the discrete representation of its continous counterpart. A signal with 2^n sampled data points will be possible to decompose to a level $N \leq N_{max} \leq n$. N_{max} depends on n and which type of wavelet that is used. The *DWT*-spectra consist of $N+1$ different parts arranged as follows $\{A_N, C_N, C_{N-1}, ..., C_2, C_1\}$. A_N has 2^{n-N} coefficients and together they discribe to overall flux of the signal, see figure 2.2 b). C_{N-L}, $L = \{0, 1, 2, ..., N-1\}$ has $2^{n-(N-L)}$ coefficients and they describe the behaviour of the signal on time scale that characteristic for the level $N-L$, see figure 2.4. [6]

Figure 2.2, 2.3 and 2.4 illustrates how the wavelet decompostion works in practice. The signal is composed by adding gausian noise to a fluctuating signal. On top of that a deviation was added to illustare an anomaly. Even if the anomaly is not greater in amplitude than the fluctuations and the noise, it is still very much noticeable on the wavelet coefficients, see figure 2.4.

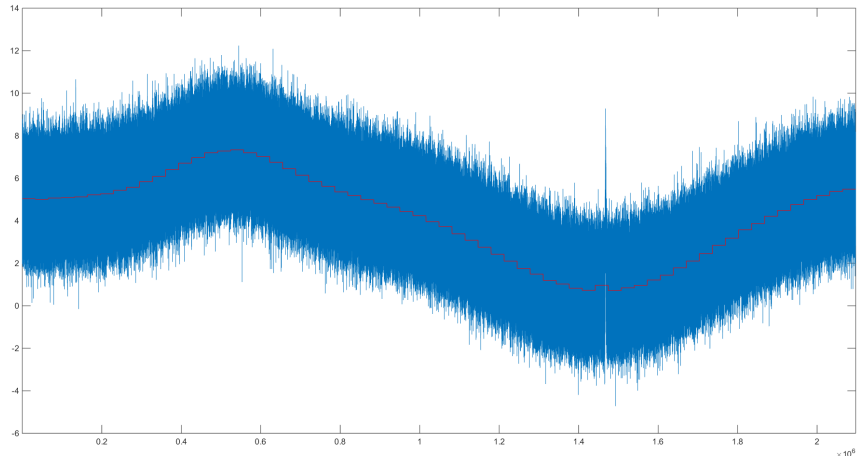


Figure 2.2: *A test signal signal composed of: two wide peaks around 0.5 and 1.5, one sinus wave, white noise and one narrow peak at 1.5 representing an anomaly.*

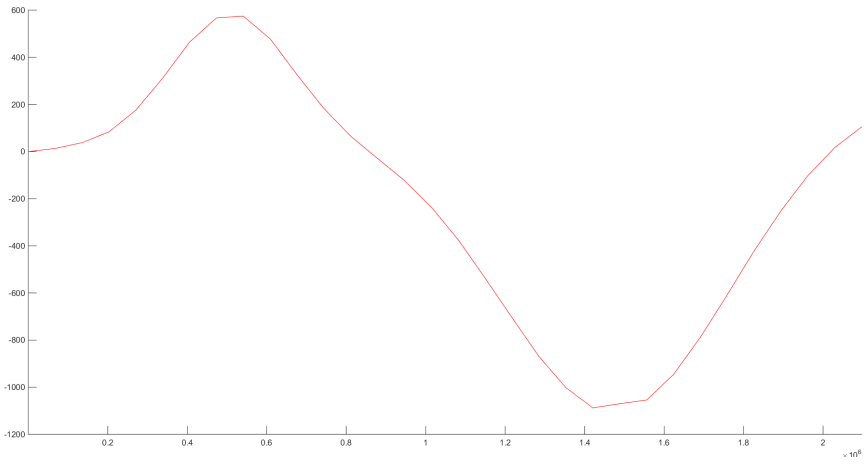
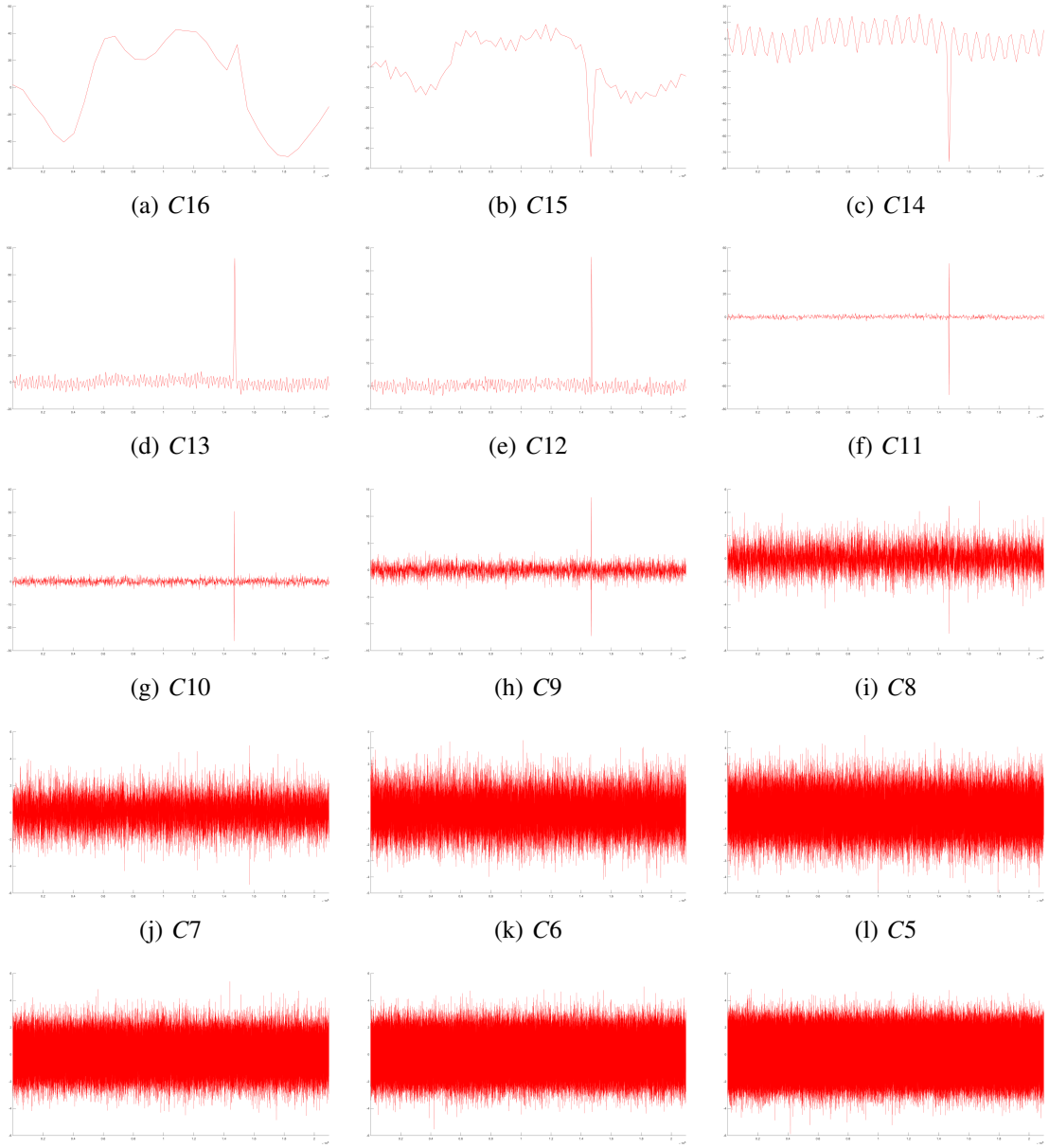


Figure 2.3: *The A16 coefficients, from the DWT decomposition, of the signal in figure 2.2. The signal consisted of* 2^{19} *data points. The Haar wavelet was used up to level 16. Maximum level for the Haar wavelet in this case is 19. The A coefficients represents the overall flux of the signal and it is possible from this figure to distinguish the two wide peaks added at 0.5 and 1.5, that were added to the signal*



(a) $C16$ (b) $C15$ (c) $C14$

(d) $C13$ (e) $C12$ (f) $C11$

(g) $C10$ (h) $C9$ (i) $C8$

(j) $C7$ (k) $C6$ (l) $C5$

(m) $C4$ (n) $C3$ (o) $C2$

Figure 2.4: *The DWT decomposition, using the Haar wavelet, of the signal in figure 2.2 a). Figure a) - o) displays the detail coefficients, C on various sublevels. It is possible to display the anomaly added to the signal in figure b) - h). In figure c) we can also see trails of the sinus wave that was added to the original signal.*

3 Method

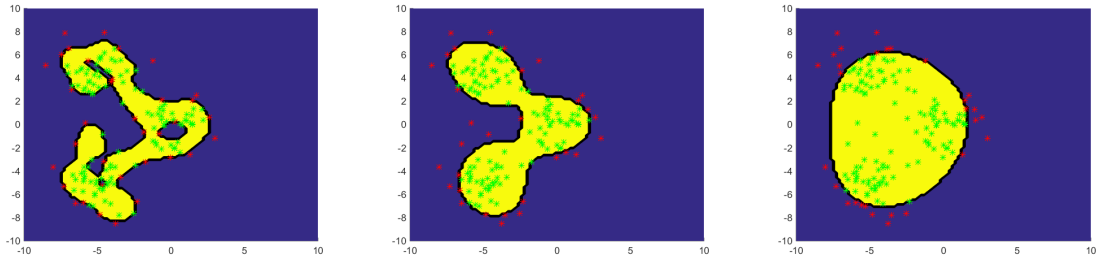
Unlike an ICE or a Jet Engine, the APU that Powercell designs have very few revolving parts since it does not depend on combustion. The only revolving parts you can find are electricly driven pumps, compressors and fans. Typically, the sensor data of importance are not periodic in there appearance; thus a fourier transform would tell very little of the sensor signal. Extreme values could in some cases tell that an importan and abnormal event has occured, but looking at figure 2.2 we see that it would be very hard to set a treshold over what is extreme and what is normal. From this figure, we can tell that it is more the shape of the signal, with its small but significant spike, that strikes as odd rather than the absolute values.

PowerCell have have a series of test units that are deliviering a continous data stream to PowerCells server. The data is sampled from multiple sensors on the unit, at a rate of $\sim 10[Hz]$, batched and is sent to the server at approximatly minute intervalls. This yeilds a multi dimensional time series data containing all the information of what state the unit is currently in. During main operation the unit should behave in a predictable way. Non the less Powercell has experinced disruptions when the unit deviates from the main operation normalty state, such that forced shutdown was the only option. We will use data from a 6 days continous data collection, with known deviations, as a test case to evalute the performance of the anomaly detection framework. The work was performed using object-orientend programming in Matlab.

At the core of the framework is an open source library called *LIBSVM* [7] that implements the *One Class Support Vector Machine* (OCSVM) described in section 2.1. With a low value of the parameter ν, OCSVM can be used as an anomaly detection algorithm. *LIBSVM* is written in *C++* but it can be compiled and integrated in Matlab using the *MATLAB Compiler Toolbox*. *LIBSVM* is a highly reputed tool within in the machine learning community [7] with a well documented interface, support for multiple standard kernels and a built in application for cross-validation. In order to obtain a nonlinear decision boundary, a RBF-kernel was used.

3.1 Training, and parameter selection

Training a *OCSVM*, using a RBF-kernel, will require the selection of two parameters ν and σ. How these parameters are chosen will impact the classification enormously. As we concluded in section 2.1.1 the parameter ν is the fraction of data points that should be classified as abnormal; thus ν can be translated into a level of tolerance. To obtain an anomaly detection algorithm this parameter should be relativly low, if we assume that the training data contains few anomouls points. In practice it could be benefitial to choose a hole range of tolerance levels and subsequently classify the new data on these various levels. This way one gets an understanding of how anomalous the new data potentially is. In this report we set the range $\nu = \{1\%, 2\%, 5\%, 10\%\}$. When ν is decided the question remains how to set σ as a function of ν. There are two phenomenons that can emerge if $\sigma(\nu)$ is chosen porly, these are called overfitting and underfitting. Both of these are undesirable properties that needs to be avoided; in figure 3.1 a) we can see an example of overfitting and in figure 3.1 c) an example of underfitting for a 2-dimensional test problem.



(a) $\sigma = 5$ (b) $\sigma = 20$ (c) $\sigma = 80$

Figure 3.1: *Classification made using the One-Class Suport Vector Machine and an RBF Kernel with different values of the hyperparameter* σ. *In the yellow area the data points are classified as normal and in the blue area they are anomalous. The training data is drawn from three differnet multivariate Gaussian distributions, centered around* $(-5,5)$, $(0,0)$ *and* $(-5,-5)$. *Tolerance level* $\nu = 0.2$

Overfitting seems to occurs when σ is chosen too small and underfitting when σ is too large. Note that the desired classification, Figure 3.1 b), is derived when σ has an intermediate value. Therefore we needed a method for finding: what is an intermediate value of σ for each specific case. The only tool availiable is cross-validation; the idea with this tool is to split the training data into n subsets, where $n-1$ subsets are used as training data and the remaining set will be the test data. By alternating the test set it is possible to get statics of how sensitive the classification is. In this enitre report we used 10-fold cross validation.

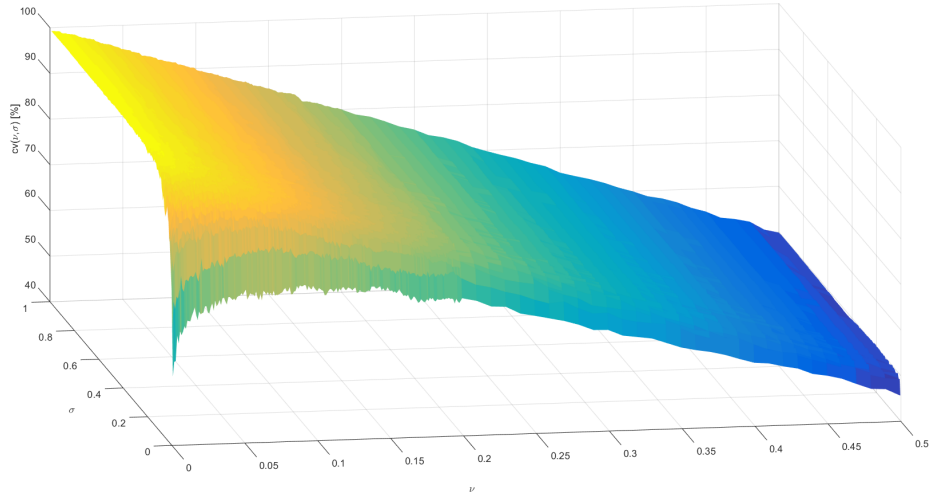
Figure 3.2 a) shows a plot of the cross-validation value $cv(\nu,\sigma)$. It tells us that: if no overfitting has occured the cross-validation value should be approximatly equal to ν, overfitting occurs for small value of σ and underfitting can not be detected directly through cross-validation. By studying 2-dimensional cases, as in figure 3.2, it becomes clear that the optimal classification occurs on the ridge of Figure 3.2 a), in the transition from overfitting to underfitting. This transition is possible to detect, even if cross-validation can not tell the difference between good classification and underfitting. Figure 3.2 b) shows a plot of the difference $\Delta = |\nu - cv(\nu,\sigma)|$ and it appearce as though Δ has an exponential decline towards zero with increasing σ. Note that Δ is bounded such that $0 < \Delta < 1$ and the limit $\sigma = 0$ is prohibited. For each value of ν there exist a $\sigma_0 > 0$ for which cross-validation becomes numerically viable. Optimally σ_0 should be chosen such that $0 << \Delta(\nu,\sigma_0) < 1$. The exponentially declining behaviour of Δ, for a fix value of ν, could then be modeled as

$$\Delta_\nu(\sigma) = \Delta(\sigma_0)\exp\{-a(\sigma-\sigma_0)^b\};\ \ \sigma > \sigma_0,\ a,b > 0 \tag{3.1}$$

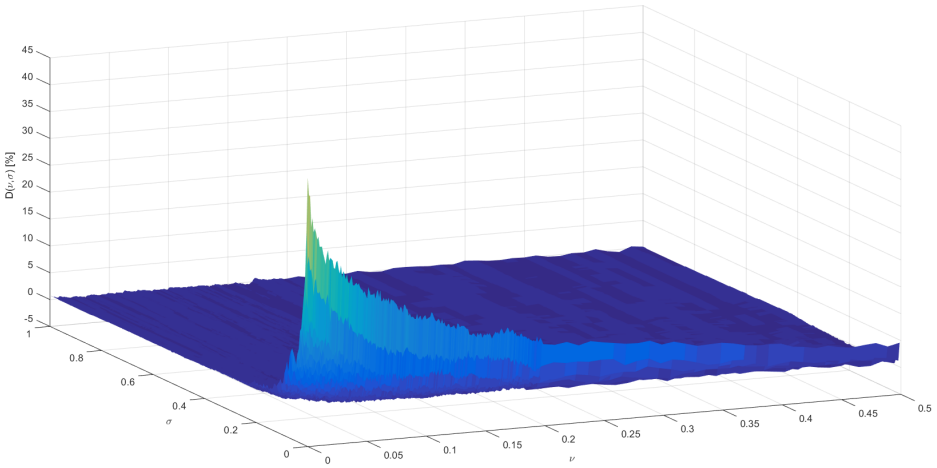
The parameters a,b was calculated using a curvefitting algorithm that minimizes the least square error between samples of Δ and the model, equation (3.1). The empircal values of $\Delta(\sigma_i)$ were sampled from the set

$$\{\sigma_0,...,\sigma_{N-1},\sigma_N,...,\sigma_{2N}:\ \ \Delta(\sigma_{N-1}) > c > \Delta(\sigma_N),\ \ |\sigma_i - \sigma_{i+1}| = \delta\} \tag{3.2}$$

When Δ_ν has dropped to a level of around $1\% \sim 5\%$, then we know that we are on the brink of the ridge where the transintion from overfitting to underfitting occurs. The corresponding



(a) $cv(\nu, \sigma)$



(b) $\Delta(\nu, \sigma)$

Figure 3.2: *a) shows the crossvalidation value* $cv(\nu, \sigma)$. *b) shows the difference* $\Delta = |cv(\nu, \sigma) - \nu|$. *Overfitting can be detected when* $cv \neq \nu$. Δ *declines exponentially when* σ *increases.*

σ^* value will then be

$$\begin{aligned} \Delta_\nu(\sigma^*) &= c; \quad c \in [0.01, 0.05] \\ \sigma^* &= \left[\frac{\ln \Delta(\sigma_0) - \ln c}{a} \right]^{1/b} + \sigma_0 \end{aligned} \tag{3.3}$$

Further testing on 2-dimensional test cases showed that it could be profitable to multiply σ^* with a factor, $d \in [1, 2]$. Rather than being on the brink of the ridge ($d = 1$) we move a little bit past it ($d > 1$).

Using this new approach for parameter selection of the OCSVM we have a found robust

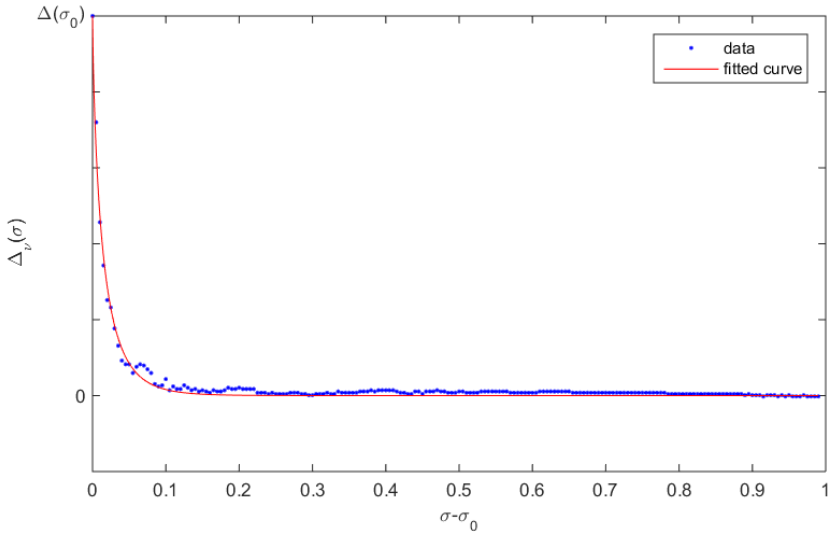


Figure 3.3: *Plot of the exponentially declining behaviour of Δ_ν for a fix choice of ν. The blue data is the sampled value of Δ and the red curve is the fitted model, equation 3.1, to that data.*

method to derive a set of parameters $\{\nu, \sigma(\nu)\}$. The method introduces two new parameters c and d. The drawback of two paramters, rather than one, was compensated by the fact that both c and d are bounded to very small intervals; in contrast to σ that virtually could take any value. Any choice of $\{c,d\} \in [0.01, 0.05] \times [1,2]$ has been proven viable in the 2-dimensional test cases, see figure 4.2.

The multiple cross-validation needed to decide $\sigma(\nu)$ will be the most computationally heavy task. Once the parameters $\{\nu, \sigma\}$ are set, the training of an *OCSVM* object will only be a fraction of the overall computational time. A *OCSVM* object is a data structure that contains all information required for classification; in particular this includes: *Suport-Vectors*, corresponding *Lagrangian multipliers* α_i, normalization values and the offset ρ. All *SVM* methods are sparse which implies that only few data point called *Suport-Vectors* are needed for classification. The sparsity property is advantageous when CPU/memory usage is limiting.

3.2 Flow of Data

In order to create a tool that monitored the gathered data from an APU unit it was important to understand how the flow of data looks and also understand what was the purpose of the data collected. Only then is it possible to design a program that fullfills the criteras that PowerCell has on such system. The end goal is to make a binary classification of the data collected. The classification will be made by an algorithm called *One Class Support Vector Machine*. Since that algorithm and many other machine learning techqniques requires an input data of the form $X_i = \{x_{1,i}, x_{2,i} ..., x_{n,i}\}$, we will need a method of converting timeseries data $Y = \{y(t_1), y(t_2), ..., y(t_m)\}$ to that appropriate form. In this section we will describe in detail how we are able to calculate the feature data X_i

The concept of having units online and gathering abundently with data is a trend that has progressed during reacent years in industry, it is called *Big Data*. The generall objective, for companys like PowerCell, to collect all this data is to monitor how there units are performing and how are they being operated. With that knowledge it is then possible to improve there products, either with software upgrades or with hardware changes in the product development phase. PowerCell is still in its developent phase therefor they are eager to learn how there units behaves during operation. A specific application could be that PowerCell knows, before the customer does, that there is something deviating with there unit. They could then contact the customer and alert them of this issue or remotely make adjustments that might solve the problem that has occured.

A common tool that is widely used within the field of *Big Data*, is called *Hadoop*. The aim of *Hadoop* is to distribute storage and processing of very large data sets on multiple workers. However *Hadoop* is only possible to use if the files are on a *HDFS* format. PowerCell is working with a *HDF5* filesystem rather than *HDFS*, therefore it is not possible to work with the mapreducer functionallity of *Hadoop*. Even if *Hadoop* is a powerfull tool when working with *Big Data*, it is mostly optimized for finding features in historical data. Due to this fact, the design of the program must be such that it reads and analyse the data the moment it arrives rather than to store it and analys it later.

PowerCell has fitted there APU units with a data-logg filesystem that enables them to logg time-series data from an enitre set sensor. With a given frequency these logg-files are uploaded to a cloud storage server. From this server the logg files can be download to the PowerCell server. After all files from all sensorns are downloaded they are processed and stacked into a *HDF5* file format. The structure of that file has been standardise to a standard the company has chosen to name *PSM1*. Knowing that the files were structured in a similar and predictable way, it enabled us to create a tool that potentially could work for any PowerCell units. Figure 3.4 shows a flow chart of how the *PSM1* files are created.

Each *PSM1* file is given a timestamp that correlates to the earliest time of any sensor data

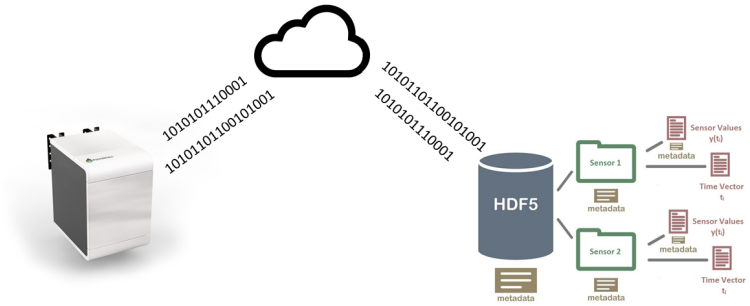


Figure 3.4: *An illustration of the data flow. Raw files for all the different sensors are sent to a cloud storage at minutes interval. Subsequently the raw data is collected from the cloud to PowerCells server. At the server the raw data is processed and merged into a PSM1 file format.*

that has been logged in that perticular file. The timestamp is a double precision variable that denotes the number of days plus the fraction of days elapsed from year 0 (B.C). Note that all sensor data in the *PSM1* are unlikely to start and end at the same time. Each sensor contains

two vectors of equal size. One with the signal values and one time vector with fractions of days elapsed since the timestamp of the *PSM1* file. To get a continous time-series data with complementay timestamps for each data point we need to add the timestamp of the file to the time vector of each sensor. At the edges of each file the data-loggs are known to be both insuffcient or redundant, meening that data could either be missing or duplicated. To join the old data with the new data requiered some preprosseing, where each sensor needed seperate treatment. Missing data could obviously not be recovered, the only thing we could do was to set a limit of how much data we tolerate to loose before we report an error. Since each data point has a time stamp, the duplicated data could easily be removed.

When a new *PSM1* file appears on the server and it is read by the program, the program scans all sensors and convert their time vector into a timestamp vector. It checks that all sensors data-logg are, continuos with a given tolerance and marks the sensor that stoped logging data earliest. The timestamp where this occurs is considered to be the timefront of that *PSM1* file. To avoid any issues, no data will be extracted in front of the timefront. Since the data in front of the timefront will still be useful when an additional file is added it is saved for later use. In Figure 3.5 we explain how *PSM1* files are merged into a multi dimensional timeseries data vector.

It is more benefitial to have one continous data vector, rather than multiple disconected

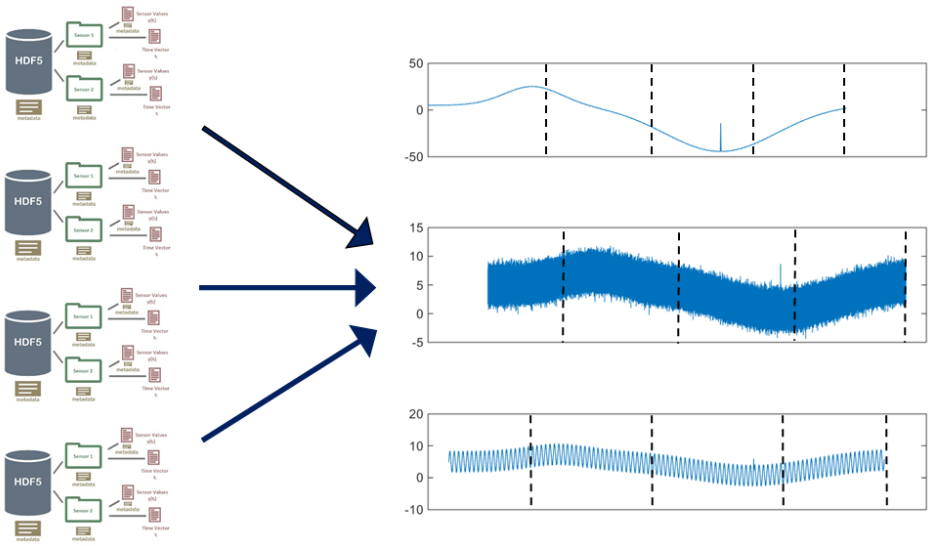


Figure 3.5: *An Illustration of how multiple PSM1 files are merged into a multiple timeseries sensor data. Note that the timefront for the different sensors are not alligned.*

vectors, since the intention is to extract feature data from a timeseries interval rather than from one single data point. If the timeseries vectors are disconected one can not extract as many intervals as is otherwise possible when they are all connected, thus we would get less unique data points. With only one single data point it is harder to detect trends in the data; in addition it is impossible to filter out what is noise and what is a signal. Precisely how the feature data is calculated on this interval is described in section 3.3.

Given a timestamp T_i the program should be able to extract an interval of timeseries data,

Y_i with a fixed amount of data points, from multiple sensors $S_k \ : k = \{1,..,m\}$. From each sensor, S_k, a subset of the feature data, $X_{j^*} \ : j^* = \{j_{k_1},..,j_{k_n}\}$, is calculated. Subsequently the entire feature data vector $X_i = \{X_j \ : j = \{j_{1_1},..,j_{1_n},j_{2_1},...,j_{k_1},...,j_{k_n},..,j_{m_n}\}\}$ can be assembled for the associated timestamp T_i.

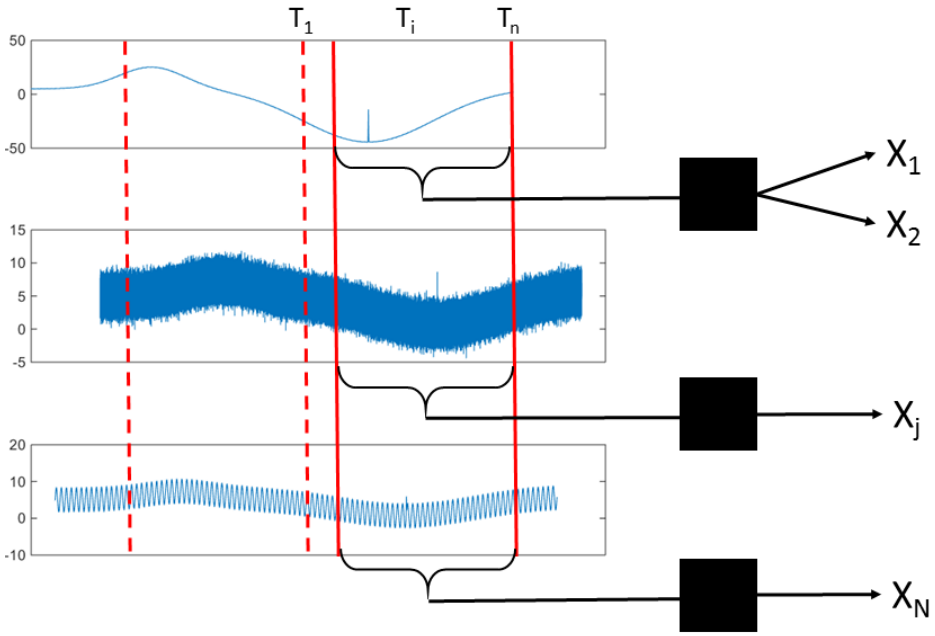


Figure 3.6: *An illustration of how timeseries interval Y_i is converted into feature data X_j, used by the OCSVM. The black boxes representes one or multiple operations from Table 1.*

3.3 Calculating feature data from a sensor

There are many mathematical operations you can perform on a time-series interval with fixed length. Each one of theses operations reflects different properties about that interval. Basic operations like *max/min*, *mean* and the *standard deviation* are meassures of extreme, expectancy value and volatility; all are interesting properties that might be able to detect anomalies as they occur. However it is easy to find examples when they can not detect an event that appears anomalous for the human eye, see figure 2.2.

A key aspect to this report was to investigate if the *Wavelet Transform* could be used with advantagous results. The idea is to use the wavelet coefficients of the *Discrete Wavelet Transform* (DWT) as the input feature data to the *OCSVM*. The localization property, of the wavelet coefficents, ensures that anomalies occuring at a specific part and timescale of the signal will also be reflected as an anomaly at a specific part of the DWT spectra. If a signal is known to have normal fluctuations at some specific timescales, typically it would be interesting to monitor all the other timescales where there should be no fluctuations. With this technique it is possible to distinguish an anomaly, that is relatively small in comparison to the normal fluctuations, with a much greater significance level.

Table 1 shows a list of all the operations that has been integrated so far into the framework that are possible to use as input data to the *OCSVM* algorithm.

Decomposition	Level	Sub-level	Operation 2	Operation 1
-	-	-	mean/std(Y)	-
-	-	-	max/min(Y)	-
-	-	-	max($\|Y - mean(Y)\|$)	$\|...\|_{100\%-Z\%}$
Wavedec(Y)	N	-	max/min(A_N)	
Wavedec(Y)	N	-	$\|$max(A_N)-min(A_N)$\|$	-
Wavedec($\|Y - mean(Y)\|$)	N	-	max/min(A_N)	-
Wavedec($\|Y - mean(Y)\|$)	N	-	std(A_N)	-
Wavedec(Y)	N	L	max($\|C_{N,n}\|$)	$\|C_{N,n}\|_{Z_1\%-Z_2\%}$
Waverec(Y)	N	L	mean/std(Y)	-
Waverec(Y)	N	L	max/min(Y)	-
Waverec(Y)	N	L	max($\|Y - mean(Y)\|$)	$\|...\|_{100\%-Z\%}$

Table 1: *List of the various mathematical operations possible to perform on the timeseries interval Y.*

4 Results

The reason for using only 2-dimensional test cases is due to the fact they are the only ones can be visualized and verified so that the classification is satisfying. Its not the algorithm that does not work in higher dimensions but rather the verification of hyperparameters. The idea of the method developed in section 3.1 is to have an unbiased selection of the parameter $\sigma(d)$. The curse of dimensionality says that a good classification will require more data in a higher dimensional space, as the euclidean distance between the data points increases; therefore it is likely to assume that the optimal choice of σ is very much dependent on the dimensionality. The parameter d should theoretically not be affected by the dimensionality as it is mearly a meassure of overfitting versus underfittning, independent of the numder of dimensions in feature space.

Two sensors with distinguishable characteristics were choosen in order to test the performance of the developed method. For each sensor two features were selected from Table 1. Subsequently, feature data was calculated at minutes interval from the entire 6 days of data. This yielded a $\sim 1200 \times 2$ feature data vector, where the first half of it was used as training data. Training and parameter selection, of the *OCSVM*, was made on multiple tolerance levels $\nu = \{0.01, 0.02, 0.05, 0.1\}$. When the training was completed all feature data points were classified as either normal ($f = 1$) or anomolous ($f = -1$), equation (2.10). Figure 4.1 shows the sensor signal along with the classification results of all the tolerance levels, for one of the two sensors selected. Similarly figure 4.2 shows the signal and the classification for the second sensor.

To make good evaluations of how the method is performing it is important to get an understanding of how the input feature data is localized in feature space. In Figure 4.3 and 4.4 a series of snapshots are made of the feature space where data points are mostly localized. In addition the subspace where datapoints are classified as normal is highlighted as the yellow area and the abnormal area is highlighted as blue. From these figures it becomes apparent how the classification depends on the parameter selection of ν and $\sigma = d\sigma^*$. The data points that are inside the decision boundary are plotted in green and the ones are outside are plotted

in red. Note that all feature data has been normalized and scaled such that the mean equals zero and the standard deviation equals one. One type of feature data could have fluctuations that are many orders of magnitude larger than other types of feature data. Theoretically it should not matter if normalization is performed or not, but numerically it can be essential to get the *SMO* to converge.

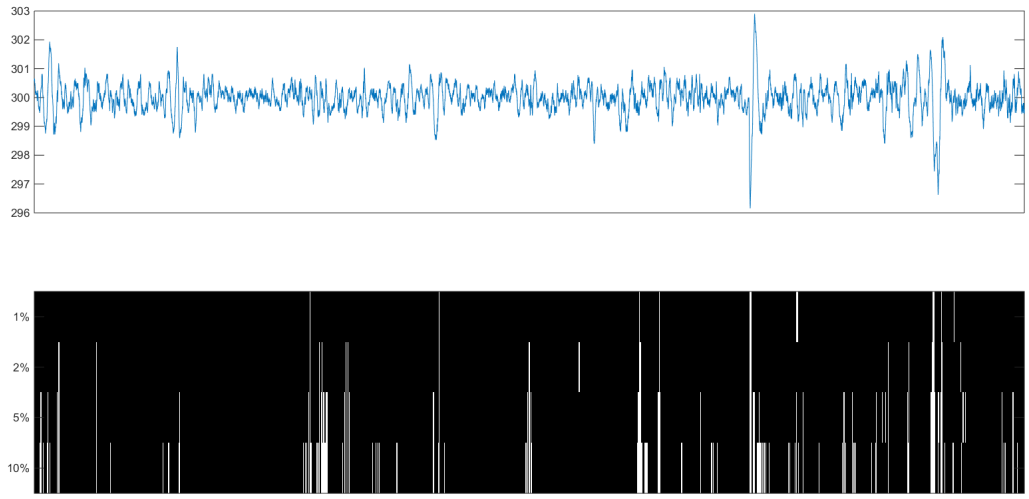


Figure 4.1: *Plot of the timeseries sensor data (TS101) for continuous operation over 6 days. The corresponding anomaly classification on the multiple tolerance levels* $\nu = \{0.01, 0.02, 0.05, 0.1\}$. *A white pixel indicates an anomaly detection at that perticular time and tolerance level. The parameter* $d = 2$. *The 2 types of input feature data are mean and Wavedec(max(*$\|C_{17,12}\|_{0\%-100\%}$*), from sensor TS101.*

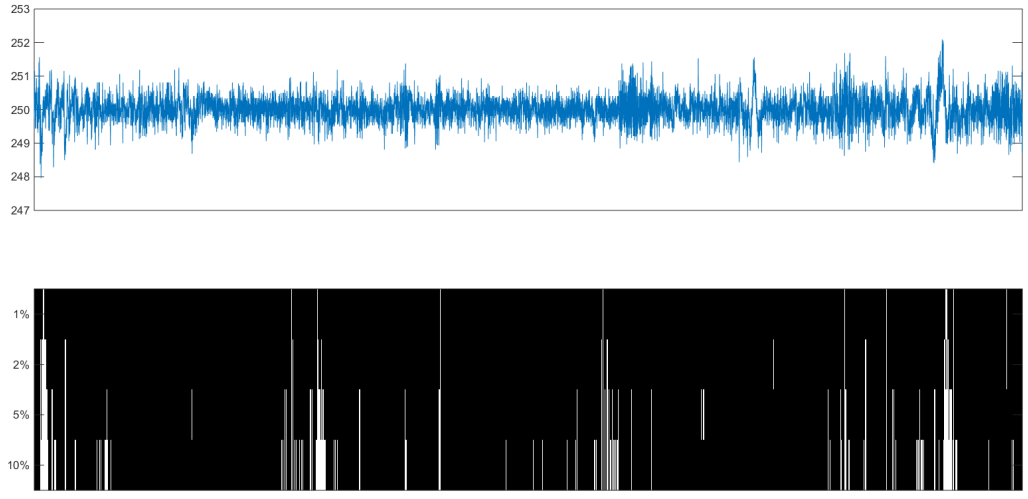
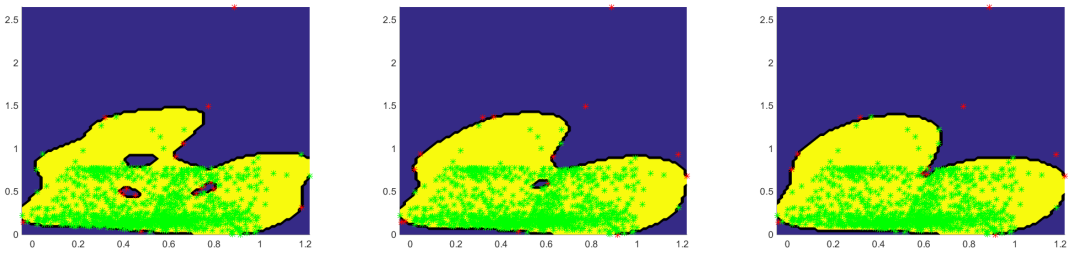
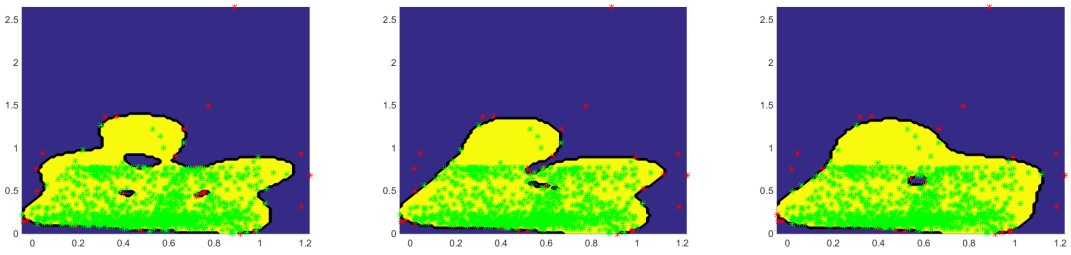


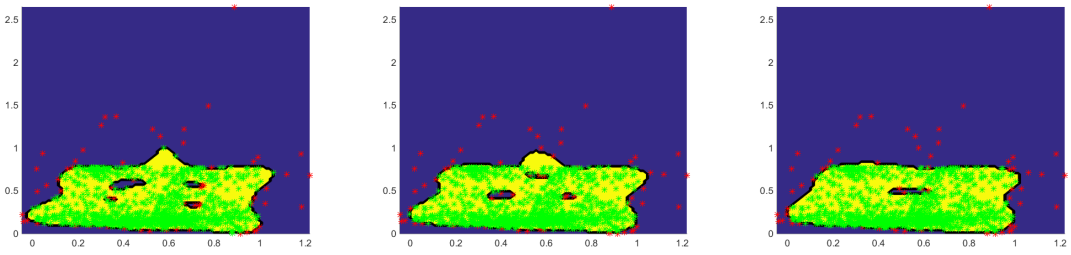
Figure 4.2: *Plot of the timeseries sensor data (TS104) for continuous operation over 6 days. The corresponding anomaly classification on the multiple tolerance levels* $\nu = \{0.01, 0.02, 0.05, 0.1\}$. *A white pixel indicates an anomaly detection at that perticular time and tolerance level. The parameter* $d = 2$. *The 2 types of input feature data are mean and Wavedec(max(*$\|C_{17,12}\|_{0\%-100\%}$*), from sensor TS104.*



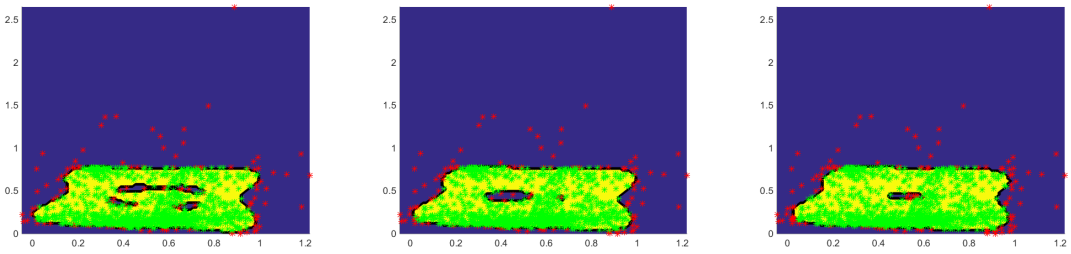
(a) $\nu = 0.01, d = 1.0$ (b) $\nu = 0.01, d = 1.5$ (c) $\nu = 0.01, d = 2.0$



(d) $\nu = 0.02, d = 1.0$ (e) $\nu = 0.02, d = 1.5$ (f) $\nu = 0.02, d = 2.0$

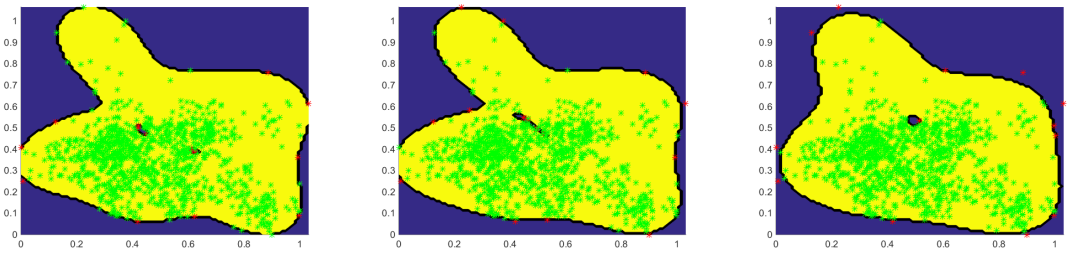


(g) $\nu = 0.05, d = 1.0$ (h) $\nu = 0.05, d = 1.5$ (i) $\nu = 0.05, d = 2.0$

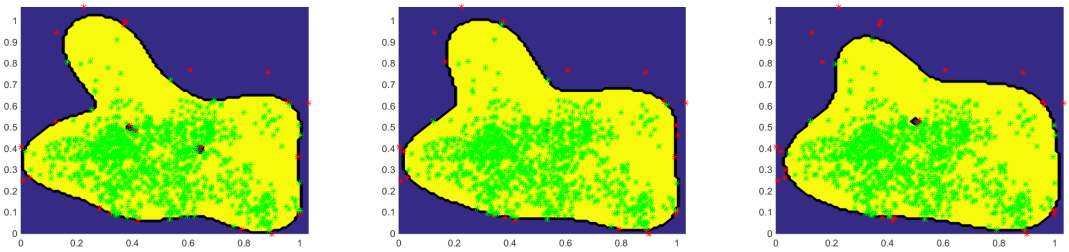


(j) $\nu = 0.10, d = 1.0$ (k) $\nu = 0.10, d = 1.5$ (l) $\nu = 0.10, d = 2.0$

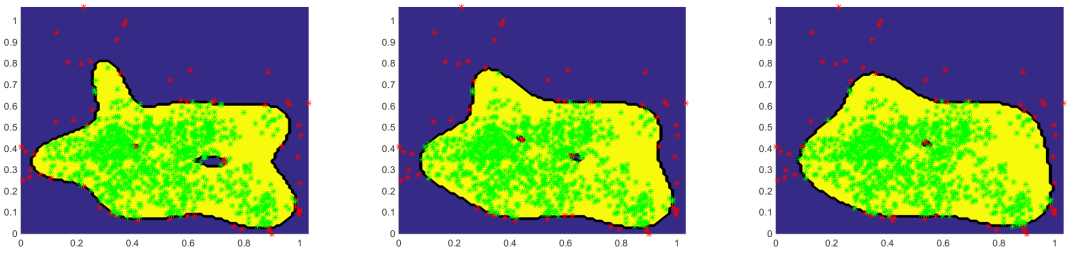
Figure 4.3: *Plot of the decision boundary in feature space for an array of the the parameters, ν and d. The 2 types of input feature data are mean and Wavedec(max($\|C_{17,12}\|_{0\%-100\%}$), from sensor TS101.*



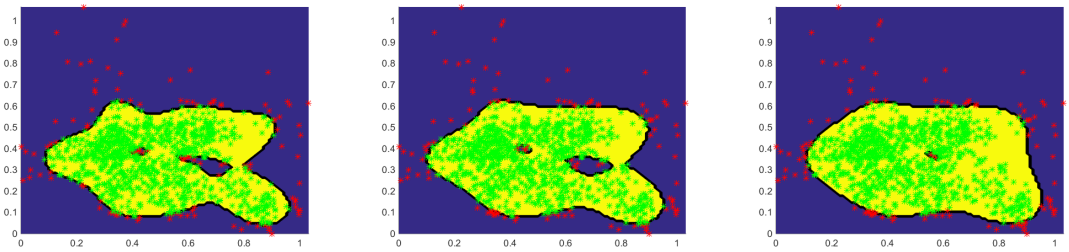
(a) $\nu = 0.01, d = 1.0$ (b) $\nu = 0.01, d = 1.5$ (c) $\nu = 0.01, d = 2.0$



(d) $\nu = 0.02, d = 1.0$ (e) $\nu = 0.02, d = 1.5$ (f) $\nu = 0.02, d = 2.0$



(g) $\nu = 0.05, d = 1.0$ (h) $\nu = 0.05, d = 1.5$ (i) $\nu = 0.05, d = 2.0$



(j) $\nu = 0.10, d = 1.0$ (k) $\nu = 0.10, d = 1.5$ (l) $\nu = 0.10, d = 2.0$

Figure 4.4: *Plot of the decision boundary in feature space for an array of the the parameters, ν and d. The 2 types of input feature data are mean and Wavedec(max($\|C_{17,12}\|_{0\%-100\%}$), from sensor TS104.*

5 Discussion

The main objective of the work that has been done was to design a framework that can be used as a health monotoring system in PowerCells current data infrastructure. One major concerns was to investigate the feasibility of the *DWT + OCSVM* algorithm, that represents the foundation of the entire framework.

The inclusion of the *DWT* into the framework was done in such a way that it only added functionality to the framework. If it will be proven in the future that wavelets does not add any extra usability it can easily be excluded again. However the results shows that, if the parameters of the *Wavelet Transform* are choosen correctly it can indeed detect properties of the signal that is otherwise hard to detect with conventional mathematical operations. The problem is that, it is only true if the parameters are choosen correctly. The typical things that needs to be decided are: type of decomposition wavelet, decomposition level and wavelet coefficient level. These three properties in combine can be translated into an approximate timescale where we are looking for anomalies at. It is possible to map the parameter selection of the *DWT* into an approximate timescale but that work has not been included in this report. Before the framework can be put in to use it will be necessary to conduct such a mapping. Subsequently one will also need to decide what timescales are relevant for each perticular sensor.

Another aspect that is important for good classification is the amount of input data to the *OCSVM* algorithm. PowerCell supplied approximately 6 days of usefull continous data sampled at a rate of $\sim 10[Hz]$. Even if this is a lot of data much of it is reduntant when we look at the behaviour of the signal, using *DWT*. With this approach its less important how often the signals are sampled and more important over how long time the signals collected its data. Those 6 days of data also needs to be devided into a training set and test set, reducing the reliability of the classification even further. Some of the signals have normal fluctuations that extends in the order of hours; for that reason we need longer time intervalls and more data. All these facts put together accumlates to the assumption that, 6 days is a too short time period for the unit to exhibit all possible types of normal behavior necessary for a good classifiction. Note that the framework can easily be converted to only input the signal values directly, into the *OCSVM* algorithm, thus eliminating some of the issues with insufficient data. Nevertheless, that idea has been disregarded since it would only be possible to detect anomalous absolute values and not trends.

One of the key reasons for PowerCell to initiate this work was to learn more about there own product. There goal is to increase the life expectancy of the unit and if it behaves unstable, in an unpredictable way, the life expectancy might be compromised. They are in such an early stage of their development that they don not know exactly what they are looking for; for that reason an unsupervised algorithm is more suitable for PowerCells needs. The *OCSVM* is advantagous because it is regared as an unsupervised or semi-unsupervised learning algorithm. The method developed for parameter selection of the hyperparameter $\sigma(\nu)$ proved to be robust and without any need of supervision. The remaining parameter $\nu \in]0,1[$ can be interpreted as a tolerance level; rather than choosing one level it is possible to choose multiple tolerance levels for classification of a data point. Since we are looking for anomalies it is only interesting if ν is choosen somewhere in the range from $0-10\%$.

The purpose of the entire framework is to derive *OCSVM* objects, containing all information necessary for classification. The combination of a *OCSVM* object and instructions of how to calculate feature data are the only tools needed too perform a classification. The sparsity of the *OCSVM* object makes it favourably when computation time is limited, as it would be if classification were to be performed locally on the APU unit. If a classifier is set to operate locally on a unit, the kind of feature data that is choosen to be observed needs to be optimized for the limiting computing power conditions. The *DWT* is a fairly fast method, due to its downsampling properties. The Haar wavelet is the least advanced wavelet and thus it is also the fastest. Since we are not interested in reconstructing the signal after decomposing it, the Haar wavelet satisfies the needs of our application just as well as any of the more advanced wavelet. The Haar wavelet is even preferable because it is possible to decompose to a higher level than any other wavelet; this allows us to study longer time scales with less data.

The result presented are focused on the potential and feasibilty of the method for choosing the hyperparameter to the RBF-kernel, which is really important for good classification. Even if the results shows that anomoulos behaviour is detected, it should not be interpret as a conclusive proof that it was those behaviours which caused the forced shutdown of the unit. There is still some work left to do in order to assess if any or multiple sensor showed conclusive signs of anomalies that could explain the shut down. Naturally this work will also have to incorporate PowerCells knowledge of the unit; a sensor can have bad readings but still have no correlation to the error that emerged.

With the framework it is possible to have multiple monitors. If a certain type of error is assumed to correlate with one set of sensors and another error correlates with a different set of sensors they can still be monitored simultaneously. Each individuall monitor object consists of an *OCSVM* object along with instructions of how to extract feature data from all the sensors. If there is any douplet instructions the framework will notice that and only calculate the feature data ones, in order to save computation time.

5.1 Recommendations

The conclusion from the work conducted so far at PowerCell can be summorized into some recommendations for further development that needs be adressed.

- The success of a machine learning algorithm depends heavily on the amount of training data. PowerCell will require more data if they wish to implement this anomaly detection framework any further.
- To have any use of the work that has been done so far it is essential to start mapping what timescales are relevant for each sensor and translate that into an appropriate feature data instruction.
- It also important to start reflecting upon on how many monitors there should be? Should there be one for each sensor or should there be just a few, monotoring a collection of sensors with high correlation? The framework is versatile and would potentially support both appraoches simultaneously.
- The *OCSVM* can be converted into a statistical method, making it possible to estimate the probability that a new data point is anomalous. Investigating how a statistical *OCSVM* can be incorporated into the framework is recommended.
- The entire report has had clear focus on the *OCSVM* as the anomaly detection algorithm of choice. An interesting topic, that has been disregarded in this report, is to investigating how other methods, like the *Principal Component Analysis*, would perform in comparison to the *OCSVM*.

6 Reference

[1] P. Hayton, S. Utete, D. King, S. King, P. Anuzis and L. Tarassenko (2007). Static and dynamic novelty detection methods for jet engine health monitoring. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 365, ss. 493-514.

[2] S. Takianngam, W. Usaha (2011). Discrete Wavelet Transform and One-Class Support Vector Machines for anomaly detection in wireless sensor networks. *Intelligent Signal Processing and Communications Systems*, page. 1-6.

[3] X. Liu, Y. Zhu, Y. Zhang, X. Wang (2011). Prediction Based on Wavelet Transform and Support Vector Machine. *Communications in Computer and Information Science*, volume 243, page. 618-625.

[4] V. Vapnik, B. Scholkopf, and C. Burges.Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors. *Proceedings, First International Conference on Knowledge Discovery & Data Mining*, Menlo Park, 1995. AAAI Press

[5] B. Scholkopf, J Platt, J. Shawe-Taylor, A. Smola and R. Williamson (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, volume 13, issue 7, page. 1443 - 1471.

[6] A. Romeo, C. Horellou, J. Bergh (2004). A wavelet add-on code for new-generation N-body simulations and data de-noising. *Monthly Notices of the Royal Astronomical Society*, volume 354, page. 1208-1222

[7] C. Chang, C. Lin (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, volume 2, issue 3, no. 6.

Anomaly Detection in PowerCells Auxiliary Power Unit

HAMPUS G. HJORTBERG

Examiner: DEVDATT DUBHASHI

Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden June 2015