Utrecht University

Doctoral Thesis

# Temporal Segmentation using Support Vector Machines in the context of Human Activity Recognition

*Author:*
R.Q. Vlasveld

*Supervisor:*
Dr. James Smith

*A thesis submitted in fulfilment of the requirements for the degree of Master of Science*

*in the*

Research Group Name
Department or School Name

October 2013

# Declaration of Authorship

I, R.Q. Vlasveld, declare that this thesis titled, 'Temporal Segmentation using Support Vector Machines in the context of Human Activity Recognition' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

*"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."*

Dave Barry

UNIVERSITY NAME (IN BLOCK CAPITALS)

# *Abstract*

Faculty Name
Department or School Name

Master of Science

**Temporal Segmentation using Support Vector Machines in the context of Human Activity Recognition**

by R.Q. Vlasveld

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

# *Acknowledgements*

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

# Contents

# List of Figures

# List of Tables

# Acronyms

**AIC** Akaike Information Criterion. 9, 57

**AR** Autoregressive. 31

**BIC** Bayesian Information Criterion. 9, 57

**changeFinder** is a unifying framework proposed by Takeuchi and Yamanishi [65] which combines outlier detection and change detection. 14, 33, 34

**CUSUM** Cumulative Sum. 3, 4, 7–9, 36, 52, 57

**GMM** Gaussian Mixture Model. 17

**HAR** Human Activity Recognition. 1, 6, 11, 13, 28

**HMM** Hidden Markov Model. 9, 65

**ICSS** Iterated Cumulative Sums of Squares. 4, 9, 51

**i.i.d.** independently and identically distributed. 14, 33, 67

**I-SVDD** Incremental Support Vector Data Description. 32, 33

**KKT** KarushKuhnTucker. 32

**KL** Kullback-Leibler. 12, 14, 16

**KLIEP** Kullback-Leibler Importance Estimation Procedure. 4, 5, 8, 33, 52, 57

**MLE** Maximum Likelihood Estimates. 9

**MOSUM** Moving Sum. 3, 7

**OCC** One-Class Classification. ix, 13, 15–18, 57

**OC-SVM** One-Class Support Vector Machine. 12, 18, 23, 32, 57, 59

**PCA** Principal Component Analysis. 7, 8, 18

**PDF** Probability Density Function. 2, 3, 16

**QP** Quadratic Programming. 21, 23

**RBF** Radial Base Function. 22, 25–28, 36, 70

**ROC** Receiver-Operating Characteristic. 56

**RT** Ratio-Thresholding. 36

**SV** Support Vector. 24–26

**SVC** Support Vector Classifier. 66

**SVCPD** Support Vector based Change Detection. 12, 14, 33, 34

**SVDD** Support Vector Data Description. vi, ix, 12–14, 17, 18, 23, 25, 26, 28, 32, 33, 36, 56, 57, 66, 71

**SVM** Support Vector Machine. ix, 3, 5, 6, 11, 13, 18–25, 27, 32–34, 56, 59, 70, 71

**$\nu$-SVM** $\nu$-Support Vector Machine. vi, ix, 11–13, 18, 23–26

**SWAB** Sliding Window And Bottom-up. 6, 7

# Symbols

| | | |
|---|---|---|
| $a$ | distance | m |
| $P$ | power | W ($Js^{-1}$) |
| $\omega$ | angular frequency | $rads^{-1}$ |

*For/Dedicated to/To my...*

# Chapter 1

# Introduction

## 1.1 Outline

*Not intented for the reader.*

- Context of research (Human Activity Recognition (HAR)), real-world applications
- Current methods, wrapper vs. filter methods
- Problem statement with current filter methods (which follows from Chapter 3 which goes in-depth with methods).
- Purpose of this research. E.g. "Find a better algorithm for short-activity segmentation"
- Relate to real-world applications
- Outline for rest of the thesis

# Chapter 2

# Literature review

## 2.1 Outline

*Not intented for the reader.*

- Literature review about Temporal Segmentation (previous draft was more about classification)
- Consider methods for the context of filter-methods for classification
- Take a loot at 3-4 different kind of methods for change detection:
  - Introduction with a lot of techniques
  - Explain why look at a few
  - CUSUM - or other more traditional methods
  - Density-ratio estimation
  - Support Vector Machines (?) - if there are more sources about this
  - (Dimensionality reduction) –¿ probably not
  - *Not to much attention to all techniques, focus is on SVM*
- With each method, shortly look at characteristics, strengths and weaknesses and consider applicability to accelerometer sensor data

## 2.2 Statistical framework

Many applications require the detection of time points at which the underlying properties of a system change. This problem has received a lot of attention in the fields of data

mining, etc... ****** TODO: list and refs ******. Often this problem is formulated in a statistical framework, by inspecting the underlying data generating Probability Density Function (PDF) of the time series data. A change point is then defined as a significant change in the properties of the PDF, such as the mean and variance.

The widely used Cumulative Sum (CUSUM) method by Basseville *et al.* [10] takes this approach. It originates from control methods for detection from bench marks. This method and some derivatiges are discussed and analyzed in section 2.3.

Many methods rely on pre-specified parametric model assumptions and considers the data be independent over time, which makes it less flexible to real-world applications. The methods proposed by Kawahara *et al.* [38] and Lui *et al.* [46] try to overcome these problems by estimating the *ratio* between the PDF, instead of estimating each PDF. This approach is discussed and analyzed in section 2.4.

The density-estimation methods rely on the log-likelihood ratio between PDFs. The method of Camci [14] takes an other approach within the statistical framework, by using a SVM. One problem it tries to overcome is the (claimed) weakness of many methods to detect a decrease in variance. The method represents the distribution over the data points as a hyper-sphere in a higher dimension using the kernel trick. A change in the PDF is represented by a change in the radius of this sphere. Section 2.6 discusses the SVM-methods. ****** TODO: Put emphasis that this is the source of inspiration for the chosen method? ******

****** TODO: Maybe add dimensionality reduction, but for now leave out ******

## 2.3 CUSUM

*Notes:* Non-Bayesian change detection algorithm (i.e. no prior distribution believe available for the change time). The CUSUM method is developed by Page [56] for the application of statistical quality control (it is also known as a control chart). Primary for detection of mean shift. The Moving Sum (MOSUM) of squares test for monitoring variance changes [35]. Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance [36]

An often used approach in the statistical framework of change detection is the CUSUM as introduced by Page [56]. Originally used for quality control in production environments, its main function is to detect change in the mean of measurements and has been applied to this problem [10]. It is an non-Bayesian method and thus makes no assumptions (or: prior belief distributions) for the change points. Many extensions to this method

have been proposed. Some focus on the change in mean, such the method of Alippi and Roveri [4]. Others apply the method the problems in which the change of variance is under consideration. Among others are there the centered version of the cumulative sums, introduced by Brown, Durbin and Evans [13] and the MOSUM of squares by [35].

The method of Inclán and Tiao [36] builds on the centered version of CUSUM [13] to detect changes in variance. Using the Iterated Cumulative Sums of Squares (ICSS) algorithm they are able to find multiple change points in a reflective manner. Let $C_k = \sum_{i=1}^{k} \alpha_t^2$ be the cumulative sum of squares for a series of uncorrelated random variables $\{\alpha_t\}$ of length $T$. The centered (and normalized) sum is squares is defined as

$$D_k = \frac{C_k}{C_T} - \frac{k}{T}, \quad k = 1, \ldots, T, \quad \text{with } D_0 = D_T = 0. \tag{2.1}$$

For a series with homogeneous variance, the value of $D_k$ will oscillate around 0. In case of a sudden change, the value will increase and exceed some predefined boundary with high probability. The behavior of $D_k$ is related a Brownian bridge. By using an iterative algorithm, the method is able to minimize the masking effect of successive change points.

One of the motivations for the ICSS algorithm was the heavy computational burden involved with Bayesian methods, which need to calculate the posterior odds for the log-likelihood ratio testing. The ICSS algorithm avoids applying a function at all possible locations, due to the iterative search. The authors recommend the algorithm for analysis of long sequences.

## 2.4 Change-detection by Density-Ratio Estimation

Many approaches to detect change points, regarded as a change in the underlying probabilistic generation, monitor the logarithm of the likelihood ratio between two consecutive intervals. Some methods which rely on this are novelty detection, maximum-likelihood estimation and online learning of autoregressive models [38]. A limitation of these methods is that they rely on pre-specified parametric models. Non-parametric models for density estimation have been proposed, but it is said to be a hard problem [30, 64]. A solutions to this is to estimate the *ratio* of probabilities instead of the probabilities themselves. One of the recent methods to achieve this is the Kullback-Leibler Importance Estimation Procedure (KLIEP) by Sugiyama *et al.* [63].

The method proposed by Kawahara and Sugiyama [38] is composed of an online version of the KLIEP algorithm. The method also considers *sequences* of samples (rather than samples directly) because the time series samples are generally not independent over time. An advantage over other non-parametric approaches, such as sequential one-class

support vector machines, is that the model has an natural cross-validation procedure. This makes that the value of tuning parameters, such as the kernel bandwidth, can be objectively obtained.

In their formulation change is detected by monitoring the logarithm of the likelihood ratio between the reference (past) and test (current) time intervals

$$S = \sum_{i=1}^{n_{\text{te}}} \ln \frac{p_{\text{te}}(\boldsymbol{Y}_{\text{te}}(i))}{p_{\text{rf}}(\boldsymbol{Y}_{\text{te}}(i))} \tag{2.2}$$

Where $\boldsymbol{Y}_{\text{te}}(i)$ is a sequence of samples from the test interval. A change is detected when $S > \mu$, for some predetermined threshold $\mu$. The question is then how to calculate the density ratio

$$w(\boldsymbol{Y}) := \frac{p_{\text{te}}(\boldsymbol{Y}))}{p_{\text{rf}}(\boldsymbol{Y})} \tag{2.3}$$

because this ratio is unknown and should be estimated. The naive approach is to estimate the ratio by taking the ratio of the estimated densities. Since this is known to be a hard problem and sensitive for errors, the solution would be to estimate the ratio directly.

The procedure of the method proposed by Kawahara and Sugiyama [38] is to first apply the batch KLIEP algorithm with model selection for initial parameter $\boldsymbol{\alpha}$ and kernel width calculation. Then for every new sample the reference and test intervals are shifted and the calculated parameters $\boldsymbol{\alpha}$ are updated. Finally the logarithm of the likelihood ratio is evaluated. If it is above the threshold $\mu$ the current time is reported as a change point.

Improvements in this line of research, by Liu *et al.* [46] has led the application of improved density-ratio estimation methods to the problem of change detection. Such an improvement is the Unconstrained Least-Squares Importance Fitting (uLSIF) method [37] and an extension which possesses a superior non-parametric convergence property: Relative uLSIF (RuLSIF) [75].

## 2.5 Temporal Segmentation

- Given overview of segmentation techniques, for times series data
- Use different "point-of-views", or terms
- "Segmentation"
- "Change detection"
- "Novelty detection"

- Specific view on SVMs

This section gives an overview of the literature on temporal segmentation in the context of HAR. It takes a look on different implementations and methodologies. A wide range of terms and subtle differences are used in the field, such as 'segmentation', 'change detection', 'novelty detection' and 'outlier detection'. These will be the categorical terms for which we discuss the literature. Finally we will discuss other applications of SVMs in the context of these terms.

### 2.5.1 Segmentation

****** TODO: This subsection is mainly from previous draft version ******

****** TODO: Create compact notation, one sentence per paper max ******

Segmentation methods can roughly be categorized in three methods in the way the data is processed, as discussed by Avci *et al.* [7]:

- **Top-down** methods iteratively divide the signal in segments by splitting at the best location. The algorithm starts with two segments and completes when a certain condition is met, such as when an error value or number of segments $k$ is reached. These methods process the data points recursively, which results in a complexity of $O(n^2k)$.
- **Bottom-up** methods are the natural complement to top-down methods. They start with creating $n/2$ segments and iteratively join adjacent segments while the value of a cost function for that operation is below a certain value. Given the average segment length $L$, the complexity of this method is $O(nL)$.
- **Sliding-window** methods are simple and intuitive for online segmenting purposes. It starts with a small initial subsequence of the time series. New data points are joined in the segment until the fit-error is above a threshold. Since the data is only processed very locally, these methods can yield in poor results [40]. The complexity is equal to the bottom-up approach, $O(nL)$, where $L$ is the average segment length.
- **Sliding Window And Bottom-up**, as introduced by Keogh *et al.* [40], joins the ability of the sliding window mechanism to process time series online and the bottom-up approach the create superior segments in terms of fit-error. The algorithm processes the data in two stages. The first stage is to join new data points in the current segment created by a sliding window, and pass this to a

buffer with space for a few segments. The buffer then processes the data Bottom-up and returns the first (left-most) segment as final segment. Because this second stage retains some (semi-)global view of the data, the results are comparative with normal Bottom-up. It is stated by Keogh *et al.* that the complexity of Sliding Window And Bottom-up (SWAB) is a small constant factor worse than that of regular Bottom-up.

It is clear that for the application of this research sliding-window and preferably SWAB-based algorithms should be considered.

The SWAB method proposed by Keogh *et al.* [40] is dependent on an user setting, providing the maximum error when performing both stages. Each segment is approximated by using piecewise linear representation (PLR), an often used method. The user provided error threshold controls the granularity and number of segments. Other methods have been proposed, such as an adaptive threshold based on the signal energy by Guenterberg *et al.* [26], the adaptive CUSUM-based test by Alippi *et al.* [4] and the MOSUM by Hsu [35] in order to eliminate this user-dependency. The latter of these methods is able to process the accelerometer values directly, although better results are obtained when features of the signal are processed, as done in the former method. Here the signal energy, mean and standard deviation are used to segment activities and by adding all the axial time series together, the Signal-To-Noise ration is increased, resulting in a robuster method.

The method of Guenterberg *et al.* extracts features from the raw sensor signal to base the segmentation on other properties than the pure values. The method of Bernecker *et al.* [12] uses other statistical properties, namely autocorrelation, to distinguish periodic from non-periodic segments. Using the SWAB method the self-similarity of a one-dimensional signal is obtained. The authors claim that only a slight modification is needed to perform the method on multi-dimensional data. After the segmentation phase, the method of Bernecker *et al.* extracts other statistical features which are used in the classification phase.

The proposal of Guo *et al.* [27] dynamically determines which features should be used for the segmentation and simultaneously determines the best model to fit the segment. For each of the three dimensions features such as the mean, variance, covariance, correlation, energy and entropy are calculated. By extending the SWAB method, for every frame a feature set is selected, using an enhanced version of Principal Component Analysis (PCA). The research also considered the (Stepwise) Feasable Space Window as introduced by [47], but since it results in a higher error rate than SWAB the latter was chosen to extend. Whereas the before mentioned algorithms use a linear representation, this

methods considers linear, quadratic and cubical representations for each segment. This differs from other methods where the model is fixed for the whole time series, such as [23], which is stated to perform inferior on non-stationary time series such as daily life.

The time series data from a sensor can be considered as being drawn from a certain stochastic process. Probabilistic models can be constructed on that signal, yielding in probabilistic and Bayesian based segmentation methods. The CUSUM-methods takes a statistical approach and relies on the log-likelihood ratio [29] to measure the difference between two distributions. To calculate the ratio, the probability density functions need to be calculated. The method of Kawahara *et al.* [38] proposes to estimate the ratio of probability densities (known as the *importance*), based on the log likelihood of test samples, directly, without explicit estimation of the densities. The method by Liu *et al.* [46] uses a comparable dissimilarity measure using the KLIEP algorithm. They claim this results in a robuster approach for real-world scenarios. Although this is a model-based method, no strong assumptions (parameter settings) are made on the models.

The method of Adams and MacKay [1] builds a probabilistic model on the segment run length, given the observed data so far. Instead of modeling the values of the data points as a probabilistic distribution, the length of segments as a function if time is modeled by calculating the posterior probability. It uses a prior estimate for the run length and a predictive distribution for newly-observed data, given the data since the last change point. This method contrasts with the approach of Guralnik and Srivastava [28] in which change points are detected by a change in the (parameters of an) underlying, observed, model. For each new data point, the likelihoods of being a change point and part of the current segment are calculated, without a prior model (and thus is a non-Bayesian approach). It is observed that when no change point is detected for a long period of time, the computational complexity increases significantly.

Another application of PCA is to characterize the data by determining the dimensionality of a sequence of data points. The proposed method of Berbič *et al.* [8] determines the number of dimensions (features) needed to approximate a sequence within a specified error. With the observation that more dimensions are needed to keep the error below the threshold when transitions between actions occur, cut-points can be located and segments will be created. The superior extension of their approach uses a Probabilistic PCA algorithm to incorporate the dimensions outside the selected set as noise.

In the method by Himberg *et al.* [33] a cost function is defined over segments of data which is to be minimized. The cost functions thereby searches for internally homogeneous segments of data, reflecting states in which the devices and the user are. The cost function can be any arbitrary function and in the implementation the sum of variances over the segments is used. Both in a local and global iterative replacement procedure (as

an alternative for the computationally hard dynamic programming algorithm) the best breakpoint locations $c_i$ for a pre-defined number of segments $1 \leq i \leq k$ are optimized.

Many methods obtain an implicit segmentation as a result of classification over a sliding window ****** TODO: add refs ******. The method of Yang *et al.* [76] explicitly performs segmentation and classification simultaneously. It argues that the classification of a pre-segmented test-sequences becomes straightforward with many classical algorithms to choose from. The algorithm matches test examples with the *sparsest* linear representation of mixture subspace models of training examples, searching over different temporal resolutions.

The method of Chamroukhi *et al.* [16] is based on a Hidden Markov Model (HMM) and logistic regression. It assumes a $K$-state hidden process with a (hidden) state sequence, each state providing the parameters (amongst which the order) for a polynomial. The order of the model segment is determined by model selecting, often using the Bayesian Information Criterion (BIC) or the similar Akaike Information Criterion (AIC) [3], as in [31].

Field of computer vision: [79], [44].

— Segmentation —

"Segmentation and Recognition of Motion Streams by Similarity Search" [44]. 29, 2007

"Aligned Cluster Analysis for Temporal Segmentation of Human Motion" [79]. 63, 2008

### 2.5.2 Change detection

****** TODO: change order of this and temporal segmentation sections? So first change, then segmentation? ******
Whereas the above mentioned researches focus on *segmentation*, many have focused on *change detection*. Although these techniques are closely related, there is a subtle difference. In the case of *change detection* to goal is to find, possibly unrelated, sudden change points in a signal [65]. In contrast, the goal of *temporal segmentation* is to find homogeneous segments of data, which can be the result of multiple detected changes.

The ICSS by Inclán and Tiao [36] is a statistical method which obtains results (when applied to stock data) comparable to Maximum Likelihood Estimates (MLE) and Bayesian ****** TODO: Bayesian What? ******. Whereas CUSUM can be applied to search for a change in mean, the ICSS is adapted to find changes in variance. It obtains a *likelihood*

*ratio* for testing the hypothesis of one change against no change in the variance. Using an iterative approach, all possible change points are considered. The proposal of [18] extends on the CUSUM-based methods to find change points in mean and variance, by creating a more efficient and accurate algorithm.

Section 3.2 discusses the problem of change detection in time series further, and gives a formal problem definition.

****** TODO: move CUSUM based techniques to this subsection ******

— Change detection —

"A unifying framework for detecting outliers and change points from time series" [65]. 87, 2006

"Sequential change-point detection based on direct density-ratio estimation" [39]. 22, 2012

"Change point detection in time series data using support vectors" [14]. 3, 2010

### 2.5.3 Novelty detection

— Novelty detection —

"Online novelty detection on temporal sequences" [48]. 146, 2003
"Time-series novelty detection using one-class support vector machines" [49]. 78, 2003

"Novelty detection: a reviewpart 1: statistical approaches" [50]. 697, 2003

"Support Vector Method for Novelty Detection" [60]. 337, 1999

### 2.5.4 Outlier detection

— Outlier detection — "A unifying framework for detecting outliers and change points from time series" [65]. 87, 2006

"Outliers in statistical data" [9] (book). 3745, 1994

"A survey of outlier detection methodologies" [34]. 791, 2004

"Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection" [61]. 2, 2012

## 2.6 Support Vector Machines in Change Detection

Many proposals in the field of remote sensing and HAR make use of SVMs as a (supervised) model learning method. An elaborate overview of applications is given in [52]. In this section we review methods of change detection based on the applications of SVMs as model construction method. A number of proposals have been made using this method. Schölkopf *et al.* [60] applies Vapnik's principle, to never solve a problem which is more general than the one that one is actually interested in, to novelty detection. In the case of novelty detection, they argue there is no need for density estimation of the distribution. Simple algorithms estimate the density by considering how many data points fall in a region of interest. The $\nu$-SVM method instead starts with the number of data points that should be within the region and estimates a region with that desired property. It builds on the method of Vapnik and Vladimir [72], which characterizes a set of data by separating it from the origin. The $\nu$-SVM method adds the kernel method, allowing non-linear decision functions, and incorporates 'softness' by the $\nu$-parameter. Whereas the method in [72] focuses on two-class problems, the $\nu$-SVM method introduces the method the one-class problems.

The method introduced by Ma and Perkins [49] creates a *projected phase* of a time series data, which is intuitively equal to applying a high-pass filter to the time series. The projected phase of the time series combines a history or data points to a vector, which are then classified by the $\nu$-SVM method. The method is applied to a simple synthetic sinusoidal signal with small additional noise and a small segment with large additional noise. The algorithm is able to detect that segment, without false alarms.

The algorithms of SVMs has been applied to the problem of HAR, as by Anguita *et al.* [5]. In that research a multi-class classification problem is solved using SVMs and the

One-Vs-All method. The method exploits fixed-point arithmetic to create a hardware-friendly algorithm, which can be executed on a smartphone.[1]

With the same concepts as $\nu$-SVM, the SVDD method by Tax and Duin [68, 70] uses a separating hypersphere (in contrast with a hyperplane) to characterize the data. The data points that lie outside of the created hypersphere are considered to be outliers, which number is a pre-determined fraction of the total number of data points.

The method by Yin and Yang [77] uses the SVDD method in the first phase of a two-phase algorithm to filter commonly available normal activities. The Support Vector based Change Detection (SVCPD) method of Camci [14] uses SVDD applies it to time-series data. By using an One-Class Support Vector Machine (OC-SVM) the method is able to detect changes in mean variance. Especially the detection of variance *decrease* is an improvement over other methods that rely in the Kullback-Leibler (KL) divergence, since these are unable to detect decreases in variance [65]. This main research subject of this thesis is to apply the method of Camci [14] to sensor data (such as accelerometer time series) obtained by on-body smartphones.

The inner workings of OC-SVMs are discussed in detail in Section 3.4.

****** TODO: Add notion of 'supervised' vs 'unsupervised' learning. OC-SVM is unsupervised. ******

[1] Smartphones have limited resources and thus require energy-efficient algorithms. In [6] a Hardware-Friendly SVM is introduced. It uses less memory, processor time and power consumption, with a loss of precision.

# Chapter 3

# Change detection by Support Vector Machines

## 3.1 Outline

*Not intented for the reader.*

- Content follows blogpost, but more formal
- Explain the connection between change detection and {outlier detection, density estimation, etc. }
- Explain change detection in relation with "M-Estimators".
- Explain why and how SVM can be used for outlier detection, and thus for change detection in time series
- Explain options when using SVM, such as RBF kernel
- Relate characteristics of (accelerometer) data to the used (RBF?) kernel
- Define some quality metrics –¿ this should be in a different chapter (About experiments)
- Note: try to avoid the perspective of *density estimation*. Instead: *data description* or *boundary description*.

This chapter discusses the concepts and algorithms that will be used as a basis for the proposed method, as discussed in Chapter 4. The first sections formulates the problem of change detection and relates it to outlier and novelty detection. It transforms the

traditional problem to change detection for times series data. In Section 3.3 the problem of One-Class Classification is discussed and shows a number of implementations. Two Support Vector Machine-based methods, $\nu$-SVM and SVDD, are further disussed in section 3.4. For the application to the problem of Human Activity Recognition, the characteristics of the collected sensor data is discussed in 3.5. Finally a theoretic set of quality metrics is discussed in Section 3.6.

## 3.2 Change Detection in Time Series

*** The section should relate the problem of outlier detection to change detection.
It should transform the traditional problem formulation to that of time series data. Build to one-class classification, which can be used for outlier detection. ***

### 3.2.1 Relation to outlier detection

The problems of outlier and novelty detection, segmentation, and change detection are closely related. The terminology depends on the field of application, but there are subtle differences. The problem of outlier detection is concerned with finding data objects in a data set which have small resemblance with the majority of the data objects. These objects can be regarded as erroneous measurements. In the case of novelty detection these objects are considered to be member of a new class of objects. The unifying framework of Takeuchi and Yamanishi [65], changeFinder, creates a two stage process expressing change detection in terms of outlier detection. The first stage determines the outliers in a time series by giving a score based on the deviation from a learned model, and thereby creates a new time series. The second stage runs on that new created time series and calculates a average over a window of the outlier scores. The problem of change detection is then reduced to outlier detection over that average-scored time series. The implementation by Camci [14], SVCPD, implements outlier detection with the SVDD algorithm to detect changes in mean and variance.

### 3.2.2 Problem formulation

****** TODO: Give the problem of change definition. Follow definition of Camci [14] ******

****** TODO: Note: the following block is copied from 4.6 ******
The problem of change point detection can be formulated using different type of models, as discussed in 2.5.1. The methods by Takeuchi and Yamanishi [65] and Camci [14] use

the following formulation for change detection, which we will also use for our problem formulation. The algorithm searches for *sudden* changes in the time series data. In other words, slowly changing properties in the data are not considered to be changes. Considered a time series $x_1x_1\ldots$, which is drawn from a stochastic process $p$. Each $x_t$ (t = 1, 2, ...) is a $d$-dimensional real valued vector and $p$ a probability density function of the sequence $x_1x_2\ldots$. Assume $p$ can be decomposed in two different independently and identically distributed (i.i.d.) stationary stochastic processes $p^1$ and $p^2$ and are one-dimensional Gaussian density functions. For a time point $a$ data points for which $t < a$ are drawn from $p^1 = N(\mu_1, \sigma_1^2)$ and for $t \geq a$ from $p^2 = N(\mu_2, \sigma_a^2)$. If $p^1$ and $p^2$ are different, then the time point $t = a$ is a *change point*. In [65] the similarity between the stochastic processes are expressed by the KL divergence $D(p^2||p^1)$. The problem with this measure is that, as the authors conclude and Camci discusses, it is not able to detect a change by decrease in variance.

The definition of change point being *sudden* changes in the time series data is in line with the search of changes in activities. Since we are only interested in different activities (which are represented by sudden changes), slight changes within an activity are not of interest. Section 3.5 discusses the representation of activities and changes between them in the data in more detail.

****** TODO: add more formal definition or analysis of change detection ******

– Literature –

"Online segmentation of time series based on polynomial least-squares approximations", [23]. 25, 2010

## 3.3 One-Class Classification

****** TODO: bridge that explains why OCC is interesting, relating it to outlier/change detection ******

### 3.3.1 Problem formulation

****** TODO: Introduce problem of OCC. Relate to two-class classification. ******

The problem of OCC is closely related to the (traditional) two-class classification situation[1]. In the case of traditional classification algorithms, the problem is to classify an unknown object to one of the pre-defined categories. Every object $i$ is represented as a vector $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d}), x_{i,j} \in \mathbb{R}$ with $d$ real-valued measurements. Using this notation, an object $\boldsymbol{x}_i$ thus represents one point in a feature space: $\boldsymbol{x}_i \in \mathcal{X} = \mathbb{R}^d$. The two classes of objects, $\omega_1$ and $\omega_2$, are labeled $-1$ and $+1$ respectively. The objects with a label $y_i \in \{-1, +1\}$ attached are in the training set $\mathcal{X}^{tr}$ (note that it can be both positive and negative example objects). This problem is solved by determining a decision boundary in the feature space of the data objects and label the new data object based on the location relative to this boundary. This is expressed by the decision function $y = f(\boldsymbol{x})$:

$$f : \mathbb{R}^d \to \{-1, +1\} \tag{3.1}$$

In case of the OCC problem, only one class (often referred as the target class, or positive examples) of training data is used to create a decision boundary. The goal is to determine whether a new data object belongs to the target class or to the unknown class and thus is an outlier. One could argue that this problem is equal to the traditional two-class problem by considering all other classes as negative examples, although there are important differences. In pure OCC problems there are no negative example objects available. This could be because the acquisition of these examples is very expensive, or because there are only examples of the 'normal' state of the system and the goal is to detect 'abnormal' states. Since the algorithm's goal is to differentiate between normal and abnormal objects (relative to the training objects), OCC is often called outlier, novelty or anomaly detection, depending on the origin of the problem to which the algorithm is applied[2]. The difference between two and one-class classification and the consequence for outlier objects is illustrated in Figure 3.1. In the two-class classification problem the object **o** will be member of the **-1** class whilst the OCC problem will label it as an outlier. In [67] a more detailed analysis of the OCC is given.

The OCC algorithms have been applied to a wide range of applications. The first is, obviously, outlier detection of objects which do not resemble the bulk of the training data. It can be a goal by itself and can be used as a filtering mechanism for other data processing methods. Often methods that rely on data-characteristics are sensitive for remote regions in the data set. Using OCC these remote regions can be removed from the data set. A second application is for the problem as described above, in which only data from a single target class is available. When this problems originates from e.g. a monitoring process, the OCC is able to recognize abnormal system states, without

[1] Two-class problems are considered as to be the basic problem, since multi-class classification problems can be decomposed into multiple two-class problems [24].

[2] The term One-Class Classification originates from Moya *et al.* [53].

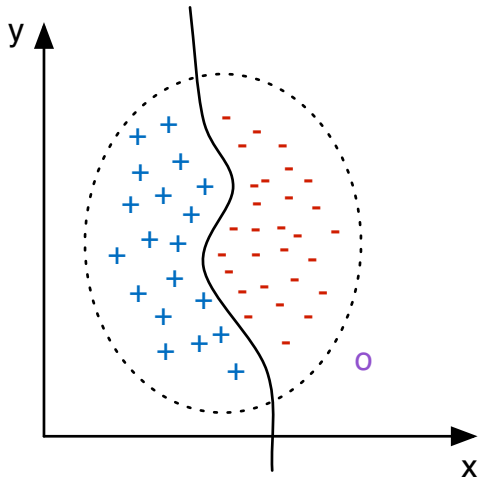


FIGURE 3.1: This plot shows the difference between two and one-class classification. The solid line indicates a possible decision boundary between the **+1** and **-1** example objects. The dashed circle indicates the closed boundary around all the data objects. In the first type the object **o** is considered to be member of the **-1**-class, whilst in the latter (OCC) formulation it is an outlier.

the need to create (or simulate) faulty states beforehand. The final possible application given by Tax [67] is the comparison of two data sets. By constructing a OCC-classifier using a training data set, one can compare that set to a new data set. This will result in a similarity-measure. It is easy to see here a relation with similiary measures over PDFs and the KL divergence.

****** TODO: Add reference to 3.2.2 which also mentions KL divergence? ******

### 3.3.2 One-Class Classification methods

The methods and algorithms used for the OCC-problem can be organized into three categories [55, 67], visually represented in Figure 3.2. The first category consists of methods that estimate the density of the training data and set a threshold on this density. Among those are Gaussian models, Gaussian Mixture Models (GMMs) and Parzen density estimators. In order to get good generalization results with these methods, the dimensionality of the data and the complexity of the density need to be restricted. This can cause a large bias on the data. When a good probability model is assumed, these methods work very well, since when one threshold is optimized, a minimum volume is automatically found for the given probability density model [67].

Boundary methods are based on Vapnik's principle[3] which imply in this case that estimating the complete data density for a OCC may be too complex, in case one is only

[3]With a limited amount of data available, on should avoid solving a more general problem as an intermediate step to solve the original problem [73].

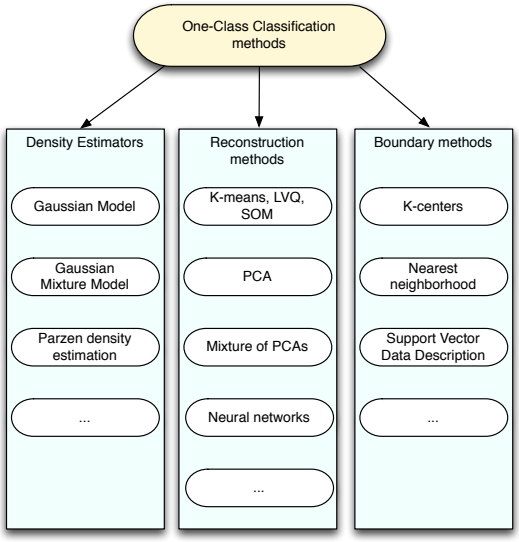


FIGURE 3.2: Overview of OCC methods categorized in Density Estimators, Reconstruction methods and Boundary methods. This categorization follows the definition of Tax in [67].

interested in the closed boundary. Examples of methods that focus on the boundary (a direct threshold) of the training data distribution are K-centers, Nearest-neighborhood and SVDD or a combination of those methods [32]. Especially the latter has a strong bias towards minimal volume solutions. These type of methods are sensitive to scaling of features, since they rely on a well-defined distance measure. The number of objects that is required is smaller than in the case of density methods. The boundary method SVDD, constructed by Tax and which has shown good performance [41], will be further discussed in Section 3.4.6.

Reconstruction methods take a different approach. Instead of focusing on classification of the data and thus on the discriminating features of data objects, they model the data. This results in a compact representation of the target data and for any object a reconstruction error can be defined. Outliers are data objects with a high reconstruction error, since they are worse represented by the constructed model. Examples of reconstruction methods are K-means, PCA and different kind of neural network implementations.

In [41, 55] an overview of applications for OCC-algorithms, and explictly for SVM-based methods (such as SVDD and $\nu$-SVM), is given. It shows succesful applications for, amonst others, problems in the field of Handwritten Digit Recognition, Face Recognition Applications, Spam Detection and Anomaly Detection [45, 57]. As discussed in Section 3.2, this can be used for change detection in time series data.

In this Section we have discussed the problem of OCC and different kind of implementations. An often used implementation is the SVM-based method [55], since is shows

good performance in comparitive researches. ****** TODO: Add refs for good results of OC-SVM ****** In the following section (3.4) two implementations of OC-SVM will be discussed, the SVDD method of Tax and Duin [68] and the $\nu$-SVM-algoritm by Schölkopf.

## 3.4 One-Class Support Vector Machine

### 3.4.1 Support Vector Machine

We will first discuss the traditional two-class SVM before we consider the one-class variant, as introduced by Cortes and Vapnik in [21]. Consider a data set $\Omega = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$; points $x_i \in \mathbb{R}^d$ in a (for instance two-dimensional) space where $x_i$ is the $i$-th input data point and $y_i \in \{-1, 1\}$ is the $i$-th output pattern, indicating the class membership.

A SVM can create a boundary between linear-separable data points, making it a non-probabilistic binary linear classifier. More flexible non-linear boundaries can be obtained by the use of a non-linear function $\phi(x)$, as illustrated in Figure 3.5. This function maps the input data from space $\mathcal{I}$ to a higher dimensional space $\mathcal{F}$. The SVM can create a linear separating hyperplane in the space $\mathcal{F}$ that separates the data points from the two classes. When the hyperplane is projected to the (lower) original input space $\mathcal{I}$ it creates a non-linear separating curve. This is illustrated in Figure 3.3. The mapping and projection of data points can be efficient (and implicit) performed by using the kernel trick, which is discussed in section 3.4.2.

The separating hyperplane is represented by

$$w^T x + b = 0, \tag{3.2}$$

with $w \in F$ and $b \in R$. The hyperplane that is created determines the *margin* between the classes; the minimal distance from one of the data points to the hyperplane. In geometric sense, $w$ is the normal vector indication the direction of the hyperplane and $\frac{b}{\|w\|}$ determines the offset of the hyperplane to the origin. Since the distance between the two margins is equal to $\frac{2}{\|w\|}$, the maximum-margin hyperplane is found by minimizing $\|w\|$. The data points which lie on the margin are the *support vectors*. This geometrical interpretation is illustrated in Figure 3.4. All data points for which $y_i = -1$ are on one side of the hyperplane and all other data points (for which $y_i = 1$) are on the other side. The minimal distance from a data point to the hyperplane is for both classes equal.

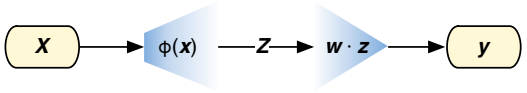


FIGURE 3.3: The input vector $\boldsymbol{x}$ from input space $\mathcal{I}$ is mapped by a non-linear function $\phi(\boldsymbol{x})$ to a feature vector $\boldsymbol{z}$ in the high dimensional feature space $\mathcal{F}$. The weights of $\boldsymbol{w}$ create a linear separating hypeplace, which maps the high dimensional vector to the predicted outcome $\boldsymbol{y}$. **** TODO: Is this figure obsolete? Figure 3.5 shows almost the same ******

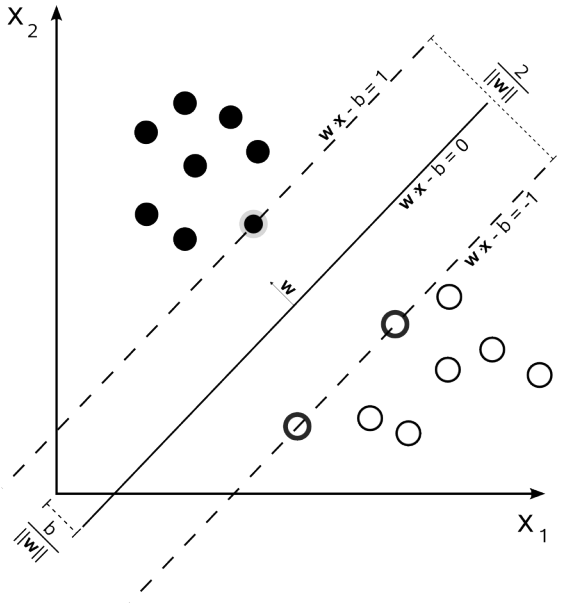


FIGURE 3.4: Illustration of the separating hyperplane of a SVM. Here $w$ is the normal vector for the separating hyperplane and the distance between the two margins is $\frac{2}{\|w\|}$. Image from Wikipedia.org[4]

This results in a *maximal margin* between the two classes. Thus, the SVM searches for a maximal separating hyperplane.

With every classification method there is a risk of overfitting. In that case the random error or noise of the data set is described instead of the underlying data. The SVM classifier can use a *soft margin* by allowing some data points to lie within the margin, instead of on the margin or farther away from the hyperplane. For this it introduces *slack variables* $\xi_i$ for each data point and the constant $C > 0$ the determines the trade-off between maximizing the margin and the number of data points within that margin (and thus the training errors). The slack variables minimize the *sum of deviations* rather than the *number* of incorrect data points [19]. The objective function for a SVM is the following minimization function:

[4] http://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

$$\min_{w,\ b,\ \xi_i} \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i \tag{3.3}$$

subject to:

$$\begin{aligned} y_i(w^T\phi(x_i) + b) &\geq 1 - \xi_i \quad \text{for all } i = 1, \ldots, n \\ \xi_i &\geq 0 \quad \text{for all } i = 1, \ldots, n \end{aligned} \tag{3.4}$$

****** TODO: better format of above formulas. Check with notation and formulas of chapter 9 of [19]. ******

This minimization problem can be solved (using Quadratic Programming (QP)) and transformed to its (Lagrange) dual formulation. In the dual formulation the problem scales with the number of training examples $n$ instead of the dimensionality $d$ of the samples. Solving this problem directly in the high dimensional feature space $\mathcal{F}$ makes it untractable. The linear approximation function corresponds to the kernel function in the dual formulation. Solving this dual formulation is equivalent to solving the primal formulation [19]. In the dual formulation the Lagrange multipliers $a_i >= 0$ are introduced and the decision function becomes:

$$f(x) = \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b), \tag{3.5}$$

where $K(x, x_i) = \phi(x)^T\phi(x_i)$ (which is further discussed in section 3.4.2). Here every data point $\mathcal{I}$ for which $a_i > 0$ is weighted in the decision function and thus "supports" the classification machine: hence the name "Support Vector Machine". Since it is shown that under certain circumstances SVMs show an equality to sparse representations [25, 62], there will often be relatively few Lagrange multipliers with a non-zero value.

****** TODO: Add more on sparsity and good results, because of sparse representations, even in case of data of high dimensionality. e.g. [19]. ******

### 3.4.2 Kernel trick

In the previous section, 3.4.1, the mapping function $\phi(x)$ and the kernel function $K$ were briefly mentioned. The decision function in equation 3.5 only relies on the dot-products of mapped data points in the feature space $\mathcal{F}$ (i.e. all pairwise distances between the data points in that space). It shows [22] that as long as any function has the same result, without an explicit mapping to the higher dimension $\mathcal{F}$, the dot-products can be substituted by the kernel function $K$. This is known as the *kernel trick* and gives

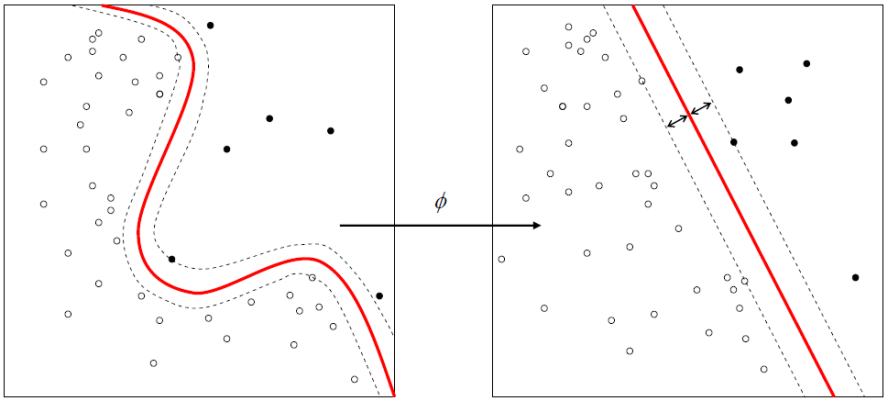


FIGURE 3.5: The non-linear boundary in the input space $\mathcal{I}$ (left) is transformed to a linear boundary in the feature space $\mathcal{F}$ (right) by mapping the data points with the function $\phi$. The kernel trick uses a function $K$ which performs an implicit mapping. Image from Wikipedia.org[5]

the SVM the ability to create non-linear decision function without high computational complexity. This mapping is illustrated in Figure 3.5. Here the non-linear separating boundary in the input space $\mathcal{I}$ is mapped, via $\phi$, to a linear boundary in the feature space $\mathcal{F}$.

The kernel function $K$ can have different forms, such as linear, polynomial and sigmoidal but the most used (and flexible) form is the Gaussian Radial Base Function (RBF). As Smola *et al.* [62] state, this Gaussian kernel yields good performance, especially when no assumptions can be made about the data. The kernel maps input space $\mathcal{I}$ to the feature space $\mathcal{F}$ which is a Hilbert Space of infinite dimensions ****** TODO: ref needed ******:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \tag{3.6}$$

where $\sigma \in R$ is a kernel parameter and $\|x - x'\|$ is the dissimilarity measure expressed in Euclidean distance.

### 3.4.3 Reproducing kernel Hilbert space

*Write something on this topic? Or leave it implicit?*

### 3.4.4 One-Class Support Vector Machines

The classical implementation of a SVM is to classify a dataset in two distinct classes. This is a common usecase, although sometimes there is no training data for both classes

[5] http://en.wikipedia.org/wiki/File:Kernel_Machine.png

available. Still, one would like to classify new data points as regular, or in-class, or out-of-class, e.g. in the case of a novelty detection. With that problem only data from one class is available and the objective is to recognize new data points that are not part of that class. This unsupervised learning problem is closely related to density estimation. In that context, the problem can be the following. Given an underlying probabilty distribution $P$ the goal is to find a subset $S$ for which the probability that a data point from $P$ lies outside $S$ equals some predetermined value $\nu$ between 0 and 1 [60].

We will discuss two implementations of OC-SVMs. The first is the $\nu$-SVM by Schölkopf *et al.* [60] and closely follows the above problem statement regarding density estimation. The second, on which we will base our change detection method, is the SVDD by Tax and Duin [68].

****** TODO: Add something about robustness, discusses in section 9.2 from [17, 19] ******

### 3.4.5 $\nu$-Support Vector Machine

The first of the OC-SVM methods we will discuss is often referred to as $\nu$-SVM and introduced by Schölkopf *et al.* [60]. Instead of estimating the density function of an distribution $P$, it focuses on an easier problem: the algorithm find regions in the input where the "probability density lives". This results in a function such that most of the data is in the region where the function is nonzero.

The constructed decision function $\mathcal{F}$ resembles the function discussed in Section 3.4.1. It returns the value $+1$ in a (possibly small) region capturing most of the data points, and $-1$ elsewhere. The method maps the data points from input space $\mathcal{I}$ to a feature space $\mathcal{F}$ (following classical SVMs). In that space $\mathcal{F}$ it separates the data points with maximal margin from the origin, with a hyperplane. For a new data points $x$, the function value $f(x)$ determines wheter the data point is part of the distribution (i.e. the value is $+1$) or a novelty (i.e. the value is $-1$). The hyperplane is represented by $g(x) = w \cdot \phi(x) + \rho = 0$ and the decision function is $f(x) = \operatorname{sgn}(g(x))$. This hyperplane and the separation from the origin is illustrated in Figure 3.6.

The objective function to find the separating hyperplane is the following minimization function, which can be solved using QP:

$$\min_{w,\ \xi_i,\ \rho} \frac{\|w\|^2}{2} + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho \tag{3.7}$$

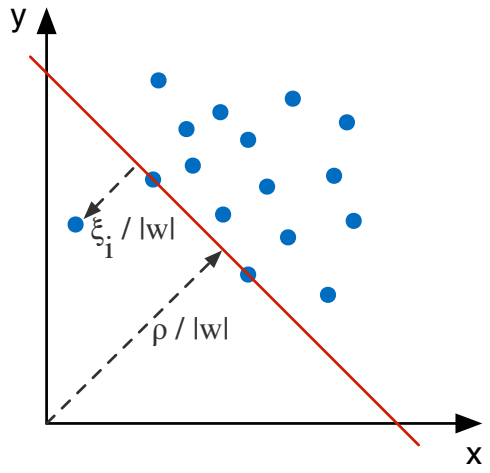


FIGURE 3.6: Graphical representation of $\nu$-SVM. The separating hyperplane $w \cdot \phi(x_i) + \rho = 0$ create a maximal margin, in the feature space, between the data points and the origin. Slack variables $\xi_i$ are used to create a soft margin.

$$\begin{aligned} &\text{subject to:} \\ &(w \cdot \phi(x_i)) \geq \rho - \xi_i \quad \text{for all } i = 1, \ldots, n \\ &\xi_i \geq 0 \quad \text{for all } i = 1, \ldots, n \end{aligned} \tag{3.8}$$

The decision function in the dual formulation with Lagrange multipliers is denoted as:

$$f(x) = \text{sgn}((w \cdot \phi(x_i)) - \rho) = \text{sgn}(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho) \tag{3.9}$$

In the classical SVM objective function, as denoted in Equation 3.3, the parameter $C$ decided the smoothness of the boundary, with respect to the slack variables $\xi_i$. In the formulation of $\nu$-SVM the equivalent parameter is $\nu \in (0, 1)$ (hence the name). It characterizes the solution in two ways:

1. $\nu$ is an upper bound on the fraction of outliers, i.e. training examples regarded as out-of-class.
2. $\nu$ is a lower bound on the fraction of Support Vectors (SVs), i.e. training examples with a nonzero Lagrange multiplier $\alpha_i$.

When $\nu$ approaches 0, the penalty factor for nonzero Lagrange multipliers ($\frac{1}{\nu n}$) becomes infinite, and thus the solution resembles a *hard margin* boundary.

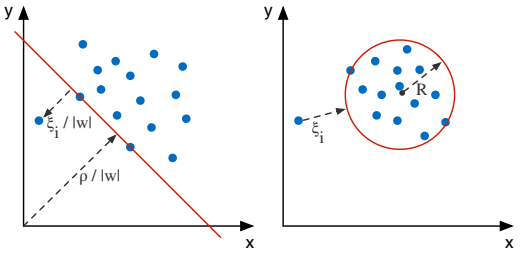


FIGURE 3.7: Graphical representation of the difference between $\nu$-SVM (left) and SVDD (right). Note that for the sake of simplicity the kernel functions are not applied.

This method creates a *hyperplane*, characterized by $w$ and $\rho$, that separates the data with maximal margin from the origin in the feature space $\mathcal{F}$. In the following section we will discuss an alternative method, which uses an circumscribing *hypersphere* to characterize the data (and the region of distribution $P$ where the density (or: support) lives).

### 3.4.6 Support Vector Data Description

The method introduced by Tax and Duin [68], known as Support Vector Data Description, takes a spherical instead of planar approach. The boundary, created in feature space $\mathcal{F}$, forms a hypersphere around the (high density region of the) data. The volume of this hypersphere is minimized to get the smallest enclosing boundary. The chance of accepting outlier objects is thereby also minimized [71]. By allowing outliers using slacks variables, in the same manner as classical SVM and $\nu$-SVM, a soft margin is constructed.

The constructed hypersphere is characterized by a center $\mathbf{a}$ and a radius $R > 0$ as distance from the center to (any data point that is a SV on) the boundary, for which the volume, and thus the radius $R$, will be minimized. The center $\mathbf{a}$ is a linear combination of of the support vectors. Like the classical SVM and SVDD it can be required that all the distances from the data points $x_i$ to the center $\mathbf{a}$ are strict less then $R$ (or equivalent measure, to create a hard margin) or soft margins are allowed by using slack variables $\xi_i$. In the case of a soft margin, the penalty is determined by $C$ and the minimization is expressed as Equation 3.10. This principle is illustrated in the right image or Figure 3.7. Instead of a separating hyperplane, constructed by $\nu$-SVM and illustrated on the left of the Figure, the SVDD creates a hypersphere (in the illustration a cricle) around the data points. By using kernel functions (e.g. the RBF) the hyperspheres in the high dimensional feature space $\mathcal{F}$ corresponds to a flexible and tight enclosing boundary in input space $\mathcal{I}$. Possible resulting closed boundaries are illustrated in Figure 3.8. This enclosing boundary is obtained by minimizing the following error function $L$ which contains the volume of the hypersphere and the distance from the boundary to the

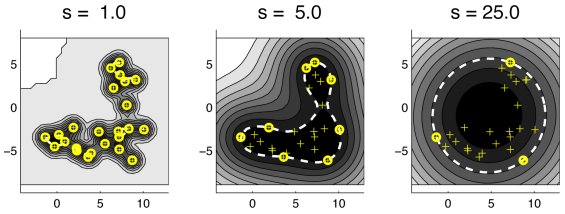


FIGURE 3.8: The SVDD method trained on a banana-shaped data set with different sigma-values for the RBF kernel. Solid circles are support vectors, the dashed line is the boundary. Image by Tax [67].

outlier objects:

$$L(R, \boldsymbol{a}, \boldsymbol{\xi}) = R^2 + C \sum_{i=1}^{n} \xi_i \tag{3.10}$$

subject to:

$$\begin{aligned} \|x_i - \mathbf{a}\|^2 \leq & R^2 + \xi_i \quad \text{for all } i = 1, \ldots, n \\ \xi_i \geq & 0 \quad \text{for all } i = 1, \ldots, n \end{aligned} \tag{3.11}$$

In the dual Lagrangian formulation of this error function $L$ the multipliers $\boldsymbol{\alpha}$ are maximized:

$$L = \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \tag{3.12}$$

subject to:

$$0 \leq \alpha_i \leq C, \sum_i \alpha_i = 1 \tag{3.13}$$

In the maximization of Equation 3.12 a large fraction of the multipliers $\alpha_i$ become zero and for a small fraction $\alpha_i > 0$. This small fraction, for which $\alpha_i$ is non-zero, are called the SVs and these objects lie on the boundary of the description. The center of the hypersphere only depends on this small number of SVs and the objects for which $\alpha_i = 0$ can be disregarded from the solution. Testing the membership of a (new) object $\boldsymbol{z}$ is done by determining if the distance to the center $\boldsymbol{a}$ of the sphere is equal or smaller to the radius $R$:

$$\|\boldsymbol{z} - \boldsymbol{a}\|^2 = (\boldsymbol{z} \cdot \boldsymbol{z}) - 2 \sum_i \alpha_i (\boldsymbol{z} \cdot \boldsymbol{x}_i) + \sum_{i,j} (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \leq R^2 \tag{3.14}$$

****** TODO: Note of inner products and that it can be replaced by kernel functions. Good ref: page 4/158 from [69] and [68]. ******

****** TODO: Note that SVDD and $\nu$-SVM have identical solutions when data is preprocessed to have unit form (in the case of $\nu$-SVM) [69]. ******

### 3.4.7 Kernel Function

- Explain RBF kernel
- Explain RKHS? (Hilbert Space)

A common class for kernel function is the Radial Base Function. These function are of the form

$$g(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i \exp\left\{\frac{|\boldsymbol{x} - \boldsymbol{x}_i|^2}{\sigma^2}\right\}\right), \tag{3.15}$$

where $\sigma$ defines the width. The inner product kernel is of the form

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\frac{|\boldsymbol{x} - \boldsymbol{x}'|^2}{\sigma^2}\right\}. \tag{3.16}$$

Note that with the RBF class of kernel functions the number of basis functions, the center parameters that correspond to the support vectors, and the weights in the output layer are all automatically determined via the optimal hyperplane [19]. The width parameter $\sigma$ is equal for all basis functions and is set a priori and determines the flexibility and complexity of the boundary. In Section 3.4.8 this (hyper)parameter for a SVM is further discussed. Other examples of kernel function classes are *pylonomial functions* of degree $q$, *neural networks*, *splines* of order $m$ with $b$ nodes, and *Fourier expansions*. ****** TODO: More on basis functions. [19], chapter 9.4 ******

– Quotes and refs to use – $k(\mathbf{x}_i, \mathbf{x})$ is a kernel function computing a dot product in feature space, introduced by Aizerman *et al.* [2]. "Training a SVM with Gaussian RBF kernels corresponds to minimizing the specific cost function with a regularization operator" [62].
"Gaussian kernels tend to yield good performance under general smoothness assumptions and should be considered especially of no additional knowledge of the data is available." [62].
"Choosing a small width of the kernels leads to high generalization error as it effectively decouples the separate basis functions of the kernel expansion into very localized functions which is equivalent to memorizing the data, whereas a wide kernel tends to oversmooth." [62].

sparsity, RKHS: [25]

### 3.4.8 SVM model parameters

SVM-model selectiong and tuning depends on two type of parameters [19]:

1. Parameters controlling the 'margin' size,
2. Model parameterization, e.g. the kernel type and complexity parameters. For the RBF kernel the width parameter determines the model complexity.

In case of a RBF kernel, the width parameter $\sigma$ determines the flexibility and complexity of the boundary. The value of this parameter greatly determines the outcomes of the algorithm (e.g. SVDD) as illustrated in Figure 3.8. With a small value for the kernel width $\sigma$, each data point will tend to be used as a support vector and will represent a small Gaussian. The solution then corresponds with standard non-parametric density estimation. With a large value for the width $\sigma$, the boudary approximates the spherical boundary.

****** TODO: More on SVM parameters, [19], section 9.8, page 446 ******

****** TODO: Analysis of $C/\nu$-parameter, which sets a lower bound for the fraction of support vectors and an uppor bound for samples that will be outliers. [22] section 9.8 and [58] and other Schölkopf material. ******

****** TODO: Maybe add section about "Change indication" rather than "detection"? ******

– Literature –

****** TODO: These papers need to be referenced and included ******

"A geometric approach to support vector machine (SVM) classification" [51]. 136, 2006

"Least squares one-class support vector machine" [20]. 27, 2009

"On simple one-class classification methods" [55]. 2012 –¿ decouples radius and center optimization, gives fast approximations instead of precise results.

## 3.5 Sensor data characteristics

****** TODO: Other title? More in Human Activity Recognition in general? ******

– Literature – "Sensor-based abnormal human-activity detection" [77]. 74, 2008 (builds one-class SVM of all normal traces)

## 3.6 Quality metrics

****** TODO:* This section should be in the chapter about experiments *****

# Chapter 4

# Proposed method

## 4.1 Outline

*Not intented for the reader.*

- Based on the problem statement with current research as stated in Chapter 3
- Adjust method to needs
- Explain using graphs, pseudo-algorithms. Make clear distinction in origin of ideas and why to apply

### 4.1.1 Experiment notes

- ICSS/CUSUM is goed in het vinden van variance changes. Niet goed in mean veranderingen.
- Problemen met SVM methode: ook na de verandering blijft de threshold nog stijgen;
- geprobeerd direct model-reset te doen na change point, maar dan kan je achteraf niet meer verder analyseren.
- Veranderingen in mean geven "extra" change point wanneer alle data weer in het window zit. Zie **??** op de 50-tallen.

**Data gathering** Explain data gathering methods. Refer to chapter 5 and 6 for articifial and real-world details.

**Model construction** Explain SVDD model construction and updating.

**Model properties** Explain the SVDD properties that are used to calculate the change indication.

**Change detection** Explain the interpretation of the properties, the possible (modulair) methods to give change indication

****** TODO: Write chapter introduction/outline. Give overview of method approach and refer to each section for details of that stage. ******

## 4.2 Data Gathering

In this section we briefly discuss the different data gathering methods used for the change detection algorithms and experiments. Section 4.2.1 reviews the artificial data sets we will use. In Section 4.2.2 an overview of the real-world data sets used is provided. Both sections refer to Chapters 5 and 6 for more details, respectively.

### 4.2.1 Artificial data

In order to provide an objective comparison to other methods, we will use artificial data sets which are also used in the researches on which our method is based. The data sets are from Takeuchi and Yamanishi [65] and Camci [14]. Both construct a collection of one-dimensional time series data according to the second order Autoregressive (AR) model:

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \epsilon_t. \tag{4.1}$$

In the different data series the mean and variance of the Gaussian random variable $\epsilon_t$ differs and changes at pre-determined change points. Using this data set an objective quality measure over the change detection methods can be obtained and compared. All the used data sets are listed and analyzed in Chapter 5.

### 4.2.2 Real-world data

In the second type of data sets we apply our method of change detection and temporal segmentation to real-world data sets. Our setup records the activities of humans performed both in and out door in an uncontrolled environment. Activities performed include walking, running in both a straight line and a curve, standing, walking up and

downstairs and sitting. Using the time series data sensor output in the form of acceleration, gravity and rotation metrics, our method gives change points based on those signals. By comparing the discovered change points with annotated change points (based on video recordings of the performed activities) we are able to give subjective results. In Chapter 6 we give a detailed analysis of the performed activities and the recorded data sets.

## 4.3 Model Construction: Incremental SVDD

After the data is collected and pre-processed (or in the case of the artificial data sets: generated), we construct an online incremental sliding window model construction algorithm. We follow the method and implementation introduced by Tax and Laskov [71], the Incremental Support Vector Data Description method. This method combines the techniques of online, unsupervised and incremental learning methods with the earlier introduced OC-SVM algorithm SVDD. The method is first initialized with a window length and then in every step a new data object is added to and the last data object is removed from the working set.

Using the following abstract form of the SVM optimization problem, the extension of the incremental SVM to the SVDD can be carried out:

$$\max_{\mu} \min_{\substack{0 \leq x \leq C \\ \boldsymbol{a}^T \boldsymbol{x} + b = 0}} : W = -\boldsymbol{c}^T \boldsymbol{x} + \frac{1}{2} \boldsymbol{x}^T K \boldsymbol{x} + \mu(\boldsymbol{a}^T \boldsymbol{x} + b), \tag{4.2}$$

where $\boldsymbol{c}$ and $\boldsymbol{a}$ are $n \times 1$ vectors, $K$ is a $n \times n$ matrix and $b$ is a scalar. The SVDD implementation of this abstract form is set by the parameters $\boldsymbol{c} = \text{diag}(K)$, $\boldsymbol{a} = \boldsymbol{y}$ and $b = 1$. The procedure for the incremental version has two operations: adding and removing a data object $k$. When a data object $k$ added, its weight $x_k$ is initially set to 0. In case of an object removal, the weight is forced to be $x_k = 0$. Both the operations conclude with the recalculation $\mu$ and the weights $\boldsymbol{x}$ for all the objects, in order to obtain the optimal solution for the enlarged or reduced data set. The incremental learning algorithm follows from these two operations: new data objects are added to and old data objects are removed from the working set.

The size of the initial window of data objects has a lower bound determined by the hyperparameter $C$ (Equation 3.10). Because of the equality constraint $\sum_{i=1}^{n} a_i x_i = 1$ and the box constraint $0 \leq x_i \leq C$, the number of objects in the working set must be at least $\lceil \frac{1}{C} \rceil$. Thus the algorithm is initialized by selection the first $\lceil \frac{1}{C} \rceil$ objects for the working set. In every step of the loop of the algorithm at least the same number

of objects must be added as there are removed. By analyzing the KarushKuhnTucker (KKT) conditions, [71] shows the optimality of the algorithm.

From experiments it shows that the online, Incremental Support Vector Data Description (I-SVDD), method results in less false alarms than the static SVDD. An explanation for this is that I-SVDD follows the changing data distribution, such that small changes over time, like a drift in mean or increase in frequency, continuously re-model the SVM representation.

## 4.4 Model Properties

## 4.5 Change Detection

— OLD —

## 4.6 Change Indication

****** TODO: change section title *** *** TODO: *** ***** *Follow methodology of "A unifying framework for detecting outliers and change points from time series" [65]. It creates a two-stage process of first searching for outliers, and then using the "outlier-score" to find (sudden) change points, by a weighted average of a moving window. That eventual score can be thresholded (as in the paper) or processed with something like CUSUM (proposal). Looks like my proposed methods, in that is combines outliers and gives a score to change points.*

The proposed method of this thesis follows the unifying framework as introduced by Takeuchi and Yamanishi [65] and an similar implementation by Camci [14] with SVMs. The unifying framework combines the detection of outliers with change points and divides it in two stages. The first stage determines the outliers in a time series by giving a score based on the deviation from a learned model, and thereby creates a new time series. The second stage runs on that new created time series and calculates a average over a window of the outlier scores. The problem of change detection is then reduced to outlier detection over that average-scored time series. This method is named changeFinder by the authors. The implementation by Camci, which uses SVMs to detect changes is named Support Vector based Change Detection.

****** TODO: the following paragraph is copied into 3.2.2. Remove here? Or shorten? ******

The problem statement and formal definition, following Takeuchi and Yamanishi [65] and Camci [14] is the following. The algorithm needs to find *sudden* changes in the time series data. In other words, slowly changing properties in the data are not considered to be changes. This is in line with the search of changes in activities, since we are only interested in different activities (which are represented by sudden changes) instead of changes within an activity. Considered a time series $x_1 x_1 \ldots$, which is drawn from a stochastic process $p$. Each $x_t$ (t = 1, 2, ...) is a $d$-dimensional real valued vector and $p$ a probability density function of the sequence $x_1 x_2 \ldots$. Assume $p$ can be decomposed in two different i.i.d. stationary stochastic processes $p^1$ and $p^2$ and are one-dimensional Gaussian density functions. For a time point $a$ data points for which $t < a$ are drawn from $p^1 = N(\mu_1, \sigma_1^2)$ and for $t \geq a$ from $p^2 = N(\mu_2, \sigma_a^2)$. If $p^1$ and $p^2$ are different, then the time point $t = a$ is a *change point*. In [65] the similarity between the stochastic processes are expressed by the KLIEP divergence $D(p^2||p^1)$. The problem with this measure is that, as the authors conclude and Camci discusses, it is not able to detect a change by decrease in variance.
****** TODO: End copied paragraph ******

Whereas changeFinder uses double probability estimation algorithm, our approach follows SVCPD by constructing a SVM over a sliding window. The SVCPD algorithm uses the location of new data points in the feature space $\mathcal{F}$ with respect to the hypersphere and the hypersphere's radius $R$ to determine whether the new data point represents a change point.

****** TODO: What is my contribution compared to Camci? Distance of outliers to hypersphere? ******

## 4.7 Overview

This section gives a description of the method used for the experiments and change detection mechanism. First described is the method to process the gathered sensor data. A schematic overview is given in figure 4.1 and shows the steps of the method. A more detailed explanation of the "Update model" step follows. This section is finalized with the experiments setup, annotation of data streams and quality measure.

### 4.7.1 Change detection method

As graphically represented in figure 4.1, the change detection method starts by processing the data from sensor, such as the accelerometer, magnetic orientation and rotation

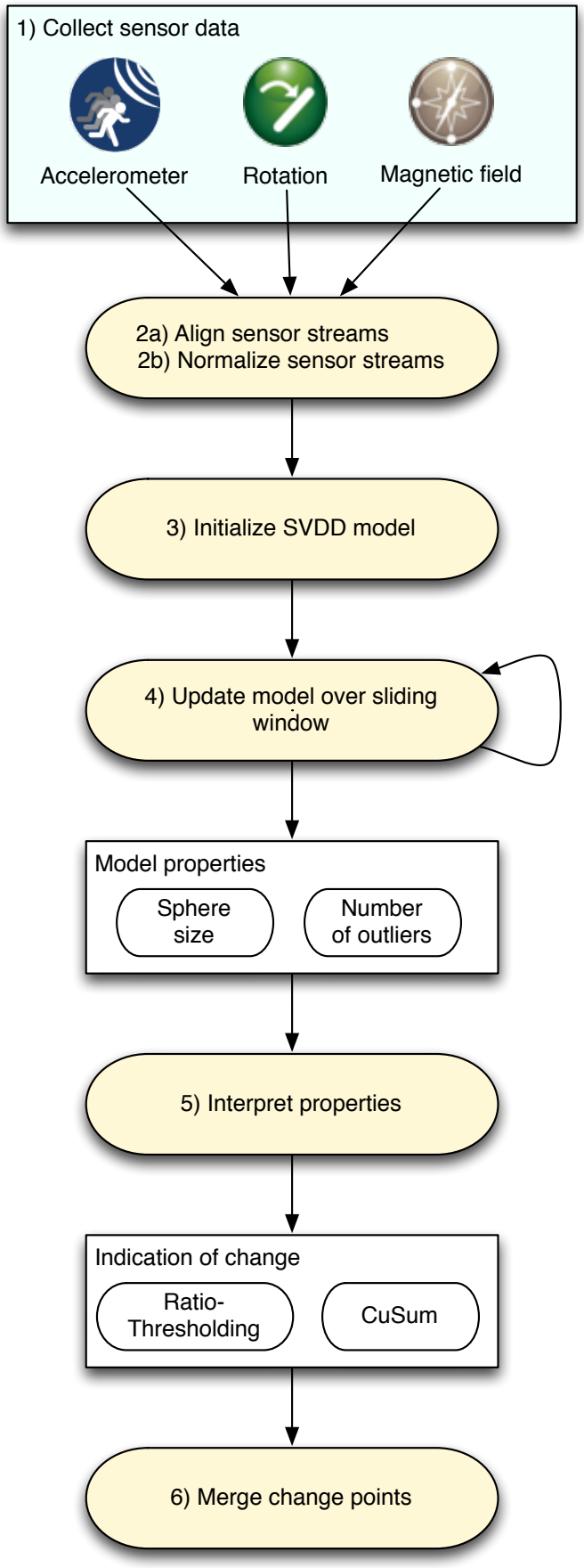


FIGURE 4.1: Schematic representation of the change detection method.

measures [1].

The first step is to process the raw streams of data originating from a multiple of sensors. The two processes applied are alignment and normalization. Due to noisy sampling, not all the timestamps in the data streams are sensed at the same timestamp. Since the SVDD method requires all the data stream at every timestamp and can not handle missing data on one of the timestamps, all the unique timestamps are filtered out. Whilst this results in an overall filtering effect, in practice between 1% and 5% of each data stream is disregarded. The effect of this filtering is not significant and the data is not modified.

Due to the nature of the sensor signals, a normalization step is required in order to set the weight for all the data streams equal. The range of the accelerometer signal typically spans $-20$ to 20, the magnetic field from $-60$ to 60 and the rotations range is from $-1$ to 1. This means that a relative small change in the accelerometer stream could have a much larger impact on the model than the same (absolute) change in the rotation stream, whilst the latter has a larger relative impact. The normalization step ensures that all data is weighted equally and changes in the data are all proportional.

In step 3 the SVDD model is initialized. The first full window over the data stream is used to construct an initial model. During the initialization the parameters for the SVDD are provided, begin the kernel type (radial), RBF with $\sigma$ and the outlier-fraction $C$.

Step 4 is executed for every step-size $s$ data points in the stream. Every update the oldest $s$ data points are removed from and $s$ new data points are added to the SVDD model. The model is (partially) reconstructed and new model properties, such as the radius of the hypersphere and the number of outliers, are the result of this step [2]. ****** TODO: MORE ON THE UPDATING STEP ******

This final step of this method, step 5, is the interpretation of the model properties. Many algorithms can be used for this process, all which take a one-dimensional time series as input and determine where change has occured. In our setup we used the Ratio-Thresholding (RT) and CUSUM methods, to show the modularity of this step.

[1] Something about the origin of the streams; all from the same sensor or different sensors?

[2] Other measures are also possible, for instance the distance from all the outliers to the boundary of the hypersphere

| Metric | Description | Units of measure | Typical range |
|---|---|---|---|
| Accelerometer | Acceleration force along each axis. | $m/s^2$ | $-20$ – 20 |
| Gravity | Force of gravity along each axi.s | $m/s^2$ | $-10$ – 10 |
| Gyroscope | | $rad/s$ | $-15$ – 15 |
| Light | Light sensitive sensor at the front of the phone. | | 0 – 10000 |
| Linear acceleration | | $m/s^2$ | $-20$ – 20 |
| Magnetic field | Geomagnetic field strength along each axis. | $\mu T$ | $-60$ – 60 |
| Orientation | | | $-100$ – 360 |
| Rotation | | | $-1$ – 1 |

TABLE 4.1: Measured metrics. The set of axis is always the triple (x, y, z) direction.

### 4.7.2 Model updating

### 4.7.3 Experiments

For the experiments we used a HTC Sensation XE smartphone as recording device. The activities were recorded using a free Android application [43]. This application was chosen for its convenient data format of the sensor recording and its regularity of the sampling interval. Table 4.1 lists the recorded metrics. For our expiments we used the data for the accelerometer, magnatief field and rotation.

## 4.8 Algorithms

### 4.8.1 Parameters

****** TODO: Describe how all the parameters are set, for the SVM methods and change-indication methods. Kernel type, width, $\nu/C$ values etc. ******

****** TODO: Explain workings of the incremental SVDD algorithm. Het is gebaseerd op [71]. ******

# Chapter 5

# Artificial data results

## 5.1 Outline

*Not intented for the reader.*

- Compare proposed method with methods of Chapter 2
- Provide KL-analysis (as Takeuchi does)
- Provide plots, tables, graphs, error rates, precision, etc.
- Apply to a multiple of data, to compare to previous research - use that data
- Give theoretical analysis about performance. Big-O, memory, run-time, precision.
- This sections needs programmed implementations of own method and the ones compared

# Chapter 6

# Real-world results

## 6.1 Outline

*Not intented for the reader.*

- Apply proposed method to real-world applications, such as
  - Daily life activity recognition (as the original context of this thesis is)
  - PowerHouse sensor data
  - Stock data?
- Relate back to filter vs. wrapper methods - give results with different methods?

## 6.2 Data Gathering

****** TODO: These plots should be in ******

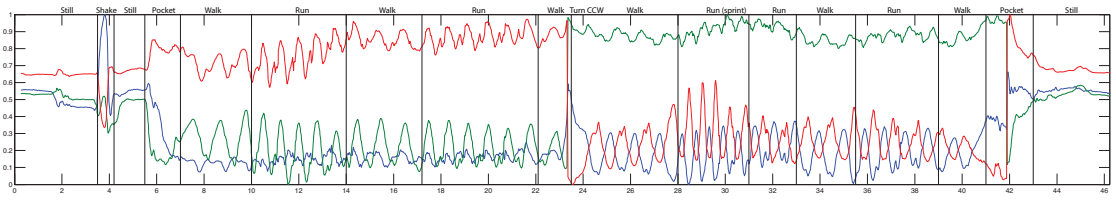


FIGURE 6.1: Run 1: Walk-run-roemer, rotation

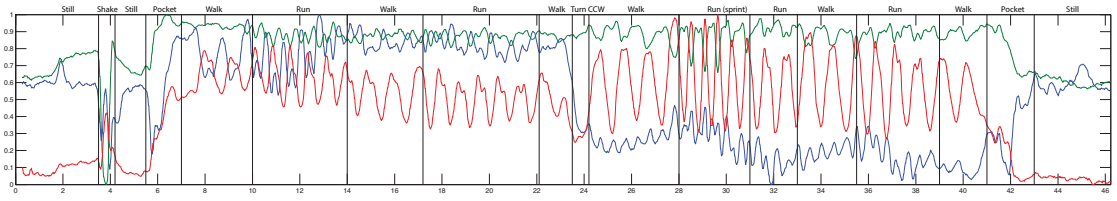


FIGURE 6.2: Run 1: Walk-run-roemer, Mag

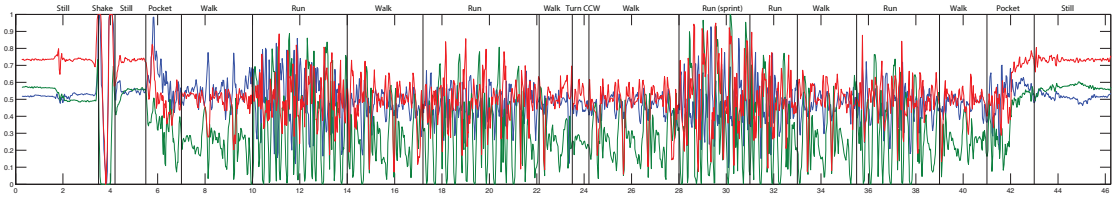


FIGURE 6.3: Run 1: Walk-run-roemer, accelerometer

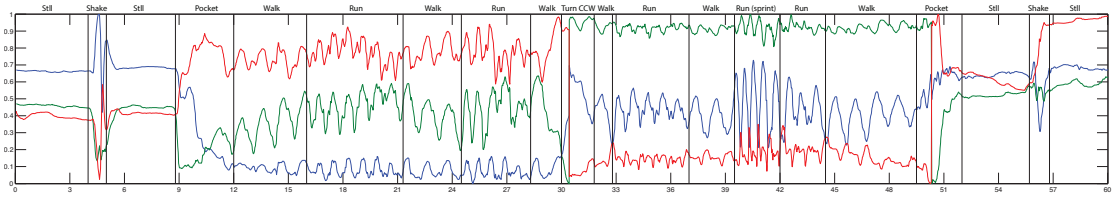


FIGURE 6.4: Run 2: Walk-run-jos, rotation

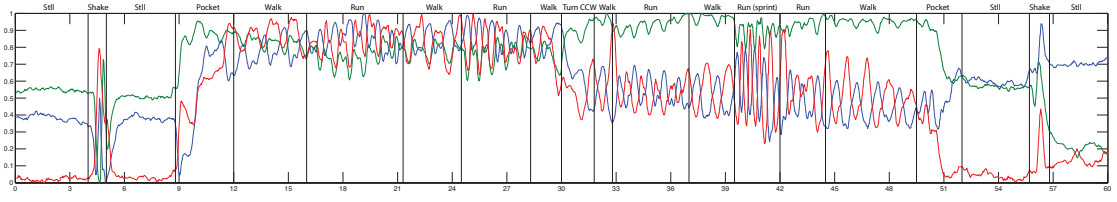


FIGURE 6.5: Run 2: Walk-run-jos, Mag


FIGURE 6.6: Run 2: Walk-run-jos, accelerometer

# Chapter 7

# Conclusion

## 7.1 Outline

*Not intented for the reader.*

- Conclude research
- Future research

# Appendix A

# Summaries

*Please ignore this Appendix. This appendix is for my own personal use. It contains summaries of articles I have read.*

## A.1 Support Vector Machines

### A.1.1 Machine learning: the art and science of algorithms that make sense of data

Book by Peter Flach: [22]. Mainly about chapter 7, "Linear Models". Most important: section 7.3 - 7.5, about support vector machines and non-linearity. **Some parts are direct text; do not use this text directly!**

#### A.1.1.1 Linear models

Models can be represented by their geometry of $d$ real-values features. Data points are represented in the $d$-dimensional Cartesian coordinate system/space $\mathcal{X} = \mathbb{R}^d$. Geometric concepts such as lines and planes can be used for *classification* and *regression*. An alternative approach is to use the distance between data points as a similarity measure, resulting from the geometrical representation. Linear methods do not use that property, but rely on understanding of models in terms of lines and planes.

Linear models are of great interest in machine learning because of their simplicity. A few manifestations of this simplicity are:

- Linear models are *parametric*, thus fixed small number of parameters that need to be learned from the data.

- Linear models are *stable*, thus small variations in training data have small impact on the learned model. In logical models they can have large impact, because "splitting rules" in root have great impact.
- Due to relative few parameters, less likely to *overfit* the training data.

The last two are summarized by saying that *linear models have low variance but high bias*. This is preferred with limited data and overfitting is to be avoided.

Linear models are well studied, in particular for the problem of linear regression. This can be solved by the *least-squares* method and classification as discussed in section A.1.1.2, the *perceptron* as explained in section A.1.1.3. Linear regression with the *support vector machine* is handled in section A.1.1.4 and used for probability density estimation in section A.1.1.5. The kernel trick used for learning non-linear models is explained in section A.1.1.6.

#### A.1.1.2 Least-squares method

The regression problem is to learn a function estimator $\hat{f} : \mathcal{X} \to \mathbb{R}$ from the examples $(x_i, f(x_i))$ where we assume $\mathcal{X} = \mathbb{R}^d$. The difference between the actual and estimated function values are called *residuals* $\epsilon_i = f(x_i) - \hat{f}(x_i)$. The *least-squares method* finds the estimation $\hat{f}$ by minimizing $\sum_{i=1}^{n} \epsilon_i^2$. Univariate regressesion assumes a linear equation $y = a + bx$, with parameters $a$ and $b$ chosen such that the sum of squared residuals $\sum_{i=1}^{n}(y_i - (a + bx_i))^2$ is minimized. Here the estimated parameter $\hat{a}$ is called the *intercept* such that it goes through the (estimated) pooint $(\hat{x}, \hat{y})$ and $\hat{b}$is the *slope* which can be expressed by the (co)variances: $\hat{b} = \frac{\sigma_{xy}}{\sigma_{xx}}$. In order to find the parameters, take the partial derivatives, set them to 0 and solve for $a$ and $b$.

Although least-squares is sensitive to outliers, it works very well for such a simple method. This can be explained as follows. We can assume the underlying function is indeed linear but contaminated with random noise. That means that our examples are actually $(x_i, f(x_i) + \epsilon_i)$ and $f(x) = ax + b$. If we know $a$ and $b$ we can calculate what the residuals are, and by knowing $\sigma^2$ we can estimate *the probability of observing the residuals*. But since we don't know $a$ and $b$ we have to estimate them, by estimating the values for $a$ and $b$ that maximizes the probability of the residuals. This is the *maximum-likelihood estimate* (chapter 9 in the book).

The least-squares method can be used for a (binary) classifier, by encoding the target variable $y$ as classes by real numbers $-1$ (negative) and 1 (positive). It follows that $\boldsymbol{X}^T(y) = P\boldsymbol{\mu}^+ - N\boldsymbol{\mu}^-$, where $P$, $N$, $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$ are the number of positive and negative

examples, and the $d$-vectors containing each feature's mean values, resp. The regression equation $y = \bar{y} + \hat{b}(x - \bar{x})$ can be used to obtain a decision boundary. We need to determine the point $(x_0, y_0)$ such that $y_0$ is half-way between $y^+$ and $y^-$ (the positive and negative examples, i.e. $y_0 = 0$).

#### A.1.1.3 Perceptron

Labeled data is *linearly separable* if the exists a linear boundary separating the classes. The least-squares may find one, but it is not guaranteed. Image a perfect linearly separable data set. Move all the positive points away from the negative, but one. At one point the new boundary will exclude (mis qualify) the one original positive outlier, due to the mean-statistics it relies on. The *perceptron* will guaranteed perform perfect separation when the data allows it to be. It was originally proposed as a *simple neural network*. It works by iterating over the training set and modifying the weight vector for every misclassified example ($\boldsymbol{w} \cdot \boldsymbol{x}_i < t$ for positive examples $\boldsymbol{x}_i$). It uses a learning rate $\eta$, for a misclassified $y_i = \{-1, +1\}$: $\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$. The algorithm can be made *online* by processing a stream of data points and and updating the weight vector only when a new data point is misclassified.

When the algorithm is completed, every $y_i \boldsymbol{x}_i$ is added $\alpha_i$ times to the weight vector (every time it was misclassified). Thus, the weight vector can be expressed as: $\boldsymbol{w} = \sum_{i=i}^{n} \alpha_i y_i \boldsymbol{x}_i$. In other words: the weight vector is a linear combination of the training instances. The dual form of the algorithm learns the instance weights $\alpha_i$ rather than the features weights $\boldsymbol{w}_i$. An instance $\boldsymbol{x}$ is then classified as $\hat{y} = sign(\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x})$. This means that during the training only the pairwise dot-products of the data is needed; this results in the $n$-by-$n$ Gram matrix $\boldsymbol{G} = \boldsymbol{X}\boldsymbol{X}^T$. This instance-based perspective will be further discussed in section A.1.1.4 about the support vector machine.

#### A.1.1.4 Support Vector Machine

A training example can be expressed by its *margin*: $c(x)\hat{s}(x)$, where $c(x)$ is $+1$ for positive and $-1$ for negative examples and $\hat{s}(x)$ is the score. The score can be expressed as $\hat{s}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} - t$. A true positive example $\boldsymbol{x}_i$ has a margin $\boldsymbol{w} \cdot \boldsymbol{x}_i > 0$ and a true negative $\boldsymbol{x}_j$ has $\boldsymbol{w} \cdot \boldsymbol{x}_j < 0$. If $m^+$ and $m^-$ are the smallest positive and negative examples, then we want the sum of these to be as large as possible. *The training examples with these minimal values are closest to the decision boundary $t$ and are called the support vectors.* The decision boundary is defined as a linear combination of the support vectors. The margin is thus defined as $\frac{m}{\|\boldsymbol{w}\|}$. Minimizing the margin (which is often set to 1 and rescaling is allowed) yields to minimizing $\|\boldsymbol{w}\|$, or: $\frac{1}{2}\|\boldsymbol{w}\|^2$, restricted that none of the

training points fall inside the margin. This gives the following quadratic, constrained optimization problem:

$$\boldsymbol{w}^*, t^* \in \underset{\boldsymbol{w},t}{\operatorname{argmin}} = \frac{1}{2}\|w\|^2 \quad \text{subject to } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - t) \geq 1, 1 \leq i \leq n \tag{A.1}$$

This equation can be transformed with the Lagrange multipliers by adding the constraints to the minimization part with multipliers $\alpha_i$. Taking the partial derivative with respect to $t$ and setting it to 0, we find that for the optimal solution (threshold) $t$ we have $\sum_{i=1}^{n} \alpha_i y_i = 0$. When we take the partial derivative with respect to $w\boldsymbol{w}$ we see that the Lagrange multipliers define the weight vector as a linear combination of the training examples. This partial derivative is 0 for an optimal weight we get that $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$, *which is the same expression as for the perceptron derived in section A.1.1.3*. By plugging $\boldsymbol{w}$ and $t$ back into the Lagrange equation, we can eliminate these and get the dual optimization problem entirely formulated in terms of the Lagrange multipliers:

$$\Lambda(\alpha_1, \ldots, \alpha_n) = -\frac{1}{2} \sum_{i=i}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \sum_{i=1}^{n} \alpha_i \tag{A.2}$$

The dual problem maximizes this function under positivity constraints and one equality constraint: ****** TODO: fix equation (now commented) ******

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{A.3}$$

This shows to important properties:

1. Searching for the maximum-margin decision boundary is equivalent to searching for the support vectors; they are the training examples with non-zero Lagrange multipliers.
2. The optimization problem is entirely defined by pairwise dot products between training instances: the entries of the Gram matrix.

The second property enables powerful adaption for support vector machines to learn non-linear decision boundaries, as discussed in section A.1.1.6.

An other solution to non-linear separable data, that is when the constraints $\boldsymbol{w} \cdot \boldsymbol{x}_i - t \geq 1$ are not jointly satisfiable, is to add *slack variables* $\xi_i$, one for each example. This allows them to be in the margin, of even at the wrong side of the boundary – known as boundary errors. Thus, the constraints become $\boldsymbol{w} \cdot \boldsymbol{x}_i - t \geq 1 - \xi_i$.

"In summary, *support vector machines are linear classifiers that construct the unique decision boundary that maximizes the distance to the nearest training examples (the support vectors).* Training an SVM involves solving a large quadratic optimization problem and is usually best left to a dedicated numerical solver."

#### A.1.1.5 Density Functions from linear classifiers

The score of an data point can be used to obtain the signed distance of $\boldsymbol{x}_i$ to the decision boundary:

$$d(\boldsymbol{x}_i) = \frac{\hat{s}(\boldsymbol{x}_i)}{\|w\|} = \frac{\boldsymbol{w} \cdot \boldsymbol{x}_i - t}{\|w\|} = \boldsymbol{w}' \cdot \boldsymbol{x}_i - t' \tag{A.4}$$

where $\boldsymbol{w}' = \boldsymbol{w}/\|\boldsymbol{w}\|$ rescaled to unit length and $t' = t/\|w\|$ corresponds to the rescaled intercept. this geometric interpretation of the scores enables them to turn into probabilities. Let $\bar{d}^+ = \boldsymbol{w} \cdot \boldsymbol{\mu}^+ - t$ denote the mean distance of the positive examples to the boundary, where $\boldsymbol{\mu}^+$ is the mean of positive examples (in the grid) and $\boldsymbol{w}$ is unit length. We can assume that the distance of the examples is normally distributed around the mean (which give a bell curve when plotted).

If we obtain a new point $\boldsymbol{x}$ we can get the class by $sign(d(\boldsymbol{x}))$. We would like, instead, to get the probability (using Bayes' rule)

$$\hat{p}(\boldsymbol{x}) = P(+|d(\boldsymbol{x}) = \frac{P(d(\boldsymbol{x})|+)P(+)}{P(d(\boldsymbol{x})|+)P(+) + P(d(\boldsymbol{x})|-)P(-)} = \frac{LR}{LR + 1/clr} \tag{A.5}$$

where $LR$ is the likelihood ratio obtained from the normal score distributions, and $clr$ is the class ratio. With some rewriting we can convert $d$ into a probability by means of the mapping $d \mapsto \frac{exp(d)}{exp(d)+1}$, which is the *logistic function.* The logarithm of the likelihood ratio is linear in $\boldsymbol{x}$ and such models are called *log-linear models.* This logistic calibration procedure can change the location of the decision boundary but not ts direction. There may be an alternative weight vector with a different direction that assign a higher likelihood to the data. Finding that maximum-likelihood linear classifier using the logistic model is called *logistic regression.*

#### A.1.1.6 Non-linear models

Linear methods such as least-squares for regression can be used for binary classification, yielding in the basic linear classifier. The (heuristic) perceptron guaranteers to classify correctly linear separable data points. Support vector machines find the unique decision boundary with maximum margin and can be adapted to non-linear separable data. These methods can be adjusted to learn non-linear boundaries. The main idea is to

transform the data from the *input space* non-linearly to a *feature space* (which can, but does need to be in a higher dimension) in which linear classification can be applied. The mapping back from the feature space to the input space is often non-trivial (e.g. mapping $(x, y)$ to feature space by $(x^2, y^2)$, yields in four coordinates when transformed back to the input space).

*The remarkable thing is that often the feature space does not have to be explicitly constructed, as we can perform all necessary operations in input space.* For instance; the perceptron algorithm mainly depends on the dot product of $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$. Assuming $\boldsymbol{x}_i = (x_i, y_i)$ and $\boldsymbol{x}_j = (x_j, y_j)$, the dot product can be written as $\boldsymbol{x}_i \cdot \boldsymbol{x}_j = x_i x_j + y_i y_j$. The instances in quadratic feature space are $(x_i^2, y_i^2)$ and $(x_j^2, y_j^2)$ and their dot product is $(x_i^2, y_i^2) \cdot (x_j^2, y_j^2) = x_i^2 x_j^2 + y_i^2 y_j^2$. This is almost equal to $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^2 = (x_i x_j)^2 + (y_i y_j^2 + 2 x_i x_j y_i y_j$, but not quite because of the third term. We can make the equations equal by *extending the feature space* (to a higher dimension) with a third feature $\sqrt{2xy}$, so the feature space is $\phi(\boldsymbol{x}_i) = (x_i^2, y_i^2, \sqrt{2 x_i y_i})$.

If we define $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i, \boldsymbol{x}_j)^2$ and replace $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ with $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the (perceptron) algorithm, we obtain the *kernel perceptron* with the degree $p = 2$. We are not restricted to polynomial kernels; an often used kernel is the *Gaussian kernel*, defined as:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}) \tag{A.6}$$

where $\sigma$ is known as the *bandwidth* parameter. We can think of the Gaussian kernel as imposing a Gaussian (i.e. , multivariate normal) surface on each support vector in instance space, so that the boundary is defined in therms of those Gaussian surfaces. Kernel methods are best known in combination with support vector machines. Notice that the soft margin optimization problem is defined in terms of dot product between training examples, and thus the 'kernel trick' can be applied. Note that the decision boundary learn with a non-linear kernel cannot be represented by a simple weight vector in input space. Thus, to classify a new example $\boldsymbol{x}$ we need to evaluate $y_i \sum_{j=1}^{n} \alpha_j y_j \kappa((x), \boldsymbol{x}_j)$ (the Gram matrix?) involving all training examples, or at least all with non-zero multipliers $a_j$ (the support vectors).

### A.1.2 Change Point Detection In Time Series Data Using Support Vectors

Paper by Fatih Camci [14] About segmentation with SVMs. Will be main material for section 2.6 about SVMs.

#### A.1.2.1 Introduction

Interprets change detection as finding the transition points from one underlying time series generation model to another. The change point is mostly represented in a sudden change in mean or variance. Existing models detect changes in mean and increase in variance, *but fail to recognize decrease in variance.* Many methods require some model (like Auto-Regressive [AR]) to fit the time series in order to eliminate the noise. Thus, the effectiveness of the method is tied to the fitness degree of the model to the time series data. These two problems (lack of variance decrease detection and model-bound fitness degree) leads to this work; Support Vector based Change Point Detection targeting changes in variance and/or mean without any assumption of model fitting of data distribution. This method *does not use a time series model* for fitting and targets *both increase and decrease* in mean and variance.

#### A.1.2.2 Related work

Change Point Detection (CPD) can be categorized in posterior (off-line) and sequential (on-line). Sequential receive data sequentially and analyze previously obtained data to detect the possible change in current time. This method is based on sequential analysis and focuses on change on mean and variance in time domain. Other methods generally suffer from:

- Inability / inefficiency in detecting variance decrease.
- Assumptions about the statistical distribution of the data, obtained as error of fitting the (AR) model.
- Necessity of training the model with possible changes.

#### A.1.2.3 Support vector based one-class classification

Although SVM was originally designed for two-class classification, it has been successfully applied to multi-class and one-class classification. SVM-based one-class classification gives the minimum volume closed spherical boundary around the data, represented by center $c$ and radius $r$. It minimizes $r^2$ (representing structural error), and uses a penalty coefficient $C$ for each outlier with distance $\xi_i$ from the hyper-sphere boundary:

$$\begin{gathered} \text{Min } r^2 + C\sum_i \xi_i \\ \text{Subject to : } \|\boldsymbol{x}_i - c\|^2 \leq r^2 + \xi_i \quad \xi_i \geq 0, \quad \forall i, \boldsymbol{x}_i : i\text{th data point} \end{gathered} \tag{A.7}$$

This quadratic optimization problem can be transformed to its dual form by introduction Lagrange multipliers $\alpha_i$. If, for a data point, the multiplier $\alpha_i = 0$, then that point is inside the sphere. When it is $0 < \alpha_i < C$, then it is on the boundary. Data points for which the multiplier is $\alpha_i = C$ are located outside the sphere (and are penalized). The dual form is:

$$\begin{aligned} \text{Max} \sum_i \alpha_i(\boldsymbol{x}_i \cdot \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \\ \text{Subject to : } 0 \leq \alpha_i \leq C \quad \forall i, \quad \sum_i \alpha_i = 1 \end{aligned} \tag{A.8}$$

Note that only dot-products of the data points $\boldsymbol{x}$ appear. In order to transform the data points to a higher dimension, to create a good representational hyper-sphere, kernels replace the dot products without compromising computational complexity. The problem then becomes:

$$\text{Max} \sum_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{A.9}$$

It has been shown that Gaussian kernels offer better performance for one-class classification the others. The optimization of the *scale parameter* has led to several implementations. As can been seen, there are no assumptions about the data distribution or independency made.

#### A.1.2.4 Problem formulation

[Not summarized here, useful for e.g. section 2.2]

#### A.1.2.5 SVCPD: The algorithm

Instead of using statistical properties of the data, for each window of size $w$ a hyper-sphere is constructed without increasing computational complexity due to the *kernel trick*. The window size is related to the sensitivity of the method to change; small windows are sensitive with high false alarm rate whilst large windows are slow to detect change and have low alarm rates. The algorithm is listed in table A.1. Note that SVCPD can be applied directly to multidimensional data, whilst many other methods can only be applied to one-dimensional data.

| Step | Action |
|---|---|
| 1 | Start with $n$ observations and construct hyper-sphere |
| 2 | Add next observation $x_t$ and drop first one |
| 3 | Identify new hyper-sphere and its *approximate radius* |
| 4 | if $x_t$ is outside hyper-sphere, mark $t$ as change points and continue from step 2 |
| 5 | calculate radius average of last $w$ hyper-planes |
| 6 | calculate radius ratio $\hbar$. If lower than $th_{low}$ or greater than $th_{high}$ then mark $t$ as change point |
| 7 | continue from step 2 |

TABLE A.1: Support Vector machine based Change Point Detection algorithm

## A.2 CUSUM for variance

### A.2.1 Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance

Carla Inclan and George C. Tiao [36], 1944, 162 refs.

#### A.2.1.1 Introduction

This paper is about reflective detection of multiple changes of variance in a sequence of independent observations. This is a statistical method, which differs from others (in that field) such as Bayesian method (Bayes ratio, posterior odds), maximum likelihood methods and (autoregressive) models. This approach uses the centered version of cumulative sums of squares to search for change points systematically and iteratively (and reflective).

#### A.2.1.2 Centered Cumulative Sum of Squares

The cumulative sum of squares is often used for change detection in the mean. It is defined as $C_k = \sum_{i=1}^{k} \alpha_t^2$ for a series of uncorrelated random variables $\{\alpha_t\}$ with mean 0 and variance $\sigma_t^2, t = 1, 2, \ldots, T$. The centered (and normalized) cumulative sum of squares is:

$$D_k = \frac{C_k}{C_T} - \frac{k}{T}, \quad k = 1, \ldots, T, \quad \text{with } D_0 = D_T = 0 \tag{A.10}$$

For homogeneous variance the plot of $D_k$ against $k$ (the first $k$ elements of the series) will oscillate around 0. When a sudden change in variance occurs, the pattern of the plot of $D_k$ will break out some specified boundaries with high probability. For $C_k$ it holds that, under homogeneous variance, the plot will be a straight line with slope $\sigma^2$.

With one of more change points the plot appears as a line of several straight pieces. The plot of $D_k$ creates a peak for a smaller and a trough for a larger variance, is visually more clear and breaks out a predefined value. The search for a change point is variance is than to find $k^* = \max_k |D_k|$. If the value of $D_k$ at $k^*$ exceeds a predefined value (e.g. $D^*_{0.5} = 1.358$, for $\sqrt{T/2D_k}$ because of the Brownian bridge property), that value of $k^*$ will be an estimate for a change point.

There is a relation between $D_k$ and the $F$ statistic, which is used for testing equality of variances between two independent samples. For a fixed $k$, $D_k(F)$ is a monotone function of $F$ (it depends only on $k$ through $k/T$). An important distinction: the $F$ statistic is *used with known* $k$, whereas we are looking for $\max_k |D_k|$ to determine the location of the change point.

When we assume that $\{\alpha_t\}$ is normally distributed with mean 0 and variances $\sigma_t^2$, then we can obtain the *likelihood ratio* for testing the hypothesis of one change against the hypothesis of no change in the variance. When maximizing the likelihood estimator for a location $\kappa$, we can find the log-likelihood ratio $LR_{0,1}$. Although $LR_{0,1}$ and $\max_k |D_k|$ are related, they are not the same. The latter puts more weight near the middle of the series is thus biased toward $T/2$.

The (expected) value of $D_k$ given a change in variance differs in the context. If a smaller variance corresponds to the smaller portion of the series, then it will be harden to find the change point using $D_k$. There is a masking effect when there are multiple change points in the series; the order of small, medium and large variances result in the value of $D_k$. The iterative algorithm presented in this paper in section A.2.1.3 is designed to lessen the masking effect.

#### A.2.1.3 Multiple changes: Iterated cumulative sums of squares

In case of a single change point the $D_k$ method would succeed. But we are interested in multiple change points of variance, and thus the usefulness of the $D_k$ reduces due to the masking effect. A solution is to iteratively applying the method and dividing the series at each possible change point. The algorithm is presented in table A.2. It is the third steps which reduces the masking effect and helps to "fine tune" the algorithm by (re)moving the potential change points by checking each location given the adjacent ones.

| Step | Action |
|---|---|
| 0 | Let $t_1 = 1$ |
| 1 | Calculate $D_k(\alpha[t_1 : T])$. Let $k^*(\alpha[t_1 : T])$ be the point at which $\max_k \lvert D_k(\alpha[t_1 : T])\rvert$. Let $D^*$ be the asymptotically critical value and $M$ the max value in the series segment (?). If $M > D^*$ then consider $k^*$ to be a change point. Else, there is no change point and the algorithm stops. |
| 2a | Repeat for the first part (up to the change point), until no more change points are found. |
| 2b | Repeat for the second part (from the change point forward), until no more change points are found. |
| 3 | When two or more change points are found; check for each $\alpha[j-1 : j+1]$ if there is indeed a change point ($j$). Repeat until the number of change points does not change and each new found change point is "close" enough to previous. |

TABLE A.2: Iterated Cumulative Sums of Squares Algorithm

#### A.2.1.4 Results

When the ICSS algorithm was applied to stock data, it resulted in comparable results as the maximum likelihood estimates and Bayesian analysis. The performance (CPU-time and correct observations with artificial data) of ICSS outperforms the other two. The heavy computational burden of posterior odds can be partially alleviated by the maximum log-likelihood method. The ICSS algorithm avoids calculating a function at all possible locations of change points due the iterative manner.

## A.3 Density Ratio Estimation

### A.3.1 Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation

Kawahara and Sugiyama [38], 2009, 45 refs.

"This paper provides a change-point detection algorithm based on direct density-ratio estimation that can be computed very efficiently in an online manner".

#### A.3.1.1 Introduction

The problem of change-point detection is well studied over the last decades in the field of statistics. A common statistical formulation of change-point detection is to consider the probability distributions over past and present data intervals, and regard the target time

as a change-point if the two distributions are significantly different. Some approaches (such as CUSUM and GLR) make use of the *log-likelihood ratio*, and are extensively explored in the data mining community. Many approaches (novelty detection, maximum-likelihood ratio, learning of autoregressive models, subspace identification) rely on pre-specified parametric models (probability density models, autoregressive models, state-space models). That makes it less applicable to real-life problems. There have been some non-parametric density estimation approaches proposed, but that is known to be a hard problem. The key idea of this paper to directly estimate the *ratio* of the probability densities (also known as *importance*). The KLIEP is an example, but it is a batch algorithm. This paper introduces an online version of the KLIEP algorithm and develops a flexible and computationally efficient change-point detection method. This method is equipped with a natural cross validation procedure and thus the value of tuning parameters can be objectively determined.

#### A.3.1.2 Problem formulation and Basic Approach

Let $\boldsymbol{y}(t)(\in \mathbb{R}^d)$ be a $d$-dimensional time series sample at time $t$. The task is to detect whether there exists a change point between two consecutive time intervals, called the *reference* and *test* intervals. The conventional algorithms consider the likelihood ratio over samples from the two intervals. Since time-series samples generally are not independent over time it is hard to deal with them directly. To overcome this difficulty, we consider *sequences* of samples in the intervals: $\boldsymbol{Y}(t)(\in \mathbb{R}^{dk})$ is the forward subsequence of length $k$ at time $t$. This is a common practice in subspace identification since it takes implicitly time correlation into consideration. The algorithm in the paper is based on the logarithm of the likelihood ration of the *sequence sample* $\boldsymbol{Y}$:

$$s(\boldsymbol{Y}) = \ln \frac{p_{\mathrm{te}}(\boldsymbol{Y})}{p_{\mathrm{rf}}(\boldsymbol{Y})} \tag{A.11}$$

where $p_{\mathrm{te}}(\boldsymbol{Y})$ and $p_{\mathrm{rf}}(\boldsymbol{Y})$ are the probability density functions of the reference and test sequence samples. Let $t_{\mathrm{rf}}$ and $t_{\mathrm{te}}$ be the starting points of the reference and test intervals. Decide if there is a change-point between the reference and test interval by monitoring the logarithm of the likelihood ratio:

$$S = \sum_{i=1}^{n_{\mathrm{te}}} \ln \frac{p_{\mathrm{te}}(\boldsymbol{Y}_{\mathrm{te}}(i))}{p_{\mathrm{rf}}(\boldsymbol{Y}_{\mathrm{te}}(i))} \tag{A.12}$$

If, for a predefined $\mu > 0$, it holds that $S > \mu$ then a change occurs. The remaining question is how to calculate the density ratio, because it is unknown and we need to estimate it from examples. The naive approach would be to first estimate the densities for the reference and test interval separately and then take the ratio. This approach

via non-parametric density estimation may not be effective — directly estimating the density ratio without estimating the densities would be more promising.

The direct estimation of the density ratio is based on the Kullback-Leibler Importance Estimation Procedure. Let us model the density ratio $w(\boldsymbol{Y})$ by a non-parametric Gaussian kernel model:

$$\hat{w}(\boldsymbol{Y}) = \sum_{l=1}^{n_{\mathrm{te}}} \alpha_l K_\sigma(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{te}}(l)), \tag{A.13}$$

where $\{\alpha_l\}_{l=1}^{n_{\mathrm{te}}}$ are parameters to be learned from the data samples and $K_\sigma(\boldsymbol{Y}, \boldsymbol{Y}')$ is the Gaussian kernel function with mean $\boldsymbol{Y}'$ and standard deviation $\sigma$.

#### A.3.1.3 Online Algorithm

...

## A.4 Outlier Detection methods

### A.4.1 A Survey of Outlier Detection Methodologies

Hodge and Austin, [34], 2004, 734 refs.

### A.4.2 Summary

This survey takes a look at different methodologies that perform outlier detection. It gives two definitions of outliers, one of which relates to the problem statement in this thesis (taken from [9]):

> An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.

The survey introduces three fundamental approaches to the problem of outlier detection:

**Type 1** - no prior knowledge about the data; analogous to *unsupervised clustering*. Flags the remote points as outliers; mainly batch-processing systems.

**Type 2** - model (and requires) both normal and abnormal data; analogous to *supervised classification*. Is able to process new data online.

**Type 3** - Model only normal data (and in very few cases abnormal). Generally referred to as *novelty detection*, analogous to *semi-supervised recognition or detection*. It only requires pre-classified normal data. It aims to define a boundary of normality.

The type of approaches of interest in this research is of **type 3**. It is characterized by the ability to recognize new data as normal when it lies within the constructed boundary and as a novelty otherwise. This ability removes the need of sample-abnormality data, which may be hard (or costly) to produce. The method of Tax *et al.* [66] is stated to be of **type 2**, and one can argue that other methods of Tax *et al.* [67, 70] are by their one-class classification instances of **type 3** methods.

The survey states that density-based methods estimate the density distribution of the training data. Outliers are identified as data points lying in a low-density region.

# Appendix B

# Session with Anne 24-06-2013 - Paper Camci analysis

*Please ignore this Appendix. This appendix is for my own personal use. This chapter will look at the paper of Camci [14] ("Change point detection in time series using support vectors") and will answer many question that the paper leaves open. The goal is to make a better justification for the used techniques and made assumptions.*

## B.1 Density estimation / Data description / Vapnik's principle

Following Vapnik's principle, one should "When solving a problem of interest, do not solve a more general problem as an intermediate step" [73] when a limited amount of data is available. For the problem of change detection we are only interested in some characteristics of the data. Solving the complete density estimation might require more data than actually needed when the requested characteristic is a closed boundary around the data.

In [46] it is stated that the SVM by Cortes and Vapnik [21] is a representative example of this principle. Instead of estimating the more general data generating probability distributions, it only learns a decision boundary to differentiate between the two distributions.

The proposed method SVDD of Tax and Duin [68, 70] models the boundary of data under consideration. Thereby it characterizes a data set and can be used to detect novel data or outliers. The performance is compared to methods which model the

distribution's density instead, using Receiver-Operating Characteristic (ROC) curves and false negative rates. The compared methods are: *(1)* normal density which estimates the mean and covariance matrix; *(2)* the Parzen density where the width of the Parzen kernel is estimated; *(3)* a Gaussian Mixture Model optimized using EM; and *(4)* the Nearest-Neighbor Method which compares the local density of an object with the density of the nearest neighbor in the target set (and is thus, just as SVDD, a boundary-based method). The results show that when the problem formulation is to characterize an area in a feature space (and not the complete density distribution) SVDD gives a good data description.

The study of Tax on OCCs [67] further compares density methods, boundary methods (amongst which SVDD) and reconstruction methods. One promesing result for SVDD is that it performs well on read-world data, for which generalization is needed.

The method of Camci [14] uses a Support Vector based method to find change points in the data. It does not explicitly create a density estimation, but instead relies on the spherical boundary and uses its ability to detect novel data or outliers to detect change points in time series data.

****** TODO:* Thus methods of data descriptions should be compared (which are more general) and not only density estimation? *****

The paper of Yin *et al.* [77] makes a distinction between similarity based (using a defined distance measure) and model based (which characterize the data using predictive models) approach. The OC-SVM is used in the model-approach to filter out normal activities in order to detect abnormality behavior.

## B.2 Change point definition

The method of Camci [14] regards a change point as the moment in time that the underlying stochastic process has changed, say from $p^1$ to $p^2$. It assumes that each of these stochastic processes is modeled following a Gaussian distribution, such that a change can occur in the value of the mean and/or the variance; $p^1 \sim N(\mu_1, \sigma_1^2)$. The CUSUM-based method of [36] also regards each semgent as a Gaussian distribution.

The method if Kawahara *et al.* [38] is based on the log likelihood ratio of test samples, and the method by Liu *et al.* [46] uses a comparable dissimilarity measure using the KLIEP algorithm.

The method of Chamroukhi *et al.* [16] is based on a Hidden Markov Model and logistic regression. It assumes a $K$-state hidden process with a (hidden) state sequence, each

state providing the parameters (amongst which the order) for a polynomial. The order of the model segment is determined by model selecting, often using the BIC or the similair AIC [3], as in [31].

****** TODO: REVIEW zoeken: change points in time series ******

Periodicity/consecutive data vs unique/irregular data?

Change in model parameters (mean/variance, of linear/non-linear)? –¿ Model selection?

Definition of continuity –¿ windows the domain and problem. Will result in definition of dis-continuity –¿ this is the goal to find Relation with double differentiation.

## B.3 Data and model

Why assume (model of) data is Gaussian/normal distribution? Thus, piecewise linear with mean and variance as changing properties. Why not set of polynomial models, as for example in [16]?

What is the best model for accelerometer data of human activities? –¿ Look to result of model-selecting papers.

Should we build a model of the data? (Gaussian distribution is also the model). And be able to reconstruct?

Why is it better (as Camci states) that a method makes no assumptions about the form of data/distribution of data? Is it that there are less parameters to estimate?

Many methods describe and compare methods to construct classifier models for the classification of accelerometer data, such as [42] and [78] (often using extracted features from the raw data signal). In contrast, we could not find a clear characterization of accelerometer data obtained from human activities. When the problem of temporal segmentation is regarded in this context, a formalization of the data under consideration is needed. Some assume the data follows a piecewise linear set of segments with a mean and noise/variance modeled by a normal (Gaussian) distribution (such as [14]). Other approaches regard the data as a set of polynomials, which can be estimated by regression (such as [16]) and apply a form of model selection to each segment.

****** TODO: make a clear distinction between model types; similarity (distance based) or model based, as in [77] ******

## B.4 Segmentation (SVM) method

Overview of segmentation methods. Collection of papers use relative and direct density-ratio estimation [38, 46].

Why use SVM for density estimation? Look for justification, perhaps a review paper which compares/mentions SVM for temporal segmentation of (human activity type of) data?

Why use RBF/Gaussian kernel? Is it because of OC-SVM of because of form the data? Why not polynomial/linear?

- "Use RBF when relation between class and data is non-linear"
- "RBF uses less paramters (C for penalty/soft margin and gamme for kernel width) than non-linear polynomial kernels"
- (arguments from [78] "Optimal model selection for posture recognition in home-based healthcare")
- Survey [34] states RBF is similair to Gaussian Mixture model (page 25).

****** TODO: read [51] and [11] on geometry of SVMs ******

Re-evaluatie [27], uses (amongst others) a pca-based dimension-detection method (?). Also uses model-selection as an intermediate step.

### B.4.1 Higher dimension mapping (including kernel)

What does the mapping from the data-space to higher dimension looks like?

What is a RBF kernel?

What form has the higher dimensional space?

How do relations over data, such as distance, volume and noise, act in the higher dimensional space?

What is the kernel trick to not explicit do the mapping to the higher space?

****** TODO: Note: in [51] it is explained why the inner-product between two vectors is a logical choice for the distance/similarity measure. ******

## B.5 Relation to other methods

### B.5.1 Novelty/outlier detection

****** TODO: Read [34] "A survey of outlier detection methodologies" to compare with other methodologies. ******

- One-class classification is referred to as "Type 3", with *semi-supervised recognition or detection.*
- It explains why one-class can be beneficial over type-2 where negative examples needs to be provided.
- We are interested in type-3, so compare SVM with the others stated in this survey regarding type-3.
- Often refers to the convex hull of the data set. Link with geometric approach as in [11, 51]?
- It states that PCA and regression techniques are linear models and thus often are too simple for practical applications. SVMs try to find a hyperplane in higher dimensional space; linear models to implement complex class boundaries. It refers to [68].

### B.5.2 Scale parameter

### B.5.3 Robust statistics / "M-Estimators"

Is the method robust, in the sense that outliers have restricted impact on the quality.

What is the relation to M-Estimators (Wikipedia: "M-estimators are a broad class of estimators, which are obtained as the minima of sums of functions of the data", http://en.wikipedia.org/wiki/M-estimator)

## B.6 Quality metrics

As used in Camci: benefit, false alarm rate

Asymmetric test: feed data from front-to-back and back-to-front; how far are matched datapoint apart.

Model reconstruction error: try to reconstruct the simulated models, test similarity (BIC?). Is that a good measure? (If we are only interested in finding change points...)

# Appendix C

# Summary of papers and principle formulas

## C.1 Change point detection in time series data using support vectors, by Camci

Paper: [14]. The main concept of this paper is to construct a hypersphere around the data and thereby generating a boundary. A change point is detected when the radius of the hypersphere grows or shrinks significantly, or when a data point falls outside the boundary.

The main cost function being minimized:

$$\begin{aligned} &\underset{r}{\text{minimize}} \quad r^2 + C\sum_i \xi_i \|x\| \\ &\text{subject to} \quad \|\boldsymbol{x}_i - c\|^2 \leq r^2 + \xi_i, \quad \xi_i \geq 0 \quad \forall i, \boldsymbol{x}_i : i\text{th data point} \end{aligned} \tag{C.1}$$

Where $r$ is the radius of a (hyper)circle with center $\boldsymbol{c}$, $C$ is the penalty coefficient for every outlier and $\xi_i$ is the distance from the $i$th data point to hypersphere (also known as the slack variable).

The dual form by introducing the Lagrange multipliers ($\alpha_i, \alpha_i \geq 0$) and eliminating the slack variables $\xi$ is:

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \sum_i \alpha_i(\boldsymbol{x}_i \cdot \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \\ &\text{subject to} \quad 0 \leq \alpha_i \leq C \quad \forall i, \quad \sum_i \alpha_i = 1 \end{aligned} \tag{C.2}$$

To allow for a non-linear relation between the data points and the data boundary, the inner product can be replaced by a (e.g. Gaussian) kernel function: $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

## C.2 Change-Point detection in time-series data by Direct Density-Ratio Estimation

Paper: [38].

The density ratio $w(\boldsymbol{Y})$ is modeled by a Gaussian kernel over sequences $\boldsymbol{Y}$ of samples (sequence $\boldsymbol{Y}_{te}(l)$ is the test sequence from the $l$th position on):

$$\hat{w}(\boldsymbol{Y}) = \sum_{l=1}^{n_{\mathrm{te}}} \alpha_l K_\sigma(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{te}}(l)), \tag{C.3}$$

where $\{\alpha_l\}_{l=1}^{n_{\mathrm{te}}}$ are parameters to be learned from the data samples and $K_\sigma(\boldsymbol{Y}, \boldsymbol{Y}')$ is the Gaussian kernel function with mean $\boldsymbol{Y}'$ and standard deviation $\sigma$. The learned parameters minimize the Kullback-Leibler divergence from the sequence to the test sequence. The maximization problem then becomes:

$$\begin{aligned} \underset{\{\alpha_l\}_{l=1}^{n_{\mathrm{te}}}}{\text{maximize}} \quad & \sum_{i=1}^{n_{\mathrm{te}}} \log \left( \sum_{l=1}^{n_{\mathrm{te}}} \alpha_l K_\sigma(\boldsymbol{Y}_{\mathrm{te}}(i), \boldsymbol{Y}_{\mathrm{te}}(l)) \right) \\ \text{subject to} \quad & \frac{1}{n_{\mathrm{rf}}} \sum_{i=1}^{n_{\mathrm{rf}}} \sum_{l=1}^{n_{\mathrm{te}}} \alpha_l K_\sigma(\boldsymbol{Y}_{\mathrm{rf}}(i), \boldsymbol{Y}_{\mathrm{te}}(l)) = 1, \text{ and } \alpha_1, \ldots, \alpha_{n_{\mathrm{te}}} \geq 1 \end{aligned} \tag{C.4}$$

With the estimated parameters the logarithm of the likelihood ratio between the test and reference interval can be calculated, which signals a change point if it is beyond a certain threshold $\mu$:

$$S = \sum_{i=1}^{n_{\mathrm{te}}} \ln \frac{p_{\mathrm{te}}(\boldsymbol{Y}_{\mathrm{te}}(i))}{p_{\mathrm{rf}}(\boldsymbol{Y}_{\mathrm{te}}(i))} \tag{C.5}$$

## C.3 Joint segmentation of multivariate time series with hidden process regression for human activity recognition, by Chamroukhi

Paper: [16]. This approach models the time series data by a parameterized regression, filtered with a HMM to smooth out high frequency activity transitions. With each observation $i$, generated by a $K$-state hidden process, an activity label $z_i$ (and thus sequence) is associated. Observations follow a regression model:

$$y_i = \beta_{z_i}^T \boldsymbol{t}_i + \sigma_{z_i}\epsilon_i; \quad \epsilon_i \sim \mathcal{N}(0,1), \quad (i = 1, \ldots, n) \tag{C.6}$$

The observations get a label assigned by maximizing the logistic probability ($\pi_k$):

$$\hat{z}_i = \underset{1 \leq k \leq K}{\operatorname{argmax}}\, \pi_k(t_i; \hat{\boldsymbol{w}}), \quad (i = 1, \ldots, n) \tag{C.7}$$

## C.4 Support Vector Data Description, by Tax and Duin

Paper: [70]. The paper proposes the SVDD method, analogous to the Support Vector Classifier (SVC) of Vapnik [73], based on the separating hyper-plane of Schölkopf *et al.* [59]. Where SVC is ablo to distinguish data between two classes, SVDD obtains a closed boundary around the target class and can detect outliers.

Method and formulas very similair to description in Section C.1.

## C.5 Support Vector Density Estimation, by Weston et al.

Paper: [74]. Using the notation of [74], the distribution function of a density function $p(x)$ is represented as:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} p(t)\,\mathrm{d}t \tag{C.8}$$

To find the density the following linear equation need to be solved:

$$\int_{-\infty}^{\infty} \theta(x-t)p(t)\,\mathrm{d}t = F(x) \tag{C.9}$$

where

$$\theta(x) = \begin{cases} 1, & x > 0 \\ 0, & otherwise \end{cases}$$

In this problem the distribution function $F(x)$ is unknown and instead we are given the i.i.d. data $x_1, \ldots, x_l$ generated by $F$.

The empirical distribution function can now be constructed as:

$$F_l(x) = \frac{1}{l}\sum_{i=1}^{l} \theta(x - x_i) \tag{C.10}$$

## C.6 An online algorithm for segmenting time series, by Keogh et al.

Paper: [40]. This method approximates the signal with piecewise linear representation. Change points are encountered at the time at which a new segment is used to represent the signal. The method uses linear regression by taking the best fitting line in the least squares sense, since that minimizes the Euclidian distance which is used as a quality metric. Linear interpolation is considered but since that always has a greater sum of squares error it is disregarded.

Linear regression assumes a relation from $n$ observations $\boldsymbol{X}$ to the dependend variable $\boldsymbol{y}$ using the parameter vector $\boldsymbol{\beta}$:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon \tag{C.11}$$

The error function, the sum of squared residuals, being minimized by searching for the best estimation of $\boldsymbol{\beta}$ is:

$$b^* \in \underset{b}{\operatorname{argmin}} = \sum_{i=1}^{n} (y_i - x_i' b)^2 = (\boldsymbol{y} - \boldsymbol{X}b)^T (\boldsymbol{y} - \boldsymbol{X}b) \tag{C.12}$$

## C.7 Online novelty detection on temporal sequences, by Ma and Perkins

Paper: [48]. This method uses support vectors for regression (in contrast with of classification). The regression function, using a kernel function $K(x_i, x_j)$ can be written as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \theta_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{C.13}$$

where $\theta_i$ is a coefficient resulting from the Lagrange multipliers of the original minimization problem. A small fraction these of coefficients are non-zero, and the corresponding samples $\boldsymbol{x}_i$ are the *support vectors*. The regression function $f(\boldsymbol{x})$ is non-linear when a non-linear kernel is chosen.

The regression function is used to created a model of past observations. A matching function is constructed which determines the matching value $V(t_0)$ of a new observations with the constructed model. This matching value is the residual of the regression function at $t_0$.

The algorithm determines *(novel) events*, *occurrences* and *surprises*. *Novel events* are defined as a series of observations for which the confidence value over the number of suprises (out-of-model observations) is high enough. Events thus have a length; they are constructed of a sub-series of observations.

The papers presents an alternative implementation in order to handle fixed-resource environments and thus induce an online algorithm. After $W$ observations have been observed and used for the trained model, the oldest observation is disregarded before the newly obtained observation is incorporated.

*Note:* the Support Vector approach in this paper is used to select the observations to use in the regression model. This differs from one-class applications of support vector machines. The same authors have also presented a paper which does use one-class construction using support vector machines: [49].

## C.8 Time-series novelty detection using one-class support vector machines, by Ma and Perkins

Paper: [49] This approach is very similar to the other paper of this author discussed in the preview section [48]. The difference is that this method does create a SVM-classifier to detect in and out of class examples, whilst the other uses the support vectors to construct a regression function.

The method constructs a hyper-plane which separates as many as possible data points in the feature space with the largest margin from the origin. This is a different from (and more like the original SVM-proposal by Schölkopf [60]) the one-class methodology by Tax which creates a boundary around the data [70].

The hyper-plane in feature space is represented as:

$$f(\boldsymbol{x}) = \boldsymbol{W}^T \Phi(\boldsymbol{x}) - \rho, \tag{C.14}$$

where $\Phi(\boldsymbol{x})$ maps a vector $\boldsymbol{x}$ from the input space $I$ to the (potentially infinite dimensional) feature space $F$. $\boldsymbol{W}$ and $\rho$ are determined by solving a quadratic optimization problem. The dual formulation (using Lagrange multipliers $\alpha_i$) is:

$$\boldsymbol{W} = \sum_{i=1}^{l} \alpha_i \Phi(\boldsymbol{x}_i), \tag{C.15}$$

where $0 \leq \alpha_i \leq \frac{1}{\nu l}$. The parameter $\nu \in (0, 1)$ is set to trade-off the smoothness of $f(\boldsymbol{x})$ and acts as a upper bound on the fraction of outliers over all the data examples in $\boldsymbol{x}$ [60].

Using the *kernel trick* the inner product of two vectors in feature space $F$ can be replaced by a kernel function $K$, which is often the RBF. The equation of the hyper-plane (C.14) then becomes the following (non-linear) function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) - \rho \tag{C.16}$$

The form of the input vector $\boldsymbol{x}$ is considered to be the *(projected) phase space* representation of the original time series. Just like [39] which constructs *sequences* of samples and in contrast with [14], each element of $\boldsymbol{x}$ is a vector with the size of the *embedding dimension* $E$ of the time series. Thus, a time series $x(t)$ is converted to a set of vectors $T_E(N)$:

$$T_E(N) = \{\boldsymbol{x}_E(t),\ t = E \cdots N\}, \tag{C.17}$$

where

$$\boldsymbol{x}_E(t) = [x(t-E+1)\ x(t-E+2)\ \cdots\ x(t)] \tag{C.18}$$

If a point of this vector $\boldsymbol{x}_E(t)$ is regarded (in the feature space $F$) as an outlier, all corresponding values in the original time series are also regarded as such.

## C.9 Least squares one-class support vector machine, by Choi

Paper: [20]. The proposed method uses a SVM for a similarity/distance comparison for testing examples to training examples. Instead of other methods, such as the standard one-class SVM by Schölkopf [60] or the SVDD of Tax and Duin [70], it does not create a boundary for the training data. Instead it uses a least-squared approach to construct a hyperplane to which most of the training examples lie close to.

The objective function to be minimized of the standard one-class method by Schölkopf is formulated as:

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 - \rho + C\sum_j \xi_j \\ \text{subject to} \quad & \boldsymbol{w}\cdot\phi(\boldsymbol{x}_j) \geq \rho - \xi_j \quad \text{and} \quad \xi_j \geq 0 \end{aligned} \tag{C.19}$$

where $\phi$ is a mapping to the feature space.

The least-squares one-class support vector machine has a small variation on the above formula, which results in the following minimization problem:

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 - \rho + \frac{1}{2}C\sum_j \xi_j^2 \\ \text{subject to} \quad & \boldsymbol{w}\cdot\phi(\boldsymbol{x}_j) = \rho - \xi_j \end{aligned} \tag{C.20}$$

The slack variable (for which in the original formulation $\xi_j \geq 0$ should hold) now represents an error caused by a training example $\boldsymbol{x}_j$ with relation to the hyperplane, i.e. $\xi_j = \rho - \boldsymbol{w}\cdot\phi(\boldsymbol{x}_j)$.

In other words, the minimization problem (C.20)) results in a hyperplane with maximal distance from the origin and for which the sum of the squares of errors $\xi_j^2$ are minimized.

# Appendix D

# Planning

| Chapter | Title | Extra | Deadline |
|---|---|---|---|
| 1 | Introduction, abstract | Write at last | 15/10/2013 |
| 2 | Literature review | Partially writting. Maybe focus more on SVM and quick look at other segmentation techniques | 15/9/2013 |
| 3 | (Change detection By) One-Class Support Vector Machines | Mix of current chapter and blog post | 1/10/2013 |
| 4 | Proposed method | Propose new method, or just setup of testing the OC-SVMs with accelerometer data | Date |
| 5 | **Results** | Test with known and public databases of accelerometer data. For this, two implementations (in Matlab?) are needed | 15/8/2013 |
| 6 | Real-world applications | Test with own labeled accelerometer data | 1/9/2013 |
| 7 | Conclusion | Write last, together with introduction | 15/10/2013 |

# Appendix E

# Data structure quotes

[54], "Predicting Time Series with Support Vector Machines", p.6.
"Our experiments show that SVR methods work particularly well if the data is sparse (i.e. we have little data in a high dimensional space). This is due to their good inherent regularization properties."

[52], "Support vector machines in remote sensing: A review", p.10.
"Most of the findings show that there is empirical evidence to support the theoretical formulation and motivation behind SVMs. The most important characteristic is SVMs ability to generalize well from a limited amount and/or quality of training data. Compared to alternative methods such as backpropagation neural networks, SVMs can yield comparable accuracy using a much smaller training sample size. This is in line with the "support vector" concept that relies only on a few data points to define the classifier's hyperplane. This property has been exploited and has proved to be very useful in many of the applications we have seen thus far, mainly because acquisition of ground truth for remote sensing data is generally an expensive process."
(And more)

[15], "Kernel-based framework for multitemporal and multisource remote sensing data classification and change detection", p.30.
"As core learners, the binary SVC and the one-class SVDD classier were used, and they were also benchmarked with neural networks in real scenarios. In general, neural networks show inferior results compared to non-linear kernel classiers, which is a direct consequence of their difculties when working with very high dimensional input samples particularly important when stacking together other information sources such as contextual or multi-temporal"

# Bibliography


[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[3] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[4] Cesare Alippi and Manuel Roveri. An adaptive cusum-based test for signal change detection. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[5] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine.

[6] Davide Anguita, Alessandro Ghio, Stefano Pischiutta, and Sandro Ridella. A hardware-friendly support vector machine for embedded automotive applications. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 1360–1364. IEEE, 2007.

[7] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–10. VDE, 2010.

[8] J. Barbič, A. Safonova, J.Y. Pan, C. Faloutsos, J.K. Hodgins, and N.S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[9] Vic Barnett and Toby Lewis. *Outliers in statistical data*, volume 3. Wiley New York, 1994.

[10] M. Basseville, I.V. Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, NJ, 1993.

[11] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, pages 57–64, 2000.

[12] Thomas Bernecker, Franz Graf, Hans-Peter Kriegel, Christian Moennig, Dieter Dill, and Christoph Tuermer. Activity recognition on 3d accelerometer data (technical report). 2012.

[13] Robert L Brown, James Durbin, and John M Evans. Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 149–192, 1975.

[14] Fatih Camci. Change point detection in time series data using support vectors. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(01):73–95, 2010.

[15] Gustavo Camps-Valls, Luis Gómez-Chova, Jordi Muñoz-Marí, José Luis Rojo-Álvarez, and Manel Martínez-Ramón. Kernel-based framework for multitemporal and multisource remote sensing data classification and change detection. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(6):1822–1835, 2008.

[16] F Chamroukhi, S Mohammed, D Trabelsi, L Oukhellou, and Y Amirat. Joint segmentation of multivariate time series with hidden process regression for human activity recognition. *Neurocomputing*, 2013.

[17] Jiun-Hung Chen. M-estimator based robust kernels for support vector machines. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 168–171. IEEE, 2004.

[18] Tsung-Lin Cheng. An efficient algorithm for estimating a change-point. *Statistics & Probability Letters*, 79(5):559–565, 2009.

[19] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. Wiley. com, 2007.

[20] Young-Sik Choi. Least squares one-class support vector machine. *Pattern Recognition Letters*, 30(13):1236–1240, 2009.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[23] Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick. Online segmentation of time series based on polynomial least-squares approximations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(12):2232–2245, 2010.

[24] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Access Online via Elsevier, 1990.

[25] Federico Girosi. An equivalence between sparse approximation and support vector machines. *Neural computation*, 10(6):1455–1480, 1998.

[26] E. Guenterberg, S. Ostadabbas, H. Ghasemzadeh, and R. Jafari. An automatic segmentation technique in body sensor networks based on signal energy. In *Proceedings of the Fourth International Conference on Body Area Networks*, page 21. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[27] Tian Guo, Zhixian Yan, and Karl Aberer. An adaptive approach for online segmentation of multi-dimensional mobile data. In *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 7–14. ACM, 2012.

[28] V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–42. ACM, 1999.

[29] Fredrik Gustafsson. The marginalized likelihood ratio test for detecting abrupt changes. *Automatic Control, IEEE Transactions on*, 41(1):66–78, 1996.

[30] Wolfgang Härdle. *Nonparametric and semiparametric models*. Springer Verlag, 2004.

[31] Zhen-Yu He and Lian-Wen Jin. Activity recognition from acceleration data using ar model representation and svm. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 4, pages 2245–2250. IEEE, 2008.

[32] Kathryn Hempstalk, Eibe Frank, and Ian H Witten. One-class classification by combining density and class probability estimation. In *Machine Learning and Knowledge Discovery in Databases*, pages 505–519. Springer, 2008.

[33] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H.T.T. Toivonen. Time series segmentation for context recognition in mobile devices. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 203–210. IEEE, 2001.

[34] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[35] Chih-Chiang Hsu. The mosum of squares test for monitoring variance changes. *Finance Research Letters*, 4(4):254–260, 2007.

[36] Carla Inclán and George C Tiao. Use of cumulative sums of squares for retrospective detection of changes of variance. *Journal of the American Statistical Association*, 89(427):913–923, 1994.

[37] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10: 1391–1445, 2009.

[38] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of 2009 SIAM International Conference on Data Mining (SDM2009)*, pages 389–400, 2009.

[39] Yoshinobu Kawahara and Masashi Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining*, 5 (2):114–127, 2012.

[40] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[41] Shehroz S Khan and Michael G Madden. A survey of recent trends in one class classification. In *Artificial Intelligence and Cognitive Science*, pages 188–197. Springer, 2010.

[42] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2): 74–82, 2011.

[43] KZ-S. Sensor logger. URL https://play.google.com/store/apps/details?id=com.kzs6502.sensorlogger.

[44] C. Li, SQ Zheng, and B. Prabhakaran. Segmentation and recognition of motion streams by similarity search. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 3(3):16, 2007.

[45] Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 5, pages 3077–3081. IEEE, 2003.

[46] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 2013.

[47] Xiaoyan Liu, Zhenjiang Lin, and Huaiqing Wang. Novel online methods for time series segmentation. *Knowledge and Data Engineering, IEEE Transactions on*, 20 (12):1616–1626, 2008.

[48] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.

[49] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1741–1745. IEEE, 2003.

[50] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[51] Michael E Mavroforakis and Sergios Theodoridis. A geometric approach to support vector machine (svm) classification. *Neural Networks, IEEE Transactions on*, 17 (3):671–682, 2006.

[52] Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3):247–259, 2011.

[53] MM Moya, MW Koch, and LD Hostetler. One-class classifier networks for target recognition applications. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1993.

[54] K-R Müller, Alex J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *Artificial Neural NetworksICANN'97*, pages 999–1004. Springer, 1997.

[55] Zineb Noumir, Paul Honeine, and Cedue Richard. On simple one-class classification methods. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 2022–2026. IEEE, 2012.

[56] ES Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[57] Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 488–498. IEEE, 2006.

[58] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels*. The MIT Press, 2002.

[59] Bernhard Schölkopf, R Williamson, Alex Smola, and John Shawe-Taylor. Sv estimation of a distributions support. *Advances in neural information processing systems*, 12, 1999.

[60] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588, 1999.

[61] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, pages 1–48, 2012.

[62] Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4): 637–649, 1998.

[63] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[64] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[65] Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *Knowledge and Data Engineering, IEEE Transactions on*, 18(4):482–492, 2006.

[66] D Tax, Alexander Ypma, and R Duin. Support vector data description applied to machine vibration analysis. In *Proc. 5th Annual Conference of the Advanced School for Computing and Imaging*, pages 15–17. Citeseer, 1999.

[67] David MJ Tax. One-class classification. 2001.

[68] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11):1191–1199, 1999.

[69] David MJ Tax and Robert PW Duin. Uniform object generation for optimizing one-class classifiers. *The Journal of Machine Learning Research*, 2:155–173, 2002.

[70] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[71] David MJ Tax and Pavel Laskov. Online svm learning: from classification to data description and back. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 499–508. IEEE, 2003.

[72] Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[73] Vladimir Vapnik. Statistical learning theory. 1998, 1998.

[74] J Weston, A Gammerman, MO Stitson, V Vapnik, V Vovk, and C Watkins. Support vector density estimation. 1999.

[75] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370, 2013.

[76] A.Y. Yang, S. Iyengar, S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari. Distributed segmentation and classification of human actions using a wearable motion sensor network. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[77] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *Knowledge and Data Engineering, IEEE Transactions on*, 20 (8):1082–1090, 2008.

[78] Shumei Zhang, Paul McCullagh, Chris Nugent, Huiru Zheng, and Matthias Baumgarten. Optimal model selection for posture recognition in home-based healthcare. *International Journal of Machine Learning and Cybernetics*, 2(1):1–14, 2011.

[79] F. Zhou, F. Torre, and J.K. Hodgins. Aligned cluster analysis for temporal segmentation of human motion. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–7. IEEE, 2008.