UTRECHT UNIVERSITY

DOCTORAL THESIS

Human Acitivity Recognition Using Accelerometer Data

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

May 2013

Declaration of Authorship

I, R.Q. Vlasveld, declare that this thesis titled, 'Human Acitivity Recognition Using Accelerometer Data' 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

Human Acitivity Recognition Using Accelerometer Data

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

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

List of Figures **vii**

List of Tables **viii**

Abbreviations **ix**

Physical Constants **x**

Symbols **xi**

1 Introduction **1**

1.1 Outline . . . 1

2 Literature review **2**

2.1 Outline . . . 2

2.2 Statistical framework . . . 2

2.3 CUSUM and GLR . . . 3

2.3.1 CUSUM . . . 3

2.3.2 GLR . . . 4

2.4 Change-detection by Density-Ratio Estimation . . . 4

2.5 Change-detection by Support Vector Machines . . . 4

2.5.1 One-class Support Vector Machine . . . 5

2.6 Change-detection by Dimensionality Reduction / Covariance structure . . 6

3 Change detection by Density-Estimation **7**

3.1 Outline . . . 7

4 Proposed method **8**

4.1 Outline . . . 8

5 Result **9**

5.1 Outline 9

6 Real-world applications **10**

6.1 Outline 10

7 Conclusion **11**

7.1 Outline 11

A Summaries **12**

A.1 Support Vector Machines 12

A.1.1 Machine learning: the art and science of algorithms that make sense of data 12

A.1.1.1 Linear models 12

A.1.1.2 Least-squares method 13

A.1.1.3 Perceptron 14

A.1.1.4 Support Vector Machine 14

A.1.1.5 Density Functions from linear classifiers 16

A.1.1.6 Non-linear models 16

A.1.2 Change Point Detection In Time Series Data Using Support Vectors 17

A.1.2.1 Introduction 18

A.1.2.2 Related work 18

A.1.2.3 Support vector based one-class classification 18

A.1.2.4 Problem formulation 19

A.1.2.5 SVCPD: The algorithm 19

A.2 CUSUM for variance 20

A.2.1 Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance 20

A.2.1.1 Introduction 20

A.2.1.2 Centered Cumulative Sum of Squares 20

A.2.1.3 Multiple changes: Iterated cumulative sums of squares . . 21

A.2.1.4 Results 22

A.3 Density Ratio Estimation 22

A.3.1 Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation 22

A.3.1.1 Introduction 22

A.3.1.2 Problem formulation and Basic Approach 23

Bibliography **25**

List of Figures

List of Tables

A.1 Support Vector machine based Change Point Detection algorithm 20
A.2 Iterated Cumulative Sums of Squares (ICSS) Algorithm 22

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
PCA	**P**rinciple **C**omponent **A**nalysis

Physical Constants

Speed of Light	c	$=$	$2.997\ 924\ 58 \times 10^{8}$ $\mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Outline

- Context of research (human activity recognition), 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

Chapter 2

Literature review

2.1 Outline

- 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:
  - Dimensionality reduction
  - Density-ratio estimation
  - Support Vector Machines (?) - if there are more sources about this
  - CUSUM - or other more traditional methods
- 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 thus has received a lot of attention in the fields of data mining, etc... *** list and refs ***. Often this problem is formulated in a statistical framework, by inspecting the data generating PDF (Probability Density Function) 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 CUSUM (cumulative sum) method by Basseville *et al.* [1], and the GLR (Generalized Likelihood Ratio) by Gustafsson [2, 3] take this approach. The former originates from control methods for detection from bench marks. The latter compares the logarithm of the likelihood ratio over two consecutive intervals. These two methods is discussed and analyzed in section 2.3.

The GLR method, as with others, relies on pre-specified parametric model assumptions and considers the data be independent over time, which makes it less flexible to real-world applications. The proposed methods by Kawahara *et al.* [4] and Lui *et al.* [5] 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, as with the GLR, rely on the log-likelihood ratio between PDFs. The method of Camci [6] takes an other approach within the statistical framework, by using a SVM (Support Vector Machine). 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 kernel trick. A change in the PDF is represented by a change in the radius of this sphere. Section 2.5 discusses the SVM-method.

The final method under consideration, change detection via (intrinsic) dimensionality reduction, takes a different point of view. Opposed to the other discussed methods, dimensionality reduction is framed in the MDL (Minimum Description Length) framework. It uses the estimated underlying number of parameters of the time series as a model for change detection. Section 2.6 discuses this method.

2.3 CUSUM and GLR

*** read book Basseville [1] ***

2.3.1 CUSUM

Non-Bayesian change detection algorithm (thus: no prior distribution believe available for the change time).

The CUSUM (cumulative sum) method is developed by Page [7] for the application of statistical quality control (it is also known as a control chart).

Primary for detection of mean shift.

The MOSUM of squares test for monitoring variance changes [8].

Use of Cumulative Sums of Squares for Retrospective Detection of Changes of Variance [9]

2.3.2 GLR

Also: maximum-likelihood estimation. "When applied to a dataset and a given statistical model, it provides estimates for the model's parameters."

2.4 Change-detection by Density-Ratio Estimation

Formulate the problem of detecting change in the statistical framework. Consider the probability distributions from which two consecutive segments of time series around a target time point are generated. When the disitrubtions differ significantly the target time point is regarded as a change point.

CUSUM (cumulative sum) [1] and GLR (generalized likelihood ratio)

The distribution over the values of time series data can be represented with a probability density function (pdf). Two sections of a time series data can be generated with the same underlying pdf or each with a different.

2.5 Change-detection by Support Vector Machines

Introduced by Vapnik [10, 11], Support Vector Machines offer a way to segment, and classify, linear separable data. When combined with a mapping function, which maps the data from the input space I to a higher dimension feature space F, the input data can be non-linear separable. The linear hyperplane, which segments the data in the feature space F, yields to a non-linear segmentation in the lower-dimensional input space I. Instead of explicitly mapping the input data to the higher dimensional space, a kernel function can be used. This kernel function can calculate values of the feature space directly, without the need to first map the input values to this space. This process is referred to as the kernel trick.

*** Let sigma be a mapping from I to F such that the dot product in F can be computed using some simple kernel ***

2.5.1 One-class Support Vector Machine

*** Replace with / add information from summary in A.1.1.4. ***

The proposed method of Camci [6] uses a one-class support vector machine to segment time series data. One-class SVMs are used to describe the current data under consideration, by assuming all data points are from the same class [12]. The class is described by a spherical boundary around the data with center c and radius r, such that the volume is minimized. Following the definition of Camci [6], the class description is obtained by minimizing r^2:

$$\text{Min } r^2 \tag{2.1}$$

$$\text{Subject to} : \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 \ \forall i,\ \mathbf{x}_i : i\text{th data point} \tag{2.2}$$

To be able to handle outliers in the input data, a penalty cost function ε_i for each outlier can be added.

*** Add new function and constraints? ***

Using this one-class SVM formulation, differences between two (consecutive) windows of data points with size w can be obtained. The first window is used as the input set, h_1 and the second as the test set h_t. For the first window a one-class SVM is constructed, yielding in a representation by c_1 and r_1. When the data points of the second window belong to the same class, the representation of that one-class SVM would equal the first:

$$c_1 = c_2, r_1 = r_2 \tag{2.3}$$

*** First tell more about (underlying) probability density functions, to relate to other methods ***

In case the second window of data points does not belong to the same class, i.e. the probability density function that describes the data differs from the first, the describing values of the second window will differ from the first. The amount of difference can be expressed by a dissimilarity measure over the representations. When the dissimilarity between the two windows exceeds some predefined threshold th, there exists a change point between the windows.

This process can be visualized as done in *** insert figure of four circles ***. The second window, h_2 can be constructed from the first by e.g. a shift of one data point. *** explain data point positions by circle ***.

Note that a difference in the SVM center c or radius r represent a change in the mean and variance, respectively.

2.6 Change-detection by Dimensionality Reduction / Covariance structure

Chapter 3

Change detection by Density-Estimation

3.1 Outline

- In-depth analysis on one of the methods of Chapter 2
- This method (e.g. Density-Ratio estimation) will be the basis for the real research
- Explain why this methods seems worthy and interesting
- Look at problems when applied to accelerometer sensor data
- *The problems discovered here will give rise to the problem statement at the Introduction / beginning of research*
- Opens the possibility for own method

Chapter 4

Proposed method

4.1 Outline

- 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

Chapter 5

Result

5.1 Outline

- Compare proposed method with methods of Chapter 2
- 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 applications

6.1 Outline

- 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?

Chapter 7

Conclusion

7.1 Outline

- 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: [13]. 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 σ^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 η, 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 α_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 α_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:

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

This equation can be transformed with the Lagrange multipliers by adding the constraints to the minimization part with multipliers α_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:

$$\begin{gathered} \alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1, \ldots, \alpha_n}{\operatorname{argmax}} -\frac{1}{2} \sum_{i=i}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \sum_{i=1}^{n} \alpha_i \\ \text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \end{gathered} \tag{A.3}$$

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* ξ_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 $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^2 = (x_i x_j)^2 + (y_i y_j^2 + 2x_i x_j y_i y_j$, 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 $\phi(\boldsymbol{x}_i) = (x_i^2, y_i^2, \sqrt{2x_i y_i})$.

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 σ 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 [6] About segmentation with SVMs. Will be main material for section 2.5 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 ξ_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 α_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:

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

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.2. 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 [9], 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 σ^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 σ_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 κ, 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 **??**. 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 (ICSS) 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 [4], 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 Kullback-Leibler Importance Estimation Procedure (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* $\boldsymbol{Y}$:

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

where $p_{\text{te}}(\boldsymbol{Y})$ and $p_{\text{rf}}(\boldsymbol{Y})$ are the probability density functions of the reference and test sequence samples. Let t_{rf} and t_{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:

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

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 σ.

Bibliography

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

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

[3] Fredrik Gustafsson. *Adaptive filtering and change detection*, volume 1. Wiley Londres, 2000.

[4] 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.

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

[6] 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.

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

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

[9] Carla Inclan 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.

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

[11] Vladimir Vapnik. *The nature of statistical learning theory.* springer, 1999.

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

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