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. 48

BIC Bayesian Information Criterion. 48

CUSUM Cumulative Sum. 43, 48

HMM Hidden Markov Model. 56

ICSS Iterated Cumulative Sums of Squares. 42

i.d.d. independently and identically distributed. 58

KLIEP Kullback-Leibler Importance Estimation Procedure. 43, 48

OCC One-Class Classifier. 48

OC-SVM One-Class Support Vector Machine. 48, 50

RBF Radial Base Function. 61

ROC Receiver-Operating Characteristic. 47

SVC Support Vector Classifier. 57

SVDD Support Vector Data Description. 47, 48, 57, 62

SVM Support Vector Machine. 47, 50, 61, 62

Symbols

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

For/Dedicated to/To my…

*** [GRAPHICS diagram: sensors –¿ signal pre-processing –¿ feature extraction –¿ analysis [* segmentation, * labeling] –¿ conclusions] *** Give context problems, means to reach (e.g. monitoring of patients, getting conclusions over health and activity).

Chapter 1

Introduction

1.1 Literature review

This section will provide a literature review per coherent subject of this thesis.

1.1.1 Segmentation and classification

In recent years much research is done in the field of segmentation and classification of signals into human activities. Due to the nature of many applied algorithms and the field of research (computer vision, data mining, robotics), often there is no clear distinction between the segmentation and classification phase. As a result, these two subjects will be discussed together although, where possible, the different techniques for the two will be mentioned. All the reviewed papers have a global setup in common; a single- or multiple- sensory device is worn by subjects. From the raw signals features are extracted which are used to train some models. During the testing phase the same extracted features are compared with the trained models to classify the activity. Some methods rely on clear segmentations of the signals, whilst other ignore this possible feature and classify on sliding windows. The content of this section is summarized in table 1.1.

During the last ten years much research is performed, as a result of smaller and cheaper measuring devices. *** cite needed ***. With the introduction of smartphones with sensory hardware the research has shifted to these devices, as they create a more natural and ubiquitous setting for the test subject. *** cite needed ***.

A distinction can be made in the sensor setup, like wearing multiple sensors, a single sensor and the location of the sensors. Research have been performed with multiple sensors, refered to as Body Sensor Networks, as done in [6, 23, 24, 61]. Locations used include the upper arm, waist, thigh, wrist and ankle. Single sensors have be worn at different locations, including the upper arm [39], the front leg pocket [18, 30, 40, 64], the waist [41, 42, 57], the chest [2, 31], the shoulder [43] and the dominant wrist [46, 72]. Other approaches have not used body-worn sensors at all, but instead relied on other meaures, such as motion capture data [7, 76] and visual [55] and acoustic data in office surroundings [53]

In [6] the effectiveness of different body locations for sensors is discussed. It is shown that using two locations (the hip and wrist or thigh and wrist) instead of five does not significantly reduce the activity recognition. The (dominant) wrist is the preferred location when only a single sensor is used.

Some experiments have only been performed in an artificial or laboratory setting [6, 24, 30, 50, 55, 57, 61, 72], thus probably resulting in less robust trained models. In [40, 64] the experiments are performed in a uncontrolled environment. This results in a more natural setting with more noise due to free movement. It also introduces transition-states in the data, because there is no clear distinction between activities. The method of [39] explicitly filters the signals for transitions. Others have provided the test subjects with a measuring device for a longer period of time, some up to a few days [39, 46].

A more objective comparison can be made based on the segmentation and classification technique used. In [6] and [57] extensive comparisons are made between much used algorithms which are widely implemented, such as in the WEKA toolkit [28]. These include the base-classifiers Decision Trees (also used in [18, 40]), Decision Tables, K-nearest neighbors (also used [18, 64]), Support Vector Machines (also used in [16, 29, 30]) and Naive Bayes (also used in [46]). The research of [6] also implemented a number of meta-level classifiers, of which Plurality Voting resulted in the highest accuracy. A very well known and applied algorithm from the field of data mining is k-means clustering. A drawback of this method is that the number of expected clusters needs to be predetermined. To overcome this shortcoming an iterative implementation can be used, which selects the best clustering. A variant is known as Fuzzy c-means clustering, in which a data point can belong to one or more clusters, each with a degree of membership. The method of k-means clustering (or a modified version) is used in [39, 41, 76].

Very much used in machine learning problems are neural networks. *** cite needed *** Although these kind of networks typically perform well on spatial-typed problems *** cite needed ***, adjustments have been made to apply them for temporal classification. The methods of [40, 72] create a multi-level neural network to generate complex discriminators as activity classifiers. A variant of neural networks known as Kohonen Self

Organizing Maps is used in [39] to create a vector codebook, which is further processed by a k-means clustering.

Due similarity with the problem of continuous speech recognition, Hidden Markov Models have been applied to activity recognition tasks. A layered approach proved to be robust for small changes in [55]. In [62] five activities have been modeled as HMMs. The method of [21] introduces HMMs with Markov Jump Linear Systems to segment a signal, with a fixed number of models. In [42] a layer of HMMs of on top of a layer of static classifiers to estimate the current activity. This has a commonality with [43] in which HMMs are used to capture regularities and smooth activity transitions. The application of [23] constructs separate HHMs for each activity and then joins them in a single model to estimate the likelihood of an activity.

A common property of the above mentioned classification algorithms is that there is no explicit segmentation phase. Data points can be segmented a posteriori, by joining adjacent data points which have the same label. An approach in which the data points are processed per segment include [50], in which motifs over sensor data from actions are discovered. In [24] the signal is segmented (without classification) by applying adaptive threshold values on the energy of the signal. The estimates of the direct density-ratio for probability distributions are compared to create change-points in [36]. By making minimal model assumptions a non-parametric segmentation is obtained, without classification. A further application of probability distributions is discussed in [2], which uses an non-parametric Bayesian method to create an Infinite Gaussian Mixture Model to represent activities. Very interesting is the ability to recognize past and new activities without training. The method of [7] considers the intrinsic dimensionality of poses by using Probabilistic Principal Component Analysis and creates cut-points. This method also is independent of trained models and thus also is a non-parametric approach. The method of [38] creates a piecewise linear representation of the signal as thus approximates it. This does not result in a segmentation in the sense of some homogeneity over the data points, but it can support one. In contrast, the method of [31] splits and merges segments the signal in a constant number k segments using a cost function to minimize the heterogeneity of a segment.

Each research outcome is defined in some kind of correctness to segment and classify the signal in the correct and preferred manner. In many methods an error was measured as the difference between the algorithms outcome and a domain-experts ground truth. In case of segmenting the time series, the method of [24] measures the delayed number of seconds before a new segment is created. For classification the measure takes on many forms and names, such as recognition rates [6] or hit precision [2]. Confusion matrices often are used to give insight in which activities are hard to discriminate from each other,

Refs.	#	Location	Activities	Env.	Algorithms	Results
[50]	Single	Wrist	Signal motifs	Artificial data	Motif discovery, HHMs, DTW	overall 87%
[24]	Multiple	wrist, legs, waist	12 daily routine	Controlled	Adaptive threshold, segmenting	85%
[62]	Single	Unknown	walking, upstairs, falling, running, standing	Controlled	FFT, HMMs	90% - 100%
[36]	-	-	-	Artificial data	Non-parametric Density-Ratio estimation	???
many more...						

Table 1.1: Overview of researches with the focus on segmentation and classification techniques.

as for instance in [40]. The term false alarm rate is often used to indicate incorrectly new introduced elements, such as activities in [2] and [36].

The overall results of many methods are fairly high and promising. Many methods, for a fixed number of activities or an iterative approach, obtain precision up to the range from 95% to 100% [18, 30, 40, 41, 50, 62, 64]. Problems for which the number of activities is unknown perform up to 95% [2, 7]. The most problems arise from discriminating between similar activities, such as taking the stairs up or down and walking when the movement of legs or waist is considered [18, 40].

Some researchers have made their used data set explicitly public available and other data sets are just available, providing valuable labeled activity patterns. *** This should be in experiments setup? *** *** List here the datasets used ***

1.1.2 Temporal Segmentation

Many different approaches have been applied to segmenting time series data. Due to the nature of this research, only segmenting techniques with an on-line approach (or easy translation to one ***?***) will be considered. Interesting are algorithms which have been applied to multi-dimensional data, since we are considering segmenting accelerometer data provided by three-axial sensors. Nonetheless, online segmenting algorithms on single dimensional data will also be considered.

*** Piecewise linear approximation ***

Segmentation methods can roughly be categorized in three methods in the way the data is processed, as discussed by Avci *et al.* [5]:

- **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. Because of the simple approax, the data is only processed very locally which can yield in poor results [38]. The complexity is equal to the bottom-up approach, $O(nL)$, where L is the average segment length.

- **Sliding Window And Bottom-up** (SWAB), as introduced by Keogh *et al.* [38], 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 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.* [38] 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.* [24] and an adaptive CUSUM-based test by Alippi *et al.* [4], 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.* [10] 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.* [25] 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). 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 [22], 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 a stochastic process *** is this a correct term? ***. Probabilistic models can be constructed on that signal, yielding in probabilistic and Bayesian based segmentation methods. The CUSUM-method relies on the log-likelihood ratio 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.* [36] proposes to estimate the ratio of probability densities (known as the *importance*) directly, without explicit estimation of the densities. 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.

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.* [7] 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.

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 [26] 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.

*** TODO *** aan het einde van de lopende tekst richting-zinnen maken, die het einde aanduiden of een erg beknopte samenvatting geven / conclusie van literatuuronderzoek.

1.1.3 List of papers

The following is a list of used papers. This list is for my own personal convenience. It is possible this will be in tabular form in the final thesis document.

*** NOTE: this list will not be in the final thesis (in this form at least) ***

- "Discovering characteristic actions from on-body sensor data" [50], cited: 63. Motifs, HHM, DTW, 96%, overall 87% accuracy. Discovers motif (actions) from data. Worn on wrist, single. Environment artificial.
- "Recognition of human activities using layered hidden Markov models" [55], cited: 3. HMM, layered (primitive and abstract actions). Uses vision for workplace-activities. Direct classification. No sensors. Artificial environment for region labeling.
- "Accelerometer-Based Gait Analysis, A survey", [16], cited: 4. Compares methods to distinguish walking of normal, fast, slow. Focus on gait, but compares classification methods such as SVM, PCA, KSOM.
- "An Automatic Segmentation Technique in Body Sensor Networks based on Signal Energy", [24], cited: 13. Automatic segmenting, adaptive threshold. Pure segmenting, no classification. Nice and clean method? Weakness: single action segmented as multiple. Used multiple sensors.
- "Towards HMM based Human Motion Recognition using MEMS Inertial Sensors", [62], cited: 13. Uses HHM, Fourier transform for features. 5 fixed activities, trained HMM. Correct rates from 90% to 100%. Classification, not just segmentation. Single sensor.

- “Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation”, [36], cited: 41. Non-parametric approach, online method, no strong model assumptions. Uses multiple datasets, origin referenced. Much follow-up work done. Artificial datasets. (no sensors)

- “Unsupervised, Dynamic Identification of Physiological and Activity Context in Wearable Computing”, [39], cited: 106(!). Combination of KSOM, iterative K-means clustering, transient states removal (using Markov model). On-line algorithm. Single sensor on upper arm. Tested for a few days and controlled environment.

- “Activity Recognition using Cell Phone Accelerometers”, [40], cited: 122 (!), 2011. Uses WEKA: decision trees, logistic regression and multilayer neural networks. Results up to 98% (jogging). Much difficulty with stairs up and down. Single accelerometer, front leg pocket. Offline method. Many recent references. Controlled environment.

- “Activity recognition from accelerometer data”, [57], cited: 434 (!), 2005. Compares 18 (offline) classifiers, base level (WEKA) and meta-level. Single device, pelvic region. Controlled setting.

- “Activity recognition from user-annotated acceleration data”, [6], cited: 1008 (!!), 2004. Five bi-axial sensors. Decision tables, instance-based learning, C4.5 (trees, highest accuracy 89.30%) and naive Bayes (weka), twenty activities. Controlled environment, common room.

- “Using Hierarchical Clustering Methods to Classify Motor Activities of COPD Patients from Wearable Sensor Data”, [61], cited: none, 2005. Uses Linear Discriminant Analysis for cluster classification (using Cluster Quality Index to determine k) and simple rule-based separation for high level ambulatory. Hierarchical Dendogram to merge clusters when similarity to high. Sensors on forearms and legs. Controlled clinical environment.

- “Non-Parametric Bayesian Human Motion Recognition Using a Single MEMS Tri-Axial Accelerometer”, [2], cited: none, 2012. Recognizes the number of human activities, single sensor on the chest. No training data. Infinite Gaussian Mixture Model, collapsed Gibbs sampler. Compares with parametric Fuzzy C-mean (data point belongs to multiple clusters), unsupervised K-means, non-parametric mean-shift. Outperforms all significantly, high hit rate and low false alarms.

- “An Online Algorithm for Segmenting Time Series”, [38], cited: 487, 2001. Reviews algorithms to get a piecewise linear representation, proposes a novel sliding-window and bottom up approach, on-line. Mere segmentation. No classification; only approximation of signal. (Dataset available).

- “Segmenting Motion Capture Data into Distinct Behaviors”, [7], cited: 242, 2004. On-line methods: cut-points on increased intrinsic dimensionality, distribution of poses is observed to change. Batch: cut when consecutive frames belong to different Gaussian mixture models. Fixed fourteen motions, compared with manual segmentation. Uses PCA and Probabilistic PCA (models the non-pc subspaces as noise). PPCA is the best. (paper anne)

- “Aligned Cluster Analysis for Temporal Segmentation of Human Motion”, [76], cited: 54, 2008. Uses extension on kernel k-means clustering and Dynamic Time Alignment Kernel (kernel of DTW) for temporal invariance, robust temporal matching metric. Coordinate descent algorithm solves ACA. Fixed number of clusters, trapped in local minima. (paper anne). Uses motion capture data.

- “Bayesian Nonparametric Methods for Learning Markov Switching Processes”, [21], cited: 10, 2010. Uses HMM for state-space model for segmentation, Markov Jump Linear systems. Unbounded number of Markov modes (parameters). Number of models is fixed, though? (paper anne). No sensors.

- “Layered Representations for Human Activity Recognition”, [53], cited: 215, 2002. Layers of Hidden Markov Models HMM, multiple levels of granularity (Based on intuition) and context. Less retraining, only lower models. Classified also. No sensors, visual and akoustic data in office surroundings.

- “Time series segmentation for context recognition in mobile devices”, [31], cited: 151, 2001. Splits and merges segments, while keeping k constant using cost function on segments for internal heterogeneity. Multiple instances determine number of k. Microphone and accelerometer data, in front of chest. Artificial data

- “A practical approach to recognizing physical activities”, [42], cited: 259, 2006. Uses method of [43], activity classification algorithm. Selects most useful features and then recognizes walking, sitting, etc. First layer static classifier on features or data (energy, mean, variance, correlation, etc), then a layer of HMM to estimate activity. All offline. Shows trained model is robust to location of wearing on body.

- “A hybrid discriminative/generative approach for modeling human activities”, [43], cited: 252, 2005. Used in method above. Combines boosting to select and reduce useful features and learn static classifiers with HMM to capture regularities and smooth activities (quick switching is unlikely). Single sensor, multiple measures (accelerometer, audio, etc), worn on the shoulder.

- "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers", [72], cited: 67, 2008. Separates dynamic from static activities. Uses multilayer feedforward neural networks to generate complex discriminating surfaces as activity classifiers. Feature subset selection approach is developed. Neural pre-classifier with constant threshold criterion. Uses Common Principal Component Analysis. Recognizes eight activities with 95% overall accuracy. Laboratory environment. Worn on dominant wrist.

- "Single-accelerometer-based daily physical activity classification", [46], cited: 46, 2009. Uses Naive Bayes classifier, sensor worn on wrist. Five activities. PCA to reduce dimensions and create independence for NB. Comparable results as Decision Trees, but with flexibility to add activities. Performed outside laboratory, 10 hours long.

- "Distributed Continuous Action Recognition Using a Hidden Markov Model in Body Sensor Networks", [23] cited: 17, 2009. Single HMM, sensor network cluster movements, HMM constructs continuous actions using postures and actions, merges left-to-right HMM for each action to a single. Transcript generation per sensor uses Gaussian Mixed Model, multiple models with m mixtures are trained with Expectation-Maximization (EM), best is chosen.

- "Offline and online activity recognition on mobile devices using accelerometer data", [18], cited: none, 2012. Compares offline and online methods, cellphone in pocket. Reduces feature extraction (mean and standard deviation (for resources of cellphone). Training is done by database and labeling, classification with K-nearest neighbors, decision trees (C4.5) and decision rules. Online and offline KNN-3 is best, 99% and 97%. Six daily activities. Trousers pocket.

- "Recognizing human activities user-independently on smartphones based on accelerometer data", [64], cited: none, 2012. Classifiers K-nearest neighbors and QDA (quadratic discriminant analysis). Online recognition implemented on phone. five daily activities. Model training by decision tree (active/inactive). Results for QDA, online: 95%. Activities performed outside. Trousers pocket.

- "Activity recognition from acceleration data based on discrete cosine transform and SVM", [30], cited: 25, 2009. Uses Discrete Cosine Transform (DCT) to get features, PCA to reduce them an multiple SVMs to classify windows. Four daily activities, 97% precision. Trousers pocket. Laboratory setting.

- "Physical Activity Recognition Using a Single Tri-Axis Accelerometer", [41], cited: 16, 2009. Five daily activities, FFT, fuzzy c means classification, offline. 99% accuracy. No smartphone, other device. Outside, although restricted (standing, sitting, lying, walking, running)

1.1.3.1 Pure segmentation

- "Bayesian Online Changepoint Detection", [1], cited: 67, 2007. Uses a priori and posteriori distribution prediction of the current run-length. Online. Uses well-drilling data, Dow Jones return value and coal-mine accidents rate, which are all used by others. Does not compare results. Decouples algorithm from model (?).

- CUSUM Method: piece wise segments of gaussian mean with noise. More refs needed.

- "An Automatic Segmentation Technique in Body Sensor Networks based on Signal Energy", [24], cited: 13. Automatic segmenting, adaptive threshold, standard deviation, niet bayesian. Pure segmenting, no classification. Nice and clean method? Weakness: single action segmented as multiple. Used multiple sensors.

- "Segmenting Motion Capture Data into Distinct Behaviors", [7], cited: 242, 2004. On-line methods: cut-points on increased intrinsic dimensionality, distribution of poses is observed to change. Batch: cut when consecutive frames belong to different Gaussian mixture models. Fixed fourteen motions, compared with manual segmentation. Uses PCA and Probabilistic PCA (models the non-pc subspaces as noise). PPCA is the best. (paper anne). From computer graphics / motion capture data.

- "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey", [5], cited: 22, 2010. Discusses segmentation as form of sliding window, top-down, bottom-up or swab. Interesting overview of five steps: preprocessing, segmentation, feature extraction, dimensionality reduction and classifiers. Classifiers only with fixed number of activities (no non-parametric).

- "Event Detection From Time Series Data", [26], cited: 291, 1999. Iterative (online, incremental) method to fit model to segment, likelihood to determine if partitioned further (number of change points), when batch. Maximum likelihood methods. Also model selection, loss and risk functions. With incremental (online) method, every new data point had likelihood criteria of being a new segment or current one. If no segment for long time, computations become increasingly expensive. Solution: sliding window for last w points only.

- “An adaptive approach for online segmentation of multi-dimensional mobile data”, [25], cited: 1, 2012. Adaptive model for selecting suitable dimensions to build model and select model (linear, quadratic, cubic). Multiple metric for results evaluation; traditional regression error, information retrieval (precision, recall, F-measure) and segmentation delay. Uses three-axis accelerometer data. Extends the SWAB method. Natural setting, no restriction on body location.

- “Activity Recognition on 3D Accelerometer Data (Technical Report)”, [10], cited: 0, 2012. First makes distinction between periodic and non-periodic activities, directly on accelerometer data. Autocorrelation as a measure for self-similarity (periodicy). After segmentation, features are extracted for clustering. Then dimensionality reduction and (re)classification. Interesting post-processing; small segments with neighbors of the same activity are merged. Removes small mis-activities.

- “Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation”, [36], cited: 41. Non-parametric approach, online method, no strong model assumptions. Uses multiple datasets, origin referenced. Much follow-up work done. Artificial datasets. (no sensors) Uses likelihood/density ratios estimator, not the densities themselves. Probability distributions that generate the time series.

- “Adaptive Filtering and Change Detection” (book), [27], cited: 682 (!!), 2000. Lots of algorithms, hard to read.

- “Online segmentation of time series based on polynomial least-squares approximations”, [22], cited: 17, 2010. Creates higher-order polynomial model of data, sliding and growing window. Works well for traditional time series (stocks), but not for non-stationary such as daily life.

- “Novel online methods for time series segmentation”, [45], cited: 17, 2008. (Stepwise) Feasable Space Window. Compares with SWAB. Referenced in [25], but that one choses SWAB because it should be lower time complexity and fast. (S)FSW creates less segments, while SWAB has lower error. FSW is fastest and reliable (due to this paper self).

- “An adaptive cusum-based test for signal change detection”, [4] cited: 16, 2006. CUSUM-test with adaptive parameter settings. Can detect non-stationary processes by drift of signal. Applicable for extracted features, e.g. mean.

1.2 Problem statement

Chapter 2

Techniques

2.1 Signal pro-processing and sensor fusion for timed patterns

Describe how data is collected and processed, the form of the data (continuous to discrete, amplitude, energy, frequencies, etc). Include feature extraction, like PCA, FFT, energy, mean, but discuss it further in subsection. Only the concepts will be explained and the scheme of the computations (or simply formulas), no precise implementations. Mention only shortly with meaning, no further explanation.

———

2.1.1 Signal fusion

Signal fusion is a broad field of research and application and many different interpretations exist. The function and interpretation used in this thesis follows the definition from [19], stating:

> **Sensor fusion** is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually.

This can essentialy be considered as a form of synergy, in which the combination of the sensors provide more information than the sum of the raw output. For the application of this thesis we will focus on direct fusion of raw data, which can be applied to both homogeneous and heterogeneous sensors. The fusion stage will combine the data into a single representation, which can be interpreted by the controlling system. This differs from mere multisensor integration, in which the controlling system is responsible for processing and interpreting the data from different sources. Figure 2.1 shows the difference.

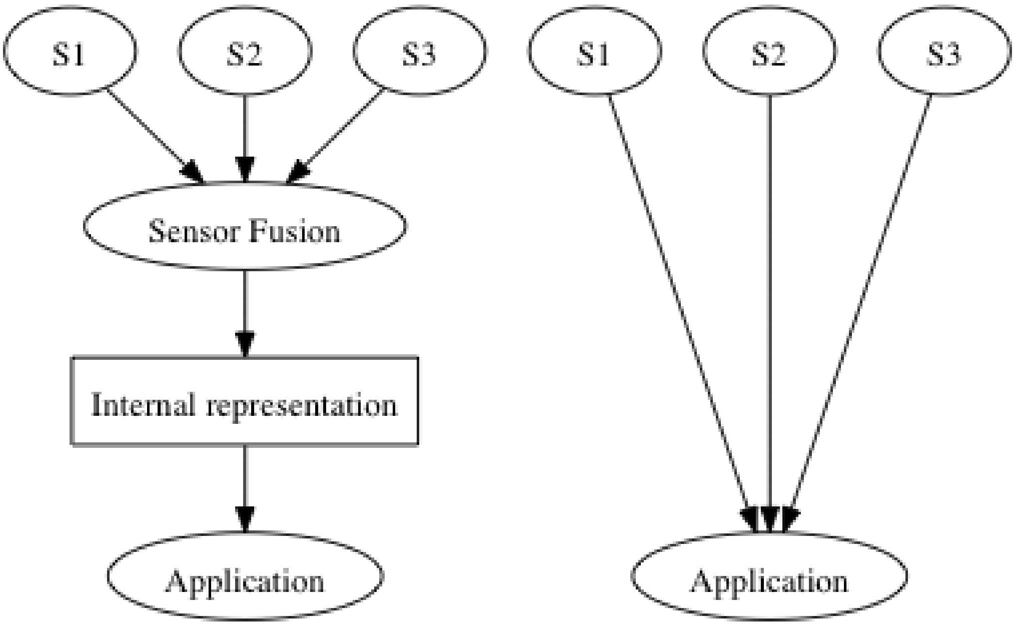

FIGURE 2.1: On the left *Sensor Fusion* and on the right *Multisensor integration*

Using the single representation, the computational and system complexity can be reduced. The representation forms a standardized and normalized interface from the sensors to the system, which create independence and modularity of sensors. Other benefits as a result from sensor fusion include [19]:

- **Robustness en reliability:** When the system incorporates multiple sensors as redundancy, the system becomes more stable in case of partial failure.
- **Extended spatial and temporal coverage:** With multiple sensors a broader spectrum can be analyzed, both in the case of homogeneous and heterogeneous combinations.
- **Increases precision:** Sensors typically have a range and domain of measurements. By combining a set of heterogeneous sensors, the overall range of observervation is increased.
- **Reduced ambiguity and uncertainty:** In case of high uncertainty about a observation, the observation of other sensors can confirm it.

Sensor fusion can be applied on multiple levels. A rough diversion over three levels is:

- **Low-level:** Raw sensor data is combined which is considered to be more informative than the original sources.
- **Intermediate-level:** Also known as *feature fusion* combines the features which can be extracted from the data. See also section 2.1.3. This data would be applicable to segmentation ***[???].
- **High-level:** Fusion of this level will support decision making, optionally followed by actions taken by the system to the environment or conclusion being drawn from the environment.

2.1.2 Signal smoothing

What, why and how.

2.1.3 Feature extraction

What, why and how.

2.2 Temporal Segmentation

This section will give an introduction and in-depth analysis of temporal segmentation.

2.2.1 Aims of segmentation

When processing and analyzing time series of data, e.g. motion measurements, stock market fluctuations or natural language, first a low-level division between the discriminative parts of the stream must be made. One can view this as splitting the series into the *atoms*, which are the building blocks of the total stream. These building blocks will be the aggregation of non-overlapping, internally homogeneous segments [31]. This means that the data points inside a segment should have some resemblance relation to each other and their difference lies between some boundary. The process of segmenting can be viewed as a subproblem to context analysis of time series. Temporal segmentation is closely related to temporal clustering, although it is a stricter, and simpler, process. Whereby clustering only restricts the data points on their distance relation (as used in a Voronoi diagram), within a segment the data points must also be contiguous.

The task of segmentation can be performed in a manual matter, by cutting and labeling parts of the stream into coherent parts. This would require human (expert) knowledge and does not yield a clear cut because of ambiguity. With increasing storage abilities and easier motion capture systems, there is a desire for automated systems which perform the segmentation task unsupervised. Some algorithms used have the (often desired) side-effect of also clustering the segments, such that classes of segments can be discovered in the time series. These algorithms would not only be able to make a distinction between walking, sitting and walking, but would also recognize the reappearance of the walking activity.

2.2.2 Formal definition

Formally, temporal segmentation is dividing a time series s, which consists of N samples $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)$ from $\mathbf{R}^d$. Individual *segments* are referenced by $s(a, b)$, consisting of the consecutive samples $\mathbf{x}(a), \mathbf{x}(a+1), \ldots, \mathbf{x}(b)$, $a \leq b$. Let $s_1 = s(a, b)$ and $s_2 = s(b+1, c)$ be two segments, then their concatenation is $s_1 s_2 = s(a, c)$. A segmentation S of s consists of a sequence of k non-empty segments $s_1 s_2 \ldots s_k = s$. This notation is adopted from [31].

As stated, informally each segment should be internally homogeneous. This can formally be measured with an cost function F, indication the heterogeneity of a segment. The overall aim is to minimize the cost F. The cost of a segment is a function from the data points and the number of data points $n = b - a + 1$ and is expressed as

$$\text{cost}_F(s(a, b)) = F(\mathbf{x}; n | \mathbf{x} \in s(a, b)) \tag{2.1}$$

The cost of a *k-segmentation* S is the summation of the costs of the k segments:

$$\text{Cost}_F(s_1 s_2 \ldots s_k) = \sum_{i=1}^{k} \text{cost}_F(s_k) \tag{2.2}$$

With the objective of minimizing the cost function, the optimal k-segmentation $S_F^{opt}(s; k)$ is the segmentation with minimal $\text{Cost}_F(s_1 s_2 \ldots s_k)$ over all possible k-segmentations.

The cost function, to calculate the heterogeneity of a (set of) segment(s), can be any function. A simple and natural function would be the sum of variances of the segments. The overall cost function would then be

$$\text{Cost}_V = \frac{1}{N} \sum_{i=i}^{k} \sum_{j=c_{i-1}+1}^{c_i} \|\mathbf{x}(j) - \mu_i\|^2 \tag{2.3}$$

where μ_i is the mean vector of data points in segment s_i.

2.2.3 Application in research fields

[CHARACTERISTICS OF HUMAN MOTION] temporal variability, invariance over time, metrics over actions.

[COMPUTER VISION]

[GRAPHICS/VIDEO]

[DATA-MINING]

[MODEL BASED]

To analyze time series it is often preferred to divide the stream in segments of correlated data. After dividing, each segment represent a period in time in which the same activity is performed. Or, stated otherwise, it results in transitions moments between activities.

2.2.4 PCA Based Methods

Many fields of research have been active in the unsupervised segmentation of data. Many authors rely on a form of Principal Component Analysis (PCA), as used a.o. in [7]. Often PCA is used to reduce the dimensionality of the data being processed [REFERENCE] by only using the top r dimensions to describe the data set. It is observed that data series of simple motions have a lower dimensionality then complexer motions. When a simple (repetitive) motion is about to end and fluently transforms in a new motion, there will be a window of time in which a high dimensionality will be present, due to the new motion. After this period of transition, the dimensionality will decrease, since only the new simple motion is present in the window of time. The first algorithm of [7] is based on this principle.

Given a set of data points, a lower dimensional hyperplane can by constructed to which the data points can be projected. This projection on a lower dimension introduces a error to the original position. When the error is fixed, less dimensions are needed for simple motions in which movements of body parts are highly correlated. For segments in which the data points are lesser correlated, e.g. because of transition state, a higher degree of dimensions of the hyperplane is needed to represent the data with equal error degree. When the dimensionality is reduced from d to $r \leq d$, the ratio of error E_r can be calculated as

$$E_r = \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_{j=1}^{d} \sigma_j^2} \tag{2.4}$$

where σ_j are the singular values as a result from Singular Value Decomposition (SVD), which is closely related to PCA [63].

In [7] the stream of frames is analyzed on cut-frames to find transitions between action. First, for a number of k frames (e.g. the equivalent of 2.5 seconds) the required dimensionality r to keep the error E_r below some threshold τ is calculated. This will yield in an error e_i for the first i frames. When more frames are added to the window, the error will increase with a low constant when it is still in the same activity, due to noise in the activity. When a new activity starts, e.g. at frame j, the error at frame j will start increasing faster. This can be expressed by the derivative of the error rate $d_i = e_i - e_{i-l}$, where l is a constant to remove noise. From this derivative the mean and standard deviation can be calculated, for each point. When a derivative d_j rises more than a factor $k_\sigma = 3$ standard deviations from the mean, a transition point is encountered. The previous frames are then cut from the sequence (as a segment) and the algorithm starts over.

*** [figure to illustrate derivative and standard deviation]

A second approach in [7] uses the probabilistic variant of PCA (PPCA) to model the data set as a Gaussian distribution instead of ignoring the frames which do not fit in the subspace. Over windows of frames the mean and variance are calculated. In a forward manner the Mahalanobis distance of a new window of frames is calculated, which represents the likelihood of the new window belonging to the same segment as the original widow. When the distance decreases, the likelihood increases which happens when the motions in the becomes more homogeneous. When a peak in the distance is reached, the new window of frames indicates a heterogeneous collection of motions in the window and thus a low likelihood of membership and a indication of a transition. In order to distinct activities and sub-activities (which require a subset of motions is a distinct activity) the algorithm is also processed backward over the data series.

*** [figure to illustrate mahalanobis distance measure and peaks]

The third algorithm in [7] is based on the observation that data points (frames) tend to form clusters in the space. These clusters are represented by k Gaussian distributions for which each the Expectation-Maximization (EM) algorithm estimates the mean m_j, covariance matrix $\sum_j$ and prior π_j. With all the Gaussian distributions estimated, the data points are assigned to the cluster with the highest membership likelihood. When two consecutive frames x_i and x_{i+1} belong to different clusters, a transition of activities is recognized. Note that this algorithm succeeds in segmenting the data and also labels the similar simple activities.

A drawback in this system, and many others which implement a variant of the k-means algorithm, is that the number of clusters k need to be predetermined. To cope with this, often the algorithm is performed multiple times for different values of k. Using some criteria, e.g. the Bayesian Information Criterion [54] or the Davies-Bouldin Index which guides k-means clustering as used in [39].

2.2.5 Statistical methods

An method to process data, or create a basis for further processing, is by analyzing the statistical properties of a data set. When there is a continuous stream of data, it is possible to extract events by comparing the statistical properties of sliding time windows. By definition, events have a different statistical profile then their background. When two consecutive actions are regarded as the background, then a transition is an event between them.

In [24] a automatic system is used to segment a continuous stream of data. By calculating features as activity level from the standard deviation of a sliding window and applying Adaptive Thresholds, the data can be segmented in active and rest parts. There is no model representation.

2.2.6 Hidden Markov Models Based Methods

2.2.7 Bayesian Methods

2.2.8 Principal Component Analysis

When working with simple 2- or 3-dimensional data it is often, for humans, easy to discover patterns in the set. The data can be plotted and the lines or planes among which the data points lie gives an indication of the pattern. With Principal Component Analysis (PCA) this process is also possible for automated systems. With this method the overall form of a set of data points can be represented, the most discriminative dimensions can be found, the dimensionality of a set can be reduced (which yield in data compression) or the result can be used to characterize and differentiate sets of data points.

The following steps are performed, which are explained below and illustrated in figure *** [add figure with plot of sets]:

1. **Gather data** and represent them in a chosen number of features,

2. **Subtract mean** to center the data around the axis. The new data set will have mean zero,

3. **Generate covariance matrix** by calculating all pairwise feature variances,

4. **Calculate Eigenvectors and Eigenvalues** of the covariance matrix. The Eigenvectors are then normalized to make further calculations easier,

5. **Generate feature vector** which indicated which features are characteristic or meant to keep in de data set, by comparing the Eigenvalues.

The workings of the PCA [65] relies on the concepts of standard deviation, variance, covariance, eigenvectors and eigenvalues of matrices. These concepts will be discussed very briefly [1]. The standard deviation and variance of a set are measures for the spread around the mean for a single dimension, or feature. The covariance between two features (dimensions of the data points) indicates how they are related; when positive the two features will increase together; when negative one will decrease when the other increases and when zero they are unrelated. The covariance matrix gives all the covariances for all pairs of features. This matrix is symmetrical about the diagonal and the values on the diagonal are the variances for each feature. For this matrix the eigenvectors can be computed. Eigenvectors have the characteristic that when they multiply a matrix, the resulting vector is a multiple of the Eigenvector. The amount by which it is the multiple is the Eigenvalue. This is illustrated in formulae 2.5 and 2.6. The second formula shows an Eigenvector $\binom{3}{2}$ and its associated Eigenvalue, 4. The vector in the first formula is not an Eigenvector.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 + 3 \cdot 3 \\ 2 \cdot 1 + 1 \cdot 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix} \tag{2.5}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 3 + 3 \cdot 2 \\ 2 \cdot 3 + 1 \cdot 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{2.6}$$

It are these Eigenvectors and Eigenvalues that makes PCA possible. Given a matrix of size $n \times n$ (only square matrices have Eigenvectors), n Eigenvectors can be found. Each Eigenvector is perpendicular, or orthogonal, to the others. The Eigenvectors combined describes the lines over which the data is plotted. The Eigenvector with the largest Eigenvalue is the *principal* component of the data set. It is said that it is the most significant feature. Thus the Eigenvectors, and thereby the features or dimensions of the data set, can be sorted on significance by the Eigenvalues.

[1] For a more in-depth discussion we would like to refer the reader to [35]

With this ordering there are a few applications possible. The first is to just make a (comprehensible) representation of the data. The Eigenvectors describe the cloud of data points and thus are a compressed representation of the data. When the original data points are to be compressed, it is possible to remove the least significant features and reconstruct the data points from the resulting Eigenvectors. This will yield in a lossy compression. *** [figure to illustrate]. An extension of this application is to determine the number of features needed to keep the compression within a certain error criterion, as used in [7]. Sets of data can then be distinguished by the number of features needed to describe the points.

*** [graph, a bit of formulae with sets and dimensions]

2.2.9 Hidden Markov Models

Many dynamic systems can be viewed as some sequence of consecutive states which are observable by the world. These kind of systems are known as Markov Chains. When the state in which the system is can not be observed with complete certainty, a system known as a Hidden Markov Model (HMM) can be constructed. It has been shown that HMMs are effective on in the task of speech recognition [56]. Because of some resemblance between natural speech construction and human activities, modifications have been able to classify human activities [23] and extensions are implemented which create layers of HMMs to handle different granularity over time and reduce retraining [53, 55]. A HMM is a state-space model representing a single model for each label which needs to be classified.

Formally, a Hidden Markov Model is characterized by the following elements [56]:

1. A number of N states in the model. These states are a representation of the underlying mechanism of the system and can not be directly observed from outside (hence the *hidden* model). Individual states are referenced by $S = \{S_1, S_2, \dots, S_N\}$ and a single state in time t is denoted by q_t.

2. An alphabet of M discrete observable symbols for each state. An observed symbol is the only indication of the systems current situation. Each individual symbol is referenced by $V = \{v_1, v_2, \dots, v_m\}$.

3. A state transition probability distribution $A = \{a_{ij}\}$ indication the probability of being in state j at time $t+1$ when the state on time t was i:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N \tag{2.7}$$

Note that by setting the probability $a_{ij} = 0$ for some i and j indicates that j is not reachable from state i.

4. A symbol observation probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], \quad 1 \leq j \leq N \\ 1 \leq k \leq M \tag{2.8}$$

Note that every symbol can be observed in every state.

5. The initial state distribution in which the system starts $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N \tag{2.9}$$

This system is often abbreviated to

$$\lambda = \{\pi_i, a_{ij}, b_j(k)\} \tag{2.10}$$

The HMM will act as a generator giving a sequence of T observations, each being a symbol from V:

$$O = O_1, O_2 \ldots O_T \tag{2.11}$$

Given the characteristics of a specific HMM, there are three major application which serve the classification of an activity. The first application is, given a sequence of observations $O = O_1, O_2, \ldots, O_T$, to label this sequence with the activity with the highest associated probability for that label, e.g. compute $P(O|\lambda)$ for each model λ. The second task a HMM can perform is to *explain* a observation sequence O, by choosing an optimal corresponding state sequence $Q = q_1, q_2, \ldots, q_T$. The last application is more of a problem statement, being the task to train the HMM to adjust the parameters $\lambda = \{\pi_i, a_{ij}, b_j(k)\}$ to maximize $P(O|\lambda)$. In the overall task of activity recognition, one would first use the last application to train a set of HMMs with the provided and labeled training data. The second application can be used to further optimize the HMM (set the number of states, adjust vector codebook etc.) and improve the capability of modeling the activities. Finally, when the system is provided with unlabeled data the first application is used to identify the observed sequence with trained examples, by comparing the probability result of multiple HMMs.

Generally a HMM can be fully connected, or ergodic model. Other configurations are also possible, of which the left-right model, or Bakis model, is popular for temporal applications [56]. This because the structure of the state probability distribution resembles the construction of possible states over time. This left-right model can be generalized by setting

$$a_{ij} = 0, \quad j > i + \mathit{\Delta} \tag{2.12}$$

for some $\mathit{\Delta}$, indicating the possible *distance* of the jumps over states.

*** [INSERT GRAPHICS SHOWING STRUCTURE OF HMM]

*** [discrete and continuous HHMs]

*** Applications of HHM: [62] uses 10 coefficients extracted with FFT from 6 axis (3 times accelerometer and 3 times gyroscope). Example of continuous HMM (?). Recognition rates fo 50%-100%.

In [53] it is stated that HMMs are robust mechanisms with respect to changes in temporal segmentation of observations, although they suffer from a few problems when applied to reason about longer and complexer sequences over time. They state that, especially with limited training data, the HMM tend to be overfitted and tend to lack structure and have an excess of parameters. The overcome this limitation, they introduce a Layered Hidden Markov Model (LHHM) which they show is more robust to temporal changes in observations. They make use of the hypothesis from the field of psychology, that human behavior is hierarchically structured [74]. In [55] a layered construction of HMMs is used which feeds the low-level data from a sliding window to the lower layer. This layer will recognize simple motions, such as moving the arm in a pre-defined segment. The results from these lower HMMs are then fed to a higher-level layer of HMMs which are able to recognize abstract motions, such as adjusting a monitor and picking up a pen. The results showed that the layered system was better at recognition then a similar single-layered setup. It was also robust in the sense that when the environment was changed, only the relatively simple low-level HMMs needed to be retrained.

*** [DIAGRAM WITH EXAMPLE OF LOW- AND HIGH-LEVEL HMMs, SUCH AS MOVING ARM, PICKING UP PEN]

*** Viterbi algorithm / path; find the most likely sequence of cause-states for the observed measures.

2.2.10 Segmentation as clustering

In the previous section the discussed methods all relied on the stream of data points and tried to find cuts the discriminate between successive different type of activities. An other approach is to consider the tasks of segmentation as a type of clustering [76]. In clustering the objective is to assign labels, or classes, to all the data points indication a similar type of activity. A clustering $\mathcal{L}$ is thereby more informative then a segmentation but is also harder to produce.

A clustering $\mathcal{L}$ is generated from a sequence of elements $\mathbf{X}$ which is decomposed in m disjoint segments, each belonging to one of the k classes. A segment $\mathbf{Y}_i \hat{=} \mathbf{X}_{[s_i, s_{i+1})}$ is composed of frames from position s_i to s_{i+1}. A vector $g_{ci} = 1$ indicates class membership if $\mathbf{Y}_i$ belongs to class c, otherwise $g_{ci} = 0$.

When regarding segmentation of human motion as a task of clustering the difficulty is to model the temporal variability of actions and defining a robust metric between temporal actions. To overcome this, [76] introduces Aligned Cluster Analysis (ACA), by minimizing

$$J_{ACA}(\mathbf{G}, \mathbf{s}) = \sum_{c=i}^{K} \sum_{i=1}^{m} g_{ci} \, dist_c(\mathbf{X}_{[s_i, s_{i+1})}) \tag{2.13}$$

The characteristic of ACA is that is enables segments to span over different number of data points, whereas the standard kernel k-means algorithm results in equally sized segments. [!!! TRUE?!] The second difference is that the kernel used in $dist_c$ to measure the distance from a segment to the class which it is assigned to uses the Dynamic Time Alignment Kernel [REFERENCE?] to measure between time series.

2.3 Clustering

This section will give an introduction and overview to clustering of data. When processing time series of data the line of distinction between segmentation and clustering is very fine. This section will introduce clustering for the purpose as used in this project.

Data clustering can be used with two different applications, exploratory or confirmatory [34]. In the former, the purpose is to discover clusters in a point cloud of data points. Here the cluster is defined as a group of homogeneous data points measured by some coherence measure, often a distance function with the data points being defined as vectors. The produced clusters, which together form a representation of the data examined, can be used in the latter application to assign new provided data points to a cluster and thereby classify them.

Depended on the precise setup, clustering can be used as a successive step after or an implementation of temporal segmentation. When temporal segmentation is used to create successive coherent data points, clustering can be used to recognize the same activities in different points of time. The data points provided to the clustering will be some representation of each segment and each segment will be labeled with a activity.

Another mechanism could be to extract features from the raw data points and use these to create clusters directly *** [temporal modifications]. Both mechanisms requires the exploratory and confirmatory phases.

The exploratory phase roughly consists of the following five steps [34]

1. data point representation,
2. definition of coherence,
3. grouping data points,
4. cluster abstraction,
5. output qualification

The first step is to find a *representation* of the data points and clusters, considering the number, type and scale of the features and the number of desired classes to cluster in. When dealing with high-dimensional data points a *feature selection* can be performed to use only the most discriminating features. When the raw features are not useful enough, *extraction* can be used to create new synthetic features. *** [qualitative versus quantitative/conceptual]

When the data points are represented in a meaningful way, the measure of *coherence* between pairs of points must be defined. Often this is implemented as the Euclidean or Mahalanobis distance when the data points are represented as vectors or other similarity measures for conceptual patterns.

In the most critical step, the *grouping*, the data points with the highest coherence are linked together to form a cluster. The result can be a hard partition over the data or a fuzzy membership degree to each cluster for each data point. When applying rules for merging and splitting cluster, a hierarchical partition is constructed. Partitional clustering algorithms assign data points to clusters by optimizing some criterion, e.g. the mean squared distance from each data point to the clusters' centroid.

Especially in case of large data sets, the resulting clusters will be defined by many data points. To form a compacter and simpler representation, *abstraction* can be used. This will simply analysis by humans or automated systems. A common abstraction is to use the clusters' centroid [17] or parameters of Gaussian patterns.

The final step which can be applied to the resulting partition is some *qualification*. The quality of multiple partitions can be compared and the validity of the partition can be determined. A partition is valid if reasonably it could not been constructed by change or as some random process. External references of models can be used to compare, or the internal data points can be examined. *** [better wordings]

2.3.1 Formal definition

2.3.2 Application is research fields

2.4 Temporal pattern recognition

2.4.1 Dynamic Time Warping

Used to measure similarity between time sequences. Exact matching is high-cost, so approximations such as Minimum Bounding Rectangles are used.

2.4.2 k-means clustering

[EXPLAIN] divide data set n into k clusters. Follows the outline in figure 2.2

Among many unsupervised clustering techniques, k-means is successfully applied to large data sets. It is simple to implement and linear in time complexity so computationally attractive [34]. A drawback of the method is that the results of the algorithm greatly depends on the initial configuration (the data points which will act as centroids) and the number of cluster k must be determined beforehand.

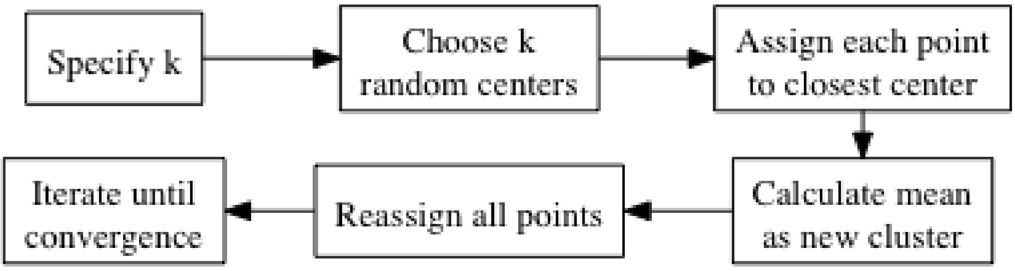
Specify k
Choose k random centers
Assign each point to closest center
Calculate mean as new cluster
Reassign all points
Iterate until convergence

FIGURE 2.2: Outline of the k-means algorithm.

Generally, the k-means methods will minimize the squared error for a clustering $\mathcal{L}$ criterion which is defined as the distance from thedata points the centroid for each cluster in $\mathcal{K}$. This is expressed as optimizing to a local optimum the energy function

$$e^2(\mathcal{K},\mathcal{L}) = \sum_{j=i}^{K}\sum_{i=1}^{n_j} \|\mathbf{x}_i^{(j)} - \mathbf{c}_j\|^2 \tag{2.14}$$

There are several limitation on the k-means method. One of these is that only spherical shapes of cluster can be generated. One of the extensions is kernel k-means [58], which implicitly projects the data points to a higher dimension and thereby is able to form irregular shaped cluster.

2.4.3 Self-organizing Map

2.4.4 Support Vector Machine

2.4.5 Naïve Bayes

2.5 Unsupervised clustering of temporal patterns

Chapter 3

Experiments

3.1

Chapter 4

Results

4.1

Chapter 5

Discussion

5.1

Chapter 6

Conclusions

6.1

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: [20]. 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:

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

This equation can be transformed with the Lagrange multipliers by adding the constraints to the minimization part with multipliers 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: *** 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* 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 [11] About segmentation with SVMs. Will be main material for section **??** 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 **??**]

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 [33], 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:

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

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 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 Iterated Cumulative Sums of Squares (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 [36], 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 Cumulative Sum (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* $\boldsymbol{Y}$:

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

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_{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:

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

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 σ.

A.3.1.3 Online Algorithm

...

A.4 Outlier Detection methods

A.4.1 A Survey of Outlier Detection Methodologies

Hodge and Austin, [32], 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 [8]):

> 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, 69] 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 [11] ("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" [70] 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 [44] it is stated that the Support Vector Machine (SVM) by Cortes and Vapnik [15] 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 Support Vector Data Description (SVDD) of Tax and Duin [68, 69] 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 One-Class Classifiers (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 [11] 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.

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

The paper of Yin *et al.* [73] makes a distinction between similarity based (using a defined distance measure) and model based (which characterize the data using predictive models) approach. The One-Class Support Vector Machine (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 [11] 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 [33] also regards each semgent as a Gaussian distribution.

The method if Kawahara *et al.* [36] is based on the log likelihood ratio of test samples, and the method by Liu *et al.* [44] uses a comparable dissimilarity measure using the KLIEP algorithm.

The method of Chamroukhi *et al.* [13] 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 Bayesian Information Criterion (BIC) or the similair Akaike Information Criterion (AIC) [3], as in [29].

*** 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 [13]?

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 [40] and [75] (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 [11]). Other approaches regard the data as a set of polynomials, which can be estimated by regression (such as [13]) 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 [73] ***

B.4 Segmentation (SVM) method

Overview of segmentation methods. Collection of papers use relative and direct density-ratio estimation [36, 44].

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 [75] "Optimal model selection for posture recognition in home-based healthcare")
- Survey [32] states RBF is similair to Gaussian Mixture model (page 25).

*** TODO: read [49] and [9] on geometry of SVMs ***

Re-evaluatie [25], 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?

*** Note: in [49] 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

*** Read [32] “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 [9, 49]?
- 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: [11]. 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:

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

Where r is the radius of a (hyper)circle with center $\boldsymbol{c}$, C is the penalty coefficient for every outlier and ξ_i is the distance from the ith 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 ξ is:

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

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: [36].

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 lth 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 σ. 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 μ:

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

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

Paper: [13]. This approach models the time series data by a parameterized regression, filtered with a Hidden Markov Model (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:

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

The observations get a label assigned by maximizing the logistic probability (π_k):

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

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

Paper: [69]. The paper proposes the SVDD method, analogous to the Support Vector Classifier (SVC) of Vapnik [70], 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: [71]. Using the notation of [71], 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 independently and identically distributed (i.d.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: [38]. 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: [47]. 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 θ_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: [48].

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

Paper: [48] This approach is very similar to the other paper of this author discussed in the preview section [47]. 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 [69].

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 ρ are determined by solving a quadratic optimization problem. The dual formulation (using Lagrange multipliers α_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 Radial Base Function (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 [37] which constructs *sequences* of samples and in contrast with [11], 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: [14]. 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 [69], 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 ϕ 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 ξ_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

[52], "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."

[51], "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)

[12], "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] M Ejaz Ahmed and Ju Bin Song. Non-parametric bayesian human motion recognition using a single mems tri-axial accelerometer. *Sensors*, 12(10):13185–13211, 2012.

[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] 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.

[6] Ling Bao and Stephen Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[7] 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.

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

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

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] M.O. Derawi. Accelerometer-based gait analysis, a survey. *Norsk informasjonssikkerhetskonferanse (NISK)*, 2010.

[17] E. Diday and JC Simon. Clustering analysis. *Digital Pattern Recognition*, 10:47–94, 1976.

[18] Andrés Duque, Fco Javier Ordóñez, Paula de Toledo, and Araceli Sanchis. Offline and online activity recognition on mobile devices using accelerometer data. In *Proceedings of the 4th international conference on Ambient Assisted Living and Home Care*, pages 208–215. Springer-Verlag, 2012.

[19] W. Elmenreich. An introduction to sensor fusion. 2001.

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

[21] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. Bayesian nonparametric methods for learning markov switching processes. *Signal Processing Magazine, IEEE*, 27(6):43–54, 2010.

[22] 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.

[23] E. Guenterberg, H. Ghasemzadeh, V. Loseu, and R. Jafari. Distributed continuous action recognition using a hidden markov model in body sensor networks. *Distributed Computing in Sensor Systems*, pages 145–158, 2009.

[24] 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.

[25] 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.

[26] 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.

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

[28] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[29] 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.

[30] Zhenyu He and Lianwen Jin. Activity recognition from acceleration data based on discrete consine transform and svm. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 5041–5044. IEEE, 2009.

[31] 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.

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

[33] 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.

[34] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[35] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[36] 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.

[37] 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.

[38] 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.

[39] A. Krause, D.P. Siewiorek, A. Smailagic, and J. Farringdon. Unsupervised, dynamic identification of physiological and activity context in wearable computing. page 88, 2003.

[40] 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.

[41] Mi-hee Lee, Jungchae Kim, Kwangsoo Kim, Inho Lee, Sun Ha Jee, and Sun Kook Yoo. Physical activity recognition using a single tri-axis accelerometer. *Lecture Notes in Engineering and Computer Science*, 2178.

[42] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. *Pervasive Computing*, pages 1–16, 2006.

[43] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 766–772, 2005.

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

[45] 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.

[46] Xi Long, Bin Yin, and Ronald M Aarts. Single-accelerometer-based daily physical activity classification. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6107–6110. IEEE, 2009.

[47] 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.

[48] 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.

[49] 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.

[50] David Minnen, Thad Starner, M Essa, and Charles Isbell. Discovering characteristic actions from on-body sensor data. In *Wearable Computers, 2006 10th IEEE International Symposium on*, pages 11–18. IEEE, 2006.

[51] 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.

[52] 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.

[53] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 3–8. IEEE, 2002.

[54] D. Pelleg, A. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, volume 1, pages 727–734. San Francisco, 2000.

[55] S. Perdikis, D. Tzovaras, and M.G. Strintzis. Recognition of human activities using layered hidden markov models. In *Cognitive Information Processing Workshop*, pages 114–119. Citeseer, 2008.

[56] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[57] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *Proceedings of the national conference on artificial intelligence*, volume 20, page 1541. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[58] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[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] Delsey Sherrill, Marilyn Moy, John Reilly, and Paolo Bonato. Using hierarchical clustering methods to classify motor activities of copd patients from wearable sensor data. *Journal of NeuroEngineering and Rehabilitation*, 2(1):16, 2005.

[62] G. Shi, Y. Zou, Y. Jin, X. Cui, and W.J. Li. Towards hmm based human motion recognition using mems inertial sensors. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1762–1766. IEEE, 2009.

[63] J. Shlens. A tutorial on principal component analysis. *Systems Neurobiology Laboratory, University of California at San Diego*, 2005.

[64] Pekka Siirtola and Juha Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *IJIMAI*, 1(5):38–45, 2012.

[65] L.I. Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

[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. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

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

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

[72] Jhun-Ying Yang, Jeen-Shing Wang, and Yen-Ping Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16):2213–2220, 2008.

[73] 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.

[74] J.M. Zacks and B. Tversky. Event structure in perception and conception. *Psychological bulletin*, 127(1):3, 2001.

[75] 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.

[76] 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.