Blind segmentation of time-series

A two-level approach

Vana Panagiotou

Blind segmentation of time-series

A two-level approach

by

Vana Panagiotou

in partial fulfillment of the requirements for the degree of

Master of Science

in Electrical Engineering

at the Delft University of Technology,

to be defended publicly on Monday October 26, 2015 at 14:00 PM.

Supervisor:	Dr. ir. Richard Heusdens ,	TU Delft
Thesis committee:	Dr. ir. Richard Heusdens,	TU Delft
	Dr. ir. Richard Hendriks,	TU Delft
	Prof. dr. ir. Geert Leus,	TU Delft
	Dr. ir. Aki Härmä,	Philips Research

Delft
University of
Technology

Abstract

Change-point detection is an indispensable tool for a wide variety of applications which has been extensively studied in the literature over the years. However, the development of wireless devices and miniature sensors that allows continuous recording of data poses new challenges that cannot be adequately addressed by the vast majority of existing methods. In this work, we aim to balance statistical accuracy with computational efficiency, by developing a hierarchical two-level algorithm that can significantly reduce the computational burden in the expense of a negligible loss of detection accuracy. Our choice is motivated by the idea that if a simple test was used to quickly select some potential change-points in the first level, then the second level which consists of a computationally more expensive algorithm, would be applied only to a subset of data, leading to a significant run-time improvement. In addition, in order to alleviate the difficulties arising in high-dimensional data, we use a data selection technique which gives more importance to data that are more useful for detecting changes than to others. Using these ideas, we compute a detection measure which is given as the weighted sum of individual dissimilarity measures and we present techniques that can speed up some standard change-point detection methods. Experimental results on both artificial and real-world data demonstrate the effectiveness of developed approaches and provide a useful insight about the suitability of some of the state-of-the-art methods for detecting changes in many different scenarios.

Keywords - change-point detection, segmentation, time-series data, data selection techniques, speedup

Acknowledgments

The research reported in this thesis has been conducted in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering at the Delft University of Technology and was financially supported by Philips Research, Eindhoven. This work would not have been possible without the help and support of many people that I am grateful to.

First and foremost, I would like to express my heartfelt appreciation and gratitude to my supervisor Dr. Aki Härmä from Philips Research, who believed in me and gave me the great opportunity to carry out my Master thesis at Philips Research. His scientific advice, guidance and unconditional support in both professional and personal life was the key in achieving the thesis' objectives. Our valuable discussions and his constructive comments and suggestions from the very beginning to the end have had a great impact on my work and have significantly helped me in improving my thesis report.

I am also profoundly grateful to my academic supervisor, Associate Professor Richard Heusdens who helped me to shape my research interests and connected me to this work in the first place. His long-lasting guidance, support and encouragement throughout my Master studies at TU Delft were invaluable for the successful completeness not only of this project but also of much of the work required for the Master's program. Grateful acknowledgement is extended to the other members of the thesis committee, Professor Richard Hendriks and Professor Geert Leus for taking the time to evaluate my thesis.

I would also like to express my sincere gratitude to Silvia Bertagna de marchi, Natalia Díaz Rodríguez, Rim Helaoui and Dietwig Lowet for the interesting meetings and project ideas that we together developed at Philips Research. Special thanks to Silvia for spending time on reading my thesis and providing comments and criticism that helped me shape the report into its present form.

Last, but certainly not least, I heartily thank my parents, sister, grandmother and friends for their love, patience, sacrifice and constant support throughout my long educational journey. Very special thanks go to my friend Marina for her countless help and assistance during the past two years. I would be still far away from this achievement without those people.

You all made it happen! Thank you!

Vana Panagiotou
Delft, October 2015

To my grandmother

Contents

1 Introduction **1**

1.1 Problem Statement: Definitions and notation 1

1.2 Segmentation Categories 2

1.3 Applications 3

1.4 Properties of a segmentation algorithm 5

1.5 Thesis goal and outline 5

2 Literature review **7**

2.1 Segmentation methods overview 7

2.2 General Framework of window-based change-point detection 13

2.3 Parametric Change-point Detection methods 15

2.4 Non-parametric Change-point Detection methods 15

2.4.1 Direct Density Estimation Methods 18

3 Methodology **22**

3.1 Where does the problem arise in traditional methods? 22

3.2 A two-level change-point detection test: General Overview 23

3.3 First-Level algorithm 24

3.4 Second-Level algorithm 28

3.4.1 Data Selection Step 29

3.4.1.1 Hilbert-Schmidt independence Criterion (HSIC) 32

3.4.2 Jensen Shannon divergence (JSD) based on k-nearest neighbor (kNN) density estimation 32

3.5 Modified algorithm for online detection 34

3.6 Speeding-up existing change-point detection algorithms 34

4 Experiments **38**

4.1 Performance metrics 38

4.2 Experiments on Artificial Datasets 40
4.2.1 Data with fixed change-step 40
4.2.2 Data with random change-step. 41
4.2.3 Artificial data results 44
4.3 Experiments on Real-World Datasets 48

5 Conclusions and future work **59**

A Appendix A - Performance Tables **61**

Bibliography **64**

1

INTRODUCTION

1.1. PROBLEM STATEMENT: DEFINITIONS AND NOTATION

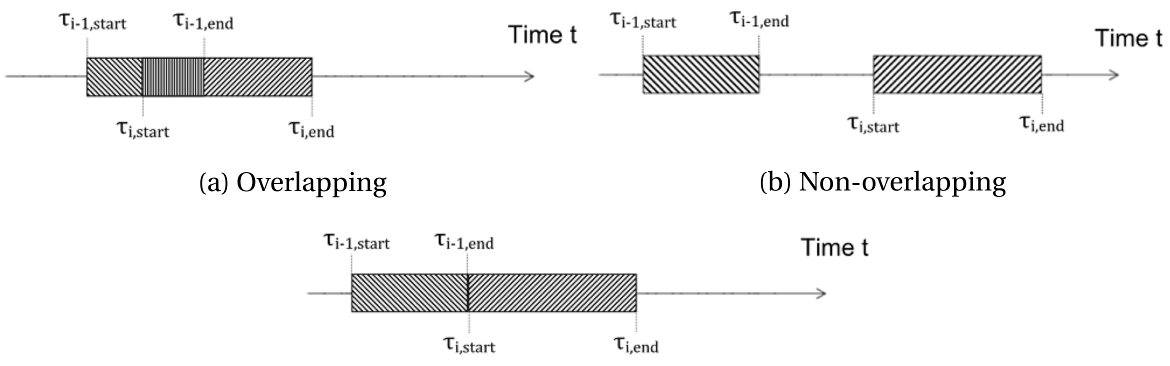
One of the fundamental characteristics of any information and database system is its tendency to evolve and change over time [1]. Identifying the transitions from one state of a system to another is a critical issue that constitutes the subject of change-point detection analysis and concerns many scientific fields. Over the years, many different terms, including segmentation, event detection, zonation analysis, breakpoint detection and concept drift, have been used to describe the same problem. In this thesis, we will use the most widely known terms *change-point detection* and *segmentation* interchangeably.

In the time-series context, change detection is the process of segmenting a data series into different regimes or *segments*, by identifying the points where the statistical properties change [2]. These points are known in the literature as *change-points* [3].

The idea behind the segmentation analysis is that data series do not behave as stationary along the whole timeline, but can be represented by approximately stationary intervals or segments. In this way, the signals can be partitioned into homogeneous segments with similar properties such that the subsequence within a segment is contiguous, while successive segments, which are separated by the change-points, are heterogeneous with respect to each other [4]. However, in practical situations, the location, size and number of segments are generally unknown and should be inferred only from the recorded time-series.

In the context of segmentation, time-series are defined as a set of n observations of d-dimensional vectors $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n]$, where $\mathbf{y}_i \in \mathbb{R}^d, i = 1, \dots, n$ corresponds to time-ordered observations, i.e., $\mathbf{Y}$ is a $d \times n$ matrix. Although in this definition, each observation at time t, $\mathbf{y}_t$, is assumed to be multivariate, a reduction to the univariate case can be easily obtained, by substituting $d = 1$.

The goal of segmentation is to find the intervals of homogeneity by identifying the number of change-points, m, as well as their locations, $\tau_{1:m} = (\tau_1, \tau_2, ..., \tau_m)$. Each change-point position is an integer between 1 and $n-1$ inclusive and typically the number of change points m is much less than n. Moreover, the change-points are assumed to be ordered so that $\tau_i < \tau_j$ if and only if $i < j$ [5].



(a) Overlapping (b) Non-overlapping

(c) Tight

Figure 1.1: Type of segments.

Generally, every segment i is limited by its starting and ending points, $\tau_{i,start}$ and $\tau_{i,end}$, respectively. A pair of segments $i-1$ and i can then be overlapping ($\tau_{i-1,end} > \tau_{i,start}$), non-overlapping ($\tau_{i-1,end} < \tau_{i,start}$), or tight ($\tau_{i-1,end} + 1 = \tau_{i,start}$) , as illustrated in Figure 1.1. An overlapping segmentation method on time-series data could be beneficial in defining overlapping clusters for classification data mining, where an object can belong to more than a single segment [6]. On the other hand, non-overlapping segmentation is useful for applications where we want clearly delineated segments with no intersection between them. Tight segmentation is a special category of the non-overlapping case, which aims to remove huge jumps and discontinuities between successive segments. This work will focus entirely on tight segmentation. In that way, the sequence of observations is partitioned into $m+1$ blocks or *segments*, as follows [5, 7]:

$$\left[\mathbf{y}_1^{1:d},\ldots,\mathbf{y}_{\tau_1}^{1:d}\right],\left[\mathbf{y}_{\tau_1+1}^{1:d},\ldots,\mathbf{y}_{\tau_2}^{1:d}\right],\ldots,\left[\mathbf{y}_{\tau_m+1}^{1:d},\ldots,\mathbf{y}_n^{1:d}\right] \tag{1.1}$$

1.2. SEGMENTATION CATEGORIES

Time-series segmentation methods have been studied in a number of different contexts over the years and, hence, there has been a wide variety of proposed techniques that can generally be classified into one of the following categories:

- *Aided or Blind*

  Segmentation methods differ mainly in how much prior information and external knowledge is used for the time-series processing. Aided algorithms use some sort of external knowledge or previously obtained data, whereas in blind algorithms there is no prior information regarding the signal [8].

- *Offline or Online*

  Segmentation algorithms are further distinguished on the basis if the entire batch of data is available from the beginning or it arrives in real-time at a specific rate [9, 10]. Other terms used to describe the offline setting are batch, a posteriori, retrospective or fixed sample change detection. Two of the major drawbacks of these algorithms are the large storage requirements and the inability to provide online service to solve any potential problems that might arise. On the other hand, the online algorithms intend to detect changes with minimum delay while using only finite computational resources. They are also known in the literature as sequential, incremental or streaming methods.

- *Univariate or Multivariate*

  Univariate change detection methods consider data which consist of only one variable at each time step, whereas multivariate approaches can handle time-series that span multiple dimensions within the same time range [11]. Although, most research has been conducted on univariate time-series data, multivariate segmentation enables the incorporation of multiple variables into the change detection process and consequently enhances the overall accuracy of the algorithm [12, 13].

- *Parametric or Non-Parametric*

  Parametric methods rely on assumptions about the data (e.g., they assume that the samples are drawn from a Gaussian distribution) and even about the parameters of the assumed distribution (e.g., mean and standard deviation). On the contrary, non-parametric tests make no such assumptions about the data [14, 15]. If the considered assumptions of parametric methods are correct, they can produce more accurate and precise results. Moreover, they have the advantage of being simple and fast to compute as opposed to the non-parametric ones. However, if the assumptions are incorrect, parametric methods can be very misleading and this is the reason why they are often not considered robust [16].

1.3. APPLICATIONS

Modern world is changing rapidly and various technologies have been developed over the past years to monitor activities in almost all aspects of personal and professional life ranging from health, economy and entertainment to industry. The ability to detect changes in the system's behavior is of the utmost importance, since the reported changes can be utilized to adapt the system to the new state, prevent potential emergency situations or mitigate the consequences that those changes entail.

Change-point detection analysis encompasses a wide variety of applications and holds a tremendous potential for the advance of technology in the future [17]. The origins of this field can be traced back to 1924, when Walter Shewhart [18] developed a simple control chart approach for process monitoring and quality improvement in the manufacturing industry. In this framework, it was assumed that the product quality follows a normal distribution and any variation from that distribution indicates a deterioration in the quality that should be appropriately detected and corrected.

Change-point detection is particularly beneficial in the biomedical and healthcare fields. As an example of a biomedical application, change-point detection methods are used to identify the regions of genome amplification and deletion in tumor cells by analyzing the

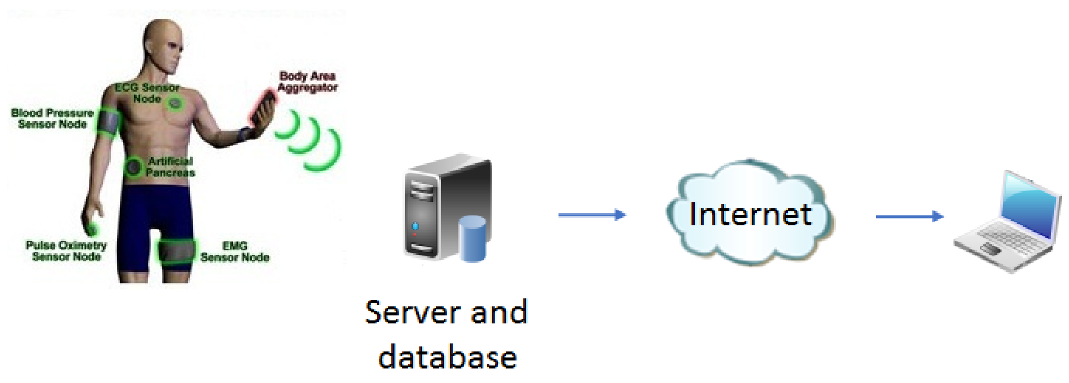


Figure 1.2: The general architecture of an online change-point detection system for health monitoring [33].

array-based comparative genomic hybridization (array-CGH) data [19, 20]. Another example of this category is the detection of epileptic seizures or the different sleep stages measured by the electroencephalogram (EEG) signal [21]. Aside from that, with the increasing prevalence of sensor-based applications in the daily use of smartphones and wearable devices, change detection in time-series data has become a crucial component of many Human Activity Recognition (HAR) tasks, such as health, wellness or fitness monitoring [22].

Figure 1.2 shows the general architecture of an online change detection system for health monitoring. The sensors forward the recorded data to a coordinator device, such as a wrist-worn smart watch, a tablet or a smart phone. All the collected data are then transmitted using long-range communications (e.g., through 3G/4G or WiFi) to remote servers that can be accessed by a variety of consumer devices (e.g., computers), which can finally process the data in real-time and send a notification if a significant change or abnormality has been detected. Sometimes, the coordinator device includes professional embedded software to process the recorded data directly without the intermediate transmission to a server [23]. Thus, it becomes clear that the goal of an online change detection system for health monitoring is to combine data from multiple sensors/sources in order to identify important events as quickly as possible and under resource-limited constraints. In such cases, the computational complexity becomes a critical issue as much as the algorithm's performance and robustness. In addition, the system may have to deal with high-dimensional data recorded from a large number of sensors.

Time-series segmentation methods are a cornerstone in many other disciplines, such as environmental science and especially the climatology [24], meteorology [25], finance [26], speech segmentation and voice activity detection (VAD) [27], satellite tracking [28], traffic control [29], robotics [30], safety-warning systems [31] and many others.

Note that segmentation can also be used as a pre-processing step for a variety of data mining tasks, including data representation, clustering, classification and visualization [4]. In classification, for example, by exploiting dynamically established windows obtained by segmentation, we can achieve better results and avoid the common errors associated with a fixed window approach [32].

1.4. PROPERTIES OF A SEGMENTATION ALGORITHM

Since time-series segmentation can be applied in fundamentally different scientific disciplines, it is obvious that there is a wide variety of ways of formulating and solving this problem. This means that a universal algorithm which results in an optimal solution for all cases does not exist, and therefore, the system requirements depend heavily on the type of application. However, it is desirable for any segmentation algorithm to satisfy a number of basic criteria [4, 34] :

- *Generality*: It is very important to create an algorithm capable of providing accurate segmentation results regardless of the nature of the data.
- *Accuracy*: Any approach should achieve a high rate of correctly identified segmentation boundaries with as few incorrectly determined (false positives) boundaries as possible.
- *Scalability*: It is always preferable to design an algorithm that is able to handle high-dimensional data with thousands of observations along time.
- *Complexity*: For efficiency reasons, the computational time and the processing latency must be sufficiently low. In many cases, system resources, such as the memory or the computing power, should also be taken into account.

Hence, there are many issues, which directly or indirectly determine the choice of the algorithm, and that researchers should examine beforehand in order to select the most appropriate approach for each situation.

1.5. THESIS GOAL AND OUTLINE

As technology continues to evolve and permeate every aspect of personal and professional life, it becomes more and more necessary to track changes in the state of a system in a fast and efficient manner and answer questions like:

- Did a change occur?
- Did more than one change occur?
- When did the change (changes) occur and how can we estimate its (their) location?

Although change-point detection has a long history of research, the explosive growth in both sample size and dimensionality of data poses new challenges that cannot be adequately addressed by the vast majority of existing methods. The primal goal of this thesis is to alleviate the difficulties associated with large datasets, by developing a flexible method that can perform equally well on a variety of datasets. In addition, since in many cases it may be difficult to decide which of the many proposed methods is more suitable for a particular problem, we intend to test many different approaches and come up with some guidelines that may help engineers and scientists to select a good method for the problem at hand.

The remainder of this thesis is organized as follows. Chapter 2 provides a literature review of the various segmentation and change-point detection techniques that have been developed over the years. In Chapter 3, we propose a new approach and some modifications to existing methods that can help to alleviate the difficulties that arise with large and/or real-world datasets. In Chapter 4, we report experimental results on artificial and real-world datasets and show how the different methods behave under several scenarios. Finally, the thesis is concluded in Chapter 5, in which we summarize our contributions and propose some possible extensions for future work.

2

Literature review

2.1. Segmentation methods overview

Several methods have been proposed so far in the literature to address the segmentation and change-point detection problems. In this section we will describe and compare the most well-known approaches for estimating change-points and segmenting the time-series data.

It should be noted that, in some contexts, segmentation has been used to simply compress the original time-series into a more compact representation and not to identify the change-points [35]. In this case, the data are partitioned into homogeneous segments and each segment is represented by a model or a single value (e.g., its mean value). The key components of a segmentation method formulated in that way are the model (linear, polynomial, regression, etc.) that will be fitted to the data to create a more compact representation and a stopping criterion that will be used to determine when to stop creating new segments. Many of those algorithms may also require information from the user, such as the desired or maximum number of segments and a maximum value for the reconstruction error, which determines the loss of accuracy in the data representation. Then, the problem is transformed into that of finding the segmentation which minimizes this reconstruction error [36].

Hence, it becomes clear that whenever the goal is to transform the data into another representation, there is a distinction between change-point detection and segmentation. Nevertheless, most of the times, the two terms have been used interchangeably in the literature, since, by definition, the transition points between successive segments are considered as the change-points. As our interest is in identifying the changes, except from the most important algorithms used exclusively for identifying change-points, we will also explain how the methods that were initially designed for data compression have been used in the change detection framework.

In the following paragraphs, we denote by n the length of the time-series and by m the desired number of segments. Depending on how the data are processed, change-point detection and segmentation algorithms can be roughly divided into the following categories or their hybrids:

1. *Dynamic programming based segmentation*

   An optimum segmentation for compression purposes can be found by a dynamic programming (DP) based approach [37]. Given a time series, an upper bound on the number of segments and the segment error of each possible segment, these algorithms search all possible segmentations and select the one whose overall error is minimal.

   In the change detection framework, the initial problem is divided into subproblems and an overall solution is obtained by combining the solutions of the subproblems. Existing approaches for identifying change-points via dynamic programming are mainly based on least-square fitting. However, the dynamic programming approaches face several challenges[38]. First, the number of change-points cannot be estimated accurately; when we have an upper bound m_{max} on the total number of change-points, the DP algorithm will always return m_{max} change-points. This means that the DP algorithm cannot find the true number of change-points and this constitutes a significant restriction for the applicability of those methods in real-world applications. Second, the computational complexity of DP based methods is very high. In particular, the running time of the first DP algorithm developed by Bellman [39] was of order $O(n^m)$, which was later improved to $O(n^3m)$, $O(n+n^2m)$, $O(n^2 log n + n^2m)$ or $O(n^2m)$ by some alternative techniques [40]. However, when the size of the data is very large, even the fastest DP algorithms become computationally very demanding and not well-suited to the segmentation task.

2. *Heuristics*

   The prohibitively high running time of dynamic programming based algorithms led the scientists to develop heuristic methods that do not necessarily construct an optimum segmentation, but produce good results in many cases and take less time than dynamic programming methods. These heuristic methods save time by considering only some possible segments to hopefully obtain a good segmentation and compress the data appropriately [37].

   According to [4, 41–43] most of the heuristic methods, which convert time-series into segmented versions, can be grouped into one of the following three categories:

   - *Sliding Window*: a window slides over the data and each segment is grown until an error criterion is met. The same process repeats with the next available data point which was not included in the previously approximated segment.

   - *Top-Down*: starting with only one segment, the time-series is recursively partitioned until an error criterion is met.

   - *Bottom-Up*: starting from the best possible short segments, segments are merged until an error criterion is met.

   A brief description for each of the three categories in the context of both data compression and change identification follows below.

(a) *Sliding Window*

The Sliding Window algorithm [4, 43] works by anchoring the first point of the time series t_k and step-by-step extending the segment to the right by including $t_{k+1}, t_{k+2}, \dots$ as long as the reconstruction error remains below a user-specified threshold. At some point t_i the threshold is exceeded, so the sequence from the anchor t_k to t_{i-1} is considered as a segment. The above process repeats with the anchor locating at the data point t_i until the entire time-series has been segmented. When sliding window methods are considered for detecting change-points, they look ahead in the time-series until they come up against a point that is considerably different from the points of the current window.

These algorithms have the advantage of being online and simple with little computational requirements, since only a restricted history of past data values needs to be maintained. However, they lack a global view of the data and when they are used to create a more compact representation of the data they generally produce very poor results and tend to over-segment time-series that contain noisy observations.

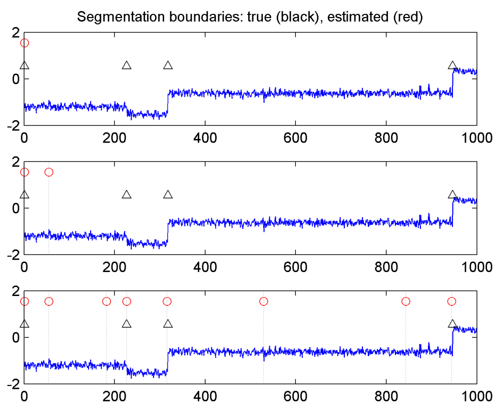
(b) *Top-Down*

The Top-Down method [4, 43] starts by considering the entire time-series as a segment and then splits it at the best location, in a way that the difference between these two segments is maximum. If the reconstruction error of the new segments is below a user-specified threshold, the segmentation process stops. Otherwise, the algorithm recursively continues splitting each of the newly formed segments until all created segments have reconstruction errors below the threshold. The procedure remains the same for change detection, although a different stopping criterion should be used here (e.g., a maximum allowed number of change-points).

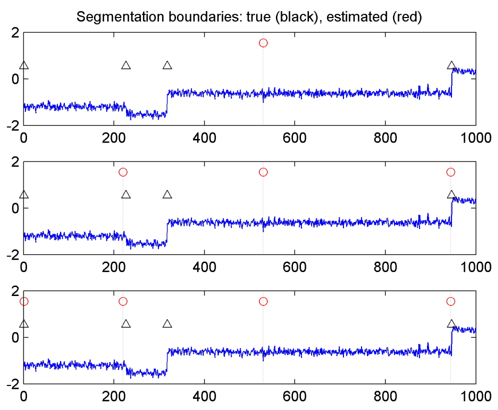
The main disadvantage of this approach is its inherent inflexibility which stems from the fact that the boundaries determined in the previous iterations remain unchanged until the end of the segmentation process. In actual fact, the boundaries identified as best location points at the initial steps of the algorithm will generally not be optimal in the later steps of the process, leading thus to a not optimally segmented time-series. Moreover, although the algorithm performs quite well, it can be used only in an offline setting and does not scale well with the size of the data.

(c) *Bottom-Up*

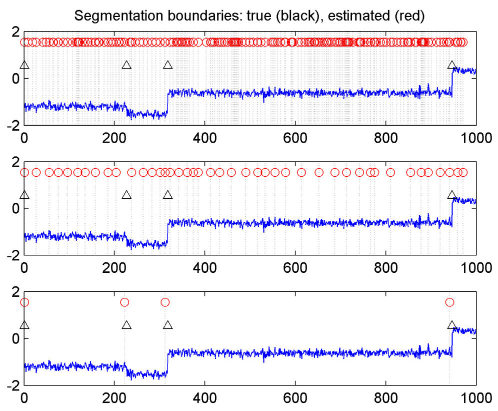
The Bottom-Up algorithm [4, 43], as the natural complement to the Top-Down algorithm, begins by splitting the entire time series of length n into a large number (typically $n/2$) of very small segments with equal lengths. In the following step, the cost of merging each pair of consecutive segments (including left and right neighbor) is calculated, and the algorithm starts iteratively merging the pairs that cause the smallest increase in the error. The above process repeats until the reconstruction error exceeds a predefined threshold. When this strategy is exploited in the change-point detection problem, it starts by finding the



(a) Sliding Window



(b) Top-Down



(c) Bottom-Up

Figure 2.1: Sliding Window, Top-Down and Bottom-Up segmentation examples. For each of the three different segmentation categories, the first, an intermediate and the last step of the segmentation procedure is depicted. In all plots, the true segmentation boundaries are represented by triangles and the estimated by vertical dotted lines and circles.

smallest possible homogeneous segments and then it uses some measure of similarity to compare the formed segments and combine them accordingly. A stopping criterion is also required here.

The Bottom-Up algorithm scales linearly with the size of the data and produces high quality results. However, it belongs to the category of offline methods and, thus, it is inefficient for processing of real-time data.

Figure 2.1 illustrates the differences among the three heuristic methods when the same data were used for all of them. As we can see, the sliding window algorithm over-segments the data due to the noisy samples, the top-down is highly affected from the first wrong identified boundary, while the bottom-up method seems to correctly identify the different segments.

3. *"Hybrid" Online Segmentation algorithm*

An algorithm called SWAB (Sliding Window And Bottom-up), was introduced in [44], in order to combine the advantages of online (Sliding Window) and offline (Bottom-Up) algorithms. It basically includes two steps; the Sliding Window step that performs a harsh pre-segmentation and the Bottom-Up step which improves the previously obtained result. In the context of data compression, the algorithm uses a buffer of size w to store a number of data points that initially should be sufficient enough to create 5 to 6 segments. Then, Bottom-Up is applied to the data in the buffer and the leftmost segment (oldest) is reported. The data corresponding to the reported segment are removed from the buffer and a number of new points is added.

The SWAB algorithm scales linearly with the size of the dataset and produces high quality segmentation results. However, defining the size of the initial buffer is of high importance for achieving an efficient performance. A large size of the initial buffer will convert the SWAB method into a Bottom Up method, while a small buffer size will convert it to a Sliding Window method.

There have also been developed probabilistic, clustering based and many other segmentation methods that have been used exclusively for identifying the change-points.

4. *Probabilistic based segmentation algorithms*

In [7], Barry and Hartigan introduced a model for multiple change-point analysis known as the Product Partition Model (PPM). This model, which assumes that the number and positions of the change points are random, is capable of providing the product estimates for the parameters of interest at each individual time, as well as the posterior distributions of the partitions and the number of change points. Prior distributions are assumed for the means, variances and for the probability that each individual time is a change-point [45].

Recently, Ferreira et al. [46] proposed a product partition model, which, compared with the original model of Barry and Hartigan [7] , includes across-cluster dependence between the segments (or clusters). This dependence is introduced into the model through the prior distributions of the parameters. The experimental comparison of those two methods revealed that although they demonstrated similar performance regarding the estimation of the number and positions of the change-points, the method with the incorporated dependence presented a substantial improvement in the parameter estimates.

There have also been proposed bayesian based segmentation techniques [47, 48] which assume that the data originate from a probabilistic distribution and use a criterion to find the number and location of segments. Among the most commonly used criteria are the AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), MDL (Minimum Description Length) and MML (Minimum Message Length) with the latter providing the best performance compared to the others [47]. Approximations of the above methods which are suitable for processing of large amounts of data are described in [48].

Hidden Markov Models (HMM) are often used to detect segmentation boundaries [49]. In this case, each segment corresponds to a state in the HMM, and, hence, a change-point occurs when switching from one state to another. However, these methods have several undesirable properties and require a large computational time.

5. *Clustering based segmentation algorithms*

   The problem of segmenting time series can be formulated as a clustering problem with the constraint that all time points of a cluster must be successive in time. Based on this idea, a fuzzy clustering approach [50] and a generalized Eigen-decomposition [51] have been exploited to segment non-stationary or chaotic time-series.

6. *Window-based segmentation algorithms*

   The window-based algorithms compare small pieces (or windows) of data with their neighbors by using a dissimilarity measure, and if the difference between the two windows is above a threshold then a change-point occurs [3, 14, 52–57]. The major advantage here is that this technique is applicable in both the online and offline frameworks. However, sometimes it may be difficult to select a good threshold and thus a sufficient performance may not be achieved. This category of algorithms will be discussed in more detail in Section 2.2.

7. *Other segmentation algorithms*

   In [58], Magnusson proposes a bottom-up, level-by-level search for repeated temporal patterns, named T-patterns, in symbolic time-series, where each symbol represents the onset of a specific event. The algorithm is based on the detection of possible relationships between pairs of events and the creation of binary trees of such temporal dependencies in a hierarchical way.

   In [52], Chung et al. propose an evolutionary time series segmentation algorithm which allows the user to determine a set of pattern templates and search for them during the segmentation process. In addition, Perceptually Important Points (PIP) can be used to adjust the lengths of the segments in order to achieve a better match with the predefined patterns. However, this approach is not suitable for applications in which users do not know what patterns they want. As a special case of this technique, [59] focuses on stock time-series where a set of stock patterns is generated and exploited in the procedure.

   Yin et al. [60] proposed a segmentation method for financial time-series based on Turning Points (TPs), which correspond to local maximum and minimum points and represent the change in the trend of the stock. In more detail, the first stage of the algorithm divides the time-series into different periods such that each segment has a single trend during that period. At the second step of the algorithm small trends are merged into more significant ones to better form the segmentation boundaries.

   A threshold-free online time-series segmentation has been proposed in [61]. It is based on the concurrent estimation of two models (a model with one regressive segment and a temporal mixture model with two regressive components) and uses the Bayesian Information Criterion (BIC) to select the best model between them. The potential change-points of the time-series are derived from the proportions of the temporal mixture model.

In [62], change-points are detected in time-series by minimizing a cost function that depends on covariance matrix using a low-complexity Pruned Exact Linear Time (PELT) method. Since graphical models are known to be powerful tools for describing complex systems, this method generates graphical models (Copula Gaussian graphical models, with and without hidden variables) for each stationary segment based on their covariances in order to determine the number and location of change-points.

2.2. GENERAL FRAMEWORK OF WINDOW-BASED CHANGE-POINT DETECTION

The window-based technique is a powerful way of dealing with large datasets and online applications and, as a consequence, various approaches to change-point detection have been investigated within this framework. In the rest of this chapter we will describe the most important aspects of this concept and we will introduce some of the most well-known methods following this strategy.

It may be noted here that the window-based framework has always been referred to as "sliding window". However, it should not be confused with the sliding window segmentation approach that was described in the previous section. Although both techniques share the idea of sliding a window over the data, the way that they process the data is fundamentally different. In the rest of this document, unless otherwise noted, the term "sliding window" is used to refer to the window-based framework.

The main idea underlying the sliding window based change-point detection method is to compare data samples extracted from two windows: the *reference* or *past* window and the *testing* or *present* window. Depending on the sliding step, there are two commonly used approaches to segment data series using the sliding window method. The first approach employs non-overlapping sliding windows, where consecutive windows do not share common data samples. The second approach employs overlapping sliding windows which share some common data samples. In addition, we consider two types of windows depending on whether the number of samples in the window is fixed (fixed-size window) or variable (variable-size window). The choice of window widths and sliding step depends on the application requirements.

We will now present how the detection is performed using the general framework depicted in Figure 2.2. We consider time series data of the form $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$, where $\mathbf{y}_i \in \mathbb{R}^d, i = 1, \ldots, n$. At time position t, two non-overlapping windows w_1 and w_2 are generated containing the subsets $\mathbf{y}_t = \{y_i^{1:d}\}_{i=t-l_1+1,\ldots,t}$ and $\mathbf{y}'_t = \{y_i^{1:d}\}_{i=t+1,\ldots,t+l_2}$, respectively, where l_1 is the size of window w_1 and l_2 is the size of window w_2. For the sake of simplicity, we consider that the two data sequences contain the same number of samples v and we denote them as $\mathcal{Y} := \{y_i^{1:d}\}_{i=t-v+1,\ldots,t}$ and $\mathcal{Y}' := \{y_j'^{1:d}\}_{j=t+1,\ldots,t+v}$, respectively.

The change-point detection problem can now be transformed into a hypothesis testing problem, where the null hypothesis H_0 is tested against the alternative hypothesis H_1 as below [14]:

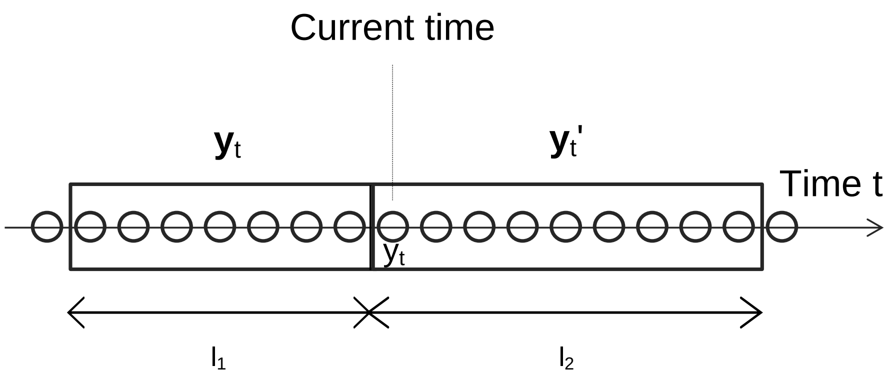


Figure 2.2: General change-point detection framework based on the sliding-window approach. The data samples $y_t^{1:d}, t = 1, 2, \ldots$ are represented by circles.

$$\begin{cases} H_0 : D(\mathcal{Y}, \mathcal{Y}') \le \eta, & \text{No change occurs} \\ H_1 : D(\mathcal{Y}, \mathcal{Y}') > \eta, & \text{A change occurs} \end{cases} \tag{2.1}$$

where $D(\mathcal{Y}, \mathcal{Y}')$ is a distance function or metric which measures the dissimilarity of the two windows and η is a threshold used to decide whether a change occurs or not. More specifically, the higher the dissimilarity measure is, the more likely the point t is a change-point. We consider that a change occurs when the dissimilarity measure between the windows exceeds the specified threshold, which is actually a parameter that can control the sensibility/robustness tradeoff. The windows are then slid throughout the whole signal to get the distance function.

Hence, it becomes clear that the detection performance depends strongly on the dissimilarity measure selected and the key issues that now need to be addressed are what kind of dissimilarity measure should be used and how it can be estimated from the data samples.

There have been proposed many methods that consider the dissimilarity measure as a *distance* between two underlying probability distributions P and P' with probability density functions (PDFs) $p(\mathbf{y})$ and $p'(\mathbf{y})$ computed from the sets of samples $\mathcal{Y}$ and $\mathcal{Y}'$, respectively. This distance can be expressed as the likelihood density ratio $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$, or as the density difference $p(\mathbf{y}) - p'(\mathbf{y})$, or as a divergence measure. However, since only the two sets of samples are available, several density estimation methods have been proposed to effectively estimate the densities and compute the above distances [3, 17, 52, 54–57, 63]. Other methods assume that the data originate from some unknown model, e.g., an autoregressive model [64], and identify the changes by comparing the constructed models. Alternatively, the data samples could be compared directly without any model assumptions [14, 65]. All the above-mentioned different ways of comparing the two sets of samples can be further divided into parametric and non-parametric (see Section 1.2). Each of these categories will be explored in detail in the following sections. Note also that the change-point detection methods can be applied either to raw or filtered data, or even in sequences of features derived from the data. In this work, the detection is performed by using only raw data.

2.3. PARAMETRIC CHANGE-POINT DETECTION METHODS

Parametric methods generally model the data with a pre-fixing model and incorporate some sort of knowledge into the detection scheme. A common assumption made in these approaches is that the underlying distribution functions, corresponding to the two windows, belong to some known family, such as the Gaussian. The unknown parameter sets of the PDFS, denoted by θ and θ', are estimated by the data samples and the resulting PDFs, $p(\mathbf{y};\widehat{\theta})$ and $p'(\mathbf{y};\widehat{\theta}')$, are compared via a likelihood ratio $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$ [17, 63] or by using a divergence measure, such as the Kullback Leibler (KL) divergence [66]. Some pioneering works based on this concept include the Generalized Likelihood-Ratio (GLR) [63] and cumulative sum methods (CUSUM) [17]. However, these approaches are particularly problematic in high-dimensional problems, since usually in these cases a large number of noisy features is present and it has been known that density estimation tends to be degraded by noise [65].

There are also parametric change-point detection approaches which rely on the Bayes decision theory (Bayesian approaches) [14]. In these methods, a joint prior distribution is placed over the number and location of change-points and an observation model describes the distribution of the data, given the change-points. However, selecting suitable priors for both the unknown parameters θ and θ' of the PDFs and the frequency of change-points is an important issue [67]. A widely known method of this category is the parametric Bayesian change-point test of Barry and Hartigan [7].

Some other parametric methods that have attracted increasing attention in recent years are the subspace methods, which have been thoroughly studied in the area of control theory [68–70]. By using a pre-designed time-series model, a subspace is detected by Principal Component Analysis (PCA) from trajectories in past and present data samples, and their dissimilarity is measured by the distance between the subspaces. The most promising approach in this category is the subspace identification, which compares the subspaces spanned by the columns of an extended observability matrix generated by a state-space model with system noise [69].

Generally, the methods described above are limited by relying on pre-specified models, such as probability density functions or state-space models, for tracking specific statistics including the mean and the variance. This actually means that the parametric methods tend to be less flexible in real-world scenarios and more efficient methods should be developed.

2.4. NON-PARAMETRIC CHANGE-POINT DETECTION METHODS

To overcome the disadvantages of the parametric methods, many non-parametric approaches have been described in the literature. These methods can be divided into two main categories based on how they compute the dissimilarity measure between the two sets of samples. The first category considers the dissimilarity as a distance between two underlying PDFs, which, as was mentioned above, can be computed as a density ratio, or a density difference or a divergence measure. The second category compares the data sequences directly (for example by means of a rank statistics test). All these strategies are further described in the following paragraphs.

Density estimation methods

One way to deal with the density estimation problem, is to first estimate the two densities separately and then compare the estimated densities, $\widehat{p}(\mathbf{y};\theta)$ and $\widehat{p}(\mathbf{y};\theta')$, using a dissimilarity measure. In general, density estimation can be approached in two ways: by Kernel Density Estimation (KDE) or by k-Nearest Neighbor (kNN) density estimation [71].

In Kernel Density Estimation, an estimate of the density is made by centering normalized kernels on each sample and computing weighted averages. The kernels typically have a bandwidth parameter, which is estimated by means of cross validation [71]. For well-behaved low dimensional distributions, KDE often performs well. However, as the dimensionality and non-uniformity of the problem increase, more and more weights in the KDE become small and, thus, the estimation accuracy is negatively affected [71, 72]. Additionally, the choice of an appropriate bandwidth is a critical issue for achieving a high performance, and since in this approach it is estimated by means of cross-validation, the computational expense is further increased.

With k-Nearest Neighbor density estimation, the density is estimated by computing the volume required to include the k nearest neighbors of the current sample [71–73]:

$$p(\overrightarrow{\theta}) = \frac{1}{N}\frac{k}{\rho_k(\overrightarrow{\theta})^d v_d} \tag{2.2}$$

where $\rho_k(\overrightarrow{\theta})$ represents the distance to the k-th nearest neighbors, d the number of dimensions and N the number of included samples. Furthermore, v_d denotes the volume of the unit ball in $\in \mathbb{R}^d$ and is given by:

$$v_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \tag{2.3}$$

where Γ corresponds to the Gamma function. The major benefit of using kNN estimate is that this estimator adapts to the local sampling density, adjusting its volume where sampling is sparse [71]. However, this kind of estimation has not received much attention in the literature, because it has been known to be biased and less accurate in many cases and especially in high-dimensional data [73]. In addition, similar to the KDE approach, this estimator also suffers from a loss of accuracy when estimating high-dimensional densities [71].

There have also been proposed non-parametric approaches that use a density-ratio based dissimilarity measure calculated directly without going through separate density estimation of $p(\mathbf{y})$ and $p'(\mathbf{y})$ [52, 65, 74]. The rationale of this idea is that knowing the two densities implies knowing the density ratio, but not vice versa; knowing the ratio does not necessarily imply knowing the two densities since such decomposition is not unique, as illustrated in Figure 2.3. This is often referred to as Vapnik's principle [75]: "*If you possess a restricted amount of information for solving some problem, try to solve the problem di-*

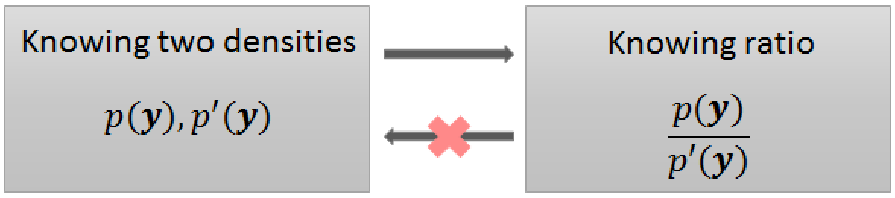

Knowing two densities
$p(\mathbf{y}), p'(\mathbf{y})$
Knowing ratio
$\frac{p(\mathbf{y})}{p'(\mathbf{y})}$

Figure 2.3: Rationale of direct density-ratio estimation.

rectly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem."

Well known recent direct density-ratio estimation techniques are the Kernel Mean Matching (KMM) [76], the Kullback-Leibler Importance Estimation Procedure (KLIEP) [57] based on the Kullback-Leibler divergence, the WKV method [77], the unconstrained Least Squares Importance Fitting (uLSIF) [78] and its robust extension named relative uLSIF (RuLSIF) [54] that both rely on the Pearson-divergence. In statistical machine learning, avoiding density estimation is essential because it is often more difficult than direct density-ratio estimation. However, as with density estimation based methods, the performance of direct density-ratio estimation methods is likely to be degraded by noisy data [65].

An alternative way of comparing the PDFs corresponding to the two data sequences is by computing the difference $p(\mathbf{y}) - p'(\mathbf{y})$ instead of the ratio $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$. In general, density differences would be more desirable than density ratios because density ratios can sometimes diverge to infinity even under mild conditions (e.g., Gaussian assumption), whereas density differences are always finite as long as each density is bounded [55]. One such method is the recently developed Least-Squares Density Difference (LSDD) method [55], which directly estimates the difference without separately estimating the two densities.

Rank statistics methods

Nonparametric approaches have also been performed based on rank statistics, to avoid the difficulties associated with density estimation [14, 15]. These methods compare the sample sets $\mathcal{Y}$ and $\mathcal{Y}'$ directly without any intermediate density estimation step. However, they tend to be less accurate in high-dimensional problems because of the curse of dimensionality [52, 79]. Kernel Change Detection (KCD) [14], which belongs to this category, builds a dissimilarity measure based on One-class Support Vector Machine (SVM) coefficients. Although KCD is robust to outliers, its performance is deteriorated by noisy features. In [65], a kernel-based independence measure called additive Hilbert-Schmidt independence Criterion (aHSIC), which is defined as the weighted sum of the HSIC scores, was used as a detection measure. A big advantage of this method is that it basically incorporates feature selection such that only important features are being used to compute the aHSIC score.

The direct density estimation methods, KLIEP, uLSIF, RuLSIF and LSDD, that have attracted much attention in recent years due to their high performance rate in the non-parametric setup are briefly described in the following subsection.

2.4.1. DIRECT DENSITY ESTIMATION METHODS

The goal of direct density estimation methods is to compare the probability distributions P and P' corresponding to the sample sets $\mathcal{Y} = \{y_i^{1:d}\}_{i=t-v+1,...,t}$ and $\mathcal{Y}' = \{y_j'^{1:d}\}_{j=t+1,...,t+v}$ in a direct way without seperately estimating the densities $p(\mathbf{y})$ and $p'(\mathbf{y})$.

The direct density-ratio estimation methods use the divergence $D(P,P')$ as the plausibility of change-points, where $D(P,P')$ denotes the f-divergence that was introduced independently by Csiszàr [80] and Ali and Silvey [81], and is defined by the following formula [52, 80, 81]:

$$D(P,P') = \int p'(\mathbf{y}) f\left(\frac{p(\mathbf{y})}{p'(\mathbf{y})}\right) d\mathbf{y} \tag{2.4}$$

where f is a convex function such that $f(1) = 0$, and $p(\mathbf{y})$ and $p'(\mathbf{y})$ are assumed to be strictly positive. Moreover, since the f-divergence is asymmetric (i.e., $D(P,P') \neq D(P',P)$), the above divergence is symmetrized into a new dissimilarity measure $D(P,P') + D(P',P)$.

The f-divergence incorporates various popular divergences, such as the Kullback-Leibler (KL) divergence by $f(t) = t \cdot log t$ and the Pearson (PE) or χ^2-divergence by $f(t) = (t-1)^2$ [82]:

- KL divergence: $KL(P,P') = \int p(\mathbf{y}) \log\left(\frac{p(\mathbf{y})}{p'(\mathbf{y})}\right) d\mathbf{y}$
- Pearson divergence: $PE(P,P') = \int p'(\mathbf{y}) \left(\frac{p(\mathbf{y})}{p'(\mathbf{y})} - 1\right)^2 d\mathbf{y}$

The key restriction of direct density-ratio estimation methods is to avoid estimating the densities $p(\mathbf{y})$ and $p'(\mathbf{y})$ when estimating the ratio or *importance* $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$. In order to meet this requirement, the ratio has to be modeled by the following kernel model [3, 52, 54–56]:

$$g(\mathbf{y};\boldsymbol{\theta}) = \frac{p(\mathbf{y})}{p'(\mathbf{y})} := \sum_{\ell=1}^{v} \theta_\ell K(\mathbf{y}, \mathbf{y}_\ell) \tag{2.5}$$

where $\boldsymbol{\theta} := (\theta_1, ..., \theta_v)^T$ are parameters to be learned from the data samples and $K(\mathbf{y}, \mathbf{y}')$ is a kernel basis function. In practice, the Gaussian kernel is used:

$$K(\mathbf{y}, \mathbf{y}') = \exp\left(-\frac{\|\mathbf{y} - \mathbf{y}'\|}{2\sigma^2}\right) \tag{2.6}$$

where σ(>0) is the kernel width which is usually determined based on cross-validation. The remaining question of this procedure is how to accurately estimate the kernel model using only the data samples.

In the direct density-difference approach the L^2-distance approximation between $p(\mathbf{y})$ and $p'(\mathbf{y})$, defined by [55]:

$$L^2(P,P') := \int \left(p(\mathbf{y}) - p'(\mathbf{y})\right)^2 d\mathbf{y} \tag{2.7}$$

is used as the plausibility of change-points. This method basically follows the same idea as stated above for the direct density-ratio approach, with the only difference that the kernel model is used to model the difference $p(\mathbf{y}) - p'(\mathbf{y})$ instead of the ratio $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$, i.e., in this case the kernel model is defined as:

$$g(\mathbf{y};\boldsymbol{\theta}) = p(\mathbf{y}) - p'(\mathbf{y}) := \sum_{\ell=1}^{v} \theta_\ell K(\mathbf{y},\mathbf{y}_\ell) \tag{2.8}$$

The exact methodology used in each of the four different direct density estimation approaches is briefly explained in the following paragraphs. We refer the interested reader to the original papers for further details about each method.

- **KLIEP** (Kullback-Leibler Importance Estimation Procedure)

  KLIEP [3, 56] is a direct density-ratio estimation algorithm that is suitable for estimating the KL divergence. The parameters $\boldsymbol{\theta}$ in the kernel model $g(\mathbf{y};\boldsymbol{\theta})$ are determined so that the KL divergence from $p(\mathbf{y})$ to $g(\mathbf{y};\boldsymbol{\theta})p'(\mathbf{y})$ is minimized. This means that the initial problem can be transformed into a convex optimization problem with a unique global optimal solution $\widehat{\boldsymbol{\theta}}$ that could be obtained, for example, by a gradient-projection iteration. The solution to the optimization problem gives a density-ratio estimator:

$$\widehat{g}(\mathbf{y};\boldsymbol{\theta}) := \sum_{\ell=1}^{v} \widehat{\theta}_\ell K(\mathbf{y},\mathbf{y}_\ell) \tag{2.9}$$

  and, finally, an approximator of the KL divergence can be computed by:

$$\widehat{KL} := \frac{1}{v}\sum_{i=1}^{v} \log \widehat{g}(\mathbf{y}_i) \tag{2.10}$$

- **uLSIF** (unconstrained Least Squares Importance Fitting)

  Recently, another direct density-ratio estimator called uLSIF has been proposed for estimating the Pearson divergence [78]. This method fits the density-ratio model to the true density-ratio under the squared loss, or in other words, the parameters $\boldsymbol{\theta}$ of the kernel model are determined so that a squared loss function is minimized. The solution $\widehat{\boldsymbol{\theta}}$ of the corresponding convex optimization problem can be used to estimate the PE divergence:

$$\widehat{PE} := -\frac{1}{2v}\sum_{j=1}^{v} \widehat{g}(\mathbf{y}'_j)^2 + \frac{1}{v}\sum_{i=1}^{v} \widehat{g}(\mathbf{y}_i)^2 - \frac{1}{2} \tag{2.11}$$

  uLSIF has some distinct advantages compared to the KLIEP method. First, its solution can be computed analytically and second, it achieves the optimal non-parametric convergence rate. In addition, it has the optimal numerical stability and it is more robust than KLIEP [52, 78].

- **RuLSIF** (Relative unconstrained Least Squares Importance Fitting)

  A major disadvantage of uLSIF is that depending on the condition of the denominator density $p'(\mathbf{y})$, the density-ratio value $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$ can be unbounded, (i.e., they can be infinity) [52]. To overcome this problem, the relative density-ratio estimation was defined as:

$$r_\alpha(\mathbf{y}) = \frac{p(\mathbf{y})}{p'_\alpha(\mathbf{y})} = \frac{p(\mathbf{y})}{\alpha p(\mathbf{y}) + (1-\alpha)p'(\mathbf{y})} \tag{2.12}$$

  where $p'_\alpha(\mathbf{y}) = \alpha p(\mathbf{y}) + (1-\alpha)p'(\mathbf{y})$ is the α-mixture density and $\alpha \in [0,1)$.

  Using this definition, the relative Pearson divergence was introduced by Yamada et al. as [54]:

$$PE_\alpha(P, P') := PE_\alpha\left(P, \alpha P + (1-\alpha)P'\right) = \int p'_\alpha(\mathbf{y})\left(\frac{p(\mathbf{y})}{p'_\alpha(\mathbf{y})} - 1\right)^2 d\mathbf{y} \tag{2.13}$$

  The novelty of the relative density-ratio is that it is always bounded above by $\frac{1}{\alpha}$ for $\alpha > 0$, even when the plain density-ratio $\frac{p(\mathbf{y})}{p'(\mathbf{y})}$ is unbounded. It has been shown that when the relative density-ratio is exploited, the estimation accuracy is improved and the convergence rate is faster compared to when the plain density-ratio is used. It should also be noted that when $\alpha = 0$, the relative density-ratio is reduced to the plain density-ratio, and, thus, the RuLSIF method is transformed to the previously explained uLSIF method [52].

  In the same way as the uLSIF method, the parameters $\boldsymbol{\theta}$ of the kernel model for approximating the relative density-ratio are specified by minimizing a squared loss function between the true and the estimated relative ratios. By using the solution $\widehat{\boldsymbol{\theta}}$ of the corresponding convex optimization problem, the relative PE-divergence can be approximated as [54]:

$$\widehat{PE}_\alpha := -\frac{\alpha}{2v}\sum_{i=1}^{v}\widehat{g}(\mathbf{y}_i)^2 - \frac{1-\alpha}{2v}\sum_{j=1}^{v}\widehat{g}(\mathbf{y}'_j)^2 + \frac{1}{v}\sum_{i=1}^{v}\widehat{g}(\mathbf{y}_i) - \frac{1}{2} \tag{2.14}$$

  The superiority of RuLSIF over uLSIF lies in its better non-parametric convergence property and its improved estimation accuracy, since its advantages regarding the analytic solution, numerical stability and robustness are also present in the uLSIF method [52, 54].

- **LSDD** (Least-Squares Density Difference)

  For the density-difference estimation problem, Sugiyama et al. [55] proposed a method called Least-Squares Density Difference (LSDD) that directly estimates the density difference without separately estimating the two densities. As with the direct density-ratio estimation methods, the density difference has to be modeled by a (Gaussian) kernel model $g(\mathbf{y};\boldsymbol{\theta})$. Next, the algorithm fits the model to the true density difference function under the squared loss. The solution $\widehat{\boldsymbol{\theta}}$ of the corresponding convex optimization problem that minimizes the squared loss function gives a density difference estimator $\widehat{g}(\mathbf{y};\boldsymbol{\theta})$. This estimator can then be utilized in the L^2-distance approximation between $p(\mathbf{y})$ and $p'(\mathbf{y})$, which is then used to decide whether a change occurs or not. This method has the advantage of providing a solution that can be computed analytically in a computationally efficient and stable manner. Further, it achieves the optimal convergence rate in a non-parametric setup and is more robust against outliers than the Kullback-Leibler divergence [55].

3

Methodology

3.1. Where does the problem arise in traditional methods?

With the recent advancement of data collection technologies and data storage hardware, massive amounts of large sample size and high-dimensional data (Big Data) are present in almost all modern applications, and according to the estimates such growth in datasets sizes is expected to accelerate in the future. This brings both opportunities and new challenges to research scientists. One the one hand, Big Data hold great promises for discovering hidden structures and heterogeneities that are not possible to be determined with small-scale of data. On the other hand, many computational and statistical challenges are introduced, including noise accumulation, latent correlations, data redundancy, measurement errors, corrupted and/or missing data in case a sensor reports malfunction as well as heavy computational cost [83].

The traditional change-point detection algorithms are not designed to cope with this kind of explosive growth in data. In general, these algorithms perform multivariate tests to determine the change-points and since they make direct use of the correlations between variables in an overarching way, they give a more comprehensive view of the data, providing thus an extremely powerful tool for detection tests. However, their performance tends to be degraded by noisy and corrupted data and they require particularly high demands of computational resources due to their complex nature.

Recently, Yamada et al. [65], proposed a change-point detection measure called the additive Hilbert-Schmidt Independence Criterion (aHSIC), which is given as the weighted sum of the HSIC scores computed separately for each dimension of the data. An advantage of the aHSIC measure over existing detection methods is that it can incorporate feature selection during detection, and, thus, only data that are important for an abrupt change are being used. The conducted experiments showed that aHSIC is suited for high-dimensional time-series data and it is more robust as regards noise than existing multivariate measures [65]. However, this approach considers only small sample size and high-dimensional scenarios, and it does not scale well to large datasets.

To overcome the drawbacks mentioned above and effectively control the computation cost, we propose a new algorithm with hierarchical structure that will be fully described in the following sections.

3.2. A TWO-LEVEL CHANGE-POINT DETECTION TEST: GENERAL OVERVIEW

Since real-world applications do not exhibit well-defined behavior, we will approach the problem using a blind (non-parametric) segmentation algorithm that does not consider any prior knowledge about the nature of the data or the changes. To design an effective statistical procedure that can be applied in both small- and large-scale datasets, we need to address the problems associated with massive amounts of data in addition to balancing the statistical accuracy and computational efficiency. For this purpose, we have developed a Two-Level hierarchical blind segmentation method, as shown in Figure 3.1. Our choice was motivated by the idea that if a simple test was used to quickly select some potential candidates in the First-Level, then the Second-Level would be applied only to a small portion of the data. For example, if we assume that the computational cost of computing a divergence measure in one-dimensional data of length n is $n \cdot O(l)$, then by quickly selecting $N << n$ candidates over the data, the cost is reduced to approximately $N \cdot O(l)$. Consequently, compared to the existing methods that require examination of all samples in the dataset, our algorithm would be significantly faster.

Using this idea, our First-Level algorithm relies on a simple and easy-to-compute approach, because, as it was mentioned before, at this stage we are more interested in the computational speed rather than the segmentation precision. More specifically, this preselection step is based on the derivative of the filtered signal combined with a significant extrema determination step and aims to detect the most likely change-points. It should be noted that each dimension of the data is examined separately and thus we obtain a different list of potential candidates for each dimension.

The Second-Level aims at validating or rejecting detections raised by the First-Level and is based on the work presented in [65]. To deal with the problems associated with high dimensional data, we first select data that are important in relation to a change in a supervised manner and then compute a dissimilarity measure by using only those selected data. Our final detection measure is given as the weighted sum of the JSD scores, where JSD is the Jensen Shannon divergence (JSD) based on k-nearest neighbor (kNN) density estimation. Each JSD score is computed separately for each dimension using its corresponding candidates from the First-Level. The proposed detection measure is called Hierarchical additive Jensen Shannon divergence (HαJSD).

We will now introduce the notation used throughout the proposed algorithm and formally describe our problem setting. Let $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_n] \in \mathbb{R}^{d \times n}$ denotes the input data. The First-Level processes each vector $\mathbf{y}_k$ separately and returns a list of potential change-points per data dimension. In the Second-Level, we suppose that two non-overlapping d-dimensional sequences $\mathcal{Y} = \{y_i^{1:d}\}_{i=t-v+1,...,t}$ and $\mathcal{Y}' = \{y_j'^{1:d}\}_{j=t+1,...,t+v}$ with v samples each are extracted from the data in a sliding-window manner at time t. Note that we extract these

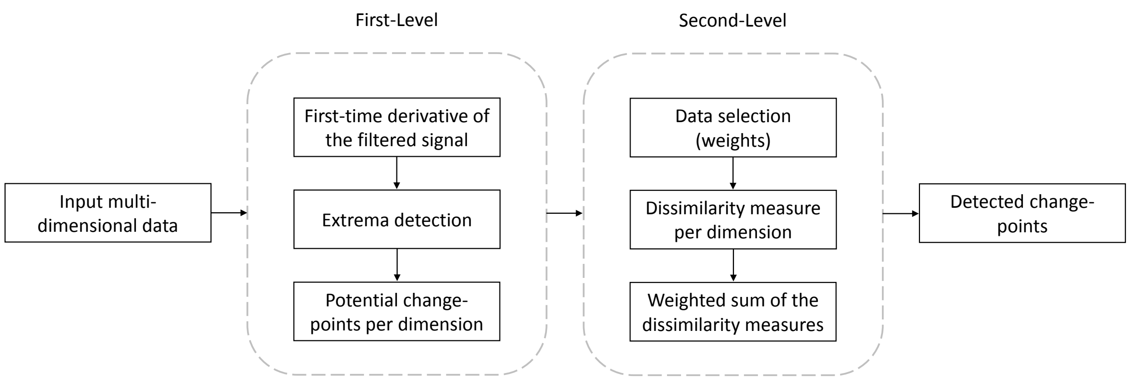

First-Level
Second-Level
Input multi-dimensional data
First-time derivative of the filtered signal
Extrema detection
Potential change-points per dimension
Data selection (weights)
Dissimilarity measure per dimension
Weighted sum of the dissimilarity measures
Detected change-points

Figure 3.1: The architecture of the proposed Two-Level change-point detection algorithm.

sequences v samples before and v samples after each time position corresponding to a potential change-point. Using this framework as a basis, we compute the weighted dissimilarity measure as:

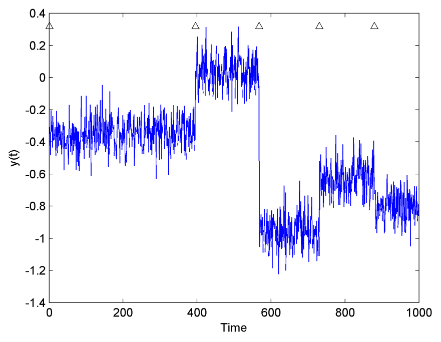
$$D\left(\mathcal{Y},\mathcal{Y}'\right) = \sum_{k=1}^{d} \alpha_k \cdot \mathrm{JSD}\left(\mathbf{y}_k^T(t), \mathbf{y}_k'^T(t)\right) \tag{3.1}$$

where $\mathbf{y}_k^T(t) = [y_k(t-v+1),\ldots,y_k(t)]^T$ and $\mathbf{y}_k'^T(t) = [y_k(t+1)\ldots,y_k(t+v)]^T$ are the k-th data samples of the two intervals,respectively, T is the matrix transpose, JSD is a dissimilarity measure called Jensen Shannon divergence and $a_k \in [0,1]$ is a coefficient representing the relevance of the corresponding data to a change.

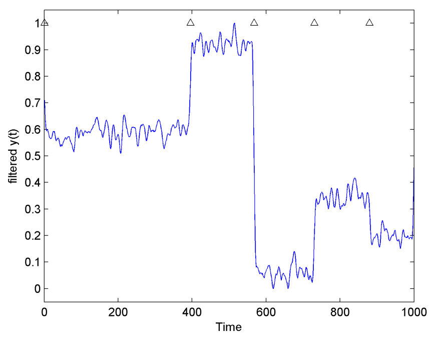
The proposed algorithm is described in detail in the following sections.

3.3. FIRST-LEVEL ALGORITHM

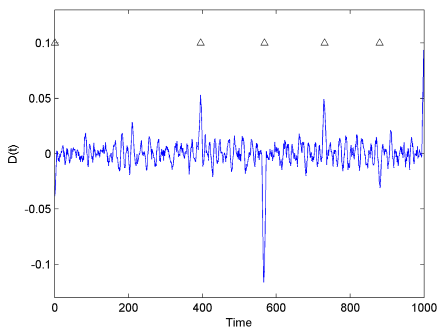
The central idea of the First-Level algorithm lies in the observation that changes in the original signal are represented by sharp spikes in the corresponding first-time derivative. This means that change-points can be seen as positive or negative peaks of the derivative signal. Note that the magnitude of the derivative signal depends not on the magnitude of the original signal but on its rate of change. In addition, the derivative is equally sensitive to a signal change irrespective of the level of the original signal. Thus, for two signals presenting changes in the mean value with similar rates of changes but with different orders of magnitude, the derivative will produce similar values for both of them [84]. All these properties make the derivative appealing for detecting changes in the time-series. However, the process of computing the time derivative inherently increases the noise that is present in the original signal. Therefore, the differentiation has to be preceded by filtering in order to reduce the noise and enhance the prominent interest points of the time-series. We should mention here that the idea of exploiting the extrema points of the filtered derivative signal in the change-point detection problem has been investigated again in some recent research works with very promising results [85–87].



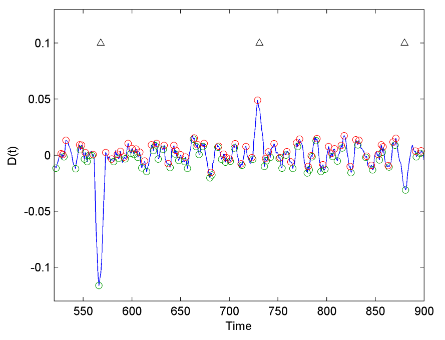
(a) Raw data



(b) Filtered data



(c) First-time derivative of the filtered data



(d) Potential change-points in the derivative of the filtered data (only a fragment is shown here)

Figure 3.2: Figure (a) depicts the original signal corrupted by noise, Figure (b) illustrates the filtered data, Figure (c) shows the first-time derivate of the filtered signal and Figure (d) presents the potential change-points as peaks of the derivative of the filtered data. The true segmentation boundaries are represented by black triangles, the detected local maxima by red circles and the detected local minima by green circles.

To sum up, the change-point detection problem can be effectively transformed into a peak detection problem, where by identifying the local maxima and minima of the derivative of the filtered signal, a list of potential change-points can be obtained. Hence, the First-Level of our algorithm is divided into two main steps: the filtering and derivative computation step and the extrema detection step.

Figure 3.2 provides an example of our First-Level algorithm. The benefit of using the First-Level is more than obvious here: the original set of 1000 points is very quickly reduced to only 350 points. The list of potential change-points contains all the right change-points as well as many false detections. In the Second-Level of our algorithm, a test will be carried out to remove the false detections from the list of candidates obtained in Level 1, without removing the already correctly detected points.

First-time Derivative of the Filtered Signal

Unlike existing methods which utilize filters "inspired" from either domain knowledge or intuition, we will use the approach described in [88], where the filter is optimized based on the time-series. Major advantage of this technique is that the filter is adapted to the particular dataset at hand. Moreover, as it was proven in [88], the underlying filter optimization problem is reduced to a generalized eigenvalue problem and therefore it admits a

Algorithm 1 Potential Extrema Determination

```
**Input:** $D$ - First-time derivative of the filtered signal of size $d \times n$
**Output:** $L$ - A cell containing the extrema positions per dimension that were selected from the algorithm as potential extrema

for j=1 to d do {for every dimension of the input signal}
    {Identify extrema points}
    $x \leftarrow 0$
    for i=2 to n-1 do
        if $D[j,i] > D[j,i-1] \wedge D[j,i] > D[j,i+1]$ then
            $x \leftarrow x+1$
            $a[j,x] = D[j,i]$ {Maxima amplitudes}
            $p[j,x] = i$ {Maxima positions}
        end if
        if $D[j,i] < D[j,i-1] \wedge D[j,i] < D[j,i+1]$ then
            $x \leftarrow x+1$
            $a[j,x] = D[j,i]$ {Minima amplitudes}
            $p[j,x] = i$ {Minima positions}
        end if
    end for
    $L\{j\} \leftarrow p[j,1:x]$
end for
```

simple, tractable solution. Besides that, this filter has the ability to preserve the most robust extrema, i.e., the most desirable extrema that remain identifiable and unaffected when distortions are introduced into the signal. This is the reason why this filter is also referred to as the "optimal" filter in this context. Examples of the filter's frequency response using four different datasets are given in Figure 3.4, where we can see that the filter shows a different behavior depending on the data used.

Figure 3.3 illustrates why it is important to use a filter that can be adapted to different datasets. Filtering too little will produce a time-series that shows too many extrema points, whereas, filtering too much will remove too much of the interesting signal, such that important points may be overlooked [89].

The used filtering method is based on the observation that the extrema selection process is equivalent to a geometric problem of partitioning data points in a hyperspace. Then, the filtering operation can be interpreted as bounding the selected extrema by two hyperplanes. The filter is basically an acausal linear FIR filter and the formulation problem is transformed into that of determining the "optimal" coefficients for each particular time-series data. It should be noted, however, that the filtering operation, although is data-adaptive, may cause a loss of information in some cases. A complete explanation of how this filter is derived is out of the scope of this thesis; we refer the interested reader to [88].

Once the filtered signal has been obtained, its first-time derivative can be computed very easily and the problem of finding change points in the original signal can be reduced to finding positive and negative peaks in the derivative signal.

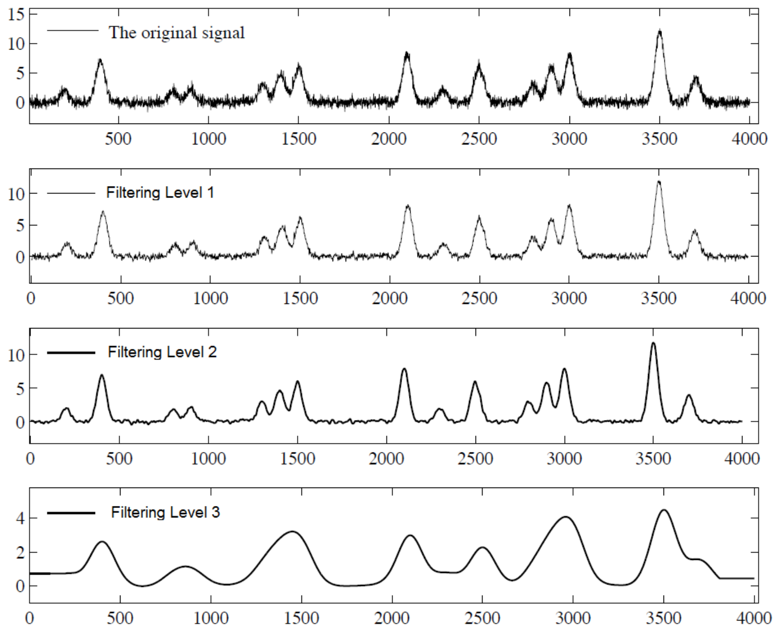
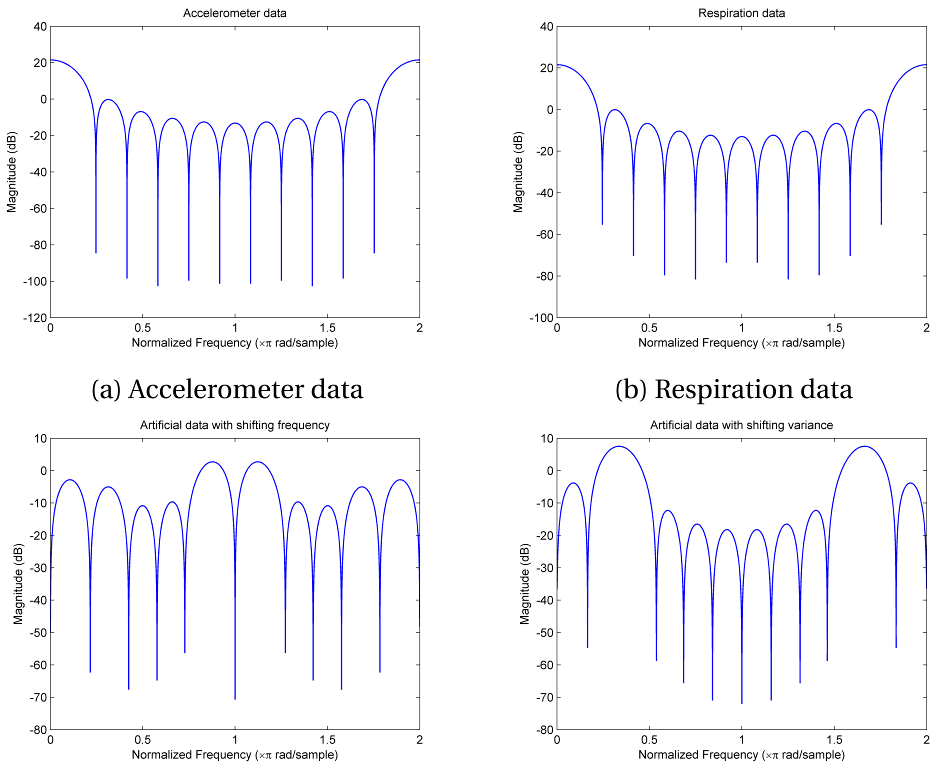


Figure 3.3: The first picture shows the original time-series without any filtering; the following pictures show the filtered time-series with increasing filtering scales, where the second picture still contains considerable noise, the third suppresses the noise and preserves the interesting points, while the fourth suppresses both the noise and some of the interesting points. [89]



(a) Accelerometer data (b) Respiration data

(c) Artificial data with shifting frequency (d) Artificial data with shifting variance

Figure 3.4: Frequency response of the filter for accelerometer data, respiration data , artificial data with shifting frequency and artificial data with shifting variance.

Change-point detection as peak detection of the filtered derivative

We used a two-step process for extrema detection. In the first step, all the extrema candidates are detected as described in the pseudocode given in Algorithm 1. However, not all these extrema contain important information for detecting changes, and hence,the second step involves pruning from the list of obtained extrema in order to cull the "weak" candidates which would easily appear or disappear depending on the distortions introduced into the underlying signal. Figure 3.5 shows an example of the extrema elimination step using the same time-series data as in Figure 3.2. For this case, the initial list of 437 candidates is reduced to 350 candidates, which offers an additional benefit of avoiding unnecessary computations in the following steps of the algorithm.

For elimination of the non-significant extrema we take into account the amplitude differences of the identified extrema with their neighbors as they were defined in [88], but we also consider the proximity position information. More specifically, the significant extrema determination is based on the following rule:

If two adjacent extrema are too close, then the extrema that has the lowest absolute diferrence to its neighbors is considered to be not significant and it is discarded from the list of candidates. (3.2)

The threshold, T_d, used to decide which extrema are considered to be too close was set equal to the median value of the distances between the positions of the obtained extrema and the amplitude differences of the extrema with their neighbors were computed as:

$$A_i = |a_i - a_{i-1}| + |a_i - a_{i+1}|, \quad i = 2, \ldots, x-1 \tag{3.3}$$

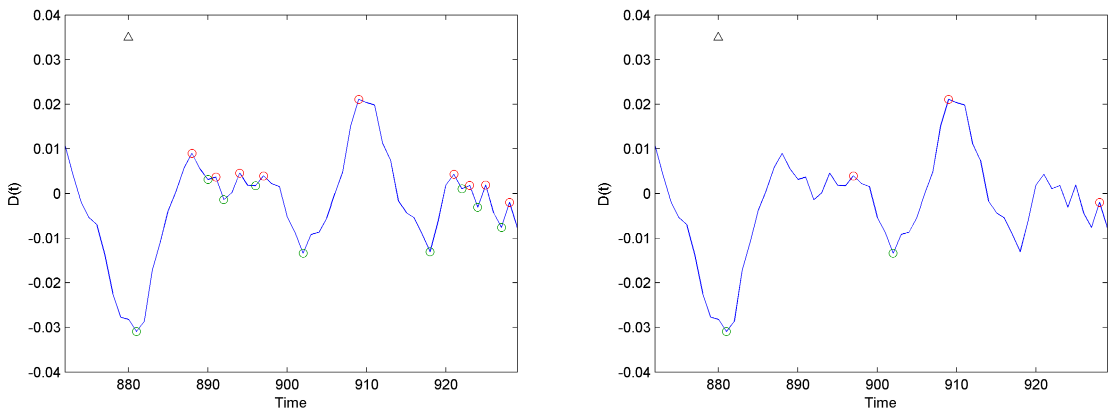
where a_i is the amplitude of the i^{th} detected extrema and x is the total number of extrema. Having defined these quantities, the rule 3.2 can be mathematically formulated as follows:

if $p_{i+1} - p_i \leq T_d$, then the candidate p_c, where $c = \arg\min(A_i, A_{i+1})$ is discarded from the list of candidates. (3.4)

In the above equation, the terms p_i and p_{i+1} represent two consecutive candidate positions. It is also possible to add an upper limit on the number of significant extrema. This is very useful, especially in problems like change detection, where not all changes are necessarily linked with significant extrema in the derivative signal. In our problem, we decided to use a limit such that 3.5% of the points of the time-series are chosen as significant extrema. Obviously, this is applicable only in case the number of potential extrema is sufficient enough to reach that limit.

3.4. SECOND-LEVEL ALGORITHM

The Second-Level detection test is used to remove the incorrectly identified change-points from the list of candidates and fine-tune the segmentation results. In order to com-



(a) Both significant and insignificant extrema (b) Only Significant extrema

Figure 3.5: Extrema detection step.

pute the detection measure defined in Equation 3.1, we first need to perform a data selection test in order to decide which data are important for detecting changes and compute a dissimilarity measure per each dimension by using the corresponding candidates from the First-level. We can then compute the final detection measure as the weighted sum of the dissimilarity measures computed independently for each dimension. The next sections thoroughly describe each step of the Second-Level algorithm and explain its suitability for the problem of change-point detection.

3.4.1. DATA SELECTION STEP

Figure 3.6 provides an illustrative example of two intervals $\mathcal{Y}$ and $\mathcal{Y}'$ derived from a 4-dimensional time-series, where only three of the dimensions are associated with a change, and indicates why it is important to perform the final detection test using only relevant data. Motivated by this observation, we used a data selection algorithm, which has been designed to give more weight (i.e., importance) to the data that are more important for predicting changes than to others. The importance of the data is represented by the coefficients $a_k \in [0,1]$ in Equation 3.1, with 1 indicating highest importance and 0 lowest importance. For example, in Figure 3.6 the data of the fourth dimension will acquire a small weight and, thus, the detection accuracy will not be negatively affected. This approach is also very useful for human activity change detection problems. For example, when detecting the change from "Standing" to "Brushing teeth" using multi-sensor data, it is reasonable to use only right/left hand information into the detection procedure [65].

Various approaches have been suggested in the literature to address the feature selection problem including the Least Absolute Shrinkage and Selection Operator (Lasso) based algorithms. In this section, we first review the Lasso based feature selection methods and indicate their limitations and then propose a method to overcome these limitations. Note here that although these algorithms are generally called feature selection algorithms, the selection in our problem is performed using raw data, and this is the reason why we use the term *data selection* instead.

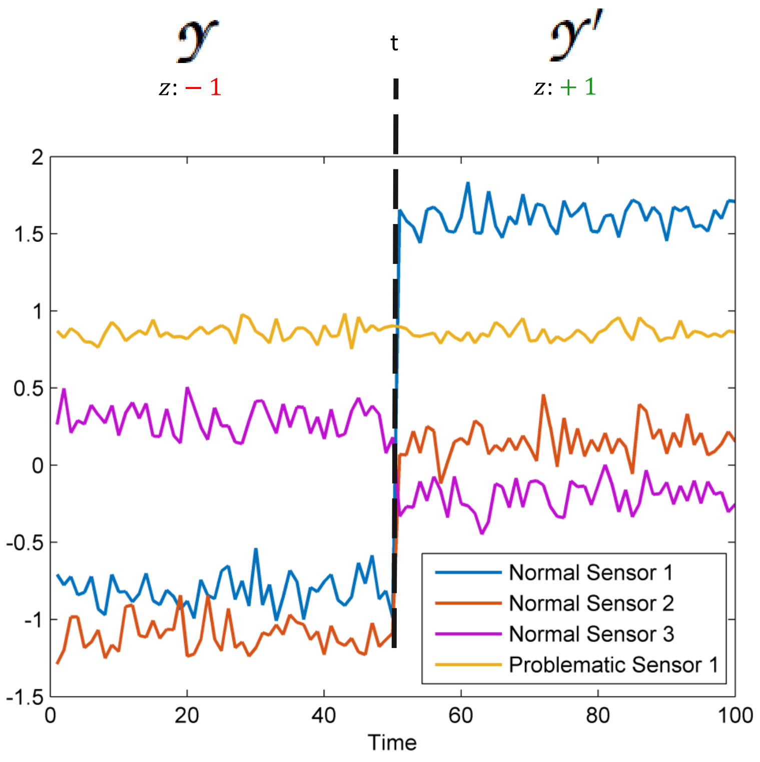


Figure 3.6: Example of two sequences $\mathcal{Y}$ and $\mathcal{Y}'$ as they were extracted from a 4-dimensional time-series. As we can see only three sensors provide data that are relevant to the observed change.

Before describing how these algorithms work, let us first introduce the special notation that will be used for their mathematical formulation. Using a pseudo-binary label $z_i \in \{+1, -1\}$, the samples in $\mathcal{Y}$ are annotated as $\mathbf{z} = [-1, ..., -1]^T \in \mathbb{R}^v$ and the samples in $\mathcal{Y}'$ as $\mathbf{z} = [1, ..., 1]^T \in \mathbb{R}^v$ and the whole set of labels is denoted as $\mathbf{z} = [z_1, ..., z_n]^T \in \mathbb{R}^n$.

Lasso (Least Absolute Shrinkage and Selection Operator) [90] is a computationally efficient linear feature selection method that optimizes the cost function:

$$\min_{\alpha \in \mathbb{R}^d} \quad \frac{1}{2} \|\mathbf{z} - \mathbf{Y}^T \boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \tag{3.5}$$

where $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_d]^T$ is a coefficient vector, α_k is the regression coefficient of the k-th feature, $\|\cdot\|_1$ and $\|\cdot\|_2$ are the ℓ_1- and ℓ_2-norms, and $\lambda > 0$ is the regularization parameter. The regression coefficients in Lasso represent the relevance of the corresponding features to a change and they become zero for irrelevant features. Lasso is known to scale well with both the number of data and the dimensionality. However, it can only capture linear dependency between input features and output values (labels) and this is a critical limitation [53, 91].

HSIC Lasso (Hilbert Schmidt Independence Criterion Lasso) [53] can select features from high-dimensional data in a nonlinear manner by optimizing the cost function:

$$\begin{aligned} \min_{\alpha\in\mathbb{R}^d} \quad & \|\bar{\mathbf{L}} - \sum_{k=1}^{d} \alpha_k \bar{\mathbf{K}}^{(k)}\|^2_{Frob} + \lambda \|\boldsymbol{\alpha}\|_1 \\ \text{s. t.} \quad & \alpha_1, \ldots, \alpha_d \geq 0 \end{aligned} \tag{3.6}$$

where $\bar{\mathbf{K}}^{(k)} = \boldsymbol{\Gamma}\mathbf{K}^{(k)}\boldsymbol{\Gamma}$ and $\bar{\mathbf{L}} = \boldsymbol{\Gamma}\boldsymbol{L}\boldsymbol{\Gamma}$ are centered and normalized Gram matrices $\mathbf{K}^{(k)}_{i,j} = K(y_{k,i}, y_{k,j})$ and $\mathbf{L}_{i,j} = L(z_i, z_j)$ are Gram matrices, $K(y, y')$ and $L(z, z')$ are kernel functions, $\boldsymbol{\Gamma} = \boldsymbol{I}_v - \frac{1}{v}\mathbf{1}_v\mathbf{1}_v^T$ is the centering matrix, $\boldsymbol{I}_v$ is the v-dimensional identity matrix, $\mathbf{1}_v$ is the v-dimensional vector with all ones, and $\|\cdot\|_{Frob}$ is the Frobenius norm. After estimating α, each of its elements is normalized as $\alpha_k \leftarrow \alpha_k / \sum_{k=1}^{d} \alpha_k$, such that $\sum_{k=1}^{d} \alpha_k = 1$. Note that due to the normalization, the use of these coefficients in one-dimensional data does not provide any information into the detection test (i.e., $\alpha = 1$).

The first term in Equation 3.6 means that we are regressing the output kernel matrix $\bar{\mathbf{L}}$ by a linear combination of feature-wise input kernel matrices $\left\{\bar{\mathbf{K}}^{(k)}\right\}_{k=1}^{d}$. This term can be rewritten as [53]:

$$\|\bar{\mathbf{L}} - \sum_{k=1}^{d} \alpha_k \bar{\mathbf{K}}^{(k)}\|^2_{Frob} = \text{HSIC}(\mathbf{z},\mathbf{z}) - 2\sum_{k=1}^{d} \alpha_k \text{HSIC}(\mathbf{y}_k^T, \mathbf{z}) + \sum_{k,l=1}^{d} \alpha_k \alpha_l \text{HSIC}(\mathbf{y}_k^T, \mathbf{y}_l^T) \tag{3.7}$$

where $\text{HSIC}(\mathbf{z},\mathbf{z}) = tr(\bar{\mathbf{L}}\bar{\mathbf{L}})$ and $\text{HSIC}(\mathbf{y}_k^T, \mathbf{y}_k'^T) = tr(\bar{\mathbf{K}}^{(k)}\bar{\mathbf{K}}^{(k')})$ are kernel-based independence measures called as HSIC, and $tr(\cdot)$ denotes the trace. Note that HSIC always takes a non-negative value , and is zero if and only if two random variables are statistically independent when a universal reproducing kernel [92] such as the Gaussian kernel is used. The HSIC measure is further described in subsection 3.4.1.1.

From Equation 3.7, if the k-th feature $\mathbf{y}_k^T$ has high dependence on the output $\mathbf{z}$, $\text{HSIC}(\mathbf{y}_k^T, \mathbf{z})$ takes a large value and thus α_k will also be large. On the other hand, if $\mathbf{y}_k^T$ and $\mathbf{z}$ are independent, $\mathbf{y}_k^T$ tends not to be selected by the ℓ_1- regularizer, which means that $\text{HSIC}(\mathbf{y}_k^T, \mathbf{z})$ and α_k are close to zero. Moreover, if $\mathbf{y}_k^T$ and $\mathbf{y}_l^T$ are strongly dependent (i.e., redundant features), $\text{HSIC}(\mathbf{y}_k^T, \mathbf{y}_l^T)$ becomes large and thus either α_k or α_l tends to be zero. That is, non-redundant features that have strong dependence on the output $\mathbf{z}$, and hence are important for a change, tend to be selected by the HSIC Lasso and this is a very important property for the change-point detection problem.

HSIC Lasso outperforms existing feature selection methods in small and high-dimensional settings. However, it tends to be expensive compared to the simple Lasso for large-n and high-d data [91]. In [53], a table lookup based approach was proposed to reduce the memory usage but the computational cost was still large. Another limitation of HSIC Lasso is that it needs to tune the regularization parameter λ which is usually difficult to be set manually.

In [65], where feature selection is used for the first time in the change-point detection context, the HSIC Lasso problem is solved using the Dual Augmented Lagrangian (DAL) technique [93]. However, the computational expense of this approach makes it impractical for real-size problems. In this work to deal with a large and high-dimensional problem and efficiently solve the HSIC Lasso we use the NN-LARS (Non-Negative Least Angle Regression and Selection) algorithm proposed by [94]. A major advantage of the LARS based formulation over Lasso is that LARS can find m features by iterating over m steps while Lasso requires fine tuning the regularization parameter λ to obtain m features, which is a very expensive procedure in large and high-dimensional problems. In addition, by using the NN-LARS algorithm, the entire regularization path can be found for the cost of an ordinary least squares solution [91, 94].

3.4.1.1. HILBERT-SCHMIDT INDEPENDENCE CRITERION (HSIC)

The HSIC (Hilbert-Schmidt independence Criterion) was first proposed in [95] for measuring the dependence between two kernels. This criterion can be applied in feature selection algorithms to select a subset of features, such that the kernel constructed using the selected feature subset maximizes HSIC when compared to a given kernel.

In order to compute HSIC we need to express it in terms of kernel functions. A universal kernel such as the Gaussian kernel allows HSIC to detect dependence between two random variables [95]. Furthermore, it has been shown that the delta kernel is useful for classification problems [96]. Therefore, we use the Gaussian kernel for inputs **y** and the delta kernel for the outputs (labels) **z**. Before computing the kernel, the input data **y** need to be normalized to have unit standard deviation. Then, we can use the Gaussian kernel:

$$K(y, y') = \exp\left(-\frac{(y-y')^2}{2\sigma_y^2}\right) \tag{3.8}$$

where σ_y is the Gaussian kernel width. In this work we set σ_y equal to the median distance between points in the aggregate sample.

For the outputs (labels) z we use the delta kernel:

$$L(z, z') = \begin{cases} 1 & \text{if } z = z' \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

3.4.2. JENSEN SHANNON DIVERGENCE (JSD) BASED ON K-NEAREST NEIGHBOR (kNN) DENSITY ESTIMATION

In order to compute the final weighted detection measure and decide whether there is a change or not, we should first compute a dissimilarity measure for each dimension of the data using the corresponding candidates from the First-Level. More specifically, we need to compare the sequences $\mathcal{Y}$ and $\mathcal{Y}'$ of v samples each, which are extracted in a sliding window manner at the candidate locations. A brief description of some of the most well-known non-parametric dissimilarity measures was provided in Section 2.4. Of particular interest are the recently proposed direct density estimation methods that either compute a density ratio or a density difference directly without estimating the two densities separately. However, although these methods can give very accurate results in many circumstances, they choose their parameters by means of cross-validation, which is a computationally expensive procedure to be performed for each candidate.

A cheaper, though less accurate, way would be to use a density estimation algorithm to estimate the densities corresponding to the two sequences separately, and then select an appropriate measure to compare the estimated densities. kNN density estimation is preferable over other ways of estimating a density, since it has the ability to adapt to the local sampling density. Nevertheless, as it was mentioned in Section 2.4, this kind of estimation has been known to be biased and similar to all density estimation methods, its accuracy is affected when estimating high-dimensional densities [71, 73]. Recently though, new methodologies have been proposed to cancel the bias and compute consistent kNN-based statistical measures [73]. In our setting, in particular, the disadvantages of kNN density estimation could be further mitigated by the fact that the dissimilarity measure is computed for each dimension separately and that one additional step (i.e., the First-Level algorithm) is used to find only the most relevant data for the density estimation procedure.

For all reasons mentioned above, the dissimilarity measure of our change detection approach will be based in the kNN density estimation, and more specifically, in the kNN-based Jensen Shannon divergence (JSD) estimation, which was recently proposed by Vanlier et al [71]. JSD, which presents many appealing properties and is very useful for detection purposes, is defined as the averaged Kullback-Leibler divergence (KL) [97]:

$$\mathrm{JSD}(P,P') = \frac{1}{2}\mathrm{KL}\left(P,\frac{1}{2}\left(P+P'\right)\right) + \frac{1}{2}\mathrm{KL}\left(P',\frac{1}{2}\left(P+P'\right)\right) \tag{3.10}$$

where P and P' are the two compared densities. Nevertheless, considering that only some sample points of the densities are available, an exact calculation of the divergence is not possible, and thus, the proposed kNN-based JSD estimation aims to provide an accurate and efficient estimation of the original measure.

A complete explanation of this approach, which was originally developed to compute the JSD between multivariate predictive densities of two competing models for optimal experiment design purposes, can be found in [71]. Although we will not give the details of this algorithm here, we should mention that it uses Markov Chain Monte Carlo (MCMC) to sample from the posterior distributions and these posterior distribution values are used in turn to sample from the predictive distributions, which was found to be a computationally intensive procedure [98]. To keep the computational burden low and make as less assumptions as possible, we do not use any prediction method and instead we compute the non-parametric kNN-based JSD measure directly from the two immediate sets of observations $\mathcal{Y}$ and $\mathcal{Y}'$.

It should also be noted that this method is limited by its dependency on the value k, i.e., the number of nearest neighbors that will be used in the density estimation. One way to deal with this, would be to use cross-validation over multiple folds to find the best value for k, but this would increase the running time of the algorithm. In this work, we propose a fast and automatic data-driven k value selection method, which sets the number of nearest neighbors k equal to the Euclidean pairwise distance between the two comparing sets of observations, $\mathcal{Y}$ and $\mathcal{Y}'$.

3.5. MODIFIED ALGORITHM FOR ONLINE DETECTION

As it was mentioned in Section 1.3, one of the numerous applications of change-point detection algorithms is their use in online tracking systems. The hierarchical structure and the low computational time of the two-level algorithm proposed in this thesis make it appealing for handling streaming data in real-time applications. However, since the First-Level requires a larger number of observations than the Second-Level, where a sliding window of a few positions at a time is being utilized, the algorithm needs to be adapted in the online context. For this purpose, we exploit the idea of the Sliding Window And Bottom-up (SWAB) algorithm [44], which enables the combination of a sliding-window test and a bottom-up test through a buffer, so that the advantage of having a more global view of the data stream will not disappear.

Thus, by using a buffer to store a sufficient number of observations in the same manner as in SWAB algorithm, the proposed Two-Level change detection test can be applied in the set of observations that is currently present in the buffer, allowing a fast detection of changes. Once the data that were stored in the buffer have been processed, they are deleted and the buffer slides to the oldest received sample point that has not yet been processed. Note that we have assumed here a perfect synchronization of the data across the multimodal sensors. However, in practice, we should first synchronize the data since the sampling rate varies among different kinds of sensors [99].

3.6. SPEEDING-UP EXISTING CHANGE-POINT DETECTION ALGORITHMS

In this section we explain how our algorithm could be exploited to speed up some standard change-point detection techniques and improve their performance. One way to achieve that, would be to use the same hierarchical structure, described in the previous sections, where only the dissimilarity measure that is computed in the Second-Level would be replaced. That is, we could compute another dissimilarity measure, instead of the Jensen Shannon divergence (JSD) based on k-nearest neighbor (kNN) density estimation. Then, the final detection measure would be given as the weighted sum of these dissimilarity scores that were computed separately for each dimension of the data using their corresponding candidates from the First-level.

However, many of the widely used methods require extremely large computational time even for the simple case of one-dimensional data. Therefore, a combination of the above idea with these computationally expensive methods, would increase dramatically the computational time for large sample size and high-dimensional data, despite the fact that not all the data samples would be used for the underlying computations. Hence, the above modification can be efficiently used only with dissimilarity measures that require a relatively small amount of time. One such measure is the Hilbert-Schmidt Independence Criterion (HSIC), which has been utilized in the change-point detection problem by the

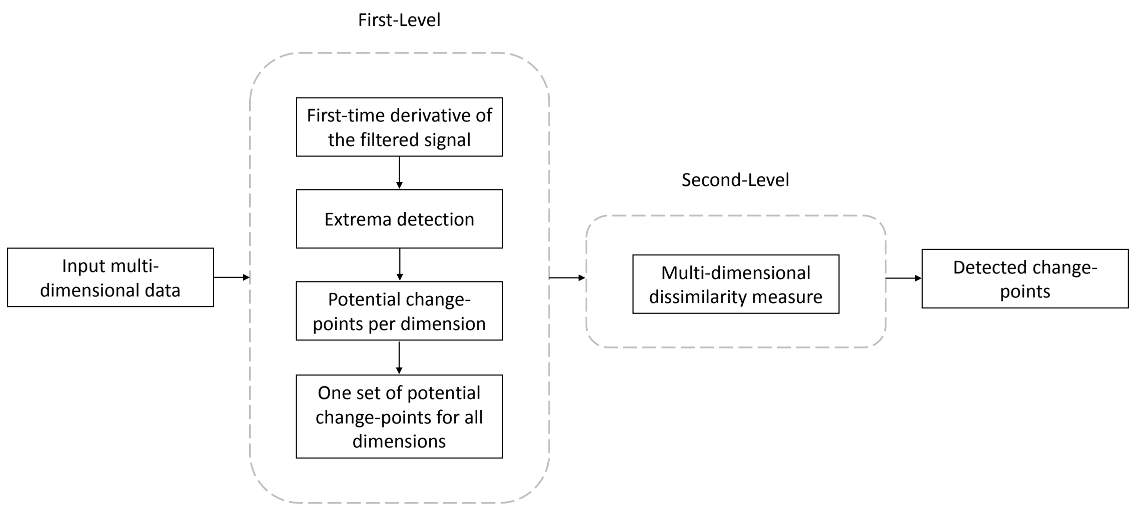

First-Level
First-time derivative of the filtered signal
Extrema detection
Potential change-points per dimension
One set of potential change-points for all dimensions
Input multi-dimensional data
Second-Level
Multi-dimensional dissimilarity measure
Detected change-points

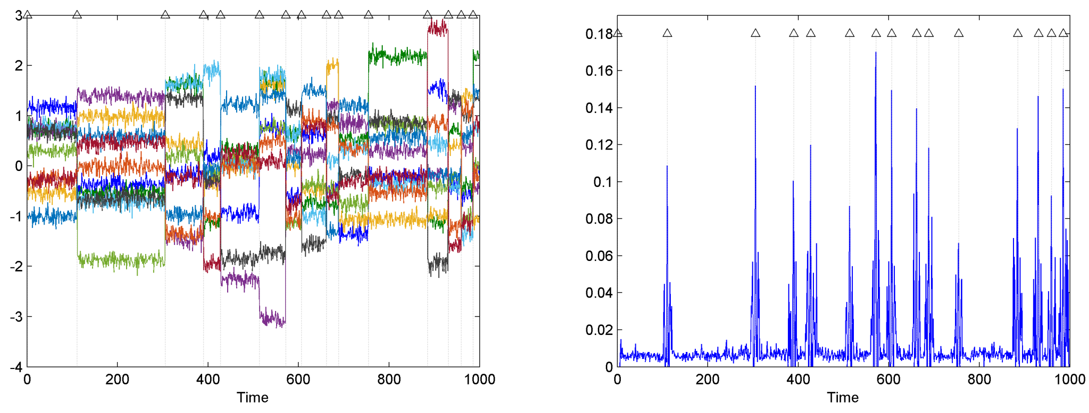
Figure 3.7: The architecture of the modified Two-Level change-point detection algorithm using multivariate dissimilarity measures.

additive HSIC (αHSIC) method. Thus, by combining the Two-Level change detection algorithm presented in this thesis with the HSIC test, we propose a new detection measure called Hierarchical additive Hilbert-Schmidt Independence Criterion (HαHSIC).

Let us now explain how the problem arises in the high-dimensional case. We assume one-dimensional data of length n and computational cost $n \cdot O(l)$, when applying one of the original existing methods (without any speeding up procedure). If we combine our Two-Level algorithm with one of these methods, then the computational cost will be reduced from $n \cdot O(l)$ to $N \cdot O(l)$ (where $N << n$). However, if we increase the data dimensionality to m, the computational complexity of the modified method becomes approximately $m \cdot N \cdot O(l)$, which, for high-dimensional data, is generally much larger than $n \cdot O(l')$, where $O(l')$ is the computational cost for multivariate data. Therefore, in order to speed up the methods that involve heavy computations we should adapt our First-Level algorithm in the multidimensional case in such a way that the cost of processing multivariate data, $n \cdot O(l')$, will be reduced to $N \cdot O(l')$, with $N << n$. The procedure that will be used is described in the following paragraphs and is depicted in Figure 3.7.

Modified First-Level algorithm for multivariate data

In order to speed up the change-point detection procedure for the computationally expensive multivariate methods, we first apply our First-Level algorithm, which processes each dimension separately and returns a list of potential candidates per dimension. Then, we concatenate the individual lists into a single one, which contains all the candidates (locations) that were detected at least in one dimension. For example, if we have 2-dimensional data and the candidates (b, d, f, q, z) and (a, c, d, q, x) per dimension, respectively, then, we obtain a final list of candidates (a, b, c, d, f, q, x, z). Afterwards, we apply a modified version of the significant extrema detection process used in the last step of the original First-Level algorithm. In that step, we had taken into account the amplitude differences and the proximity position information of the identified extrema with their neighbors. However, now,



(a) 10-dimensional raw time-series data (b) The new amplitude differences vector

Figure 3.8: This Figure illustrates how the modified First-level algorithm is computed for a 10-dimensional time-series data. Figure (a) shows the raw data and Figure (b) shows the new amplitude difference vector that will be exploited for the detection of the final candidates. The triangles and the vertical dotted lines represent the true change-points and as we can see at these locations there are sharp spikes.

since we concatenated the individual lists of candidates into a single one, we should compute a new vector with the amplitude differences between the candidates of the final list and then apply the significant extrema determination rule 3.2, as it was explained in Section 3.3. More specifically, the modified First-Level algorithm for multidimensional data consists of the following steps:

1. The new list of candidates, L_{new}, is obtained by concatenating the individual lists $L(1),\dots,L(m)$, where m is the dimensionality of the data, into a single one, which contains all the candidates that were detected at least in one dimension of the data.

2. For each candidate of the list L_{new}, we retrieve all the dimensions where it was selected as a candidate, as well as the amplitude differences of this candidate with its neighbors for all detected dimensions, and then, we compute the mean value of the retrieved amplitude differences.

3. Finally, we apply the significant extrema determination rule 3.2 to remove the candidates that are too close to each other and have the lowest absolute differences to their neighbors, according to the new amplitude differences vector.

Here is an example to clarify the above procedure. Suppose we have 10-dimensional data and we examine the candidate e of the list L_{new} and let us assume that this candidate was detected in the 1^{st}, 2^{nd} and 4^{th} dimensions. Then, the new amplitude difference value of this candidate with its neighbors is computed as: $A_{new}(e) = \frac{A(1,e)+A(2,e)+A(4,e)}{3}$, where $A(1,e), A(2,e)$ ans $A(4,e)$ are the amplitude differences of the candidate e with its neighbors from dimensions 1,2 and 4, respectively. Figure 3.8 illustrates an example of the new amplitude difference values of a 10-dimensional time series data presenting changes in the mean value. As we can see, at the true change-points the amplitude differences have very large values, and, hence, this is a very effective way of identifying the most likely change-points. Note also that since the amplitude differences are computed in the derivative and not in the original signal, there is no possibility to add values of different orders of magnitude.

Once we have obtained the final list of candidates, we can compute a multivariate dissimilarity measure using only these candidates, instead of examining all the observations. Although this algorithm could be used in combination with all dissimilarity measures discussed in this thesis, our focus is on multivariate methods that have attracted a lot of attention in the literature the last years, as are the Least-Squares Density Difference (LSDD) and the Relative unconstrained Least Squares Importance Fitting (RuLSIF) approaches. The methods that are developed by using these two measures in combination with the modified First-Level algorithm presented in this section, are called Hierarchical LSDD (HLSDD) and Hierarchical RuLSIF (HRuLSIF), respectively.

In the experimental section, we demonstrate the effectiveness of all modified methods and we show that by using the First-Level algorithm either in combination with a univariate or a multivariate dissimilarity measure, we basically obtain the same detection results as with conventional approaches but with significant run-time improvements.

4 EXPERIMENTS

In this chapter we provide an experimental evaluation and comparison of the methods that we developed with some of the most prominent algorithms proposed in the literature for the change-point detection problem. We begin by presenting the metrics used for the performance evaluation and then apply the different algorithms on both synthetic and real-world datasets.

4.1. PERFORMANCE METRICS

One of the key issues associated with a change-point detection problem is the selection of appropriate performance measures, or "metrics", that can ensure a meaningful analysis of the system's behavior. The most widely-known criteria used to evaluate the performance of a change detection algorithm usually originate from information retrieval, machine learning and data mining fields. Change-point detection could actually be seen as a classification problem, where in each time step the system has to decide whether a change occurs (S^+) or not (S^-). A standard approach to evaluate such a system is to construct a confusion matrix, as illustrated in Table 4.1, with the following entries:

- ***True Positive (TP)***: The number of change-points that are correctly identified as change-points.
- ***False Positive (FP)***: The number of non change-points that are incorrectly identified as change-points.
- ***False Negative (FN)***: The number of change-points that are incorrectly identified as non change-points.
- ***True Negative (TN)***: The number of non change-points that are correctly identified as non change-points.

However, the strategy that assigns a ground truth label in every point at the time-series and forms a vector $s \in \{S^+, S^-\}_{1:n}$, where n is the length of the data, is rather a naïve approach for the change-point detection problem, since the temporal adjacency is not taken

		Ground Truth	
		Change-point	Non change-point
Predicted	Change-point	TP	FP
	Non change-point	FN	TN

Table 4.1: Confusion matrix.

into account. For almost all applications, a segmentation $s = S^+$ at time t would be considered as a good-enough hit if the target segmentation point S^+ was located in the immediate neighborhood $(t \mp \varepsilon) = S^+$, where ε is a tolerated deviation. But, in the above-mentioned naïve approach, such a result would lead to both a false positive and a false negative result, as there is not an exact match between the ground truth and the predicted labels. This means that the evaluation metric has to be modified to incorporate temporal neighborhood in a small area around a real segmentation point. Furthermore, the evaluation result depends not only on a particular segmentation decision in time, but on the result in conjunction with the predicted labels in the temporal neighborhood; or in other words, while one segmentation at the right location is desirable, multiple segmentation points at the same location have to be penalized [100].

One more issue associated with the change-point detection problem is that in most cases there are many more points where there is no change (S^-) than points where there is a change (S^+), i.e., we have a heavily imbalanced dataset regarding the class distribution. Hence, some basic evaluation measures, such as the accuracy, which do not take into consideration the distribution of the classes, are invalid in our problem context. Another one well-known classification measure that is problematic in our case is the Receiver Operating Characteristic (ROC) curve, which describes the relationship between the False Positive Rate (FPR):

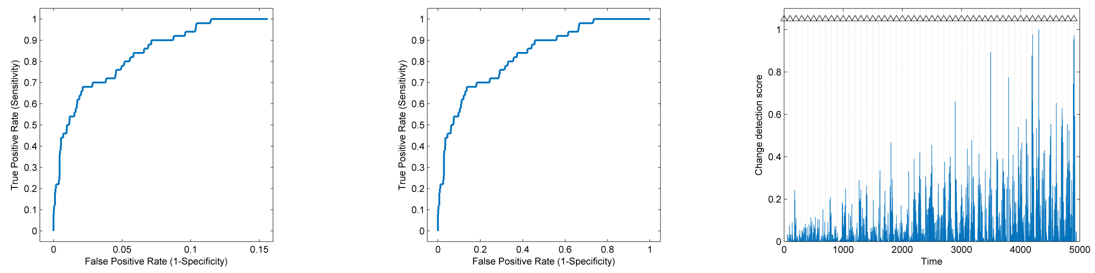
$$FPR = \frac{FP}{FP + TN}$$

and the True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN}.$$

The problem here is that, because of the small values that FPR adopts, only a small area of the ROC curve is covered [100]. For that reason, in 2012, Kawahara and Sugiyama [56], defined the True Positive Rate and False Positive Rate in the following way:

$$\begin{aligned} TPR &= \frac{n_{cr}}{n_{cp}} \\ FPR &= \frac{n_f}{n_{al}} \end{aligned} \tag{4.1}$$



(a) ROC curve computed with the original TPR and FPR definitions (b) ROC curve computed with the TPR anf FPR definitions of Kawahara and Sugiyama (c) Change detection score

Figure 4.1: Figures (a) and (b) show the ROC curves computed with the original classification measures and the measures of Kawahara and Sugiyama, respectively, and Figure (c) depicts the corresponding change detection score. The true change-points are marked by vertical dotted lines and triangles.

where n_{cr} denotes the number of times change-points are correctly detected, n_{cp} the number of all change points, n_f the number of times non change-points are detected by mistake and n_{al} the number of all detection alarms.

Figure 4.1 shows how the ROC curve differs when using the original classification measures instead of those of Kawahara and Sugiyama. As it can be observed, the maximum value of FPR is around 0.15 when using the original measures. In this case, by extending the FPR to reach the point $(1,1)$ in order to calculate the area underneath, we will obtain an AUC value equal to 0.974 (against to 0.831 when using the measures of Kawahara and Sugiyama), which is clearly unrealistic, as it can be seen by the change detection score used to create these plots.

In this work, in order to avoid providing overly optimistic results and solve the problems associated with imbalanced datasets, we decided to use the recently proposed definitions of TPR and FPR, as they were given above.

4.2. EXPERIMENTS ON ARTIFICIAL DATASETS

First, we examine how the investigated methods behave in different scenarios using two groups of artificial datasets with fixed and random change-steps. The data with fixed change-steps are denoted as "Group 1" and the data with random change-steps as "Group 2".

4.2.1. DATA WITH FIXED CHANGE-STEP

In this section, we describe three synthetic time-series datasets as they were introduced in [101]. The datasets are defined in the univariate context and will later be extended to the multivariate case:

- ***Dataset 1 (Jumping mean)***: The following 1-dimensional second-order autoregressive model is used to generate n data samples:

$$y(t) = 0.6 \cdot y(t-1) - 0.5 \cdot y(t-2) + \varepsilon_t \tag{4.2}$$

where ε_t is a Gaussian distribution modeling the noise with mean μ and standard deviation 1.5. The initial values are set as $y(1) = y(2) = 0$. A change-point is inserted at every 100 time steps by setting the noise mean μ at time t as:

$$\mu_N = \begin{cases} 0 & N = 1 \\ \mu_{N-1} + \frac{N}{16} & N = 2, \ldots, \frac{n}{100} - 1 \end{cases}$$

where N is a natural number such that $100(N-1)+1 \le t \le 100N$, giving a total $\frac{n}{100}$ of change-points.

- ***Dataset 2 (Scaling variance)***: The same auto-regressive model as in Dataset 1 is used, but here the noise mean is equal to 0 and a change-point is inserted at every 100 time steps by setting the noise standard deviation σ at time t as:

$$\sigma_N = \begin{cases} 1 & N = 1, 3, , \ldots, \frac{n}{100} - 1 \\ ln\left(e + \frac{N}{4}\right) & N = 2, 4 \ldots, \frac{n}{100} - 2 \end{cases}$$

- ***Dataset 3 (Changing frequency)***: 1-d dimensional data samples of size n are generated as:

$$y(t) = sin(\omega x) + \varepsilon_t \tag{4.3}$$

where ε_t is a origin-centered Gaussian noise with standard deviation 0.8. A change-point is inserted at every 100 points by changing the frequency ω at time t as:

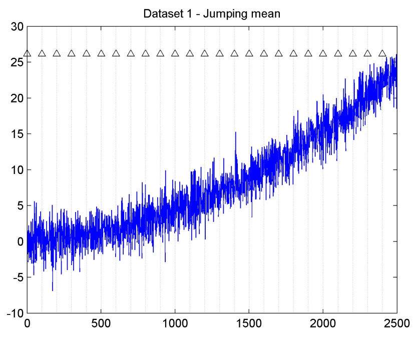
$$\omega_N = \begin{cases} 1 & N = 1 \\ \omega_{N-1} ln\left(e + \frac{N}{4}\right) & N = 2, 4 \ldots, \frac{n}{100} - 1 \end{cases}$$

Note that in the above datasets later change-points are more significant than earlier ones. This allows us to explore the ability of each algorithm to detect change-points with different significance. Examples of these datasets are shown in Figure 4.2.

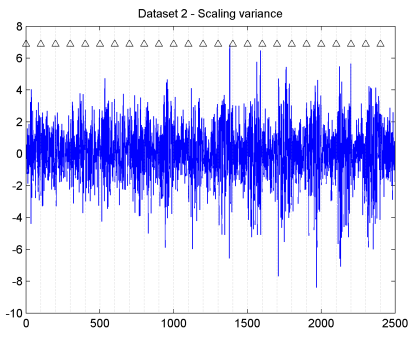
4.2.2. Data with random change-step

For a more complete comparison of the different techniques, we have also generated synthetic time-series datasets with random change-point positions and random change values. As in the previous section, the datasets are first defined in the univariate context:

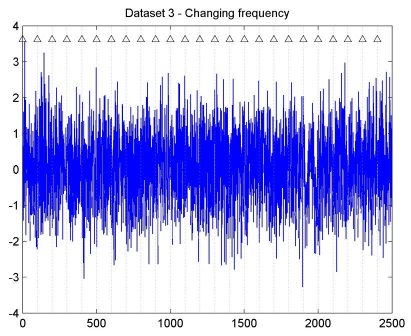
- ***Dataset 1 (Shifting mean)***: The following 1-dimensional model is used to generate n data samples characterized by changes in the mean value:



(a) Dataset 1: Jumping mean



(b) Dataset 2: Scaling variance



(c) Dataset 3: Changing frequency

Figure 4.2: Group 1 - Artificial time-series data with fixed change-step. The true change-points are marked by vertical dotted lines and triangles.

$$y_t = \begin{cases} \mu_1 + \varepsilon_t & \varepsilon_t \sim N(0, \sigma^2), \quad t = 1, ..., p \\ \mu_2 + \varepsilon_t & \varepsilon_t \sim N(0, \sigma^2), \quad t = p+1, ..., p' - 1 \end{cases} \tag{4.4}$$

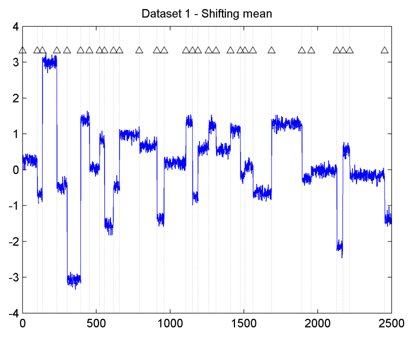
where y_t is the response variable, μ_1 and μ_2 are the means before and after the unknown change at time p, respectively, and ε_t is a Gaussian noise with mean 0 and standard deviation σ^2.

- ***Dataset 2 (Shifting variance)***: The following 1-dimensional model is used to generate n data samples presenting changes in the variance:

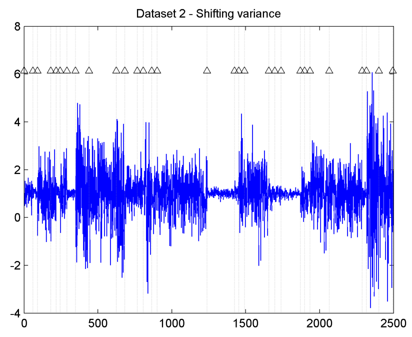
$$y_t = \begin{cases} \mu + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_1^2), \quad t = 1, ..., p \\ \mu + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_2^2), \quad t = p+1, ..., p' - 1 \end{cases} \tag{4.5}$$

where σ_1^2 and σ_2^2 are the standard deviations before and after the unknown change at time p, respectively, and μ is the overall mean.

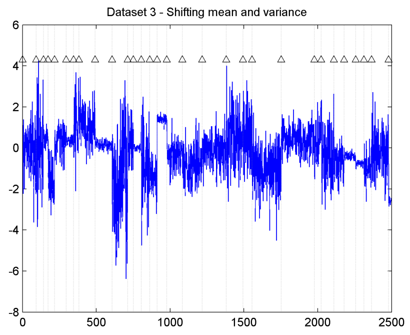
- ***Dataset 3 (Shifting mean and variance)***: The following 1-dimensional model is used to generate n data samples that experience changes in both the mean and the variance:



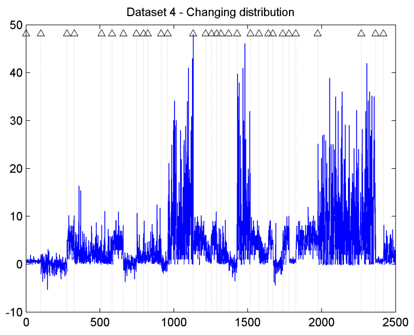
(a) Dataset 1: Shifting mean



(b) Dataset 2: Shifting variance



(c) Dataset 3: Shifting mean and variance



(d) Dataset 4: Changing distribution

Figure 4.3: Group 2 - Artificial time-series data with random change-step. The true change-points are marked by vertical dotted lines and triangles.

$$y_t = \begin{cases} \mu_1 + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_1^2), \quad t = 1, \ldots, p \\ \mu_2 + \varepsilon_t & \varepsilon_t \sim N(0, \sigma_2^2), \quad t = p+1, \ldots, p' - 1 \end{cases} \tag{4.6}$$

where μ_1, σ_1^2 and μ_2, σ_2^2 are the means and standard deviations before and after the unknown change at time p, respectively.

- ***Dataset 4 (Changing distribution)***: Time-series of this category are generated by using a sequence of independent random numbers, where the originating distributions before and after the unknown change-point positions are different. In this simulation, we use in total six distributions, namely Chi-Square, Exponential, Geometric, Poisson, Rayleigh and Student's T, and, thus, there are many possible transitions between different segments. Gaussian noise with mean 0 and standard deviation σ^2 is also added in these time-series data.

It is interesting to note that since the change-points have random positions in the above datasets, we can generate time-series with as many change-points as we want. However, in order to allow easy comparison among them we always inserted the same number of change-points (equal to 0.01 of the data length) in all time-series of this category. Examples of these datasets are given in Figure 4.3.

4.2.3. ARTIFICIAL DATA RESULTS

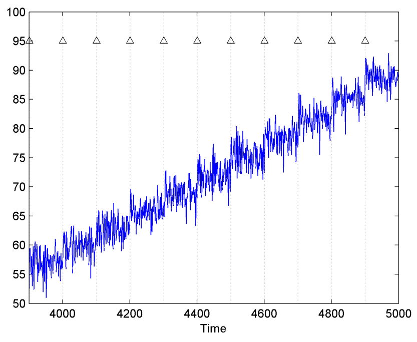
In this section, we illustrate the behavior of the proposed methods and compare their performance with three popular non-parametric algorithms from the categories of rank statistics (αHSIC), direct density-difference estimation (LSDD) and direct density-ratio estimation (RuLSIF) using synthetic datasets. More specifically, for each experiment we test 7 different algorithms: the proposed HαJSD (Hierarchical additive Jensen Shannon divergence), the existing approaches αHSIC, LSDD and RuLSIF, and their modifications HαHSIC (Hierarchical αHSIC), HLSDD (Hierarchical LSDD) and HRuLSIF (Hierachical RuLSIF), as they were presented in Section 3.6.

The experiments were conducted on a Linux-based cluster system of Philips Research using Matlab R2014a. For LSDD and RuLSIF we use publicly available codes [102] and [103], respectively. In addition, the parameters of these methods are chosen by grid search via 5-fold cross-validation and the α value in RuLSIF is set equal to 0.1. For data selection by HSIC Lasso in the αHSIC method we use the publicly available code [104] and we fix the regularization parameter λ at 0.01. Note that in the data selection procedure of the modified HαHSIC method, the HSIC Lasso problem is solved using the NN-LARS algorithm, as it was discussed in Section 3.4.1. For all methods which use a sliding window procedure, each window contains $v = 50$ sample points. Moreover, each of the experiments was run 50 times with different random seeds and the reported results are the averages calculated over all 50 runs.

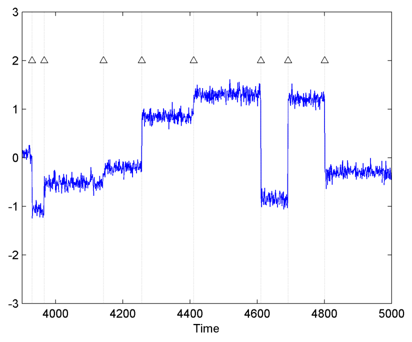
All examined methods are compared in terms of the ROC curves, the Area Under the ROC Curve (AUC) values and the time consumption. The AUC obtains a value in the range $[0,1]$ and is a very useful indicator of the performance. In general, the closer the ROC gets to the left-top corner, i.e., (FPR,TPR) = $(0,1)$, the closer the AUC gets to 1 and the better the algorithm is. For all considered methods, the ROC curves are constructed by examining the list of the obtained change-point scores. More specifically, a detection alarm at the time point t is regarded as correct if there exists a true alarm at point t^* such that $t \in [t^* - 10, t^* + 10]$. Following the strategy of the previous researchers [14, 105], peaks of a change-point score are regarded as detection alarms. We then setup a threshold η for filtering out all alarms whose change-point scores are lower than or equal to η. Initially, we set η to be equal to the score of the highest peak. Then by lowering η gradually, both TPR and FPR become non-decreasing. Finally, for each η, we plot TPR and FPR on the graph, and thus a monotone ROC curve is drawn. If we want to evaluate the performance of the First-Level of our algorithms separately, since in this case there is no change-point score computed, we simply compare the detected points with the ground truth ones.

Most research works on change-point detection considered only small datasets for the evaluation procedure (for example 5,000 sample points and only one dimension). In this work, in order to test the performance in high-dimensional data as well, we will extend the above artificial datasets in the multidimensional case, where the change-points occur at the same location among the different dimensions, while the data of each dimension are randomly generated by the model corresponding to each dataset. The effect of having large sample-size datasets will also be examined in our experiments.

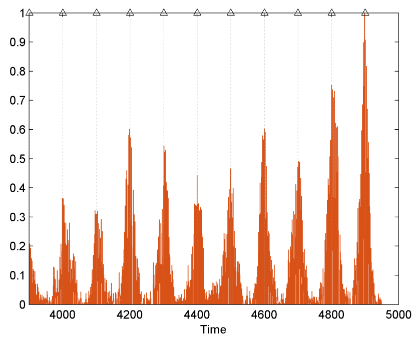
First, we consider one-dimensional time series with a fixed number of samples in order



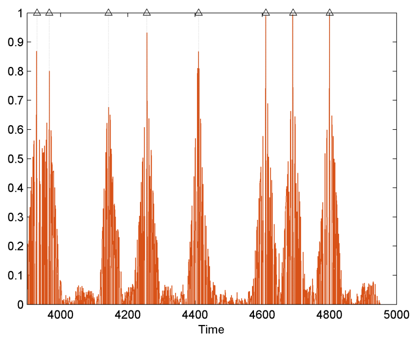
(a) Group 1: Jumping mean



(b) Group 2: Shifting mean



(c) Change-point score obtained by the HαJSD method for data in Figure (a)



(d) Change-point score obtained by the HαJSD method for data in Figure (b)

Figure 4.4: Time-series samples (upper) and the change-point score obtained by the HαJSD method (lower). The true change-points are marked by vertical dotted lines and triangles.

to explore how the algorithms perform on each specific dataset. Figure 4.4 shows examples of two datasets and the corresponding change-point score obtained by the HαJSD method. As can be seen, the change-points were correctly detected in both cases. Tables 4.2 and 4.3 describe the mean and standard deviation of the AUC values as well as the computational time over 50 runs, and Figure 4.5 illustrates ROC curves averaged over 50 runs with different random seeds for each of the 7 datasets.

	HαJSD			HαHSIC			αHSIC		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std	
Group 1									
Jumping mean	**0.836**	0.013	0.179	0.810	0.012	0.050	0.815	0.011	0.115
Scaling variance	0.855	0.019	0.186	0.856	0.016	0.066	**0.863**	0.015	0.124
Changing frequency	**0.786**	0.027	0.162	0.739	0.025	0.057	0.771	0.019	0.138
Group 2									
Shifting mean	**0.931**	0.021	0.140	0.914	0.018	0.042	0.893	0.018	0.167
Shifting variance	0.806	0.022	0.150	0.808	0.022	0.049	**0.815**	0.018	0.158
Shifting mean and variance	0.848	0.018	0.165	**0.852**	0.016	0.052	0.839	0.015	0.116
Changing distribution	**0.831**	0.018	0.184	0.821	0.024	0.055	0.823	0.021	0.114
Average	**0.842**	0.046	0.167	0.829	0.054	0.053	0.831	0.039	0.133

Table 4.2: Mean AUC values and standard deviation, and mean computational time of the methods that use a data selection technique combined with a univariate score for data of size $1 \times 5{,}000$. The best methods in terms of mean AUC values are described in boldface.

	HLSDD			LSDD			HRuLSIF			RuLSIF		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std		mean	std	
Group 1												
Jumping mean	0.833	0.014	0.863	**0.854**	0.013	2.542	0.818	0.016	1.702	0.829	0.013	4.182
Scaling variance	0.861	0.019	0.953	**0.884**	0.017	2.320	0.865	0.013	1.656	0.872	0.013	4.529
Changing frequency	0.772	0.024	0.975	**0.823**	0.020	2.503	0.762	0.023	1.532	0.806	0.018	4.414
Group 2												
Shifting mean	0.920	0.017	1.038	0.908	0.017	2.184	**0.929**	0.020	1.611	0.919	0.021	4.312
Shifting variance	0.797	0.026	0.796	0.798	0.022	2.065	0.822	0.021	1.634	**0.827**	0.018	4.091
Shifting mean and variance	0.829	0.019	0.926	0.812	0.018	2.177	**0.863**	0.018	1.601	0.850	0.017	4.046
Changing distribution	0.825	0.020	0.911	0.822	0.020	2.239	0.839	0.024	1.636	**0.843**	0.021	4.196
Average	0.834	0.047	0.923	0.843	0.041	2.290	0.843	0.051	1.625	**0.849**	0.037	4.253

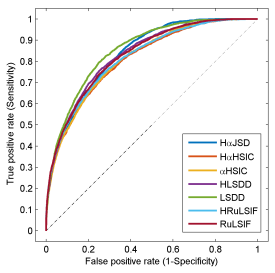
Table 4.3: Mean AUC values and standard deviation, and mean computational time of the methods that compute a multivariate score for data of size $1 \times 5,000$. The best methods in terms of mean AUC values are described in boldface.

The experimental results show that although all methods perform approximately the same, the running time differs from one method to the other. For example, while the RuLSIF method achieves the highest average AUC value, the proposed HαJSD method provides basically the same result (reduced by a value of only 0.007) with a significant run-time improvement (of a factor of about 25.47). It is interesting to note also that by exploiting the hierarchical idea, the αHSIC, LSDD and RuLSIF methods are sped up by a factor of 2.51, 2.48 and 2.62, while the AUC values are reduced only by a value of 0.002, 0 and 0.006, respectively. In addition, Figure 4.5 shows that there are no significant differences in performance of the examined methods for each specific dataset in terms of AUC values. Worth mentioning is also the fact that all methods, except from the LSDD, show the worst performance in the dataset with the shifting frequency and the best performance in the dataset with the random changes in the mean value, as depicted in Figures 4.5 (c)-(d).

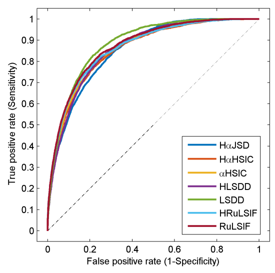
Investigation of the sample-size of the data

In order to investigate the sensitivity of the performance on different values of the sample size n in terms of AUC values and computational time we will test all methods using one-dimensional data of increasing sample size. Figures 4.6 (a)-(b) show the AUC values and the computational time of all examined methods averaged over all 7 datasets for sample size $n = 5,000$, $10,000$, and $50,000$. The detailed tables for all datasets and methods are provided in Appendix A.

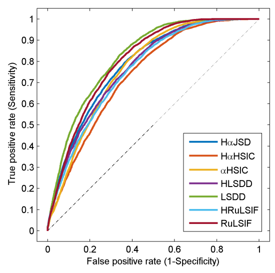
As can be seen, the running time is approximately linear to the number of samples n, while the performance remains at the same levels as sample size increases. In addition, the speedup of αHSIC, LSDD and RuLSIF methods seems to be unaffected by the number of samples, remaining at a factor of about 2.5 for all these methods, whereas the AUC values are maintained at the same levels as in conventional methods. It is noteworthy that the standard deviation of AUC for all methods becomes higher for larger sample sizes, with the RuLSIF method showing the highest increase and the HαJSD method showing the lowest increase.



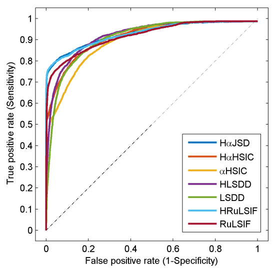
(a) Group 1: Jumping mean



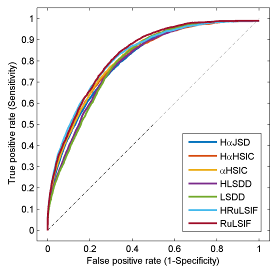
(b) Group 1: Scaling variance



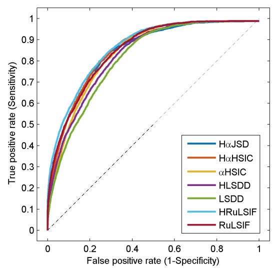
(c) Group 1: Changing frequency



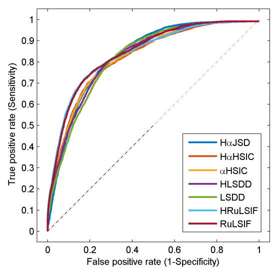
(d) Group 2: Shifting mean



(e) Group 2: Shifting variance



(f) Group 2: Shifting mean and variance



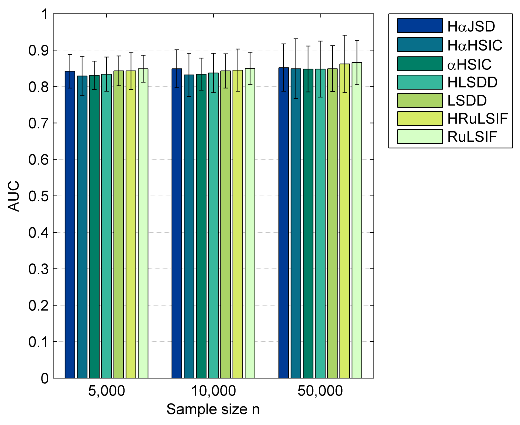
(g) Group 2: Changing distribution

Figure 4.5: Average ROC curves of HαJSD, HαHSIC, αHSIC, HLSDD, LSDD, HRuLSIF and RuLSIF methods for synthetic datasets of size $1 \times 5,000$.

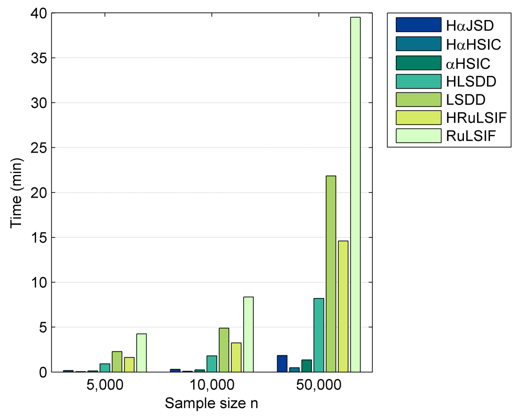
Investigation of dimensionality

In order to determine how the dimensionality of data affects the performance of algorithms we consider data of 5,000 samples along time with an increasing dimensionality. Figures 4.7 (a)-(b) display the AUC values and the computational time of all examined methods averaged over all 7 datasets for dimensionality d = 1, 25 and 50. The detailed tables for all datasets and methods are provided in Appendix A.

As can be noticed, although all methods except from LSDD and HLSDD, present higher AUC values as the dimensionality of data increases from 1 to 25, their performance is not further increased for higher-dimensional data. In particular, the methods that demonstrate the most significant improvement in performance with higher AUC values are the HαJSD and HαHSIC methods. On the contrary, the LSDD and HLSDD methods perform substan-



(a) Average AUC values and standard deviation



(b) Mean computational time (in minutes)

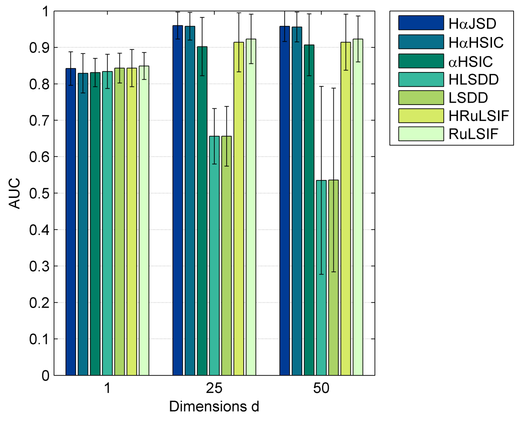
Figure 4.6: Figure (a) shows the average AUC values and standard deviation of all methods for sample size $n = 5{,}000, 10{,}000$ and $50{,}000$, and Figure (b) shows the corresponding computational times.

tially worse as the dimensionality of data increases. The opposite behavior is observed in the computational time, which grows dramatically for the HαJSD, HαHSIC and αHSIC methods that compute a weighted sum of univariate dissimilarity measures, whereas it remains constant for HLSDD, LSDD, HRuLSIF and RuLSIF methods that compute a multivariate dissimilarity measure. However, the relationship between the computational time of the HαJSD, HαHSIC and αHSIC methods and the data dimensionality does not appear to be linear, since the time tends to increase sharply as the dimensions are increased from 1 to 25 and less steeply thereafter. We should mention here that when the dimensionality is increased from 1 to 2 only HαJSD and HαHSIC methods present higher AUC values, while all others perform relatively the same. The results stated above (i.e., the better performance for all methods except from LSDD and HLSDD and the worse performance for LSDD and HLSDD) appear when the dimensions are increased to 4. As regards the computational time, it raises sharply for HαJSD, HαHSIC and αHSIC methods after the dimensions have been increased to 4, while it always remains stable for all other methods. Moreover, it is worth noting that although the speedup of the LSDD and RuLSIF methods is not affected by the number of dimensions, remaining at a factor of about 2.5, the αHSIC method is sped up by a factor of about 2.5, 6.5 and 4.8 for 1-, 25- and 50-dimensional data, respectively. In all cases, the AUC values are maintained at the same levels as in conventional methods. The last point to consider is that the HαHSIC method presents the lowest increase in the standard deviation of AUC, whereas the highest increase is shown in the LSDD method.

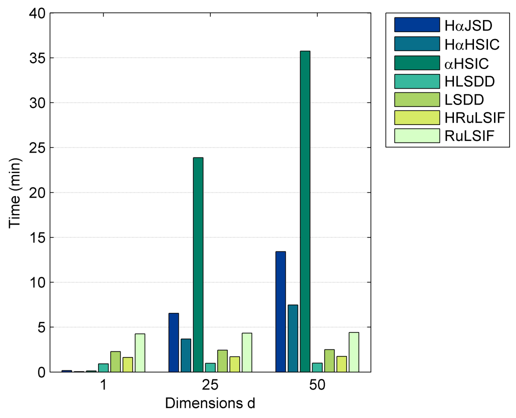
4.3. EXPERIMENTS ON REAL-WORLD DATASETS

In the previous section we compared the change-point detection methods using two artificial datasets with fixed and random change steps. In this section, we investigate how the methods perform in real-world datasets. All parameters are selected in the same manner as described above for the synthetic datasets.

The real-world dataset used here is the publicly available *PAMAP2* dataset, which is a



(a) Average AUC values and standard deviation



(b) Mean computational time (in minutes)

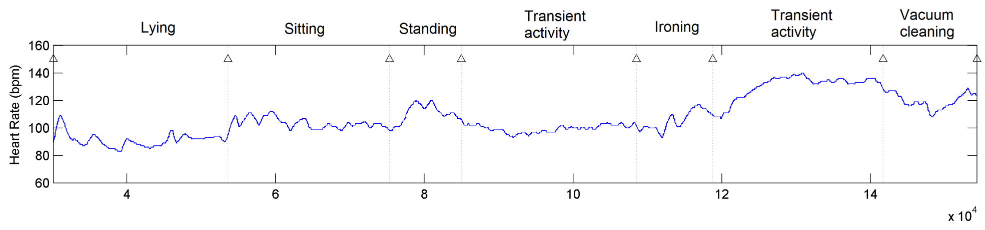
Figure 4.7: Figure (a) shows the average AUC values and standard deviation of all methods for dimensionality $d = 1, 25$ and 50, and Figure (b) shows the corresponding computational times.

subset of the project Physical Activity Monitoring for Aging People (PAMAP) [106]. This dataset provides human activity information recorded from 9 subjects, wearing 3 inertial measurement units (IMUs) and a heart rate (HR) monitor and performing a total of 18 different everyday, household and sport activities, such as walking, cycling and house cleaning. Each IMU contains two 3-axis MEMS accelerometers, a 3-axis MEMS gyroscope and a 3-axis magneto-inductive magnetic sensor, all sampled at 100 Hz. To obtain heart rate information, the BM-CS5SR HR monitor from BM innovations GmbH was used, providing heart rate values with approximately 9 Hz [107, 108].

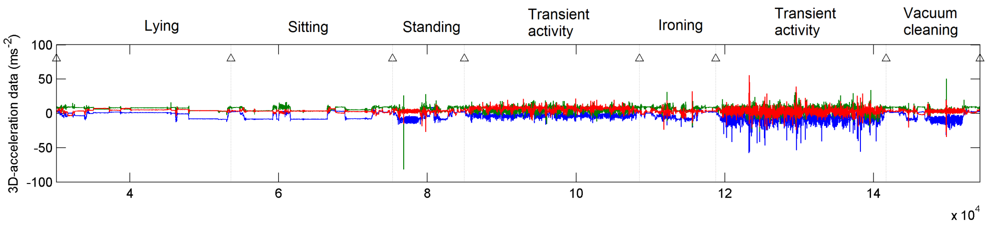
The sensors were placed onto 3 different body positions [108]. A chest sensor fixation includes one IMU and the heart rate chest strap. The second IMU is attached over the wrist on the dominant arm, and the third IMU on the dominant side's ankle. In addition, an inertial data collection unit was carried by the subjects in a pocket fixed on their belt, to collect the data from different sensors. Figure 4.9 illustrates the placement of all IMUs and the data collection unit. The duration of recorded activities for all Subjects except Subject 9 was about 1 hour, resulting in a total of 8 hours of data. More information about the data collection protocol, the subjects and the performed activities can be found in [107, 108].

Note that since this is a realistic dataset, there are many missing values in the data due to wireless data dropping or problems with the hardware setup. To deal with this issue and obtain valid time-series measurements, we reconstructed the missing values using cubic spline interpolation. Figure 4.8 provides an example of accelerometer, gyroscope, magnetometer and heart-rate data as they were recorded for one subject along with their annotations.

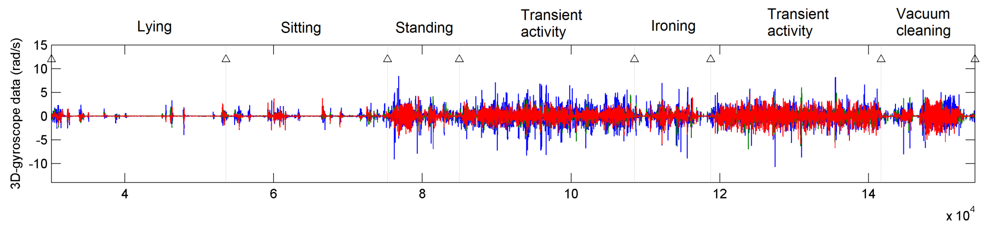
First, we will investigate how the methods work for each type of data separately. For this reason, we use all available data from Subject 2 (the subject with the longest time-series along time) and Subject 9 (the subject with the shortest time-series along time). Tables 4.4 - 4.5 for Subject 2, and 4.6 - 4.7 for Subject 9, describe the AUC values and the computational time when each of the 10 different data was used separately, as well as when all sensors were used together resulting in 28-dimensional data. As can be noticed from these tables, there



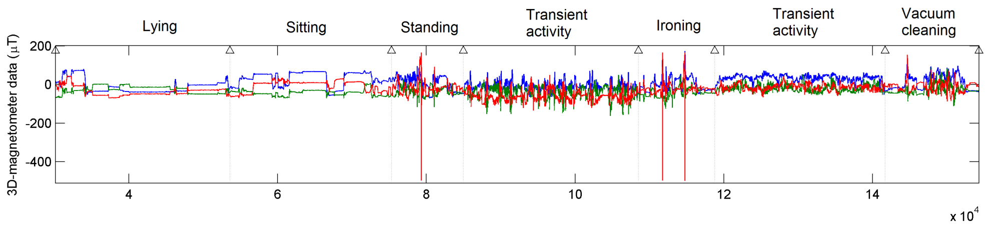
(a) Heart Rate



(b) Accelerometer



(c) Gyroscope



(d) Magnetometer

Figure 4.8: Raw-time series data recorded by Subject 1 in PAMAP2 dataset. The recorded data originate from the heart rate, accelerometer, gyroscope and magnetometer sensors and are provided along with their annotations.

are significant differences in performance between Subjects 2 and 9, and, hence, we will analyze the results separately.

For Subject 2, the HαJSD and HαHSIC methods exhibit their best performance in terms of AUC values when using accelerometer data, whereas for HLSDD, LSDD, HRuLSIF and RuLSIF methods the best performance is observed when using gyroscope data. On the other hand, all methods, except from HRuLSIF and RuLSIF, perform worst for the heart rate data, while the HRuLSIF and RuLSIF methods present their worst performance for accelerometer and magnetometer data obtained from the ankle IMU. The αHSIC method was tested only for a few data due to its excessive computational time. Overall, the HαJSD method shows the best performance in five out of eleven different cases (10 types of data and their combination), the LSDD method in three cases, the HαHSIC in two cases and the RuLSIF only in one case. As regards the computational time, it is worth observing that the HαJSD and HαHSIC methods, which compute a weighted sum of univariate dissimilarity

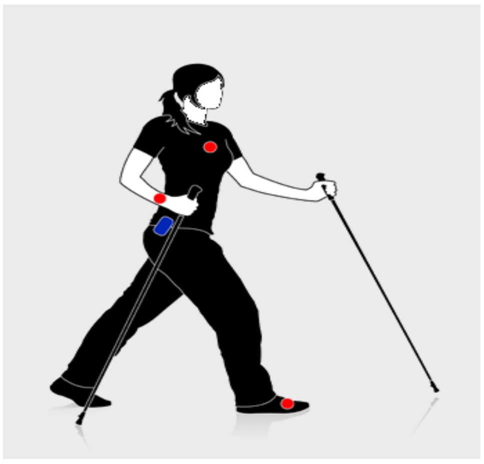


Figure 4.9: Placement of the three IMUs and the data collection unit [109].

measures, have relatively low running times for all data, except from the case where all data sensors are combined in one, whereas for HLSDD, LSDD, HRuLSIF and RuLSIF methods the running time is generally higher.

It is interesting to note also that, by exploiting the hierarchical idea, the LSDD and RuLSIF methods are sped up by a factor of about 4.2 and 4.3 on average, while the AUC values are reduced by a value of 0.029 and 0.017 on average, respectively. For both these methods the best speed-up is achieved with the one-dimensional heart rate data and the worst with the combination of all sensor data. Finally, the highest loss in accuracy is observed for heart rate data.

In Subject 9, which contains only a few samples (8,477 samples along the time vector, having a duration of about 1.5 min), the methods display different behavior in many cases. First of all, the HαJSD method presents its best performance in terms of AUC values for chest magnetometer data, the HαHSIC method for chest gyroscope data, the αHSIC method for chest accelerometer data, the LSDD method for hand magnetometer data, and the HLSDD, HRuLSIF and RuLSIF methods for data combined from all sensors. On the other hand, the worst performance in HαJSD, HαHSIC, HLSDD and LSDD methods is observed for heart rate data, and in αHSIC, HRuLSIF and RuLSIF methods for ankle magnetometer data. Overall, the HαJSD method shows the best performance in four out of eleven different cases, the LSDD in four cases and the HLSDD in three cases. The computational time among different methods follows the same behavior as in Subject 2, with the only difference that here all running times are kept at lower levels.

Furthermore, note that by exploiting the hierarchical idea, the LSDD and RuLSIF methods are sped up by a factor of about 5.37 and 4.86 on average, while the AUC values are reduced by a value of 0.011 and 0.028 on average, respectively. It is surprising to observe that the αHSIC method is sped up by a factor of 21.46 on average, while the AUC values are increased by a value of 0.049 on average. This may be explained by the fact that sometimes the First-Level algorithm rejects some false positives that could have strongly affected the method's performance. Moreover, the best speed-up for αHSIC method is achieved with the hand gyroscope data and for RuLSIF and LSDD methods with the one-dimensional heart rate data. On the contrary, the worse speed-up for αHSIC method is obtained with

Subject 2	HαJSD		HαHSIC		αHSIC	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
Hand Accelerometer	0.853	54.429	**0.855**	23.553	0.672	829.124
Hand Gyroscope	**0.826**	44.379	0.820	19.430	-	-
Hand Magnetometer	**0.877**	78.350	0.857	32.771	-	-
Ankle Accelerometer	**0.865**	64.686	0.713	28.051	-	-
Ankle Gyroscope	0.748	59.453	**0.750**	25.888	-	-
Ankle Magnetometer	**0.855**	86.383	0.749	36.695	-	-
Chest Accelerometer	0.930	51.565	**0.934**	22.619	-	-
Chest Gyroscope	0.808	51.140	**0.822**	22.515	-	-
Chest Magnetometer	**0.843**	89.880	0.788	39.699	-	-
Chest Heart Rate	0.537	6.074	0.542	2.112	**0.577**	10.952
All sensors	**0.844**	715.898	0.829	394.338	0.560	2662.543

Table 4.4: AUC values and computational time of all methods that use a data selection technique combined with a univariate score for Subject 2. The best methods in terms of AUC values are described in boldface. Runs of αHSIC method with excessive computational time were cancelled.

Subject 2	HLSDD		LSDD		HRuLSIF		RuLSIF	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
Hand Accelerometer	**0.766**	77.148	0.755	224.343	0.581	222.885	0.584	966.656
Hand Gyroscope	0.652	75.598	**0.717**	302.553	0.653	218.124	0.629	894.574
Hand Magnetometer	0.772	85.724	**0.794**	257.944	0.534	231.858	0.540	854.153
Ankle Accelerometer	0.829	80.492	**0.848**	242.080	0.331	225.435	0.356	817.667
Ankle Gyroscope	0.902	83.714	**0.912**	211.195	0.410	224.395	0.434	650.301
Ankle Magnetometer	0.800	93.742	**0.838**	221.281	0.331	231.904	0.330	611.790
Chest Accelerometer	0.840	80.666	**0.857**	221.112	0.515	226.018	0.553	684.007
Chest Gyroscope	0.809	80.606	**0.830**	218.673	0.642	220.781	0.654	630.526
Chest Magnetometer	0.840	97.220	**0.853**	216.226	0.400	238.993	0.418	635.427
Chest Heart Rate	0.384	19.772	0.510	381.586	0.584	55.384	**0.629**	878.493
All sensors	**0.581**	189.825	0.577	226.000	0.548	321.020	0.567	597.564

Table 4.5: AUC values and computational time of all methods that compute a multivariate score for Subject 2. The best methods in terms of AUC values are described in boldface.

heart rate data, for LSDD method with ankle gyroscope data and for RuLSIF method with chest gyroscope data. Finally, the LSDD method shows the worst loss in accuracy for heart rate data and the RuLSIF method for chest magnetometer data, while the αHSIC method shows the higher increase in accuracy for data combined from all sensors.

In order to investigate how the methods behave when combining data across different types of sensors, we use all available sensor data (i.e., accelerometer, gyroscope, magnetometer and heart rate from hand, chest and ankle IMUs) resulting in 28-dimensional data. Tables 4.8 - 4.9 describe the AUC values and the computational time for all methods and all subjects of PAMAP2 dataset. The AUC values and standard deviation of all methods averaged over all subjects as well as the running time for processing the whole dataset of 9 subjects are given in Table 4.10 and Figure 4.10.

As can be seen from Table 4.10 and Figure 4.10, the proposed HαJSD method demon-

Subject 9	HαJSD		HαHSIC		αHSIC	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
Hand Accelerometer	**0.958**	0.635	0.916	0.275	0.838	11.013
Hand Gyroscope	**0.870**	0.559	0.837	0.310	0.792	12.885
Hand Magnetometer	**0.996**	0.867	0.911	0.480	0.894	11.194
Ankle Accelerometer	**0.928**	0.923	0.792	0.507	0.769	12.115
Ankle Gyroscope	0.932	0.784	**0.940**	0.449	0.885	10.510
Ankle Magnetometer	**0.789**	1.003	0.685	0.488	0.613	9.195
Chest Accelerometer	**0.934**	0.835	0.926	0.473	0.902	9.341
Chest Gyroscope	0.942	0.759	**0.943**	0.430	0.878	10.125
Chest Magnetometer	**0.990**	1.267	0.911	0.681	0.883	8.720
Chest Heart Rate	**0.764**	0.100	0.683	0.045	0.681	0.163
All sensors	**0.962**	10.383	0.927	7.099	0.800	37.630

Table 4.6: AUC values and computational time of all methods that use a data selection technique combined with a univariate score for Subject 9. The best methods in terms of AUC values are described in boldface.

Subject 9	HLSDD		LSDD		HRuLSIF		RuLSIF	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
Hand Accelerometer	0.975	1.332	**0.982**	5.816	0.789	2.489	0.808	9.946
Hand Gyroscope	0.938	1.106	**0.961**	5.870	0.780	1.994	0.786	10.828
Hand Magnetometer	0.985	1.279	**0.991**	4.593	0.827	2.377	0.878	7.998
Ankle Accelerometer	**0.955**	1.279	0.951	4.523	0.613	2.368	0.613	7.920
Ankle Gyroscope	**0.956**	1.345	0.954	4.531	0.825	2.474	0.828	7.767
Ankle Magnetometer	0.912	1.281	**0.948**	4.454	0.545	2.370	0.574	7.547
Chest Accelerometer	**0.981**	1.349	0.980	4.798	0.888	2.478	0.890	7.708
Chest Gyroscope	0.963	1.289	**0.965**	5.171	0.867	2.516	0.868	7.665
Chest Magnetometer	0.888	1.393	**0.892**	6.833	0.749	2.540	0.873	11.230
Chest Heart Rate	0.561	0.412	0.614	7.735	0.692	0.782	**0.735**	12.829
All sensors	**0.990**	1.385	0.990	5.707	0.931	2.469	0.964	9.904

Table 4.7: AUC values and computational time of all methods that compute a multivariate score for Subject 9. The best methods in terms of AUC values are described in boldface.

strates the highest performance in terms of AUC values among all competitive methods. This result confirms the advantage of combining the First-Level algorithm with the data selection technique of the Second-Level algorithm when different types of data have to be processed at the same time. Note also that the HαJSD method presents the lowest AUC standard deviation value, indicating that its performance is nearly the same across all subjects. However, this method requires a relatively high computational time, which can be explained by the fact that by increasing the data dimensionality, the cost of computing the weights and the univariate scores for all dimensions is increased as well. On the other hand, the worst performance in terms of AUC values is observed in the αHSIC, HRuLSIF and RuLSIF methods. The αHSIC method, in addition, requires excessive amounts of running time to compute its change detection dissimilarity measure. It is worth noting also that by using the First-Level algorithm, the αHSIC method is sped up by a factor of 5.77, while its AUC value is increased by a value of 0.326, and the LSDD and RuLSIF methods are sped up by a factor of 1.24 and 1.90, while their AUC values are reduced by a value of 0.002 and 0.019,

Subject ID	HαJSD		HαHSIC		αHSIC	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	**0.778**	613.688	0.737	335.229	0.332	2111.330
2	**0.844**	715.898	0.829	394.338	0.560	2662.543
3	**0.889**	427.226	0.870	225.026	0.531	1552.933
4	**0.802**	547.395	0.760	294.222	0.309	1697.059
5	**0.780**	595.304	0.754	326.498	0.407	2127.035
6	**0.819**	765.688	0.796	577.874	0.515	2088.429
7	**0.882**	694.517	0.862	282.561	0.478	1679.935
8	**0.878**	682.899	0.850	368.433	0.522	2263.983
9	**0.962**	10.383	0.927	7.099	0.800	37.630

Table 4.8: AUC values and computational time of all methods that use a data selection technique combined with a univariate score for data obtained from all sensors of all 9 subjects. The best methods in terms of AUC values are described in boldface.

Subject ID	HLSDD		LSDD		HRuLSIF		RuLSIF	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	**0.857**	148.240	0.851	189.367	0.279	257.833	0.294	488.195
2	**0.581**	189.825	0.577	226.000	0.548	321.020	0.567	597.564
3	0.691	83.246	**0.723**	127.501	0.624	156.277	0.650	339.933
4	**0.788**	120.801	0.783	165.536	0.272	219.714	0.288	441.507
5	**0.820**	144.742	0.820	188.562	0.358	253.287	0.362	504.186
6	0.757	135.874	**0.759**	182.414	0.502	241.509	0.512	486.197
7	**0.792**	150.478	0.788	158.068	0.388	234.700	0.420	419.937
8	0.832	192.326	**0.836**	204.896	0.426	336.868	0.446	554.326
9	**0.990**	1.385	0.990	5.707	0.931	2.46	0.964	9.904

Table 4.9: AUC values and computational time of all methods that compute a multivariate score for data obtained from all sensors of all 9 subjects. The best methods in terms of AUC values are described in boldface.

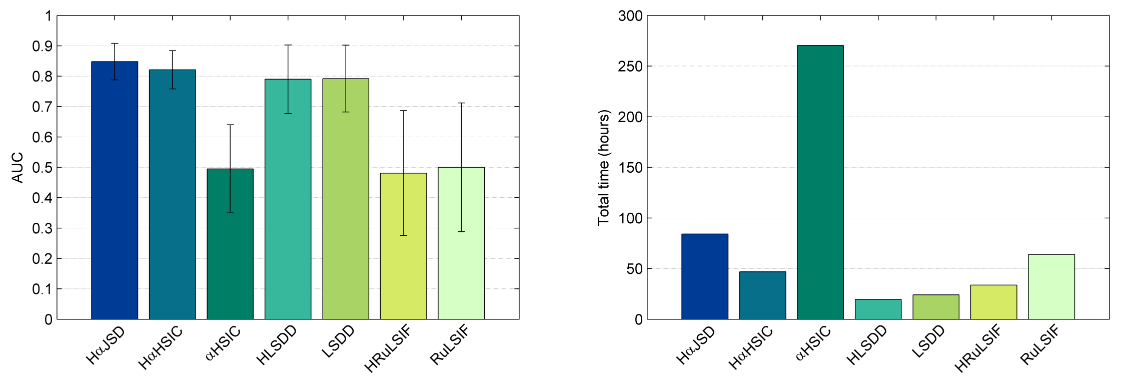
respectively.

In the last experimental set, only accelerometer data obtained from hand and chest IMUS will be used, because these are the most common mounting positions for wearable lifestyle sensors. Tables 4.11- 4.12 for hand data and 4.13 - 4.14 for chest data describe the AUC values and the computational time for all methods and all subjects of PAMAP2 dataset. The average AUC values and the total running time for processing the hand and chest accelerometer data of all 9 subjects are summarized in Table 4.15 and Figure 4.11. Note that in this experiment we use data as they were originally obtained from the PAMAP2 dataset, where we have replaced the missing values by zeros and not by valid measurements.

By comparing the results of Tables 4.11 - 4.15, we can see that there are no significant differences in performance of methods between hand and chest data. Overall, the best performance in terms of AUC values for both hand and chest data is achieved by the LSDD method, which, in addition, shows the lowest AUC standard deviation value. Nevertheless, the computational time of this method is quite high. It should be noted here, that the HαJSD and HαHSIC methods provide quite good results in substantially lower running

	AUC		Total time (hours)
	mean	std	
HαJSD	**0.848**	0.060	84.217
HαHSIC	0.821	0.063	46.855
αHSIC	0.495	0.145	270.348
HLSDD	0.790	0.113	19.449
LSDD	0.792	0.110	24.134
HRuLSIF	0.481	0.206	33.728
RuLSIF	0.500	0.212	64.029

Table 4.10: AUC values and standard deviation of all methods averaged over all subjects using data obtained from all sensors (28-dimensional data). The table also gives the running time of each method for processing the whole dataset of 9 subjects corresponding to a total of 8 hours of data. The best method in terms of AUC values is described in boldface.



(a) Average AUC values and standard deviation (b) Total computational time (in hours)

Figure 4.10: Figure (a) shows the average AUC values and standard deviation of all methods using a combination of all sensor data, and Figure (b) shows the corresponding running times for processing the whole dataset of 9 subjects comprising a total of 8 hours of data.

times. On the other hand, the worst performance in terms of AUC values is obtained by HRuLSIF method and in terms of computational time by αHSIC method. Finally, by using the First-Level algorithm, the αHSIC method is sped up by a factor of 27. 19, while its AUC value is increased by a value of 0.208 on average, and the LSDD and RuLSIF methods are sped up by a factor of 2.65 and 2.75, while their AUC values are reduced by a value of 0.005 and 0.014 on average, respectively.

Subject ID	HαJSD		HαHSIC		αHSIC	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	0.684	47.049	**0.738**	20.265	0.484	639.979
2	0.853	54.111	**0.853**	23.968	0.676	687.502
3	**0.872**	31.129	0.839	14.004	0.671	366.383
4	**0.829**	41.752	0.687	18.835	0.449	471.818
5	**0.768**	44.123	0.749	19.938	0.533	522.662
6	**0.852**	42.364	0.808	19.349	0.598	530.451
7	0.798	41.666	**0.811**	18.889	0.603	450.154
8	0.804	50.334	**0.892**	23.064	0.525	586.039
9	0.958	0.678	**0.916**	0.304	0.839	11.393

Table 4.11: AUC values and computational time of all methods that use a data selection technique for accelerometer data obtained from the hand IMU of all 9 subjects. The best methods in terms of AUC values are described in boldface.

Subject ID	HLSDD		LSDD		HRuLSIF		RuLSIF	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	**0.809**	65.277	0.807	173.436	0.409	195.769	0.428	532.009
2	**0.770**	79.501	0.754	214.387	0.586	228.770	0.589	613.598
3	**0.797**	43.573	0.795	117.926	0.625	128.570	0.638	360.465
4	0.871	57.465	**0.876**	149.957	0.357	168.902	0.361	462.391
5	**0.917**	66.088	0.917	165.820	0.439	190.196	0.449	518.692
6	**0.858**	63.834	0.854	162.247	0.567	186.649	0.580	484.556
7	**0.877**	55.399	0.876	139.101	0.542	157.805	0.539	551.024
8	0.860	72.544	**0.885**	181.540	0.450	208.643	0.452	552.578
9	0.975	1.464	**0.982**	4.309	0.789	4.245	0.808	12.610

Table 4.12: AUC values and computational time of all methods that compute a multivariate score for accelerometer data obtained from the hand IMU of all 9 subjects. The best methods in terms of AUC values are described in boldface.

Subject ID	HαJSD		HαHSIC		αHSIC	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	0.653	46.888	**0.718**	21.626	0.454	600.395
2	0.931	49.966	**0.934**	22.373	0.705	659.639
3	0.862	34.329	**0.887**	15.543	0.574	357.876
4	**0.760**	40.253	0.702	18.265	0.473	469.414
5	**0.765**	42.719	0.745	19.866	0.543	521.523
6	**0.811**	44.104	0.806	20.236	0.606	525.603
7	**0.813**	40.451	0.799	18.337	0.614	444.284
8	**0.827**	50.074	0.818	23.231	0.639	802.313
9	**0.934**	0.824	0.925	0.369	0.902	11.755

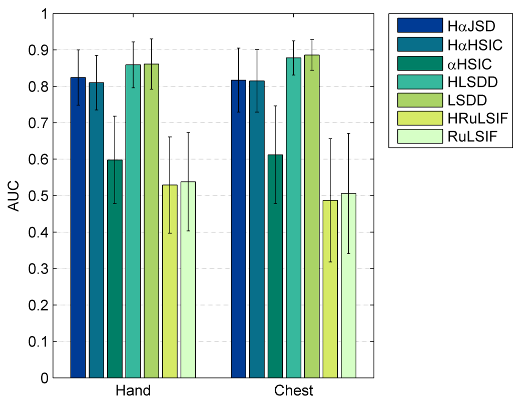
Table 4.13: AUC values and computational time of all methods that use a data selection technique combined with a univariate score for accelerometer data obtained from the chest IMU of all 9 subjects. The best methods in terms of AUC values are described in boldface.

Subject ID	HLSDD		LSDD		HRuLSIF		RuLSIF	
	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)	AUC	Time(min)
1	0.873	66.379	**0.878**	174.431	0.303	188.980	0.327	522.278
2	0.837	77.940	**0.857**	221.486	0.515	224.918	0.553	614.033
3	0.817	43.848	**0.830**	118.729	0.444	128.436	0.464	354.092
4	0.851	58.002	**0.869**	151.447	0.325	165.879	0.340	462.345
5	0.892	65.417	**0.905**	166.448	0.421	189.553	0.436	500.684
6	0.874	64.382	**0.877**	167.390	0.501	184.658	0.523	483.348
7	0.901	54.271	**0.906**	139.458	0.496	160.531	0.509	427.570
8	0.872	71.792	**0.873**	216.641	0.486	205.186	0.511	579.957
9	**0.981**	1.369	0.980	4.297	0.889	4.257	0.891	12.332

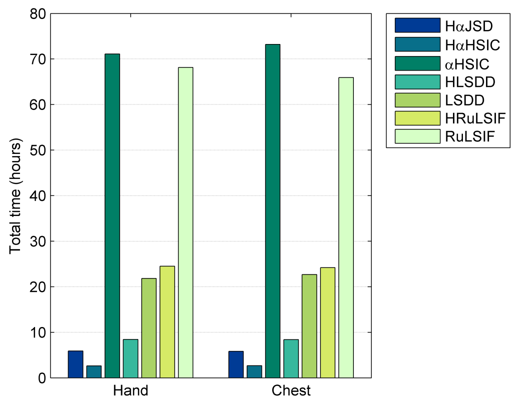
Table 4.14: AUC values and computational time of all methods that compute a multivariate score for accelerometer data obtained from the chest IMU of all 9 subjects. The best methods in terms of AUC values are described in boldface.

Methods	Hand Accelerometer			Chest accelerometer		
	AUC mean	AUC std	Total time (hours)	AUC mean	AUC std	Total time (hours)
HαJSD	0.824	0.076	5.887	0.817	0.088	5.827
HαHSIC	0.810	0.075	2.644	0.815	0.086	2.664
αHSIC	0.598	0.120	71.106	0.612	0.134	73.213
HLSDD	0.859	0.063	8.419	0.878	0.047	8.390
LSDD	**0.861**	0.069	21.812	**0.886**	0.042	22.672
HRuLSIF	0.529	0.132	24.493	0.487	0.169	24.207
RuLSIF	0.538	0.135	68.132	0.506	0.165	65.944

Table 4.15: AUC values and standard deviation of all methods averaged over all subjects using data from hand and chest IMUs. The table also gives the running time of each method for processing the whole dataset of 9 subjects corresponding to a total of 8 hours of data. The best methods in terms of AUC values are described in boldface.



(a) Average AUC values and standard deviation



(b) Total computational time (in hours)

Figure 4.11: Figure (a) shows the average AUC values and standard deviation of all methods using accelerometer data from the hand and chest IMUs, and Figure (b) shows the corresponding times for processing the whole dataset of 9 subjects comprising a total of 8 hours of data.

5 Conclusions and future work

In this work, we developed a non-parametric hierarchical algorithm for change-point detection, whose key idea is that if a simple test was used to quickly select some candidates in the First-Level, then the Second-Level, which in general requires higher computational time, would be applied only to a small subset of data, leading to a significant run-time improvement. In addition, in order to alleviate the difficulties arising in high-dimensional data, we used a data selection algorithm which has been designed to give more weight to the data that are important for identifying changes than to others. Using these ideas, we computed the final detection measure as the weighted sum of dissimilarity measures computed individually for each dimension of data. The hierarchical structure of the proposed algorithm has been further exploited to speed up some conventional change-point detection techniques that compute a univariate dissimilarity measure, and it has also been modified in such a way that it can be used in combination with multivariate dissimilarity measures. Experimental results on both artificial and real-world data demonstrate the usefulness of all developed methods.

To sum up the obtained results, we can say that the HαJSD method, proposed in this thesis, exhibits high performance in general for both artificial and real-world datasets and is particularly advantageous when different types of data have to be processed at the same time. The results of existing approaches revealed that the LSDD method, which performs very well in most cases for both artificial and real-world datasets, presents a high variability in performance in some cases, while the αHSIC and RuLSIF methods seem to perform better for artificial than for real-world data.

As regards the computational time, it is worth observing that HαJSD, αHSIC and its modification HαHSIC that compute a weighted sum of univariate dissimilarity measures, present low running times for small- and large-sample size datasets with low dimensionality, and high running times when the data dimensionality is increased. On the other hand, LSDD, RuLSIF and their modifications HLSDD and HRuLSIF, that compute a multivariate dissimilarity measure require high running times for both small- and large sample size datasets. However, these methods appear to be unaffected by an increase of the data dimensionality.

One of the main contributions of this work relies on the hierarchical structure of the proposed algorithms, which offers the great benefit of improving the computational time of traditional methods while maintaining their performance unaffected. For all examined methods the highest speedup is observed for low-dimensional and large sample size data, while the lowest speedup appears for high-dimensional data. For LSDD and RuLSIF methods in particular, the AUC values remain basically at the same levels (with only a slight decrease), whereas for αHSIC method the AUC values when using real-world data are always increased. This could be explained by the fact that in many cases the First-Level algorithm rejects some false positives that could have negatively affected the detection test.

Another interesting conclusion of this study is that a universal algorithm which results in an optimal solution for all cases does not always exist and, thus, if we have prior knowledge of the nature of the data, we can utilize it in order to select the most suitable algorithm. It should be noted also that even though we investigated a number of different scenarios with various sample sizes and dimensionalities, the huge variety of change-point detection applications does not allows us to derive general assertions. However, the obtained results provide a certain insight into the change detection problem and can serve as a tool to shape the scientists expectations of the behavior of examined methods.

We are convinced that with the development of wireless devices and miniature sensors that can continuously record data, the importance of fast change detection techniques will increase. Therefore, as a future work we can apply the proposed hierarchical algorithms to many other change-point detection techniques in order to improve their computational times. Another important future challenge comes from the use of multi-sensor data and indicates why data selection techniques, as the one used in the HαJSD method presented in this thesis, should be incorporated in the change detection procedure. We leave the application of such selection methods to change detection problems as future work. Moreover, investigating the performance of the examined methods over other real datasets, such as respiration and music datasets, is a possible extension of this work.

A

Appendix A - Performance Tables

	HαJSD			HαHSIC			αHSIC		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std	
Group 1									
Jumping mean	**0.861**	0.005	0.316	0.814	0.006	0.089	0.810	0.005	0.228
Scaling variance	0.872	0.010	0.317	0.873	0.007	0.105	**0.877**	0.007	0.279
Changing frequency	**0.783**	0.019	0.279	0.740	0.015	0.086	0.776	0.013	0.243
Group 2									
Shifting mean	**0.943**	0.013	0.257	0.926	0.012	0.089	0.904	0.012	0.238
Shifting variance	0.805	0.015	0.306	0.804	0.015	0.104	**0.813**	0.012	0.248
Shifting mean and variance	0.850	0.013	0.316	**0.854**	0.012	0.095	0.840	0.012	0.284
Changing distribution	**0.829**	0.015	0.311	0.816	0.018	0.087	0.817	0.015	0.245
Average	**0.849**	0.052	0.300	0.832	0.059	0.094	0.834	0.044	0.252

Table A.1: Mean AUC values and standard deviation, and mean computational time of the methods that use a data selection technique combined with a univariate score for data of size $1 \times 10,000$. The best methods in terms of mean AUC values are described in boldface.

	HLSDD			LSDD			HRuLSIF			RuLSIF		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std		mean	std	
Group 1												
Jumping mean	0.844	0.008	1.787	**0.850**	0.008	4.603	0.810	0.006	3.364	0.812	0.006	7.993
Scaling variance	0.879	0.010	1.864	**0.895**	0.008	4.434	0.885	0.008	3.043	0.888	0.008	8.015
Changing frequency	0.769	0.019	1.774	**0.825**	0.014	5.136	0.760	0.018	3.006	0.809	0.014	8.190
Group 2												
Shifting mean	0.930	0.010	1.693	0.917	0.010	5.125	**0.939**	0.012	3.070	0.929	0.013	8.426
Shifting variance	0.794	0.015	1.776	0.796	0.014	4.902	0.820	0.014	3.313	**0.825**	0.012	8.513
Shifting mean and variance	0.824	0.012	1.790	0.806	0.011	5.224	**0.865**	0.014	3.190	0.851	0.014	8.824
Changing distribution	0.817	0.014	1.984	0.812	0.012	4.764	0.835	0.017	3.729	**0.836**	0.015	8.600
Average	0.837	0.054	1.810	0.843	0.047	4.884	0.845	0.058	3.245	**0.850**	0.044	8.366

Table A.2: Mean AUC values and standard deviation, and mean computational time of the methods that compute a multivariate score for data of size $1 \times 10,000$. The best methods in terms of mean AUC values are described in boldface.

	HαJSD			HαHSIC			αHSIC		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std	
Group 1									
Jumping mean	0.920	0.002	1.862	**0.924**	0.001	0.408	0.911	0.001	1.504
Scaling variance	0.858	0.006	1.754	0.913	0.003	0.464	**0.914**	0.003	1.418
Changing frequency	**0.749**	0.007	1.895	0.704	0.005	0.498	0.747	0.006	1.330
Group 2									
Shifting mean	**0.944**	0.006	1.877	0.927	0.006	0.524	0.903	0.008	1.383
Shifting variance	**0.815**	0.006	1.780	0.808	0.003	0.499	0.814	0.003	1.278
Shifting mean and variance	0.852	0.004	1.833	**0.856**	0.005	0.471	0.838	0.005	1.263
Changing distribution	**0.829**	0.005	1.909	0.813	0.008	0.482	0.812	0.007	1.261
Average	**0.852**	0.065	1.844	0.849	0.082	0.478	0.848	0.063	1.348

Table A.3: Mean AUC values and standard deviation, and mean computational time of the methods that use a data selection technique combined with a univariate score for data of size $1 \times 50,000$. The best methods in terms of mean AUC values are described in boldface.

	HLSDD			LSDD			HRuLSIF			RuLSIF		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std		mean	std	
Group 1												
Jumping mean	**0.930**	0.002	8.489	0.914	0.002	21.052	0.929	0.001	14.700	0.924	0.001	40.513
Scaling variance	0.910	0.004	8.184	0.916	0.002	21.700	**0.925**	0.003	14.020	0.924	0.002	39.440
Changing frequency	0.730	0.008	8.471	**0.787**	0.008	22.641	0.720	0.006	14.726	0.772	0.006	38.923
Group 2												
Shifting mean	0.932	0.005	7.966	0.918	0.006	21.993	**0.941**	0.006	14.975	0.930	0.008	39.772
Shifting variance	0.796	0.009	8.119	0.799	0.004	21.825	0.821	0.008	14.702	**0.826**	0.006	39.308
Shifting mean and variance	0.825	0.003	8.020	0.803	0.004	21.732	**0.867**	0.006	14.676	0.850	0.006	39.332
Changing distribution	0.813	0.005	8.197	0.809	0.006	21.937	0.834	0.008	14.398	**0.835**	0.006	39.283
Average	0.848	0.077	8.207	0.849	0.063	21.840	0.862	0.079	14.600	0.866	0.061	39.510

Table A.4: Mean AUC values and standard deviation, and mean computational time of the methods that compute a multivariate score for data of size $1 \times 50,000$. The best methods in terms of mean AUC values are described in boldface.

	HαJSD			HαHSIC			αHSIC		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std	
Group 1									
Jumping mean	0.896	0.007	6.848	**0.901**	0.007	3.730	0.799	0.006	23.816
Scaling variance	**0.975**	0.004	6.811	0.960	0.005	4.036	0.876	0.008	24.341
Changing frequency	**0.917**	0.008	6.608	0.907	0.012	3.588	0.794	0.020	23.412
Group 2									
Shifting mean	0.991	0.011	6.905	0.991	0.011	3.647	0.991	0.008	24.682
Shifting variance	0.979	0.010	6.208	**0.980**	0.010	3.624	0.952	0.013	24.269
Shifting mean and variance	**0.985**	0.011	6.263	0.983	0.011	3.563	0.956	0.009	22.921
Changing distribution	0.975	0.011	6.227	**0.981**	0.011	3.587	0.943	0.011	23.778
Average	**0.960**	0.037	6.553	0.958	0.038	3.682	0.902	0.080	23.888

Table A.5: Mean AUC values and standard deviation, and mean computational time of the methods that use a data selection technique combined with a univariate score for data of size $25 \times 5,000$. The best methods in terms of mean AUC values are described in boldface.

	HLSDD			LSDD			HRuLSIF			RuLSIF		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std		mean	std	
Group 1												
Jumping mean	0.787	0.015	0.936	0.784	0.019	2.504	0.878	0.006	1.719	**0.881**	0.007	4.355
Scaling variance	0.677	0.016	0.916	0.672	0.017	2.530	**0.898**	0.004	1.792	0.896	0.004	4.397
Changing frequency	0.546	0.004	1.042	0.537	0.003	2.406	0.761	0.023	1.781	**0.806**	0.019	4.280
Group 2												
Shifting mean	0.677	0.053	1.000	0.723	0.052	2.478	0.985	0.011	1.499	**0.989**	0.011	4.285
Shifting variance	0.609	0.033	1.026	0.607	0.034	2.307	0.911	0.015	1.775	**0.921**	0.014	4.277
Shifting mean and variance	0.682	0.068	0.972	0.665	0.070	2.328	0.983	0.011	1.788	**0.984**	0.011	4.481
Changing distribution	0.615	0.031	1.011	0.605	0.029	2.470	0.982	0.011	1.576	**0.983**	0.011	4.255
Average	0.656	0.076	0.986	0.656	0.082	2.432	0.914	0.081	1.704	**0.923**	0.068	4.333

Table A.6: Mean AUC values and standard deviation, and mean computational time of the methods that compute a multivariate score for data of size $25 \times 5,000$. The best methods in terms of mean AUC values are described in boldface.

	HαJSD			HαHSIC			αHSIC		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std	
Group 1									
Jumping mean	0.885	0.006	13.584	**0.889**	0.005	7.748	0.797	0.006	36.419
Scaling variance	**0.977**	0.005	13.462	0.957	0.005	7.516	0.877	0.005	36.191
Changing frequency	**0.910**	0.009	13.352	0.909	0.009	7.375	0.790	0.020	35.310
Group 2									
Shifting mean	0.986	0.013	13.220	**0.986**	0.013	7.212	0.983	0.013	35.334
Shifting variance	0.983	0.012	13.447	**0.984**	0.011	7.119	0.967	0.011	35.468
Shifting mean and variance	**0.984**	0.014	13.131	0.983	0.013	7.569	0.969	0.014	35.234
Changing distribution	0.982	0.010	13.703	**0.986**	0.010	7.802	0.964	0.012	36.297
Average	**0.958**	0.042	13.414	0.956	0.041	7.477	0.907	0.085	35.750

Table A.7: Mean AUC values and standard deviation, and mean computational time of the methods that use a data selection technique combined with a univariate score for data of size $50 \times 5,000$. The best methods in terms of mean AUC values are described in boldface.

	HLSDD			LSDD			HRuLSIF			RuLSIF		
	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)	AUC		Time(min)
	mean	std		mean	std		mean	std		mean	std	
Group 1												
Jumping mean	0.742	0.012	1.073	0.748	0.014	2.326	**0.875**	0.003	1.612	0.874	0.005	4.075
Scaling variance	0.698	0.009	1.041	0.694	0.011	2.375	**0.928**	0.003	1.833	0.922	0.003	4.025
Changing frequency	0.111	0.011	0.999	0.117	0.012	3.272	0.774	0.016	1.772	**0.824**	0.014	4.047
Group 2												
Shifting mean	0.366	0.037	0.955	0.370	0.038	2.538	0.977	0.013	1.658	**0.983**	0.013	4.662
Shifting variance	0.343	0.078	1.017	0.359	0.060	2.270	0.878	0.020	1.785	**0.891**	0.018	3.899
Shifting mean and variance	0.720	0.022	0.962	0.711	0.019	2.154	**0.980**	0.014	1.908	0.979	0.013	4.993
Changing distribution	0.762	0.016	0.970	0.753	0.018	2.571	0.984	0.010	1.622	**0.985**	0.010	5.127
Average	0.535	0.258	1.002	0.536	0.252	2.501	0.914	0.077	1.741	**0.923**	0.063	4.404

Table A.8: Mean AUC values and standard deviation, and mean computational time of the methods that compute a multivariate score for data of size $50 \times 5,000$. The best methods in terms of mean AUC values are described in boldface.

BIBLIOGRAPHY

[1] John F. Roddick, Lina Al-Jadir, Leopoldo Bertossi, Marlon Dumas, Florida Estrella, Heidi Gregersen, Kathleen Hornsby, Jens Lufter, Federica Mandreoli, Tomi Männistö, Enric Mayol, and Lex Wedemeijer, "Evolution and Change in Data Management - Issues and Directions," *SIGMOD Rec.*, vol. 29, pp. 21–25, Mar. 2000.

[2] Gordon J. Ross, Dimitris K. Tasoulis, and Niall M. Adams, "Online Annotation and Prediction for Regime Switching Data Streams," in *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, (New York, NY, USA), pp. 1501–1505, ACM, 2009.

[3] Yoshinobu Kawahara and Masashi Sugiyama, "Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation," in *Proceedings of the SIAM International Conference on Data Mining, SDM 2009*, (Nevada, USA), pp. 389–400, 2009.

[4] Miodrag Lovrić, Marina Milanović, and Milan Stamenković, "Algorithmic Methods For Segmentation of Time Series: An overview," *Journal of Contemporary Economic and Business Issues (JCEBI)*, vol. 1, no. 1, pp. 31–53, 2014.

[5] Idris A. Eckley, Paul Fearnhead, and Rebecca Killick, "Analysis of changepoint models," in *Bayesian Time Series Models* (David Barber, A. Taylan Cemgil, and Silvia Chiappa, eds.), pp. 205–224, Cambridge University Press, 2011.

[6] Phipps Arabie, J. Douglas Carroll, Wayne DeSarbo, and Jerry Wind, "Overlapping Clustering: A New Method for Product Positioning," *Journal of Marketing Research*, vol. 18, no. 3, pp. 310–317, 1981.

[7] Daniel Barry and J. A. Hartigan, "Product Partition Models for Change Point Problems," *The Annals of Statistics*, vol. 20, pp. 260–279, Mar. 1992.

[8] Md. Mijanur Rahman and Md. Al-Amin Bhuiyan, "Dynamic Thresholding on Speech Segmentation," *International Journal of Research in Engineering and Technology*, vol. 02, pp. 404–411, Sept. 2013.

[9] Babak Azimi-Sadjadi and P. S. Krishnaprasad, "A Particle Filtering Approach to Change Detection for Nonlinear Systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 15, pp. 2295–2305, 2004.

[10] O. Seidou and T. B. M. J. Ouarda, "Recursion-based multiple changepoint detection in multiple linear regression and application to river streamflows," *Water Resources Research*, vol. 43, July 2007.

[11] Kumar Vasimalla, "A Survey on Time Series Data Mining," *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, vol. 2, pp. 170–179, Oct. 2014.

[12] Timothy Patterson, Sally McClean, Chris Nugent, Shuai Zhang, Leo Galway, and Ian Cleland, "Online Change Detection for Timely Solicitation of User Interaction," in *Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services* (Ramón Hervás, Sungyoung Lee, Chris Nugent, and José Bravo, eds.), no. 8867 in Lecture Notes in Computer Science, pp. 116–123, Springer International Publishing, Dec. 2014.

[13] Hesam Komari Alaei, Seyed Iman Pishbin, and Karim Salahshoor, "A New PCA Cluster-Based Granulated Algorithm Using Rough Set Theory for Process Monitoring," *International Journal of Database Theory and Application*, vol. 4, no. 4, pp. 1–12, 2011.

[14] Frédéric Desobry, Manuel Davy, and Christian Doncarli, "An Online Kernel Change Detection Algorithm," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2961–2974, Aug. 2005.

[15] David S. Matteson and Nicholas A. James, "A Nonparametric Approach for Multiple Change Point Analysis of Multivariate Data," *Journal of the American Statistical Association*, vol. 109, pp. 334–345, June 2013. arXiv: 1306.4933.

[16] Long Yu and Zhongqing Su, "Application of Kernel Density Estimation in Lamb Wave-Based Damage Detection," *Mathematical Problems in Engineering*, vol. 2012, Aug. 2012.

[17] Michèle Basseville and Igor V. Nikiforov, *Detection of Abrupt Changes: Theory and Application.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[18] Walter A. Shewhart, *Economic control of quality of manufactured product.* New York: D. Van Nostrand Company, Inc., 1931.

[19] Nancy R. Zhang and David O. Siegmund, "A Modified Bayes Information Criterion with Applications to the Analysis of Comparative Genomic Hybridization Data," *Biometrics*, vol. 63, pp. 22–32, Mar. 2007.

[20] Weil R. Lai, Mark D. Johnson, Raju Kucherlapati, and Peter J. Park, "Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data," *Bioinformatics (Oxford, England)*, vol. 21, p. 3763, Oct. 2005.

[21] M. F. Mohamed Saaid, W. A. B. Wan Abas, H. Aroff, N. Mokhtar, R. Ramli, and Z. Ibrahim, "Change Point Detection of EEG Signals Based on Particle Swarm Optimization," in *5th Kuala Lumpur International Conference on Biomedical Engineering 2011* (Noor Azuan Abu Osman, Wan Abu Bakar Wan Abas, Ahmad Khairi Abdul Wahab, and Hua-Nong Ting, eds.), no. 35 in IFMBE Proceedings, pp. 484–487, Springer Berlin Heidelberg, 2011.

[22] 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 *Proceedings of the 23th International Conference on Architecture of Computing Systens, ARCS 2010*, (Hannover, Germany), pp. 167–176, VDE Verlag, 2010.

[23] Sana Tmar-Ben Hamida, Elyes Ben Hamida, and Beena Ahmed, "A New mHealth Communication Framework for Use in Wearable WBANs and Mobile Technologies," *Sensors*, vol. 15, pp. 3379–3408, Feb. 2015.

[24] Henri Caussinus and Olivier Mestre, "Detection and correction of artificial shifts in climate series," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 53, pp. 405–425, Aug. 2004.

[25] Daniela Jarušková, "Change-Point Detection in Meteorological Measurement," *Monthly Weather Review*, vol. 124, pp. 1535–1543, July 1996.

[26] David Allen, Michael McAleer, Robert Powell, and Abhay Singh, "Nonparametric Multiple Change Point Analysis of the Global Financial Crisis," KIER Working Paper 866, Kyoto University, Institute of Economic Research, May 2013.

[27] R. Tahmasbi and S. Rezaei, "Change Point Detection in GARCH Models for Voice Activity Detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 1038–1046, July 2008.

[28] Ryan Turner, "Bayesian change point detection for satellite fault prediction," in *Proceedings of Interdisciplinary Graduate Conference (IGC)*, (Cambridge, UK), pp. 213–221, 2010.

[29] David C. Carslaw, Karl Ropkins, and Margaret C. Bell, "Change-point detection of gaseous and particulate traffic-related pollutants at a roadside location," *Environmental Science & Technology*, vol. 40, pp. 6912–6918, Nov. 2006.

[30] Luis J. Manso, Pedro Nunez, Sidnei da, and Paulo Drews-Jr, "A Novel Robust Scene Change Detection Algorithm for Autonomous Robots Using Mixtures of Gaussians," *International Journal of Advanced Robotic Systems*, vol. 11, no. 18, p. 1, 2014.

[31] Amadou Ba and Sean A. McKenna, "Water quality monitoring with online change-point detection methods," *Journal of Hydroinformatics*, vol. 17, p. 7, Jan. 2015.

[32] Javier Ortiz Laguna, Angel García Olaya, and Daniel Borrajo, "A Dynamic Sliding Window Approach for Activity Recognition," in *User Modeling, Adaption and Personalization* (J. A. Konstan, Ricardo Conejo, José L. Marzo, and Nuria Oliver, eds.), no. 6787 in Lecture Notes in Computer Science, pp. 219–230, Springer Berlin Heidelberg, 2011.

[33] "ASSIST - Heterogeneous Integration - Research Expertise Nanofab Labs and Cleanroom at Penn State." `http://www.mri.psu.edu/facilities/nanofab/research-areas/assist.asp`.

[34] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi, "Change (Detection) You Can Believe in: Finding Distributional Shifts in Data Streams," in *Advances in Intelligent Data Analysis VIII* (N. M. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, eds.), no. 5772 in Lecture Notes in Computer Science, pp. 21–34, Springer Berlin Heidelberg, 2009.

[35] Luis Martí, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia, "Anomaly Detection Based on Sensor Data in Petroleum Industry Applications," *Sensors*, vol. 15, pp. 2774–2797, Jan. 2015.

[36] Ella Bingham, "Finding Segmentations of Sequences," in *Inductive Databases and Constraint-Based Data Mining* (Sašo Džeroski, Bart Goethals, and Panče Panov, eds.), pp. 177–197, Springer New York, 2010.

[37] Parvathi Chundi and Daniel J. Rosenkrantz, "Segmentation of Time Series Data," in *Encyclopedia of Data Warehousing and Mining, Second Edition* (John Wang, ed.), pp. 1753–1758, IGI Global, 2009.

[38] Bingwen Zhang, Jun Geng, and Lifeng Lai, "Multiple Change-Points Estimation in Linear Regression Models via Sparse Group Lasso," *IEEE Transactions on Signal Processing*, vol. 63, pp. 2209–2224, May 2015.

[39] Richard Bellman, "On the Approximation of Curves by Line Segments Using Dynamic Programming," *Commun. ACM*, vol. 4, p. 284, June 1961.

[40] Aristides Gionis and Heikki Mannila, "Segmentation algorithms for time series and sequence data." Tutorial at 5th SIAM International Conference on Data Mining, 2005.

[41] Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick, "Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2232–2245, Dec. 2010.

[42] Andre Gensler, Thiemo Gruber, and Bernhard Sick, "Blazing Fast Time Series Segmentation Based on Update Techniques for Polynomial Approximations," in *2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW)*, pp. 1002–1011, Dec. 2013.

[43] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani, "Segmenting Time Series: A Survey and Novel Approach," in *Data Mining In Time Series Databases* (Mark Last, Abraham Kandel, and Horst Bunke, eds.), vol. 57, pp. 1–22, World Scientific Publishing Company, 2004.

[44] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani, "An Online Algorithm for Segmenting Time Series," in *Proceedings of IEEE International Conference on Data Mining (ICDM) 2001*, (San Jose, CA), pp. 289–296, 2001.

[45] R. H. Loschi and F. R. B. Cruz, "Extension to the product partition model: computing the probability of a change," *Computational Statistics & Data Analysis*, vol. 48, no. 2, pp. 255–268, 2005.

[46] Jacqueline A. Ferreira, Rosangela H. Loschi, and Marcelo A. Costa, "Detecting changes in time series: A product partition model with across-cluster correlation," *Signal Processing*, vol. 96, Part B, pp. 212–227, Mar. 2014.

[47] Jonathan J. Oliver, Rohan A. Baxter, and Chris S. Wallace, "Minimum Message Length Segmentation," in *Research and Development in Knowledge Discovery and Data Mining* (Xindong Wu, Ramamohanarao Kotagiri, and Kevin B. Korb, eds.), vol. 1394 of *Lecture Notes in Computer Science*, pp. 222–233, Springer Berlin Heidelberg, 1998.

[48] Leigh J. Fitzgibbon, David L. Dowe, and Lloyd Allison, "Change-Point Estimation Using New Minimum Message Length Approximations," in *PRICAI 2002: Trends in Artificial Intelligence* (Mitsuru Ishizuka and Abdul Sattar, eds.), vol. 2417 of *Lecture Notes in Computer Science*, pp. 244–254, Springer Berlin Heidelberg, 2002.

[49] Taketoshi Mori, Yu Nejigane, Masamichi Shimosaka, Yushi Segawa, Tatsuya Harada, and Tomomasa Sato, "Online Recognition and Segmentation for Time-Series Motion with HMM and Conceptual Relation of Actions," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005)*, pp. 3864–3870, Aug. 2005.

[50] Yevgen Gorshkov, Illya Kokshenev, Yevgeniy Bodyanskiy, Vitaliy Kolodyazhniy, and Oleksandr Shylo, "Robust Recursive Fuzzy Clustering-Based Segmentation of Biological Time Series," in *2nd International Symposium on Evolving Fuzzy Systems*, (Ambleside), pp. 101–105, Sept. 2006.

[51] Yadunandana N. Rao and Jose C. Principe, "A Fast On-line Generalized Eigendecomposition Algorithm for Time Series Segmentation," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC) 2000*, (Lake Louise, Alta.), pp. 266–271, 2000.

[52] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama, "Change-point Detection in Time-series Data by Relative Density-Ratio Estimation," *Neural Networks*, vol. 43, pp. 72–83, July 2013.

[53] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P. Xing, and Masashi Sugiyama, "High-Dimensional Feature Selection by Feature-Wise Kernelized Lasso," *Neural Computation*, vol. 26, pp. 185–207, Jan. 2014.

[54] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama, "Relative Density-Ratio Estimation for Robust Distribution Comparison," *Neural Computation*, vol. 25, no. 5, pp. 1324–1370, 2013.

[55] Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Marthinus Christoffel du Plessis, Song Liu, and Ichiro Takeuchi, "Density-Difference Estimation," *Neural Computation*, vol. 25, pp. 2734–2775, June 2013.

[56] Yoshinobu Kawahara and Masashi Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining*, vol. 5, pp. 114–127, Apr. 2012.

[57] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Bünau, and Motoaki Kawanabe, "Direct Importance Estimation with Model Selection And its Application to Covariate Shift Adaptation," in *Advances in Neural Information Processing Systems (NIPS) 2008*, 2008.

[58] Magnus S. Magnusson, "Discovering hidden time patterns in behavior: T-patterns and their detection," *Behavior Research Methods, Instruments, & Computers*, vol. 32, pp. 93–110, Mar. 2000.

[59] Fu-lai Chung, Tak-chung Fu, Robert Luk, and Vincent Ng, "Evolutionary Time Series Segmentation for Stock Data Mining," in *IEEE International Conference on Data Mining (ICDM)*, pp. 83–90, 2002.

[60] Jiangling Yin, Yain-Whar Si, and Zhiguo Gong, "Financial Time Series Segmentation Based On Turning Points," in *2011 International Conference on System Science and Engineering (ICSSE)*, (Macau, China), pp. 394–399, June 2011.

[61] Allou Samé and Gérard Govaert, "Online Time Series Segmentation Using Temporal Mixture Models and Bayesian Model Selection," in *11th International Conference on Machine Learning and Applications (ICMLA)*, vol. 1, (Boca Raton, FL), pp. 602–605, Dec. 2012.

[62] Hang Yu, Chenyang Li, and Justin Dauwels, "Network Inference and Change Point Detection for Piecewise-Stationary Time Series," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Florence, Italy), pp. 4498–4502, May 2014.

[63] Fredrik Gustafsson, "The Marginalized Likelihood Ratio Test for Detecting Abrupt Changes," *IEEE Transactions on Automatic Control*, vol. 41, pp. 66–78, Jan. 1996.

[64] Kenji Yamanishi, Jun-ichi Takeuchi, Graham Williams, and Peter Milne, "On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms," *Data Mining and Knowledge Discovery*, vol. 8, pp. 275–300, May 2004.

[65] Makoto Yamada, Akisato Kimura, Futoshi Naya, and Hiroshi Sawada, "Change-point Detection with Feature Selection in High-dimensional Time-series Data," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, (Beijing, China), pp. 1827–1833, AAAI Press, 2013.

[66] M. Seck, I. Magrin-Chagnolleau, and F. Bimbot, "Experiments on speech tracking in audio documents using Gaussian mixture modeling," in *IEEE International Conference on Acoustics, Speech, and Signal Processing 2001 (ICASSP '01)*, vol. 1, (Salt Lake City, UT), pp. 601–604, IEEE, 2001.

[67] Nick Whiteley, Christophe Andrieu, and Arnaud Doucet, "Particle Markov Chain Monte Carlo for Multiple Change-point Problems," Technical Report 0911, Department of Mathematics, Bristol University, 2009.

[68] Valentina Moskvina and Anatoly Zhigljavsky, "An Algorithm Based on Singular Spectrum Analysis for Change-Point Detection," *Communications in Statistics - Simulation and Computation*, vol. 32, pp. 319–352, Jan. 2003.

[69] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida, “Change-Point Detection in Time-Series Data Based on Subspace Identification,” in *7th IEEE International Conference on Data Mining (ICDM 2007)*, (Omaha, NE), pp. 559–564, IEEE, Oct. 2007.

[70] Tsuyoshi Idé and Koji Tsuda, “Change-Point Detection using Krylov Subspace Learning,” in *Proceedings of the SIAM International Conference on Data Mining (SDM 2007)* (C. Apte, ed.), (Minneapolis, MN, USA), pp. 515–520, Society for Industrial and Applied Mathematics, Apr. 2007.

[71] Joep Vanlier, Christian A. Tiemann, Peter AJ Hilbers, and Natal AW van Riel, “Optimal experiment design for model selection in biochemical networks,” *BMC Systems Biology*, vol. 8, p. 20, Feb. 2014.

[72] Marcin Budka, Bogdan Gabrys, and Katarzyna Musial, “On Accuracy of PDF Divergence Estimators and Their Applicability to Representative Data Sampling,” *Entropy*, vol. 13, pp. 1229–1266, July 2011.

[73] Sylvain Boltz, Eric Debreuve, and Michel Barlaud, “High-dimensional statistical distance for region-of-interest tracking: Application to combining a soft geometric constraint with radiometry,” in *IEEE Conference on Computer Vision and Pattern Recognition 2007 (CVPR ’07)*, (Minneapolis, MN), pp. 1–8, IEEE, June 2007.

[74] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori, *Density Ratio Estimation in Machine Learning*. Cambridge: Cambridge University Press, 2012.

[75] Vladimir N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 1 edition ed., Sept. 1998.

[76] Jiayuan Huang, Arthur Gretton, Karsten M. Borgwardt, Bernhard Schölkopf, and Alex J. Smola, “Correcting sample selection bias by unlabeled data,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 601–608, 2006.

[77] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan, “Estimating divergence functionals and the likelihood ratio by convex risk minimization,” *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5847–5861, 2010. arXiv: 0809.0853.

[78] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama, “A Least-squares Approach to Direct Importance Estimation,” *Journal of Machine Learning Research*, vol. 10, pp. 1391–1445, Dec. 2009.

[79] Richard Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, New Jersey: Princeton University Press, 1961.

[80] Imre Csiszár, “Information-type measures of difference of probability distributions and indirect observations,” *Studia Scientiarum Mathematicarum Hungarica*, vol. 2, pp. 299–318, 1967.

[81] Syed Mustafa Ali and Simon Daniel Silvey, “A General Class of Coefficients of Divergence of One Distribution from Another,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 28, no. 1, pp. 131–142, 1966.

[82] Nicolas Veyrat-Charvillon and François-Xavier Standaert, "Mutual Information Analysis: How, When and Why?," in *Cryptographic Hardware and Embedded Systems - CHES 2009* (Christophe Clavier and Kris Gaj, eds.), no. 5747 in Lecture Notes in Computer Science, pp. 429–443, Springer Berlin Heidelberg, 2009.

[83] Jianqing Fan, Fang Han, and Han Liu, "Challenges of Big Data Analysis," *National Science Review*, vol. 1, pp. 293–314, June 2014.

[84] Joseph I. Goldstein, Dale E. Newbury, Patrick Echlin, David C. Joy, A. D. Romig Jr, Charles E. Lyman, Charles Fiori, and Eric Lifshin, "Image Formation and Interpretation," in *Scanning Electron Microscopy and X-Ray Microanalysis*, pp. 149–271, Springer US, 1992.

[85] Pierre Raphael Bertrand, Mehdi Fhima, and Arnaud Guillin, "Off-Line Detection of Multiple Change Points by the Filtered Derivative with p-Value Method," *Sequential Analysis*, vol. 30, pp. 172–207, Apr. 2011.

[86] Mohamed Elmi, "Detection of Multiple Change Points by the Filtered Derivative and False Discovery Rate," *International Journal of Statistics and Probability*, vol. 3, Dec. 2013.

[87] Dan Cheng and Armin Schwartzman, "Multiple Testing of Local Extrema for Detection of Change Points," *arXiv:1504.06384 [math, stat]*, Apr. 2015. arXiv: 1504.06384.

[88] Pramod K. Vemulapalli, Vishal Monga, and Sean N. Brennan, "Robust extrema features for time-series data analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 1464–1479, June 2013.

[89] Shengfa Miao, *Structural health monitoring meets data mining.* PhD dissertation, Universiteit Leiden, Dec. 2014.

[90] Robert Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[91] M. Yamada, A. Saha, H. Ouyang, D. Yin, and Y. Chang, "N3lars: Minimum Redundancy Maximum Relevance Feature Selection for Large and High-dimensional Data," *arXiv:1411.2331 [cs, stat]*, Nov. 2014. arXiv: 1411.2331.

[92] Ingo Steinwart, "On the Influence of the Kernel on the Consistency of Support Vector Machines," *Journal of Machine Learning Research*, vol. 2, pp. 67–93, 2001.

[93] Ryota Tomioka and Masashi Sugiyama, "Dual-Augmented Lagrangian Method for Efficient Sparse Reconstruction," *IEEE Signal Processing Letters*, vol. 16, pp. 1067–1070, Dec. 2009.

[94] M. Morup, K. H. Madsen, and L. K. Hansen, "Approximate L0 constrained non-negative matrix and tensor factorization," in *IEEE International Symposium on Circuits and Systems 2008 (ISCAS 2008)*, (Seattle, WA), pp. 1328–1331, IEEE, May 2008.

[95] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf, "Measuring Statistical Dependence with Hilbert-Schmidt Norms," in *Algorithmic Learning Theory* (Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, eds.), no. 3734 in Lecture Notes in Computer Science, pp. 63–77, Springer Berlin Heidelberg, Oct. 2005.

[96] Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt, "Feature Selection via Dependence Maximization," *Journal of Machine Learning Research*, vol. 13, pp. 1393–1434, May 2012.

[97] Jianhua Lin, "Divergence measures based on the Shannon entropy," *IEEE Transactions on Information theory*, vol. 37, pp. 145–151, 1991.

[98] Elizabeth G. Ryan, *Contributions to Bayesian experimental design.* Thesis, Queensland University of Technology, 2014.

[99] Andreas Bulling, Ulf Blanke, and Bernt Schiele, "A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors," *ACM Computing Surveys*, vol. 46, pp. 33:1–33:33, Jan. 2014.

[100] André Gensler and Bernhard Sick, "Novel Criteria to Measure Performance of Time Series Segmentation Techniques," in *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM*, (Aachen, Germany), pp. 193–204, Sept. 2014.

[101] J. Takeuchi and K. Yamanishi, "A unifying framework for detecting outliers and change points from time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 482–492, Apr. 2006.

[102] "Least-Squares Density-Difference (LSDD)." `http://www.ms.k.u-tokyo.ac.jp/software.html#LSDD`. `http://www.ms.k.u-tokyo.ac.jp/software.html#LSDD`.

[103] "Change-Point Detection in Time-Series Data by Relative Density Ratio Estimation." `http://sugiyama-www.cs.titech.ac.jp/~song/change_detection/`.

[104] "Hilbert-Schmidt Independence Criterion Lasso (HSIC Lasso)." `http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/`.

[105] Zaïd Harchaoui, Eric Moulines, and Francis R. Bach, "Kernel Change-point Analysis," in *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 609–616, Curran Associates, Inc., 2009.

[106] "Physical Acitivity Monitoring for Aging People." `http://www.pamap.org/index.html`.

[107] Attila Reiss and Didier Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring," in *16th International Symposium on Wearable Computers (ISWC)*, (Newcastle), pp. 108–109, IEEE, June 2012.

[108] Attila Reiss and Didier Stricker, "Creating and Benchmarking a New Dataset for Physical Activity Monitoring," in *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments*, PETRA '12, (New York, NY, USA), pp. 40:1–40:8, ACM, 2012.

[109] A. Reiss and D. Stricker, “Towards Global Aerobic Activity Monitoring,” in *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*, PETRA ’11, (New York, NY, USA), pp. 12:1–12:8, ACM, 2011.